High Resolution Capabilities of Free-space Optical Neural Networks

Riad Ibadulla

Department of Computer Science
City, University of London

This dissertation is submitted for the degree of
Doctor of Philosophy (Ph.D) in Computer Science

December 2024

"As complexity rises, precise statements lose meaning, and meaningful statements lose precision." " — Lotfi Zadeh

Dedication

To my parents and sister, whose unwavering love, support, and encouragement have been my greatest strength throughout this journey; to my grandparents, whose support and values continue to guide me; and to my uncle, who is a constant source of motivation and inspiration.

This work is as much yours as it is mine.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text.

Riad Ibadulla
December 2024

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my first supervisor, Professor Thomas M. Chen, who sadly passed away during the final stage of my PhD. Professor Chen's guidance, encouragement, and expertise were invaluable throughout the initial stages of this research. He was enthusiastic about admitting me to the PhD program and about my project. His belief in my potential and his optimism have left an indelible mark on my academic journey. I am profoundly grateful for his mentorship and support.

I am also deeply thankful to my second supervisor, Dr. Constantino Carlos Reyes-Aldasoro, who stepped in as my primary supervisor during a challenging time. Dr. Reyes-Aldasoro has been as supportive and insightful as my first supervisor, always there when I needed him. His unwavering support, insightful feedback, and ability to seamlessly continue the guidance while offering new perspectives have been immensely appreciated.

I would also like to express my gratitude to Professor Eduardo Alonso for taking on the role of second supervisor and providing me with the feedback on my thesis.

I am grateful to other academics and staff members: Ernesto Jimenez Ruiz, Tillman Weyde, Rajkumar Roy, Veselin Rakocevic, Konstantin Pozdniakov, Ilir Gashi, Golnaz Badkobeh, Asif Nawaz, Ann Marie De Hare and others.

Furthermore, I extend my heartfelt regards to my peers: Ananda, Youssef Arafat, Daniel Brito, Kaleem Peeroo, Nadine El Naggar, Dany Laksono, Kevin Allain, Mirela Reljan-Delaney, Alex Clay, Daniel Sikar, Maeve Hutchinson, Robab Aghazadeh Chakherlou, Sevinj Teymurova, Saidu Sokoto, Naman Singh, Amir Hosseini and others.

Publications

Refereed Journal article

- **R. Ibadulla**, T. M. Chen, and C. C. Reyes-Aldasoro, “FatNet: High-Resolution Kernels for Classification Using Fully Convolutional Optical Neural Networks,” AI, vol. 4, no. 2, pp. 361–374, Apr. 2023, doi: 10.3390/ai4020018

Refereed Conference article

- **Riad Ibadulla**, Constantino C. Reyes-Aldasoro, and Thomas M. Chen "Fat-U-Net: non-contracting U-Net for free-space optical neural networks", Proc. SPIE 12903, AI and Optical Data Sciences V, 1290308 (13 March 2024); doi: 10.1117/12.3008618

Abstract

Deep Learning (DL) models are powerful tools for computer vision tasks, such as image classification and segmentation. To meet the computational demands of modern deep learning, many DL models rely on AI accelerators. In addition to these hardware-based accelerators, optical accelerators, such as 4f free-space systems, take advantage of Fourier optics to efficiently perform convolutions, bypassing Moore's law limitations. While 4f system offers high-resolution capabilities, it faces limitations in modulation speed and data readout.

This thesis addresses these limitations by developing methods to adapt traditional neural network architectures for high-resolution tasks within 4f free-space optical AI accelerators. We introduce FatNet, an algorithm specifically designed to convert conventional neural network models into a format optimised for the 4f system by accounting for the system's advantages and constraints. FatNet reduces the number of channels while increasing the resolution of feature maps, aligning with the high-resolution capabilities of the 4f system. Since a bottleneck in 4f optical accelerators lies in the readout process, FatNet enhances model efficiency by decreasing the number of channels. This conversion assumes that the number of trainable parameters and pixels in the feature maps remains equal or as close as possible to those in the original layers.

FatNet was applied to convert architectures such as ResNet, AlexNet, and VGGNet into Res-FatNet, Alex-FatNet, and VGG-FatNet, respectively. These models were trained and evaluated on the CIFAR-100 dataset using a custom-built simulator of the 4f system. Our results demonstrate significant acceleration with minimal loss in accuracy. Furthermore, the FatNet approach was scaled to the U-Net architecture, resulting in Fat-U-Net, which was tested on image segmentation tasks using the Oxford-IIIT Pet and HeLa cells datasets, showcasing its effectiveness in image segmentation within the free-space optical accelerator. The efficacy of FatNet was further examined in the Fat-U-Net study through experiments involving Intuitive-Fat-U-Nets, which prioritised layer weight equality over pixel count in feature maps to avoid overfitting, demonstrating that the FatNet conversion is the optimal approach. Additionally, the impact of skip connections in U-Net and Fat-U-Net was investigated to evaluate Fat-U-Net's ability to preserve localisation accuracy.

Moreover, this thesis explored the potential for implementing Vision Transformers (ViTs) within the 4f optical system. Methods are proposed for realising ViTs using only convolutional operations to enable full functionality on the 4f system, with a particular focus on investigating potential parallelism techniques suitable for optical settings. Additionally, the study included visualising attention maps to determine if the methods are training using feature extraction, similar to CNNs, or genuinely learning attention mechanisms as intended by ViTs.

This research also addressed challenges in optical computing, such as the lack of support for negative values, by introducing algorithmic solutions to mitigate the issue.

Overall, this work contributes to the advancement of optical neural networks, providing a pathway toward faster and more efficient deep learning models tailored for the emerging era of optical computing.

Contents

List of Figures

List of Tables

List Of Acronyms

ASIC - Application-Specific Integrated Circuit
CCD - Charge Coupled Device
CIFAR - Canadian Institute For Advanced Research
CMOS - Complementary Metal-Oxide-Semiconductor
CNN - Convolutional Neural Network
DMD - Digital Micromirror Device
ELU - Exponential Linear Unit
FFT - Fast Fourier Transform
GCN - Global Convolutional Network
ILSVRC - ImageNet Large Scale Visual Recognition Challenge
K-NN - K Nearest Neighbours
MZI - Mach-Zehnder Interferometer
ONN - Optical Neural Network
SGD - Stochastic Gradient Descent
SLM - Spatial Light Modulators
SVM - Support Vector Machines
TPU - Tensor Processing Unit
VIT - Vision Transformer
VGG - Visual Geometry Group
MLP - Multi Layer Perceptron
FCN - Fully Convolutional Networks
IoU - Intersection over Union
ROI - Region of Interest
D^2NN - Diffractive Deep Neural Networks
MNIST - Modified National Institute of Standards and Technology
MHSA - Multi-head Self attention
QKV - Querry (Q), Key (K), and Value (V) matrices from the Transformer models

Notations

Symbol	Description
	Fourier transform operator
X	Input of the model (e.g., feature map or image)
Y	Output of the model (e.g., predicted result or label)
r	Row index of the image or matrix
c	Column index of the image or matrix
	Angular frequency for Fourier transforms
$f\ t$	Function in the time domain
F	Fourier transform of $f\ t$
i	Imaginary unit $\sqrt{1}$
	Time shift variable for convolution
t	Time variable
u, v	Frequency domain variables (in 2D Fourier Transform)
W	Width of the image/matrix
H	Height of the image/matrix
I	Image (input for convolution or processing)
K	Kernel (convolution filter applied to the image)
	Convolution operation

Chapter 1

Introduction

Deep learning has revolutionised the field of artificial intelligence, enabling the development of sophisticated models for various applications. Among these, computer vision stands out as a particularly challenging area due to the large volume of data and the need for precise, high-speed processing. Within the deep learning approaches, convolutional neural networks (CNNs) have become a standard approach for various computer vision problems. CNNs have been successfully applied to image classification [3, 9, 2], object detection [10, 11], localisation [12], and segmentation [13, 14], among many other applications [15–19].

With the recent advancements of the transformer networks [20] in natural language processing, this approach has also migrated into computer vision [21] with the potential of replacing the standard of computer vision, CNNs. However, due to the computational efficiency of CNNs, traditional CNN networks still dominate the computer vision area.

CNNs are suitable for computer vision tasks because neurons in CNNs are only connected to the pixels of their receptive field rather than to every single neuron of the next layer as in fully connected networks. This localised connectivity reflects the fact that neighbouring pixels in images are more closely related to each other than distant pixels. This approach also reduces the number of trainable parameters, which accelerates the inference and makes the neural network more immune to overfitting. Although CNNs are computationally less expensive than fully connected neural networks, accelerating CNNs is still an important task, especially with the ever growing number of images and videos that are captured. As the complexity of these machine learning models grows, so does the computational demand and the challenge of real-time applications.

There have been many software methods to accelerate the deep learning training process, including using shallow networks [22], pruning [23], quantisation and network binarisation [24]. One of the ways would be to reduce the depth of the deep neural networks, as in the studies of Ba and Caruna [22], which proved that it is possible to estimate the deep

neural network with the shallower model and learn the same functions as deep networks on the example of the CIFAR-10 dataset. A similar accuracy can be achieved by having the same number of trainable parameters in the shallow network as in the original deep network. Pruning redundant weights in a previously trained network to reduce the size of the network for the inference acceleration can be considered another software-based method. In their experiments, Han *et al.* [23], trained the network first using the standard network training methods. Then, they pruned the small weights by deleting all weights below the threshold and retrained the network. According to [23], pruning reduces the number of parameters by thirteen times in VGG-16 and nine times in AlexNet.

While hardware accelerators, such as graphics processing units (GPUs), field-programmable gate arrays (FPGAs), and application-specific integrated circuits (ASICs), have emerged as a potential solution to this challenge, their effectiveness may be limited in the long run as Moore's Law begins to lose its predictive power [25].

In response to this challenge, researchers have begun to explore the potential of optics for accelerating deep learning [26]. By using the properties of light, optical devices can perform operations faster and more efficiently than traditional electronic devices, making them a promising option for deep learning applications.

This thesis explores the potential of free-space optics, specifically the 4f system, for accelerating deep learning and constructing neural network architectures accordingly. The 4f system is well-suited for high-resolution tasks as its performance does not degrade with increased input or kernel resolution, which is a significant limitation in conventional electronic architectures. The main performance bottleneck of high-speed cameras is the readout time, which, in fact, scales with the camera's resolution [7]. This thesis explores the adaptation of deep learning models, particularly CNNs and ViTs, to take full advantage of the 4f optical system. The FatNet conversion algorithm is at the core of this adaptation. It restructures traditional CNNs to match the capabilities of free-space optical computing better. By increasing the resolution of feature maps and kernels while reducing the number of channels, FatNet transforms the traditional cone-shaped architecture of CNNs into a barrel-shaped structure optimised for the high resolution offered by an optical system. Building on this foundation, this thesis introduces the implementation of FatNet on models like ResNet-18 [4], AlexNet [3], and VGG-19 [2] for classification evaluated on the CIFAR-100 [27] dataset, resulting in significant speedups in optical environments with minimal performance trade-offs.

The CIFAR-100 dataset is a widely used benchmark in image classification, containing 100 classes of real-life images, with each class representing a distinct object category, such as animals, vehicles, and everyday objects. Each class consists of 600 images, with 500 images

allocated for training and 100 for testing. The image resolutions are 32×32, which presents a significant challenge due to the low resolution combined with the high number of classes. This makes CIFAR-100 a much more complex and difficult version of the CIFAR-10 dataset, which only has 10 classes.

Building on the foundation of FatNet, this thesis introduces the Fat-U-Net architecture, a specialised adaptation of the U-Net [13] model for image segmentation tasks within the 4f system following the principles of the FatNet conversion. The effectiveness of Fat-U-Net is demonstrated through evaluations of benchmark datasets, such as the Oxford-IIIT Pet and HeLa cell segmentation tasks, showcasing its advantages over traditional GPU-based implementations.

The Oxford-IIIT Pet [28] is a well-known dataset for image segmentation. It contains images of cats and dogs with pixel-level annotations. Due to its variability in pose, scale, and breed, it is often used to benchmark segmentation models, providing a challenging test for model generalisation.

The HeLa cell datasets are widely used in biomedical image segmentation tasks [13], particularly for cell identification and analysis. HeLa cells, an immortal cell line derived from cervical cancer cells [29, 30], present unique challenges in segmentation due to their diverse shapes and sizes. These datasets are crucial for evaluating the performance of segmentation models in biomedical contexts and examining their ability to define cell boundaries [13, 31], nuclei [32] and elements like mitochondria [33] accurately.

This thesis then explores the potential implementation of Vision Transformers on the 4f free-space optical accelerators, with the idea that the same 4f system can be used for all kinds of neural networks. The analyses involve experiments on whether replacing the linear layers with the convolutional layers can make the model learn attention in a similar manner to regular multi-head self-attention. Several types of convolutions were tested, starting from regular convolutions and ending with the shared depthwise convolutional layers developed in this work to emulate regular MLP using convolutions. This study also investigates how our convolutional ViTs (ConvShareViTs) can be adapted to benefit from the high parallelism and resolution capabilities of optical systems.

1.1 Problem statement

Before diving into the problem statement, it is important to summarise the general specifications of the system under consideration. The observed system focuses on two fundamental tasks in computer vision: image classification using CNNs and ViTs, and image segmentation using CNNs. To accelerate the inference speed of these tasks, we explore the 4f free-space

optical system, a device capable of performing a convolution using optical components. The 4f system takes advantage of the principles of Fourier optics to compute convolutions in parallel across high-resolution input images, offering potential for accelerating deep learning models.

However, the 4f system introduces specific constraints that complicate its direct integration with traditional deep learning.

1. It can only perform convolutions without activation or pooling layers, requiring a hybrid optical-electronic approach to execute complete neural network architectures.

2. The system processes high-resolution data, necessitating architectural adaptations to utilise its parallelism effectively.

3. Finally, the readout process—extracting convolution results from the optical system—is a significant bottleneck, as it slows down the overall inference process, negating the potential speedup from optical computation.

Addressing these challenges requires rethinking how neural networks are designed and integrated with the 4f system. The goal is to create architectures that can operate efficiently in high-resolution settings, mitigate the bottlenecks caused by electronic components, and broaden the versatility of optical accelerators to handle diverse models. These considerations form the foundation for the core problems explored in this thesis.

Lack of research in barrel-shaped networks: Limited attention has been given to research and investigation of high-resolution training due to its inefficiency for CPU/GPU training. Traditionally, CNNs (Convolutional Neural Networks) are used for feature extraction, and the extracted features are then fit into a classifier, which can be any type of classifier. This structure exists due to older techniques where features used to be extracted using methods like SIFT (Scale-Invariant Feature Transform) [34] or manually and then fit into the classifier. However, with the advent of CNNs, it became possible to cascade the feature extractor and classifier into one model, allowing the convolutional feature extractor and linear layers (classifier) to be trained together.

Despite this advancement, the architecture of CNNs has retained its cone shape. This shape is a result of the pyramid-like feature extraction process in CNNs, where the feature maps are progressively downsampled, eventually forming a vector of encoded features that is input into the classifier. The early layers of CNNs naturally learn to extract basic features, such as horizontal and vertical lines, while the deeper layers extract more complex features based on the features from the previous layers. Consequently, the last layer of the convolutional part contains sufficiently complex features that can be effectively used by the classifier.

This cone-shaped CNN architecture is supported by historical context, particularly in the development of multi-scale representations such as the Laplacian pyramid introduced by Burt and Adelson [35]. Their work showed how using image encoding methods with multiple layers of image representation, which are similar in shape but different in size, can efficiently process both the spatial frequency and features of an image. Similar to the modern CNN architectures, this pyramid structure was designed to enhance efficiency by progressively downsizing the input data. This approach is still favoured in CNNs for its efficiency in CPU/GPU inferences. The downsizing operation significantly reduces the computation, making subsequent layers more efficient. It is important to note that the effective receptive field of these models increases with the depth of the layers [36]. The effective receptive field refers to the region of the input space that a particular CNN layer's output neuron is influenced by. Although larger kernels could also contribute to an increased effective receptive field in CNNs, they are not commonly used due to their inefficiency.

The optical setup discussed in this thesis, specifically the 4f free-space optical accelerator, is fully utilised only when the inputs and kernels are at maximum resolution. In this 4f system, the resolution of the kernels and inputs does not impact the speed of inference. Therefore, the exploration of high-resolution training becomes particularly relevant.

Constraints of the 4f system: Although the 4f system's advantages lie in its parallelism, high-resolution capabilities, passive Fourier transformation, and low energy consumption, the main limitation is the frequency rate of the modulators and cameras.

Convolutional neural networks contain a non-linear activation function after each convolutional layer. The 4f system can perform a set of convolution operations, and if the non-linearity could be applied optically, the output could potentially be fit into another optical convolutional layer. Unfortunately, non-linearity in optics is still an active area of research [37, 38]. Consequently, most research on 4f systems involves reading the output with a camera to introduce the non-linearity electronically [39]. This means that to perform all layers of the CNN optically, the output must be read by a camera, the non-linearity applied electronically, and then the input modulated again. This introduces a significant challenge, as the frequency rate of modulators and the read-out speed of the camera are the main bottlenecks of the system. The overall number of convolution operations in the model increases the inference time of the system. Therefore, the fewer convolution operations the system has, regardless of the resolution of the inputs, the faster the inference will be. This necessitates research on CNNs with fewer convolution operations and larger inputs/kernels without substantial loss in performance.

Limitations in the Versatility of Optical Accelerators Unfortunately, optical accelerators are usually designed with a specific neural network architecture in mind, meaning

they are specialised for a particular type of network, such as a CNN or a transformer model. Additionally, they are often customised for a specific architecture within that network type. For instance, some studies use fixed weights in optical CNNs and apply the optical setup only for the first layer or a single convolutional layer [1]. While some research explores the use of the same device across all layers of a CNN, optical setups for transformer models are typically designed specifically for those models [5]. This leads to the question: can a 4f system be flexible enough to work with all types of CNNs and transformer models? Developing a universal optical accelerator that can efficiently handle different types of neural networks is still an open research challenge, and the solution may lie more in software adaptation than in hardware changes.

Summary of the Problem Statement: The limitations of current deep learning architectures and optical accelerators highlight several key challenges. First, traditional cone-shaped CNN architectures, optimised for CPU/GPU efficiency, are not efficient for high-resolution training, which the 4f optical system inherently supports due to its resolution-independent inference speed. Second, the 4f system faces bottlenecks due to the reliance on electronic non-linear activation function and the frequency limitations of modulators and cameras, necessitating fewer convolution operations with larger inputs and kernels to optimise its performance. Lastly, optical accelerators lack versatility, often being tailored to specific architectures or layers, making the development of a universal optical accelerator for diverse neural networks an ongoing challenge that requires both hardware and software innovations.

1.2 Objectives

Building upon the problem statement, the objectives of this research are as follows:

1. Develop, test, and compare novel CNN architectures for classification and segmentation specifically designed and adapted to the 4f free-space systems, which provide high-resolution capabilities that can be exploited for parallelism. To achieve this objective, it is necessary to:

    - Consider that the main bottleneck of optical systems is the camera readout and thus, it is necessary to implement strategies to minimise camera readouts of the 4f system while maintaining performance.

2. Design and implement a simulator for the 4f system and optical convolutional layers. Use the simulator to benchmark performance of the traditional electronic implementations and compare them with the novel proposed architectures. To achieve this objective, it will be necessary to:

- Address and mitigate the optical accelerator's limitation in handling negative values, and enhance existing algorithmic solutions to improve memory efficiency.

3. Develop, test and compare novel Vision Transformer architectures that are compatible with the 4f system. To fulfil this objective, it will be necessary to:

   - Redesign multi-head self-attention mechanisms in Vision Transformers to use convolution operations exclusively. The novel self-attention mechanisms should be capable of learning attention scores rather than features, as in regular CNNs.
   - Evaluate the redesigned Vision Transformers for performance, computational efficiency and parallelisation potential in the 4f system compared to measure inference on GPU.

1.3 Scope and limitations

This research exclusively focuses on deep learning architectures using the 4f optical system introduced in related work. Whenever proof of concept is required to demonstrate that the architectures can run in optics, a simulation is used. It is not focused on the development of new optical setups or addressing hardware-related issues, though some minor coverage is provided. The study considers the advantages, limitations, and constraints of the 4f system from previous research to enhance the architectures and discover more efficient solutions.

The primary objective is to adapt standard neural networks, such as Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs), to formats that are more efficient within the 4f system. This work involves analysing these models to evaluate their benefits and drawbacks, with a particular focus on the improvements in inference speed and potential performance trade-offs.

Moreover, this research is solely focused on accelerating inference rather than backpropagation. Although the 4f system simulator has been used for training purposes in this thesis, the question of using the 4f system for backpropagation is beyond the scope of this work.

Additionally, several issues that exist in the free-space acceleration of AI are not considered in this research. These include the potential misalignment of optical elements, which could likely affect performance, system noise, various quantisation levels, and the impact of noise on low-precision training.

1.4 Contributions

1. **Introduction of the FatNet Conversion (Chapter 3):** Introduced the FatNet conversion, which optimises neural networks for optical computing by increasing the resolution of feature maps and kernels while reducing the number of channels. This conversion maintains the computational complexity and adapts the network to the constraints of the 4f system, significantly reducing the number of convolution operations required.

   (a) **Development of the FatSpitter Algorithm (Chapter 3):** Created the FatSpitter algorithm, which automatically converts any PyTorch model into its FatNet equivalent. This automated conversion process enables the transformation of existing neural network architectures into versions that are optimised for optical acceleration, facilitating broader application and experimentation. FatSpiter can potentially be integrated into the 4f system to convert all networks to a more optimised format before training if needed by the user.

   (b) **Design and Implementation of the FatNet Architectures (Chapters 3 and 4):** FatNet was designed and implemented on off-the-shelf architectures such as ResNet-18, AlexNet, and VGG-19. Following the successful implementation and evaluation of the CIFAR-100 dataset, the method was scaled to segmentation tasks. This led to the development of Fat-U-Net, the FatNet equivalent of U-Net, by applying the FatNet principles to the contracting path of the U-Net. Fat-U-Net features optimised kernel resolutions and reduced channel counts, making it suitable for high-resolution image segmentation tasks in the 4f optical system. The architecture underwent rigorous testing and evaluation on Pets segmentation and HeLa cells nucleus segmentation.

   (c) **Performance Evaluation of FatNets (Chapters 3 and 4):** Conducted comprehensive performance evaluations of classification FatNet networks and segmentation network Fat-U-Net compared to traditional U-Net on benchmark image segmentation tasks. Metrics such as accuracy, Intersection over Union (IoU), and Dice coefficient were used to assess the effectiveness and efficiency of FatNet, demonstrating its advantages in optical computing environments.

   (d) **Theoretical Analysis of Optical Acceleration Speedups (Chapters 3 and 4):** Provided a theoretical analysis and quantification of the potential speed-up gains achievable by FatNets when executed on free-space optical accelerator hardware. Comparisons were made with conventional electronic architectures to highlight

the performance benefits of optical acceleration, particularly in terms of reduced inference times and enhanced parallelism.

2. **Development of a Custom PyTorch Layer with 4f Optical Accelerator Simulation (Chapter 3):** Created a custom PyTorch layer that incorporates a built-in simulator of the 4f free-space optical accelerator. This simulator models the light propagation and diffraction processes, enabling the testing and validation of optical neural. It incorporates a new pseudo-negativity mechanism to address the inherent negativity problem in optical computing. This mechanism ensures the accurate simulation of optical convolutions that involve negative weights, which are not directly supported by optical systems. Unlike other methods in the related work, the method of this work stores only one version of the kernel instead of two.

3. **Development of Convolutional Methods for Vision Transformers (Chapter 5):** Innovated novel methods for running Vision Transformers using only convolution operations, making them compatible with the 4f system. This approach takes advantage of the parallel processing capabilities of optical computing, providing a scalable solution for Vision Transformer models. Provided the analyses of different methods of integrating convolution operation into the ViT.

4. **Analyses and Visualisations (Chapters 3, 4, and 5):** Conducted a visualisation of the first layer weights of the FatNets to compare them with the original layer. This was done to find out whether the FatNets are training similarly to the original models. The efficacy of the FatNets was proven by training Fat models that did not follow the FatNet principles (the Intuitive-Fat-U-Nets). Furthermore, it was trained without skip connections to demonstrate that the Fat-U-Net preserves localisation accuracy better when compared to the U-Net without skip connections (autoencoder). Similarly, the average attention scores were visualised and analysed to find out whether the ConvShareViTs learn attention like ViTs.

1.5 Structure of Thesis

Figure 1.1 shows the logical flow of chapters in this thesis.Chapter 2, the Literature Review, discusses the background, related work, datasets, and methodology relevant to the contributions of this study. Chapters 3, 4, 5 are the research chapters, which build upon the literature review and address the gaps in knowledge highlighted in that chapter. Chapter 3 details the development of the FatNet conversion, while Chapter 4 extends the methodologies presented

in Chapter 3 to adapt them for segmentation tasks. Chapter 5 uses methods distinct from the previous two chapters. Chapter 6 summarises conclusion and future work.

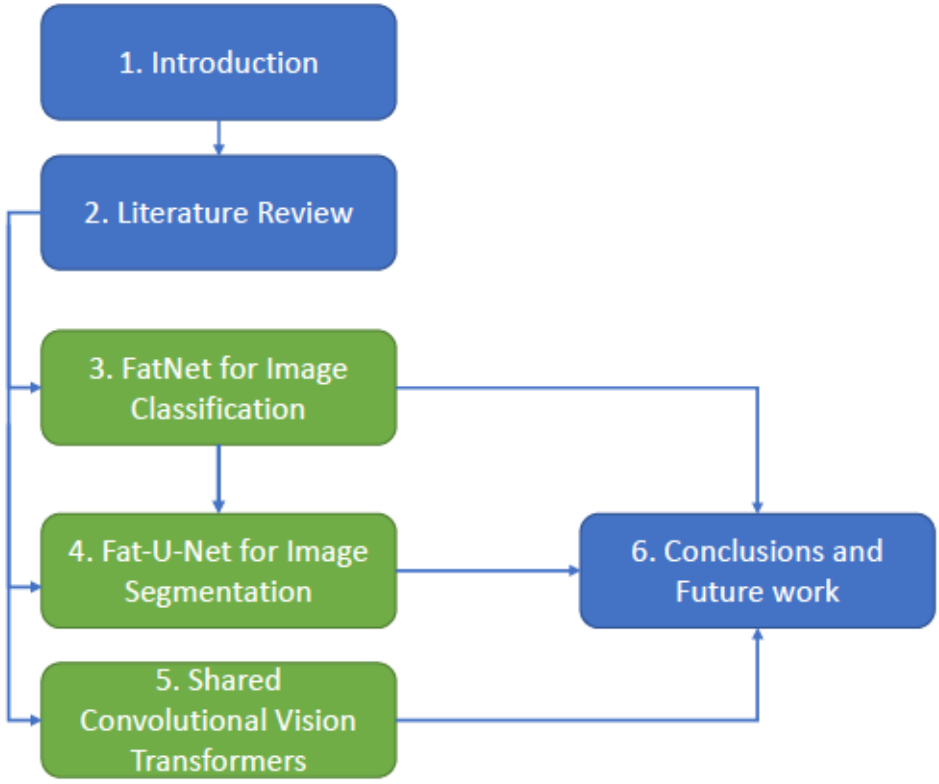


Fig. 1.1 **Structural flow of the Thesis.** This flow chart illustrates the logical progression and dependencies between the chapters of the thesis. The arrows indicate the flow of content, showing how each chapter builds upon the previous ones.

The detailed descriptions of all chapters are listed below:

Chapter 2: Literature Review -This chapter combines the background and related work to provide a comprehensive understanding of the field covered in this thesis. It introduces the foundational concepts and methods, including Optical Computing, the 4f system, Convolution operations, Convolutional Neural Networks (CNNs), Transformer networks, and the Attention mechanism. These methods, which form the basis of this research, are common knowledge in the field.

The chapter also reviews related work relevant to the contribution of this thesis, discussing the application of CNNs in classification and segmentation tasks. The datasets used in this research are described. Specifically, the use of the CIFAR-100 dataset for image segmentation, as well as the HeLa cell and Oxford Pets datasets for image segmentation tasks, are reviewed in detail, with benchmarks provided for comparison purposes. Additionally, the application of Transformers in image classification, particularly Vision Transformers, is explored.

The chapter further examines the impact of high-resolution training in CNNs and the implementation of high-resolution or parallel inference in optical accelerators. Lastly, it investigates the acceleration of deep learning within hardware, focusing on optical processors, with an emphasis on free-space optics and the 4f system.

Chapter 3: FatNet for Image Classification This chapter details the primary method, FatNet conversion, and its automated algorithm structure. It also explains the optical simulator developed to evaluate model performance, the pseudo-negativity method used in the simulator. The chapter then covers the conversion and evaluation of three off-the-shelf networks (ResNet-18, AlexNet, and VGG-19) into their FatNet equivalents, trained with the CIFAR-100 dataset.

Chapter 4: Fat-U-Net for Image Segmentation This chapter discusses the conversion and evaluation of a U-Net type architecture into its FatNet equivalent (Fat-U-Net), using the HeLa cancer cells dataset and the Oxford-IIIT Pet dataset. It also analyses the U-Net and Fat-U-Net without skip connections and further evaluates the effectiveness of the FatNet conversion using "Intuitive Fat-U-Nets".

Chapter 5: Shared Convolutional Vision Transformers (ConvShareViT) This chapter explores using the 4f optical accelerator for Vision Transformers (ViTs). Although optical setups for transformers exist, they are typically task-specific. We investigated using the same 4f system for ViTs by implementing ViTs with convolution operations. Different methods were analysed, and the most effective was a shared depthwise convolution developed by us. This method emulates linear layers using convolutions, making it suitable for the 4f system.

Chapter 2

Literature Review

Overview

This chapter introduces optical systems and neural networks, focusing on 4f optical accelerators. It explains optical computing, the 4f system, the Fourier Transform's role in computer vision, and its connection to convolution operations. CNNs and their applications in image classification and segmentation are discussed, along with a review of Transformers, attention mechanisms, and ViTs.

The chapter also examines high-resolution training, free-space optical accelerators, their benchmarks, and the 4f system's capabilities, including parallelism and experimental results. It concludes with the application of free-space optical accelerators to transformers and details on datasets and benchmarks used in this work.

2.1 Optical Computing and 4f system

Optical computing refers to information processing systems that use light or photons to perform computation, unlike electronics, which use electrons [40]. It started in the 1960s when lasers were invented and found to have coherent optical properties suitable for information transmission and processing. Optical computing exploits the high bandwidth and low interference properties of photons to process data in parallel at the speed of light.

Some key advantages of optical computing include high bandwidth of the light beam, high speed, zero resistance, lower energy consumption, and immunity to overheating [41]. However, optical computing also has some disadvantages:

- Lack of optical memory makes storage difficult

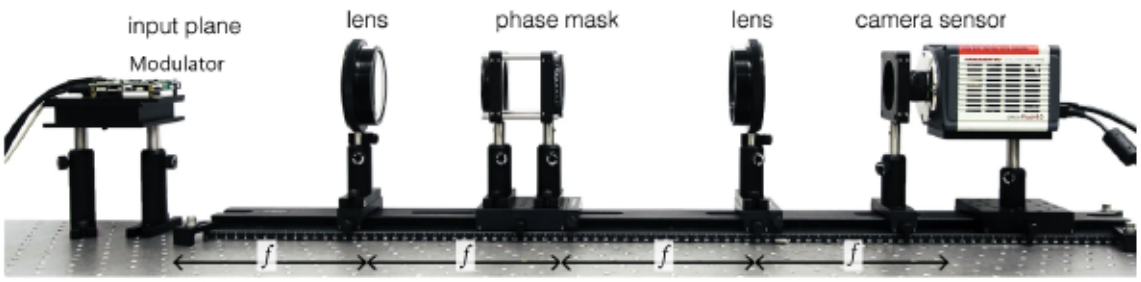


Fig. 2.1 **4f System hardware setup as per Chang *et al.* [1].** In this particular configuration, laser input, not shown in the figure, shines its beam onto the input plane modulator. The beam then passes through a lens, performing a Fourier transform of the image onto a phase mask in the Fourier plane. The phase mask applies a predefined filter by modulating the Fourier-transformed image. This fixed-phase mask can be replaced with a dynamic modulator if required. The fixed phase mask used by the authors in the Fourier plane can be replaced by another modulator if needed. The final wavefront, captured by the camera, represents the convolution of the input image with the phase mask kernel.

- Difficulty of performing non-linear function in optics
- Impossible to represent negative values in 4f correlator (4f free-space optical system used in this work to perform convolution and correlation operations) as the camera reads out the intensity
- Slow optics-electronics and electronics-optics conversion
- Flipped output: The input of the system is vertically flipped at the output due to the physics of the geometrical optics in the 4f free-space correlator.

Since this work focuses on optical neural networks, it is essential to mention that there are two main approaches to optical neural networks: free-space using Spatial Light Modulators (SLM) [7, 1] or silicon photonics approach using Mach–Zehnder Interferometers (MZI) [42, 43]. Unlike silicon photonics, free-space optics use wireless propagation through a medium, which can be air, outer space or vacuum. Although the silicon photonics approach's clock speed can reach several GHz, making it faster than free-space optics, it is inferior to the free-space system in parallelism [26]. This research is focused on the 4f free-space approach as described in Li *et al.* [7], which takes advantage of the parallelism of free-space optics. The 4f free-space optical system can be used to perform convolution operations faster than traditional electronic processors.

Weaver and Goodman introduced the optical convolution method using the 4f system in 1966 [44]. Although this research urged interest in optical neural networks, such as the work

by Jutamulia and Yu [45] in 1996, the idea of optical neural networks only began gaining broader popularity in the 21st century with the rise of neural networks and deep learning.

A standard 4f optical system consists of an input source, two convex lenses, two light modulators and a sensor (see Figure 2.1). The input source is the laser emitting the light modulated right in the beginning with the input image by altering the light intensity.

All these elements can have different parameters, and in the previous research on the 4f system, different parameters and elements have been used to build the correlator. These different setups and their performance is discussed in Section 2.8 of the next chapter.

Laser source: The laser source is a semiconductor device similar to the LED that emits a laser beam when current is supplied. In the previous research, different researchers used different wavelengths. For example, in their experiments, Chang *et al.*[1] used the green laser with a wavelength of 532nm, while Wu *et al.*[46] used a red laser with a wavelength of 632.8nm.

Modulators: Modulators are the core components of the system, responsible for adjusting both the amplitude and phase as required. Usually, the light emitted from the laser source is projected onto the modulator to regulate its amplitude at the source. Numerous modulator types exist, with preferences varying among 4f system researchers. The most common modulators used in 4f systems are the Spatial Light Modulators (SLMs) and Digital Micromirror Devices (DMDs). SLMs can regulate the signal's amplitude and phase by manipulating the cell's refractive index [7]. SLMs usually contain a liquid crystal panel, similar in some ways to the displays used in TVs and monitors. By controlling the orientation of the liquid crystal particles, it can change the phase or amplitude of the light passing through each pixel. Specifically designed for lasers with a certain wavelength, SLMs vary in their resolutions, with some offering as high as 4K [47]. DMDs are commonly used in projectors and are based on the Digital Light Processing (DLP) technology of Texas Instruments. DMDs contain hundreds of thousands or even millions of tiny mirrors, each corresponding to a pixel. Each of these mirrors can tilt either towards an 'on' position, reflecting light through a lens and onto a screen or an 'off' position, where the light is directed elsewhere and not displayed, achieving an amplitude modulation [48]. Comprised of a head and a controller, the modulator setup allows for user-friendly adjustment via a DVI connection of the controller to a PC, facilitating the customisation of modulator settings.

Lenses: Lenses are the optical components that manipulate the light path by refraction, allowing for the focusing or dispersion of light beams. They are commonly made of transparent materials like glass or plastic. Lenses can come in various forms, the most common being converging or convex lenses, which focus light to a point, and diverging or concave lenses, which spread light out. The 4f system uses convex converging lenses, which concentrate

parallel rays of light at the focal point. The reason why we use the convex lens is due to the Fourier properties of the convex lenses. In a 4f system, a pair of lenses is typically used to transform the spatial distribution of light in the input plane to its Fourier transform in the focal plane [49] and then back to a spatial domain in the output plane.

Camera: The camera is used at the final stage of the system to read out the results. This is one of the crucial elements of the system, as the camera and the modulators play a major role in keeping the system's high frame rate and resolution. Different types of cameras exist, with varying sensor technologies, resolutions, and capabilities. CCD (Charge Coupled Device) and CMOS (Complementary Metal-Oxide-Semiconductor) cameras are common in scientific applications, each having their strengths and limitations.

The general procedure of performing a Fourier transform and an Inverse Fourier transform using an optical system is outlined as follows. Light is directed onto the modulator, where it gets modulated and then passed through the first convex lens after covering the lens's focal distance. The light is then projected onto the focal plane, where the Fourier transform of the input is formed. The light subsequently passes through the second lens, executing an inverse Fourier transform, and is recorded by the camera. The captured output image is a flipped version of the input resulting from the principles of geometrical optics. The key concept here is the rapid attainment of an image's Fourier transform and its reconversion to the spatial domain at the speed of light without any energy loss.

2.2 Fourier transform

The *Fourier transform* is a well-known mathematical technique that decomposes the signal into the fundamental sinusoids in the frequency domain that, when combined, form the initial function [50]. By applying the Fourier transform, we obtain the frequency coefficients that describe the signal. A key advantage of this method is its ability to switch between time (or spatial) and frequency representations. Additionally, it converts the convolution operation into an elementwise multiplication. The Fourier Transform is defined as:

$$F(\omega) = \mathscr{F} f(t) = \int f(t) e^{-i\omega t} dt \tag{2.1}$$

where $f(t)$ is an input signal in the time domain, ω represents the frequency variable in the frequency domain, and t represents the time variable in the time domain.

The discrete Fourier transform (DFT) is a mathematical method for converting a sequence of discrete values into components of different frequencies [51]. It allows Fourier transforms

to be calculated for discrete sequences using the direct formula for DFT. The discrete Fourier transform for a sequence $x_0,....,x_{N-1}$ is defined as:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-2\pi ikn/N}, \quad k = 0,...,N-1 \tag{2.2}$$

A Fourier transform is initially defined over one dimension and can be extended to two or more dimensions for the analysis of two-dimensional signals and images [52]. Two-dimensional DFT operates on the two spatial dimensions x and y to convert the 2D signal into the frequency domain with frequency components f_x and f_y. The output is a complex-valued function of these spatial frequencies. The 2D DFT is defined as:

$$F(u,v) = \sum_{r=0}^{H-1} \sum_{c=0}^{W-1} f(r,c) e^{-2\pi i(ur/H + vc/W)} \tag{2.3}$$

where $F(u,v)$ is the 2D Fourier transform and (u,v) are discrete frequency coordinates, and (r,c) are the spatial coordinates in the input.

The computational complexity of this process increases with the dimensions of the data. The direct computation of the DFT has a complexity of $O(n^2)$ for a 1D sequence where n is the number of input values. However, fast algorithms like the fast Fourier transform (FFT) [51] significantly reduce this computational load, bringing the complexity down to $O(n\log(n))$. For 2D FFT the complexity becomes $O(n^2 log(n))$, where n^2 is the number of pixels of an image [53]. On the other hand, performing a 2D Fourier transform in free-space optics can be easily achieved by passing the light through the convex lens as mentioned in Section 2.1, where the light has to travel only two focal distances (f) from the lens [49].

2.3 Convolution operation and CNNs

2.3.1 Convolution operation

The convolution operation is the mathematical operation which is frequently used in signal and image processing systems [54]. Convolution operation expresses the degree to which one function is modified by overlapping another function. It is particularly used in image and signal processing to apply filters for tasks like edge detection [55], blurring [56], and sharpening [57] etc. Convolution operation involves two signals, an input signal and the filter. It is often used in deep learning algorithms, especially convolutional neural networks (CNNs).

The 1-dimensional convolution operation of two signals f and g is defined as:

$$(f*g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{2.4}$$

For discrete-time signals, the convolution is represented as:

$$(f*g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m] \tag{2.5}$$

Similarly to Fourier transform, the convolution can be extended into the two dimensions for the convolution on the images. The convolution of the image I and filter K can be represented as:

$$(I*K)[r,c] = \sum_{m}\sum_{n} I[m,n]K[r-m,c-n] \tag{2.6}$$

The convolution operation is closely related to the Fourier transform. Convolution in the time domain is equivalent to multiplication in the frequency domain. Using the Fourier transform, convolution can be performed through element-wise multiplication in the frequency domain. Using the Fourier transform $\mathscr{F}(I)$ and $\mathscr{F}(K)$, convolution can be computed as:

$$I*K = \mathscr{F}^{-1}\{\mathscr{F}(I)\cdot\mathscr{F}(K)\} \tag{2.7}$$

where $\mathscr{F}^{-1}$ is the inverse Fourier transform. This is known as the convolution theorem.

2.3.2 Convolutional neural networks

Convolutional Neural Networks (CNNs) [9] are a class of deep neural networks initially designed for computer vision tasks. They have revolutionised the field of computer vision [58], enabling computers to perform tasks such as image classification [59], image segmentation [13] and object detection [10] with remarkable accuracy. The fundamental principle behind CNNs is their hierarchical training process and their ability of feature extraction.

Previously, feature extraction was performed manually or through traditional computer vision techniques before inputting into linear models for classification. CNNs, however, unify the processes of feature extraction and classification within a single model, enabling end-to-end learning.

CNNs work by automatically learning spatial hierarchies of features from input images. This is achieved through the use of multiple layers that process the input in a hierarchical manner. The key concept of CNNs is to stack convolutional layers, activation functions

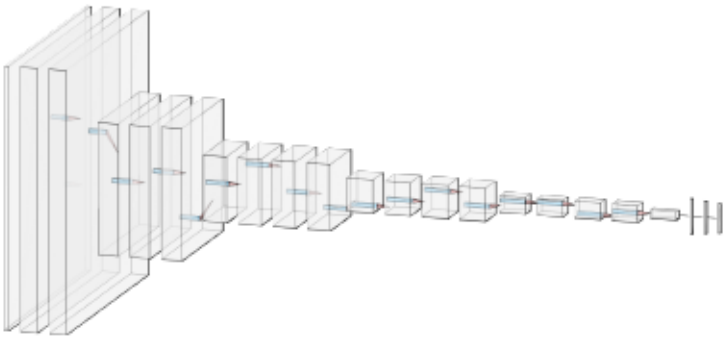


Fig. 2.2 **Example of the convolutional neural network for classification, VGG-19 [2].** A cone-shaped network is where the feature maps are pooled down, and the final one is fattened into a vector and fit into a classifier of linear layers.

for non-linear linearity and pooling operations to learn and extract the features. Extracted features can then fit into the linear layers if the goal is the classification.

Regular convolutional neural networks designed for the classification are cone-shaped, as shown in Figure 2.2. The input of the convolutional neural network is usually an image with one or more channels. Multichannel images usually contain three channels, each corresponding to red, green, and blue colour filters. By means of the convolutional layer, that image is mapped into another tensor (Multidimensional Array) with more channels. At the same time, the pooling operation reduces the resolution of the tensor, and the activation function introduces a non-linearity into it. This process is repeated, consequently the resolution of the tensors reduced, and number of channels increased throughout the network. Ideally, the convolutional part of the network ends with the vector, which contains all the information regarding the features in the original image.

The fundamental layer of the CNN is the 2D convolutional layer, usually noted as Conv2d in deep learning frameworks. These layers perform the convolution operation by filtering the input image with the filter (kernel) to generate a feature map. The kernel is not flipped, unlike the original convolution operation described in the previous subsection. Hence, it would be correct to say that CNNs perform cross-correlation, but it's still referred to as the convolution in machine learning and denoted as follows:

$$F(r,c) = \sum_{m=0}^{h-1} \sum_{n=0}^{w-1} I(r+m, c+n) \cdot K(m,n), \tag{2.8}$$

where $F(r,c)$ is the value of the output feature map at position (r,c), $I(r+m,c+n)$ is the region of the input image at the position that corresponds to the current kernel position, $K(m,n)$ is the region of the kernel at position (m,n), h and w are the height and width of the kernel.

Apart from the standard convolution operation, padding and strides can be used in the convolutional layer.

Stride is the number of pixels by which we slide the kernel over the input image. A stride of more than 1 reduces the size of the output feature map.

Padding involves adding extra pixels around the input image. This is usually done to allow the kernel to fit perfectly over the image's borders and control the spatial dimensions of the output feature map. Usually, padding is done with zeros. In this work, we have mostly used "same" padding, which means we apply so much padding that the output of the convolution keeps the same resolution as the input.

The output dimensions of the convolution of input $H \times W$ with the kernels of size $h \times w$ with padding p and stride s are calculated using the following:

$$\text{Output height} = \left\lfloor \frac{H-h+2p}{s} + 1 \right\rfloor \quad \text{Output width} = \left\lfloor \frac{W-w+2p}{s} + 1 \right\rfloor \tag{2.9}$$

The figure shows that, in practice, both the input and output involved with 2D convolutional layers are not 2D matrices as initially simplified in Equation 2.8, but rather 3D tensors. This insight leads us to a more precise mathematical formulation of a convolutional layer:

$$F(r,c) = \sum_{m=0}^{h-1} \sum_{n=0}^{w-1} \sum_{l=0}^{ch-1} I(r+m, c+n, l) \cdot K(m,n,d), \tag{2.10}$$

where $F(r,c)$ is the output of the convolution operation at coordinates r,c, I represents the input image or feature map, and K is the convolution kernel. The variables m and n are the coordinates within the kernel, ranging from 0 to $h-1$ and $w-1$, where h and w are the height and width of the kernel. The index l represents the input channel, and d represents the output channel of the convolution.

In this equation, the three-dimensional tensor's kernel slides over the input tensor, which is also three-dimensional, performing a dot product at each position. The output from this operation is a 2D matrix, indicating that each channel in the output tensor corresponds to a separate three-dimensional kernel. This means that each output channel is generated by a

distinct 3D kernel, which combines information from all the channels in the input tensor to extract specific features.

An activation function is typically applied after the convolution operation, and pooling is optional, especially when valid padding is used. Valid padding naturally reduces the output resolution, potentially making pooling less necessary. However, with "same" padding, where zeros are added around the edges to maintain the input size, pooling is commonly employed.

The main purpose of pooling is to reduce the resolution of the feature map. There are various methods for pooling, but they all serve this core purpose. By downsizing the feature map, pooling helps reduce the computational load for subsequent layers, reduce overfitting, maintain spatial hierarchy, and increase the speed of computation and memory usage in the subsequent layers.

In CNNs, the most common types of pooling operations are 2D maximum pooling (MaxPool2d) and 2D average pooling (AveragePool2d), which simply either take the maximum value in the region of the kernel and replace the region with that pixel only or use the average of it.

When it comes to optimisation and training, the values of the kernels in the convolutional layers act as weights for the model. By adjusting these kernels, the model can effectively learn complex tasks like classification and find the optimal kernel to minimise loss. The underlying hypothesis of CNNs is that images can be effectively represented and analysed through the hierarchical extraction of features, where lower-level features (like edges and corners) are used to progressively build up higher-level representations (such as objects).

2.3.3 Batch Normalisation

Batch normalisation [60] standardises the inputs to a layer for each mini-batch. Hence, for each mini-batch, the layers' inputs are adjusted so that they have a mean of zero and a standard deviation of one. The layer keeps track of the mini-batch's running mean and variance, which is used during the evaluation. Batch normalisation usually involves two main parameters per feature: the mean and variance, and if affine parameters are used, scale (γ) and shift (β) the normalised value. Affine parameters are learned during the training process.

For the full operation of Batch Normalisation applied to a 2D feature map (as in BatchNorm2D), considering a single feature channel for simplicity, the formula can be expressed in a more consolidated form as follows:

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \varepsilon}} \gamma + \beta, \tag{2.11}$$

where y is the output of batch normalisation, x is the input feature to be normalised, $\mathrm{E}[x]$ is the expectation (mean) of x, calculated over the mini-batch, as well as spatial dimensions for 2D feature maps in the case of images, $\mathrm{Var}[x]$ is the variance of x, calculated over the same dimensions as the mean, γ and β are parameters learned during training for each feature channel, ε is a small constant to prevent division by zero.

Batch Normalisation can be used as an additional regularisation technique. It will improve the speed of training and reduce the number of epochs required to train the same network.

2.4 Application of CNNs for image classification

The use of convolutional layers in deep learning dates back to 1980 when Kunihiko Fukushima developed the Neocognitron architecture, which utilised downsampling for feature extraction [61]. However, this architecture was not designed for backpropagation [62], which was later successfully implemented by LeCun *et al.* [9] in 1998 with the LeNet architecture. LeNet consists of two convolutional layers, each followed by a pooling operation and flattened output of these layers, then fit into a classifier with three fully connected layers. The activation function used after each computational layer (convolutional or linear) is a sigmoid layer.

LeNet achieved a test error of 0.95% within 10 epochs on the MNIST dataset. Although the test error stabilised at 0.95%, the error on training data continued to decrease, reaching 0.35% after nineteen epochs. The phenomenon, which LeCun *et al.* referred to as "overtraining", is now commonly known as overfitting. The issue was slightly mitigated when the images were augmented with random distortions, leading to a further decrease in test error to 0.8%.

Today, most CNNs adhere to a design similar to that of LeNet, employing convolutional layers together with downsampling to extract features. Consequently, it ends up with a feature vector, which fits into a classifier, a fully connected linear classifier.

Another classical network influenced by the principles of LeNet is AlexNet, developed by Krizhevsky et al. [27]. AlexNet consists of five convolutional layers, followed by three linear layers for classification, as shown in Figure 2.3.

Pooling operations are applied only after the first two and the last convolutional layers. Notably, AlexNet was the first network to employ the ReLU activation function, effectively mitigating the vanishing gradient problem. Furthermore, AlexNet introduced dropout regularisation with the probability of 50% in the activations of the linear layers within the classifier to improve generalisation.

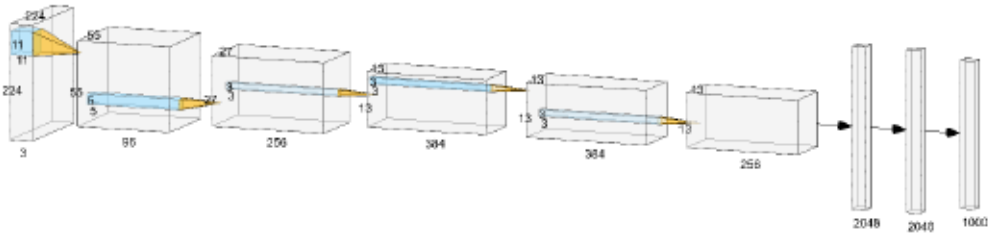


Fig. 2.3 **The original architecture of AlexNet, proposed by Krizhevksy *et al.* [3].** The model designed for the ImageNet classification is the network that uses the ReLU activation function, consisting of five convolutional layers and three dense layers.

AlexNet achieved a 37.5% top-1 error rate on the ILSVRC-2010 test set and a 15.3% top-5 error rate on the ILSVRC-2012 test set, pre-trained on ImageNet 2011. Although AlexNet is a classic convolutional neural network—which is why we used it in this research-it has higher error rates compared to deeper and more stable modern CNNs.

Building upon the groundbreaking architecture of AlexNet, the Visual Geometry Group (VGG) from the University of Oxford developed VGG nets [2] in 2014, marking another significant advancement in the field of deep learning. The VGG architecture is known for its emphasis on increasing the depth of the network using an architecture with a small 3×3 convolution filters, which proved to be a key factor in improving the model's performance. VGG used max-pooling with the kernel of 2×2 and stride equal to the kernel size (2), to downsize the feature maps. In VGG, the number of channels doubles after each max-pooling layer, which reduces the spatial dimensions of the feature maps. This expansion from 64 channels in the first layer to 512 channels in the deeper layers allows the network to capture complex features at each level.

Moreover, VGG consists of several convolutional blocks, where each block contains a sequence of convolutional layers followed by a max-pooling layer for spatial reduction. This block structure enables the network to learn hierarchies of features with each block building upon the refined outputs of the previous one. Similarly to LeNet and AlexNet, VGG ends with the fully connected layers that learn to classify the features extracted by the convolutional blocks.

There are six VGG networks introduced, known as A (11 weight layers), A-LRN (Same as A, with Local Response Normalisation), B (13 weight layers), C (16 weight layers), D (16 weight layers) , E (19 weight layers). The largest network E, also known as VGG-19, achieved the top-1 validation error of 25.5% on e ILSVRC-2012.

Following the exploration of VGG Networks in the development of deep convolutional neural networks by He *et al.* [4], the introduction of Residual Networks (ResNets) marked a significant advancement in the field. ResNets introduced a new type of residential connection

Table 2.1 Performance of residual networks on ImageNet [4]

Model	Top-1 error (%)	Top 5 error (%)
ResNet-18	27.88	Not provided
ResNet-34	25.03	7.76
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	21.43	5.71

into the networks, which helped to address the vanishing gradient problem associated with training very deep networks.

Formally, He *et al.* [4] has noted the blocks of the ResNet as:

$$y = F(x, W_i) + x, \tag{2.12}$$

where x is the original input, y is the output of the residual block, and the $F(x, W_i)$ represents the building block of the residual layer, which contains several weight layers.

The distinct characteristic of ResNet is its capability to learn the identity function – a mapping of inputs directly to their outputs – and to build more complex classifications on top of it. This is realised via these shortcut connections that perform identity mapping, with extra layers explicitly learning the residual functions. When $F(x, W_i) = 0$, it means that the network has learned to choose an identity function for that layer, making sure that even the deepest networks can easily be trained.

He *et al.*has introduced five residual networks, with 34, 50, 101, and 152 layers. Smaller ResNet architectures like ResNet-18 and ResNet-34 are built based on basic blocks that consist of two convolutional blocks with a residual connection around both blocks. Each convolutional block consists of a convolutional layer with the 3×3 kernel followed by the Batch Normalisation and ReLU activation layer. It should be noted that according to He *et al.*'s [4] implementation, the second ReLU activation function is applied after the identity summation as shown in Figure 2.4. Larger residual models, on the other hand, contain three convolutional layers in each block, where the first and the last layers are 1×1 convolutions, and the middle layer is 3×3 convolution.

The authors have tested the method with ImageNet and CIFAR-10 datasets. With the lowest error achieved by ResNet152 at 21.43% and the highest error achieved by ResNet-18 at 27.88%.

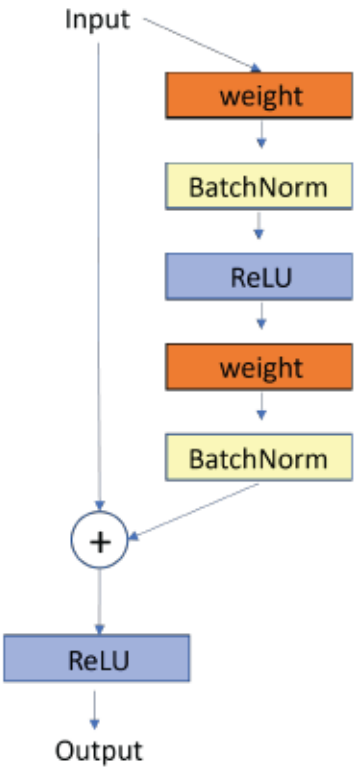


Fig. 2.4 **ResNet Block as described in the original introduction by He *et al.* [4].** The residual layer or block contains two weight layers, in this case, convolutional layers. The input of the block is saved and then added to the output of the batch normalisation of the second convolutional layer right before its activation layer.

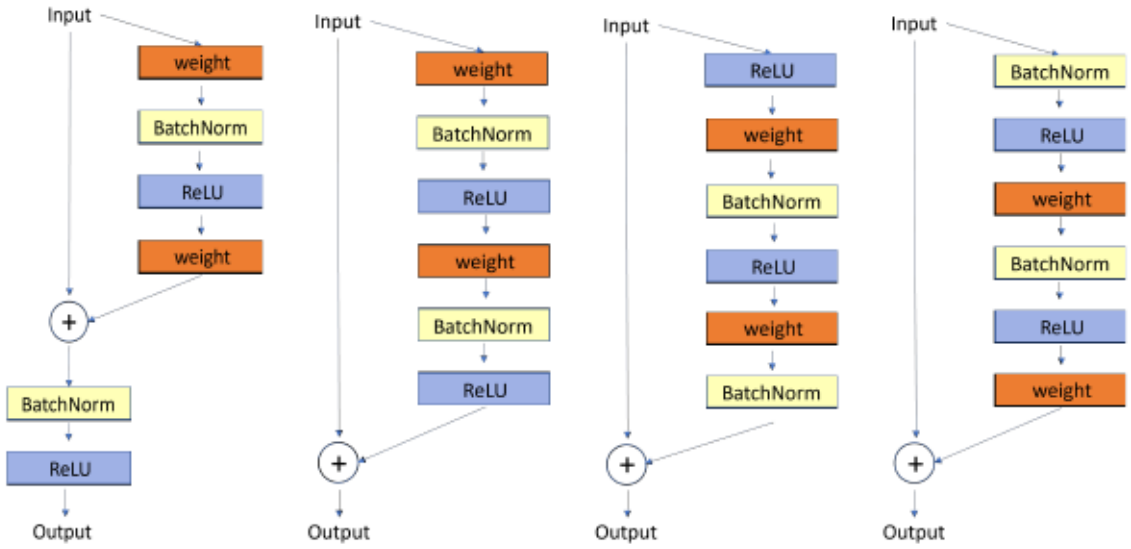


Fig. 2.5 **Other orders of layers in the residual block.** (a) BatchNorm after addition (b) ReLU before addition (c) ReLU preactivation (d)full-preactivaiton

In another work by He *et al.* [63], the authors analysed different usages of activation, as shown in Figure 2.5. The authors tested various configurations of the ResNet architectures, comparing standard activation to preactivation methods. The authors observed that the model with full preactivation, where both the batch normalisation and ReLU activation are applied before the convolutional layer, outperformed all other variations on CIFAR-100. This finding highlights the importance of the order of operations in the network architecture.

2.5 Application of CNNs for image segmentation

This section reviews a variety of learning-based segmentation methods, prioritised based on their relevance to the current work. Many of these methods share common architectural features, including encoders and decoders, skip-connections, multiscale architectures, and the recent adoption of dilated convolutions.

The backbone of image segmentation with deep learning methods can be considered Long *et al.*'s [64] Fully Convolutional Networks (FCNs). FCNs are uniquely comprised of convolutional layers, allowing them to produce a segmentation map that matches the size of the input image. The authors adapt off-the-shelf CNN architectures like VGG16 and GoogLeNet by eliminating all fully connected layers. This modification enables the model to

generate spatial segmentation maps instead of mere classification scores. FCNs incorporate skip connections, where upsampled feature maps from the network's deeper, more abstract layers are combined with the more detailed feature maps from earlier layers to mitigate localisation loss. All CNN-based architectures after FCNs follow a similar structure. The authors used PASCAL VOC2011 for the evaluation and achieved 62.7% mean IoU in their best configuration.

Another model for image segmentation is known as DeConvNet, introduced by Noh *et al.* [65]. According to the authors, FCNs face several significant issues. For example, these networks operate with a fixed-size receptive field. As a result, objects that are much larger or smaller than this receptive field are segmented incorrectly. Unlike FCNs, DeConvNet has a symmetrical architecture, where the first half consists of convolutional and pooling layers, known as the encoder, and the second half involves deconvolution and unpooling layers, known as the decoder. The symmetry is an integral part of the architecture, as the decoder explicitly mirrors the encoder.

Ronneberg *et al.* [13] proposed U-Net a convolutional neural network (CNN) architecture designed for the segmentation of biomedical images, but now is also being used outside the medical domain. Its distinctive architecture features contracting and expanding paths, enabling it to effectively capture both local and contextual information, which results in remarkable segmentation performance. It resembles the architecture of the DeConvNet, but the main difference is that the U-Net uses the deconvolution layers for the upsampling instead of unpooling. The contracting path functions similarly to a conventional CNN used for classification, comprising blocks of convolutional layers, activation functions, and pooling operations for multi-scale feature extraction. The expanding path of U-Net is responsible for upsampling the extracted features to reconstruct the segmentation mask of the input image. This process uses transposed convolution operations to upsample the feature maps and involves concatenation with the corresponding feature maps of the same resolution from the contracting path, known as skip connections. The primary function of the skip connections is to preserve the spatial information lost during the pooling process in the contracting path.

According to Peng *et al.* [66], the image segmentation method involves dividing an image into meaningful segments by classifying each pixel by effectively combining two distinct tasks: classification and localisation. However, these tasks often conflict, as enhancements in one can diminish the other. Classification models, which are generally designed to be insensitive to shifts in position or rotation, incorporate pooling operations to capture features at various scales. As a result, in the deeper layers of such networks, where smaller 3×3 kernels are used, the ratio between the kernel size and the feature map resolution is greater

than in the upper layers. This configuration allows the deeper layers to interact with a broader area of the original image, leading to the typical pyramid or cone shape of classification networks. Conversely, an ideal segmentation model would adopt a barrel shape to more accurately pinpoint the locations of different classes within the image.

Peng *et al.*proposed their architecture, the Global CNN, to tackle these challenges. However, the well-known U-Net model also effectively addresses both classification and localisation. In U-Net, the contracting path handles the classification while the expanding path and skip connections aid in localisation, seamlessly integrating both tasks.

Another important advancement in image segmentation is the SegNet model developed by Badrinarayanan *et al.* [67]. SegNet is an encoder-decoder style network, similar to U-Net. However, unlike U-Net, SegNet does not use skip connections; instead, it saves the indices of the max-pooling operations. Later, in the decoder stage, these indices are used to upsample the values to their corresponding locations. This approach makes the model more efficient, as it does not retain the feature maps of the encoder in memory and avoids the use of computationally expensive deconvolution for upsampling. Furthermore, SegNet uses the VGG model for its encoder path, including its pre-trained weights.

2.6 Transformers

2.6.1 Introduction to Transformers

Vision transformers are a new class of neural network architecture that has achieved state-of-the-art results on computer vision tasks with large datasets. Originating from natural language processing (NLP), the technique was applied in computer vision by Dosovitskiy *et al.* [21] in 2020, causing a boom in the computer vision field.

Transformers introduced by Vaswani *et al.* [20] in 2017 demonstrated that the network based solely on attention mechanism neglecting the recurrent or convolutional structures in language models has more potential in quality and is faster to train due to more parallelism capabilities. The transformer model consists of the encoder and decoder modules.

The encoder in the original work of Vaswani *et al.* consists of 6 identical transformer blocks. The depth of the encoder is the hyper-parameter, which can be adjusted depending on the task. Each transformer block comprises a multi-head self-attention layer and fully connected feed-forward layers. Both parts of this block, the attention layer and fully connected parts have residual connections [4] around them, followed by a layer normalisation [68]. Before fitting into the encoder, all tokens are embedded, and positional encoding is applied to the embeddings. The tokens fit into the encoder simultaneously, which shows high paral-

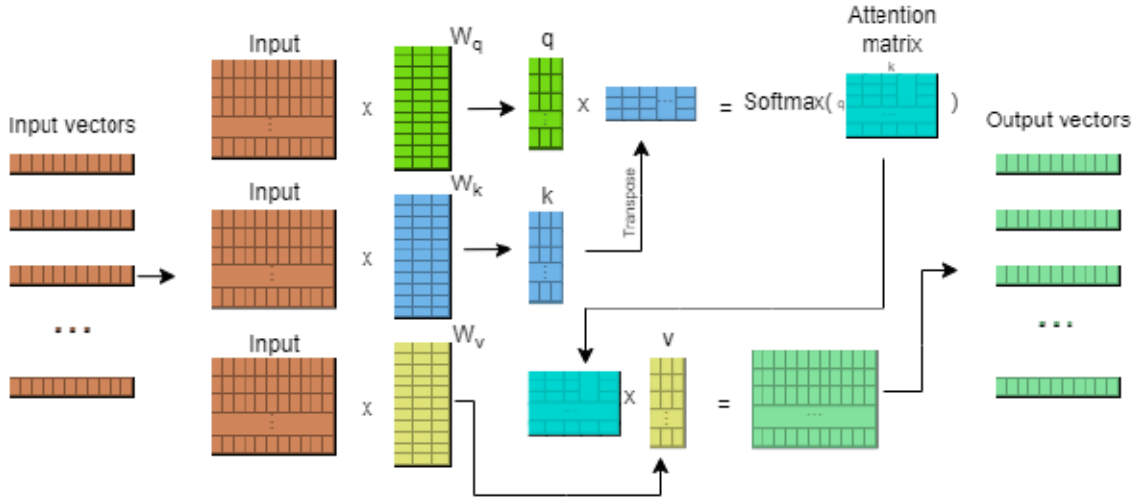


Fig. 2.6 **Visual schematic of the attention mechanism.** Input vectors are combined into one matrix and multiplied by weight matrices W_q,W_k and W_v. Outputs of W_q,W_k are used to form an attention matrix by multiplying Q with the transpose of K. This produces attention scores, which are then passed through a softmax function to create a distribution that reflects how strongly each input vector relates to the others. The attention matrix is then multiplied by the output of the W_v, forming the output of the self-attention layer.

lelism of the methods, unlike the sequence-to-sequence models, which accepted one token at a time.

The decoder, which is not used in the vision transformers but is a crucial part of the language models, generates the target sequence. While the encoder reads in the source sequence in parallel, the decoder generates the output sequence one token at a time.

2.6.2 Attention Mechanism

To understand the application of transformer encoders in computer vision, it is essential to look at the self-attention mechanism, a core component of these models.

In the self-attention mechanism, each input vector is an embedded token that can be represented as a point in an n-dimensional vector space. The primary goal of the attention layers is to compute the relationships between each pair of embeddings and update their positions relative to one another based on these relationships. This is done by using three components: queries, keys, and values. The detailed pipeline on tensor manipulations is shown in Figure 2.6.

The input to a self-attention layer is a sequence of vectors, x_i, which together form a matrix X. This matrix is multiplied by three trainable matrices: W_q (for Queries), W_k (for Keys), and W_v (for Values). These transformations produce the matrices Q, K, and V,

respectively, where each matrix is a list of transformed embedded vectors. The multiplication of the matrix X with each of W_q, W_k, and W_v is a linear transformation, similar to what is commonly known as a "Linear layer" in deep learning without a bias term.

When the input x is mapped to Q, K, and V, each vector is transformed independently, without any interaction among them during this phase. However, the interaction among vectors occurs in the subsequent steps through the computation of the dot products between each pair of Query and Key vectors. Specifically, to calculate the relationship $\alpha_{i,j}$ from input i to j, the dot product $q_i \cdot k_j$ is computed. This calculation is extended across all vector pairs to form the attention matrix through the matrix multiplication QK^T.

To ensure that each row of the attention matrix sums to one, a softmax function is applied to each row after matrix multiplication. Additionally, since the dot product values can grow large as the dimension of the vectors increases, the product QK^T is scaled down by $\sqrt{d}$, where d is the dimension of the input vectors. This scaling helps to prevent the softmax function from producing extreme values, ensuring smoother gradients during learning. The final formula for the attention matrix calculation is shown in Equation 2.13.

$$\alpha = softmax(\frac{QK^T}{\sqrt{d}}) \tag{2.13}$$

The output of the attention layer is then calculated by multiplying the α attention matrix by the value matrix V. This multiplication combines the weighted contributions from all input elements to produce the final output of the layer. This dynamic combination of inputs based on their computed contextual relationships is crucial for tasks requiring contextual understanding, such as in natural language processing and increasingly in computer vision.

2.6.3 Multi-head attention

Single-head attention provides a framework where the model learns to assign a weighting to each input token, determining how much focus to allocate where. However, this single set of weights can sometimes limit the model's ability to capture diverse relationships within the input data.

Multi-head attention extends the concept of single-head attention by incorporating multiple attention mechanisms—or "heads"—in parallel. This method allows the model to capture various aspects of the information contained in the input sequence simultaneously. Each head can be seen as an independent attention mechanism, with each focusing on different parts of the input sequence or interpreting the inputs in different ways.

The embed dimension must be divisible by the number of heads. The input embeddings fit into the multi-head attention layer and are split into h vectors, where h is the number of

heads. Each i-th head starts by linearly transforming the input using different learned weights into the Q_i, K_i, V_i. Each head operates like a regular self-attention layer. The outputs of the attention layers are then concatenated and can be linearly transformed again.

$$MultiHead(Q,K,V) = Concat(head1, head2, .., head_h)W^o \tag{2.14}$$

2.6.4 Architecture of ViT

Unlike the NLP, images in computer vision do not contain the tokens, and the image is a matrix rather than a sequence. Hence, image classification using transformers may not seem to be an ideal choice. However, every image can be considered a sequence of image patches in both x and y dimensions. Taking this into account, in 2021 Dosovitskiy *et al.* [21] proposed a method of image classification using vision transformers, using no convolutional layer at all.

The idea is to tokenise the image and fit the image tokens into the transformer encoder, just like in NLP tasks. To turn the image into the sequence of tokens, the image $x \in \mathbb{R}^{H \times W \times C}$ is split into non-overlapping patches $x_p \in \mathbb{R}^{N \times (P^2 \times C)}$, where H is hight, W is width, C is the number of channels, and P is the width and height of the patch. Following this tokensisation method, the number of patches result in $N = H \times W / P^2$. One of the Dosovitskiy *et al.*'s models, ViT-L/16, split the ImageNet images into 16×16 patches, taking the ImageNet resolution 224×224, the result is 196 tokens per image. The tokens flattened, resulting in vectors of length 256 (See Figure 2.7). Although, in Figure 2.7, the inputs are shown as distinct vertical vectors. However, it is crucial to understand that, in practice, these vectors are transposed and consolidated into a singular matrix and kept in that form across the entire network, as described in subsection 2.6.2.

Since the transformer uses the constant vector size D in all layers, the vectorised patches are fit into the linear layer to embed the input vectors into the desired D dimension. After the embedding, another trainable vector known as the classification token is added to the sequence, which keeps the information about all other patches throughout the training process, resulting in overall N+1 tokens.

Position encoding are added to the tokens to preserve the positional information of each original patch. These encodings can be either learnable vectors, like the ones used by Dosovitskiy *et al.* [21], or fixed encoding generated through functions like sine and cosine. Without any form of positional encoding, the network would treat an image with shuffled patches the same as the original, leading to a loss of spatial understanding.

Fig. 2.7 **Vision Transformer architecture.** The pipeline starts by dividing the image into patches, which are then flattened into vectors to represent the input tokens. Next, these tokens are linearly transformed using linear layers in a process known as input embedding. To indicate the position of each token in the original image, positional encoding is added to the embeddings. An additional token, called the classification token, is included in the list of embeddings. This token will ultimately contain information about all tokens and will be used for classification. As the inputs progress through the network, they are passed through self-attention layers and MLP layers. Finally, the CLS token is placed into the linear classifier.

After this process, all tokens are fit into a standard transformer encoder, and only the output of the last layer's classification token fits into the classification head, which consists of the standard MLP classifier.

It is important to note that, nevertheless, only the classification token is used for the classification; due to the attention mechanism, it contains all the required information about other patches, as it had to attend all of them during the training process.

It is important to note that, nevertheless, only the classification token is used for the classification; due to the attention mechanism, it contains all the required information about

other patches, as it had to attend all of them during the training process. As the network processes the input, the tokens' representations are adjusted and tend to align in the vector space. This alignment occurs because the layers of the network modify how close or relevant the vectors of these tokens are to each other, depending on the specific requirements of the task. Therefore, the classification token interacts with all other patch tokens via the attention mechanism to gather a detailed representation and capture crucial features for accurate classification.

Out of all operations in the pipeline, the trainable paremeters include patch embeddings, position encoding, weights for the queery, key and value, and the weights of the MLPs including the classification head. Attention matrix, although depends on the weights of the attention layer, are computed dynamically based on the inputs and multiplied by the value matrix.

2.6.5 Application of Vision Transformers

ViT has less built-in inductive bias compared to CNNs. This means that ViT does not assume translation equivariance, locality, or neighbourhood structure. However, ViT has a larger effective receptive field compared to CNNs, while CNNs require more stacked layers or larger kernels to maximise the effective receptive field.

Therefore, ViT generalises better than CNN only with large datasets. The best ViT model (ViT-H/14) achieves 88.55% top-1 accuracy on ImageNet classification. In order to train the ViT for the smaller datasets, it is required to train it first on the larger ImageNet dataset and then fine-tune it for the smaller datasets like CIFAR-100. Following that, the fine-tuned ViT-H/14 achieves 94.55% on CIFAR-100 and 77.63% on the VTAB suite of 19 diverse vision tasks.

As Dosovitskiy *et al.* [21] mentioned that ViT does not perform well on small datasets, their comparison of results on small datasets focused on fine-tuning pre-trained models on ImageNet rather than training from scratch. However, Lee *et al.*[69] reported the results of training ViT on datasets such as CIFAR-10, CIFAR-100, SVHN, and Tiny-ImageNet from scratch. They improved the performance using novel methods, including Shifted Patch Tokenization (SPT), CoordConv Position Encoding (CPE), and Locality Self-Attention (LSA). Lee *et al.*'s [69] variation of ViT consists of 9 transformer layers with 12 heads per layer and the embed dimension of 192. The patch size was set to 4×4, for images of 32×32, resulting in 64 patches per image. The authors used regularisation as label smoothing [70], stochastic depth [71], random erasing [72] and weight decay in AdamW [73] optimiser which was set to 0.05. Moreover, the model trained with augmentation used methods like CutMix, Mixup, Auto Augment, and Repeated Augment. According to the authors, ViT trained from

scratch with CIFAR-100 achieved an accuracy of **60.01%** without and **73.81%** with the augmentation methods described above.

In the work of Zhu *et al.* [74], the authors compared ResNet-18 to CIFAR-100, investigating the reasons for ViT's poor performance on small datasets. Their variation of ViT has a depth of 6, each with 8 heads and an embedding dimension of 64. The authors reported an accuracy of only 54.31%, which is lower than Lee *et al.*[69] reported. The reason for that could be a small depth and lack of augmentation and regularisation methods in the work of Zhu *et al.*.

2.6.6 Other variations of Vision Transformers

One variation of ViT is the Pyramid Vision Transformer (PVT) [75], which reshapes the feature map into a matrix form after each transformer block and applies further patching. This results in a pyramid-like or cone shape architecture that is more suitable for computer vision tasks, particularly classification. It is important to note, however, that the progressive reduction in the feature map size after each stage of the transformer block distinguishes PVT from the standard ViT principles. While Transformers were originally designed for NLP, where tokens remain independent and communicate via the attention mechanism, PVT differs from this approach. Instead, it adopts a structure similar to cone shape CNNs designed for feature extraction.

The authors showed a 0.1% decrease in top-1 ImageNet accuracy by comparing their best model, PVT-Large, to the basic model, ViT-Base/16 [21]. However, PVT-Large performs with 9.8 GFLOPs compared to ViT-Base/16, performing with 17.6 GFLOPs. Moreover, the authors also show the memory efficiency, as PVT-Large contains 61.4M parameters, while ViT-Base/16 contains 86.6M parameters.

Although PVTs have a cone-shaped architecture like CNNs, they do not include any convolutional layers in the pipeline, like Vision Transformers. However, that cannot be said about another work by Wu *et al.* [76], called Convolutional Transformers (CvT). Similar to the contribution in this thesis, Wu *et al.* applied the convolution into the vision transformers with slight deviation from our method. CvT employs a convolution operation in the token embedding and embedding projection of the Vision Transformer.

The CvT model starts by fitting the input image into a Convolutional Token Embedding layer. This performs overlapping convolutions to split the image into tokens and capture local spatial information. Moreover, this stage decreases the length of the sequence and increases the dimension of token features, mimicking the architecture of the CNN classifier, where the feature maps are spatially downsampled while the number of feature maps (channels)

increases. The output token embeddings are normalised following the token embedding layer, using Layer Normalisation.

Then the output token embeddings are passed into the Convolutional Transformer Block. In each Convolutional Transformer Block, the tokens first go through a Convolutional Projection layer, which applies depthwise separable convolutions to obtain the value, query and key embeddings instead of the matrix multiplications as in standard ViT [21]. Tokens are then flattened and passed through a standard transformer block containing a multi-head self-attention layer and MLP. The outputs of this stage are then passed to the next Convolutional Token Embedding layer.

Although this method is called a Convolutional Vision Transformer, it uses standard multi-head self-attention layers and MLP in all transformer blocks.

In contrast to standard ViTs, CvT uses overlapping patches and reconstructs the 2D image after each transformer block to re-tokenize it using convolution operations before the next block. This way, the feature map pools down and retains its cone-shaped structure typically used for classification tasks in CNNs

In standard ViTs, an image is segmented into discrete non-overlapping patches. Each of these patches progresses independently throughout the network architecture. Interaction between the patches is facilitated only via the attention mechanism, ensuring that while there is communication between patches, their representations remain distinct and non-integrated. However, in CvT, the output of each convolutional transformer block is merged into one unified feature map by rearranging the dimensions and turning input vectors into the 2D matrices and then pooled down throughout inference. Therefore, it can be noted that the CvT architecture also does not follow the principles of the transformer. Instead, it integrates the attention mechanism into the CNN.

Moreover, due to the overlapping patches and hierarchical structure of the CvT, applying a position encoding in the transforms is not required. In fact, Wu *et al.* [76] have demonstrated that the top-1 accuracy on ImageNet improved from 81.4% to 81.6% when restrained from the position encoding.

2.7 Effect of high-resolution training

Kernel size is one of the essential hyperparameters of CNNs. Standard models today use small kernel sizes, usually 3×3 or 6×6, except the first layers of the network, which can go as large as 11×11 like in the first layer of AlexNet or 7×7 in ResNet.

Relatively small kernel size standards are the results of thousands of experiments, which show that having a 3×3 kernel size will not lack the result and will be fast enough. Theoreti-

cally, having a small kernel size has a range of advantages. The reduction of kernel size not only increases the computational efficiency during training but also decreases the number of trainable parameters, thereby increasing the robustness of the network against overfitting. Larger kernels, by increasing the number of trainable parameters, not only have a tendency to overfit the network by learning too much noise and detail from the training data but also slow down the training process due to the higher computational load required per layer. Moreover, due to the dominance of CPU-GPU training, where the large kernel size has a negative effect on the training speed, the research on the larger kernel size in CNNs has been lagging.

As mentioned in Section 2.5, Peng *et al.* [66] proposed a Global Convolutional Network (GCN) architecture that enables using very large kernels, up to the size of the feature map, for semantic segmentation. The motivation is that current semantic segmentation models focus primarily on precise localisation, which can degrade their classification capability. GCN improves classification performance by using large kernels that densely connect feature maps and per-pixel classifiers.

However, the authors claim that large kernels are inefficient and use separable convolutions like $1 \times 15 + 15 \times 1$ to reduce parameters and computation in GCN. Experiments show GCN consistently outperforms regular small kernels and stacks of small kernels for segmentation. The "global convolutional" version improves IoU by 5.5% over a 1×1 baseline. Analysis reveals GCN mainly benefits internal regions of objects, while a Boundary Refinement block they propose improves localisation near boundaries.

Applying GCN to standard ResNet backbones boosts segmentation performance despite marginally decreasing image classification accuracy. Their model achieves new state-of-the-art results on PASCAL VOC 2012 (82.2% IoU) and Cityscapes (76.9% IoU), significantly outperforming prior art.

The key conclusions are that larger kernels provide substantial benefits for semantic segmentation by improving classification capability, and their GCN architecture effectively enables this in an efficient separable manner. The results demonstrate the importance of large kernels and dense classifier connections for pixel-level classification tasks.

In the work of Agrawal and Mital [6], the authors studied the effects of different kernel sizes and a number of filters in CNN to train the FER-2013 and uniquely approached the problem. They have introduced a small network called the Constituent layer, which consists of two convolutional layers and one softmax. The network was trained with varying kernel sizes, such as 2,4,8...64 and the number of output channels of 2,4,8 ... 256.

After training the small two-layered network, the authors found four combinations of two-layered networks with the number of channels and kernel sizes $16/4 \times 4$, $32/8 \times 8$, $4/16 \times 16$ and $2/32 \times 32$, which they used to find the optimal depth of the network.

Table 2.2 Constituent layer in the work of Agrawal and Mital [6]. K represents the kernel size, and C is the number of output channels.

CONV $K \times K \times C$, BATCH NORM
CONV $K \times K \times 7$, RELU, STRIDE (128×128)
SOFTMAX

By increasing the depth of the network, the authors found the best converging network, which is a number of output channels of 32 and kernel size of 8×8, with a depth of 16 convolutional layers, resulting in the model which they have called Model1. Later, the authors modified *Model 1* into *Model 2* by reducing the number of channels and increasing the depth of the networks. The standard CNNs use the VGG style approach and increase the number of layers, while *Model 2* is unique in terms of slowly decreasing the number of channels.

As expected, Model2 performed marginally worse than *Model1* (by 0.54%), but with two times fewer trainable parameters. The authors compared the model size to VGG-19, which contains 21.5 times more trainable parameters than Model1, but they did not compare the performance. Instead, they have compared the performance to other smaller networks inspired by AlexNet and VGGNet, like *Subnet1* by Liu *et al.* [77]. It has to be said, while *Model1* and *Model2* achieved an accuracy of 65.77% and 65.23% on FER2013 dataset respectfully, VGGNet by Pramerdorfer and Kampel [78] achieved 72.70% and later in 2021 VGGNet by Khaireddin and Chen [79] achieved 73.28% top-1 accuracy with FER-2013 dataset. Taking into account only the work of Pramerdorfer and Kampel [78] as it was published earlier than the *Model1* and *Model2* of Agrawal and Mital [6] the loss in performance is 6.93%, which the authors did not mention.

In the work of Ding *et al.* [36], the authors explored the potential of large convolutional kernels in modern CNNs. This research is inspired by recent advances in Vision Transformers, which have demonstrated superior performance across various visual tasks. The authors aimed to bridge the performance gap between CNNs and ViTs by using large kernels in CNNs.

A critical aspect of this study is the comparison between large-kernel CNNs and the multi-head self-attention (MHSA) mechanism used in ViTs. The authors find that large convolutional kernels can effectively mimic the behaviour of MHSA by capturing long-range dependencies and building large receptive fields (ERFs). This allows CNNs with large kernels to achieve performance comparable to ViTs without the computational complexity associated with attention mechanisms. The study reveals that while traditional small-kernel

Table 2.3 Model1 and Model2 by Agrawal and Mital [6]. The difference is indicated in bold.

Model 1	*Model2*
Input data (64 × 64) grayscale image	Input data (64 × 64) grayscale image
Data augmentation	Data augmentation
CONV 8 × 8 × 32, BATCH NORM	CONV 8 × 8 × 32, BATCH NORM
CONV 8 × 8 × 32 (stride 2), RELU	CONV 8 × 8 × 32 (stride 2), RELU
CONV 8 × 8 × 32, BATCHNORM	CONV 8 × 8 × 32, BATCHNORM
CONV 8 × 8 × 32 (stride 2), RELU	CONV 8 × 8 × 32 (stride 2), RELU
CONV 8 × 8 × 32, BATCHNORM	CONV 8 × 8 × 32, BATCHNORM
CONV 8 × 8 × 32 (stride 2), RELU	CONV 8 × 8 × 32 (stride 2), RELU
CONV 8 × 8 × **32**, BATCHNORM	CONV 8 × 8 × **16**, BATCHNORM
CONV 8 × 8 × **32** (stride 2), RELU	CONV 8 × 8 × **16** (stride 2), RELU
CONV 8 × 8 × **32**, BATCHNORM	CONV 8 × 8 × **16**, BATCHNORM
CONV 8 × 8 × **32** (stride 2), RELU	CONV 8 × 8 × **16** (stride 2), RELU
CONV 8 × 8 × **32**, BATCHNORM	CONV 8 × 8 × **16**, BATCHNORM
CONV 8 × 8 × **32** (stride 2), RELU	CONV 8 × 8 × **16** (stride 2), RELU
CONV 8 × 8 × **32**, BATCH NORM	CONV 8 × 8 × **8**, BATCH NORM
CONV 8 × 8 × **32** (stride 2), RELU	CONV 8 × 8 × **8** (stride 2), RELU
CONV 8 × 8 × **32** BATCH NORM	CONV 8 × 8 × **8**, BATCH NORM
CONV 7 × 7 × 7, RELU	CONV 7 × 7 × 7, RELU
SOFTMAX	SOFTMAX

CNNs require many layers to build large ERFs, a few large kernels can achieve this more efficiently, enhancing both performance and computational efficiency.

The methodology by Ding *et al.* centres on the implementation of a novel CNN architecture called RepLKNet, which uses re-parameterised large depthwise convolutions with kernel sizes up to 31×31. The authors provide five key guidelines for effectively incorporating large kernels into CNNs. They demonstrate that large depthwise convolutions can be computationally efficient, especially when optimised for GPUs. The use of identity shortcuts is critical for maintaining performance and stability in networks with very large kernels. They introduce a re-parameterisation technique where smaller kernels are used alongside large ones during training and then merged, addressing optimisation challenges.

The experimental results prove the efficacy of the RepLKNet architecture. On the ImageNet-1K dataset, the RepLKNet-31B model achieves an impressive 84.8% top-1 accuracy, outperforming the Swin-B model by 0.3% while being 43% faster in inference speed. This indicates not only superior accuracy but also enhanced computational efficiency. Furthermore, in semantic segmentation tasks on the ADE20K dataset, RepLKNet-31B outperforms Swin-B with higher mean Intersection over Union (mIoU) scores, particularly in single-scale evaluations.

The authors provided evidence that revisiting and scaling up convolutional kernel sizes in CNNs can lead to significant performance improvements. By effectively using large kernels, they demonstrate that CNNs can achieve, and even surpass, the performance of state-of-the-art ViTs in various tasks, especially in semantic segmentation. However, even though the authors tried to replicate the ViTs using the CNNs, it is hard to say that this is the emulation of the ViT since the general concept of CNNs is still feature extraction. Nonetheless, there is no doubt that CNNs with larger kernels can match the high effective receptive field of ViTs.

2.8 Free-space optical deep learning accelerator

The overarching goal of optical neural networks is to implement artificial neural network models directly in optical hardware by taking advantage of the unique properties of light. A core requirement for any neural network implementation is the ability to perform multiply-accumulate (MAC) operations, which involves multiplying large matrices by input vectors. In conventional digital hardware, these matrix-vector multiplications require breaking the operation into separate sequential multiply and accumulate steps. However, optics can intrinsically perform these matrix-vector multiplications in a single parallel step. This section is primarily focused on the acceleration of using different free-space optical accelerators. This section places greater emphasis on 4f-based optical correlators, as this work primarily utilises this technology.

2.8.1 Deep Learning with Free-space Optical Accelerators

Optical computing has emerged as a solution for enhancing the efficiency of deep learning processes by using the computational capabilities of light. A notable example of this technology is the implementation of the 4f optical system, which is adept at performing two-dimensional convolutions that are critical for many deep learning applications. This system consists of two lenses and a mask located in the Fourier plane, all spaced by the focal

length, f, of the lenses. This configuration allows for the convolution of an input image with a filter mask positioned within this plane, with the resultant pattern displayed on the image plane representing the convolved output.

The use of free-space optical accelerators, such as the 4f system, is prevalent due to its relevance in specific tasks across various types of neural networks. This section highlights the predominant application of the 4f system, alongside a review of both passive and active accelerators in subsequent sections. Previous implementations of the 4f system have not typically supported the complete inference of a model through a single pass. Instead, each convolution layer or, in some cases, only one layer of the network is processed sequentially. One significant limitation is the system's inability to perform nonlinear activations, which must be handled electronically after image readout.

Despite these technological advances, optical computing faces substantial challenges in neural network training. Traditional training methods, such as backpropagation, require iterative adjustments to network weights, necessitating a level of dynamic reconfigurability that traditional optical setups do not provide. Currently, most optical neural networks are trained offline using digital systems. After training, the optimised weight matrices are then implemented in the optical hardware for rapid inference tasks. This limitation prevents real-time training capabilities within current optical neural network systems. However, the inference capabilities of the 4f system could be adapted for training purposes. The 4f system could theoretically involve backpropagation as well. As backpropagation in CNNs includes the convolution operations, which the 4f system can execute.

2.8.2 Passive optical Accelerator

Passive optical accelerators refer to the setup with elements like lenses or diffractive elements that do not consume power.

One example of free-space passive optical accelerators is the diffractive deep neural networks (D^2NN) [41]. The D^2NNs demonstrate remarkable capabilities in various optical computing tasks. These networks utilise diffractive layers to modulate light for performing functions such as image classification and feature detection directly with light waves. In this particular work of Lin *et al.* [41], the D^2NN models were trained and tested on two datasets: MNIST for handwritten digits and Fashion-MNIST for clothing items. Even though the networks performed well on both datasets, achieving an accuracy of 93.39% and 86.60% for MNIST and Fashion-MNIST respectively, the performance is still far from the state of the art in the electronic setup. However, the high speed of the terahertz frequency range and the low power consumption required for performing classifications make the D^2NN a preferable accelerator for some use cases.

As the diffractive layers are physical and cannot be altered, D^2NNs are trained on a computer, and the layers are fabricated using a 3D printer once the preferred training level is achieved. Once the layers are fabricated, the device cannot be altered and therefore can be used for only one task, but will operate passively and perform the inference with the speed of light. This research opens new avenues for the development of optical components and systems for rapid, power-free computing, furthering the integration of machine learning algorithms into physical layer tasks. Since 2018, there has been increasing research based on diffractive neural networks, like the application of the non-linearity in the D^2NN.

Unlike the diffractive neural networks, the 4f system that we are focusing on in this thesis can also be active and passive. Some of the earliest research in this area is the work of Chang *et al.* [1], where the authors demonstrated the CNN implemented in the 4f system, with the phase mask in the Fourier plane. This means that the multiplication of the image and the kernel in the Fourier domain happens passively, and no modulator is used in this stage. Just like the D^2NN the main disadvantage of this method is the lack of flexibility as the device can only be used for the convolution with the preset and fabricated kernels and cannot be altered during inference. The phase masks need to be replaced if the 4f system goes through more training or fine-tuning.

Before going into the optical convolution, the authors first tested the setup in the simulator using only a correlator. This means that the read-out of the camera is partitioned into the N sub-images corresponding to N classes. The sub-image with the maximum intensity is considered the output class of the classifier. The network is trained on the Google Quick Draw dataset, which contains 16 classes. In this preliminary setup, the learned optical correlator achieved a classification accuracy of 70.1%, demonstrating its capability to match the performance of a digital convolutional layer.

In their next step, the authors implemented a simulation of a full hybrid optoelectronic CNN. This included the simulation of the optical CNN layer followed by the regular ReLU + Fully connected layer applied in the electronic domain. This experiment was conducted on the CIFAR-10 dataset, and the authors achieved $51.0 \pm 1.4\%$ accuracy on the simulation of the setup while also using a pseudo-negativity (a method described in Section 3.2.5), in contrast to regular digital convolution which achieved $51.9 \pm 1.3\%$ accuracy. Unfortunately, when the authors tried the experiment physically, the performance dropped to 44.4%, which the authors explained as due to imperfections in the hardware not considered in the simulator. One of the reasons is that the SLM utilised only 16 discrete levels of height (bit depth of 4) rather than the continuous range assumed during the optimisation process. Additionally, the alignment of the convolved sub-images with the sensor pixels was not precise due to differences in pixel sizes between the DMD and the camera sensor, introducing errors during

the subtraction of kernels to produce pseudo-negative sub-images. Moreover, some dead pixels on the DMD were detected, which was observable when comparing the simulated input image against the actual projected version without a phase mask.

These factors are important, as this kind of deviation from the simulation and the real device can be observed in any 4f-based accelerator, even the one assumed in this work. Moreover, apart from the reasons mentioned by Chang *et al.*, the noise in the system can also play a significant role in the classification process and, in some cases, may introduce an augmentation [80].

In another work by Colburn *et al.* [53], the authors designed a hybrid optoelectronic convolutional neural network architecture that uses an optical frontend for the first convolutional layer of AlexNet, while subsequent layers are processed electronically. The architecture uses an array of 288 4f correlators to perform the convolution operations in parallel, corresponding to 96 kernels for each of the three colour channels (red, green, and blue). These 4f correlators are implemented using metasurface optics to create a compact and efficient system. The correlators consist of two lenses of equal focal length spaced apart by $2f$, with input and output planes at the focal points of the first and second lenses. Similar to the work of Chang *et al.* [1], the authors used a fixed phase mask in the Fourier plane.

The optical frontend is used with a modified version of AlexNet, which typically comprises five convolutional layers and three fully connected layers. In this hybrid approach, the first convolutional layer is implemented optically, reducing the need for multiple energy-inefficient optical-electronic signal conversions. Since the first layer is the most computationally expensive, with a kernel size of 11×11, it is advantageous to use optical acceleration here. The remaining four convolutional layers and three fully connected layers are realised electronically. The authors conducted simulations rather than using real devices for their study. This design was benchmarked using the Kaggle Cats and Dogs dataset, achieving a classification accuracy of 87.1%, which is comparable to the ground truth accuracy of 87.3% for the fully electronic version of the modified AlexNet. The study demonstrates that the hybrid system maintains high classification accuracy while offering potential advantages in processing speed and energy efficiency for large image sizes due to the parallelism and low latency of the optical computations.

2.8.3 Active 4f optical Accelerator

To overcome the limitations of passive optics, a new category of optical neural network architectures includes active optoelectronic devices for high-speed reconfigurability and training. This approach uses DMDs, which contain millions of individually controllable tilting mirrors or liquid crystal SLMs. DMDs can modulate light patterns at speeds of tens

of kHz, which is orders of magnitude faster than liquid crystal SLMs. Configured into 4f optical systems, DMDs enable rapid amplitude-only optical multiplications in the Fourier domain for optical convolution operations.

Even faster reconfiguration may be possible with analogue optical devices. Prototype analogue micromirror arrays capable of MHz tuning speeds have been developed. With millions of micromirrors, these devices could pave the way toward fully optical neural network training at unprecedented speeds. Research has shown promise, and rapid improvements in microelectromechanical systems (MOEMS) and nanophotonic devices suggest continued advances [39, 81, 82].

In the work of Schultz *et al.* [83], a 4f optical correlator was explored to accelerate CNNs. The authors implemented the 4f system with SLMs and a camera, demonstrating its potential to perform convolutions optically. They used a simple architecture comprising a convolutional layer with sixteen kernels, followed by batch normalisation, max pooling, a fully connected layer, and ReLU activation. The authors trained only the positive half of the kernel, disregarding any pseudo-negativity methods, and also trained the kernels in the Fourier domain to avoid conversions into the Fourier domain during training. Achieving a classification accuracy of 91% on the MNIST dataset, the system highlighted the efficiency of 4f systems in parallel processing, although there was a drop in performance compared to the regular CNN's accuracy of 98.2%. The use of SLMs allowed the dynamic modulation of the Fourier plane, enabling the implementation of different kernels without physical alterations.

Dai *et al.* [84] proposed an innovative on-chip 4f system based on concave mirrors to implement the Fourier transform. Although in this thesis, we assume a free-space 4f system, Dai *et al.*'s work implements the 4f system in an innovative manner. This system performs on-chip Fourier transforms, using etched concave mirrors to simplify construction, reduce costs, and decrease the focal length of the mirrors to $2000\mu m$. The study simulated convolution operations on datasets such as Iris-Flower, MNIST, and Fashion-MNIST, achieving classification accuracies of 96.67%, 95.6%, and 88.8%, respectively, compared to the electronic counterparts' 97.3%, 97.9%, and 89.8%. The on-chip 4f system demonstrates performance comparable to electronic counterparts, with performance losses of 0.63%, 2.3%, and 1% for Iris-Flower, MNIST, and Fashion-MNIST, respectively.

Li *et al.* [7] analysed the parallelism capabilities of the 4f system-based optical neural networks, considering an active 4f system with the DMD in the Fourier plane for dynamic kernel updates. The authors used the pseudo-negative approach introduced by [1], where the number of convolution operations is doubled and the results are subtracted from each other to emulate negativity in the 4f system. The best performance reported by the authors using channel/mixed tiling, as described in Section 2.8.4, was 93.6% for Fashion MNIST,

95% for SVHN, and 91.6% for CIFAR-10, trained on VGG-16. The authors did not report the performance of regular VGG-16 without the optical setup, as the main focus of the work was parallelism. According to other studies, regular VGG-16 is capable of classifying Fashion MNIST with an accuracy of 94%[85], SVHN with an accuracy of 97.85%[86], and CIFAR-10 with an accuracy of 92.05%[87] or 93.13%[86].

Gupta and Li [88] explored architectural improvements for 4f optical neural networks, addressing limitations such as positive sensor readout and intensity-only modulation. They used the channel tiling method to enhance throughput and precision without additional optical hardware. Their evaluation shows that a 4f system operating at 2MHz and 4K resolution can outperform an NVIDIA GTX 2080 GPU, achieving up to a 20-fold speedup for networks like AlexNet and VGG with a low 8-bit precision rate when trained on ImageNet. According to the authors, an 8.45-fold acceleration can be achieved against the NVIDIA GTX 2080 on VGG-16 with CIFAR-10 if channel tiling is used with the 8-bit precision rate. This highlights the potential of high-speed optical modulation techniques to significantly boost CNN performance. Moreover, the authors indicate that the pseudo-negative approach required at least 12-bit camera precision to remain within a 5% accuracy drop.

In their work, Miscuglio *et al.* [39] discussed the implementation of an amplitude-only Fourier neural network using a 4F system based on DMDs. Using high-resolution DMDs and free-space optics, the method performs convolutions as pixel-wise multiplications in the Fourier domain. This enables the processing of large matrices (approximately 1000×1000) in a single time step with a latency of $100\mu s$ and 8-bit precision. Additionally, it employed batch (input) tiling, processing up to 46 images simultaneously using the same kernel to maximise throughput. The authors used a DMD in the Fourier plane to demonstrate a dynamic change of the kernels, but used only a single-layered CNN, followed by batch normalisation, max pooling, and electronic linear layers.

The experimental optical setup, validated through training a CNN on the MNIST and CIFAR-10 datasets, achieved accuracies of 98% and 54%, respectively. In contrast, the space-domain convolution (regular CNN) achieved accuracies of 98% and 63%. Considering that MNIST is a relatively simple dataset, it can be assumed that the high accuracy is achieved due to the electronic linear layers following the Fourier convolution. This was more challenging with CIFAR-10, where the optical neural network experienced a 9% loss in accuracy.

Chen *et al.* [89] introduced a multilayer optoelectronic hybrid convolutional neural network that employs an optical recurrent structure within a 4f system to perform convolution operations. This setup uses an SLM in the Fourier plane to dynamically adjust the kernels. Unlike previous implementations that only partially utilised optical layers, this innovation allows the entire CNN to be processed optically. The authors employed the pseudo-negativity

method of Chang *et al.* [1] by simply subtracting the outputs of two convolution operations in the electronic domain, followed by activation functions and pooling, before fitting the results back into the 4f system for further convolutional layers. This recurrent structure enables the reuse of optical components across multiple layers, thereby extending the network's depth without the need for additional hardware, which is crucial for handling complex tasks.

Experiments were conducted on a custom CNN consisting of three convolutional layers with kernel sizes of 7×7 and output channels of 4, 4, and 8, respectively. The resulting feature maps were then subjected to max-pooling with a kernel size of 2×2, flattened, and fit into a fully connected layer. The model was first tested electronically on a grayscale CIFAR-10 dataset. The training involved the ADAM optimiser [90] with 2000 iterations and a small learning rate of 3×10^{-4}, resulting in a test accuracy of 49.8%. Although the performance seems poor, it should be noted that the dataset was grayscaled and the model consisted of only three layers. In the optical experiments, a test accuracy of 30% was achieved. According to the authors, the gap between the optical CNN and its electronic prototype is caused by system noise and alignment issues.

Another interesting study by Li *et al.* [91] used the 4f active system purely for augmentation purposes, with the SLMs used at the input and in the Fourier plane. The first modulator, in the object plane, modulated the image using standard augmentation methods such as translation, zoom, and shear. The SLM in the Fourier plane was used to generate a random phase. The authors implemented and tested their method on the MNIST dataset, training the optically augmented data on a two-layered CNN. The results showed that incorporating the proposed optical augmentation increased the classification accuracy from 93.24% (without augmentation) to 97.84% when combined with all augmentation transformations.

In summary, integrating active optoelectronic devices like DMDs and liquid crystal SLMs into the fourier plane of the 4f system enhances their speed and reconfigurability. DMDs, in particular, enable rapid optical convolutions in 4F systems. The development of analogue micromirror arrays with MHz tuning speeds suggests the potential for fully optical neural network training. Research shows that 4F systems can effectively accelerate CNNs, demonstrating competitive performance with some trade-offs in accuracy. These advancements indicate the possibility of using the 4F system for all layers of CNNs, potentially running entire models on the 4F system, especially with the high-speed potential of MOEMS and other nanophotonic devices.

2.8.4 High resolution capabilities of 4f Optical Neural Networks

Before going into the adaptation of the CNNs for the 4f system, it is important to go through the advantages and constraints of the system once again. The advantages include the high-

speed processing of the Fourier transformation, low power consumption and high-resolution capabilities. While the biggest constraint is the speed of the cameras and the modulators and the incapability of representing negative numbers. We have mentioned that the negativity problem can be mitigated using a pseudo-negative approach, like in the work of Chang *et al.* [1], by performing two convolutions instead of one and subtracting the result of one from another in the electronic domain.

To overcome the bottleneck of the camera read-out, we can take advantage of the high-resolution capabilities and parallelism. As the primary advantage of 4f free-space optical neural networks over the on-chip silicon photonics method is that the 4f system enables massive parallelism. This allows the 4f system to effectively process high-resolution inputs and kernels without compromising the frame rate. Since the 4f system efficiently processes high-resolution kernels, it is feasible to arrange multiple smaller kernels together in one large array to execute several convolutions simultaneously [91].

In their work, Li *et al.*[7] introduce several tiling approaches to take advantage of the parallelism of the 4f system. The authors mentioned filter tiling, which was initially introduced by Chang *et al.* [1]. It involves tiling kernels in one larger block to be convolved with one input image. The system can perform multiple convolution operations in parallel by tiling kernels, vastly improving throughput and efficiency. Essentially, it convolved one image with several kernels, where the output becomes equal to a number of 2D filter tiles. Filter tiling requires careful consideration of the optical system's resolution and kernel resolutions and their quantity. The main task here is to pad the kernels in a way that after tiling they do not overlap during the convolution operation.

The process starts with zero padding of the kernels to a size of $(M+N-1)\times(M+N-1)$, where $M\times M$ represents the resolution of the input and $N\times N$ denotes the resolution of the kernel. Subsequently, these padded kernels are tiled into one large kernel block. At this stage, the input must also be padded to the same dimensions as the kernel block to facilitate optical convolution. Following this convolution, the result is a set of tiled outputs, each corresponding to the convolution of the input image with one of the individual kernels within the tiled array (See Figure 2.8).

We have tested this method by training a small network consisting of two convolutional layers and one dense layer on the MNIST dataset. Sixteen kernels were extracted from the first layer of the network, as shown in Figure 2.9. The image of the digit '7' from the MNIST dataset, which was padded as in Figure 2.10 (a), was convolved with the earlier extracted kernels. These kernels were also padded and tiled into one block, as illustrated in Figure 2.10 (b). The outputs of this example can be seen in Figure 2.10 (c), which displays 16 output images. Each image corresponds to the convolution of the input with a respective

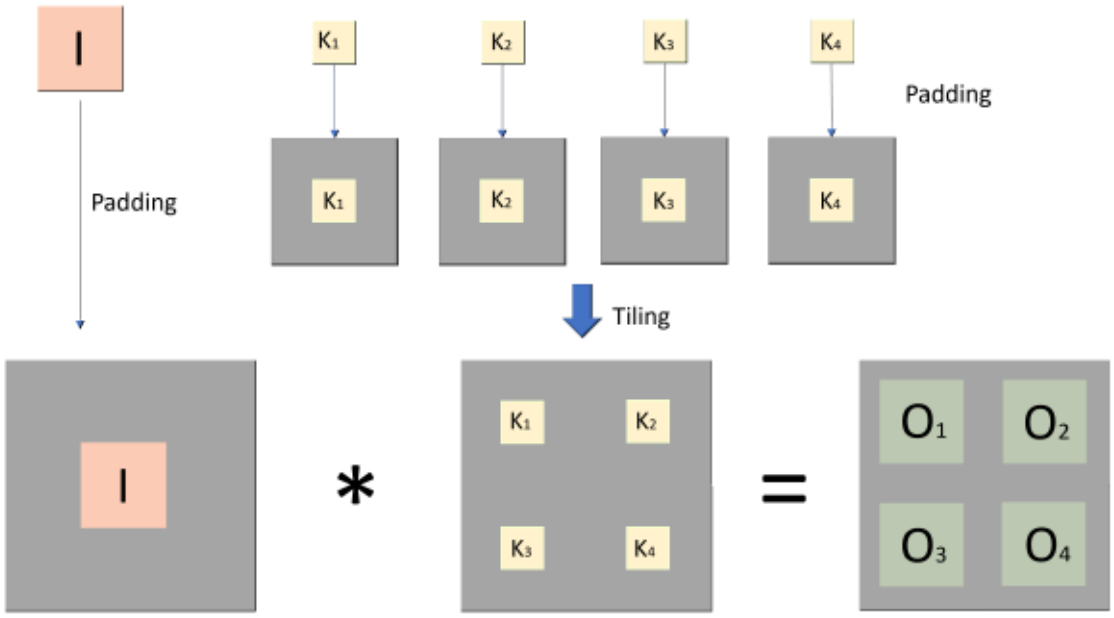


Fig. 2.8 **Implementation of the convolutional layer in optics using the kernel tiling.** Kernels are first padded and tiled into one block. Consequently, the input is padded to the resolution equal to the newly tiled kernel block. The output of the convolution of these blocks will result in the convolution of input with each kernel tiled in the kernel block.

kernel in the kernel block. Without the tiling, all 16 convolutions would have to be performed sequentially. Thus, the kernel tiling method, in this case, accelerates the convolution process by a factor of 16.

Since the convolution operation is commutative, the order of the operands does not matter:

$$f * g = g * f$$

where f and g are functions and $*$ denotes convolution.

Given this property, we can assume that inputs can be tiled similarly to the kernels, as shown in the previous example. This allows for the implementation of batch tiling for inference, enabling several convolutions to be performed simultaneously with a large minibatch. Suppose the batch size is equal to 64, then the inputs can be tiled into a "input block" of 8×8 and convolved with each kernel separately in the network. In the following chapters on CNN-based Image Classification 3 and Segmentation 4, this method of parallelism is assumed.

Another way of parallelism of the CNNs with the 4f system is channel tiling. Channel Tiling aims to use the optical system's ability to sum inputs across channels. This method effectively uses entire space on both modulators, essentially taking full advantage of the

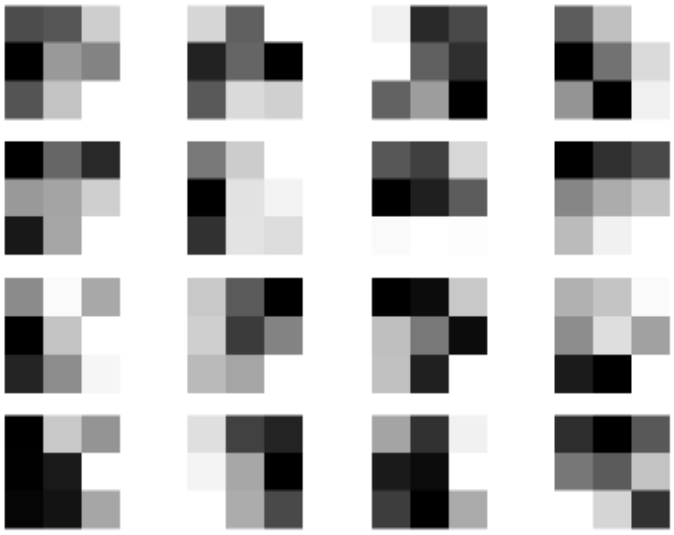
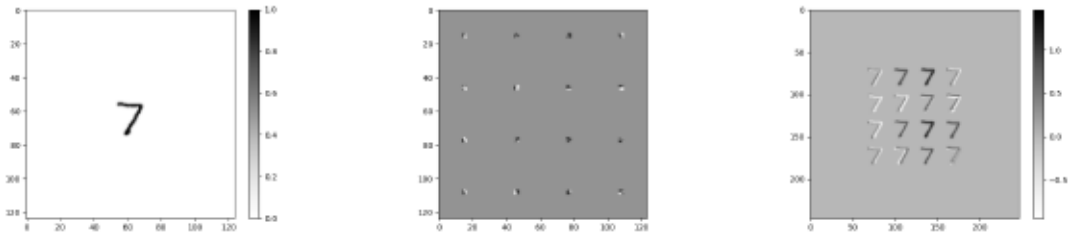


Fig. 2.9 **Kernels from the first layer of the MNIST neural network used for the example in Figure 2.10.** The kernels are extracted from the fully trained custom convolutional neural networks from scratch for demonstration purposes.



(a) (b) (c)

Fig. 2.10 **Experimental results of the kernel tiling technique applied to an input channel of MNIST digit "7" and sixteen kernels extracted from the model trained on MNIST.** (a) The input image padded the resolution of the kernel block (b) Kernel block containing all kernels padded and tiled together (c) Tiled outputs of the convolution of the input image with the corresponding kernel from the kernel block, achieved simply with the convolution of the matrix (a) and (b).

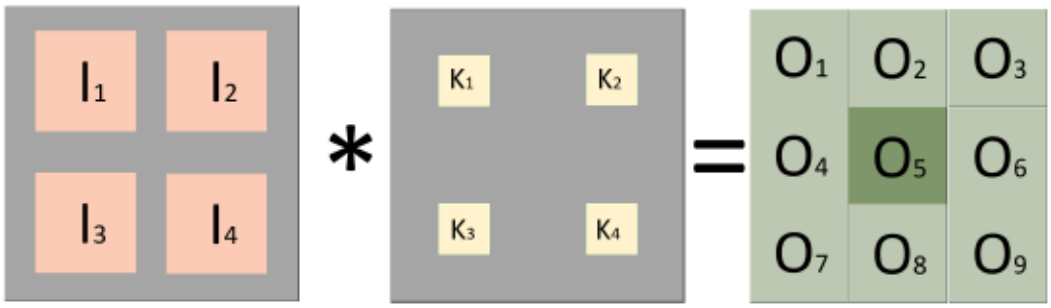


Fig. 2.11 **Implementation of the convolutional layer in optics using the channel tiling.** Channels first padded and tiled into one block. Kernels of the particular output are consequently padded and tiled in the locations corresponding to their input channels. The middle picture of the output block,O_5, is the sum of the convolutions of each input channel with the corresponding kernel.

high-resolution capabilities of the device. Just like in the kernel tiling, in this case both kernels and channels are padded to the $(M+N-1)\times(M+N-1)$, where $M\times M$ represents the resolution of the input and $N\times N$ denotes the resolution of the kernel. Both channels and kernels are tiled into a channel and kernel block and convolved.

The output of the convolution of both blocks will yield the block of outputs with $(2\sqrt{N_c}-1)\times(2\sqrt{N_c}-1)$ outputs, where N_c is the number of input channels. All outputs are invalid, except the output in the middle, which is the summation of the convolution of each input channel with its corresponding kernel (see Figure 2.11). Since this method returns the convolution and the summation of the results, it can be applied to the channel summation as described in the Figure 3.5.

The method has been tested, just like the kernel tiling, using the weights of the MNIST network, as shown in Figure 2.12.

One of the disadvantages of this method is that most of the pixels in the output are useless and must be discarded.

The third and most efficient method of parallelising the convolutional layer in the 4f system is mixed tiling. Mixed tiling involves combining kernel tiling and channel tiling into a single method, ensuring full parallelism of the entire convolutional layer. If the device resolution permits, the entire convolutional layer can be executed in one inference through the 4f system. This approach guarantees that the convolution of all input channels with their corresponding kernels is completed, and the results are summed across the output channels. However, in practice, the resolution of the modulators might not allow all channels to be tiled.

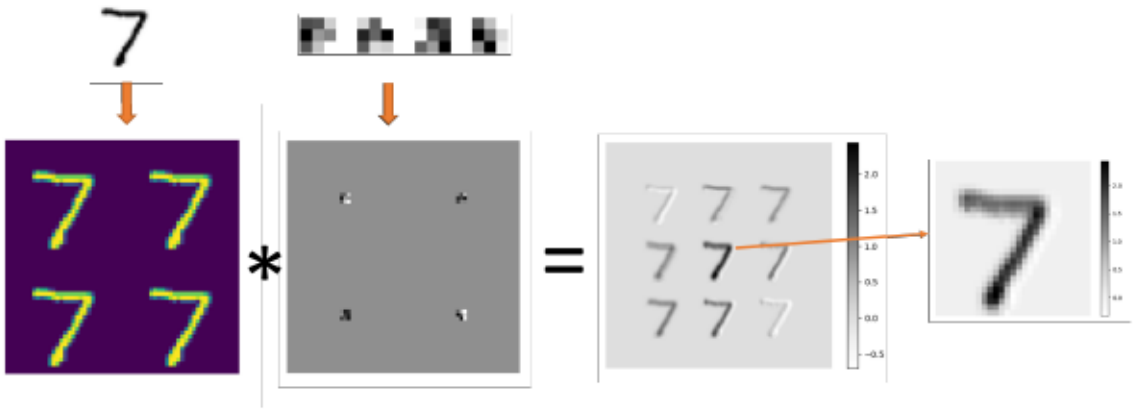


Fig. 2.12 **Experimental results of the channel tiling technique applied to four input channels of MNIST digit "7" as input and four kernels corresponding to one output channel extracted from the model also trained on MNIST.** The input image is padded and tiled four times (each corresponding to the kernel). Kernels are also padded and tiled into one kernel block. Convolution has been performed with both blocks, and the image in the middle of the output is cropped out. The cropped-out image is the sum of the convolutions of inputs with 4 kernels.

In the mix tiling method, the inputs and kernels are padded to $(M+N-1)\times(M+N-1)$ and tiled horizontally. Similarly, the kernels are padded to $(M+N-1)\times(M+N-1)$ and tiled in both x and y dimensions, where each row corresponds to the output channel. The output block, similar to the channel tiling, has invalid regions of unnecessary convolutions. The valid outputs are present in the middle of each row of the output block.

The mix tiling method has been tested with MNIST weights, as shown in the Figure 2.14. Unlike the real-life scenario, the same input image was used four times to emulate different channels, similar to what has been used in channel tiling. Kernels are tiled in a manner similar to kernel tiling; however, in this case, the rows correspond to the distinct output channels. After the input block is padded to the same resolution as the kernel block, the output's middle column is extracted as the values of the new output feature map tensor.

2.8.5 Optical Transformers

Regardless of the ViTs being a new field in Deep Learning, recently, Xu *et al.* [5] have employed a free-space optical setup to accelerate ViTs, introducing an innovative Optronic Vision Transformer (OPViT) architecture. This setup, as illustrated in Figure 2.15, combines the principles of optical computing with the Transformer model. It is a dedicated setup designed specifically for the implementation of ViTs using optical technology.

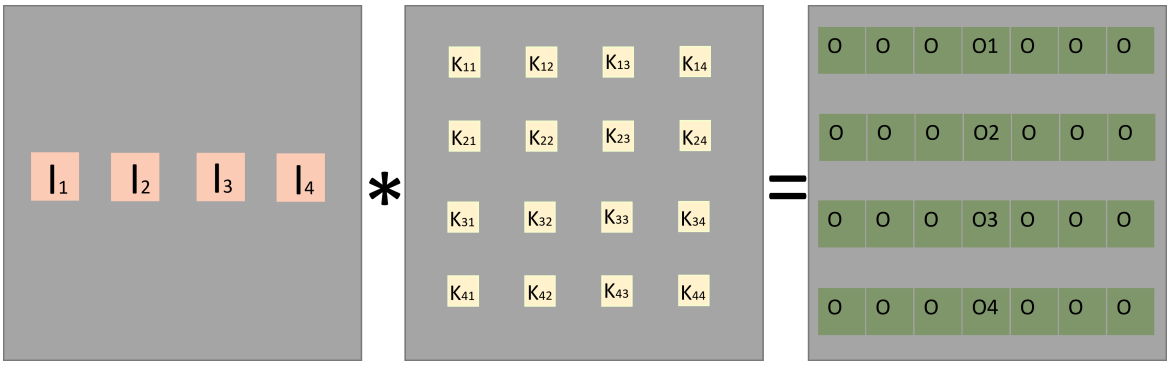


Fig. 2.13 **Implementation of the convolutional layer in optics using mix tiling.** All input channels are tiled horizontally in the middle of the input block. Kernels corresponding to the same output channel are tiled horizontally. In the output block, the output matrices in the middle of each row correspond to the output of the convolutional layer, which is the sum of the convolution of the input channel with its corresponding kernel and summation.

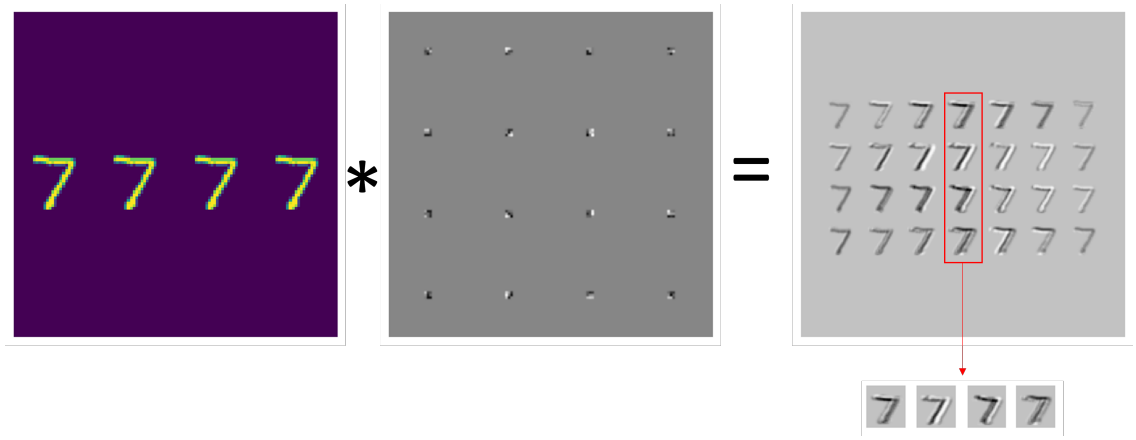


Fig. 2.14 **Experimental results of the mix tiling technique applied to a four-input-channel, four-output-channel convolutional layer, using the MNIST digit "7" as input.** Sixteen 2D kernels were extracted from a custom convolutional neural network trained on the MNIST dataset and were tiled in the kernel block to produce four output channels. The resulting block contains 28 output images, with only four images in the middle column falling within the valid region, which can be cropped and used as an output of the convolutional layer.

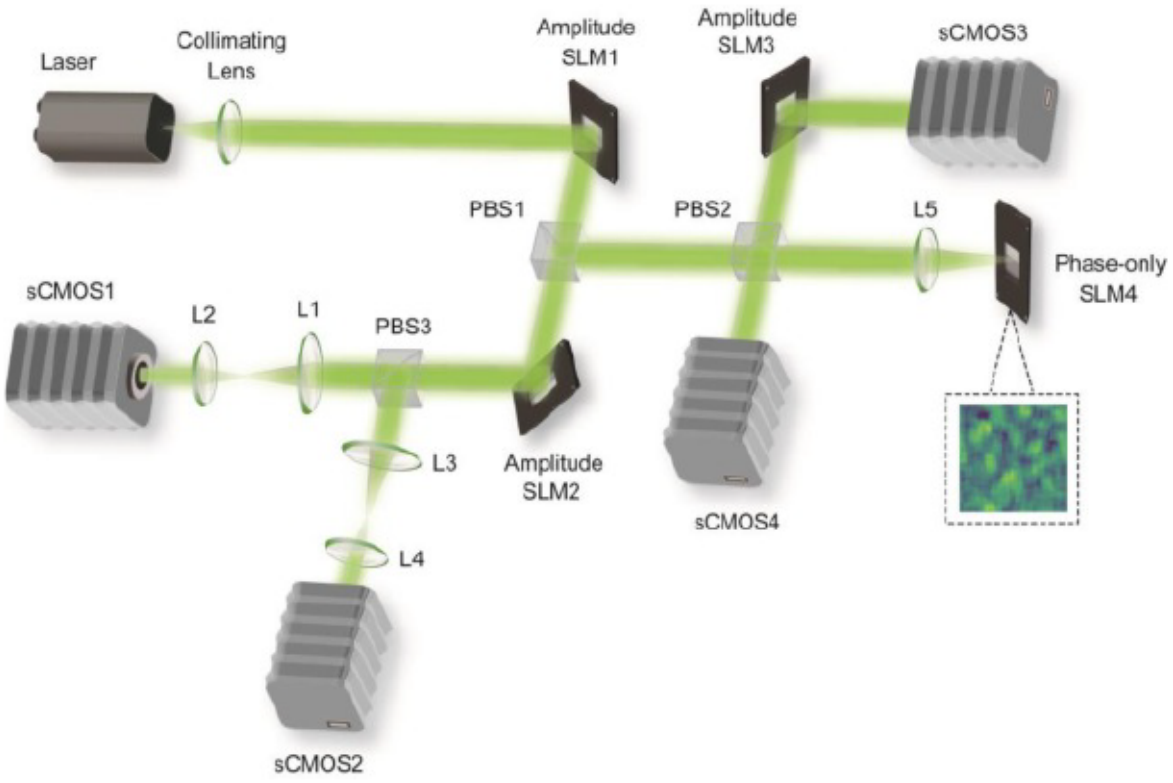


Fig. 2.15 **Optical Vision Transformer hardware architecture as per Xu *et al.* [5].** The left part of the setup performs the Transformer path, where SLMs are used for the matrix multiplication and lenses to simulate the summation while demagnifying the ray. The right side of the setup is a standard 4f system used for the classification.

The optical setup in Figure 2.15 consists of two main parts: the Transformer Encoder and the Classifier. The process within the encoder begins with SLM1, where the input image is loaded. This input image is divided into smaller patches, which are then tiled and displayed on SLM1. This tiling is crucial for enabling parallel processing during the matrix multiplication operations that follow. Concurrently, the corresponding weight matrices W_q,W_k, and W_v are loaded onto SLM2. After the input image patches are displayed on SLM1 and the weight matrices on SLM2, the light passes through a series of lenses (L1 and L2), which focus and sum the light. This optical summation is equivalent to the matrix multiplication needed to compute the queries, keys, and values. The result is captured by a sensor (sCMOS1) as Q, K, and V.

Similar to a regular attention layer, Q and K are used to calculate the attention scores. The matrices Q and K are multiplied, and the resulting product is passed through another set of lenses (L3 and L4) for summation. The attention-weighted values are then processed using SLM3, where a trained matrix is applied to prepare the encoded image for the final

classification stage. Finally, the authors used the 4f system convolution for the classification head, which happens in the SLM4 and is read with the CMOS4 camera.

From a deep learning perspective, the OPViT integrates the Transformer model's self-attention mechanism into an optical framework. The architecture uses only one layer of self-attention with single-head attention, which is significant because Transformers typically rely on multi-head attention for capturing diverse aspects of the input data. For classification, the architecture requires only two to three layers of optical convolution, depending on the complexity of the dataset.

The OPViT was evaluated on two standard image classification benchmarks: the MNIST and Fashion-MNIST datasets. It achieved accuracy rates of 98.70% on MNIST and 88.93% on Fashion-MNIST. Although the results are impressively good, the MNIST dataset can be classified with SVM up to 94.005%, with MLP of six layers up to 98.85% and two-layered CNN with test accuracy of 99.31% [92]. Unfortunately, there were no attention score visualisations to prove the concept of attention learning.

Nevertheless, the authors demonstrated the first-ever experimental implementation of the attention mechanism in free-space optics. They estimated the system's latency, highlighting that the primary delays arise from signal transduction times, such as the loading time of SLMs and the delay in image detection by high-speed cameras. Considering these factors, they suggest that the realistic clock speed for the OPViT system would be approximately 70 Hz.

The authors emphasise the energy efficiency of the OPViT system, stating that it achieves 1.76×10^{12} FLOPs/J, which is significantly more efficient than current digital devices, typically around 10^9 FLOPs/J.

2.9 Datasets

2.9.1 CIFAR-100

The CIFAR-100 (Canadian Institute For Advanced Research) [59] dataset is used to evaluate the classification models in this study. The dataset is a well-established, widely-used resource for machine learning and computer vision research. It comprises a collection of 60,000 colour images evenly distributed across 100 distinct classes. E ach of these classes represents different objects, animals, and types of vehicles and plants, among others (See Figure 2.16).

Every class in the CIFAR-100 dataset contains 600 individual images with a low resolution of 32×32. Furthermore, the dataset is split into a training set of 50,000 images (500 per

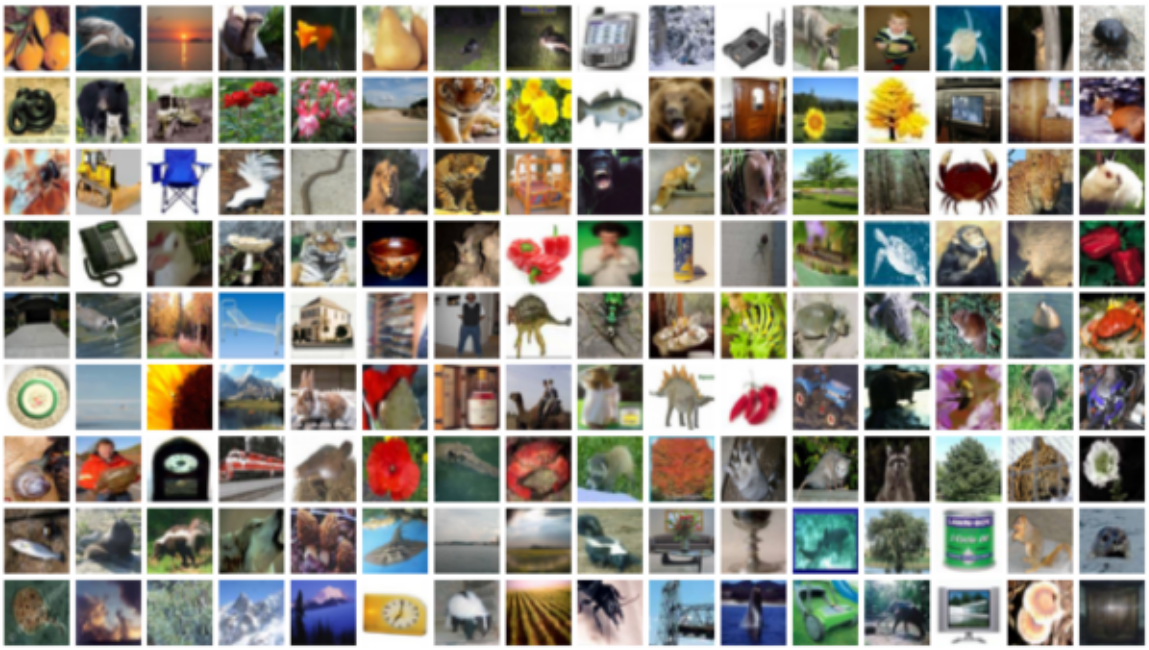


Fig. 2.16 **Subset of CIFAR-100 dataset.** Coloured images of varios objects, animals, vehicles and other 100 classes, with the resolution of 32×32

class), and a test set of 10,000 images (100 per class), providing a robust framework for both model training and evaluation [27].

An additional unique aspect of the CIFAR-100 dataset is its hierarchical label structure. The 100 classes (referred to as 'fine' classes) are grouped into 20 'coarse' classes or super-classes. Each coarse class contains five fine classes. For example, a 'coarse' class may be 'aquatic mammals', within which would exist 'fine' classes such as 'beavers', 'dolphins', 'otters', 'seals', and 'whales'. The list of coarse classes and fine classes is shown in Table 2.4. This offers researchers the ability to examine model performance on different levels of class granularity. On the other hand, the similarity of fine classes under the same coarse class in CIFAR-100 makes it harder to train.

Although CIFAR-100 is a challenging dataset for model evaluation, this was not the only reason it was chosen for this research. The design of the FatNet models, as described in Section 3.2.1, makes them more efficient for datasets with a large number of classes, such as the 100 classes in case of CIFAR-100. Additionally, FatNet models have the potential to work more efficiently with higher-resolution images. However, simulating light propagation requires a significant amount of GPU memory. Moreover, FatNets are not optimised for GPU acceleration, resulting in longer training times compared to the original networks. This is why ImageNet, despite being an obvious choice for FatNet, was not used in our experiments.

Table 2.4 Classes and Superclasses in CIFAR-100 dataset.

Superclass	**Classes**
aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household furniture	bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large carnivores	bear, leopard, lion, tiger, wolf
large man-made outdoor things	bridge, castle, house, road, skyscraper
large natural outdoor scenes	cloud, forest, mountain, plain, sea
large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

Table 2.5 Mizusawa *et al.*baseline performance of CIFAR-100 on different models

VGG11	VGG19	ResNet-20	ResNet110	ViT
67.10	70.21	64.09	66.71	60.84

In contrast, CIFAR-100 is a low-resolution dataset with a relatively large number of classes, making it an ideal choice for this work.

2.9.2 Classification of CIFAR-100 dataset

The CIFAR-100 dataset is one of several key benchmarks used to evaluate the performance of neural networks om image classification tasks. Comprising images across 100 categories, each at a resolution of 32×32 and including 600 instances per class, this dataset offers a comprehensive test of the flexibility and robustness of various deep learning models. An in-depth overview of the dataset is discussed in Section 2.9.1. Given its enhanced complexity and larger variety of classes compared to CIFAR-10, CIFAR-100 serves as an excellent benchmark in image classification technologies. This section describes and analyses the performance of previously published neural network models on CIFAR-100, providing insights into the benefits and limitations of the state-of-the-art methods.

To ensure a fair evaluation consistent with the methodologies discussed in this thesis, the analysis focuses primarily on traditional training approaches, rather than on fine-tuning or the application of transfer learning techniques.

In recent years, significant strides have been made to improve the classification accuracy of models trained on the CIFAR-100 dataset by employing advanced augmentation and regularisation techniques. These strategies aim to mitigate common issues such as overfitting and the vanishing gradient problem, particularly in deep convolutional neural networks.

Mizusawa *et al.*'s [93] research on interlayer augmentation demonstrates a sophisticated approach to data augmentation that manipulates data between network layers. This method, involving techniques known as Batch Generalization (BG) and Random Batch Generalization (RBG), was applied across various network architectures ResNet, VGG and ViT leading to an average improvement in CIFAR-100 classification accuracy by up to 1.07% for RBG and 0.30% for BG compared to baseline models. For the purpose of this project, we are interested in the baseline models, performance of which are given in the table below

The authors discussed the use of several augmentation techniques, such as cutmix, mixup, shift, random flips, and feature space augmentation. From Table 2.5, it can be seen that the improvements with the larger models are notable compared to smaller models, but the ViT

performed significantly worse than the CNNs. This can be simply justified, as discussed earlier, because ViT is not powerful on small datasets but is extremely efficient when trained on larger datasets and fine-tuned on small datasets like CIFAR-100.

Shah *et al.*'s [94] modification of the ResNet-110 architecture, which integrates Exponential Linear Units (ELU) [95] instead of the conventional ReLU activation functions, has shown promising results. This modification mitigates the vanishing gradient problem and the diminishing feature reuse, issues that are pronounced in very deep convolutional neural networks. Shah's network allows small negative outputs and shifts mean activations closer to zero, thereby reducing the bias shift commonly observed with ReLUs and speeding up the learning process. The authors have demonstrated that by replacing ReLU with ELU, the network benefits from faster learning speeds and improved accuracy as the network depth increases. This modification led to an improvement in test accuracy, achieving a new state-of-the-art performance with a 73.45% error rate on CIFAR-100, surpassing the traditional ResNet-110's 72.77% test accuracy.

In their research, Chen *et al.* [96] also uses the CIFAR-100 dataset to evaluate their proposed method, CCPrune. This work aims to build on previous methods, such as those introduced by Li *et al.* [97], which primarily focused on evaluating the importance of channels based on single-layer data. By integrating both the convolution layer weights and the Batch Normalisation layer scaling factors, CCPrune allows for a more comprehensive assessment of channel importance, ensuring that interdependencies between layers are considered in the pruning process.

Their application of CCPrune on CIFAR-100 using the Resnet-50 architecture demonstrated its effectiveness in significantly reducing computational demands—specifically, a reduction in FLOPs by 65.25%—while maintaining nearly the same level of accuracy, with only a slight decrease from 77.72% to 77.74%. In the same work by pruning the ResNet-20, the FLOPs were reduced by 53.65%, while accuracy dropped from 68.67% to 67.57%.

2.9.3 Oxford-IIIT Pet

The Oxford-IIIT Pet dataset [98, 28], developed by the Visual Geometry Group, contains 7,359 images of pets (cats and dogs) spread across 37 different breed categories, with each category roughly comprising 200 images. It was created for fine-grained image classification, segmentation and object detection tasks, featuring images of cats and dogs. Its primary aim is to facilitate algorithms capable of distinguishing between different breeds of cats and dogs.

The dataset is characterised by its diversity, featuring significant variations in scale, pose, scene and lighting conditions across the images. Each image in the dataset is labelled by breed identification, head region of interest (ROI), and the segmentation mask (See Figure 2.17).

Fig. 2.17 **Subset of the Oxford IIIT pets dataset with ground truth segmentation masks.** This is one of the three types of labels that come with the dataset, including segmentation mask, ROI (bounding box), and breed classes.

Given that we used the dataset in our research for segmentation purposes, the breed classes provided by the dataset are not our main concern. Instead, our priority lies in the segmentation tasks that distinguish pets from their backgrounds.

This dataset was chosen for several reasons. Its diversity facilitates the evaluation of segmentation models across a broad range of object shapes, sizes, and textures. Additionally, the dataset provides a good balance between different breeds and includes a wide range of backgrounds, making it ideal for testing model generalisation. The Oxford-IIIT Pet dataset is widely used in the research community, offering benchmark results that can be compared across studies (See Section 2.9.5).

While more complex datasets such as Pascal VOC or COCO could be employed, the Oxford-IIIT Pet dataset served as the primary evaluation method in this work before training on a real-life dataset: HeLa cancer cell nucleus segmentation. Unlike Pascal VOC and COCO, which are larger and more complex, the Oxford-IIIT Pet dataset is more manageable in terms of memory and computational requirements for the FatNet models and is sufficient to prove the concept before moving to HeLa nucleus segmentation.

2.9.4 HeLa cells

Since the Oxford-IIIT Pet dataset was already being used for the evaluation of the segmentation models in this work, the second dataset was chosen such that it would have a direct real-life application.

In 1951, biomedical research discovered the first immortal human cell line sourced from the cervical cancer cells of Henrietta Lacks. This cell line, known as "HeLa" today, exhibited an unprecedented ability to grow and divide outside the human body. These cells have been extensively used in a range of studies, including those on cancer [99–101], toxoplasmosis [102–104], radiation [105–107] and AIDS [108–110].

Within cancer research, a focal point is the examination of cell distribution, form, and detailed traits, including aspects like the nuclear envelope (NE) and plasma membrane.

This particular HeLa cells dataset used in this research was prepared and embedded in Durcupan for observation with serial block face scanning electron microscopy (SBF SEM) following the protocol of the National Center for Microscopy and Imaging Research (NCMIR). The images were acquired using a Gatan 3View2XP SBF SEM attached to a Zeiss Sigma VP SEM. The voxel size was $10 \times 10 \times 50$ nm with intensity values in the range of 0-255. In total, 518 slices of 8192×8192 pixels were acquired [111].

This work uses the $2000 \times 2000 \times 300$ [112] voxel region of interest (ROI) containing one cell near the centre cropped from the larger images. The ground truth used for training was acquired from the dataset by Karabag *et al.* [32]. The ground truth contains the segmentation of the central cell's nucleus, ignoring the adjacent cells' nuclei, as shown in Figure 2.18. However, in their work, Karabag *et al.* [8] has demonstrated that the U-Net managed to learn the segmentation of the nuclei of the 8192×8192 field even when trained with the ROI where the nuclei of the adjacent cells were treated as the background.

2.9.5 Segmentation of HeLa Cells and Oxford-IIIT Pet

This section consists of an analysis and review of the previous work done in the training and evaluation of the HeLa Cells and Oxford-IIIT Pet datasets. For a fair comparison with this work, it will be more focused on U-Net based methods.

One of the studies by Dippen et al. [113] examines the effectiveness of different pretraining and initialisation methods for encoder-decoder architectures in image segmentation tasks, emphasising the U-Net architecture and using the Oxford-IIIT Pet dataset for evaluation. When randomly initialised, their U-Net achieved an average Dice score of 88.2% on 20% of the dataset. Meanwhile, their best performance employing the ConRec method elevated the Dice score to 90.0%.

Fig. 2.18 **Example of six slices from the Ground Truth of HeLa Cells dataset.** The slices are displayed in increasing order from left to right. The nucleus is vivid in the middle slices, but gets smaller and even splits in the shallowest and deepest slices. Arrows are added for demonstrative purposes: the green arrow points to the manually labelled nucleus, while the red arrows point to the adjacent cells, where the nucleus was ignored by the labeller.

One way to improve the performance of U-Net involves replacing the encoder path of the U-Net with pre-trained classifier models, such as InceptionV3, DenseNet, ResNet34, and others, as described by Sundarrajan *et al.* [114]. After training a 5-stage U-Net with the Oxford-IIIT Pet dataset, the Intersection over Union (IoU) and Dice Score achieved were 33.3% and 46.4%, respectively. The authors used a batch size of 8 and trained the model for 50 epochs.

After replacing the encoder with pre-trained models, significant improvements were observed. For example, U-Net+VGG-16 achieved an IoU of 89.4% and a Dice score of 94.2%. Similarly, U-Net+Inception V3 reached an IoU of 91.6% and a Dice score of 91.5%. When the U-Net's encoder was replaced with ResNet34, the performance further improved, achieving an IoU of 91.0% and a Dice score of 95.1%. U-Net+Densenet performed exceptionally well, with an IoU of 91.5% and a Dice score of 95.3%. Lastly, U-Net+MobilenetV2 also showed considerable enhancement, with an IoU of 89.6% and a Dice score of 92.3%.

These results illustrate the substantial gains in segmentation performance achieved by leveraging pre-trained models as encoders in the U-Net architecture. This demonstrates that the encoder path of the U-Net functions as the classifier, significantly enhancing the model's ability to accurately segment images when combined with the expanding path and skip connections to maintain localisation.

While previous models for semantic segmentation, such as U-Net, relied on deep encoder-decoder architectures, the ICNet model proposed by Zhao et al. [115] introduced a more efficient approach for real-time applications. ICNet uses a multi-resolution branch structure

with Cascade Feature Fusion (CFF) and Cascade Label Guidance (CLG) to achieve high-quality segmentation with reduced computational cost.

Building on ICNet, Edwards and El-Sharkawy [116] introduced uICNet, leveraging depthwise separable convolutions from MobileNet for enhanced efficiency. They evaluated uICNet on the Oxford-IIIT Pet dataset by resizing the images to 512×512 pixels and applying data augmentation techniques such as random flips. uICNet achieved mean intersection over union (mIoU) scores of 74.53% and 74.78% for its two versions, while reducing model size by 4% compared to ICNet. In comparison, ICNet achieved an mIoU of 75.12% on the Oxford-IIIT Pet dataset. It should also be mentioned that the inference speed for both uICNet and ICNet is identical.

The Scale Equivariant U-Net (SEU-Net) introduced by Sangalli *et al.* [31] represents a significant advancement in segmentation tasks, particularly for datasets such as Oxford-IIIT Pet and DIC-C2DH-HeLa cells. SEU-Net enhances the traditional U-Net architecture by incorporating scale equivariance, which is achieved through semigroup cross-correlations and carefully designed subsampling and upsampling layers. This approach ensures that the network generalises well to images containing objects at different scales. Additionally, SEU-Net employs scale-dropout during training, a technique that enhances robustness by randomly dropping scales.

In experiments, SEU-Net outperformed both traditional U-Nets and scale-equivariant residual networks (SResNet) at different scales of the inputs and exhibited identical performance at a scale of one, achieving an IoU of approximately 77% on the Oxford-IIIT Pet dataset. Furthermore, SEU-Net demonstrated improved performance in the DIC-C2DH-HeLa cell segmentation tasks, highlighting its versatility and effectiveness in handling scale variations across different datasets.

Originally, U-Net was developed for biomedical image segmentation, and in the seminal work by Ronneberger *et al.* [13], the DIC-HeLa cells dataset was used to evaluate the model, where U-Net achieved a 77.56% IoU. This performance was significantly better than other algorithms at the time.

In contrast, the U-Net model developed by Sangalli *et al.* [31] achieved approximately 82.8% IoU on the DIC-HeLa cells dataset and slightly lower at around 81% IoU with SEU-Net. Similar to the results with the Oxford Pets dataset, SEU-Net performed better than U-Net at different scales of the image.

The dataset used in this thesis is the version evaluated in the work of Karabag et al. [8], where it achieved an IoU of 51.38% for all slices of the region of interest (ROI) and 97.12% for the middle 150-200 slices of the ROI. Since the nucleus is either very small or non-existent in the shallowest and deepest slices of the ROI, the metrics become inaccurate

and irrelevant. Hence, the best performance metrics are derived from the middle-range slices, which were also used in this thesis.

2.10 Gradient Accumulation

Since the simulation of the optics and larger kernel resolutions requires a lot of memory, training the networks with large batch sizes becomes impossible or hard. In this work to solve the issue, we have turned to the gradient accumulation technique introduced by Hermans *et al.*[117].

This approach is designed to overcome the constraints of GPU memory when training deep neural networks. It accumulates the gradients over multiple mini-batches before updating the model's parameters, reducing memory consumption during training while maintaining performance.

Suppose we would like to train with the desired batch size B, but we can only train with the batch size of B/N, due to the memory requirements. We can perform N numbers of forward and backward passes on each B/N mini-batch while accumulating the gradients.

$$\Delta\theta^{(t)} = \sum_{i=1}^{N} \Delta\theta_i^{(t)} \tag{2.15}$$

After the N iterations, we can perform the gradient step and update the weights.

$$\theta^{(t+1)} = \theta^{(t)} - \alpha\Delta\theta^{(t)} \tag{2.16}$$

2.11 Measurement of performance

This section provides an overview of the metrics used to evaluate all models trained in this work, covering both classification and segmentation tasks across all chapters.

Classification tasks have been evaluated using only the top-1 accuracy, referred to simply as accuracy throughout this thesis.

Accuracy is a standard metric for evaluating classification models. It is defined as the ratio of correctly predicted data points to the total number of data points. Mathematically, Accuracy is expressed as:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{2.17}$$

where TP is the number of true positives (correctly predicted positive instances), TN is the number of true negatives (correctly predicted negative instances), FP is the number of false positives (incorrectly predicted positive instances), and FN is the number of false negatives (incorrectly predicted negative instances)

Segmentation models were evaluated using three metrics: pixel-wise accuracy, dice score and mIoU (Mean Intersection over Union).

Pixel-wise accuracy is similar to the accuracy used for the classification, except it measures the ratio of correctly predicted pixels to the total number of pixels, rather than instances. Mathematically expressed as mentioned in the Formula 2.17.

The Dice Score is a widely used metric for segmentation models; it is equivalent to the classification's F1 score. It measures the overlap between the predicted segmentation and the ground truth. It is particularly beneficial when the target regions are small, and the cost of missing these regions is high. The Dice Score is defined as:

$$DiceScore = \frac{2|A \cap B|}{|A| + |B|} \tag{2.18}$$

where A is the set of pixels in the predicted segmentation mask, B is the set of pixels in the ground truth.

In terms of TP, FP and FN, it is expressed as follows:

$$DiceScore = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{2.19}$$

where TP is the number of true positives (correctly predicted positive instances), FP is the number of false positives (incorrectly predicted positive instances), and FN is the number of false negatives (incorrectly predicted negative instances).

Intersection over Union, also known as the Jaccard Index, is another important metric for evaluating segmentation models. It measures the overlap between the predicted segmentation and the ground truth relative to their union. Unlike the Dice score, IoU does not take the overlap twice, which makes it more interpretable and preferred when the task requires a straightforward measure of overlap. It is also more strict for over-segmentation or under-segmentation, which can be crucial in applications where precision is important.

IoU is defined as:

$$IoU = \frac{|A \cap B|}{|A \cup B|} \tag{2.20}$$

where A is the set of pixels in the predicted segmentation mask, B is the set of pixels in the ground truth.

In terms of TP, FP and FN, IoU is expressed as follows:

$$IoU = \frac{TP}{TP + FP + FN} \tag{2.21}$$

2.12 Conclusion

The literature review began by introducing optical computing and the 4f system, examining Fourier optics, the Fourier transform, and Fourier convolution. These concepts were connected to demonstrate how the 4f system performs convolution operations. Following this, the review explored CNNs and their applications in image classification by examining state-of-the-art networks in chronological order. All these networks, from early designs like LeNet to modern architectures like ResNet, were found to share a cone-shaped structure, aligning with the problem statement 1.1. Additionally, the application of CNNs for image segmentation was discussed, where encoder-decoder architectures like U-Net also exhibited cone-shaped designs for the encoding process.

The literature then shifted to high-resolution training. Agrawal and Mital [6] demonstrated that high-resolution training can improve model performance. Peng *et al.* [66] noted that barrel-shaped networks are more efficient for segmentation tasks, as they effectively incorporate localisation and classification. Furthermore, Ding *et al.* [36] showed that high-resolution kernels in CNNs can expand the effective receptive field, allowing CNNs to match the performance of Vision Transformers, which achieve large receptive fields through attention mechanisms. Despite these promising insights, no studies have integrated high-resolution training or barrel-shaped architectures with optical accelerators like the 4f system.

The next section focused on optical accelerator hardware. Passive 4f optical accelerators were shown to be restricted to fixed non-editable kernels, limiting their use to single-layer implementations. While active 4f systems can process multiple layers, they rely on electronic non-linear activations, making the read-out speed a major bottleneck. To address these limitations, researchers have taken advantage the high-resolution capabilities of the 4f system to parallelise convolution operations. However, even though the high-resolution capabilities of the 4f system enable parallelism, as discussed in Section 2.8.4, no studies have applied high-resolution training to make full use of 4f system's spatial bandwidth.

The review also examined attention mechanisms, transformers, vision transformers, and their various adaptations, such as those for the CIFAR-100 dataset. Unlike CNNs, ViTs rely on attention mechanisms instead of convolutional hierarchies and achieve a larger effective receptive field. However, optical implementations of ViTs require entirely new setups, with

no existing work exploring the feasibility of using a single optical device, like the 4f system, to implement both transformer and CNN layers.

Finally, the chapter provided an overview of the datasets used in this work, including CIFAR-100, Oxford IIIT Pets, and HeLa cells, as well as a review of prior benchmarks. Key techniques such as gradient accumulation, used in this work and the metrics for evaluation were also discussed.

In summary, the review identifies following gaps in knowledge which align with the problem statement 1.1 and objectives of this work:

- **Lack of research on barrel-shaped architectures in optical systems:** Despite advantages of high-resolutional training for higher effective receptive field, and their compatibility with 4f systems, these architectures remain unexplored in optical accelerators.

- **Hardware bottlenecks in optical accelerators:** The reliance on electronic non-linear activations, modulator frequency limitations, and camera readouts constrains scalability and limits the efficiency of optical accelerators. Hence there needs to be a solution to limit the number of optics-to-electronics conversions, which is partially solved using tiling methods, but not through experiments involving high-resolution training.

- **Lack of versatility of 4f systems:** Current implementations of optical ViTs require separate setups, raising the question of whether a single optical device can handle both transformers and CNNs and whether the same 4f system can be used to perform inference on the linear layers.

Chapter 3

FatNet for Classification

Overview

This chapter focuses on the adaptation of the convolutional neural networks used for the image classification for the 4f free-space optical accelerator.

It provides a detailed description of the methods developed. The chapter includes methods for the FatNet conversion, optical simulator, and the custom optical convolutional layer (OptNN).

Note: This chapter is related to the following publication: **R. Ibadulla**, T. M. Chen, and C. C. Reyes-Aldasoro, "FatNet: High-Resolution Kernels for Classification Using Fully Convolutional Optical Neural Networks," AI, vol. 4, no. 2, pp. 361–374, Apr. 2023, doi: 10.3390/ai4020018 [118]

This chapter provides additional experiments with other networks like VGG-16 and AlexNet.

3.1 Introduction

Image classification is a fundamental and important task in the field of computer vision, which involves solving the problem of identifying which set of categories or classes an image belongs to. With the rise of deep learning, convolutional neural networks became a standard solution for complex computer vision tasks [119, 59, 27, 3, 120–125] including image classification. In CNNs, each neuron in a convolutional layer is only connected to a small, localised region of the input image, known as the receptive field, rather than being fully connected to every neuron in the following layer. This localised connectivity enables CNNs to efficiently capture spatial hierarchies and patterns, such as edges or textures, which

are crucial for image classification tasks. It is known that the fully connected layers can also perform image classification tasks but with the cost of potential overfitting and lower computational efficiency. CNNs, on the other hand, are specifically designed to process pixel data with convolutional layers that automatically and adaptively learn spatial hierarchies of features and capture patterns from input images without the need for manual feature extraction. Initial image classification methods and algorithms were employed to extract the features from images, or alternatively, these features were extracted manually. After this feature extraction phase, those features were fit into classifiers such as Support Vector Machines (SVMs), k-nearest Neighbors (k-NN), Decision Trees or a perceptron. However, CNNs' hierarchical structure allows shallow layers to learn basic features and deeper layers to capture more complex patterns, making them highly effective for distinguishing between intricate image classes.

Furthermore, due to weight sharing in the convolutional layers, CNNs have fewer parameters than fully connected networks, which aids in reducing the computational cost and risk of overfitting. Additionally, pooling layers in CNNs introduce spatial invariance to changes in position, allowing the network to recognise objects regardless of their position in the image. Invariances to other transformations, such as rotation and scale, can be achieved through additional techniques, such as data augmentation or specific modifications to the architecture [31, 126]. Consequently, these advantages make CNNs the preferred choice for many image classification tasks, often achieving state-of-the-art performance [3, 2, 4, 127–129].

One of the nuances of the convolutional neural networks for classification is their cone shape. This design is important because it allows the network to recognise and learn patterns from input images, making the model more suitable for handling spatial data. However, this shape is the legacy of the early classification methods, as the final layers of CNNs extract high-level features, capturing complex patterns and abstractions that are well-suited for the dense classifier. Moreover, these networks usually contain small 3×3 kernels, which, when combined with the cone structure, make them well-suited for training on modern hardware like CPUs and GPUs. On the other hand, optical accelerators, as previously mentioned, can handle convolution operations with the constant speed, regardless of input size or kernel dimensions. Nevertheless, the bottleneck of such a network is the multiple conversions from optics to electronics in the camera. This suggests that a network design with fewer channels, but larger kernels might work better for optical accelerators. Although, initially, it's difficult to determine whether this network will be able to compete with a standard cone-shaped CNN, or if it will overfit and be inefficient. If FatNet is used, it can be assured that the number of pixels and number of weights per layer will be preserved. However, there are still questions: whether converting the off-shelf networks into the FatNet is efficient? And can any network

with larger kernels and larger feature maps in the barrel shape can compete with the off-shelf network converted to FatNet?

3.2 Methods

3.2.1 FatNet conversion

The main idea behind the FatNet is to convert any classification network from a cone shape into a barrel shape.

Cone-shaped neural networks have demonstrated success since the first introduction of LeNet. There are several reasons for having cone-shaped architecture. By filtering the feature maps using convolution operations followed by activation functions and pooling operations, the features from the image can be learned and extracted to end up with the feature maps or vectors essential for the current classification problem. Those feature maps can then be flattened into a vector and fit into the classifier. This can be any classifier, like MLP or KNN. Same as the convolutional part, it can be any feature extractor. But having the convolutional feature extraction and fully connected classifier all stacked into one network allows the training of the filter for the feature extraction and weights of the classifier simultaneously. Moreover, the convolutional feature extractor of these networks can be used for transfer learning, as they have the full capability of extracting the relevant features of common images.

The question arises why the research is following the traditional path of feature extraction first and classification of the extracted features if it's possible to train fully connected networks. Today, the Vision Transformers have proven that there is absolutely no point in following the traditional classification methods of combination of feature extraction and classification [21]. However, the main advantage of using convolutions followed by the pooling operations is the training speed since the network gets lighter after each feature extraction and pooling and ends up with very low-resolution feature maps. The final dense layers, in this case fit with the relatively short vector of the extracted features. However, these methods became standard due to the dominance of electronic computing. Unlike in electronics, larger resolutions for inputs and kernels in the free space optics do not affect the speed of inference, which makes it essential to explore new architectures that are compatible with optics. Due to the dominance of electronic computing, the research of larger resolution kernels has been lagging as it seemed useless.

This section focuses on converting cone-shaped architectures into barrel-shaped architectures. The output of such conversion is the network with high-resolution feature maps, with

only a few pooling operations and high-resolution kernels, sometimes the size of the feature maps, hence called FatNet (Layers are Fat).

Through the use of larger kernel sizes and feature maps in the classifier convolutional neural network (CNN), the efficiency of free-space optics is maximised. While it is true that higher resolutions might lead to overfitting, our method maintains the same number of trainable parameters as the original network, mitigating this issue. The following core rules have been established for converting any classifier into a FatNet:

1. The FatNet should preserve the same number of layers as the original network to keep the same number of non-linear activation functions. Since each layer's non-linear activation function enhances the network's ability to capture complex patterns, it is crucial to preserve the number of layers. Non-linear activations enable the network to model complex, non-linear relationships in the data, thereby supporting complex pattern recognition. By retaining the same number of non-linearities, FatNet maintains functional equivalence and ensures performance comparability.

2. The FatNet should keep precisely the same architecture as the original network on the shallow layers until the shape of the feature maps pools down to the shape where the number of elements of the feature map is less than or equal to the number of classes. This ensures that the FatNet retains the original network's early layer structure to capture essential low-level features accurately before necessary conversion occurs. A key concept behind FatNet is to eliminate linear layers and transform the network into a fully convolutional neural network to facilitate inference on the 4f system. Consequently, it is essential to have an output tensor, where the number of elements equals to the number of classes to represent each class. Ideally the output tensor should have one channel, with a resolution that matches $\overline{\text{number of channels}}$ $\overline{\text{number of channels}}$. Before the feature maps of the original cone-shaped CNN are pooled down below this desired resolution, the feature map resolutions remain high, making it unnecessary to convert those layers into their FatNet equivalents.

3. FatNet has a similar total number of pixels of the feature maps at the output of each layer as the original networks. Hence, since the feature maps' shape stays constant and does not use pooling, the new number of output channels needs to be calculated, which will be less than for the original network. As the resolution of features maps in FatNet remains high and is not downscaled, the number of features (pixels) in each layer increases. In networks such as MLPs, this would directly impact the number of trainable parameters. However, in CNNs, the kernel size is not directly linked to the number of features in the features map. Although reducing the number of output

channels is crucial part of the FatNet conversion, as it minimises the number of 2D convolution operations, the question arises whether this reduction should be done arbitrarily or by preserving the number of features in the feature map. Hypothetically, maintaining the number of pixels is essential for the model to retain similar feature extraction capabilities as the original network. This hypothesis is later substantiated in the experiments conducted in *Intuitive Fat-U-Nets* 4.2.1.

4. FatNet has a similar number of trainable parameters per layer as the original network. Since we have reduced the number of output channels based on the third rule, the number of trainable parameters has also been reduced. Hence, a new kernel size needs to be calculated based on the number of output channels. If the number of parameters is reduced too drastically, the network may lose its ability to generalise effectively or capture complex features, potentially degrading performance. By recalculating the kernel size to balance the reduced output channels, FatNet can achieve comparable accuracy, ensuring it remains as effective as the original network. However, increasing the number of parameters beyond that of the original network could lead to overfitting. Since the original network is proven to generalise well, the kernel size should be adjusted only enough to maintain a similar number of trainable parameters.

The previous rules are core intuitive rules. However, experimentally it has been concluded that the next two rules should also be considered when building a FatNet:

5. If the number of input channels and output channels is equal in the original network, that equality should be preserved for the FatNet too. This is done for the modular CNNs, such as VGG or ResNets, where the networks consist of convolutional blocks. Each block in such networks consists of a sequence of convolutional layers with the same number of channels. This rule of FatNet conversion helps preserve the modular structure of the network. When the rule was not initially applied, ResNet encountered a minor structural inconsistency in the last layers, where the convolutional block's channels alternated between 151 and 155 as it can be seen from Table 3.2

6. In case the new kernel size is larger than the resolution of the new feature map, the kernel size should be reduced to be equal to the resolution of the feature map, and the number of channels of that particular layer should be increased accordingly. This rule was introduced following experiments in which the FatNet achieved a network configuration where kernels were larger than the feature map resolution and did not train weights on the edges of the kernels, as shown in Figure 3.10.

In this work, PyTorch is used as the deep learning framework in all experiments, which is flexible regarding the resolution of the input image. The convolutional layers in PyTorch do not accept the input size as a parameter and can work with any input resolution. However, it is important to note that FatNet conversion has to be done per one particular input size because, according to rule 3, the number of pixels of the feature maps should remain the same.

Based on the rules provided, the FatSpitter algorithm is developed in this work to convert any network into its FatNet equivalent. The FatSpitter, accepts any model in PyTorch, and spits out the *"nn.Module()"* object of the FatNet. The algorithm consists of three parts: populating the construction table, conversion of layers into their FatNet equivalents and refinement of output channels.

The Construction table contains the number of weights and the number of feature pixels per layer. In order to follow core rule 2, the construction table is only observed after the layer where the number of elements of the feature map is less than or equal to the number of classes. The second stage is the replacement of layers with their FatNet equivalents. The flow chart of this algorithm for the Convolutional layers is shown in the Figure 3.1. Note that the algorithm recursively attends all layers in the model, and when the layer is sequential it treats it as another network and attends all its layers too. The flow chart in Figure 3.1 only shows the procedure of the Convolutional layers, as these ones are the most complex and important in our case. When it comes to other layers, they are either skipped or replaced in place. Finally the last stage, which is called refinement. Since the layers are being edited in order from shallow to deep layers, sometimes its necessary to correct the input channels of the current layer, meaning that the output channels of the previous layer must be refined accordingly. This is where the refinement stage comes, in order to ensure that the outputs of layers are always equal to the input of the following layers.

Batch Normalisation: As during the conversion, the number of channels of the layer always gets amended, not only Batch Normalisation but any kind of normalisation layer should be replaced by the same layer, with the number of channels equal to the number of output channels of the previous layer.

Flatten Layer: FatNet is originally developed for the classification models, which usually flatten the feature maps before fitting into the classifier. Unlike cone-shaped networks, FatNet always maintains the high resolution of feature maps. Hence the output layer of the FatNet is normally a matrix with the number of elements equal to the number of classes. Therefore, there is no need for the commonly used flattening operation in CNN, which is replaced by the identity layer in the FatSpitter algorithm. Although, the algorithm will automatically

place a flattening algorithm at the end of the network, in order to represent one hot encoding of the output in classifier networks.

Linear Layers: The linear layers are replaced with the Convolutional layer, the input channels of which are set to *next_input_channels* variable, which is determined by the output channels of the previous layer. The number of output channels of this linear layer is set to one, and the kernel size is set to the resolution of the feature maps.

3.2.2 FatSpitter in Convolutional layers

The main changes in the FatNet conversion occur in the convolutional layer. The primary goal is to maintain the resolution of the feature maps as high as possible, ensuring it is not smaller than the number of classes. Simultaneously, the number of channels is reduced accordingly to match the number of pixels in the feature maps with those in the feature maps of the original layer. Additionally, to maintain the original number of weights within the kernels, their resolution is increased following the channel reduction. This principle forms the foundation of the conversion; however, precise implementation requires adherence to specific algorithmic guidelines, as shown in Algorithm 1 and illustrated in Figure 3.1. These guidelines are outlined to ensure each layer is accurately adjusted to fit into the FatNet configuration.

The first thing to check is whether the layer is the first updatable convolutional layer. If it is, the input channels of the layer must remain unchanged. If the layer is not the first updatable layer, the number of input channels should be updated to match the number of output channels of the previous layer.

After establishing the number of input channels for the layer, the transformation of the convolutional layer proceeds with an adjustment of the output channel dimensions while keeping the input channels constant. Since the resolution of the feature map is higher than in the original network, the new output channels are calculated to match the number of pixels in the original feature map. Similarly, the kernel size is adjusted to maintain the same number of weights as in the original layer.

If the recalculated kernel size is larger than the feature map's resolution—a scenario that could lead to computational inefficiencies and hinder kernel training from the sides (See Figure 3.10)—it is resized, as it is stated in the rule 6. More precisely, the kernel size is capped using the minimum function, taking either the resolution of the feature map or the newly calculated kernel size, whichever is smaller. If the kernel had to be capped, to avoid exceeding the resolution of the feature map, the output channels are recalculated based on the new capped kernel size to match the correct number of weights.

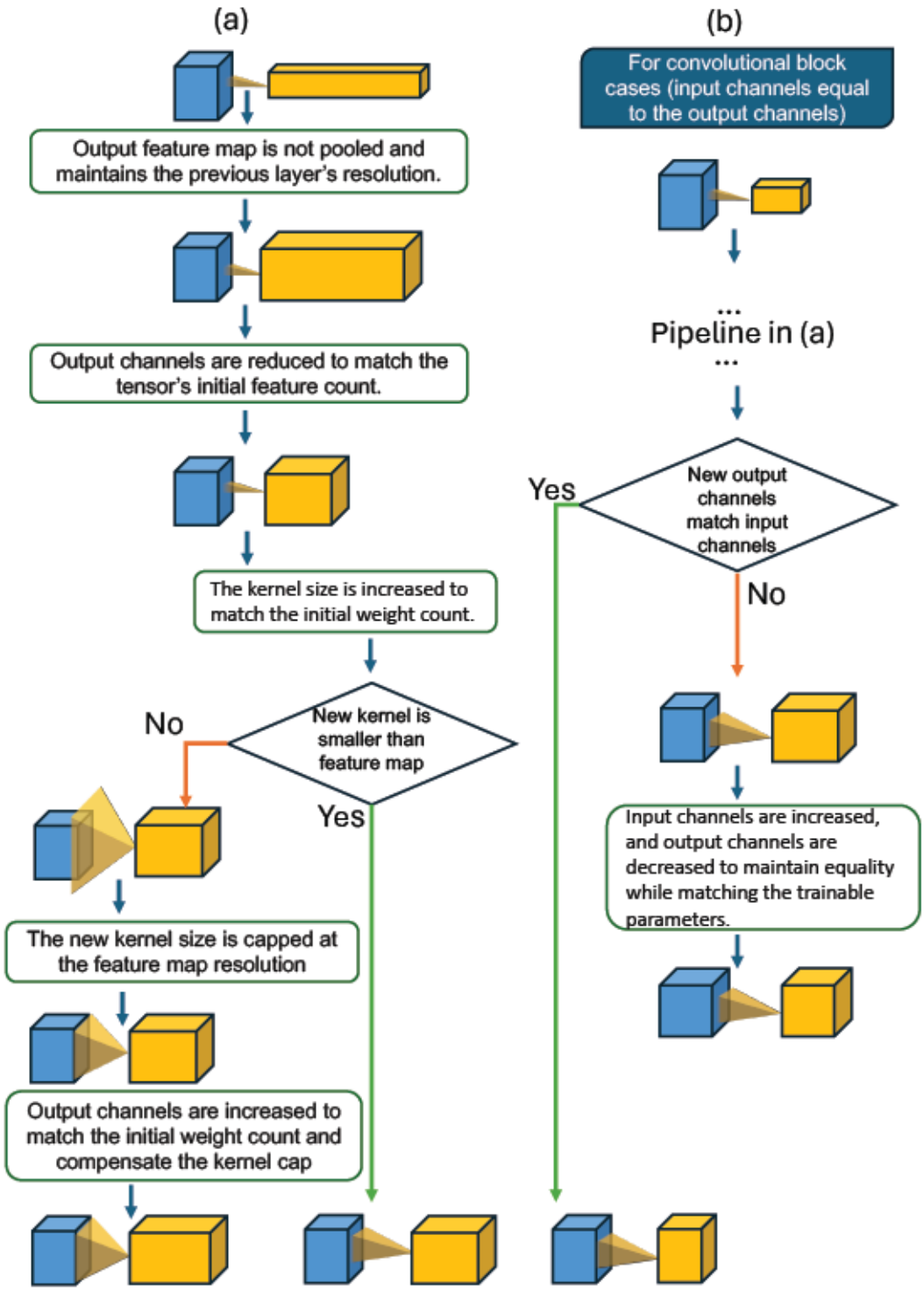


Fig. 3.1 **Flowchart of the FatSpitter, illustrating the process of increasing the resolution of feature maps and kernels while reducing the number of channels, in accordance with FatNet rules.** (a) Regular FatNet conversion. (b) FatNet for cases where the number of input channels equals the number of output channels.

Algorithm 1 Convert Convolutional Layers to FatNet

Require: *model*: neural network model, *construction_table*: table for weight count and number of features in layer, *start_index*: index to start conversion, *number_of_classes*: integer
Ensure: FatNet model

function REPLACE_LAYERS(*model*)
  for all *layer* in *model* **do**
    if *layer* has child layers **then**
      REPLACE_LAYERS(*layer*)
    else
      if *layer_index* $\geq$ *start_index* **and** *layer* is **Conv2d then**
        Step 1: Compute New number of input channels
        if *next_input_channels* is not set **then**
          $new\_in\_channels \leftarrow layer.in\_channels$
        else
          $new\_in\_channels \leftarrow next\_input\_channels$
        end if
        Step 2: Compute *new_out_channels* and *new_kernel_size*
        $feature\_pixels \leftarrow construction\_table[layer\_index].feature\_pixels$
        $weight\_count \leftarrow construction\_table[layer\_index].weight\_count$
        $new\_out\_channels \leftarrow feature\_pixels \div number\_of\_classes$
        $new\_kernel\_size \leftarrow \sqrt{weight\_count \div (new\_in\_channels \times new\_out\_channels)}$
        Step 3: Handle equal input and output channels
        if $layer.in\_channels = layer.out\_channels$ **then**
          if $new\_out\_channels \geq new\_in\_channels$
            or $new\_kernel\_size^2 \geq number\_of\_classes$ **then**
            $new\_kernel\_size \leftarrow \min(new\_kernel\_size, \sqrt{number\_of\_classes})$
            $new\_channels \leftarrow weight\_count \div (new\_kernel\_size^2)$
            $new\_in\_channels \leftarrow new\_out\_channels \leftarrow new\_channels$
          end if
        else
          if $new\_kernel\_size^2 \geq number\_of\_classes$ **then**
            $new\_kernel\_size \leftarrow \sqrt{number\_of\_classes}$
            $new\_out\_channels \leftarrow weight\_count \div (new\_in\_channels \times new\_kernel\_size^2)$
          end if
        end if
        Step 4: Replace the layer
        Replace *layer* with **Conv2d**(*new_in_channels*, *new_out_channels*, *new_kernel_size*, padding="same")
        $next\_input\_channels \leftarrow new\_out\_channels$
        Increment *layer_index*
      end if
    end if
  end for
  return *model*
end function

An essential aspect of transforming convolutional layers into the FatNet architecture is maintaining the balance between input and output channels, as stated in rule 5. If the original convolutional layer maintains equal numbers of input and output channels, this parity is generally preserved in the transformed layer to maintain the layer's internal dynamics and learning characteristics. If, after calculating the new output channels and kernel size, this equality of channels is not preserved, new input and output channels are calculated by taking the square root of the fraction of the number of weights over the newly calculated squared kernel size.

After these refinements to maintain channel equality, it is evident that when the input channels of the layers are updated, the output channels of the previous layer must be readjusted. This process of refinement, which focuses on updating the output channels of layers whose subsequent layers' input channels no longer match, begins after the conversion is finished. During this process, the output channels of these layers are updated to match the input channels of the next layers. Additionally, the kernel size is adjusted to ensure there is no mismatch in the number of weights between the FatNet equivalent and the original layer, as shown in Figure 3.2.

The FatSpitter is available at the following URL:
https://github.com/riadibadulla/FatSpitter (accessed on 25.09.2024)

3.2.3 Optical Simulator

The initial step in building the optical neural network involves simulating light propagation within the 4f system and evaluating the optical performance of the convolution operation. This simulation can be achieved using the Angular Spectrum of Plane Waves (ASPW) method [130].

Initially, it was anticipated that the PyOptica [131] library would seamlessly integrate with PyTorch's custom layer, which also uses the ASPW method and even provides its use in the simulation of the 4f system. However, due to redundant operations that would slow down the inference and the exclusive reliance on NumPy arrays in the code, the simulator had to be rewritten from scratch. This new implementation retained the same methods as those employed by the PyOptica library.

According to the Angular Spectrum method, if the initial wavefront is $U_1(r,c)$, the next wavefront is calculated as:

$$U_2(r,c) = \mathscr{F}^{-1}[\mathscr{F}[U_1(r,c)]H(f_r,f_c)] \tag{3.1}$$

where $H(f_r,f_c)$ is the transmittance function for free-space.

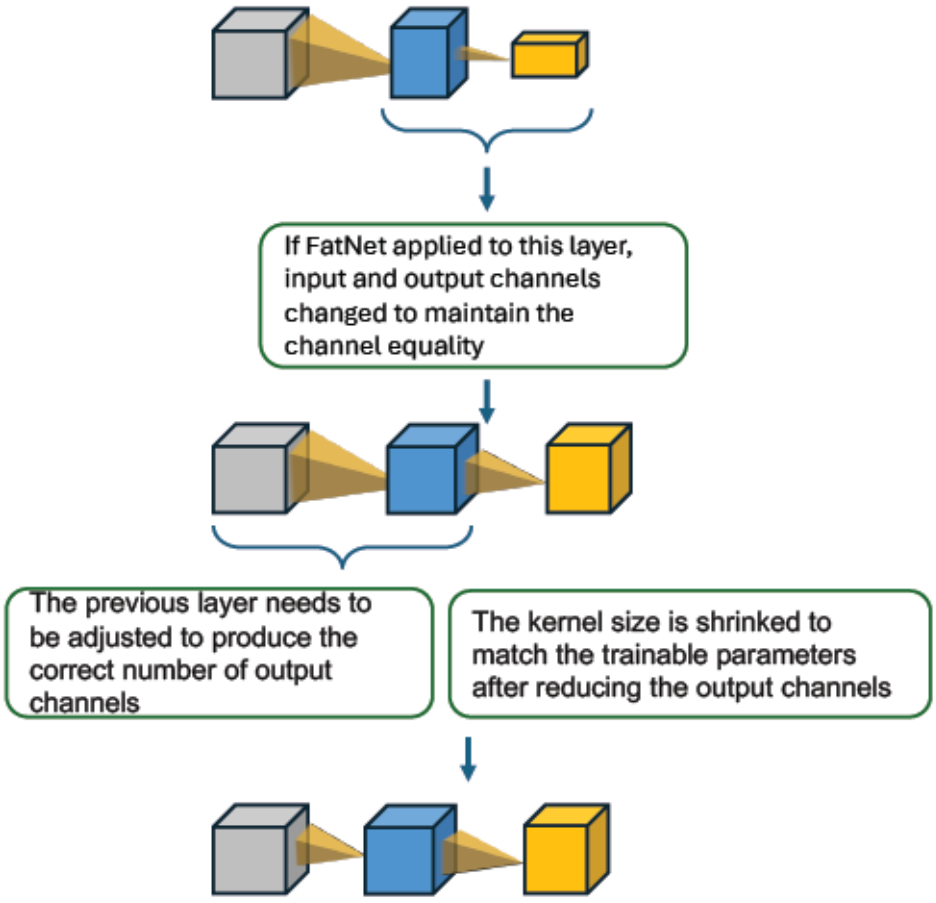


Fig. 3.2 **Flow chart for the FatNet refinement of the convolutional layers.** After the FatNet conversion, some convolutional layers' output channels will not be equal to the input channels of the next layers. Hence, their output channels are adjusted; consequently, the kernel size is adjusted accordingly.

The transmittance function of the free-space propagation comes from the Fresnel diffraction transfer function:

$$H_F(f_r, f_c) = \exp\left[ikz - i\pi\lambda z(f_r^2 + f_c^2)\right] \tag{3.2}$$

where $k = \frac{2\pi}{\lambda}$, z is the distance travelled by light and λ is the wavelength [132, 130].

Since the 4f system contains two lenses, the transmittance function of each lens is:

$$t_A(r,c) = P(r,c)\exp\left[-i\frac{k}{2f}(r^2 + c^2)\right] \tag{3.3}$$

where f is the focal length of the lens and $P(r,c)$ is the pupil function [130].

The distance at which the angular spectrum method calculates the next wavefront depends on the pixel scale and is calculated as:

$$z = \frac{N(\Delta x)^2}{\lambda} \tag{3.4}$$

where Δx is the pixel scale, N is the number of pixels, and λ is the wavelength. In case when the propagation distance needs to be longer than the above formula for the distance, the propagation can be calculated in several iterations. We chose such pixel scale for each propagation, so z becomes equal to the focal distance of the lens. In this case, we have to do only one iteration for each focal distance propagation in the 4f system.

The simulator uses pseudo-negativity, so each convolution is run twice to avoid the kernel's negative values in optics. Moreover, due to the laws of geometrical optics, the output of the 4f device is always rotated 180 degrees. Luckily, this is not a problem for convolutional neural networks since they can continue extracting the futures from the rotated feature maps.

Figure 3.3 illustrates the amplitude of the wavefront at various points within the 4f system during the convolution operation run via the simulator. The system comprises the input plane (laser), two convex lenses, a Fourier plane (modulator or phase mask), and a camera. Light passing through the first lens forms a 2D Fourier transform at the Fourier plane, where it is multiplied by the kernel. The second lens then transforms the light back into the spatial domain, and the resulting output is captured by the camera. Due to geometrical optics, the output image formed at the camera is flipped. This flipped output can be corrected after the camera readout to display the correct image orientation. Alternatively, it can be left as is, as CNNs are capable of extracting features from flipped images as well.

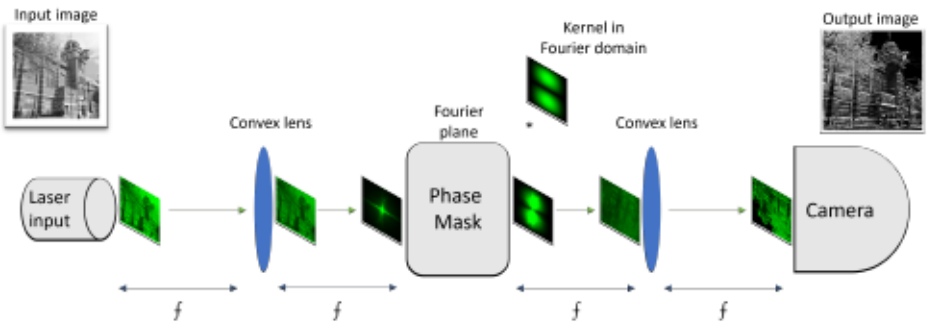


Fig. 3.3 **Graphical representation of the 4f system performing the convolution operation in the simulator.** The system consists of the input plane (laser), the convex lens, the Fourier plane (modulator or phase mask), and another convex lens and the camera is separated from each other by one focal distance of the lens. When light passes through the lens, it forms a 2D Fourier transform on the Fourier plane, where it can be multiplied by the kernel in the frequency domain. The light then passes through the second lens, which converts it back into the space domain, where the output is read by the camera.

3.2.4 OptNN layer

Section 3.2.3 described the simulation of the optical inference of the image through the 4f system. This section contains the integration of the optical simulator into the custom PyTorch layer in order to build the simulation of the optical neural network.

Generally, the built-in convolutional layers in machine learning frameworks like Tensorflow or PyTorch take full advantage of CUDA and perform the 2D convolution layer's convolution operation in three dimensions. Suppose there is a convolutional layer with an input feature map of $X_i \times Y_i \times C_i$ and an output feature map of $X_o \times Y_o \times C_o$, the number of 3D kernels in the layer is equal to the number of the output channels C_o. As per Figure 3.4, each 3D kernel tensor slides over the input tensor, performs a dot product of the overlapping section and saves it as a new pixel to the relevant channel of the output tensor. Each kernel tensor corresponds to its specifier channel in the output tensor.

In the 4f optical device, the images are treated as 2D matrices. Hence the convolution has to be performed in a traditional 2D format and the summed, which is not common for the convolutional layers. As it can be seen from Figure 3.5, the input tensor is split into the i channels, and each channel has a corresponding j number of kernels. After the convolution operations are performed, the outputs of distinct input channel convolutions are summed to form the j number of output channels, which are used to form the output tensor.

The simulator is available at the following URL:
https://github.com/riadibadulla/simulator (accessed on 25.09.2024)

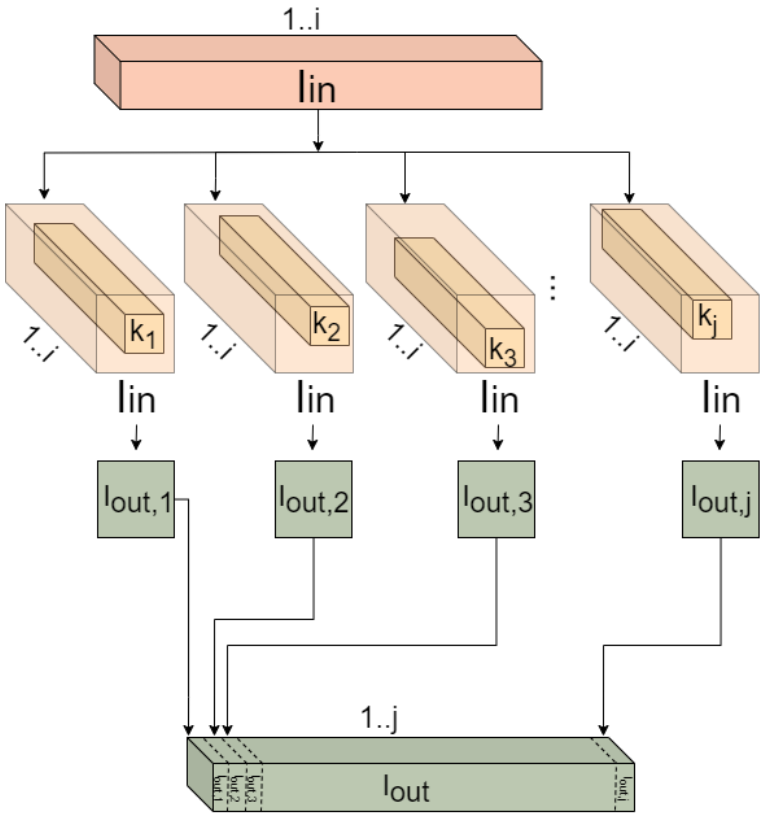


Fig. 3.4 **Visual demonstration of the regular convolutional layer performing the 2D convolution operation in 3D.** This convolutional layer has the number of input channels i and the number of output channels j. The number of 3D kernels is j, but the number of kernels taken in 2D is $i \quad j$.

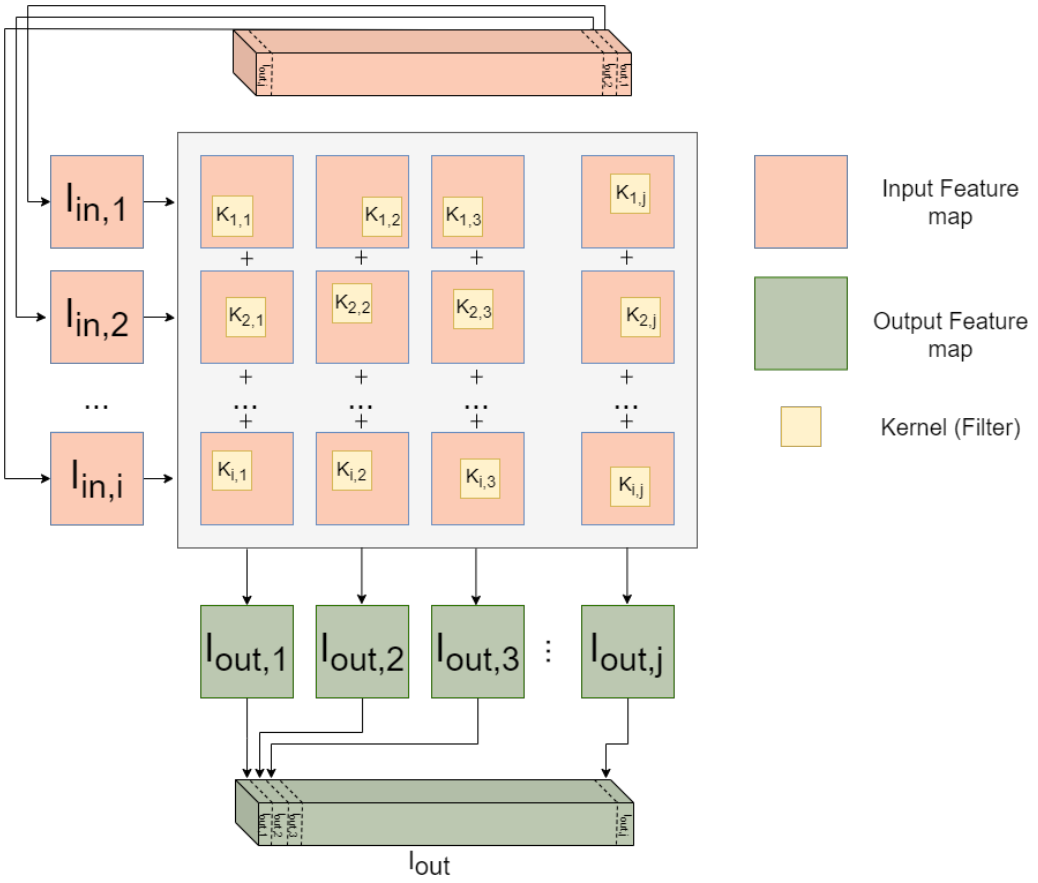


Fig. 3.5 **Visual demonstration of the convolutional layer performing convolution operation in 2D.** The convolutional layer has an input tensor of depth i and an output tensor of depth j. The number of kernels required for the layer is $i \quad j$. After all convolution operations, the outputs have to be summed to make up a channel of the output tensor

3.2.5 Pseudonegativity

As it was mentioned earlier, one of the main constraints of the 4f system is its incapability to represent negative numbers as the camera reads out the intensity of light, which is the square root of the amplitude. According to Chang *et al.* [1], one way around this constraint is called pseudo-negativity, which can address the restriction to positive values by doubling the number of filters. This technique involves labelling half of the kernels as positive and the other half as negative. After the camera readout, it required a digital operation to subtract the negative sub-images from their positive counterparts.

While the pseudonegative method has proven effective, our approach introduces a slight modification. Rather than storing double the amount of kernels to accommodate positive and negative values separately, we utilise a single kernel that contains both positive and negative values, akin to standard kernels. During inference, we generate two versions of this kernel: a positive version, retaining only the positive values and replacing negative ones with zeros, and a negative version, which keeps the absolute values of the original kernel's negative elements and substitutes zeros for the positive values. Following the principles of the original pseudonegative method, the negative version's outcome is subtracted from the positive one to achieve the desired convolution result (See Figure 3.6).

This method can be efficiently executed by applying the ReLU function to both the original and its inverted form as shown in Equation 3.5.

$$X * K = X * ReLU(K) - X * ReLU((-1)K), \tag{3.5}$$

where X is the input image, K is the kernel and $ReLU(K)$ function can be described as $max(0,K)$. Just like the pseudo negative method described by Chang *et al.* [1], this method carries two convolution operations instead of one, making it twice as expensive as the original convolution operation, but neglects the negativity in optics by performing subtraction in the electronic domain only. The result of the method is identical to the original convolution operation with the negative values, hence no performance loss is expected. The method can be applied to the input image as well, but is not necessary when the ReLU activation functions are used in the neural network.

3.2.6 ResNet-18

ResNet [4], which stands for Residual Network, was a paradigm shift in how CNNs were designed and conceptualised. Before the introduction of ResNet, deep learning research was focused on designing deeper networks in the hopes of achieving better performance, like VGG networks [2], which had configurations up to 19 convolutional layers. However,

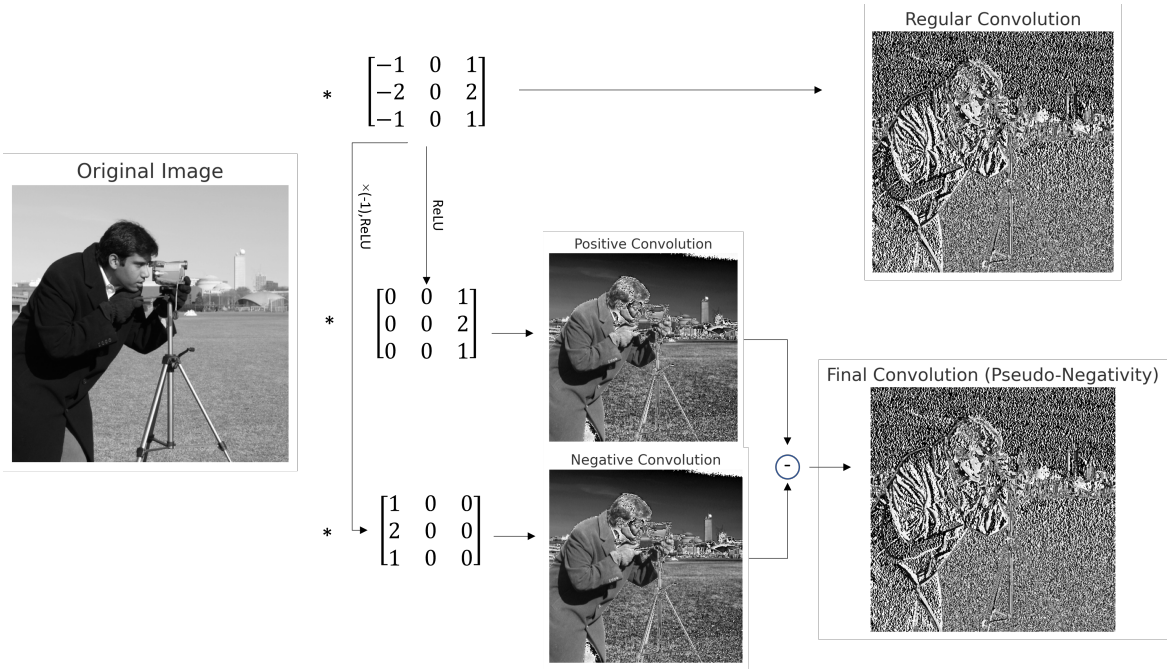


Fig. 3.6 **Comparison of regular convolution and convolution using a pseudo-negative split of the kernel on the example of the Sobel edge detection** Instead of training the model with two different positive kernels and subtracting the outputs, in this method one regular kernel can be used twice one for negative and for positive numbers using sign flip and ReLU functions and only then the outputs are subtracted to emulate the convolution with negative numbers. Both processes yield identical results for convolution operations.

deeper networks often failed to converge, and the accuracy gains from additional depth were saturating and even degrading. ResNet architecture broke the trend and has revolutionised the field of deep learning, particularly in handling the vanishing/exploding gradients problem that arises when training deeper networks. ResNet-18 is one of the five convolutional neural networks introduced by He et al. [4] for the ImageNet dataset [133] training. The name ResNet-18 indicates eighteen layers of the network; other configurations also include 34, 50, 101 and 152 layers. However, ResNet-18 became popular for applications requiring a balance between accuracy and computational efficiency. For the dataset used in this work, CIFAR-100, ResNet-18 is both sufficient for achieving strong performance and well-suited for proving the concept, which is why it was chosen over other ResNet configurations.

The key feature of ResNet architecture is the use of 'shortcut' connections, also known as residual connections, which allow the gradient to be directly backpropagated to shallower layers. This makes the network easier to optimise and mitigates the degradation problem. By skipping one or more layers and directly adding the input to the output, the network could easily learn the identity function. This, in turn, helps the model focus on learning the differences or residuals.

The ResNet-18 model is composed of five stages. The first stage is the $\times 7$ convolution with a stride of 2, followed by the max pool operation to reduce the resolution of the feature maps. Since it is impossible to perform strides convolutions in 4f system, that layer was modified with the regular convolution layer of stride one. The other four stages contain two basic blocks each. There is a max pool operation at the end of each stage, except the last one, which contains an average pool instead. Just like any other classifier CNN, the output of the convolutional part is fit into the dense layers with the number of output neurons equal to the number of classes.

The ResNet-18 model was selected for training and evaluation in this work using the CIFAR-100 dataset due to its balance between computational efficiency and model performance, as well as its proven success in training on the CIFAR-100 dataset.

The initial hypothesis was to use large kernels to fully utilise the high-resolution capabilities of the free-space Optical setup and investigate the effects of augmenting kernel size in ResNet architectures. However, empirical testing revealed that an increase in kernel size without a concomitant reduction in the number of channels led to overfitting in the resulting FatNet architecture.

The modification strategy thus evolved to simultaneously increase the kernel size and decrease the number of channels in the convolutional neural network. This approach resulted in networks with a very small number of channels and disproportionately large kernels. A

challenge arose in the deeper layers of these networks when the kernel size exceeded the dimensions of the input feature maps. The construction table of this FatNet is shown below:

Table 3.1 Construction of FatNet from ResNet-18.

Original layers		Weights	Pixels	FatNet with larger kernels	
Channels	Kernel size			Channels	Kernel size
64×128	3	73,728	8,192	64×82	4
128×128	3	147,456	8,192	82×82	5
128×128	3	147,456	82,192	82×82	5
128×128	3	147,456	82,192	82×82	5
128×256	3	294,912	4,096	82×41	9
256×256	3	589,824	4,096	41×41	19
256×256	3	589,824	4,096	41×41	19
256×256	3	589,824	4,096	41×41	19
256×512	3	1,179,648	2,048	41×21	37
512×512	3	2,359,296	2,048	21×21	73
512×512	3	2,359,296	2,048	21×21	73
512×512	3	2,359,296	2,048	21×21	73
FC(512×100)	3	51,200	100	21×1	49

Upon further analysis, it was discovered that in such cases, when the kernel is larger than the input, in the same-padded or FFT convolution, the roles of the input and kernel were effectively inverted. Given that the convolution output was cropped to the input dimensions, this led to a phenomenon where the side pixels of the kernels were not being trained. As these side pixels lay outside the network's receptive field, they constituted trainable parameters that neither contributed to performance nor impacted the network output while unnecessarily consuming memory and computational resources. Despite its extreme efficiency implemented with the optical accelerators, this kind of architecture is faulty, and the visualisation of the kernels is shown in the results section.

To address this issue, a new rule, Number 6, was introduced for the kernel dimensions in problematic layers, limiting them to the resolution of the feature maps (See Section 3.2.1). This resolution was determined to be equal to the square root of the number of classes, which is the minimum resolution of the feature maps in FatNet. Corresponding adjustments were made to the number of channels, ensuring a consistent count of feature pixels and trainable

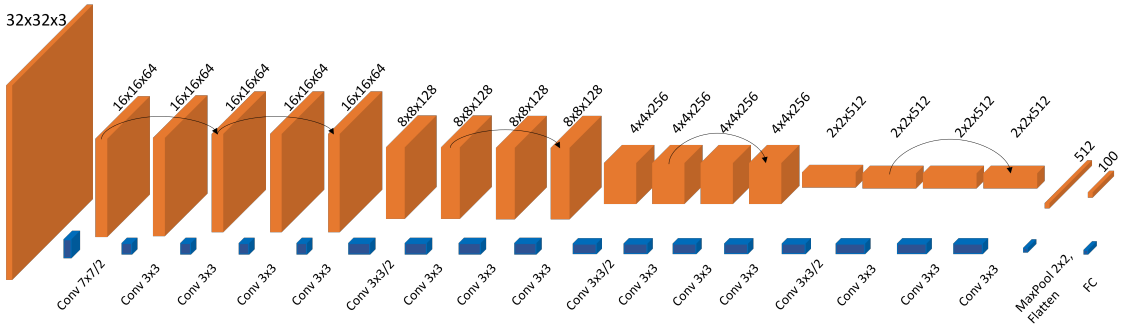
parameters. The construction network below demonstrates the construction of the FatNet using the new rule for the maximum kernel size constraint:

Table 3.2 Construction table of Res-FatNet from ResNet-18, after capping the resolution of the kernels at a certain limit equal to the resolution of the feature maps. Kernel sizes do not exceed the resolution of the feature maps. The output channels that were not equal to the input channels of the next layer were readjusted and indicated in bold.

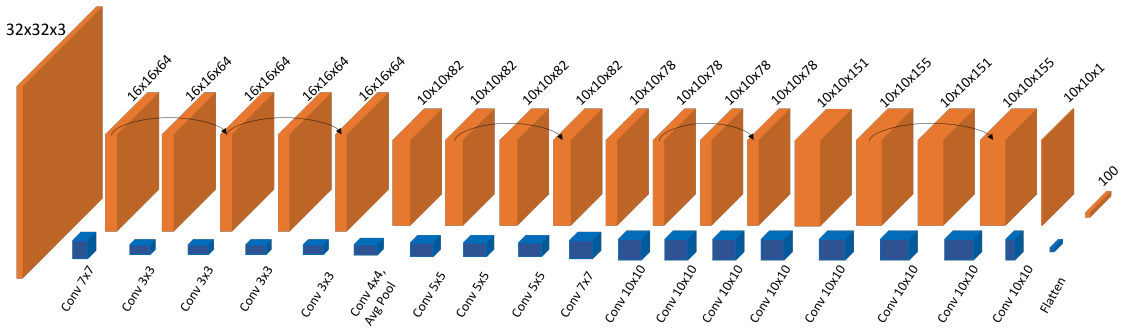
Original layers		Weights	Pixels	FatNet		Fat-Net adjusted	
Channels	Kernel			Channels	Kernel	Channels	Kernel
64×128	3	73,728	8,192	64×82	4	64×82	4
128×128	3	147,456	8,192	82×82	5	82×82	5
128×128	3	147,456	82,192	82×82	5	82×82	5
128×128	3	147,456	82,192	82×82	5	82×82	5
128×256	3	294,912	4,096	$82 \times \mathbf{41}$	**10**	$82 \times \mathbf{78}$	**7**
256×256	3	589,824	4,096	78×78	10	78×78	10
256×256	3	589,824	4,096	78×78	10	78×78	10
256×256	3	589,824	4,096	78×78	10	78×78	10
256×512	3	1,179,648	2,048	78×151	10	78×151	10
512×512	3	2,359,296	2,048	151×155	10	151×155	10
512×512	3	2,359,296	2,048	155×151	10	155×151	10
512×512	3	2,359,296	2,048	151×155	10	151×155	10
FC(512×100)	-	51,200	100	155×1	10	155×1	10

The comparison of the visualised architectures can be seen in the Figure 3.7

Observing the Res-FatNet architecture, it can be seen that the last block of layers contains fluctuations in the number of channels. The last layers, have input channels and output channels of either 155 or 151. While these fluctuations were found to have a negligible impact on performance, a more significant effect was noted in other architectures due to larger fluctuations in the number of channels. Consequently, when developing the automated conversion tool, FatSpitter, a rule Number 6 was introduced to preserve the equality of input and output channels of the original network for the corresponding FatNet iteration (See Section 3.2.1). This rule has been strictly adhered when adapting other architectures used in this work.



(a)



(b)

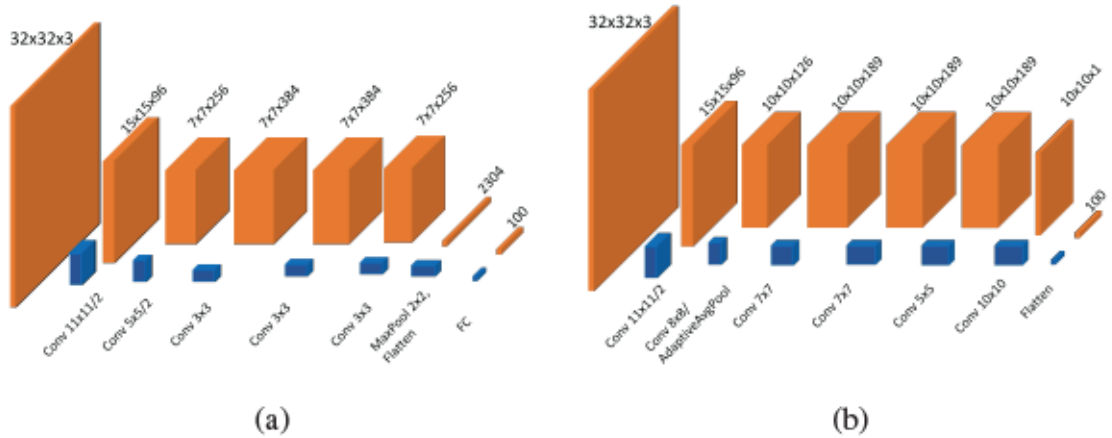
Fig. 3.7 **The comparison between our tweaked ResNet-18 used for CIFAR-100 training and FatNet, which is built on ResNet-18 and specifically designed for CIFAR-100 classification.** (a) ResNet-18 architecture is slightly altered in our version. Our model does not employ strides, given that optical systems are unable to carry out strides in convolutions. Moreover, to enhance compatibility with CIFAR-100, we've omitted the second non-residual convolutional layer. (b) FatNet, which is based on ResNet-18, is tailored for CIFAR-100. This structure has fewer channels but larger resolutions when compared to ResNet-18. The kernel sizes can reach up to 10 10, and feature maps are never pooled to smaller than 10 10. The concluding layer is a 10 10 matrix that is flattened into a 100-element vector, with each element representing a CIFAR-100 class.

3.2.7 AlexNet

AlexNet is a deep convolutional neural network proposed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton in 2012 [3]. AlexNet architecture is known for being a turning point in the deep learning era and significantly influenced the field of AI by winning the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC). AlexNet, was the first work to use ReLU activation functions to eliminate the Vanishing Gradient Problem, Dropouts for regularisation and implementation of Deep Learning on GPU.

AlexNet contains eight layers: five convolutional layers and three dense layers. The original input of AlexNet is a $224 \times 224 \times 3$ image, and the output is 1000 classes for the ImageNet classification.

The first two convolutional layers use kernels of 11×11 and 5×5, respectively, and only the next three convolutional layers use common 3×3 kernels. Moreover, the first convolutional layer is the strided convolution, which we had to replace with the regular same padded convolution (See Figure 3.8 (a)), as it is the only type of convolution possible in the 4f system.



(a) (b)

Fig. 3.8 **The comparison between our tweaked AlexNet used for CIFAR-100 training and its FatNet equivalent, built on AlexNet and specifically designed for CIFAR-100 classification.** (a) AlexNet architecture is slightly altered in our version. Our model contains only one linear layer to make it more compatible with CIFAR-100. (b) Alex-FatNet tailored for CIFAR-100. This structure has fewer channels but larger resolutions when compared to AlexNet. The kernel size reaches 10×10 in the last layer, and feature maps are never pooled to smaller than 10×10. The concluding layer is a 10×10 matrix flattened into a 100-element vector, with each element representing CIFAR-100 classes.

After the fifth convolutional layer, max pooling is performed and then the network switches to fully connected layers. The fully connected layers have 4096, 4096, and 1000 neurons, respectively. The output of the final layer is fed to a softmax function, which outputs

a probability distribution over 1000 classes. Since our classification task contains only 100 classes, the linear layers were replaced by linear layer from 4096 neurons to 100 classes directly.

The conversion to the Fat equivalent begins right after the second feature map as it can be seen from the Figure 3.8 (b).

The construction table provides a detailed process of converting AlexNet into Alex-FatNet (See Table 3.3). The classifier of the model is replaced by a 256-to-1 channel convolution with a kernel size of 10×10. Unlike Res-FatNet, the rule stating that the equality of input and output channels is preserved if they are equal in the original network is taken into account in the FatSplitter algorithm used to convert AlexNet. As seen from the Construction table, the third layer of 384 to 384 channels is converted into 189 to 189 channels.

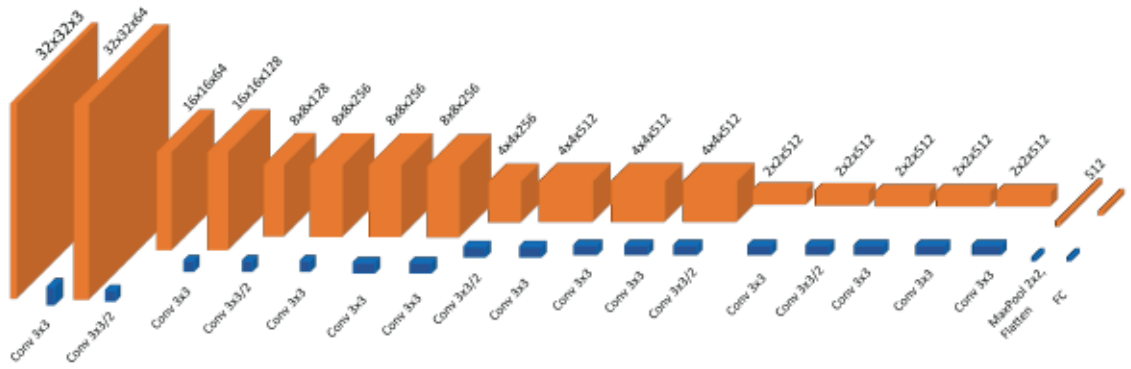
Table 3.3 Construction table of Alex-FatNet from AlexNet. Unlike the ResNet, AlexNet did not require the refinement of the layers.

Original layers		Weights	Pixels	Fat-Net	
Channels	Kernel			Channels	Kernel
96×256	5	614,400	12,544	96×126	8
256×384	3	884,736	18,816	126×189	7
384×384	3	1,327,104	18,816	189×189	7
384×256	3	884,736	18,816	189×189	5

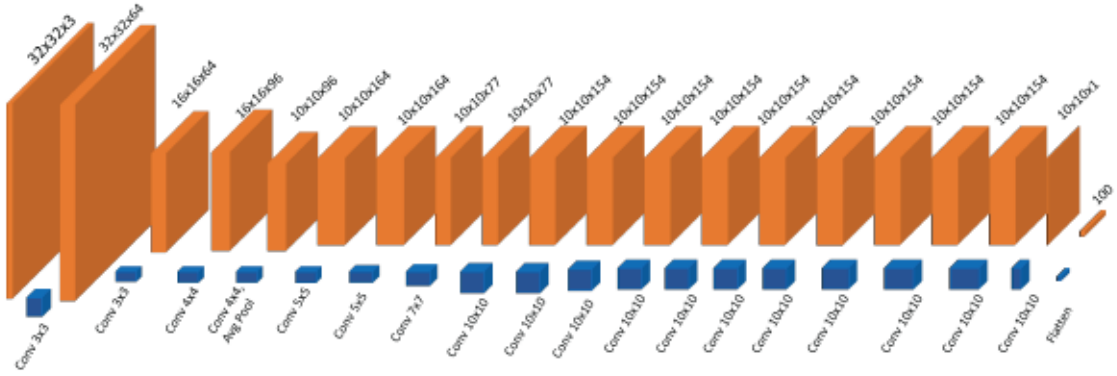
3.2.8 VGG-19

Another network tested and converted into the FatNet format in this work is VGG-19 [2]. VGG-19 is an abbreviation for Visual Geometry Group-19, which signifies a convolutional neural network architecture developed by the Visual Geometry Group at the University of Oxford. VGG network was introduced as a 16 and 19-layered convolutional network. VGG-19 refers explicitly to a variant of the VGG model that contains 19 layers; it includes sixteen convolutional layers with very small 3×3 kernels, five max-pooling layers with kernels size of 2×2 and stride 2, three linear layers, and one softmax layer. It's original version expects an input of 224×224 RGB images. VGG uses ReLU layers as the activation function. The VGG-19's linear layers start with two linear layers of 4096 nodes and a final layer of 1000, corresponding to 1000 classes. Since our dataset consists of images of resolution 32×32, the final feature map has the resolution of 2×2 only, and the number of classes is significantly

lower; the linear layers were replaced with one layer only, which maps from 512 nodes to 100 classes (See Figure 3.9 (a)). Unlike the original setup, no dropout layers were used. The Fat equivalent of the VGG-19 can be seen in the Figure 3.9 (b), where the conversion starts appearing after the fourth feature map.



(a)



(b)

Fig. 3.9 **The comparison between our tweaked VGG-19 used for CIFAR-100 training and its FatNet equivalent, built on VGG-19 and specifically designed for CIFAR-100 classification.** (a) VGG-19 architecture is slightly altered in our version. Our model contains only one linear layer from 512 nodes to 100 to make it more compatible with cifar100. (b) VGG-FatNet tailored for CIFAR-100. This structure has fewer channels but larger resolutions when compared to VGG-19. The kernel sizes can reach up to 10×10, and feature maps are never pooled to smaller than 10×10. The concluding layer is a 10×10 matrix flattened into a 100-element vector, with each element representing CIFAR-100 classes.

The Construction Table 3.4 provides a detailed process for converting VGG-19 into VGG-FatNet. The classifier of the model is replaced by a 154-to-1 channel convolutional layer with a kernel size of 10×10. Rules 6 and 7 were followed during the construction of VGG-FatNet, and the conversion was fully automated using the FatSpitter. In this particular

Table 3.4 Construction table of VGG-FatNet from VGG-19, after capping the resolution of the kernels at a certain limit equal to the resolution of the feature maps. Kernel sizes do not exceed the resolution of the feature maps. The output channels that were not equal to the input channels of the next layer were readjusted and indicated in bold.

Original layers			Weights	Pixels	FatNet			Fat-Net adjusted		
Channels		Kernel			Channels		Kernel	Channels		Kernel
128	128	3	147,456	8,192	96	96	4	96	96	4
128	256	3	294,912	16,384	96	164	5	96	164	5
256	256	3	589,824	16,384	164	164	5	164	164	5
256	256	3	589,824	16,384	164	**164**	**5**	164	**77**	**7**
256	256	3	589,824	4,096	77	77	10	77	77	10
256	512	3	1,179,648	8,192	77	**153**	10	77	**154**	10
512	512	3	2,359,296	8,192	154	154	10	154	154	10
512	512	3	2,359,296	8,192	154	154	10	154	154	10
512	512	3	2,359,296	2,048	154	154	10	154	154	10
512	512	3	2,359,296	2,048	154	154	10	154	154	10
512	512	3	2,359,296	2,048	154	154	10	154	154	10
512	512	3	2,359,296	2,048	154	154	10	154	154	10
512	512	3	2,359,296	2,048	154	154	10	154	154	10

case, it was important to preserve the equality of input and output channels when they are equal in the original network. Otherwise, fluctuations in the channel number could occur, disrupting the block-based structure of the VGGNet, where the organisation of convolutional blocks must be maintained to preserve the integrity of the VGG structure. As seen in the construction table, the output of the fourth layer was adjusted to 77 to match the number of input channels of the next layer. Since the number of channels was reduced, the kernel size was correspondingly increased to 7 7. In layer 6, however, the number of output channels was increased by one. Since this change was minor, it did not result in any adjustments to the kernel size.

3.3 Experiments

The primary aim of FatNet is not to enhance performance but to establish that the converted network architecture can sustain accuracy by leveraging the high-resolution capabilities of free-space optics for acceleration. Additionally, it aims to execute fewer inferences through the 4f system compared to the original network. Consequently, our research efforts were primarily focused on comparative analysis between the original networks and their FatNet equivalents.

CIFAR-100's training set was divided into training and validation subsets with an 80-20% ratio, yielding 40,000 images for training and 10,000 for validation. The dataset was normalised using the mean and standard deviation across all channels. Data augmentation techniques, including horizontal flipping and random cropping with padding of four, were also employed.

A modified version of ResNet-18 was replicated, transformed into its FatNet equivalent, and both versions of the networks underwent the training process. Different variations of ResNet-18 have been trained, but due to the simplicity and good performance, ReLU before the addition version was chosen (See Figure 2.5 (b)). To support Res-FatNet's accuracy within the optical device, the network was also trained in a simulator to demonstrate the feasibility of inference in the 4f system after the FatNet conversion.

Parameters for the experimental setup included setting the laser wavelength to 532 nm (green), and employing convex lenses with a diameter of 5 mm and a focal length of 10 mm. It should be noted that factors such as the device's quantisation and noise were not considered, and *float32* type was used.

The network, within a real 4f system, would have benefited from parallelism by employing tiling techniques for the batches. However, the simulator did not use such tiling techniques, given that matrices were encapsulated within PyTorch's tensor format. Operations were conducted without tensor unwrapping, with Fourier transforms and multiplications performed directly on the four-dimensional tensors. This methodology was selected due to the simulator-based network training's relatively slow pace compared to the standard PyTorch network. For example, each epoch of Res-FatNet's optical simulation requires 67 minutes, while an epoch within the standard Res-FatNet with Conv2d layer of PyTorch only takes 15 seconds.

ResNet and AlexNet, including their FatNet equivalents, were trained using the SGD optimiser with momentum set at 0.9. The initial learning rates were 0.01 and 0.005 for ResNet and AlexNet, respectively. ResNet and its FatNet variant updated the learning rate every 50 steps by a factor of 0.2. In contrast, AlexNet and VGG-19 used the Cosine Annealing scheduler [134]. The final layers of all residual networks incorporated a 20% dropout layer.

Both VGG-19 and VGG-FatNet were trained with the Adam optimiser due to experimentally better convergence, with an initial learning rate set to 0.01.

The training process was conducted on two NVIDIA A100 40 GB GPUs. Both ResNet-18 and Res-FatNet were trained with a batch size of 64 (32 per GPU), while AlexNet and VGG-19, including their FatNet equivalents, were trained with a batch size of 128 (64 per GPU).

When training the optical simulation of models, Res-FatNet was trained with SGD with an initial learning rate of 0.01, while Alex-FatNet and VGG-FatNet were trained with Adam with an initial learning rate of 0.00001.

Unlike GPU training, the optical simulation of FatNet required a reduced batch size due to the simulator's high memory requirements. Specifically, Fat-VGGNet was trained with a batch size of 12 (6 per GPU) and Res-FatNet and Alex-FatNet with a batch size of 16 (8 per GPU). The optical simulation increases the computational graph and gradient count, necessitating these adjustments.

Despite the absence of a simulation for the 4f system's parallelism, accelerating the 4f system necessitates exploiting high resolution. FatNet's optimal acceleration could be achieved through batch tiling (or input tiling). For effective batch tiling, all inputs of the same batch must be tiled into a single input block, with the kernel padded to match the input block's size. Prior to tiling the inputs, each must be padded to M+N-1, where $M \times M$ represents the input size, and N × N represents the kernel size. Given this methodology, the calculation for the number of potential batch sizes proceeds as follows:

$$n = \left\lfloor \frac{R}{M+N-1} \right\rfloor^2 \tag{3.6}$$

where R is the resolution of the 4f system

Upon success, the experiments were repeated with AlexNet and VGG-19. Unlike Res-FatNet, the cosine annealing scheduler was used for these networks. The training regime of all networks can be seen in the Table 3.5.

3.4 Results and Discussion

The results in this section evaluate the performance of FatNet-adapted models Res-FatNet, Alex-FatNet, and VGG-FatNet—in addressing the limitations of the 4f free-space optical system for image classification tasks. Specifically, we assume that the 4f system performs convolution operations for each mini-batch separately, while all other operations, including activations and pooling, are carried out electronically. The results assess how well these

Table 3.5 This table summarises the key parameters used in training each network, including the optimiser, learning rate, scheduler, batch size, dropout rate, and the number of epochs

Parameter	ResNet-18	AlexNet	VGG-19
Optimiser	SGD	SGD	SGD
Initial Learning Rate	0.01	0.005	0.01
Scheduler	$\gamma = 0.2$ step=50	Cosine Annealing	Cosine Annealing
Weight Decay	0	5e-3	5e-4
Batch Size	64	128	128
Number of Epochs	300	300	160
Data Augmentation	Random Flip, Random Crop, Standartisation	Random Flip, Random Crop, Standartisation	Random Flip, Random Crop, Standartisation

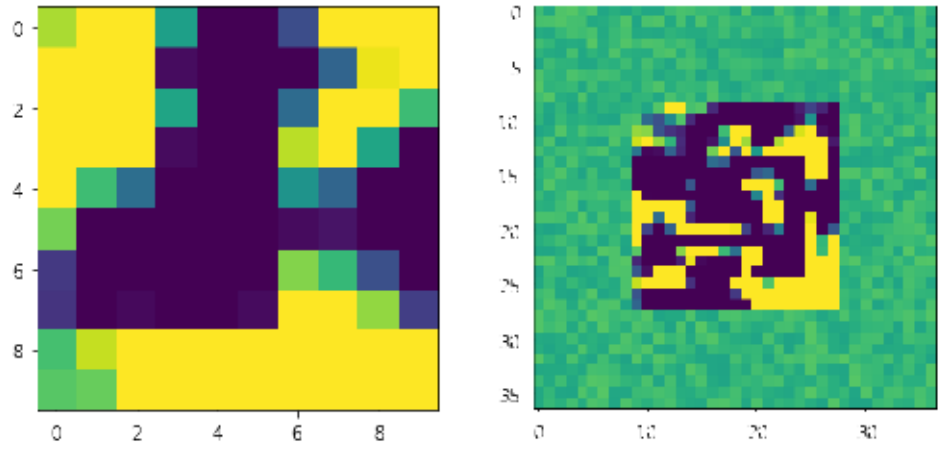
adapted architectures mitigate the bottleneck of readout speed while taking advantage of the system's high-resolution capabilities. Furthermore, the analysis explores the trade-off between model performance and inference speed.

In the methods section, the initial experiments were described, wherein the kernel size was not constrained to the maximum resolution of the input feature map. This unconstrained approach allowed the kernel size to exceed the dimensions of the feature map. Consequently, when employing same-padding convolutions, the output is cropped to align with the dimensions of the input feature map. This cropping makes the peripheral pixels of the kernel redundant, as they do not participate in the training process. This phenomenon decreases the network's efficiency, increases computational demands, and necessitates more memory.

The kernels from restricted and unrestricted networks were visualised in Figure 3.10. Figure 3.10 (b) shows that only a 10×10 section in the middle of the 37×37 kernel underwent training and has a similar colour scheme to the kernel in Figure 3.10 (a), which depicts a restriced kernel of 10×10 dimensions. The remaining pixels in Figure 3.10 (b) retain their values since initialisation, further evidencing the inefficacy of not restricting the kernels beyond the feature map resolution.

3.4.1 ResNet-18

Another kernel visualisation was conducted to compare Res-FatNet and ResNet-18, as shown in Figure 3.11. This employs a visualisation technique similar to that used by Krizhevsky *et al.* [3], to visualise the kernels of the first layer of the AlexNet. Given that the input image is

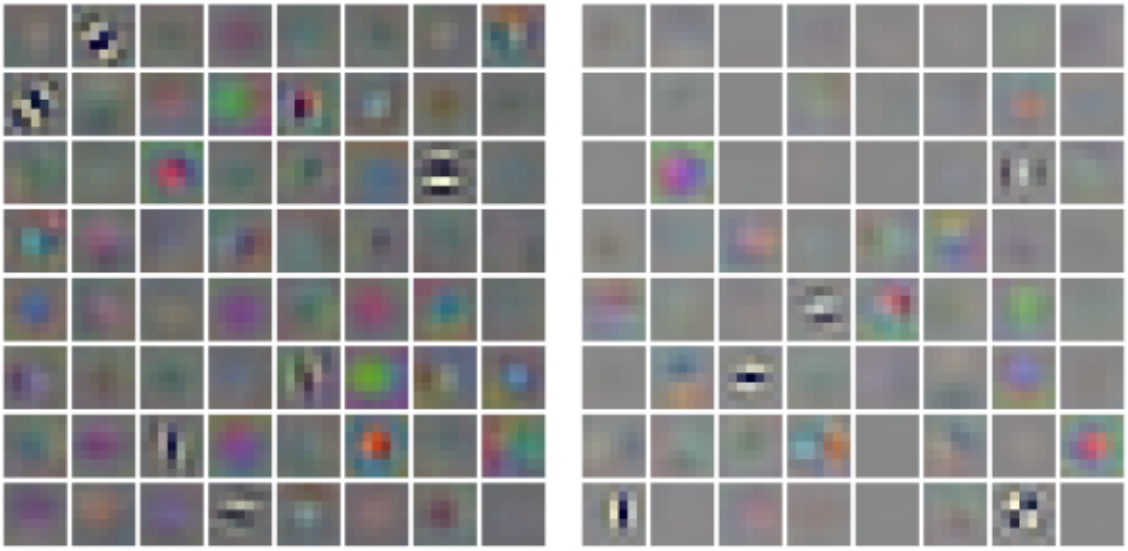


(a) (b)

Fig. 3.10 **Visual Representation of the impact of kernel size larger than the resolution of the feature map.** (a) Shows a kernel from FatNet, restricted to a 10×10 size to match the maximum feature map dimensions. (b) Illustrates an unrestricted kernel with a size of 37×37, exceeding the feature map resolution. Note the "no training" regions along the edges of the kernel in (b)

an RGB three-channelled image, the kernels can be visualised in colour. Both ResNet and its Fat variant maintain the same architecture until the feature maps are pooled down to a resolution with a pixel count lower than the number of classes, allowing for a comparison of the initial layers to ascertain if both networks train in a similar manner. It is observed that the kernels from ResNet, as shown in Figure 3.11 (a), exhibit more vivid colours than those from FatNet in Figure 3.11 (b), potentially explaining FatNet's slightly lower performance. However, both networks appear to contain similar learned kernels, indicating they indeed learn the weights in a similar manner.

The acceleration of networks inference by converting it into its Fat equivalent results in a small decrease in performance. For example, in the case of ResNet, there is a 6% drop in accuracy, from an average of 66% to 60% after converting to Res-FatNet (refer to Table 3.6). The performance remained unchanged when the Res-FatNet was trained in the optical simulator. The ResNet-18 was trained to reach state-of-the-art performance on CIFAR-100, matching the results obtained with ResNet-18. As discussed in Section 2.9.2, Mizusawa *et al.* [93] trained ResNet-20 to achieve an accuracy of 64.09%, while the smaller ResNet-18 in this work achieved 66% accuracy. In another study, however, Chen *et al.* [96] achieved an accuracy of 68.67% with ResNet-20, which is slightly higher than the ResNet-18 results in this work.



(a) (b)

Fig. 3.11 **First layer visualisation of kernels in (a)ResNet and (b)Res-FatNet.** Both sets of kernels represent various similar patterns learned during the training process.

Table 3.6 Comparison of the test accuracy and number of convolution operations used in ResNet and Res-FatNet. Pseudo-negativity is taken into account in optical setup.

Architecture	**Test Accuracy**	**Number of Conv Operations**	**Number of Conv Operations**
	mean $\pm$ std		**Ratio to Baseline**
ResNet-18	$66 \pm 1.4\%$	1,220,800	1 (baseline)
Res-FatNet	$60 \pm 1.4\%$	148,637	0.12
Optical simulation Res-FatNet	60%	297,274	0.24

Although the Res-FatNet trained in the simulator perfromed similarly to the regular Res-FatNet, from the training accuracy graph in Figure 3.12, it can be seen that it experienced slower training compared to other experiments. When simulating the 4f system, PyTorch incorporates the simulation of light propagation into the neural network's computation graph, significantly enlarging it. This enlargement leads to a slowdown in the network's training process. Regarding validation accuracy, the FatNet trained with a GPU and its optical simulation are not significantly different, particularly after the first learning rate adjustment at epoch 50. While the validation accuracy of both FatNet and its optical simulation did not

surpass 57% and 58%, respectively, the test accuracy reached 60% in both scenarios. This drop arises from the augmentation applied only to the validation and training sets, but not to the test set.

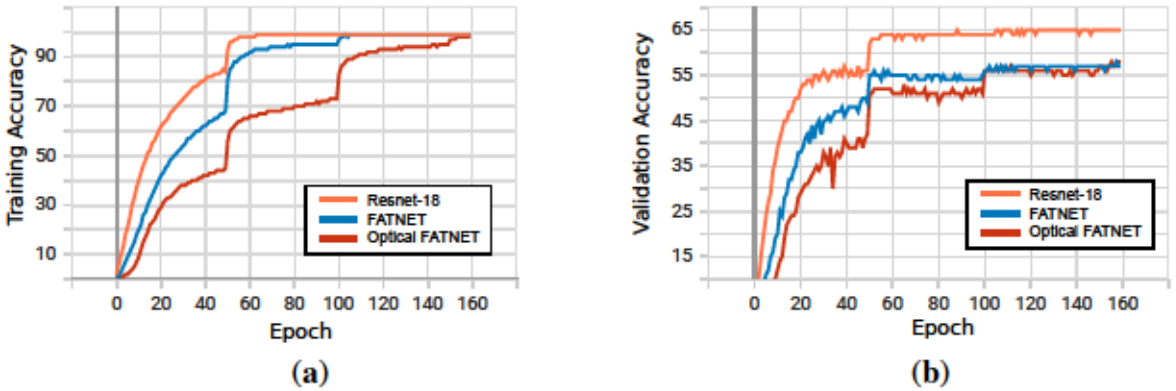


Fig. 3.12 **Comparison of training and validation accuracies per epoch for each model used in the experiment.** The steep jumps can be see every 50 epoch due to the learning rate update by a scheduler. (a) The ResNet-18, Res-FatNet, and the Optical simulation of Res-FatNet all accomplished a training accuracy of 99%, with ResNet-18 achieving it in fewer epochs. Conversely, the optical simulation took longer to train due to the complex computation needed to simulate light propagation. (b) As for validation accuracy, ResNet-18 reached up to 66%, while Res-FatNet could not exceed 60% for both validation and testing, despite its fewer convolutional operations.

This slight drop in performance came with the acceleration in optics. The fact that Res-FatNet requires 8.2 times fewer convolution operations means that it can perform inferences 8.2 times faster in the 4F system compared to ResNet in the same system. But it is important that the comparison is fairly made between GPU and 4f system too.

The measured and calculated inference time for ResNet and its FatNet equivalent with optics and GPU were obtained and observed (see Table 3.7). The observations were conducted based on the batch size of 64, such as in our experiments, and 3136 maximum utilisation of 4f system with 4k resolution modulators and camera, and the 2MHz system, as in the work of Li *et al.* [7]. The GPU measurements are the average of 300 inferences using CUDA event timing, with the GPU being warmed up initially by 10 iterations.

It is evident that for the small batch size, the acceleration between ResNet-18 and Res-FatNet-18 is 8.2 times, as discussed earlier, but there is no acceleration in comparison to GPU. The difference comes with a higher batch size. When the batch size is at a maximum of 3136, the acceleration of our Res-FatNet-18 in optics surpasses all other models on GPU, making optical Res-FatNet 2.46 times faster than ResNet-18 with the GPU. It is also evident

Table 3.7 Inference time in seconds per input for ResNet-18 and FatNet with optics and GPU with a batch size of 64 and 3136 for cases when the 4k resolution of the 4f device is fully utilised. The frame rate of the 4f device is approximated at 2MHz [7].

Architecture	Batch 64	Batch 3136
ResNet-18(GPU)	**1.3500e−4**	1.1670e−4
ResFatNet(GPU)	4.5650e−4	7.9420e−4
ResNet-18(Optics)	1.9075e−2	3.8929e−4
ResFatNet(Optics)	2.3225e−3	**4.7397e−5**

that without the FatNet conversion, the model wouldn't have any acceleration in optics. Even with the batch size of 3136, ResNet-18 would be 3.33 times faster than ResNet-18 in optics.

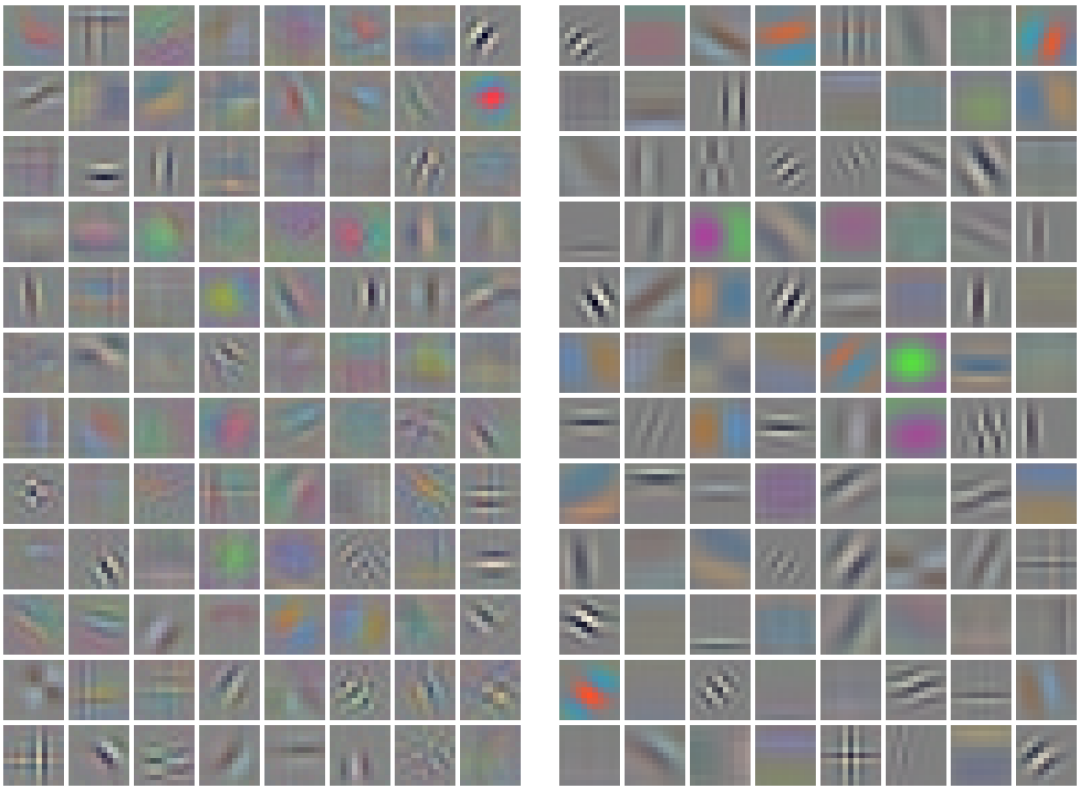
3.4.2 AlexNet

Experiments with AlexNet were conducted strictly following the rule that the resolution of the kernel must not exceed the feature map resolution. Moreover, the conversion was fully automated using FatSpitter, with the rule of equality in the number of channels being adhered to.

The first layer kernels of both AlexNet and Alex-FatNet were visualised, similar to the approach used for ResNet, shown in Figure 3.13. Interestingly, in contrast to ResNet, in the case of AlexNet, AlexNet's FatNet equivalent displayed more vivid kernels compared to the original AlexNet. Again proving that the FatNet equivalent of the model is trained in a similar convolutional manner in the early layers. It is also important to note that, unlike ResNet, most layers of AlexNet underwent the FatNet conversion, yet the first layer kernel trained to extract similar features.

AlexNet, being a small network is less efficient, particularly for complex problems like CIFAR-100. Not only the performance was lower with the test accuracy of 59.48%, but also it experiences a 7.13% decrease in accuracy (See Table 3.8) with the marginal smaller decrease when trained in the simulator. The performance drop came with the 3.4 times fewer convolution operations required to do the classification; this makes the Fat-AlexNet 3.4 times faster when both run on the same optical accelerator.

The training curves illustrated in Figure 3.14 indicate that Alex-FatNet took more epochs to converge. However, based on the validation set, the network had almost achieved its peak accuracy by epoch 100. This suggests that Alex-FatNet was somewhat more resilient to overfitting, as the training accuracy was considerably lower at epoch 100, while the

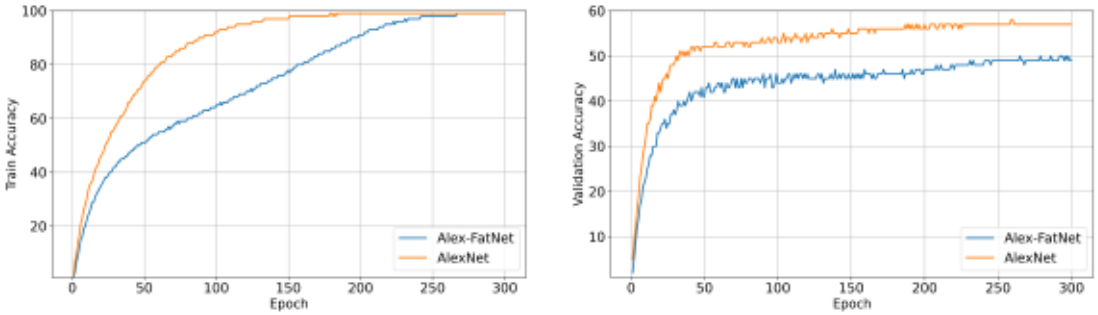


(a) (b)

Fig. 3.13 1st layer visualisation of kernels in (a)AlexNet and (b)Alex-FatNet

Table 3.8 Comparison of the test accuracy and number of convolution operations used in each AlexNet. Pseudo-negativity is taken into account in optical setup.

Architecture	Test Accuracy	Number of Conv Operations	Number of Conv Operations
	mean ± std		Ratio to Baseline
AlexNet	59.48 ± 2.48%	368,928	1 (baseline)
Alex-FatNet	52.35 ± 1.42%	107,640	0.29
Optical simulation Alex-FatNet	52.16%	215,340	0.58



(a) (b)

Fig. 3.14 **Training curves for AlexNet and Alex-FatNet with (a) training accuracy and (b) validation accuracy.** The validation accuracy seems to be similar despite Alex-FatNet being constantly lower. In contrast, training accuracy shows that Alex-FatNet took longer to reach the training accuracy of AlexNet.

validation accuracy was approaching its peak. Nonetheless, the validation accuracy continued to improve gradually until around epoch 250, when the training accuracy also peaked. In contrast, AlexNet reached its peak around epoch 150, after which both the training and validation accuracies did not show much improvement.

Similar to ResNet-18, AlexNet can be accelerated using optical Alex-FatNet over AlexNet on a GPU, only with the larger batch sizes such as 3136 (the maximum possible batch size for a 4k resolution in a 4f system) where the acceleration factor becomes 3.2 times (refer to Table 3.9). However, acceleration over the GPU was not possible at a batch size of 64.

Furthermore, AlexNet in optics did not experience any acceleration, and at a batch size of 3136, AlexNet on a GPU was only 1.06 times faster than AlexNet in optics. Although the slowdown is not significant, the energy efficiency of the 4f system can still be advantageous

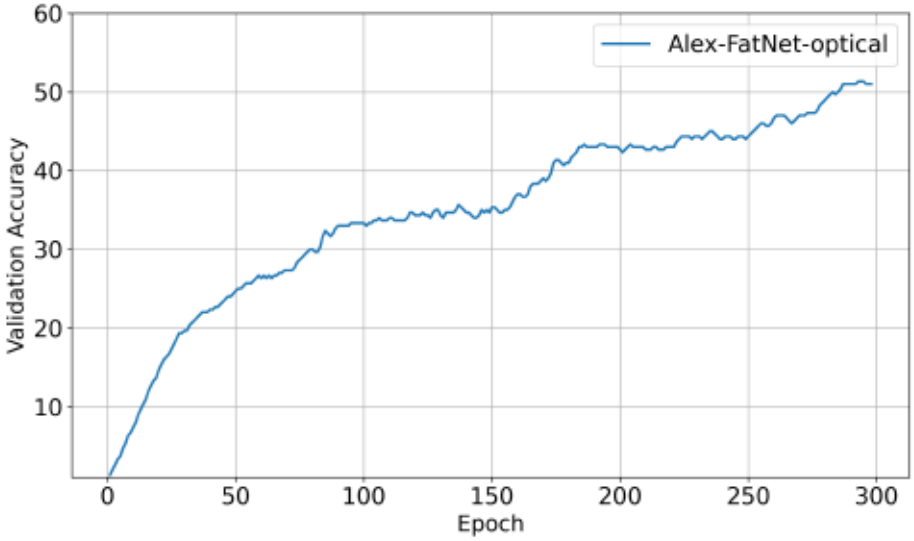


Fig. 3.15 **Training curve of the training done on Alex-FatNet in the optical simulator.**

if the speed of inference is not a primary concern. However, the energy efficiency of the 4f system is beyond the scope of this work.

Table 3.9 Inference time in seconds per input for AlexNet and FatNet with optics and GPU with a batch size of 64 and 3136 for cases when the 4k resolution of the 4f device is fully utilised. The frame rate of the 4f device is approximated at 2MHz [7].

Architecture	**Batch 64**	**Batch 3136**
AlexNet(GPU)	**1.1121e−4**	1.1094e−4
AlexFatNet(GPU)	2.0956e−4	1.3594e−4
AlexNet(Optics)	5.7645e−3	1.1764e−4
AlexFatNet(Optics)	1.6823e−3	**3.4334e−5**

3.4.3 VGG-19

Unlike ResNet and AlexNet, the first layer of VGG-19 consists of a convolutional layer with a small 3×3 kernel, which makes them harder to interpret. Therefore, after the visualisation, it is not easy to see distinct features in the kernels, but the similarity between VGG-19 and VGG-FatNet can still be observed. The arrows in Figure 3.16 point to obviously similar kernels, but even more similar kernels can be seen in the figure. Arrow 4 points to a kernel

that appears to be a flipped version of the original kernel. Nonetheless, it is evident that the weights of VGG-FatNet are not random and exhibit meaningful patterns.

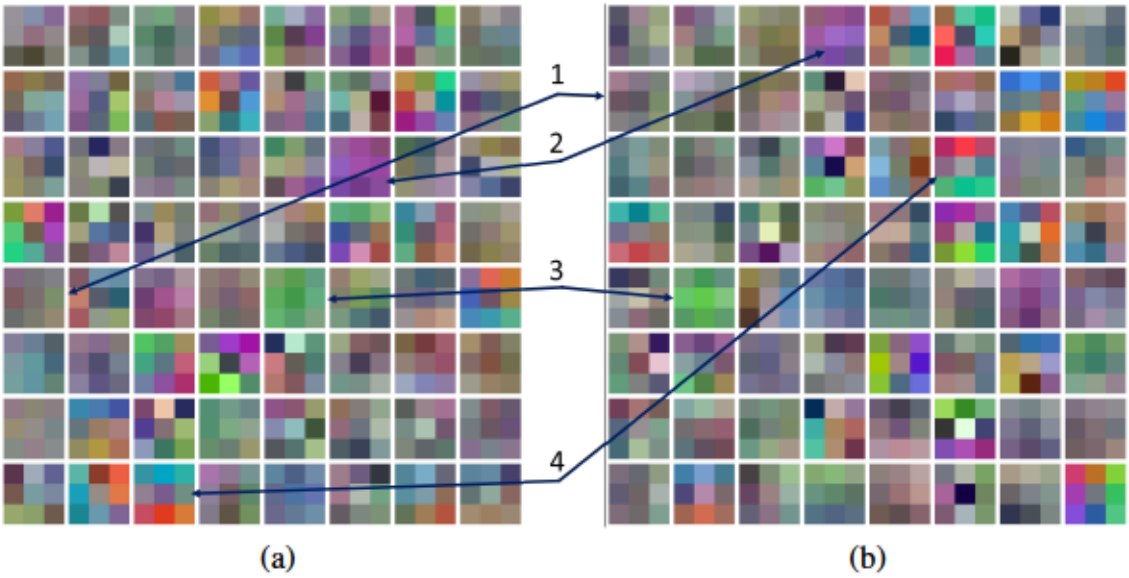


Fig. 3.16 **Visualisation of the kernels from the first layer of the (a) VGG-19 and (b) VGG-FatNet trained on CIFAR-100.** As the kernels are only 3×3, the similarity is not immediately obvious. Arrows indicate some clearly similar kernels or rotated like in case "4".

Similarly, VGG-19 exhibits around a 6% decrease in accuracy, akin to ResNet, but with 9.1 times fewer convolution operations (See Table 3.10). There was a smaller decrease in the test accuracy when the VGG-FatNet was trained in the simulator.

Moreover, as shown in Figure 3.17 the model trained in the simulator showed instability, and converged at around epoch 160. Since VGG-19 is a larger model than ResNet-18, training each epoch in the simulator required 130 minutes, resulting in a total training time of approximately 27 days. Although the real 4f system would not perform backpropagation through the simulation of light propagation, as done in this work, the purpose of this approach was to demonstrate that the model could still learn the weights in the simulator and perform inference.

Nevertheless, the VGG-FatNet's training process was comparable to, though slower than, that of the regular VGG-19, as demonstrated in Figure 3.18. This indicates that the architecture itself is robust and converges well.

The measured and calculated inference time for VGG-19 and its FatNet equivalent with optics and GPU were obtained and observed in Table 3.11. The observations were conducted based on the batch size of 64, and 3136 maximum utilisation of 4f system with 4k resolution modulators and camera, and the 2MHz system, as in the work Li *et al.* [7]. GPU

Table 3.10 Comparison of the test accuracy and number of convolution operations used in VGG-19. Pseudo-negativity is taken into account in optical setup.

Architecture	**Test Accuracy**		**Number of Conv Operations**	**Number of Conv Operations**
	mean	**std**		**Ratio to Baseline**
VGG-19	68 92%	1 31	2,211,840	1 (baseline)
VGG-FatNet	62 96%	1 80	248,283	0.11
Optical simulation VGG-19-FatNet	61 92%		496,566	0.22

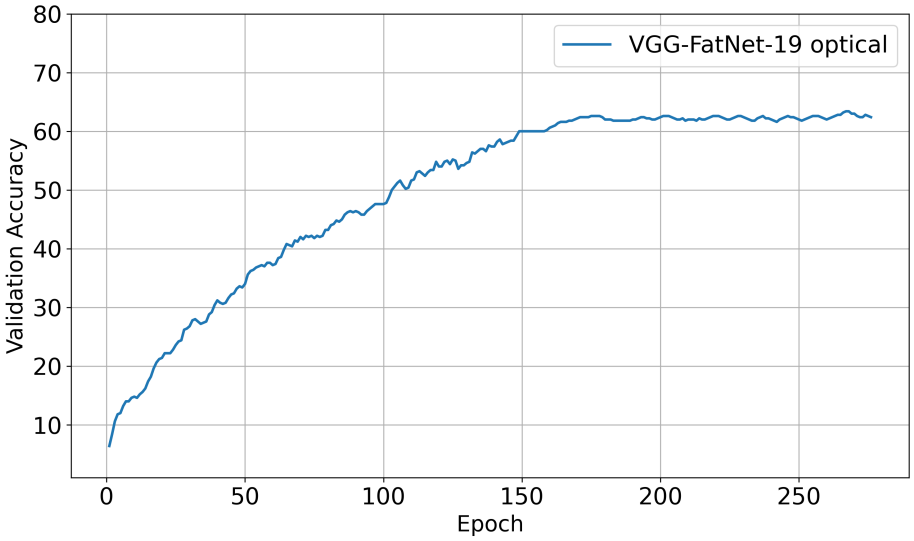
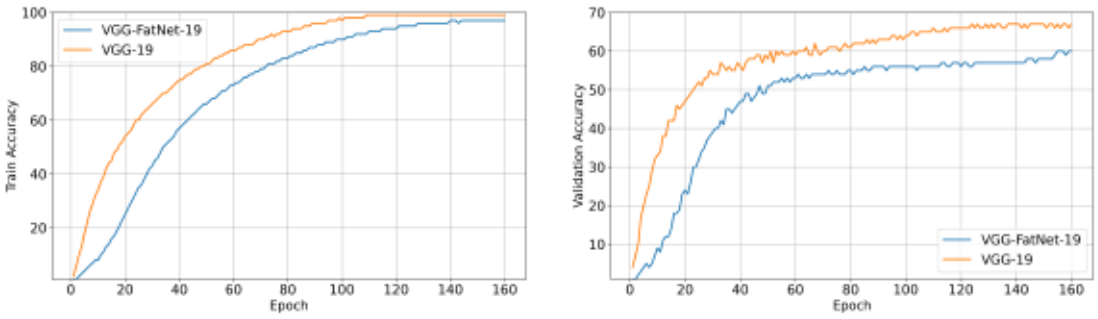


Fig. 3.17 **Training curve of the training done on Alex-FatNet in the optical simulator.** The curve is unstable, although it reaches a similar performance as the original VGG-FatNet-19.

measurements represent the average of 300 inferences, with the GPU warmed up over ten initial iterations. Notably, the VGG-FatNet in the optical setup with a batch size of 3136 is 2.23 times faster than VGG-19 on a GPU and 8.9 times faster than VGG-19 on the 4f system.

These results suggest that FatNet variants are more efficient with larger networks, making the conversion to FatNet more beneficial. Additionally, it is apparent that FatNet's efficiency increases with a larger number of classes, as the feature maps and kernels maintain higher resolution, enhancing performance in the 4F system.



(a) (b)

Fig. 3.18 **Training curves for VGG-19 and VGG-FatNet.** (a) training accuracy and (b) validation accuracy.

Table 3.11 Inference time in seconds per input for VGG-19 and FatNet with optics and GPU with a batch size of 64 and 3136 for cases when the 4k resolution of the 4f device is fully utilised. The frame rate of the 4f device is approximated at 2MHz [7].

Architecture	Batch 64	Batch 3136
VGG-19(GPU)	**2.5867e−4**	1.7703e−4
VGG-FatNet(GPU)	1.8463e−3	5.1951e−4
VGG-19(Optics)	3.4560e−2	7.0531e−4
VGG-FatNet(Optics)	3.8794e−3	**7.9172e−5**

3.5 Conclusion

This chapter introduced FatNet, a novel conversion framework for adapting regular CNN architectures to take advantage of the high-resolution capabilities of the 4f optical system. FatNet reduces the number of convolution operations while maintaining the same number of features and trainable parameters in each layer, addressing the critical challenge of minimising the readout operations—the primary bottleneck of the 4f system. This directly supports Objective 1, which aims to develop CNN architectures tailored for optical accelerators by reducing convolution operations and using the system's unique parallelism.

The chapter also detailed the development of the FatSpitter, an automated tool for converting PyTorch models into their FatNet equivalents, and described the implementation of a 4f system simulator. The simulator incorporated an enhanced pseudo-negativity method to handle optical constraints, thereby fulfilling Objective 2, which focuses on designing and implementing a simulator for the 4f system to benchmark the performance of these architectures.

FatNet was evaluated on the CIFAR-100 dataset using CNN architectures such as ResNet-18, AlexNet, and VGG-19, which were converted into their FatNet counterparts (Res-FatNet, Alex-FatNet, and VGG-FatNet). These FatNet architectures achieved significant reductions in convolution operations, with Res-FatNet requiring 8.2 times fewer operations than ResNet-18, Alex-FatNet requiring 3.44 times fewer operations than AlexNet, and VGG-FatNet achieving a 9.09-fold reduction compared to VGG-19. This optimisation aligns with Objective 1, as it demonstrates a clear reduction in convolution operations while maintaining reasonable accuracy metrics.

The performance trade-offs were also quantified to validate the speed-up gains. While the FatNet architectures resulted in faster inference times in the optical setup, there was a slight reduction in accuracy. For example, Res-FatNet experienced a 6% drop in accuracy compared to ResNet-18, Alex-FatNet saw a 7.13% decrease, and VGG-FatNet showed a 5.96% decrease.

Kernel visualisations revealed that Res-FatNet trained similarly on the shallow layers to the original ResNet-18, despite having larger kernel sizes and fewer channels. Although the kernels of Res-FatNet appeared slightly less vivid than those of ResNet-18, Alex-FatNet displayed more vivid kernel patterns than its traditional counterpart, AlexNet. These findings suggest that FatNet preserves the training characteristics of the original models while adapting them to the high-resolution capabilities of the 4f system

In conclusion, this chapter fulfilled Objectives 1 and 2 by demonstrating that FatNet architectures effectively adapt CNNs to the high-resolution capabilities of free-space optical accelerators, achieving substantial reductions in convolution operations and inference times for image classification tasks. Furthermore, the integration of the 4f simulator addressed key challenges, such as the lack of support for negative values. Although there is a minor trade-off in accuracy, the speed gains underscore FatNet's promise for optical computing. The next Chapter 4 extends the FatNet methodology to U-Net segmentation and explores alternative "Fat" architectures to examine their ability to prevent overfitting and maintain generalisation.

Chapter 4

Fat-U-Net for Image Segmentation

Overview

This chapter focuses on adapting the convolutional neural networks used for the image segmentation for the 4f free-space optical accelerator. It involves modifying the FatNet algorithm from Chapter 3 *Classification* to work with the U-Net model. Moreover, it also showcases the effectiveness of the FatNet algorithm in this application.

Since Chapter 3 focused on adapting CNN-based image classification for 4f free-space optical accelerators, it was noted that traditional classifier models are cone-shaped and naturally inefficient in free-space optics. In contrast, CNN-based segmentation models align better with the concept of FatNet, as segmentation tasks require detailed localisation information, making them more suitable for the 4f free-space optical architecture.

Note: The results in this chapter were presented at the SPIE West's 2024 *AI and Optical Data Sciences* conference [135]

The models and evaluation scripts are available at the following URL:
https://github.com/riadibadulla/FatUnet (accessed 25.09.2024)

4.1 Introduction to Image Segmentation

Image segmentation is a crucial task in many computer vision and image processing applications. It refers to dividing a digital image into multiple overlapping regions, where each region corresponds to a class and normally represents an object of interest [136]. Segmentation aims to simplify and transform the representation of an image into something that is more meaningful and easier to process. From one point of view, image segmentation can be described as pixel-wise classification [137].

There are several major techniques for image segmentation. Thresholding can be used for easier tasks, by separating foreground and background based on pixel intensities [138]. Edge detection finds objects' edges by looking for discontinuities and alterations in intensity. Region-based segmentation divides pixels into connected regions based on similarity [139]. Clustering methods, like the k-means algorithm, divide pixels into clusters based on feature similarity [140]. Graph-based techniques model images as graphs and segment them by graph partitioning [141]. Active contour methods iteratively evolve curves to fit to object boundaries [142]. Neural network-based approaches leverage deep learning to perform semantic segmentation, as described in Section 2.5. Additional techniques include watershed transformation, supervised versus unsupervised methods, global versus local segmentation, and more [143].

Each approach has its own advantages and drawbacks. The choice of segmentation method depends on factors like the application, imaging modality, and desired output. Moreover, the task of image segmentation can also be divided into two categories: semantic segmentation and instance segmentation. Instance segmentation refers to not just separating and classifying pixels into semantic classes, but also differentiating between individual object instances.

The introduction of deep learning in computer vision applications has completely changed how digital images are processed and analysed. However, as discussed in the previous chapter 2.8, the computational demand for image segmentation and the difficulty of real-time applications grow with the complexity of the models. Due to this, in this chapter, we examine the implementation of FatNet for segmentation tasks, specifically U-Net. Since U-Net is not a cone-shaped classifier network, for which the FatNet conversion was initially designed 3, converting an encoder-decoder style network presents its own challenges that need to be analysed.

4.2 Methodology

U-Net is a fully convolutional neural network developed for biomedical image segmentation by Ronnenberg *et al.* [13]. The architecture consists of a contracting encoder path and an expanding decoder path. The encoder and decoder paths stacked together resemble a "U" shape, hence the name U-Net. The encoder follows the standard architecture of a convolutional network used for the classification, with repeated convolution blocks of convolutional layers, activations, and max pooling operations to downsample the input. This part follows a cone-shaped architecture described previously in the classification chapter 3.2.1. The decoder gradually upsamples the encoder output using transposed convolutions, also

known as deconvolution layers, and concatenates it with high-resolution features from the encoder via skip connections. The encoder and decoder paths in U-Net are symmetrical, except that the first feature maps in each decoder's convolution block are doubled in depth. This is because they concatenate features both from the previous decoder layer and the corresponding encoder layer via skip connections. This architecture enables the model to leverage both contextual and localised information to make precise segmentation predictions.

Before U-Net, CNNs were used for segmentation and lacked one of two main features: they either followed the cone shape and focused on the contextual information, like in the work of Long *et al.* [64] or focused more on the localisation information like in the work of Noh *et al.* [65] by having encoder-decoder architecture without the skip connections.

The underlying principle of FatNet conversion is to maintain the constant number of trainable parameters and pixels in each layer while increasing the resolution of feature maps and kernels and decreasing the number of channels in each layer. By making this conversion, the network takes full advantage of the high-resolution capabilities of the 4f system, thereby optimising its performance and efficiency in the context of free-space optical acceleration.

The original FatNet conversion described in the previous Chapter 3.2.1 designed specifically for the classification task, maintains the same architecture as the original network until the feature maps are pooled down to the resolution with a number of pixels less than or equal to the number of classes. It is posited that when it comes to the FatNet conversion for the segmentation, pooling may be unnecessary, and the input resolution can be preserved throughout the entire network. Consequently, increasing the resolution of kernels while keeping the resolution of the feature maps constant would decrease the feature map-to-kernel resolution ratio, emulating the effect of pooling the feature maps without actual pooling implementation. This approach can significantly increase the inference time of the network run on the 4f free-space optical accelerator and hypothetically retains localisation accuracy even more effectively.

Since the original FatNet was designed for classification, only the contracting path of the U-Net was converted into the FatNet. Table 4.1 presents the U-Net equivalent of the FatNet construction table for images of 160×160. The table is used to compute the number of weights per layer, excluding the bias and the number of pixels per layer. The algorithm ensures the convolutional layers with the same number of input and output channels within convolutional blocks have an equal number of input and output channels after the conversion too. Upon completing the conversion of the contracting path of the U-Net into FatNet, the path was mirrored to generate the "expanding path", and the kernel sizes were recalculated to match the number of weights from the original layers. Since the so-called expanding path of

Table 4.1 Construction table for Fat-U-Net's first half out of the U-Net's contracting path.

U-Net contracting		weights	pixels	New layers		FatU-Net adjusted	
Channels	kernel			Channels	kernel	Channels	kernel
3×64	3	1,728	1,638,400	3×64	3	3×32	5
64×64	3	36,864	409,600	32×32	6	32×32	6
64×128	3	73,728	819,200	32×32	9	32×16	12
128×128	3	147,456	204,800	16×16	24	16×16	24
128×256	3	294,912	409,600	16×16	34	16×8	48
256×256	3	589,824	102,400	8×8	96	8×8	96
256×512	3	1,179,648	204,800	8×8	136	8×10	122
512×512	3	2,359,296	51,200	10×10	160	10×10	160
512×1024	3	4,718,592	102,400	10×18	160	10×20	154
1024×1024	3	9,437,184	102,400	20×20	160	20×20	160

the Fat-U-Net does not actually require upsampling, deconvolution operations were replaced with simple 3×3 convolutions, as illustrated in Figure 4.1(b).

4.2.1 Intuitive Fat-U-Nets

In Chapter 3 and in this chapter the same FatNet conversion technique, as described in Section 3.2.1, was applied with minor variations. In all models, the original model and its FatNet equivalent share a key similarity: they maintain the same number of trainable parameters and the same pixel count in each layer's feature maps. It was hypothesised that these networks would train similarly. However, it is important to note that maintaining the number of trainable parameters might have been more crucial for preserving FatNet's high performance. Ideally, a network with an excessive number of parameters could lead to overfitting, while one with too few parameters might result in underfitting. Assuming that maintaining the same number of trainable parameters is crucial, it must also be considered whether preserving the same number of pixels in the feature maps is equally crucial.

To explore this question, the Fat-U-Net was compared with three versions termed Intuitive Fat-U-Nets (*Intuitive Fat-U-Net 1, Intuitive Fat-U-Net 2, Intuitive Fat-U-Net 3*), which considered the number of trainable parameters but disregarded the pixel count in the feature maps, when converting the network. The architectures of all three Intuitive Fat-U-Net versions can be seen in the Table 4.2. This conversion was not performed using the Fat-

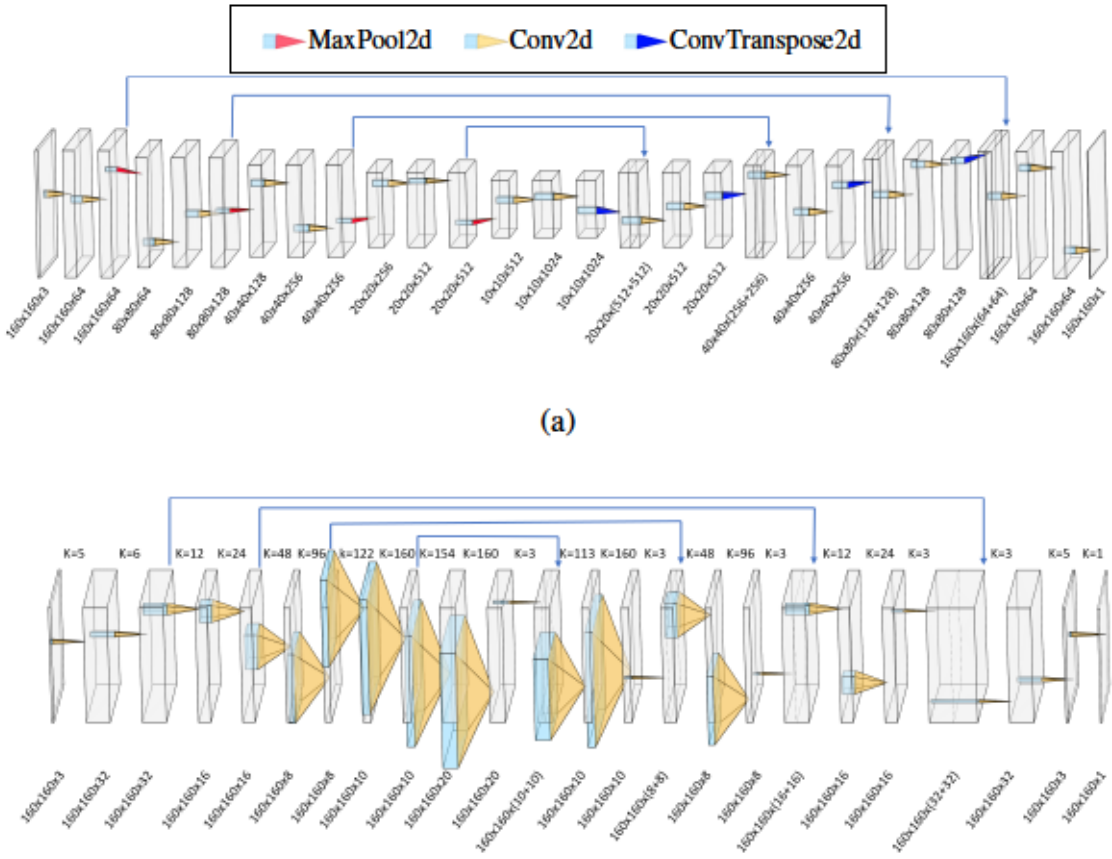


Fig. 4.1 **Graphical representation of our implementation of U-Net and Fat-U-Net architectures.** (a) U-Net architecture, with all kernel sizes 3×3, MaxPool with kernels size of 2×2 and deconvolution operations with a kernel size of 3×3. (b) Fat-U-Net architecture derived from our implementation of U-Net, with the varying kernel sizes indicated as K at each layer. The resolution of the feature maps stay constant throughout the network. Ration of kernel to feature map is preserved between U-Net and Fat-U-Net.

Spitter 3.2.2, 3.2.1; instead, it involved increasing the kernel sizes and reducing the number of channels to preserve the number of trainable parameters. *Intuitive Fat-U-Net 1* closely resembles the original U-Net, with the kernel expanding to a maximum of 24×24 and the number of channels reduced to only 128×128 in the bottleneck. *Intuitive Fat-U-Net 3* deviates most from the original U-Net but features the largest kernel size of 153×153 and a kernel size of 20×20, making it the most efficient network for the 4F system discussed in this chapter. Meanwhile, *Intuitive Fat-U-Net 2* has intermediate features, with the bottleneck channels reduced to 64×64 and the kernel size expanded to 48×48.

Table 4.2 Comparison of the architectures of the Intuitive Fat-U-Nets. Unlike a Fat-U-Net, which is converted using a FatNet algorithm for the conversion, these intuitive networks were developed manually by choosing smaller channel sizes and computing the new kernel sizes without taking into account the number of pixels in the feature map.

Layer	Intuitive Fat-U-Net 1			Intuitive Fat-U-Net 2			Intuitive Fat-U-Net 3		
	Channels		Kernel	Channels		Kernel	Channels		Kernel
Conv block 1	3	8	8	3	4	12	3	4	12
	8	8	24	4	4	48	4	4	48
Conv block 2	8	16	24	4	8	48	4	8	48
	16	16	24	8	8	48	8	8	48
Conv block 3	16	32	24	8	16	48	8	10	61
	32	32	24	16	16	48	10	10	77
Conv block 4	32	64	24	16	32	48	10	16	85
	64	64	24	32	32	48	16	16	96
Conv block 5	64	128	24	32	64	48	16	20	121
(bottleneck)	128	128	24	64	64	48	20	20	153
DeConv 1	128	64	3	64	32	3	20	16	3
Conv block 6	128	64	24	64	32	48	32	16	96
	64	64	24	32	32	48	16	16	96
DeConv 2	64	32	3	32	16	3	16	10	3
Conv block 7	64	32	24	32	16	48	20	10	77
	32	32	24	16	16	48	10	10	77
DeConv 3	32	16	3	16	8	3	10	8	3
Conv block 8	32	16	24	16	8	48	16	8	48
	16	16	24	8	8	48	8	8	48
deconv4	16	8	3	8	4	3	8	4	3
Conv block 9	16	8	24	8	4	48	8	4	48
	8	3	24	4	3	55	4	3	55
segmenter	3	1	1	3	1	1	3	1	1

4.2.2 U-Net without skip connections

Our investigation was driven by the hypothesis that the Fat-U-Net without skip connections would outperform the standard U-Net architecture without skip connections or significantly narrow the performance gap between them.

The primary advantage of Fat-U-Net lies in its ability to maintain high-resolution feature maps throughout the network. By eliminating pooling operations—which, in traditional U-Net architectures, reduce spatial resolution to decrease computational load and expand the receptive field—Fat-U-Net enhances its capability for precise localisation. This structural enhancement is crucial for maintaining the integrity of spatial information, which is essential for accurate segmentation.

Additionally, Fat-U-Net employs larger kernel sizes, allowing for a broader receptive field at each layer. This design feature helps capture more contextual information from the input image, preserving detailed spatial relationships that are vital for nuanced segmentation tasks.

In the experiments, both the traditional U-Net without skip connections and the modified Fat-U-Net were tested across various segmentation tasks. It was hypothesised that the absence of skip connections, combined with Fat-U-Net's high-resolution feature maps and large kernels, would reduce performance but make Fat-U-Net more stable compared to U-Net by preserving spatial details more effectively.

4.3 Experiments

U-Net and its Fat-U-Net equivalent were implemented and tested in two segmentation tasks of the Oxford-IIIT pet and HeLa cells.

For the HeLa cells, the region of interest (ROI) of a single cell was used for training and evaluation, while large fields containing 8000 8000 cells were used for qualitative results.

The version of Fat-U-Net used in this work is optimised for 160 160 image inputs. Therefore, patches were prepared from odd-numbered slices of the ROI, each with a 50% overlap. Since only half of the slices were used, this resulted in 529 patches per slice and a total of 79,350 image pairs with their corresponding ground truth masks. All patches underwent Gaussian low-pass filtering before being saved. A process was repeated during the evaluation of new data.

To avoid bias in the training or test sets—particularly due to the inclusion of an excessive number of background images—random shuffling of patches was replaced with a per-slice strategy for the train-test split. Test slices included every tenth slice starting from 1 up to 291, such as slices 1, 11, 21, ..., 281, 291. Slices 5, 25, 45, ..., 285 served as validation

slices. Initially, the remaining slices were designated for training. However, to counteract data imbalance caused by background-only images in the shallowest and deepest slices, the training set was limited to slices 97 through 183, excluding every slice ending in 1 or 5. This approach resulted in 26 slices being used, yielding 13,754 training patches (e.g., 97, 99, 103, 107, 109, ..., 177, 179). Although the training dataset might seem small, it is adequate for effective binary nucleus segmentation.

As mentioned before in Section 2.11, performance was evaluated using pixel-wise accuracy, mIoU (mean IoU), and Dice Score:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{4.1}$$

$$IoU = \frac{TP}{TP+FP+FN} \tag{4.2}$$

$$DiceScore = \frac{2TP}{2TP+FP+FN} \tag{4.3}$$

Moreover, the inference times of all networks were compared to demonstrate the potential acceleration when running on a 4f free-space optical device.

The training regime can be seen in the Table 4.3. Since Fat-U-Net had to remain consistent across both datasets, a standard resolution of 160×160 was selected. The dropout has been applied only for the Hela cell training in the bottleneck section of the U-Net. Both datasets were normalised by subtracting the mean and dividing by the standard deviation of each RGB channel. Same learning rate was used for both U-Net and Fat-U-Net, using cosine annealing scheduler.

Experiments were performed for U-Net and Fat-U-Net with both datasets. The three versions of Intuitive-Fat-U-Net were also trained on both datasets. Upon success, more experiments were conducted on the Hela Cells dataset, including optical simulation of Fat-U-Net and a comparison of U-Net/Fat-U-Net without skip connections.

Compared to other models, the optical training posed significant challenges, particularly in memory management. Since the optical simulation is not memory efficient and must store extensive data to simulate light propagation, the batch size had to be reduced to 2. Unfortunately, such a small batch size caused instability during training. To address this issue, the gradient accumulation technique was employed (see Section 2.10), with a step of 16, to emulate a batch size of 32. Although this technique stabilised the training, it resulted in poor statistical estimation during batch normalisation, necessitating the removal of tracking statistics in batch normalisation. Unfortunately, all these adjustments led to the model converging after a significantly higher number of epochs (300).

Table 4.3 Experimental Setup for Hela Cell and Oxford-IIIT Pet Datasets

Parameter	**Hela Cell Dataset**	**Oxford-IIIT Pet Dataset**
Initial Learning Rate	1e-3	1e-4
Scheduler	cosine annealing	cosine annealing
Batch Size	32	16
Number of Epochs	20	250
Input Resolution	160×160	160×160
Regularisation	Dropouts (50%), Weight Decay 1e-4	Weight Decay 1e-4
Data Augmentation	None	ShiftScaleRotate, RGBShift, RandomBrightnessContrast, Normalisation
Optimiser	Adam	Adam
Normalisation	Mean: 0.6379, STD: 0.0855	Mean: R-0.485, G-0.456, B-0.406, STD: R-0.229, G-0.224, B-0.225

4.4 Results and Discussion

The inference speed was assessed on a GPU and estimated for optics to highlight the efficiency gains achievable with a 4f free-space optical device. In this comparison, while our 5-staged U-Net model requires 3,833,984 convolutional operations, its FatNet equivalent, Fat-U-Net, needs only 7,123 convolution operations. Given that inference speed in the 4f optical setup is not affected by resolution changes, Fat-U-Net's optical inference is projected to be **538 times faster** than that of U-Net when both run in optics.

The inference time was measured on an Nvidia A100 for both U-Net and Fat-U-Net, and the results were compared to the theoretical inference time calculated for a 4f optical accelerator, based on the work of Li *et al.* [7], where the authors predicted the availability of a 2 MHz device in the near future. Similarly, Gupta *et al.* [88] conducted their evaluations with the assumption of a 2 MHz device. The results are shown in Table 4.4 for batch sizes of 1, 32, and 144. The batch size of 144 was chosen because it is the maximum possible batch size for the 4f system with 4k resolution if batch tiling is applied, with the necessary padding for each image.

Based on the results in Table 4.4, at the batch sizes of 1,32 and 144,the acceleration of inference of Fat-U-Net with 4f optics, compared to U-Net run on high-end GPU, is 1.32, 8.27, and **37 times** respectively. It is also important to note that with this setup and batch

size up to 144, it is impossible to get any acceleration if the vanilla U-Net is inferred on the 4f system without conversion into its FatNet equivalent.

Table 4.4 Inference time in milliseconds of U-Net and its FatNet equivalent (Fat-U-Net) model per image with different batch sizes run on 4f accelerator and Nvidia A100. The frame rate for 4f system was approximated at 2 MHz, and Nvidia A100 GPU was measured experimentally.

Model and device	Batch 1	Batch 32	Batch 144
U-Net (Optics)	1920.00	59.900	13.300
Fat-U-Net (Optics)	3.46	0.108	**0.024**
U-Net (GPU)	4.55	0.894	0.883

Having demonstrated Fat-U-Net's enhanced speed, the focus then shifted to assessing its effectiveness. Initially, U-Net was trained to achieve state-of-the-art performance, after which it was transformed into Fat-U-Net. The U-Net version was surpassed only by models that integrated pre-trained VGG16 and Inception V3 as their contracting paths, as shown in Table 4.5, though the performance difference was minimal. Given that the U-Net model in this work was developed from scratch without pre-training, it is clear that the necessary benchmarks were met prior to its conversion. The transition to Fat-U-Net resulted in a modest decline of 1.93% in pixel accuracy, 4.24% in IoU, and 2.46% in Dice coefficient, showcasing relatively minor performance compromises compared to a 6% accuracy reduction observed in classification tasks.

Table 4.5 Comparison of the evaluation results of the accuracy, mIoU, and Dice score of U-Net and its Fat-U-Net equivalent along with other works for Oxford-IIIT Pet.

Model	Accuracy (%)	IoU (%)	Dice Score (%)
U-Net (our implementation)	**95.33**	89.32	**94.33**
Fat-U-Net (ours)	93.40	85.08	91.87
SEU-Net [31]	-	$\approx$ 77.00	-
ICNet [116, 115]	90.79	75.12	-
ConRec (20% of dataset) [113, 144]	-	-	90.00
U-Net (as per Sundarrajan *et al.*) [114]	-	33.30	46.40
U-Net+VGG16 [114]	-	89.40	94.20
U-Net+InceptionV3 [114]	-	**91.60**	91.50

From Figure 4.2, it can be observed that the training process is similar for both U-Net and Fat-U-Net, with the Fat-U-Net performing slightly lower by a constant factor. However, it is also evident that neither curve has fully converged and both continue to rise slowly, indicating that even higher accuracy could be achieved with additional training.

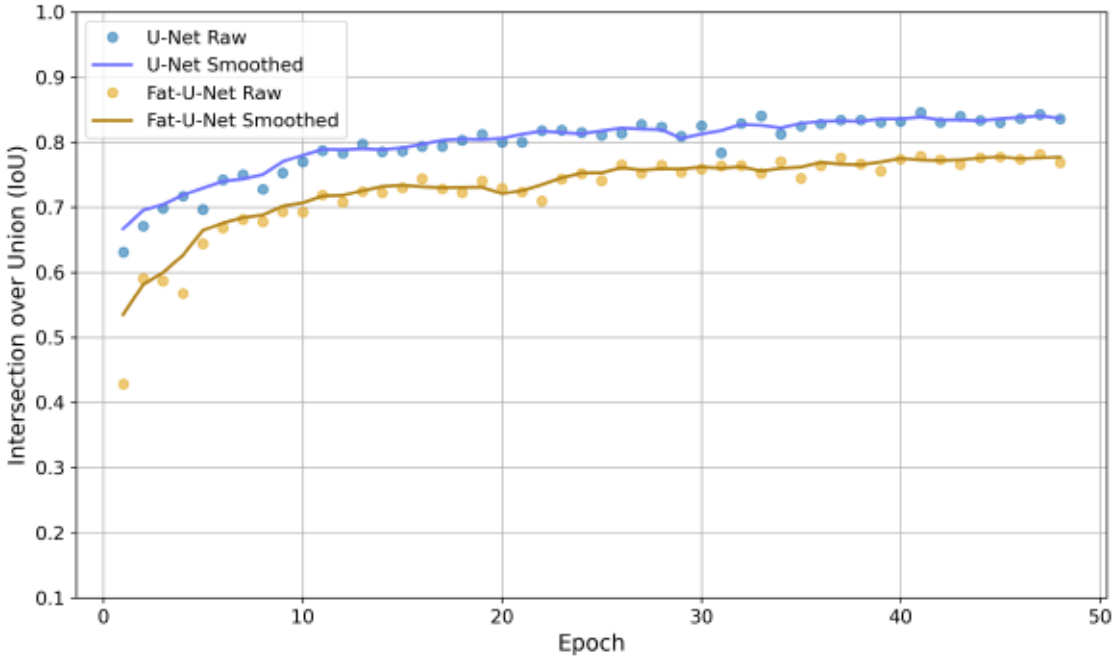


Fig. 4.2 **Training curve for Oxford-IIIT Pets validation dataset, trained with U-Net and Fat-U-Net.** A smoothed curve is plotted on top of the values. Both curves are correlated, while the Fat-U-Net is constantly lower than U-Net.

The qualitative comparisons presented in Figure 4.3 highlight the different performance of U-Net and Fat-U-Net across several cases. In Figure 4.3(a), both algorithms achieve flawless segmentation when the pets are clearly displayed against a uniform background. Figure 4.3(b) shows U-Net's superior ability to segment a background cat that falls outside the region of interest, whereas Fat-U-Net does not perform as well. However, with the case of the cat in the background, it managed to flawlessly segment the cat in the ROI. Conversely, Figure 4.3(c) showcases Fat-U-Net's strength, as it accurately segments both animals, unlike U-Net, which mistakenly identifies parts of the cat and dog as part of the background.

Poor segmentation performance can be observed in Figure 4.3(d), where the scale of the pet is notably smaller than that typically observed within the dataset. This deviation in scale leads to both algorithms failing to accurately segment the cat in the first image; U-Net tends towards over-segmentation, while Fat-U-Net is inclined to under-segmentation. Furthermore, in the analysis of the second image, both algorithms incorrectly classify the

human hand as part of the cat, highlighting a common challenge in distinguishing between closely interacting objects within the segmentation task.

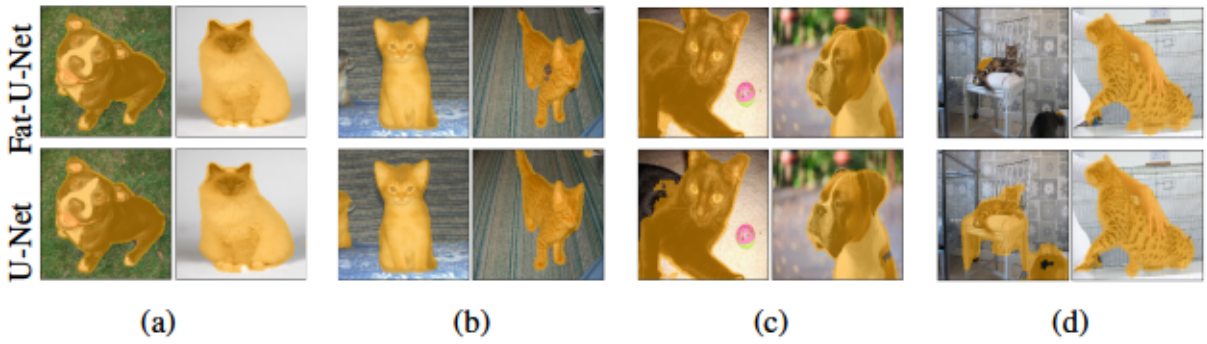


Fig. 4.3 **Qualitative results of Oxford IIIT Pet dataset.** (a) Examples of perfect segmentation by both algorithms. (b) Examples of U-Net performing better than Fat-U-Net. U-Net managed to segment a cat in the background, even though that wasn't part of the ground truth. Moreover, in the top right picture, Fat-U-Net missed pixels in the middle of the cat. (c) Examples of Fat-U-Net outperforming U-Net. U-Net missed the left-side pixels on the cat picture and a tiny bit of the pixels on the back of the dog. (d) Bad segmentation examples by both algorithms. U-Net over-segmented the cat, while Fat-U-Net under-segmented it in the left picture. In the right picture, both algorithms classified the hand as part of the animal.

Figure 4.4 shows the training curve comparison of U-Net and Fat-U-Net trained with HeLa cells. From this figure, it can be observed that U-Net converges faster than Fat-U-Net with minimal performance increase over the epochs. Fat-U-Net, on the other hand, demonstrates a very steep training curve over 20 epochs. In comparison to regular slow-learning models, Fat-U-Net is even steeper, primarily due to its sensitivity to the learning rate. It has been observed that the FatNet equivalents usually require smaller learning rates than their counterparts. In these experiments, a cosine annealing learning rate scheduler was used, resulting in performance improvements over the epochs, with the most significant step occurring at epoch 10. After epoch 10, the network quickly reached a performance level very close to that of U-Net. It can also be noted that the gap between the curves continues to shrink after epoch 10.

As discussed in the Experiments section 4.3, the optical simulation required more epochs, as shown in Figure 4.5. Compared to the original Fat-U-Net, the training process was extended; however, the learning curve appears smoother, with the IoU continuing to gradually increase even after epoch 95.

A tiny instability can be seen in the first three epochs, where the accuracy suddenly dropped. This did not affect the overall training process and possibly happened because of the high learning rate during the initial epochs.

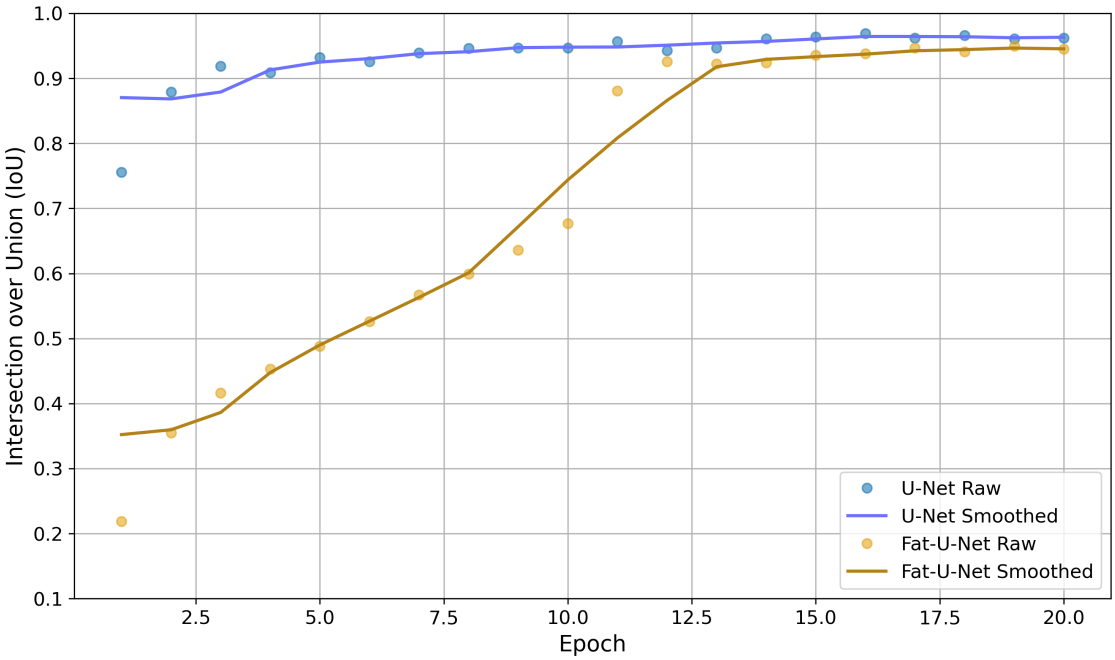


Fig. 4.4 **Training curves for HeLa cells validation dataset, trained on U-Net and Fat-U-Net.** The smoothed curve is plotted on top of the original values. The U-Net quickly reaches the desired performance around epoch 7, while the Fat-U-Net starts with a very low IoU but eventually reaches a similar performance as the U-Net at around Epoch 12. Over time, the performance of Fat-U-Net continues to approach that of U-Net.

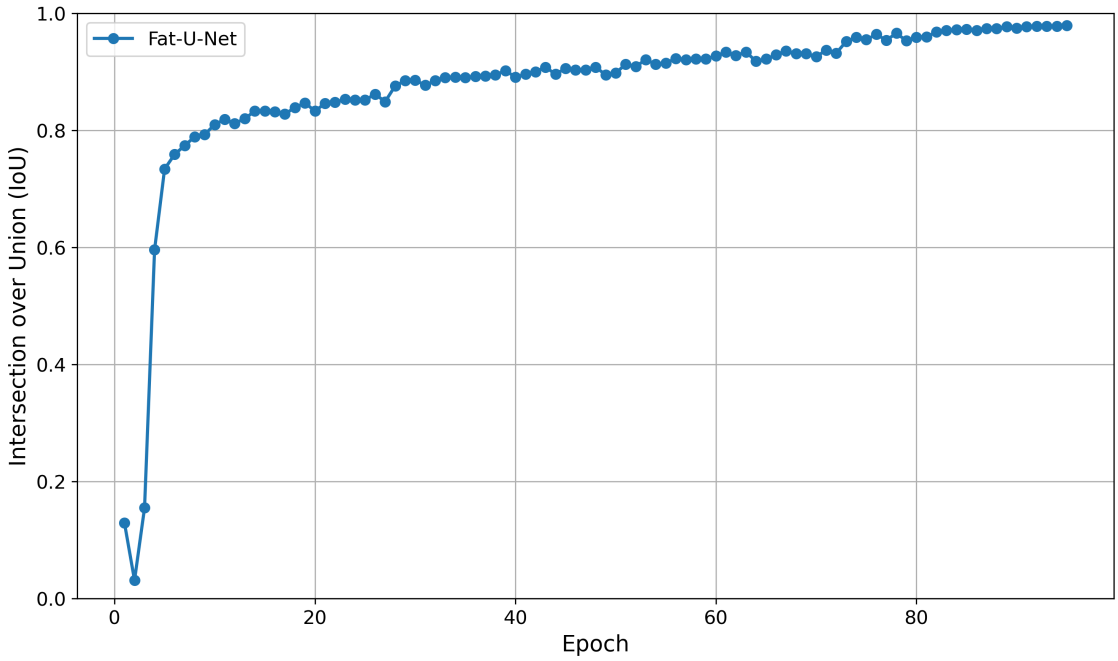


Fig. 4.5 **Training curves for validation set of Optical simulation of Fat-U-Net trained on HeLa cells.** A smoothed curve is plotted on top of the values. Optical simulation of Fat-U-Net took fast to reach the performance of IoU of 0.8. But overall, due to the complexity of the simulation, it took 95 epochs to reach the desired performance, and the projection indicates that it may continue to improve.

The evaluation of U-Net and Fat-U-Net models for HeLa cells nucleus segmentation encompassed four different testing conditions: (1) analysing all odd-numbered slices from 1 to 300, (2) focusing on a subset of odd slices (150-200) where the nucleus is prominently visible, followed by (3) and (4) applying these approaches to the test slices, as detailed in Table 4.6. The results of the first two scenarios, evaluation with the test set and the entire dataset, including training and validation slices too, were comparable to the results of the work of Karabag *et al.* [8] since no test split was conducted in that work.

Model	Acc.(all) (%)	IoU(all) (%)	Acc.(150-200) (%)	IoU(150-200) (%)
U-Net	95.71	66.32	99.59	**97.15**
U-Net (Test)	95.75	66.59	99.57	97.11
Fat-U-Net	95.31	64.27	99.42	96.05
Fat-U-Net (Test)	95.42	64.83	99.43	96.25
Opt Fat-U-Net	**95.94**	**66.91**	99.27	95.01
Opt Fat-U-Net (Test)	95.60	65.34	99.33	95.57
4 stage U-Net [8]	93.46	51.38	**99.66**	97.12

Table 4.6 Performance Comparison of our implementation of five staged U-Net, its Fat-U-Net equivalent, and a 4 staged U-Net implementation by [8]. Evaluating Accuracy and IoU Metrics Across the entire dataset and 150-200 range for all odd and test slices that have not participated in the training process.

Given that ground truth (GT) annotations were available only for ROI of $2000 \times 2000 \times 300$ voxels — a single cell within the extensive $8192 \times 8192 \times 518$ dataset — qualitative evaluations were conducted by training on one cell and testing with that cell and adjacent cells as shown in the ground truth visualisation in Figure 2.18. Additionally, qualitative evaluation extended to the segmentation of the full-sized original image of 8192×8192 containing all the cells as shown in Figure 4.7. When evaluating with the larger 8192×8192 image, Fat-U-Net provided better results than U-Net, particularly noticeable in the dividing cell located at the bottom right. The red and green arrows in Figure 4.7 indicate instances of good and poor segmentation. Overall, it can be observed that Fat-U-Net segments the cells more accurately, fitting the nuclear envelope tightly. However, unlike U-Net, Fat-U-Net contains several false positives. Additionally, both models incorrectly segmented a dead cell in the top right corner as a nucleus.

The quantitative evaluation of Fat-U-Net for segmenting the nuclei of HeLa cells was successful. When compared to the 4-stage U-Net model reported by Karabag *et al.*[8] in 2023, our 5-stage version of U-Net implementation demonstrated marginally better performance

on middle-range slices and achieved a 14.94% increase in Intersection over Union (IoU) across all slices. It is important to consider that the ground truth for region of interest (ROI) cells was limited to the segmentation of the nucleus of the central cell, treating the nuclei of the surrounding cells as background. However, both U-Net and Fat-U-Net were capable of segmenting these nuclei despite the noisy labelling in the ground truth data, as illustrated in Figure 4.6. As a result, the segmentation mask surpassed the ground truth for side slices (outside the 150-200 range), leading to a reduced IoU for all slices in comparison to those in the middle range (See Figure 4.8). Converting to Fat-U-Net resulted in a smaller decline in performance than observed in the evaluation with the Oxford-IIIT Pet dataset, with a decrease of about 1% for middle-range slices and 2% for all slices.

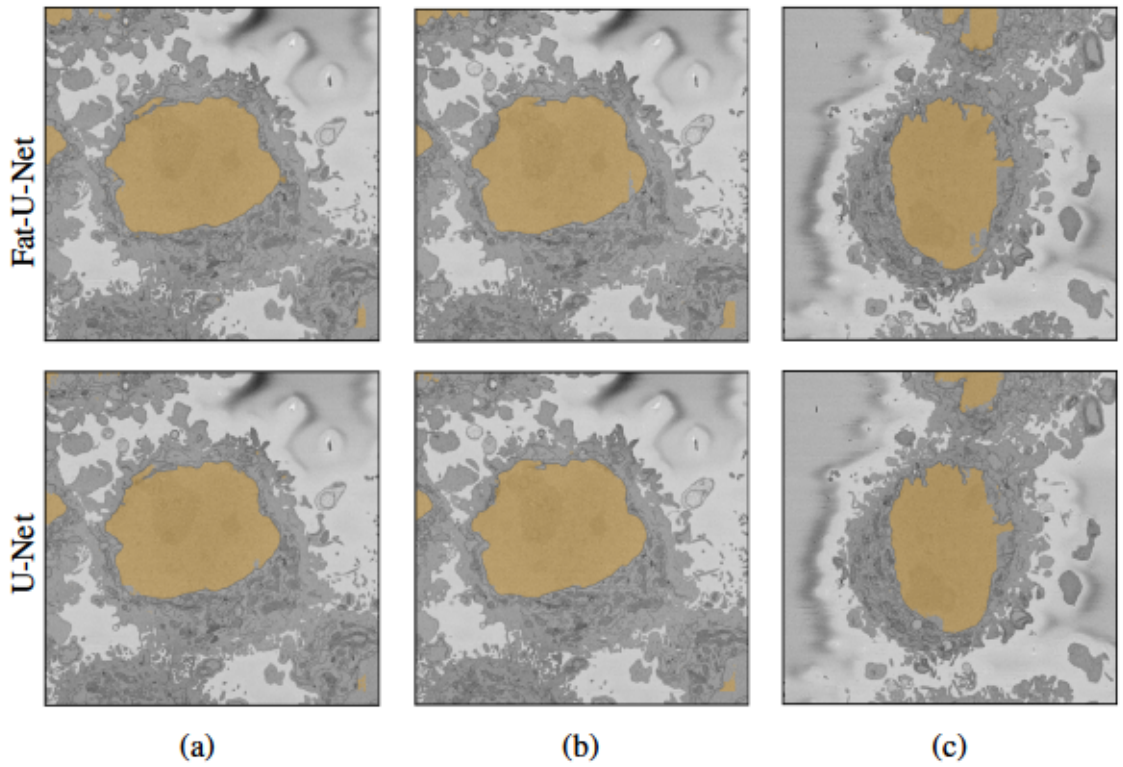


Fig. 4.6 **Qualitative results of HeLa dataset.** (a) train slice 119 (b) test slice 121 (c) Unseen cell, taken from the larger field.

4.4.1 Intuitive Fat-U-Nets

When analysing the results of the Intuitive Fat-U-Net experiments, it became apparent that the performance of Fat-U-Nets aligned with the hypothesis that preserving the number of pixels in the feature maps of the FatNet contributes to better performance. These Fat-U-Nets did

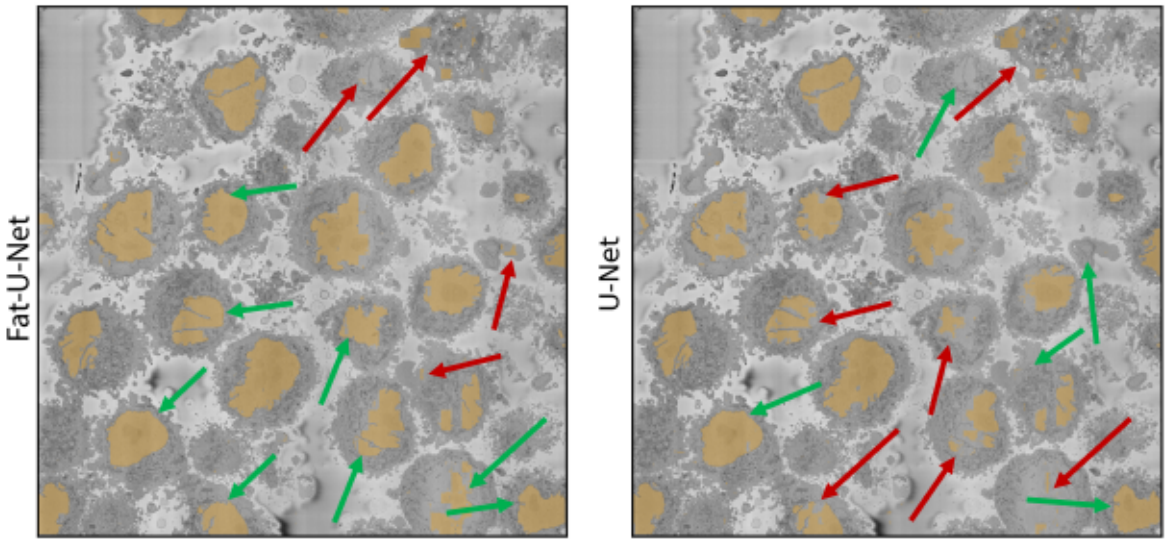
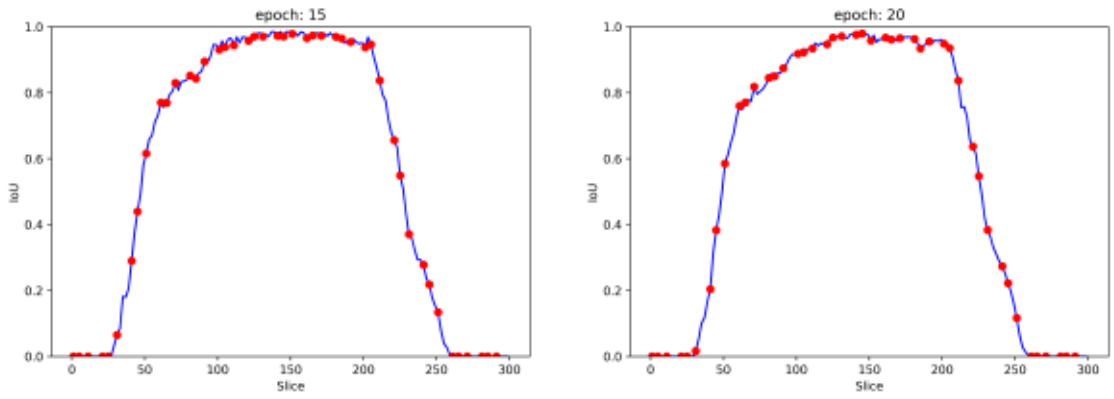


Fig. 4.7 U-Net and Fat-U-Net segmentation qualitative results on 8192×8192 images. Green arrows indicate areas of good performance, while red arrows highlight areas of poor performance.



(a) (b)

Fig. 4.8 Intersection over Union (IoU) of each slice of the HeLa cells dataset by both models: (a) U-Net and (b) Fat-U-Net. The red points indicate the test data slices whose patches were not included in the training process. It is evident that both models perform similarly poorly for shallow and deeper layers, where the nucleus is either small or non-existent, and perform comparably well for middle slices.

not follow the Fat-Spitter algorithm. Instead, they only considered the number of trainable parameters rather than the number of pixels in each feature map.

As demonstrated in Table 4.7, our standard Fat-U-Net outperformed all three Intuitive Fat-U-Nets, confirming that the number of pixels in each feature map is crucial for the FatNet conversion and its efficacy.

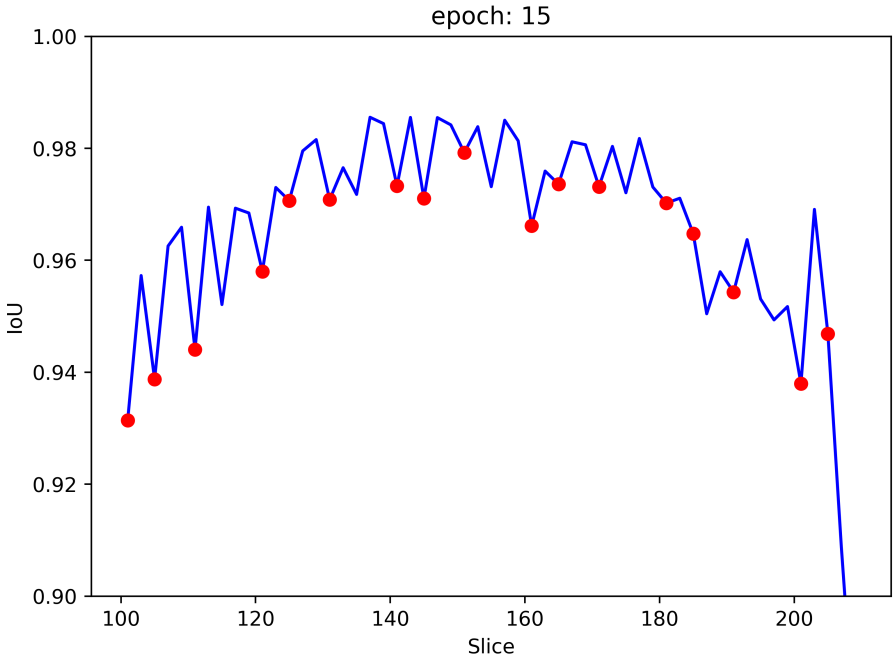


Fig. 4.9 **Closer look on line graph of IoU metric on each slice of the HeLa cells dataset.** The line graph provides a detailed view of the IoU metric for each slice of the HeLa cells dataset. The red points represent the test data slices whose patches were excluded from the training process. These test slices exhibit a slightly lower IoU metric compared to the training slices, which is particularly noticeable.

Additionally, it is evident that *Intuitive Fat-U-Net 1* performs better than all other Intuitive Fat-U-Nets across all scenarios. This performance is attributed to its architectural similarity to the original U-Net, which means the number of pixels in its feature maps is also most similar to the original U-Net.

Table 4.7 Intuitive Fat-U-Nets (other "Large kernel/Few Channel") performance in comparison with original Fat-U-Net. Fat-U-Net outperforms all three variations of Intuitive Fat-U-Nets in both datasets.

Model	Oxford-IIIT Pet		HeLa cells	
	Acc	IoU	Acc	IoU
Intuitive Fat-U-Net 1	92.71	83.71	99.08	93.90
Intuitive Fat-U-Net 2	89.39	77.84	97.95	87.58
Intuitive Fat-U-Net 3	89.18	76.98	98.45	89.75
Fat-U-Net	**93.40**	**91.87**	**99.43**	**96.25**

4.4.2 U-Net without skip connections

The experiments involving the training of Fat-U-Net and U-Net without skip connections have somewhat proven the hypothesis described in Section 4.2.2, that Fat-U-Net architecture without skip connections might surpass or considerably close the performance gap with the standard U-Net architecture that also lacks skip connections. Both U-Net and Fat-U-Net performed poorly and were unstable, failing to converge. As shown in Figure 4.10, Fat-U-Net was relatively more stable compared to U-Net, which aligns with the hypothesis. However, even Fat-U-Net exhibited fluctuations in performance throughout the epochs.

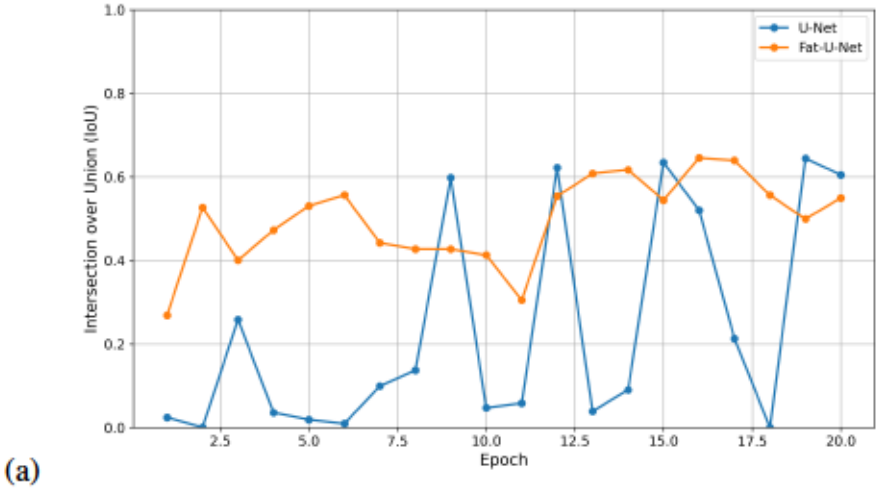
Furthermore, in Figure 4.10 (b), where the curves are smoothed, it is evident that Fat-U-Net performs more stably on average. Nevertheless, when evaluating on test data, U-Net's IoU was 86.71%, while Fat-U-Net's IoU was 86.53% on middle slices. This indicates that although the performance of both networks dropped, the gap between U-Net and Fat-U-Net without skip connections indeed shrank because Fat-U-Net manages to preserve the localisation information well.

4.5 Conclusion

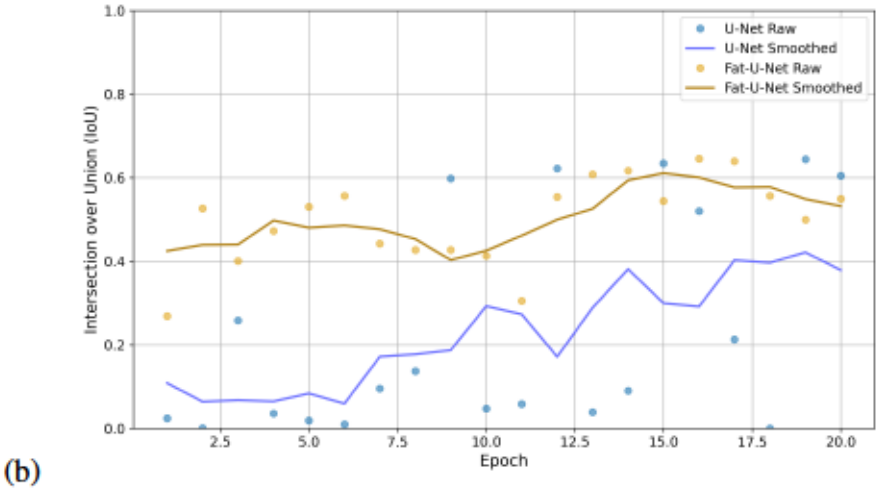
In this chapter, the expansion of the application of FatNet to segmentation tasks was successfully demonstrated by applying it to the contracting path of the U-Net model. This aligns with Objective 1, which aims to develop and adapt novel CNN architectures for acceleration in the 4f system, building on the classification tasks fulfilled in Chapter 3 and extending them to segmentation tasks here in this Chapter.

Our adapted model, Fat-U-Net, required 538 times fewer convolution operations than the traditional U-Net, leading to an inference speed 538 times faster when using optical accelerators and 37 times faster than U-Net running on a GPU, assuming a 2MHz device. Both models were tested on the Oxford-IIIT Pet dataset and the HeLa cell nucleus segmentation, achieving unparalleled results. The performance degradation was minimal, with a maximum loss of 4.24% in test IoU for the Oxford-IIIT Pet dataset and 1.76% for the HeLa cell nucleus segmentation. This makes the FatNet transformation even more advantageous than in classification tasks.

Moreover, this chapter demonstrated the efficacy of the FatNet, particularly the necessity of taking into account the number of pixels in each feature map when performing the conversion. This was achieved by training three Intuitive Fat-U-Nets. These Intuitive Fat-U-Nets only used the number of trainable parameters for the conversion, and all three underperformed compared to the original Fat-U-Net.



(a)



(b)

Fig. 4.10 **Learning curves of U-Net and Fat-U-Net without skip connections.** (a) Regular learning curve plot for both models without skip connections, and both failed to train, while Fat-U-Net was slightly more stable (b) Smoothed learning curve for both models without skip connections, showing that Fat-U-Net performed better, even though overall the final evaluation is on U-Net's favour.

This chapter also analysed the effect of preserving the localisation information in Fat-U-Net by training U-Net and Fat-U-Net without the skip connections. Experiments showed that both networks failed to converge as effectively as with skip connections, although Fat-U-Net trained more stably than U-Net.

In conclusion, this chapter successfully fulfilled Objective 1 by extending FatNet's application from classification (as demonstrated in Chapter 3) to segmentation, achieving significant efficiency gains while maintaining high performance. Additionally, this chapter validated the algorithm through experiments such as Intuitive Fat-U-Nets and U-Net without skip connections to prove the efficacy of the approach. These results further demonstrate

FatNet's adaptability across diverse tasks and its potential to optimise deep learning models for free-space optical accelerators.

Chapter 5

Shared Convolutional Vision Transformers (ConvShareViT)

Overview

In this chapter, we introduce ConvShareViT, a new architecture for Vision Transformer models that exclusively uses convolution operations. The chapter examines different methods for incorporating convolutions within the attention mechanism and the MLP blocks of the ViTs, with the main focus on enabling efficient inference in the 4f free-space accelerator. Additionally, the chapter explores various parallelisation techniques that are both feasible and effective for the 4f system.

5.1 Introduction

In the previous chapters, the conversion of convolutional networks into formats more compatible with the 4f system was discussed. It has been noted that transformers can be implemented in an optical setup, as reviewed in the literature (See Section 2.8.5). However, this necessitates a completely new setup and does not utilise the existing 4f system, which would be preferable. Since the 4f system is already employed for convolution operations, its application to transformers could be considered without any modification to the optical setup. Unlike convolutional neural networks, transformer models rely on linear layers and matrix multiplications. Matrix multiplication can be represented through convolution operations and processed using the 4f system, suggesting the feasibility of implementing transformers within this system. However, this approach is inefficient, as it results in many irrelevant pixels in the output. A question arises: Is it possible to employ only the convolution operation to learn the

attention, preserving the attention mechanism exactly as in the original Vision Transformer (ViT), but replace the matrix multiplications with convolution operations and run it efficiently on the 4f system?

5.2 Methodology

The methodology involved first training a regular vision transformer on the CIFAR-100 dataset, and, once reaching the state-of-the-art accuracy of ViT trained on CIFAR-100 without fine-tuning, the variety of methods is explored to reach the optimal solution for the convolutional neural network.

The difference between the ConvShareViT network of this work and transformers starts right at the vector projection stage. In transformers, images are typically divided into patches and embedded into a high-dimensional vector space using linear layers. In contrast, ConvShareViT networks keep the tokens in a two-dimensional format after diving into patches. If the Vision Transformers activations are in the shape of [batch, number of patches, embed dimension], in this case they are [batch, number of patches, embed dimension y, embed dimension x]. To embed the patches into the desired resolution, transpose convolution is used. Additionally, transpose convolution is employed to embed the 3 RGB channels into a single channel with higher resolution. In this case, the resolution of the patch is analogous to the embed dimension.

It is important to initially note that the linear layer is the primary component of all layers in the transformer's encoder. Linear layers are present in both the Multi-Head Self-Attention (MHSA) layers and the Multi-Layer Perceptron (MLP) layers situated between the MHSA layers. Each output node of a linear layer represents a weighted sum of the input nodes.

Since our model processes patches as matrices rather than vectors, each patch can be convolved with a weight matrix of identical resolution using valid padding to produce the output node, as shown in Figure 5.2. When using same padding, the centre pixel of the output will be within the valid region. This process must be repeated for each output node. However, in an optical context, kernel tiling can be employed to generate all output pixels simultaneously (See Figure 5.2 (c)).

5.2.1 Shared Depthwise Convolution

The regular Convolutional layer consists of the 3D feature maps and 3D kernels, where the number of kernels is equal to the number of the output channels, and depth is equal to the number of input channels as shown in Figure 3.4. This can also be viewed as a number

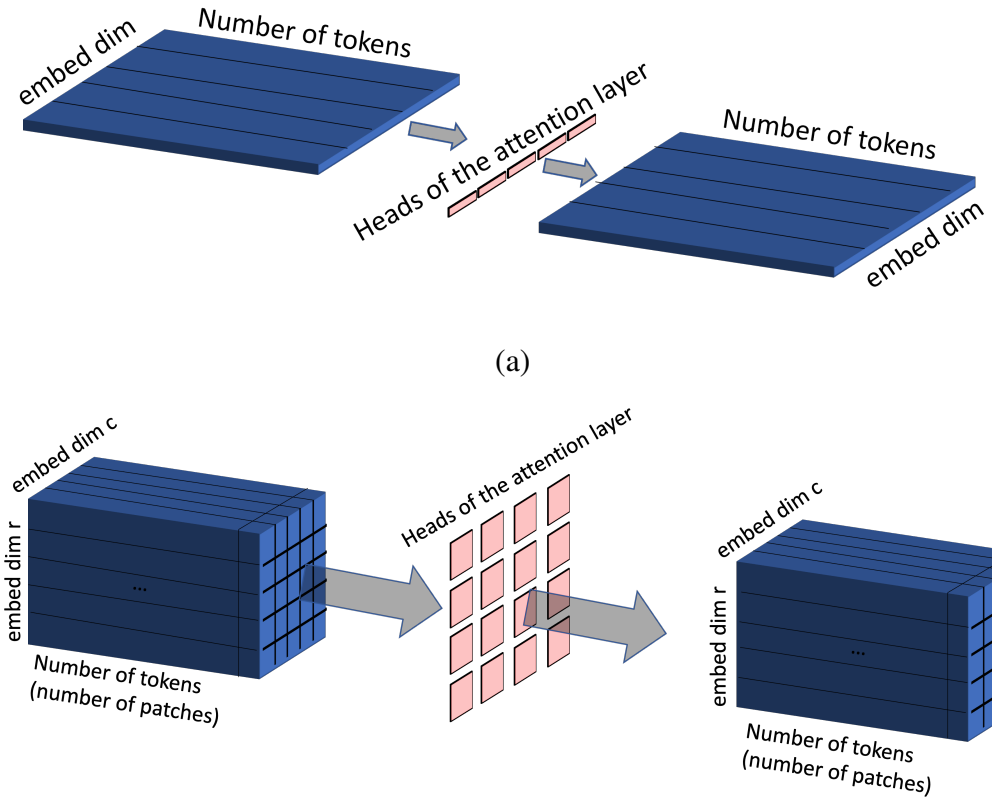


Fig. 5.1 **Visual comparison of input split in regular multi-head attention and our method when the inputs are two-dimensional.** (a) In regular multi-head attention, the input vectors are split into equal-sized vectors, each assigned to a dedicated head of attention, followed by the concatenation of the outputs. (b) In our method, the process can be viewed as patchification, where the two-dimensional input is divided into smaller patches that fit into the heads of convolutional attention layers. The outputs are then merged back into their corresponding locations.

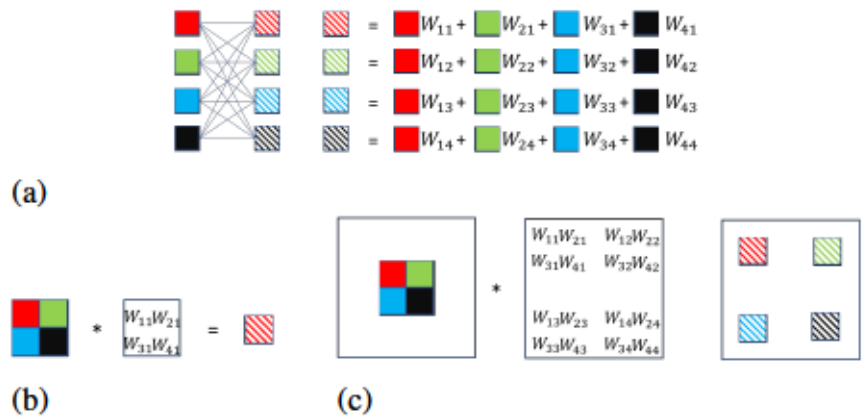


Fig. 5.2 **Implementation of the linear layer using convolution and tiled convolution for 4f system.** (a) A simple linear layer of one vector is applied to another vector of the same length. Each output pixel is the weighted sum of input pixels. (b) Input pixels are in 2D matrix format, convolved with the kernel of the same size, with the valid padding. The output is similar to one output pixel of the linear layer. (c) Kernel tiling is used to tile all weights of the linear layer in the kernel block, and input is padded to the required resolution. The output archives all output nodes of the linear layer, with the requirement of reshaping (removing zeros in invalid regions)

of input channels $\times$ number of output channels 2D kernels, where the results are summed across the input channels, as shown in Figure 3.5.

When trying to represent MLP through the convolutional layer, as shown in Figure 5.2, there is one challenge which needs to be addressed. In the linear layer, the input channel is a separate patch and should be treated independently without any interaction with other patches, and the channel summation should be avoided. Hence, depthwise convolution must be used, meaning the number of groups of convolution is equal to the number of input channels (See Figure 5.3 (b)).

However, when a tensor is passed through a linear layer, the transformation is applied to the last dimension, mapping these dimensions into new vectors. This process involves sharing the same weights across all channels of the tensor. Hence, if a linear layer is to be emulated via convolution, the kernels must be repeated across the input channels in a depthwise convolution. This method is referred to as shared depthwise convolution (See Figure 5.3 (c)).

Figure 5.4 shows a more detailed implementation of Shared Depthwise Convolutional layers. In certain scenarios, an input matrix must be mapped to two or more outputs, as shown in (See Figure 5.4 (c)). In such cases, the number of output channels is simply increased while the number of groups remains equal to the number of input channels. The weights are repeated across input channels. Conversely, when two or more channels need to be mapped

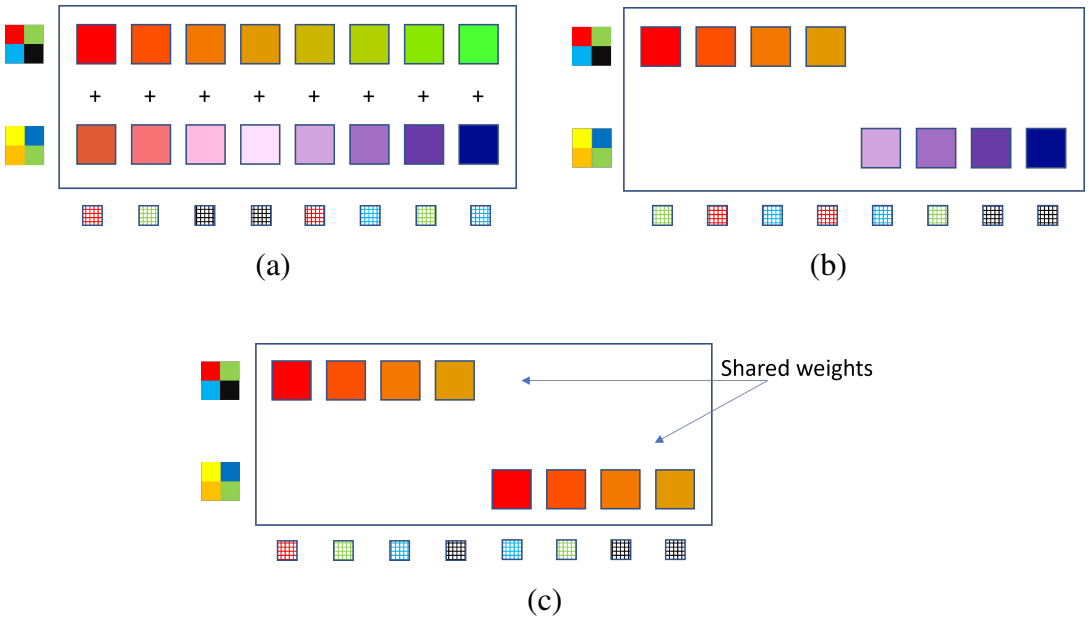


Fig. 5.3 **Comparison of regular valid convolutional layer, depthwise convolutional layer and shared depthwise convolutional layer, which copies the weights across all input channels.** (a) Regular convolutional layer, with the groups=1. The number of 2D kernels is equal to the number of input channels the number of output channels. (b) Depthwise convolution, where the number of groups is equal to the number of input channels. In this case, each output channel gets only one 2D kernel, meaning no channel summation happens. (c) In the shared depthwise convolutional layer, unlike the regular depthwise convolutional layer, the weights are shared across input channels, making it ideal for the emulation of the Linear Layer. If the kernels are the same resolution as inputs, the valid convolution yields one pixel for each output channel, which can be reshaped into the initial resolution.

back into fewer channels—such as in Figure 5.4 (b) where they are mapped into one—the channels to be merged are treated as a single input, and their weights are not shared but summed, similar to the process in the regular convolutional layer. In this scenario, the number of groups corresponds not to the total number of input channels but to the number of input channels being treated as a single input. In the case of Figure 5.4 (b), this number is two.

These methods are enough to emulate the MLP in Vision Transformers using convolution operations. For instance, by following the sequence of operations in Figure 5.4 (c) and (b), an MLP with a ratio equal to two can be emulated. This emulation can be implemented optically using the tiling method shown in Figure 5.2 (c).

5.2.2 Attention Mechanism

The attention mechanism can be divided into three main stages, these are QKV projection, attention scores calculation, and the weighted sum of the attention scores. The QKV (query, key, and value) projection, a crucial part of multi-head self-attention (MHSA), transforms input data into these three components, facilitating attention calculations as described in Section 2.6.2. In this work's experiments, several methods were used before achieving the ideal solution for the QKV projection, and the final method was to use the shared depthwise convolutional layers, which would be a full mimic of the linear layers. Apart from these, regular depthwise convolution and depthwise convolution with the reshaping of the outputs were used.

Mainly four variations of convolutional layers were used, firstly with the same padded depthwise convolution, with weights shared and not shared across the input channels. Then the valid padded depthwise convolution, with weights shared and not shared across the input channels.

The calculation of the attention scores is a simple valid convolution of all patches with each other and softmax function taken in the dimension one, or dimension two when minibatches are used.

The last step is the weighted sum of the attention scores, which we do using a point-wise convolutional layer, treating the attention scores as weights of the layer. The outputs are then located in the correct location of the main larger patches.

In the regular self-attention layer, this last step is performed by a simple matrix multiplication:

$$Y = A \times V \tag{5.1}$$

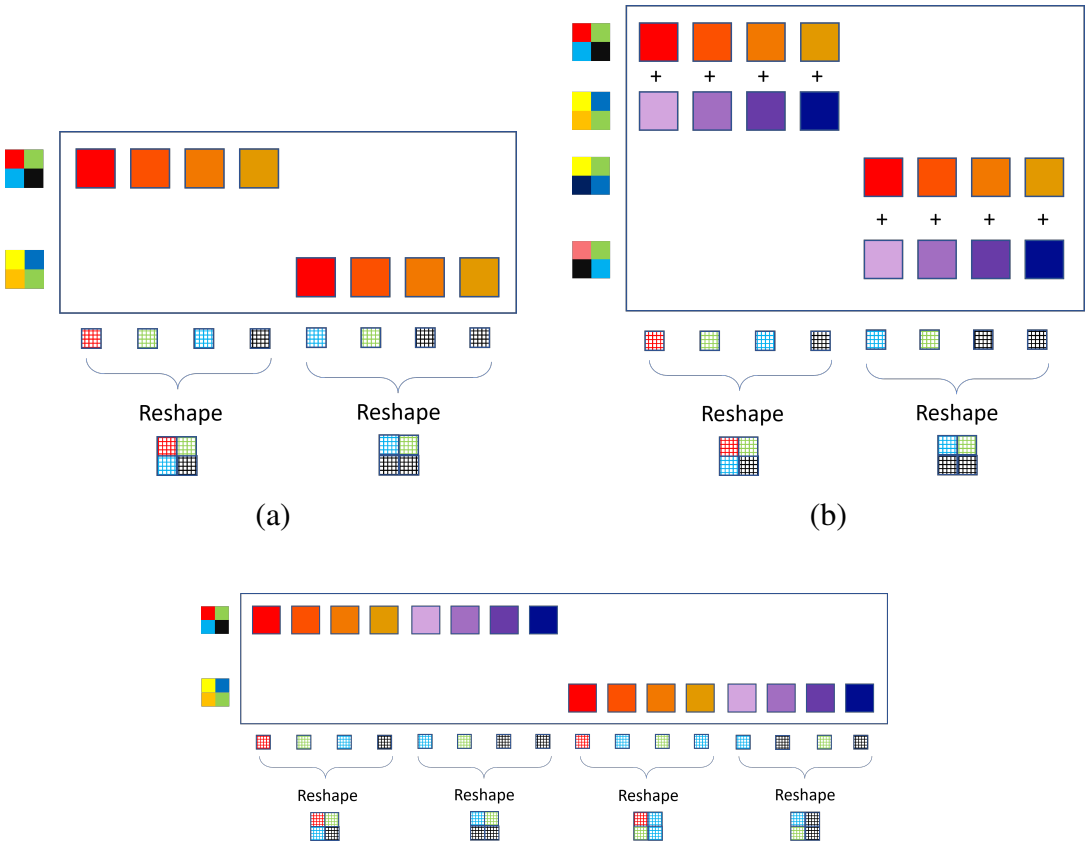


Fig. 5.4 **Shared depthwise convolutional layer with the valid convolution and reshape of the output for full emulation of the Linear layer using convolution.** (a) Shared depthwise convolutional layer from one matrix to one. In this case, two matrices have been mapped to their new corresponding matrix. (b) Shared depthwise convolutional layer from two matrices into one. In this case, four matrices have been mapped into two, each group of two into one corresponding output matrix. The technique can be used from many to fewer matrices. (c) Shared depthwise convolutional layer from one matrix into two matrices. In this case, two matrices have been mapped into 4, where each has been mapped into corresponding two outputs. The technique can be used from few to many mapping.

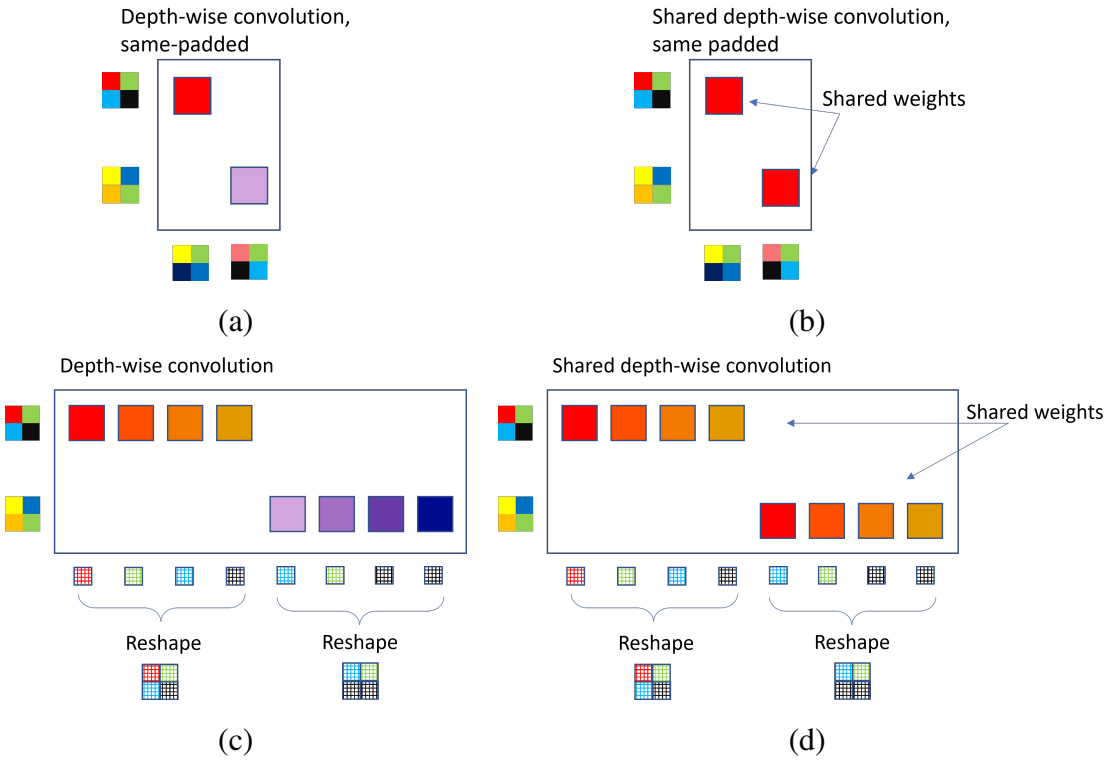


Fig. 5.5 **Types of Convolutional layers used for the QKV projection, with shared and not weights across the input channels and with same or valid padding.** (a) Simple depthwise convolutional layer with same padding. (b) Shared depthwise convolutional layer with same padding (c) Depthwise convolutional layer with valid padding, the outputs need to be reshaped into the initial resolution (d) Depthwise convolutional layer with shared weights across input channels and with valid padding, the outputs need to be reshaped into the initial resolution.

where A is the attention score tensor, and V is the tensor of values. To do this using the convolution, first need to look at the general formula for the 2d convolutional layer without the bias:

$$Y_{k,r,c} = \sum_{ch=1}^{CH_{in}} \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} X_{ch,r+i,c+j} \times W_{k,ch,i,j}, \tag{5.2}$$

where k is the output channel, r is the row index in the output, c is the column index in the output, $X_{ch,r+i,c+j}$ refers to the input feature map with the dimension $[CH_{in}, H_{in}, W_{in}]$, and the output Y with the dimensions $[CH_{out}, H_{out}, W_{out}]$. W represents the set of kernels with the dimension $[CH_{out}, CH_{in}, H, W]$.

When the convolution is 1D, the spatial dimension is reduced to one, the convolution is simplified to:

$$Y_{k,r} = \sum_{ch=1}^{CH_{in}} \sum_{i=-a}^{a} X_{ch,r+i} \times W_{k,ch,i}, \tag{5.3}$$

where a is half the kernel size(i.e., the filter has $2a+1$ elements), and r represents the spatial position in the 1D output (analogue of rows in 2D case).

When the convolutional layer uses 1×1 kernels, the convolution effectively becomes a point-wise matrix multiplication across channels, identical to dense layer operations. For 1D convolution, the formula with 1×1 kernels becomes:

$$Y_{k,r} = \sum_{ch=1}^{C_{in}} X_{ch,r} \times W_{k,ch} \tag{5.4}$$

which is equivalent to $W \times X$. This leads to the conclusion that matrix multiplication can be treated as the convolutional layer, with the left term being a weight matrix, as can be seen in Figure 5.6.

5.2.3 Multilayer perceptron

In a vanilla transformer encoder, each multi-head self-attention layer is followed by a multi-layer perceptron (MLP) layer. The MLP usually consists of two linear layers: the first maps the embedding vectors into a higher-dimensional space, and the second maps them back to the original dimension. One of the hyperparameters of the MLP is the MLP ratio, which indicates the scaling factor of the dimension, representing the ratio of the hidden layer to the input or output layer.

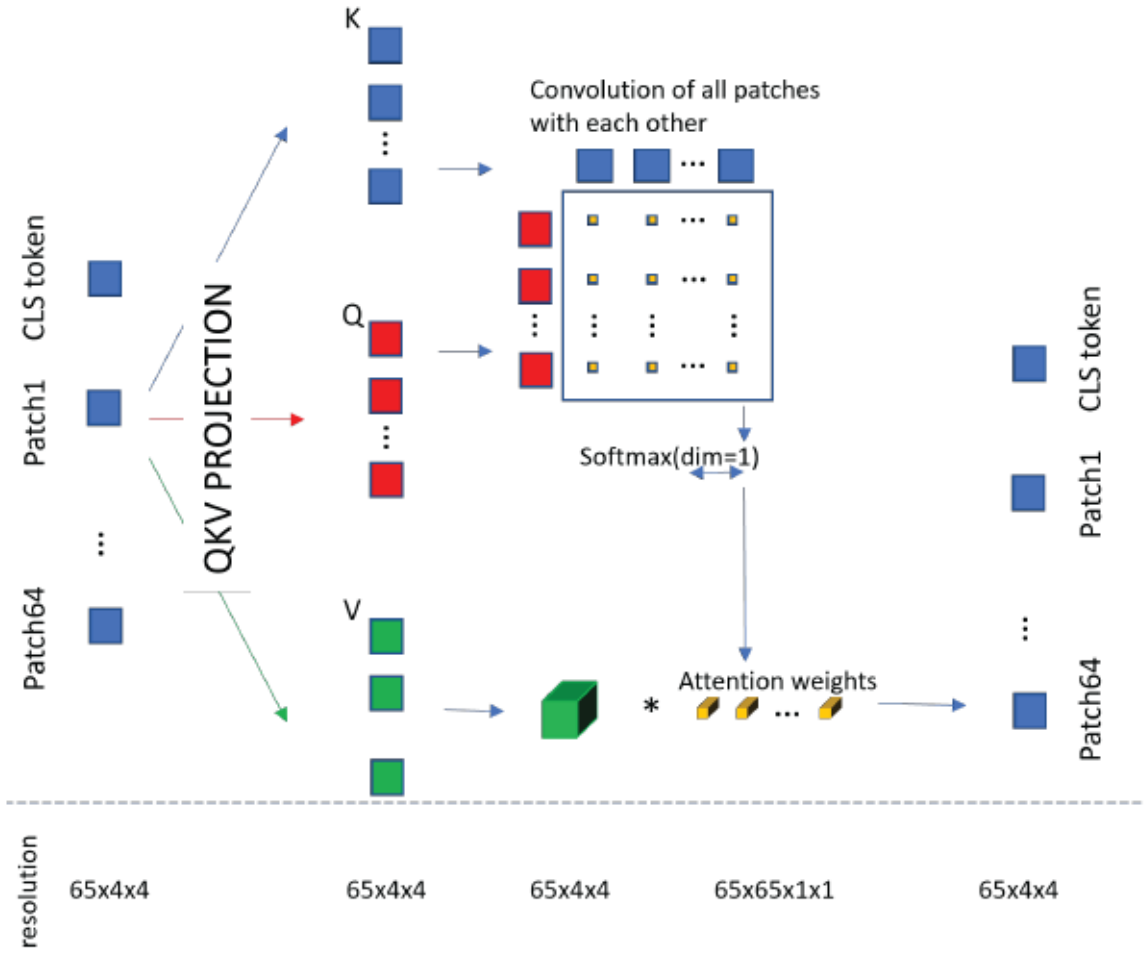


Fig. 5.6 **General flow of the attention mechanism using only convolution operations and CNN layers.** The input to the attention layer is a 3D tensor, where each token is represented as a 2D matrix. This tensor undergoes QKV projection using one of the methods described in Figure 5.5. The attention scores are computed by applying a valid convolution operation between all Q and K matrices. Finally, the weighted sum of the attention scores is obtained through a regular convolutional layer on the V tensor, with the attention scores acting as the weights of the layer.

Since the original transformer uses a linear layer, our method uses the same concept used for the QKV projection in the multi-head self-attention (MHSA) layer. First, a shared depthwise convolutional layer with a kernel size equal to the input size is used to map the input into the higher dimension. The output of this layer is $1 \times 1 \times$ (number of tokens $\times$ MLP ratio $\times x \times y$). This output is then reshaped into $x \times y \times$ (number of tokens $\times$ MLP ratio), increasing the number of tokens by the MLP ratio.

Similarly, the output of the second shared depthwise convolutional layer is $1 \times 1 \times$ (number of tokens $\times x \times y$), which can be reshaped to the input's original shape $x \times y \times$ number of tokens

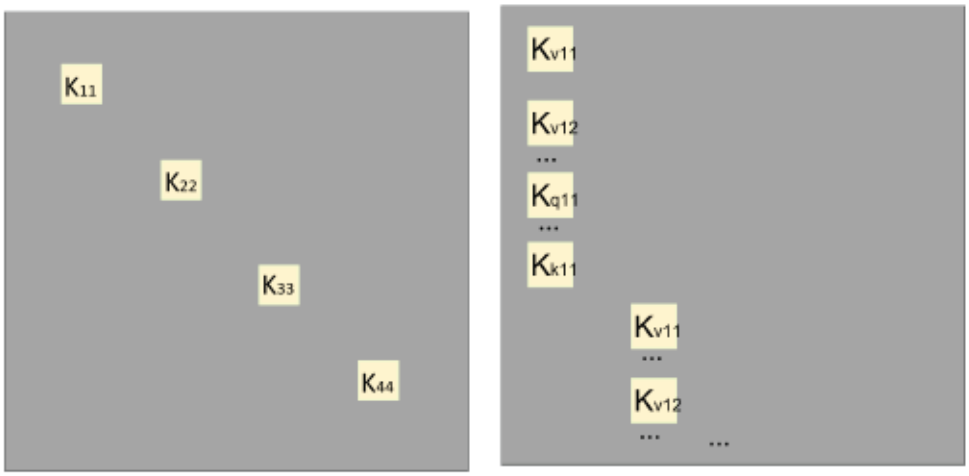
5.2.4 Potential Parallelisation of ConvShareViT in 4f system

The primary purpose of using the convolution operation in attention layers is to use the 4f system's ability to perform these operations faster and more efficiently than standard electronic components. The key advantage of free-space optics lies in its capability to execute high-resolution operations without inference delays. To take advantage of the parallelism and high-resolution capacities of 4f free-space accelerators, earlier chapters on FatNet emphasised increasing the resolution of neural networks by employing input tiling to fully utilise the system's resolution.

Unlike CNNs, which rely on convolutional layers, Transformers are based on linear layers and typically perform more efficiently on GPUs, where entire tensors are loaded, allowing for optimised and rapid computation. However, in the optical setup, regular input tiling to execute the described methodology is insufficient. Given that the FatNet conversion approach was not originally developed for Transformers, the methods discussed can be parallelised using mixed tiling, as described in Section 2.8.4, introduced by Li *et al.* [7].

While mixed tiling is traditionally applied to standard convolution operations, in our case, it needs to be adapted for depthwise convolutional layers. This can be achieved by setting all kernels, except the one corresponding to the output channel, to zero, as shown in Figure 5.7 (a).

In QKV projection, the kernels for the output pixels q, k, and v can be tiled within a single mixed tiling block and then split after the output is generated, as shown in Figure 5.7 (b). In most cases, the number of kernels will likely exceed the resolution of the 4f system, potentially requiring multiple inferences.



(a) (b)

Fig. 5.7 **Mix tiling with depthwise convolutional layers and its use in QKV projection layers for convolutional attention layers.** (a) Simple mix tiling of kernels, but kernels other than those corresponding to the output channel are set to zero to avoid summation (b) Demonstration of how Shared depthwise convolutions can be used in the qkv projection.

5.3 Experiments

In this chapter, as in Chapter 3, the CIFAR-100 dataset was used for classification. Data augmentation was applied with all models in this study, including PyTorch's built-in "CIFAR10" auto-augmentation, random cropping with a padding of 3, random horizontal flipping, and standardisation using the mean and standard deviation of the dataset.

Initially, four regular Vision Transformers (ViTs) were trained on CIFAR-100 to serve as baseline models. These included combinations of ViTs with and without trainable positional encoders, as well as models using multi-head (12 heads) and single-head attention mechanisms. The image patch size was set to 4×4 from the original image size of 32×32, resulting in 64 patches overall (65 when including the classification token). The tokens are embedded into the 192 dimensions. The number of transformer blocks was set to 9, with an MLP ratio of 2. The models were trained for 310 epochs, with 10 epochs reserved for warmup. The Adam optimiser was used, with a starting learning rate of 5×10^{-4} with the Cosine Annealing Scheduler.

When it comes to ShareConvVit models, the main twelve experiments out of multiple experiments are reported in Table 5.1. Several methodologies had to be tested to determine the optimal configurations for our models. The table specifies which techniques are applied to each model, including Trainable Positional Encoding, Multi-Head Attention, usage of Multi-Layer Perceptron (MLP), Shared Depthwise Convolution, and Valid Convolution. Each model

represents a unique combination of these methods, allowing us to systematically evaluate their impact on performance. Additional details, such as the embedding dimensions and the absence of bias in certain models, are also noted to provide a comprehensive understanding of the configurations tested. All models were trained using the Adam optimiser [73], with a learning rate of 0.0005 for Models 1–5 and 0.0008 for Models 6–12.

It can be seen that Models 2, 3, and 4 seem identical. The difference lies in the QKV projection, which is listed below:

- Model 2 uses a simple depthwise convolutional layer with the same padding and a kernel size equal to the resolution of the head patch, which is 4 in this case. The number of input channels and output channels is the same, equal to the number of patches.
- Model 3 uses four consecutive depthwise convolutional layers to increase the number of channels by 7 times, then reduces it back to the original number and repeats it again.
- Model 4 uses two consecutive depthwise convolutional layers to increase the number of channels by 14 times, then reduce it back to the original number.

A similar pattern can be observed with Models 5 and 6; both models use valid depthwise convolution, meaning they reshape the 1×1 outputs to match the original resolution. However, Model 5 contains one more depthwise convolutional layer, similar to the one in Model 2, before the valid convolution.

The primary aim of the project is to analyse not only the performance of the models but also to determine whether these models can learn attention in a manner similar to Vision Transformers. To achieve this, the average attention scores were visualised.

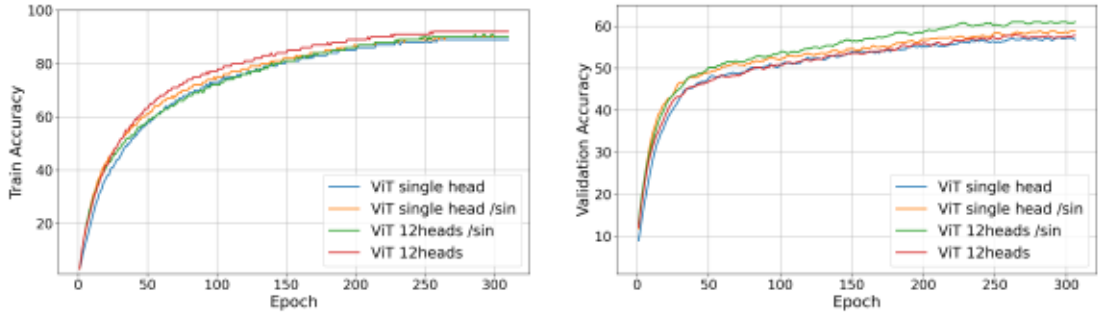
With the ShareConvViTs, the aim was to maintain as much similarity as possible to the original ViTs that were trained in this work, preserving the same MLP ratio, the same number of layers, and the same patch size of 4×4. The difference lay in the embedding dimension, as the ShareConvViTs needed to retain a square shape, as shown in Figure 5.1. All models, except for Models 10 and 12, had embedding dimensions of 16×16, resulting in a total of 256, which is higher than the original ViTs' embedding dimension. This increase was necessary to extract 16 heads from the tokens, each being 4×4. However, Models 10 and 12 had an embedding dimension of 13×13, making 169 in total, which is lower than the original ViTs' embedding dimension. Since these models were single-head, the tokens did not need to be further patchified when fit into the convolutional attention layers.

Model	Trainable Pos Encoding	Multi head Attention	MLP used	Shared dw Convolution	Valid Convolution	notes
Model 1						
Model 2						Bias in qkv
Model 3						Bias in qkv
Model 4						Bias in qkv
Model 5						Bias in qkv
Model 6						Bias in qkv
Model 7						
Model 8						
Model 9						
Model 10						Embed dim=13
Model 11						
Model 12						Embed dim=13

Table 5.1 Summary of Methods Applied to Different Models during ConvShareViT development. This table outlines the primary experiments conducted and the methods applied to each model. Each row represents a distinct model and indicates the presence of specific methods with a checkmark.

Model	ViT 12 heads	ViT 1 head	ViT 12 heads_sin	ViT 1 head_sin
Acc (%)	61	63	64	64

Table 5.2 Test accuracy on CIFAR-100 of a regular vision transformer with different configurations. Models with "_sin" indicate the use of sinusoidal position encoding instead of trainable encoding.



(a) (b)

Fig. 5.8 **Comparison of training and validation curves for four Vision Transformer (ViT) models, featuring combinations of trainable versus sinusoidal positional encoders and single head versus twelve heads.** (a) Train accuracy per epoch (b) Validation accuracy per epoch

5.4 Results and Discussion

The test accuracy of four variations of the regular Vision Transformer (ViT) is shown in Table 5.2. It is evident that the fixed sinusoidal positional encoder outperforms the models with trainable position encodings. Additionally, the single-head model performs slightly better than the twelve-headed model when the positional encoder is trainable, although the difference is not substantial. With sinusoidal positional encoding, the performance is nearly identical between the two models. Although both test accuracies are equal to 64%, the training curves in Figure 5.8 demonstrate that the ViT with 12 heads and sinusoidal positional encoding achieved the best performance in terms of validation accuracy. In contrast, the ViT with 12 heads and a trainable positional encoder, despite showing better performance on the training set, shows poor validation performance, as reflected in the test accuracy of 61%. This suggests that using 12 heads may have been excessive for a simple dataset like CIFAR-100, especially when combined with a trainable positional encoder, as the additional parameters likely led to overfitting.

These results can be directly compared to those reported in previous works. Specifically, the choice of the 12-headed, 9-layer Vision Transformer (ViT) with 4×4 patches was

Table 5.3 Test Accuracy of models described previously in Table 5.1.

Model	1	2	3	4	5	6	7	8	9	10	11	12
Acc (%)	49	42	52	52	48	53	54	25	58	62	**63**	59

motivated by the work of Lee *et al.* [69]. In their research, they also trained ViTs on the CIFAR-100 dataset from scratch, which is not a usual case as ViTs are typically pre-trained on larger datasets before being fine-tuned on smaller ones. Lee *et al.*achieved a test accuracy of 60.01% without augmentation and 73.81% with a combination of augmentations, including CutMix, Mixup, and AutoAugment. Additionally, they used techniques such as label smoothing, stochastic depth, and random erasing.

In contrast, the approach in this work used only AutoAugment, yet it achieved higher performance than Lee's model, which did not incorporate all the augmentation methods. Although our results were lower than those of Lee's model, which employed the full suite of augmentation techniques, our method still demonstrates significant effectiveness. The exclusion of CutMix and Mixup in this work was due to their potential to slow down the training process, which needed to stay efficient for running multiple experiments to validate the concept. Another study by Zhu *et al.* [74] trained a smaller ViT on CIFAR-100 with a depth of 6 layers and 8 heads, achieving a test accuracy of 54.31%.

Turning to our novel method, ShareConvViT, specific performance characteristics were observed. Interestingly, models 10, 11, and 12 were single-headed models and performed better than the multi-headed models, as shown in Table 5.1. Model 8 was the only model among the final ones to use the same padding in the qkv projection, resulting in a poor test accuracy of 25%. This suggests that using valid convolution and reshaping the outputs (essentially replicating MLP) is essential. However, it is notable that models up to and including model 4 also used the same padding and retained the original output shapes. These models did not share weights across input channels, resulting in a higher number of trainable parameters, which contributed to their relatively high performance. Unfortunately, despite their strong performance, these models did not learn classification in the traditional attention mechanism way, as shown later in the visualisation analyses.

Figure 5.9 shows the training process of all models that led to the ConvShareViT. All models ran until 310 epochs, 10 of which were reserved for warmup, except model 2, which was stopped due to poor performance and no updates in the loss. Model 11 is leading in both training and validation train curves, just like in the test accuracy Table 5.1. Although model 9 seems to be second in the train accuracy plot, it is behind model 10 in the validation accuracy plot. This is also obvious from the test accuracy Table 5.1, where models 9 and

10 achieve 58 and 62, respectively. This means that Model 9 could have minor overfitting issues. It is worth pointing out that model 10 is also a single-head model, while model 9 is a multi-head model. Moreover, model 10 uses a smaller embed dimension of 13 instead of 16. This again leads to the suggestion that single-head attention is enough to perform the classification of CIFAR-100.

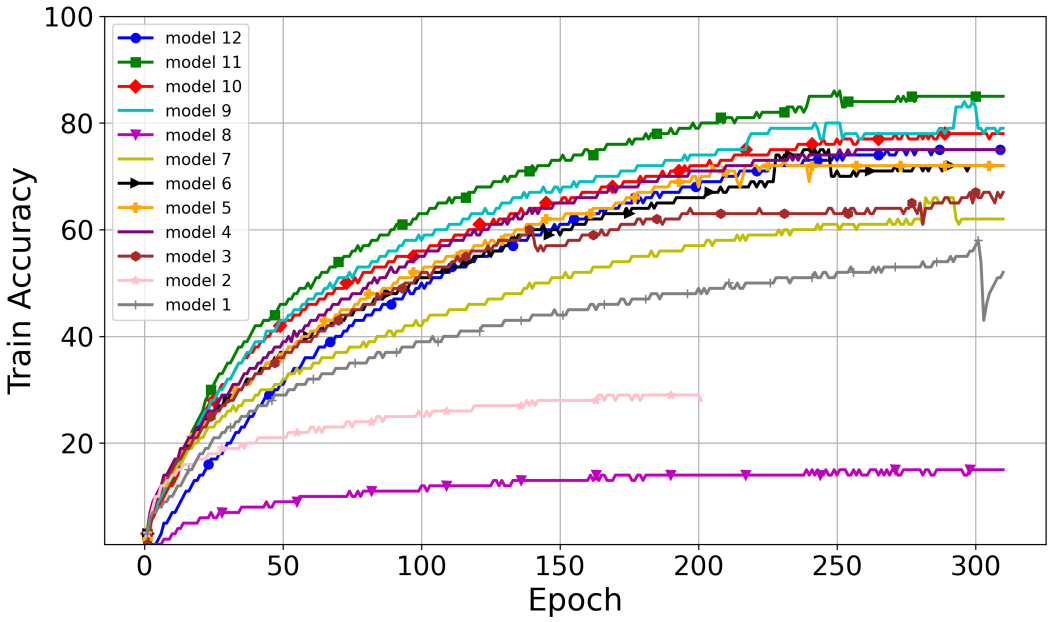
Figures 5.10 and 5.11 show the comparison of the average attention scores per layer of ViT and the ConvShareViTs. Notably, only models starting from Model 8 onward use shared depthwise convolution. However, Model 8 itself does not apply valid convolution with output reshaping, making the attention score visualisations clear only from Model 9 onwards.

In Model 9, the attention scores begin to focus primarily on the objects in the image from the 5th layer onward. Interestingly, Model 10 exhibits attention scores concentrated only on the background, yet it still achieves good performance. Model 11, which delivers the best performance, demonstrates strong attention scores across both figures, indicating effective learning of attention mechanisms. However, Model 11 also shows a bias in the learnable positional encoding, leading to high attention scores in the top corners across all layers.

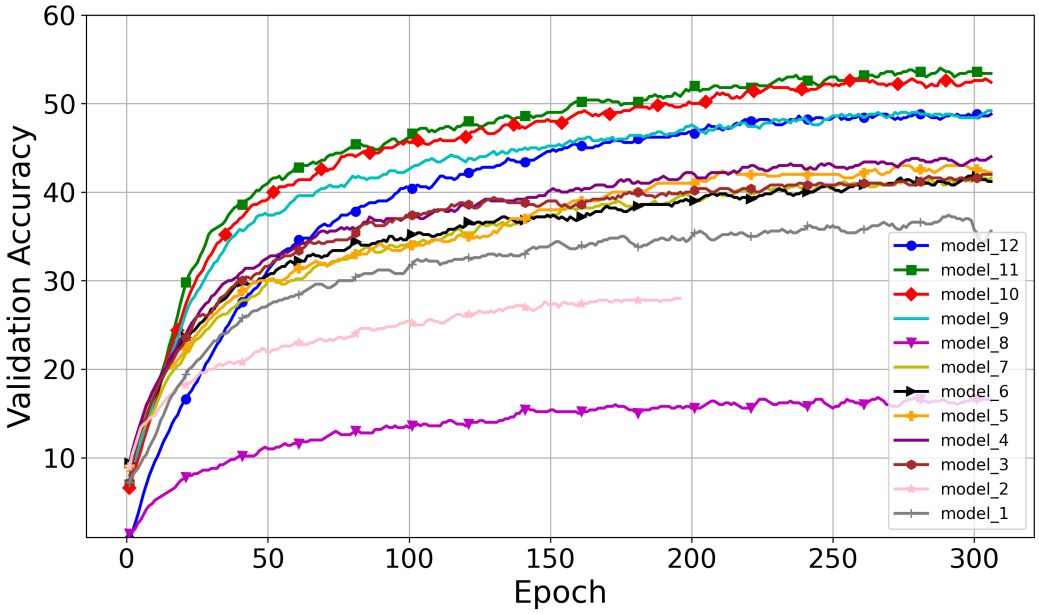
To address this issue, Model 12, which incorporates fixed sinusoidal positional encoding, was developed. This adjustment resulted in more balanced attention scores, although the performance slightly decreased, with test accuracy dropping to 59%. This performance decline is likely due to Model 12's smaller size, with an embedding dimension of only 13×13.

The results indicate that while all models can achieve high performance, the model using same padding did not perform well when applied in the shared depthwise convolution. However, models with same padding achieved relatively good performance when the weights were not shared across the input channels in the depthwise convolution. The visualisations reveal that the models only learn the attention mechanism when they use shared depthwise convolution with valid padding and reshape the outputs— essentially, when the convolution function emulates the regular linear layers.

Unfortunately, this observation suggests that the effectiveness of using convolutions in self-attention layers depends on their ability to replicate linear layers. In other cases, the convolutions may behave more like traditional convolutional neural networks, adding complexity without necessarily enhancing the model's performance. Although the performance of models 7 and below is not poor, and they do converge, it is difficult to categorise these models as transformers. Nevertheless, the results demonstrate that convolutions can be used to emulate linear layers by employing shared depthwise convolutional layers and can be effectively integrated into the 4f system.



(a)



(b)

Fig. 5.9 **Training curves comparison for the Training set and Validation set of CIFAR-100 with different ConvShareVit models** (a) Training curve for train set of CIFAR-100, with the best model being model 11. (b) Training curve for the validation set of CIFAR-100, with the best model being model 11

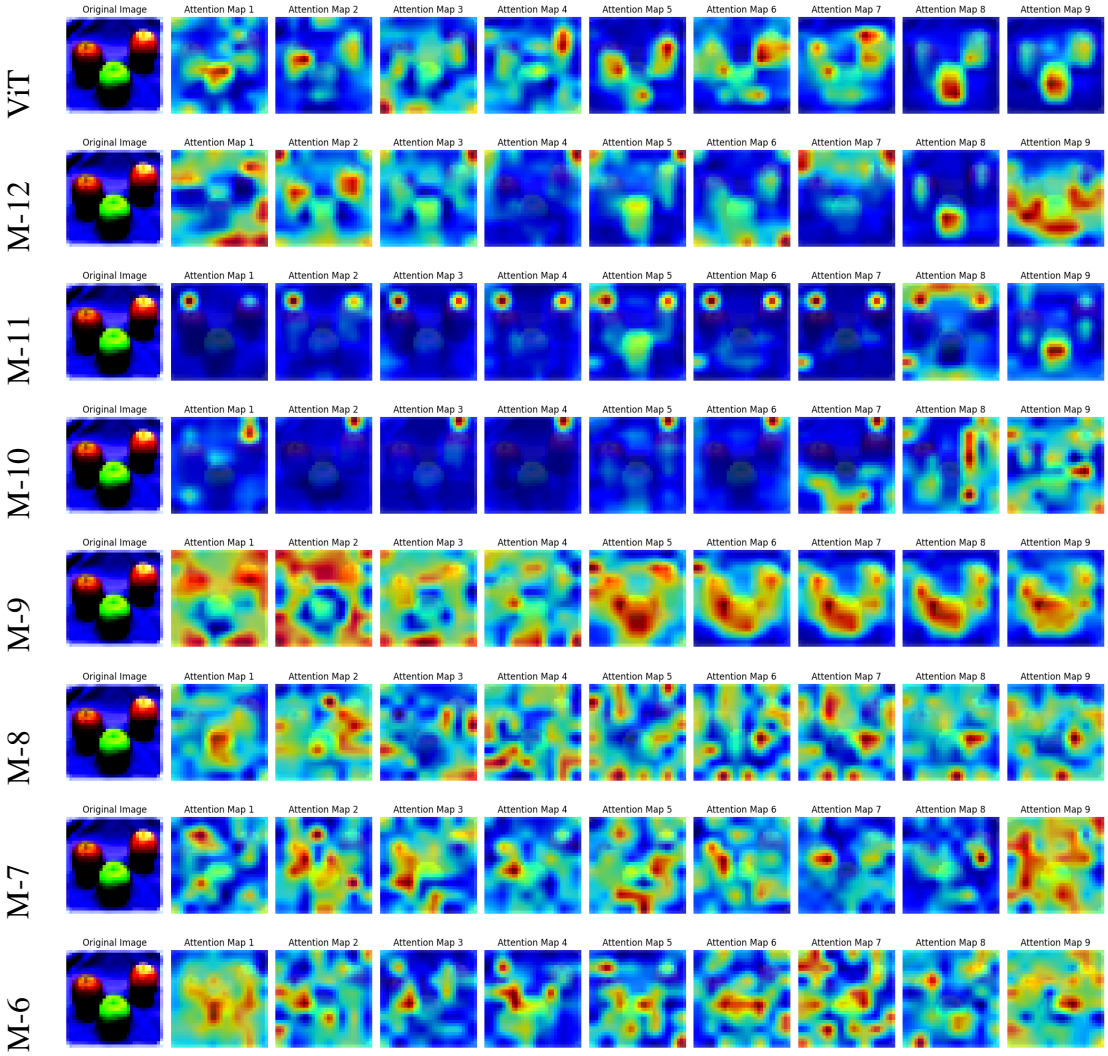


Fig. 5.10 **Visualisation of the average attention scores projected onto the original input image of Apples from the test set of CIFAR-100.** This figure compares the performance of the last seven models with the regular ViT (Vision Transformer with 12 heads). The vertical axis corresponds to the models and the horizontal to the attention layers. The ViT model achieved good attention scores in the final layers using a standard attention mechanism. Models 9, 11, and 12 also achieved attention scores similar to the original ViT. In contrast, Model 10's attention scores look incorrect as it is focusing on the background instead, as evidenced by other visualisations. Model 8 did not converge, while Models 7 and 6 did not employ the Shared DW convolutional methods without emulating the linear layer, causing the models to not learn the attention scores in the same manner as the ViT.

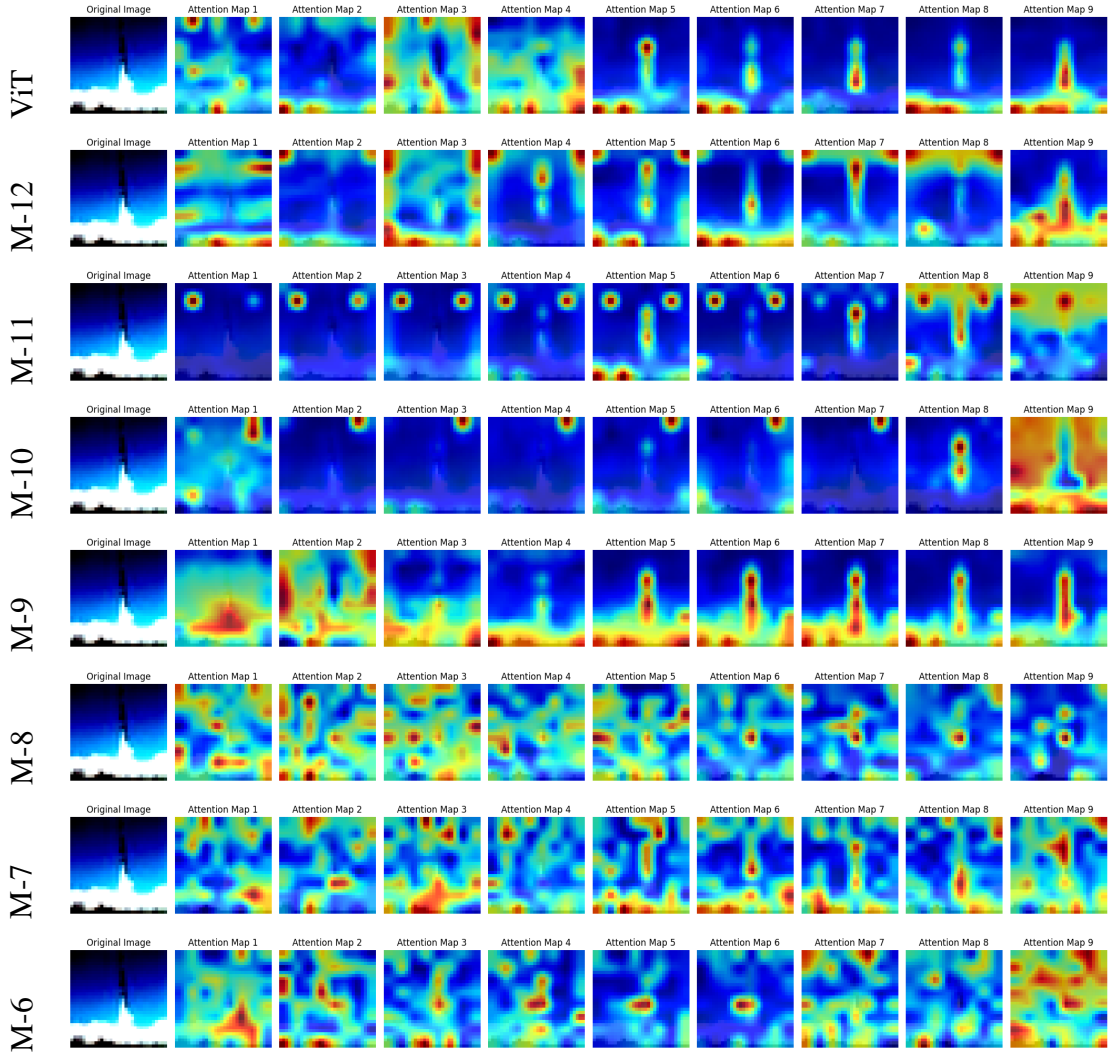


Fig. 5.11 **Visualisation of the average attention scores projected onto the original input image of a rocket from the test set of CIFAR-100.** This figure compares the performance of the last seven models with the regular ViT (Vision Transformer with 12 heads). The vertical axis corresponds to the models and the horizontal to the attention layers. The ViT model achieved good attention scores in the final layers using a standard attention mechanism. Models 9, 11, and 12 also achieved attention scores similar to the original ViT. In contrast, Model 10's attention scores focused on the background instead, which still managed to achieve a good performance. Model 8 did not converge, while Models 7 and 6 did not employ the Shared DW convolutional methods without emulating the linear layer, causing the models to not learn the attention scores in the same manner as the ViT

When it comes to paralleling inference as described in the preceding section, the most efficient approach for a 4f system network is a single head with a 13 13 embedding dimension, similar to models 10 and 12. This configuration can be parallelised using mixed tiling. Since the number of possible input or output channels in mixed tiling depends on both the kernel size and the input size, the number of required convolutions can be observed as follows:

$$n \quad \frac{R}{M \quad N \quad 1} \tag{5.5}$$

In the shared depthwise convolution with valid padding, the input sizes are equal to the kernel size, which is 13 in this case. With a resolution estimated at 4K, i.e., 2160 pixels, the number of possible input and output channels in the convolution is 86. Thus, we have 65 input channels and 65 13^2 output channels. The number of inferences required to perform all three QKV (Query, Key, Value) projections will be 384. For attention score calculation, the required output channels equal to 65 65 necessitating 50 inferences on a 4K resolution 4f system using mixed tiling. Finally, the weighted sum attention score can be computed with one inference using mixed tiled convolution.

For the convolutional MLP blocks, however, kernel tiling is more efficient. In this scenario, the number of required input channel inferences is 65 and 65 2, resulting in 195 inferences per block. This totals 5,670 inferences for all nine layers with the 4f system. With a 2 MHz device, this is equivalent to 2.8 ms, whereas the T4 GPU performs the same task in 8.5 ms. The GPU measurements are the average of 700 inferences using cuda event timing, with the GPU being warmed up by 10 iterations.

5.5 Conclusion

In this chapter, the training of ViTs on the CIFAR-100 dataset was explored, with a focus on replacing traditional attention mechanisms with convolution-based methods. Twelve alternative models were developed and evaluated, incorporating various convolutional operations within the attention mechanism. Among these, the models employing shared depthwise convolutional layers with valid padding and reshaping to emulate linear layers demonstrated the most success. These models effectively learned the attention mechanism, as evidenced by the average attention visualisation. This supports Objective 3, particularly its first sub-objective of redesigning multi-head self-attention mechanisms to use convolutional layers while preserving their ability to learn attention scores.

The development of ConvShareViT, a ViT architecture built using shared depthwise convolutions, demonstrated that linear layers, can now be efficiently implemented using shared depthwise convolutional layers within 4f system. This innovation addresses the limitations in the versatility of optical accelerators, as described in Section 1.1.

Additionally, this chapter explored potential mixed tiling methods within the convolutional attention mechanism. These methods demonstrated that ConvShareViT can theoretically achieve up to 3.04 times faster inference than GPU-based ViTs when deployed on the 4f system. This aligns with second sub-objective of Objective 3, further validating the computational efficiency and parallelisation potential of ConvShareViT in optical systems.

The visualisations of average attention scores demonstrated the learning behaviour of the models. Unlike other convolution-based approaches, the shared depthwise convolutional models showed attention scores comparable to regular ViTs. This confirms that these models were learning attention mechanisms rather than simply performing feature extraction, as is often the case with CNNs.

In conclusion, this chapter successfully addressed Objective 3 by showing that Vision Transformers can be trained using convolution-based attention mechanisms and adapted for acceleration in the 4f system. The shared depthwise convolutional layers developed in this work allow linear layers to be emulated in the optical system, making the implementation of ViTs more practical. The exploration of mixed tiling methods demonstrated potential inference speed improvements, further showcasing the advantages of using the 4f system. These contributions demonstrate the flexibility of optical systems for different types of neural networks and provide a foundation for future work on convolution-based Vision Transformers.

Chapter 6

Conclusions and Future work

6.1 Conclusion

This thesis focused on the adaptation of deep learning models for 4f free-space optical accelerators. The primary focus of this research was to enhance the efficiency and speed of neural networks by taking advantage of the capabilities of free-space optical systems. A key achievement was the development of the FatNet conversion algorithm (Contribution 1), which optimises neural networks by increasing the resolution of feature maps and kernels while reducing the number of channels, thus aligning computational demands with the capabilities of the 4f system. In this conversion, the number of trainable parameters and the number of pixels are equal or as close as possible to the one in the original network.

This research includes the application of the FatNet conversion to several well-known neural network architectures, including ResNet-18, AlexNet, and VGG-19 (Contribution 1 (b)). The conversion was executed using the FatSpitter algorithm, an automated process we developed to convert PyTorch model objects into "Fat" versions of the models (Contribution 1 (a)). The FatNet conversions demonstrated substantial reductions in the number of convolution operations, leading to significantly faster inference times when deployed on optical systems. For instance, the FatNet adaptation of ResNet-18 achieved a test accuracy of approximately 60%, slightly below the original model's 66%. However, this modest 6% reduction in accuracy was offset by a substantial speedup, with inference times up to 2.46 times faster than those achieved using traditional GPU-based processing (Contribution 1 (c,d)). This was possible because of the high parallelism and resolution capabilities of the 4f system of 2MHz [39]. Moreover, if both ResNet and Res-FatNet were implemented on the 4f system, Res-FatNet would perform inference 8.3 times faster.

Similarly, Alex-FatNet and VGG-FatNet-19 experienced decreases in test accuracies by 7.13% and 5.96%, respectively. However, the inference of Alex-FatNet in the 4f system

achieved a 3 231 acceleration compared to AlexNet on a GPU, while VGG-FatNet in optics demonstrated a 2 23 acceleration over VGG on a GPU (Contribution 1 (c,d)).

The research also extended the application of the FatNet concept to image segmentation, leading to the development of Fat-U-Net, a FatNet equivalent of the U-Net architecture (Contribution 1 (b)). This model was thoroughly tested on the Oxford-IIIT Pet and HeLa cell nucleus segmentation datasets. On the Oxford-IIIT Pet dataset, Fat-U-Net achieved a mean Intersection over Union (mIoU) of approximately 85%, which is comparable to the traditional U-Net's 89%. The computational efficiency was significantly enhanced, with inference times reduced by a factor of 37 compared to GPU processing (Contribution 1 (c)). On the HeLa dataset, Fat-U-Net maintained strong performance with a mIoU of around 96.25% on test data, while U-Net achieved 97.15% (Contribution 1 (c)), demonstrating the capability of FatNet models to handle complex segmentation tasks with high efficiency.

Additionally, the performance of both Fat-U-Net and U-Net was explored with and without skip connections to assess the preservation of localisation accuracy in Fat-U-Net, attributable to its high receptive field (Contribution 4).

To demonstrate that the performance of FatNet equivalents is not merely due to chance or favourable architectural structure, we tested the Intuitive-Fat-U-Nets (Contribution 4). These models were transformed into their FatNet equivalents without using the FatSpitter. Specifically, the models' kernels and feature maps were expanded while the number of channels was reduced. Although the constraint of preserving the number of trainable parameters was maintained, no restrictions were imposed on the number of elements in the feature maps. This led to unsatisfactory performance, with the model closest to the original U-Net performing slightly worse than the U-Net itself, and the more "Fat" models exhibiting substantially worse performance. This outcome proved the efficacy of the original FatNet conversion.

In addition to CNNs, this thesis explored the adaptation of ViTs for optical systems by developing convolutional methods that enable these models to operate within the 4f framework, referred to as ShareConvViT (Contribution 3). This innovation was particularly important as it broadens the scope of optical acceleration beyond traditional CNNs. Various methods were analysed to implement the convolutional attention mechanism. The most successful approach involved the use of Shared Depthwise Convolutional layers, developed in this work, in combination with reshape functions, which effectively emulated the linear layers using convolutional layers. The same methods were also applied in the MLP blocks of the ViTs. To demonstrate that the models learned the attention mechanism, the average attention scores of the models were visualised (Contribution 4). This analysis confirmed that models using Shared Depthwise Convolutional layers in QKV projection indeed learned the attention mechanism, whereas models using Depthwise Convolutional layers with "same" padding

did not. Additionally, the study analysed the potential parallelism and implementation of the models in the 4f system using mixed tiling and kernel tiling techniques.

The research also validated models by introducing a custom PyTorch layer with a 4f system simulator (Contribution 2). Additionally, the pseudo-negativity mechanism tackled the challenge of negative weights in optical computing, ensuring accurate simulations with efficient memory use.

All contributions in this thesis were aimed at achieving the objectives outlined in Section 1.2, focusing on adapting neural networks for acceleration using the 4f optical system. The FatNet conversion algorithm and FatSpitter tool addressed Objective 1 by creating CNNs optimised for classification and segmentation tasks, reducing convolution operations, and taking advantage of the high-resolution capabilities of the 4f system.

The custom PyTorch layer with a built-in 4f simulator supported Objective 2, while the novel pseudo-negativity mechanism addressed memory and weight-handling challenges, satisfying its sub-objective.

Finally, convolution-based Vision Transformer architectures such as ConvShareViT fulfilled Objective 3 by enabling compatibility with the 4f system. Shared depthwise convolutional layers demonstrated attention learning (sub-objective 3.1), and theoretical speed-up of up to 3.04× (sub-objective 3.2).

The innovations presented make notable contributions to the field and open opportunities for future developments in deploying deep learning models on optical systems.

6.2 Future work

Future work should build upon this foundation and explore additional applications of these models.

One potential area of exploration is the extension of this work to object detection tasks. While object detection shares similarities with classification, as it involves identifying the class of objects, it also requires localisation information. As discussed in Chapter 4, segmentation involves predicting pixel-level labels and also preserving the localisation information. Since the object detection also involves that classification and identifying bounding boxes around objects, it may also benefit from high effective receptive field of the high resolution kernels discussed in this thesis. Investigating how the techniques used in this thesis can be applied to object detection would be valuable, particularly in integrating localisation with classification tasks.

Additionally, while this research focused on encoder-decoder style segmentation models, particularly U-Net, future work could explore scaling these techniques to other segmentation

models such as DeepLab [145] or Mask R-CNN [146]. Future studies could also examine using Fat-U-Net's contracting path alone to further explore the benefits of high-resolution kernels. Generally, since the model does not necessarily contract and expand, one might suggest that half of the Fat-U-Net model is sufficient for segmentation. However, our early experiments showed that this was not the case, as half U-Net contains fewer non-linearities. Nonetheless, research in this direction can be expanded, and it may be worth exploring new methods of using half of the Fat-U-Net model for segmentation. The segmentations uses cases in this thesis proved the princinple of work and analysed the Fat-U-Net architecture, but the future work can also include task-specific models and exploration of Fat-U-Net in other domains, such as self-driving.

The FatNet conversion developed in this thesis was primarily designed for sequential cone-shaped networks. As such, it cannot be directly applied to networks using depthwise separable convolutions like EfficientNet [129] or Xception [147]. While it may be possible to adapt the algorithm to incorporate branching, the main challenge lies in implementing 1×1 convolutions. These convolutions, which reduce dimensionality, could be treated as standard convolutions in a 4f system, but it is unclear whether they should be converted into FatNet layers. Although converting 1×1 convolutions could further reduce the number of channels, this may not always be necessary.

Further future work could explore additional optimisations of FatNet architectures, such as fine-tuning the trade-offs between kernel size and the number of channels to minimise performance loss.

As Transformer models were originally developed for language tasks before being adapted for vision, researching models like ConvShareViT and others from Chapter 5 could expand their use in NLP. This would require an analysis of new methods for the decoder path of transformers, as the current work focuses primarily on the encoder. Future research could investigate FatNet-style conversion methods to optimise the use of ShareConvViTs with minimal inferences, leveraging the higher resolution capabilities of the 4f system, similar to how FatNet optimised CNNs.

On the hardware side, real-world implementations and testing on physical 4f systems would provide valuable insights into the practical applicability and performance of these networks in real-life systems. It is evident that the 4f system presents certain challenges, particularly in noisy environments. It would be beneficial to conduct further experiments to study the effects of Poisson noise on both standard models and FatNet. Since noise can play a role in data augmentation, understanding its impact could lead to improvements in model training.

Finally, since this thesis has primarily contributed to deep learning, an important question remains: how would these models perform on real devices? Real-world applications may introduce issues such as alignment errors, which could significantly impact performance. Additionally, due to the computational expense of simulations, larger datasets like ImageNet were not used in this study. However, since FatNet is more advantageous with large datasets containing numerous classes, it would be valuable to test these models on real devices to observe these benefits.

References

[1] Julie Chang, Vincent Sitzmann, Xiong Dun, Wolfgang Heidrich, and Gordon Wetzstein. Hybrid optical-electronic convolutional neural networks with optimized diffractive optics for image classification. *Scientific Reports*, 8(1):12324, August 2018. Number: 1 Publisher: Nature Publishing Group.

[2] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition, September 2014.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016. ISSN: 1063-6919.

[5] Chen Xu, Xiubao Sui, Jia Liu, Yuhang Fei, Liping Wang, and Qian Chen. Transformer in optronic neural networks for image classification. *Optics & Laser Technology*, 165:109627, October 2023.

[6] Abhinav Agrawal and Namita Mittal. Using CNN for facial expression recognition: a study of the effects of kernel size and number of filters on accuracy. *The Visual Computer*, 36(2):405–412, February 2020.

[7] Shurui Li, M. Miscuglio, V. Sorger, and Puneet Gupta. Channel Tiling for Improved Performance and Accuracy of Optical Neural Network Accelerators. *ArXiv*, 2020.

[8] Cefa Karabağ, Mauricio Alberto Ortega-Ruíz, and Constantino Carlos Reyes-Aldasoro. Impact of Training Data, Ground Truth and Shape Variability in the Deep Learning-Based Semantic Segmentation of HeLa Cells Observed with Electron Microscopy. *Journal of Imaging*, 9(3):59, March 2023. Number: 3 Publisher: Multidisciplinary Digital Publishing Institute.

[9] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[10] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, LAS VEGAS, June 2016. IEEE.

[11] Ross Girshick. Fast R-CNN. In *IEEE International Conference on Computer Vision*, pages 1440–1448. IEEE, December 2015.

[12] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient Object Localization Using Convolutional Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 648–656, 2015.

[13] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation, May 2015. arXiv:1505.04597 [cs].

[14] Risheng Wang, Tao Lei, Ruixia Cui, Bingtao Zhang, Hongying Meng, and Asoke K. Nandi. Medical image segmentation using deep learning: A survey. *IET Image Processing*, 16(5):1243–1267, 2022. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1049/ipr2.12419.

[15] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image Style Transfer Using Convolutional Neural Networks. In *IEEE Conference onComputer Vision and Pattern Recognition*, pages 2414–2423, 2016.

[16] Ramazan Enisoglu and Veselin Rakocevic. Low-Latency Internet Traffic Identification using Machine Learning with Trend-based Features. In *2023 International Wireless Communications and Mobile Computing (IWCMC)*, pages 394–399, June 2023. ISSN: 2376-6506.

[17] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[18] Tomoki Uchiyama, Naoya Sogi, Koichiro Niinuma, and Kazuhiro Fukui. Visually Explaining 3D-CNN Predictions for Video Classification With an Adaptive Occlusion Sensitivity Analysis. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1513–1522, 2023.

[19] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image Super-Resolution Using Deep Convolutional Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, February 2016. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, June 2021. arXiv:2010.11929 [cs].

[22] Lei Jimmy Ba and Rich Caruana. Do Deep Nets Really Need to be Deep? In *Advances in neural information processing systems*, volume 27, 2014. _eprint: 1312.6184.

[23] Song Han, Huizi Mao, and William J. Dally. Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding. *arXiv: Computer Vision and Pattern Recognition*, 2016.

[24] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, Lecture Notes in Computer Science, pages 525–542, Cham, 2016. Springer International Publishing.

[25] M. Mitchell Waldrop. The chips are down for Moore's law. *Nature News*, 530(7589):144, February 2016. Cg_type: Nature News Section: News Feature.

[26] Xiubao Sui, Qiuhao Wu, Jia Liu, Qian Chen, and Guohua Gu. A Review of Optical Neural Networks. *IEEE Access*, 8:70773–70783, 2020.

[27] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[28] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3498–3505, June 2012.

[29] John R. Masters. HeLa cells 50 years on: the good, the bad and the ugly. *Nature Reviews Cancer*, 2(4):315–319, April 2002. Publisher: Nature Publishing Group.

[30] Raheleh Rahbari, Tom Sheahan, Vasileios Modes, Pam Collier, Catriona Macfarlane, and Richard M. Badge. A novel L1 retrotransposon marker for HeLa cell line identification. *BioTechniques*, 46(4):277–284, April 2009. Publisher: Taylor & Francis _eprint: https://doi.org/10.2144/000113089.

[31] Mateus Sangalli, Samy Blusseau, Santiago Velasco-Forero, and Jesus Angulo. Scale-Equivariant U-Net. In *33rd British Machine Vision Conference 2022, London, UK*, London, UK, November 2022. {BMVA} Press.

[32] Cefa Karabağ, Martin L. Jones, Christopher J. Peddie, Anne E. Weston, Lucy M. Collinson, and Constantino Carlos Reyes-Aldasoro. HeLa cell images with four labels (nuclear envelope, nucleus, rest of the cell, and background) for deep learning architecture training., May 2020. doi.org/10.5281/zenodo.3874949.

[33] D. Brito-Pacheco, C. Karabağ, C. Brito-Loeza, P. Giannopoulos, and C. C. Reyes-Aldasoro. Relationship Between Irregularities of the Nuclear Envelope and Mitochondria in HeLa cells Observed with Electron Microscopy, November 2023. Pages: 2023.11.14.567016 Section: New Results.

[34] D.G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, September 1999.

[35] PETER J. Burt and EDWARD H. Adelson. The Laplacian Pyramid as a Compact Image Code. In Martin A. Fischler and Oscar Firschein, editors, *Readings in Computer Vision*, pages 671–679. Morgan Kaufmann, San Francisco (CA), January 1987.

[36] Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Scaling Up Your Kernels to 31×31: Revisiting Large Kernel Design in CNNs. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11953–11965, June 2022. ISSN: 2575-7075.

[37] Hoda Sadeghzadeh, Somayyeh Koohi, and Ali Fele Paranj. Free-Space Optical Neural Network Based on Optical Nonlinearity and Pooling Operations. *IEEE Access*, 9:146533–146549, 2021. Conference Name: IEEE Access.

[38] Hao Wang, Jianqi Hu, Andrea Morandi, Alfonso Nardi, Fei Xia, Xuanchen Li, Romolo Savo, Qiang Liu, Rachel Grange, and Sylvain Gigan. Large-scale photonic computing with nonlinear disordered media. *Nature Computational Science*, 4(6):429–439, June 2024. Publisher: Nature Publishing Group.

[39] Mario Miscuglio, Zibo Hu, Shurui Li, Jonathan K. George, Roberto Capanna, Hamed Dalir, Philippe M. Bardet, Puneet Gupta, and Volker J. Sorger. Massively parallel amplitude-only Fourier neural network. *Optica*, 7(12):1812–1819, December 2020. Publisher: Optica Publishing Group.

[40] Xiujian Li, Zhengzheng Shao, Mengjun Zhu, and Junbo Yang. *Fundamentals of Optical Computing Technology: Forward the Next Generation Supercomputer*. Springer, New York, NY, 1st ed. 2018 edition edition, May 2018.

[41] Xing Lin, Yair Rivenson, Nezih T. Yardimci, Muhammed Veli, Yi Luo, Mona Jarrahi, and Aydogan Ozcan. All-optical machine learning using diffractive deep neural networks. *Science*, 361(6406):1004–1008, September 2018. Publisher: American Association for the Advancement of Science.

[42] Yichen Shen, Nicholas C. Harris, Scott Skirlo, Mihika Prabhu, Tom Baehr-Jones, Michael Hochberg, Xin Sun, Shijie Zhao, Hugo Larochelle, Dirk Englund, and Marin Soljačić. Deep learning with coherent nanophotonic circuits. *Nature Photonics*, 11(7):441–446, July 2017. Number: 7 Publisher: Nature Publishing Group.

[43] Tyler W. Hughes, Momchil Minkov, Yu Shi, and Shanhui Fan. Training of photonic neural networks through in situ backpropagation and gradient measurement. *Optica*, 5(7):864–871, July 2018. Publisher: Optica Publishing Group.

[44] C. S. Weaver and J. W. Goodman. A Technique for Optically Convolving Two Functions. *Applied Optics*, 5(7):1248–1249, July 1966. Publisher: Optica Publishing Group.

[45] S. Jutamulia and F. T. S. Yu. Overview of hybrid optical neural networks. *Optics & Laser Technology*, 28(2):59–72, March 1996.

[46] Qiuhao Wu, Xiubao Sui, Yuhang Fei, Chen Xu, Jia Liu, Guohua Gu, and Qian Chen. Multi-layer optical Fourier neural network based on the convolution theorem. *AIP Advances*, 11(5):055012, May 2021.

[47] Gaea-2 10 megapixel phase only LCOS-SLM (reflective). Publication Title: HOLOEYE Photonics AG.

[48] Dana Dudley, Walter M. Duncan, and John Slaughter. Emerging digital micromirror device (DMD) applications. In *MOEMS Display and Imaging Systems*, volume 4985, pages 14–25. SPIE, January 2003.

[49] Suganda Jutamulia and Toshimitsu Asakura. Fourier transform property of lens based on geometrical optics. In *Optical Information Processing Technology*, volume 4929, pages 80–85. SPIE, September 2002.

[50] Ronald N. Bracewell. *The Fourier transform and its applications*. McGraw-Hill series in electrical and computer engineering Circuits and systems. McGraw-Hill, Boston, 3. ed edition, 2000.

[51] James W. Cooley and John W. Tukey. An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation*, 19(90):297–301, 1965. Publisher: American Mathematical Society.

[52] Jack D. Gaskill. *Linear Systems, Fourier Transforms, and Optics*. Wiley-Interscience, New York, 1st edition edition, June 1978.

[53] Shane Colburn, Yi Chu, Eli Shilzerman, and Arka Majumdar. Optical frontend for a convolutional neural network. *Applied Optics*, 58(12):3179–3186, April 2019. Publisher: Optica Publishing Group.

[54] Moez Krichen. Convolutional Neural Networks: A Survey. *Computers*, 12(8):151, August 2023. Number: 8 Publisher: Multidisciplinary Digital Publishing Institute.

[55] Irwin Sobel and Gary Feldman. A 3×3 isotropic gradient operator for image processing. *Pattern Classification and Scene Analysis*, pages 271–272, January 1973.

[56] D. Marr, E. Hildreth, and Sydney Brenner. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167):187–217, January 1997. Publisher: Royal Society.

[57] J. G. M. Schavemaker, M. J. T. Reinders, J. J. Gerbrands, and E. Backer. Image sharpening by morphological filtering. *Pattern Recognition*, 33(6):997–1012, June 2000.

[58] Priyanka Patel and Amit Thakkar. The upsurge of deep learning for computer vision applications. *International Journal of Electrical and Computer Engineering (IJECE)*, 10(1):538–548, February 2020. Number: 1.

[59] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. CIFAR-10 and CIFAR-100 datasets, August 2009.

[60] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 448–456. PMLR, June 2015. ISSN: 1938-7228.

[61] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, April 1980.

[62] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, October 1986. Publisher: Nature Publishing Group.

[63] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity Mappings in Deep Residual Networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 630–645, Cham, 2016. Springer International Publishing.

[64] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[65] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning Deconvolution Network for Semantic Segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, Santiago, Chile, 2015. IEEE.

[66] Chao Peng, Xiangyu Zhang, Gang Yu, Guiming Luo, and Jian Sun. Large Kernel Matters – Improve Semantic Segmentation by Global Convolutional Network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4353–4361, 2017.

[67] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, December 2017.

[68] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization, July 2016. arXiv:1607.06450 [cs, stat].

[69] Seunghoon Lee, Seunghyun Lee, and Byung Cheol Song. Improving Vision Transformers to Learn Small-Size Dataset From Scratch. *IEEE Access*, 10:123212–123224, 2022. Conference Name: IEEE Access.

[70] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, June 2016. ISSN: 1063-6919.

[71] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep Networks with Stochastic Depth. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, Lecture Notes in Computer Science, pages 646–661, Cham, 2016. Springer International Publishing.

[72] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random Erasing Data Augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13001–13008, April 2020. Number: 07.

[73] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *CoRR*, December 2014.

[74] Haoran Zhu, Boyuan Chen, and Carter Yang. Understanding Why ViT Trains Badly on Small Datasets: An Intuitive Perspective, February 2023. arXiv:2302.03751 [cs].

[75] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 548–558, October 2021. ISSN: 2380-7504.

[76] Qiuhao Wu, Xiubao Sui, Liping Wang, and Qian Chen. Convolutional Free-space Optical Neural Networks for Image Recognition, 2021. _eprint: 2103.07862.

[77] Kuang Liu, Mingmin Zhang, and Zhigeng Pan. Facial Expression Recognition with CNN Ensemble. In *2016 International Conference on Cyberworlds (CW)*, pages 163–166, September 2016.

[78] Christopher Pramerdorfer and Martin Kampel. Facial Expression Recognition using Convolutional Neural Networks: State of the Art, December 2016. arXiv:1612.02903 [cs].

[79] Yousif Khaireddin and Zhuofa Chen. Facial Emotion Recognition: State of the Art Performance on FER2013, May 2021. arXiv:2105.03588 [cs].

[80] M. Eren Akbiyik. Data Augmentation in Training CNNs: Injecting Noise to Images, July 2023. arXiv:2307.06855 [cs].

[81] Luc Haspeslagh, Jeroen De Coster, Olalla Varela Pedreira, Ingrid De Wolf, Bert Du Bois, Agnes Verbist, Rita Van Hoof, Myriam Willegems, Sabrina Locorotondo, George Bryce, Jan Vaes, Bert van Drieenhuizen, and Ann Witvrouw. Highly reliable CMOS-integrated 11MPixel SiGe-based micro-mirror arrays for high-end industrial applications. In *2008 IEEE International Electron Devices Meeting*, pages 1–4, December 2008. ISSN: 2156-017X.

[82] Jan-Uwe Schmidt, Ulrike A. Dauderstaedt, Peter Duerr, Martin Friedrichs, Thomas Hughes, Thomas Ludewig, Dirk Rudloff, Tino Schwaten, Daniela Trenkler, Michael Wagner, Ingo Wullinger, Andreas Bergstrom, Peter Bjoernangen, Fredrik Jonsson, Tord Karlin, Peter Ronnholm, and Torbjorn Sandstrom. High-speed one-dimensional spatial light modulator for Laser Direct Imaging and other patterning applications. In *MOEMS and Miniaturized Systems XIII*, volume 8977, pages 167–176. SPIE, March 2014.

[83] Eloy Schultz, Joris de Nijs, Bin Shi, and Ripalta Stabile. Optical 4F Correlator for Acceleration of Convolutional Neural Networks: 25th Annual Symposium of the IEEE Photonics Benelux Chapter. In *25th Annual Symposium of the IEEE Photonics Benelux*, Belgium, November 2021.

[84] Jun Dai, Xiaowen Dong, Chong Li, and Jian-Jun He. On-chip 4F-system based on concave mirrors for optical neural networks. In *Holography, Diffractive Optics, and Applications XIII*, volume 12768, pages 292–297. SPIE, November 2023.

[85] Jyoti Rawat, Doina Logofătu, and Sruthi Chiramel. Factors Affecting Accuracy of Convolutional Neural Network Using VGG-16. In Lazaros Iliadis, Plamen Parvanov Angelov, Chrisina Jayne, and Elias Pimenidis, editors, *Proceedings of the 21st EANN (Engineering Applications of Neural Networks) 2020 Conference*, pages 251–260, Cham, 2020. Springer International Publishing.

[86] Wei Wang, Liqiang Zhu, and Baoqing Guo. Reliable identification of redundant kernels for convolutional neural network compression. *Journal of Visual Communication and Image Representation*, 63:102582, August 2019.

[87] Yuzhe Ma, Ran Chen, Wei Li, Fanhua Shang, Wenjian Yu, Minsik Cho, and Bei Yu. A Unified Approximation Framework for Compressing and Accelerating Deep Neural Networks. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 376–383, November 2019. ISSN: 2375-0197.

[88] Puneet Gupta and Shurui Li. 4F optical neural network acceleration: an architecture perspective. In *AI and Optical Data Sciences III*, volume 12019, pages 77–84. SPIE, March 2022.

[89] Jie Chen, Huarong Gu, Hongwei Zhang, Jie Zhong, and Yi Xiong. Multilayer optoelectronic hybrid convolutional neural network with an optical 4f-system recurrent structure. In *Holography, Diffractive Optics, and Applications XIII*, volume 12768, pages 120–129. SPIE, November 2023.

[90] P. K. Diederik. Adam: a method for stochastic optimization. *(No Title)*, 2014.

[91] Baopeng Li, Okan K. Ersoy, Caiwen Ma, Zhibin Pan, Wansha Wen, and Zongxi Song. A 4F optical diffuser system with spatial light modulators for image data augmentation. *Optics Communications*, 488:126859, 2021.

[92] Ritik Dixit, Rishika Kushwah, and Samay Pashine. Handwritten Digit Recognition using Machine and Deep Learning Algorithms. *International Journal of Computer Applications*, 176(42):27–33, July 2020. arXiv:2106.12614 [cs].

[93] Satoru Mizusawa and Yuichi Sei. Interlayer Augmentation in a Classification Task. In *2021 International Conference on Computing, Electronics & Communications Engineering (iCCECE)*, pages 59–64, August 2021.

[94] Anish Shah, Eashan Kadam, Hena Shah, Sameer Shinde, and Sandip Shingade. Deep Residual Networks with Exponential Linear Unit. In *Proceedings of the Third International Symposium on Computer Vision and the Internet*, VisionNet'16, pages 59–65, New York, NY, USA, September 2016. Association for Computing Machinery.

[95] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs), February 2016. arXiv:1511.07289 [cs].

[96] Yanming Chen, Xiang Wen, Yiwen Zhang, and Weisong Shi. CCPrune: Collaborative channel pruning for learning compact convolutional networks. *Neurocomputing*, 451:35–45, September 2021.

[97] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning Filters for Efficient ConvNets. In *International Conference on Learning Representations*, April 2017.

[98] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. The Oxford-IIIT Pet Dataset, June 2012.

[99] Benjamin Yat-Ming Yung and Amy Meei-Shuu Bor. Identification of high-density lipoprotein in serum to determine anti-cancer efficacy of doxorubicin in HeLa cells. *International Journal of Cancer*, 50(6):951–957, 1992. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/ijc.2910500622.

[100] Song-Ling Zhang, Yi-Shu Wang, Tong Zhou, Xiao-Wei Yu, Zhen-Tong Wei, and Yu-Lin Li. Isolation and characterization of cancer stem cells from cervical cancer HeLa cells. *Cytotechnology*, 64(4):477–484, August 2012.

[101] Jing Yang, Beibei Zhang, Zifei Qin, Shishi Li, Jinjin Xu, Zhihong Yao, Xiaojian Zhang, Frank J Gonzalez, and Xinsheng Yao. Efflux excretion of bisdemethoxycurcumin-O-glucuronide in UGT1A1-overexpressing HeLa cells: Identification of breast cancer resistance protein (BCRP) and multidrug resistance-associated proteins 1 (MRP1) as the glucuronide transporters. *BioFactors*, 44(6):558–569, 2018. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/biof.1452.

[102] Zhaoshou Yang, Hye-Jin Ahn, and Ho-Woo Nam. Gefitinib Inhibits the Growth of Toxoplasma gondii in HeLa Cells. *The Korean Journal of Parasitology*, 52(4):439–441, August 2014.

[103] Zhaoxia Zhang, Haorong Gu, Qi Li, Jun Zheng, Shinuo Cao, Changjiang Weng, and Honglin Jia. GABARAPL2 Is Critical for Growth Restriction of Toxoplasma gondii in HeLa Cells Treated with Gamma Interferon. *Infection and Immunity*, 88(5):10.1128/iai.00054–20, April 2020. Publisher: American Society for Microbiology.

[104] Raquel Arruda Sanfelice, Laís Fernanda Machado, Larissa Rodrigues Bosqui, Milena Menegazzo Miranda-Sapla, Fernanda Tomiotto-Pellissier, Gabriela de Alcântara Dalevedo, Dielle Ioris, Guilherme Fonseca Reis, Luciano Aparecido Panagio, Italmar Teodorico Navarro, Juliano Bordignon, Ivete Conchon-Costa, Wander Rogério Pavanelli, Ricardo Sergio Almeida, and Idessania Nazareth Costa. Activity of rosuvastatin in tachyzoites of Toxoplasma gondii (RH strain) in HeLa cells. *Experimental Parasitology*, 181:75–81, October 2017.

[105] Bei Zhang, Jia-yin Liu, Jin-shun Pan, Su-ping Han, Xiao-xing Yin, Bing Wang, and Gang Hu. Combined Treatment of Ionizing Radiation With Genistein on Cervical Cancer HeLa Cells. *Journal of Pharmacological Sciences*, 102(1):129–135, January 2006.

[106] W. Ziegler, P. Birkenfeld, and K. R. Trott. The effect of combined treatment of HeLa cells with actinomycin D and radiation upon survival and recovery from radiation damage. *Radiotherapy and Oncology*, 10(2):141–148, October 1987.

[107] N Zinberg and A Kohn. Dimethyl sulfoxide protection of HeLa cells against ionizing radiation during the growth cycle. *Israel journal of medical sciences*, 7(6):719–723, June 1971.

[108] Yuhuang Zheng, Huaying Zhou, Chunying Zhang, Yan He, Hui Li, Zi Chen, and Meng Liu. The Apoptosis-Inducing Effects of HIV vpr Recombinant Eukaryotic Expression Vectors with Different Mutation Sites on Transfected Hela Cells. *Current HIV Research*, 7(5):519–525, September 2009.

[109] M. Tominaga, E. Kumagai, and S. Harada. Effect of electrical stimulation on HIV-1-infected HeLa cells cultured on an electrode surface. *Applied Microbiology and Biotechnology*, 61(5):447–450, June 2003.

[110] Talia Hahn, Ami Schattner, Zeev T. Handzel, Stanley Levin, and Zvi Bentwich. Possible role of natural cytotoxic activity in the pathogenesis of AIDS. *Clinical Immunology and Immunopathology*, 50(1, Part 1):53–61, January 1989.

[111] Christopher J. Peddie, Martin L. Jones, and Lucy M. Collinson. Serial Block Face SEM of HeLa cell pellet with 10 nm pixels and 50 nm slices (benchmark dataset), May 2019. 10.6019/EMPIAR-10094.

[112] Christopher J. Peddie, Martin L. Jones, and Lucy M. Collinson. Cropped regions from Serial Block Face SEM of HeLa cell pellet with 10 nm pixels and 50 nm slices (benchmark dataset), August 2020. 10.6019/EMPIAR-10478.

[113] Jonas Dippel, Matthias Lenga, Thomas Goerttler, Klaus Obermayer, and Johannes Höhne. Transfer Learning for Segmentation Problems: Choose the Right Encoder and Skip the Decoder, July 2022. arXiv:2207.14508 [cs].

[114] Kavitha Sundarrajan, Baskaran Kuttva Rajendran, and Dhanapriya Balasubramanian. Fusion of Ensembled UNET and Ensembled FPN for Semantic Segmentation. *Traitement du Signal*, 40(1):297–307, February 2023.

[115] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. ICNet for Real-Time Semantic Segmentation on High-Resolution Images. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, Lecture Notes in Computer Science, pages 418–434, Cham, 2018. Springer International Publishing.

[116] Justin Edwards and Mohamed El-Sharkawy. uICNet: Lightweight Image Segmentation. In *2022 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pages 99–104, October 2022.

[117] Joeri R. Hermans, Gerasimos Spanakis, and Rico Möckel. Accumulated Gradient Normalization. In *Proceedings of the Ninth Asian Conference on Machine Learning*, pages 439–454. PMLR, November 2017. ISSN: 2640-3498.

[118] Riad Ibadulla, Thomas M. Chen, and Constantino Carlos Reyes-Aldasoro. FatNet: High-Resolution Kernels for Classification Using Fully Convolutional Optical Neural Networks. *AI*, 4(2):361–374, June 2023. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

[119] Yann LeCun, Lawrence D Jackel, Léon Bottou, Corinna Cortes, John S Denker, Harris Drucker, Isabelle Guyon, Urs A Muller, Eduard Sackinger, Patrice Simard, and others. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, 261(276):2, 1995.

[120] Waseem Rawat and Zenghui Wang. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Computation*, 29(9):2352–2449, September 2017. Conference Name: Neural Computation.

[121] Leiyu Chen, Shaobo Li, Qiang Bai, Jing Yang, Sanlong Jiang, and Yanming Miao. Review of Image Classification Algorithms Based on Convolutional Neural Networks. *Remote Sensing*, 13(22):4712, January 2021. Number: 22 Publisher: Multidisciplinary Digital Publishing Institute.

[122] R. Anand, T. Shanthi, M. S. Nithish, and S. Lakshman. Face Recognition and Classification Using GoogleNET Architecture. In Kedar Nath Das, Jagdish Chand Bansal, Kusum Deep, Atulya K. Nagar, Ponnambalam Pathipooranam, and Rani Chinnappa Naidu, editors, *Soft Computing for Problem Solving*, pages 261–269, Singapore, 2020. Springer.

[123] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014.

[124] Mei Wang and Weihong Deng. Deep face recognition: A survey. *Neurocomputing*, 429:215–244, March 2021.

[125] Wei Wang, Yujing Yang, Xin Wang, Weizheng Wang, and Ji Li. Development of convolutional neural network and its application in image classification: a survey. *Optical Engineering*, 58(4):040901, April 2019. Publisher: SPIE.

[126] Alhassan Mumuni and Fuseini Mumuni. CNN Architectures for Geometric Transformation-Invariant Feature Representation in Computer Vision: A Review. *SN Computer Science*, 2(5):340, June 2021.

[127] Hieu Pham, Zihang Dai, Qizhe Xie, and Quoc V. Le. Meta Pseudo Labels. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11557–11568, 2021.

[128] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. AutoAugment: Learning Augmentation Strategies From Data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 113–123, 2019.

[129] Antonio Bruno, Davide Moroni, and Massimo Martinelli. Efficient Adaptive Ensembling for Image Classification, June 2022. Publication Title: arXiv e-prints ADS Bibcode: 2022arXiv220607394B.

[130] David G. Voelz. *Computational Fourier Optics: A MATLAB® Tutorial*. SPIE, 1000 20th Street, Bellingham, WA 98227-0010 USA, January 2011.

[131] Michal Miler Maciej Grochowicz. PyOptica Documentation. *Gitlab*, 2020.

[132] Junchang Li, Zujie Peng, and Yunchang Fu. Diffraction transfer function and its calculation of classic diffraction formula. *Optics Communications*, 280(2):243–248, December 2007.

[133] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009. ISSN: 1063-6919.

[134] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts. In *International Conference on Learning Representations*. arXiv, May 2017. arXiv:1608.03983 [cs, math].

[135] Riad Ibadulla, Constantino C. Reyes-Aldasoro, and Thomas M. Chen. Fat-U-Net: non-contracting U-Net for free-space optical neural networks. In *AI and Optical Data Sciences V*, volume 12903, pages 42–52. SPIE, March 2024.

[136] Tao Lei and Asoke Kumar Nandi. *Image segmentation: principles, techniques, and applications*. Wiley, Hoboken, NJ, 2023.

[137] Liang-Chieh Chen, Alexander Hermans, George Papandreou, Florian Schroff, Peng Wang, and Hartwig Adam. MaskLab: Instance Segmentation by Refining Object Detection With Semantic and Direction Features. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4013–4022, 2018.

[138] Mehmet Sezgin and Bülent Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):146–165, January 2004. Publisher: SPIE.

[139] Stephen Gould, Tianshi Gao, and Daphne Koller. Region-based Segmentation and Object Detection. In *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009.

[140] Nameirakpam Dhanachandra, Khumanthem Manglem, and Yambem Jina Chanu. Image Segmentation Using K -means Clustering Algorithm and Subtractive Clustering Algorithm. *Procedia Computer Science*, 54:764–771, January 2015.

[141] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*, 59(2):167–181, September 2004.

[142] Xu Chen, Bryan M. Williams, Srinivasa R. Vallabhaneni, Gabriela Czanner, Rachel Williams, and Yalin Zheng. Learning Active Contour Models for Medical Image Segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11632–11640, 2019.

[143] Ying Yu, Chunping Wang, Qiang Fu, Renke Kou, Fuyu Huang, Boxiong Yang, Tingting Yang, and Mingliang Gao. Techniques and Challenges of Image Segmentation: A Review. *Electronics*, 12(5):1199, January 2023. Number: 5 Publisher: Multidisciplinary Digital Publishing Institute.

[144] Jonas Dippel, Steffen Vogler, and Johannes Höhne. Towards Fine-grained Visual Representations by Combining Contrastive Learning with Image Reconstruction and Attention-weighted Pooling, February 2022. arXiv:2104.04323 [cs].

[145] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, April 2018. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[146] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask R-CNN. In *IEEE international conference on computer vision 2017*, pages 2961–2969, 2017.

[147] Francois Chollet. Xception: Deep Learning With Depthwise Separable Convolutions. In *EEE Conference on Computer Vision and Pattern Recognition*, pages 1251–1258, 2017.

Appendix A

Peer-reviewed publications

AI

MDPI

Article

FatNet: High-Resolution Kernels for Classification Using Fully Convolutional Optical Neural Networks

Riad Ibadulla *, Thomas M. Chen and Constantino Carlos Reyes-Aldasoro

Department of Computer Science, City University of London, Northampton Square, London EC1V 0HB, UK
* Correspondence: riad.ibadulla@city.ac.uk

Abstract: This paper describes the transformation of a traditional in silico classification network into an optical fully convolutional neural network with high-resolution feature maps and kernels. When using the free-space 4f system to accelerate the inference speed of neural networks, higher resolutions of feature maps and kernels can be used without the loss in frame rate. We present FatNet for the classification of images, which is more compatible with free-space acceleration than standard convolutional classifiers. It neglects the standard combination of convolutional feature extraction and classifier dense layers by performing both in one fully convolutional network. This approach takes full advantage of the parallelism in the 4f free-space system and performs fewer conversions between electronics and optics by reducing the number of channels and increasing the resolution, making this network faster in optics than off-the-shelf networks. To demonstrate the capabilities of FatNet, it was trained with the CIFAR100 dataset on GPU and the simulator of the 4f system. A comparison of the results against ResNet-18 shows 8.2 times fewer convolution operations at the cost of only 6% lower accuracy. This demonstrates that the optical implementation of FatNet results in significantly faster inference than the optical implementation of the original ResNet-18. These are promising results for the approach of training deep learning with high-resolution kernels in the direction toward the upcoming optics era.

Keywords: optical neural networks; high resolution; convolutional neural networks

Citation: Ibadulla, R.; Chen, T.; Reyes-Aldasoro, C.C. FatNet: High-Resolution Kernels for Classification Using Fully Convolutional Optical Neural Networks. *AI* **2023**, *4*, 361–374. https://doi.org/10.3390/ai4020018

Academic Editor: Andrea Calimera

Received: 10 February 2023
Revised: 25 February 2023
Accepted: 24 March 2023
Published: 3 April 2023

1. Introduction

One of the major problems of the modern deep learning approach is the speed of training and inference of architectures where there are a very large number of parameters to train. Computer vision, which can involve a large number of images with very slight differences, is considered to be one of the most complex problem areas for AI. Within the deep learning approaches, convolutional neural networks (CNNs) have become a standard approach for various computer vision problems. Recently, CNNs have been successfully applied to image classification [1], object detection [2], localization [3], and segmentation [4], among many other applications. CNNs are suitable for computer vision tasks because neurons in CNNs are only connected to the pixels of their receptive field, rather than to every single neuron of the next layer as in fully connected networks. This approach reduces the number of trainable parameters, which accelerates the inference and makes the neural network more immune to overfitting. Although CNNs are computationally less expensive than fully connected neural networks, accelerating CNNs is also an important task, especially with the ever growing number of images and videos that are captured.

There are many techniques to accelerate deep learning training, e.g., using shallow networks, pruning redundant weights, or using lower quantization levels [5]. In addition, hardware accelerators can be used to speed up the training and inference of neural networks, for example, in application-specific integrated circuits (ASICs), which can outperform standard CPUs and GPUs [6]. Large tech companies are actively working on their AI accelerators, such as Google's TPU [7], Intel's Loihi [8], and IBM's TrueNorth [9].

Unfortunately, these accelerators are starting to face limitations in the post Moore's law era, since the computational power of the processors is not improving at the same pace as before [10].

Optical processors are an interesting alternative to processing data with silicon chips. Optical computing uses photons of light, instead of electrons, as the information carrier for data processing [11]. Since Moore's law does not affect optical computing, optical accelerators can be used for deep learning, offering advantages such as the high bandwidth of the light beam, high speed, zero resistance, lower energy consumption, and immunity to overheating [12]. There are two main approaches to optical neural networks: free space using spatial light modulators (SLM) [13,14] or silicon photonics approach using Mach–Zehnder interferometers (MZI) [15,16]. Unlike the silicon photonics approach, free-space optics uses wireless light propagation through a medium which can be air, outer space or vacuum. Although the silicon photonics approach is faster, as its clock speed can reach several GHz, it is inferior to the free-space system in parallelism [17].

This research is focused on the 4f free-space approach as described in Li et al. [13], which takes advantage of the parallelism of free-space optics. The 4f free-space optical system can be used to perform convolution operations faster than traditional electronic processors.

The Fourier transform is a well-known mathematical operation that decomposes a signal into its fundamental sinusoids in the frequency domain that, when combined, form the original function [18]. A Fourier transform is initially defined over one dimension, and can be extended to two or more dimensions [19]. The computational complexity of this process increases with the dimensions of the data, and even with fast methods such as the fast Fourier transform [20], transforming large data can take considerable resources with complexity in the order of $O(n^2 log(n))$, where n^2 is the number of pixels of an image [21]. On the other hand, performing a 2D Fourier transform in free-space optics can be easily achieved by passing the light through the convex lens, where the light only has to travel two focal distances (f) from the lens [22].

Taking the convolution theorem into account, the convolution of two signals can be represented as the inverse Fourier transform of the pointwise product of their Fourier transforms [18]. The 4f correlator is based on the Fourier transform properties of the convex lenses [23] and performs the convolution operation based on the convolution theorem. Any convex lens projects a Fourier transform of the input object located on the front focal plane onto the back focal plane [23], where it can be pointwise multiplied by the kernel in the Fourier domain. After passing through the second lens, it can be converted back into the space domain. The system is called 4f because the light in the 4f system travels four focal distances of the lens. Hence, the 4f approach can accelerate convolutional neural networks by performing the Fourier transforms at the speed of light. The parallelism advantage of the 4f system comes from the theoretically infinite resolution that is bounded in reality by the resolution of the modulators and the camera.

The first optical convolution technique with the 4f system was described by Weaver and Goodman [24] in 1966. It was not used for the acceleration of neural networks until neural networks started gaining popularity in the 21st century [25]. A standard 4f optical system consists of an input source, two convex lenses, two light modulators, and a sensor (see Figure 1). The input source is the laser emitting the light modulated right in the beginning with the input image by altering the light intensity. The modulated light passes through the first convex lens after travelling the focal distance of the lens and is projected onto the focal plane, where the Fourier transform of the input is formed. On the focal plane using another modulator, the input is element-wise multiplied with the kernel in the frequency domain. After the multiplication in the Fourier domain, the light passes through the second lens to perform the inverse Fourier transform and is captured by the camera or the array of photodetectors. In some cases, instead of the modulator, the fixed phase mask is used to perform the multiplication in the Fourier plane, as demonstrated in Chang et al. [14].

The 4f system is used in combination with the electronic compound, called an optical–electronic hybrid system [14]. This system is used only for inference, and training is performed using the simulator. The networks were trained using the simulator, and the phase mask of the trained kernels of the first layer was fabricated. Those fabricated kernels were used only for the inference of the pre-trained first layer. Hence, the inference of the first convolutional layer is optically computed, and the output of the electronic network is then fitted into the electronic portion of the network. This allows the multiplication to be performed passively, i.e., without energy consumption or latency. It also enables high speed-up, since the first layer of the network is usually the heaviest due to the high resolutions, which the free-space optics can handle for free. Since the optical–electronic hybrid system uses kernel tiling, this system can perform several convolution operations of the first layer in parallel without losing frame rate and power. However, a passive architecture such as this lacks flexibility and can only be used with one set of kernels, meaning it cannot be reused for all network layers. This is the reason for considering only active 4f architecture in our approach, allowing the device to perform all convolutional layers of the network by altering the kernel on the Fourier plane.

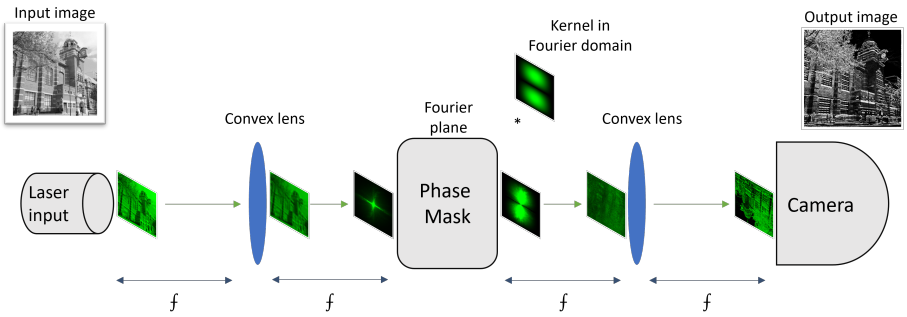


Figure 1. Graphical representation of the 4f system performing the convolution operation, consisting of the input plane (laser), the convex lens, Fourier plane (modulator or phase mask), and another convex lens and the camera separated from each other by one focal distance of the lens. When light passes through the lens, it forms a 2D Fourier transform on the Fourier plane, where it can be multiplied by the kernel in the frequency domain. The light then passes through the second lens, which converts it back into the space domain, where the output is read by the camera.

Unlike standard neural networks, optical neural networks involve various bottlenecks and constraints. Since the read-out camera captures the intensity of light, which is the square of the amplitude, it is impossible to perform the computations with negative values. One of the possible ways for getting around this problem is the non-negative constraint, which can significantly affect the accuracy. One way around this constraint is called pseudo-negativity, which can address the restriction to positive values by doubling the number of filters [14]. This method only uses positive values for the kernel by labeling half of the kernels as positive and the other half negative. After the read-out, the results of 'positive' convolutions are subtracted from the 'negative' results, thus providing the correct outcome for the convolution operation. Another bottleneck is the resolution of the modulator and the camera. Although modern cameras can capture up to 4K resolution, this limit does not allow many channels to be tiled and high-resolution feature maps to be used in combination with tiling.

Despite the many advantages of 4f systems, they have not been popular among the modern AI accelerators. The main problem lies in the very slow cameras and light modulators used in the system. However, there is the possibility to gain acceleration using parallelism and through simultaneously performing several convolution operations. For example, Li et al. [13] proposed kernel, channel, and mixed tiling approaches to better utilize the resolution of the 4f system. Their approaches enable all convolution operations for specific output channels and sum them using one inference through the 4f system. In a technique used by [13], zero-padding is applied to the input channels, which are tiled into

one big input block, while their corresponding kernels are tiled in the same manner, forming a kernel block. This method takes advantage of the massive parallelism of free-space optics. It performs all convolutions of each output channel of the convolutional layer, including the channel summation via one optical inference. By optically convolving the input block and the kernel, the summation of all convolutions of those particular input channels with output channels appears in the middle of the output tensor. This significantly reduces the number of conversions between optics and electronics. That is why it is essential to use the high-resolution capabilities of the 4f system.

Tiny kernel resolutions have become one of the nuances of building CNN architectures. Kernel sizes of 3×3 or 5×5 are now the standard for CNNs [1]. Although sometimes in ResNet architectures, a large kernel size can be seen in the first layer of the networks [26]. Theoretically, having a small kernel size has a range of advantages. The reduction of kernel size not only increases the computational efficiency during training but also decreases the number of trainable parameters, thereby increasing the robustness of the network against overfitting [27,28]. Modern neural networks are all trained on CPU/GPU, whose training time depends on the number of parameters. This led to the development of architectures with a very small resolution of kernels. For the same reason, all classifier architectures were developed in the cone shape, where the image is pooled down at every layer, making it faster for the CPU/GPU process. However, this works entirely differently for optical neural networks. Due to the nature of free-space optics, the use of large kernels in 4f system-based neural networks will not affect the inference time. Unfortunately, almost all the attempts to train the convolutional neural networks on the 4f system are based on the standard convolutional cone-shaped architectures.

To overcome underutilization of the 4f system, we propose FatNet, which takes advantage of the high-resolution capabilities of the 4f system by using fewer channels and larger input/kernel resolution in CNNs. Since the resolution does not affect the speed of inference in the 4f system, increasing the resolution and reducing the number of channels results in the network performing fewer convolution operations. This means fewer translations from optics to electronics, since the main bottleneck of the system is based on optics–electronics conversions. Our approach does not require pooling between most layers, which speeds up the inference even more for the small cost of a loss in accuracy.

2. Materials

We trained our network with the CIFAR-100 dataset (see Figure 2) and chose ResNet-18 as the backbone network. The CIFAR-100 (Canadian Institute For Advanced Research) dataset consists of 60,000 images of 32×32 resolution. It is split into 20 superclasses sub-grouped into 100 classes, with 600 images per class [29]. Only 50,000 images are used for training, and the other 10,000 data samples are in the test set. The similarity of classes under the same superclass in CIFAR-100 makes it harder to train.

Shah et al. [30] managed to train CIFAR-100 using different ResNet models, including their variation, where ELU (exponential linear unit) [31] was used as an activation function. Their test error on standard ResNet-101 achieved 27.23%. For this reason, we decided to use residual networks in our experiments. In our research, we have limited our focus to serial networks that do not contain branching structures. Therefore, networks that use depthwise convolutions, such as the highly accurate EfficientNet-B0 [32] with an accuracy rate of 88.1%, are outside the scope of our analysis.

ResNet-18 is a CNN, one of five networks introduced in He et al. [26] for the ImageNet dataset [33]. The feature distinguishing these networks from others is the residual connections between layers. Formally, He et al. [26] noted the blocks of the networks as:

$$y = F(x, \{W_i\}) + x \tag{1}$$

where x and y are the input and the output of the residual block, and $F(x, \{W_i\})$ represents the building block of the residual layer, which can contain one or several weight layers.

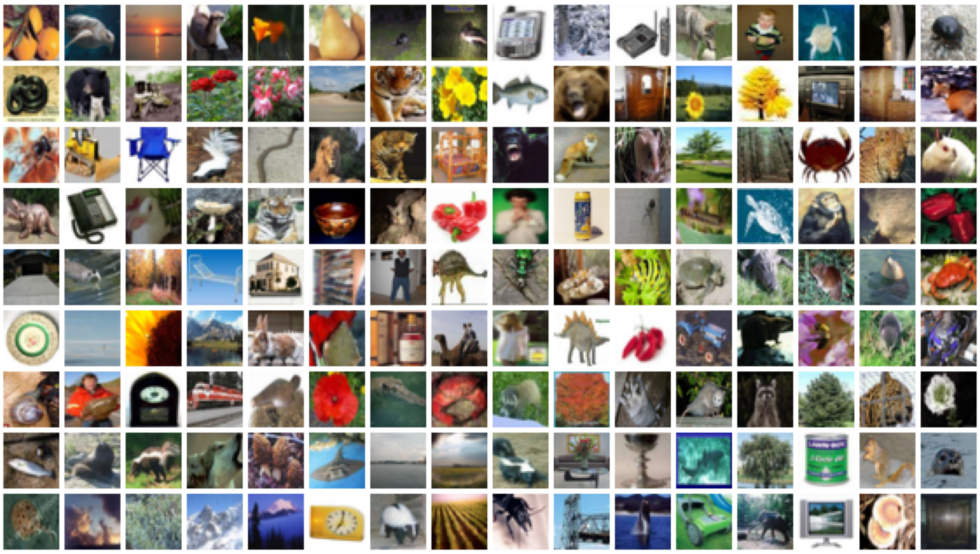


Figure 2. Illustration of CIFAR-100 dataset examples. CIFAR-100 contains tiny images of 100 classes, with a resolution of 32 × 32.

Residual connections are the connections in which one or more layers are skipped. In ResNet, those connections perform identity mapping, and the outputs of these connections are added to the output of the stacked layers. This configuration allows the use of deeper networks by avoiding vanishing/exploding gradient problems.

PyTorch was used to train all our networks [34]. PyTorch is an open source machine learning framework originally developed by Meta AI. We used PyTorch for its flexibility and ease in creating custom neural network layers. One example is the simulation of our optical layer, which we also built using PyTorch. PyTorch was also used by Miscuglio [35] to precisely simulate an actual 4f system.

3. Methods

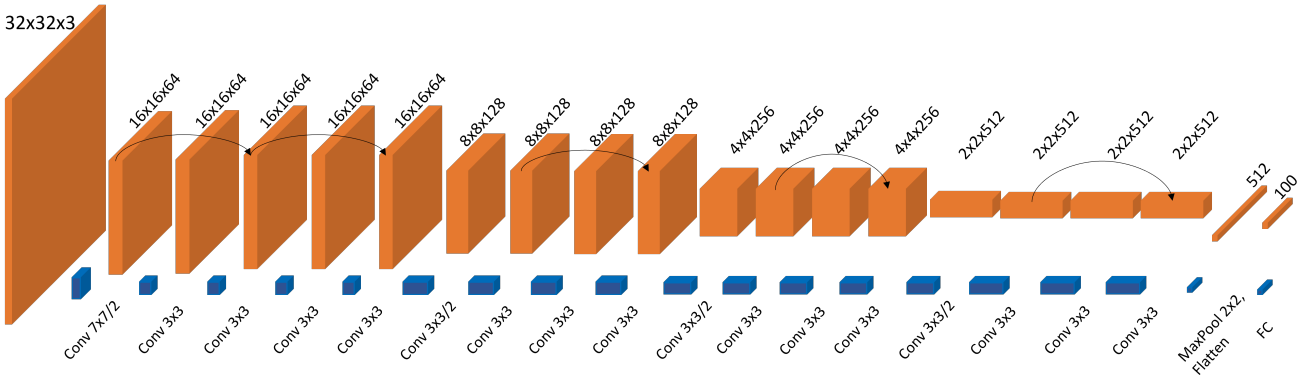
Nearly all classifier CNNs are cone-shaped and use either strides or pooling layers to reduce the resolution of the feature map [28]. This architecture has several advantages. The main advantage is the training speed, since the network gets simpler after each feature extraction and ends up with very low-resolution feature maps, which are flattened and passed to the fully connected layers for further classification. However, this kind of structure became standard only due to the dominance of electronic computing. Unlike in electronics, having larger resolutions for inputs and kernels in the 4f system does not affect the speed of inference, which makes it essential to explore new architectures that are compatible with optics. Our approach is called FatNet, due to its barrel shaped structure and most of the kernels having the same resolution as the feature maps (see Figure 3b).

By having larger feature maps and kernel sizes in the classifier CNN, we can ensure full utilization of the free-space optics. Although higher resolutions come with the problem of overfitting, our approach uses the same number of trainable parameters as the standard approach. Essentially, we have created the following rules for turning any classifier into a FatNet:

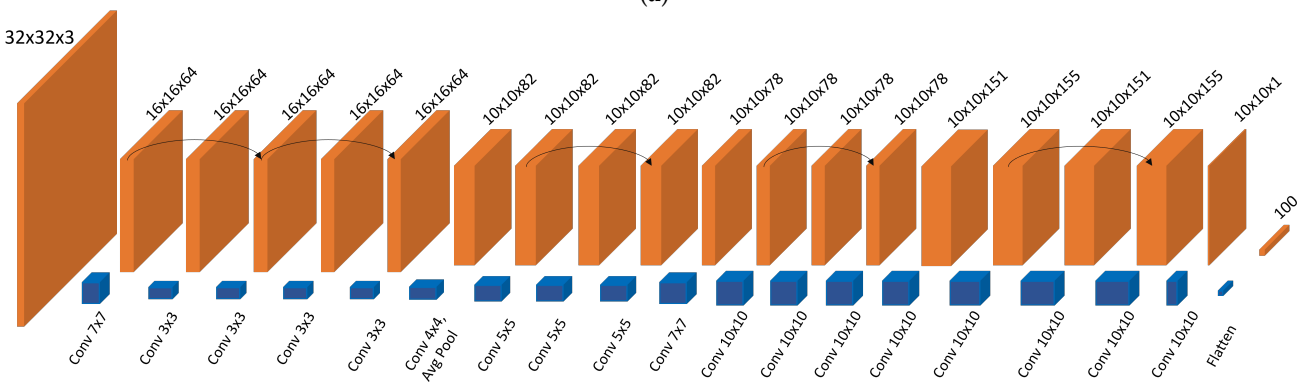
1. The FatNet should preserve the same number of layers as the original network to keep the same number of non-linear activation functions.
2. The FatNet should keep precisely the same architecture as the original network on the shallow layers until the shape of the feature maps pools down to the shape where the number of elements of the feature map is less than or equal to the number of classes.
3. FatNet has the same total number of pixels of the feature maps at the output of each layer as the original networks. Hence, since the feature maps' shape stays constant

and does not use pooling, the new number of output channels needs to be calculated, which will be less than for the original network.

4. FatNet has the same number of trainable parameters per layer as the original network. Since we have reduced the number of output channels based on the third rule, the number of trainable parameters has also been reduced. Hence, a new kernel size needs to be calculated based on the number of output channels.



(**a**)



(**b**)

Figure 3. Architecture comparison of our modified ResNet-18 used to train CIFAR-100 and FatNet constructed from ResNet-18 specifically for CIFAR-100 classification. (**a**) ResNet-18 architecture, slightly modified from the original. Our version does not use strides, since optics cannot perform strides in convolutions. We also skipped the second non-residual convolutional layer to make it more compatible with CIFAR-100. (**b**) FatNet derived from ResNet-18 for CIFAR-100. Compared with ResNet-18, this architecture contains fewer channels but larger resolutions. Kernel resolutions can go up to 10×10, while feature maps are not pooled lower than 10×10. The last layer is a 10×10 matrix flattened to form a vector of 100 elements, each representing a class of CIFAR-100.

It is also important to remember that FatNet is more efficient when the number of classes is significant; for example, ImageNET contains 20,000 classes. We chose ResNet-18 as the backbone network for proof of concept and trained the network with the CIFAR-100 dataset. We chose CIFAR-100 over CIFAR-10 due to the larger number of classes and the ability to keep the feature maps in the square shape of 10×10. It is essential to know that one of the limitations of the 4f-based convolution is the failure to perform the convolutions with the stride. Since most off-the-shelf networks contain stridden convolutions, this can be a potential problem. However, we can get around the problem by replacing the stridden convolutions with the combination of standard convolution and pooling. Because we do not want to reduce the resolution of our feature maps, we decided to ignore the strides in our ResNet-18 architecture and use 2×2 MaxPooling after the first layer.

No modification of the first five layers is carried out, since they all have 16×16 resolution. For the following layers, we have calculated the number of pixels in each feature map and

measured how many channels the layers should contain if all layers' feature maps remain 10×10. We then calculated the number of trainable parameters in the original network (excluding bias). Based on the number of trainable parameters and the new number of channels, we have calculated the new kernel resolutions as shown in Table 1.

Table 1. Construction of FatNet from ResNet-18.

Layer	Number of Weights	Feature Pixels	FatNet Layer
64×128, k = (3×3)	73,728	8192	64×82, k = (4×4)
128×128, k = (3×3)	147,456	8192	82×82, k = (5×5)
128×128, k = (3×3)	147,456	82,192	82×82, k = (5×5)
128×128, k = (3×3)	147,456	82,192	82×82, k = (5×5)
128×256, k = (3×3)	294,912	4096	82×41, k = (9×9)
256×256, k = (3×3)	589,824	4096	41×41, k = (19×19)
256×256, k = (3×3)	589,824	4096	41, 41, k = (19×19)
256×256, k = (3×3)	589,824	4096	41, 41, k = (19×19)
256×512, k = (3×3)	1,179,648	2048	41×21, k = (37×37)
512×512, k = (3×3)	2,359,296	2048	21×21, k = (73×73)
512×512, k = (3×3)	2,359,296	2048	21×21, k = (73×73)
512×512, k = (3×3)	2,359,296	2048	21×21, k = (73×73)
FC (512, 100)	51,200	100	21×1, k = (49×49)

Unfortunately, kernels larger than the input features in the last layer cause a problem. The main problem is that the convolutions are the same padding type, meaning that the input and output resolutions are the same 10×10 in our case. This means the outer regions of the kernels larger than 10×10 are redundant and will not be trained. This restricts us to the convolutions of the kernel with a maximum resolution of 10×10. Therefore, we reduced the kernel size by increasing the number of channels in those layers, which violates the third rule of the FatNet construction (see Figure 3). However, this is the better solution, since the network may underfit if the number of trainable parameters is reduced.

Usually, the image classifier neural networks are based on the convolutional layers for the feature extraction and dense layers for the classification. Sometimes, fully convolution networks end up with a convolutional layer with a 1×1 shape and the number of output channels equal to the number of classes. The FatNet's output layer is a convolutional layer with one channel and each pixel representing the probability of the class in the classification network. In our case with CIFAR-100 training, the output shape is 10×10 with one output channel. The main advantage of FatNet and its suitability for free-space optical training is that FatNet uses fewer output channels but larger resolution feature maps and kernels. Moreover, it is a fully convolutional network, which makes it fully compatible with the 4f accelerator.

As part of this work, we developed an application called FatSpitter to convert any sequential network into a FatNet model. FatSpitter accepts a PyTorch neural network object as an input and outputs a refined FatNet model that ensures that the size of the kernel never exceeds the input size of the corresponding convolutional layer. It follows an algorithm similar to that described above. Initially, a construction table, as demonstrated in Table 1, is established. Upon obtaining the construction table and determining the index of the layer at which FatNet has to start, the implementation begins by substituting original convolutional layers with the new "Fat" convolutional layer, which features altered kernel sizes and output channels. If the kernel size is larger than the input, the kernel size is adjusted to match the input, and the number of output channels is recalculated. It is important to remember that if the number of input channels in the original convolutional layer is equal to the output channels, this equality must also be maintained in FatNet. If this is not considered, the number of output channels will keep rising and falling in the network, making it impossible to train.

To validate our results, we developed a simulator as the custom layer on top of PyTorch called OptConv2d. OptConv2d replaces the convolution operation of the standard

convolutional layer with the simulation of 4f inference. In order to achieve this, we had to simulate the propagation of the amplitude-modulated light using the angular spectrum of plane waves (ASPW) method. According to the angular spectrum method, if the initial wavefront is $U_1(x, y)$, the next wavefront is calculated as:

$$U_2(x, y) = F^{-1}[F[U_1(x, y)]H(f_x, f_y)] \tag{2}$$

where $H(f_x, f_y)$ is the transmittance function for free space.

The transmittance function of free-space propagation comes from the Fresnel diffraction transfer function:

$$H_F(f_x, f_y) = \exp\left[jkz - j\pi\lambda z(f_x^2 + f_y^2)\right] \tag{3}$$

where $k = \frac{2\pi}{\lambda}$, z is the distance travelled by light, and λ is the wavelength [36,37].

Since the 4f system contains two lenses, the transmittance function of each lens is:

$$t_A(x, y) = P(x, y) \exp\left[-j\frac{k}{2f}(x^2 + y^2)\right] \tag{4}$$

where f is the focal length of the lens, and $P(x, y)$ is the pupil function [37].

The distance at which the angular spectrum method calculates the next wavefront depends on the pixel scale and is calculated as:

$$z = \frac{N(\Delta x)^2}{\lambda} \tag{5}$$

where Δx is the pixel scale, N is the number of pixels, and λ is the wavelength. In case when the propagation distance needs to be longer than the above formula for the distance, the propagation can be calculated in several iterations. We chose such a pixel scale for each propagation, so z becomes equal to the focal distance of the lens. In this case, we have to do only one iteration for each focal distance propagation in the 4f system.

The simulator uses pseudo-negativity, so each convolution is run twice to avoid negative values for the kernels in optics. Moreover, due to the laws of geometrical optics, the output of the 4f device is always rotated 180 degrees. Luckily, this is not a problem for convolutional neural networks, since they can continue extracting the future values from the rotated feature maps.

Experiments

The main goal of FatNet is not to gain accuracy but to demonstrate that the network with its prescribed architecture can maintain accuracy by being accelerated using free-space optics while performing fewer inferences through the 4f system than the original network. Hence, our experiments aimed at testing and comparing the original network and FatNet.

We recreated the modified version of ResNet-18, converted it to the FatNet, and trained both networks. To validate the accuracy of the FatNet in the optical device, we trained the network in the simulator. In the real 4f system, we would have taken advantage of the parallelism of the network by tiling the batches. However, batches were not tiled in the simulator, since the matrices are represented in PyTorch's tensor format. All operations were performed without unwrapping the tensor, and the Fourier transforms and multiplications were performed directly on the 4-dimensional tensors. We chose this approach since the simulator-based training of the network was much slower than the standard PyTorch network. Each epoch of the optical simulation of FatNet takes 67 min, while the epoch in the standard FatNet with Conv2d layer of PyTorch is 15 s only.

The wavelength of the laser was set to 532 nm (green), and convex lenses with a 5 mm diameter and focal distance of 10 mm were assumed. It should also be noted that we have not taken the device's quantization and noise into account and used type float32.

We split our training set into training and validation sets according to a 80–20% ratio, respectively, resulting in 40,000 for training and 10,000 images for validation. The dataset was normalized using the mean and standard derivation of the CIFAR-100 at all channels. Moreover, we have applied augmentation methods, including the horizontal flip and random crop with the padding of four. All networks were trained with the SGD optimizer, 0.9 for the momentum, and the starting learning rate of 0.01, updating every 50 steps by 0.2. The last layers of all networks were passed through the 20% dropout layer. We trained all the networks using 2× NVIDIA A100 40 GB GPUs.

ResNet-18 and FatNet were trained with a batch size of 64 (32 per GPU). However, the optical simulation of FatNet had to be trained with a batch size of 16 (8 per GPU) due to the high memory requirement of the simulator, as the optical simulation enhances the computational graph and number of gradients. Although we have not simulated the parallelism of the 4f system, to gain acceleration, the 4f system needs to take advantage of high resolution. FatNet's best acceleration can be achieved if batch tiling is performed. In order to use batch tiling, all the inputs of the same batch have to be tiled in one input block, and the kernel has to be padded to the same size as the input block. Before tiling the inputs, they must be individually padded to $M + N - 1$, where $M \times M$ is the input size, and $N \times N$ is the kernel size. According to this method, the number of possible batch sizes can be calculated as follows:

$$n = \lfloor \frac{R}{M + N - 1} \rfloor^2 \tag{6}$$

where R is the resolution of the 4f system and $\lfloor \cdot \rfloor$ is the floor function.

4. Results

Based on the configurations described above, our implementation of ResNet-18 achieved an accuracy of 66%. In comparison, FatNet's implementations, both with GPU and simulation of optics, lagged in accuracy with a result of 60% (see Table 2). However, FatNet implementation performs 8.2 times fewer convolution operations to reach this level of accuracy and does not require any dense layers for classification.

Table 2. Comparison of the test accuracy and number of convolution operations used in each tested network.

Architecture	Test Accuracy	Number of Conv Operations	Number of Conv Operations
	mean ± std		**Ratio to Baseline**
ResNet-18	66 ± 1.4%	1,220,800	1 (baseline)
FatNet	60 ± 1.4%	148,637	0.12
Optical simulation FatNet	60%	148,637	0.12

The same can be said about the training process. Since it may take more epochs for the FatNet to reach the desired accuracy, this architecture is only beneficial if accelerated with the 4f system.

The measured and calculated inference time for FatNet and ResNet-18 with optics and GPU were obtained and observed (see Table 3). The observations were conducted based on the batch size of 64, such as in our experiments, and 3136 maximum utilization of 4f system with 4k resolution modulators and camera.

Table 3. Inference time in seconds per input for ResNet-18 and FatNet with optics and GPU with batch sizes of 64 and 3136 for cases when the 4k resolution of the 4f device is fully utilized. The frame rate of the 4f device is approximated at 2 MHz [13].

Architecture	Batch 64	Batch 3136
ResNet-18 (GPU)	1.350×10^{-4}	1.167×10^{-4}
FatNet (GPU)	4.565×10^{-4}	7.942×10^{-4}
ResNet-18 (Optics)	3.815×10^{-2}	7.786×10^{-4}
FatNet (Optics)	4.645×10^{-3}	9.479×10^{-5}

5. Discussion

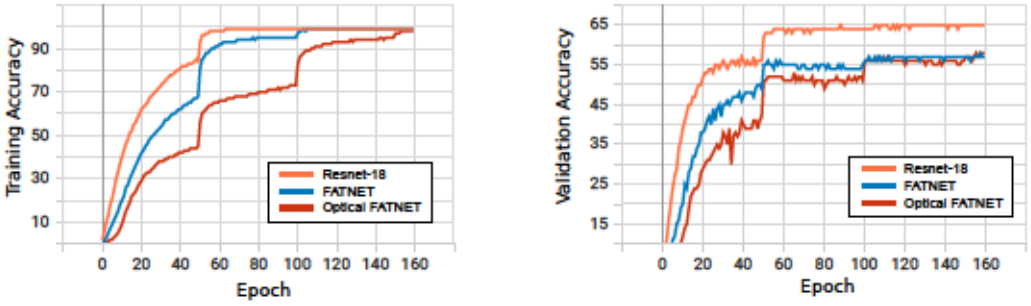
Although FatNet does not converge as well as ResNet-18, it is still 8.2 times faster, if both are trained with optics. CIFAR-100 is an extended dataset of CIFAR-10, but unlike CIFAR-10, CIFAR-100 is much harder to train. Numerous researchers have tried different augmentation and regularization methods to improve the classification performance of the CIFAR-100. For instance, Mizusawa [38] tried the interlayer regularization method and improved the accuracy of the classification of CIFAR-100 in ResNet-20 from an average of 64.09% to 65.59%. Shah [30] used ELU activation layers to improve the CIFAR-100 accuracy from 72.77% to 73.45%. Our modification of ResNet-18 achieved an average test accuracy of 66%, which is comparable to Mizusawa but lower than Shah. Then, our tests of FatNet showed that by sacrificing only 6% of test accuracy, we could perform 8.3 times fewer convolutions in optics, which will mean fewer conversions from optics to electronics and vice versa. During the training process of the original FatNet on GPU, three trials were conducted, achieving accuracies of 59%, 59%, and 62%. These results suggest that it may be possible to achieve a smaller loss with FatNet through further optimization efforts.

In contrast to our approach of reducing the number of convolution operations to improve speed and adapt the network to optical implementation, other research has focused on accelerating networks by accepting a small sacrifice in performance when run on CPU/GPU. For example, Luo et al. [39] accelerated the neural networks by discarding redundant weights. One of their implementations, ThiNet-Tiny, accelerated the forward and backward timing of VGG-16 by 6.4 and 7.3 times, respectively, at the cost of a 9% reduction in top-1 accuracy. Moreover, Rastegari et al. [5] achieved an acceleration of approximately 58 times by binarizing the inputs and weights of convolutional operations and estimating convolutions using XNOR and bit counting operations. However, when testing ImageNet trained on ResNet-18 with this method, they observed a loss of 18.1% in accuracy.

The training accuracy graph in Figure 4 shows that the network trained with the optical simulation trains slower than in other experiments. When simulating the 4f system, PyTorch uses the simulation of light propagation as part of the computation graph of the neural network, which vastly increases the computation graph. This causes a slowdown in network training. From the point of view of validation accuracy, the FatNet trained with GPU, and its optical simulation, are not altered much, especially after the first learning rate step on epoch 50. Although the validation accuracy of FatNet and optical simulation of FatNet did not exceed 57% and 58%, respectively, the test accuracy reached 60% in both cases. This difference is caused by the augmentation applied only to the validation and training sets and not to the test set.

However, it should be noted that the acceleration in a 4f system with FatNet is only possible if the parallelism of the 4f system is utilized not with the channel or kernel tiling but with batch tiling. The increase in resolution and reduction of the number of channels will not change the performance much if channel tiling is used. Unfortunately, due to the high latency of modern light modulators and cameras, it is almost impossible to get an acceptable acceleration over GPU with 4f, with the efficiency batch size shown in Table 3. However, the 4f system's acceleration is almost equalized to the GPU in comparison with non-GPU inference (see Table 3). If we fully utilize the 4K resolution of the 4f system,

the batch size of 3136 can be used, and the acceleration of the 4f system over GPU becomes obvious. Moreover, it can be seen that the use of FatNet improves the speed of the inference in optics and works in a completely opposite way with the GPU, regardless of the batch size. However, enormous batch sizes such as this are not efficient and will lead to overfitting.



(a) **(b)**

Figure 4. Training and validation accuracy for each experimented network at every epoch. **(a)** Training accuracy of ResNet-18, FatNet and Optical simulation of FatNet. All networks achieved an accuracy of 99%. However, the ResNet-18 required fewer epochs. On the other hand, the optical simulation took longer to train since it uses a more extended computation graph to simulate light propagation. **(b)** Validation accuracy of ResNet-18, FatNet and Optical simulation of FatNet. ResNet-18 trained up to 66%, while FatNet could not achieve the validation and test accuracy higher than 60%, although it performed fewer convolution operations.

Moreover, it should be mentioned that in our experiments, we have not tested the network with the different quantization levels and noise that can occur in the system. Low-precision training can potentially affect the test accuracy of the network, but there have been many successful attempts to train the neural networks with low precision to save on memory or accelerate the inference. On the other hand, noise can be used as a regularization method, since random and unpredictable noise can be a sort of augmentation method for our dataset. If we use a smaller bit depth, the noise may not affect the accuracy, since the changes in resulting light intensity will be low.

Another issue that is important to consider is the alignment of the optical elements. One of the main disadvantages of the 4f system compared with the silicon photonics approach is the alignment of optical elements. A slight alteration in the alignment of the elements of the 4f system can lead to entirely wrong results and to the inability to correctly keep track of the graph. Unfortunately, our simulator is not designed to consider alignment problems. In practice, optical cage systems can be used to keep elements fixed and aligned.

The FatSpitter algorithm, used to convert ResNet to FatNet, follows the rules we have established but only takes into account serial networks. Hence, it cannot be applied to networks that use depth-wise separable convolutions. It is possible to integrate branching into the algorithm for future implementation, but the most significant challenge remains the implementation of 1×1 convolutions. These convolutions can be executed in a 4f device as normal convolutions, but the question remains whether they need to be converted into fat layers. The main purpose of 1×1 convolutions is to reduce the dimensionality, so it makes sense not to convert them into FatNet and keep them as they are. On the other hand, converting 1×1 convolutions into FatNet would lead to a further reduction in the number of channels, which may not be necessary.

The design of the FatNet makes it more suitable for datasets with a large number of classes, such as 100 in our case, but it can also potentially work with images of a higher resolution. Unfortunately, the simulation of light propagation takes a large amount of GPU memory, which is the reason for not using ImageNet in our experiments when it seemed the most obvious choice for FatNet.

6. Conclusions

In this research, we looked at a new way of fully utilizing the high-resolution capabilities of the 4f system for classification. We introduced a transformation method, which makes the regular neural network designed for the CPU/GPU training more compatible with the free-space optical device. After testing FatNet with the CIFAR-100 dataset, using ResNet-18 as the backbone network and the optical simulation of the FatNet using the angular spectrum method, we reached a test accuracy of 66% with ResNet and 60% with FatNet. Eventually, it was demonstrated that FatNet performs 8.2 times fewer convolution operations than ResNet-18 without a loss in frame rate when both were implemented in optics. Compared with the standard ResNet-18, FatNet is always faster than ResNet-18 when run with the optical device and also than ResNet-18 run with GPU when the batch size is as large as 3136. Moreover, our research demonstrates the importance of using high-resolution kernels in CNN, especially in the future, when the speed of cameras and light modulators improves.

Author Contributions: Conceptualization, R.I.; methodology, R.I.; software, R.I.; validation, R.I., T.M.C. and C.C.R.-A.; formal analysis, R.I.; investigation, R.I.; resources, R.I.; writing—original draft preparation, R.I.; writing—review and editing, C.C.R.-A. and T.M.C.; visualization, R.I.; supervision, T.M.C. and C.C.R.-A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Simulator and the models repository: https://github.com/riadibadulla/simulator; FatSpitter: https://github.com/riadibadulla/FatSpitter; CIFAR-100 dataset: https://www.cs.toronto.edu/~kriz/cifar.html, accessed on: 1 December 2022.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CIFAR	Canadian Institute For Advanced Research
CNN	Convolutional Neural Network
ASIC	Application-Specific Integrated Circuit
ELU	Exponential Linear Unit
SGD	Stochastic Gradient Descent
FFT	Fast Fourier Transfer
TPU	Tensor Processing Unit
MZI	Mach–Zehnder Interferometer
SLM	Spatial Light Modulators

References

1. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
2. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. *arXiv* **2016**, arXiv: 1506.02640.
3. Tompson, J.; Goroshin, R.; Jain, A.; LeCun, Y.; Bregler, C. Efficient Object Localization Using Convolutional Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015 ; pp. 648–656.
4. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv* **2015**, arXiv:1505.04597.
5. Rastegari, M.; Ordonez, V.; Redmon, J.; Farhadi, A. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. In *Proceedings of the Computer Vision—ECCV 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 525–542.
6. Sunny, F.P.; Taheri, E.; Nikdast, M.; Pasricha, S. A Survey on Silicon Photonics for Deep Learning. *ACM J. Emerg. Technol. Comput. Syst.* **2021**, *17*, 1–57. [CrossRef]

7. Jouppi, N.P.; Young, C.; Patil, N.; Patterson, D.; Agrawal, G.; Bajwa, R.; Bates, S.; Bhatia, S.; Boden, N.; Borchers, A.; et al. In-Datacenter Performance Analysis of a Tensor Processing Unit. *arXiv* **2017**, arXiv:1704.04760.
8. Davies, M.; Srinivasa, N.; Lin, T.H.; Chinya, G.; Cao, Y.; Choday, S.H.; Dimou, G.; Joshi, P.; Imam, N.; Jain, S.; et al. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. *IEEE Micro* **2018**, *38*, 82–99. [CrossRef]
9. DeBole, M.V.; Taba, B.; Amir, A.; Akopyan, F.; Andreopoulos, A.; Risk, W.P.; Kusnitz, J.; Ortega Otero, C.; Nayak, T.K.; Appuswamy, R.; et al. TrueNorth: Accelerating From Zero to 64 Million Neurons in 10 Years. *Computer* **2019**, *52*, 20–29. [CrossRef]
10. Waldrop, M.M. The chips are down for Moore's law. *Nat. News* **2016**, *530*, 144. [CrossRef]
11. Li, X.; Shao, Z.; Zhu, M.; Yang, J. *Fundamentals of Optical Computing Technology: Forward the Next Generation Supercomputer*, 1st ed.; Springer: New York, NY, USA, 2018.
12. Lin, X.; Rivenson, Y.; Yardimci, N.T.; Veli, M.; Luo, Y.; Jarrahi, M.; Ozcan, A. All-optical machine learning using diffractive deep neural networks. *Science* **2018**, *361*, 1004–1008. [CrossRef]
13. Li, S.; Miscuglio, M.; Sorger, V.; Gupta, P. Channel Tiling for Improved Performance and Accuracy of Optical Neural Network Accelerators. *arXiv* **2020**, arXiv:2011.07391 .
14. Chang, J.; Sitzmann, V.; Dun, X.; Heidrich, W.; Wetzstein, G. Hybrid optical-electronic convolutional neural networks with optimized diffractive optics for image classification. *Sci. Rep.* **2018**, *8*, 12324. [CrossRef] [PubMed]
15. Shen, Y.; Harris, N.C.; Skirlo, S.; Prabhu, M.; Baehr-Jones, T.; Hochberg, M.; Sun, X.; Zhao, S.; Larochelle, H.; Englund, D.; et al. Deep learning with coherent nanophotonic circuits. *Nat. Photonics* **2017**, *11*, 441–446. [CrossRef]
16. Hughes, T.W.; Minkov, M.; Shi, Y.; Fan, S. Training of photonic neural networks through in situ backpropagation and gradient measurement. *Optica* **2018**, *5*, 864–871. [CrossRef]
17. Sui, X.; Wu, Q.; Liu, J.; Chen, Q.; Gu, G. A Review of Optical Neural Networks. *IEEE Access* **2020**, *8*, 70773–70783. [CrossRef]
18. Bracewell, R.N. *The Fourier Transform and Its Applications*, 3rd ed.; McGraw-Hill Series in Electrical and Computer Engineering Circuits and Systems; McGraw-Hill: Boston, MA, USA, 2000.
19. Gaskill, J.D. *Linear Systems, Fourier Transforms, and Optics*, 1st ed.; Wiley-Interscience: New York, NY, USA, 1978.
20. Cooley, J.W.; Tukey, J.W. An Algorithm for the Machine Calculation of Complex Fourier Series. *Math. Comput.* **1965**, *19*, 297–301. [CrossRef]
21. Colburn, S.; Chu, Y.; Shilzerman, E.; Majumdar, A. Optical frontend for a convolutional neural network. *Appl. Opt.* **2019**, *58*, 3179–3186. [CrossRef] [PubMed]
22. Jutamulia, S.; Asakura, T. Fourier transform property of lens based on geometrical optics. In Proceedings of the Optical Information Processing Technology, Shanghai, China, 14–18 October 2002; Volume 4929, pp. 80–85. [CrossRef]
23. Culshaw, B. The Fourier Transform Properties of Lenses. In *Introducing Photonics*; Cambridge University Press: Cambridge, UK, 2020; pp. 132–135. [CrossRef]
24. Weaver, C.S.; Goodman, J.W. A Technique for Optically Convolving Two Functions. *Appl. Opt.* **1966**, *5*, 1248–1249. [CrossRef] [PubMed]
25. Jutamulia, S.; Yu, F.T.S. Overview of hybrid optical neural networks. *Opt. Laser Technol.* **1996**, *28*, 59–72. [CrossRef]
26. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
27. Gron, A. *Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 1st ed.; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2017.
28. Peng, C.; Zhang, X.; Yu, G.; Luo, G.; Sun, J. Large Kernel Matters—Improve Semantic Segmentation by Global Convolutional Network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4353–4361.
29. Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. Technical Report, 2009, University of Toronto, Toronto. Available online: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf (accessed on 23 March 2023) .
30. Shah, A.; Kadam, E.; Shah, H.; Shinde, S.; Shingade, S. Deep Residual Networks with Exponential Linear Unit. In Proceedings of the Third International Symposium on Computer Vision and the Internet, Jaipur, India, 21–24 September 2016; pp. 59–65. [CrossRef]
31. Clevert, D.A.; Unterthiner, T.; Hochreiter, S. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *arXiv* **2016**, arXiv:1511.07289.
32. Tan, M.; Le, Q.V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv* **2020**, arXiv:1905.11946.
33. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255. [CrossRef]
34. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proceedings of the Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Vancouver, BC, Canada, 10–12 December 2019; Volume 32.
35. Miscuglio, M.; Hu, Z.; Li, S.; George, J.K.; Capanna, R.; Dalir, H.; Bardet, P.M.; Gupta, P.; Sorger, V.J. Massively parallel amplitude-only Fourier neural network. *Optica* **2020**, *7*, 1812–1819. [CrossRef]
36. Li, J.; Peng, Z.; Fu, Y. Diffraction transfer function and its calculation of classic diffraction formula. *Opt. Commun.* **2007**, *280*, 243–248. [CrossRef]
37. Voelz, D.G. *Computational Fourier Optics: A MATLAB® Tutorial*; SPIE: Bellingham, WA, USA, 2011. [CrossRef]

38. Mizusawa, S.; Sei, Y. Interlayer Augmentation in a Classification Task. In Proceedings of the 2021 International Conference on Computing, Electronics & Communications Engineering (iCCECE), Southend, UK, 16–17 August 2021; pp. 59–64. [CrossRef]
39. Luo, J.H.; Wu, J.; Lin, W. ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 5058–5066.

Fat-U-Net: Non-Contracting U-Net for Free-Space Optical Neural Networks

Riad Ibadulla[a], Constantino C. Reyes-Aldasoro[a], and Thomas M. Chen[a]

[a]City, University of London, Northampton Square, London, UK, EC1V 0HB

ABSTRACT

This paper describes the advantages and disadvantages of adapting the U-Net architecture from a traditional GPU to a 4f free-space optical environment. The implementation is based on an optical-based acceleration called FatNet and thus this adaption is called Fat-U-Net. Fat-U-Net neglects the pooling operations in U-Net, but maintains a similar number of weights and pixels per layer as U-Net. Our results demonstrate that the conversion to Fat-U-Net offers significant improvement in speed for segmentation tasks, with Fat-U-Net achieving a remarkable $\times 538$ acceleration in inference compared to U-Net when both are run on optical devices and x37 acceleration in inference compared to the results provided by U-Net on GPU. The performance loss after conversion remains minimal in two datasets, with reductions of 4.24% in IoU for the Oxford IIIt pet dataset and 1.76% in IoU of HeLa cells nucleus segmentation.

Keywords: FatNet, HeLa segmentation, Optical Neural Network, segmentation

1. INTRODUCTION

The introduction of deep learning in computer vision applications has completely changed how digital images are processed and analysed. The application of Deep learning approaches to image segmentation has demonstrated remarkable results.[1–3] However, as the complexity of these machine learning models grows, so does the computational demand and the difficulty of real-time applications. While hardware accelerators, such as graphics processing units (GPUs), field-programmable gate arrays (FPGAs), and application-specific integrated circuits (ASICs), have emerged as a potential solution to this challenge, their effectiveness may be limited in the long run as Moore's Law begins to lose its predictive power.[4]

Advances in optical computing have shown the potential of optical accelerators to complement electronics-based hardware accelerators. Since optical computing is unaffected by Moore's law, it can be used for deep learning through optical accelerators, offering advantages such as higher bandwidth, faster processing, no resistance, reduced power consumption and immunity to thermal disturbances.[5] Two primary methods exist for building optical neural networks: the free space approach employing spatial light modulators (SLMs), and the silicon photonics method which utilises Mach–Zehnder interferometers (MZIs). The free space approach relies on light travelling through mediums like air, outer space, or a vacuum, unlike silicon photonics which relies on guided light paths. While the silicon photonics technique offers higher speed, with potential clock speeds of several GHz, it lags behind the free-space method in terms of parallel processing capabilities.[6] Free-space optical accelerators provide massive parallelism capabilities, and 4f optical accelerators can perform convolution operations faster than the traditional electronic processor with theoretical infinite resolution.[7] In practice, they are limited by the resolution of the modulators and the speed of the cameras used. In this research, we focus on the 4f free-space approach as described in Li et al.[8] in order to accelerate the inference and training of convolutional neural networks (CNNs) for image segmentation.

One of the main tasks in computer vision, semantic segmentation aims to partition an image into meaningful segments by assigning a class to each pixel. According to Peng *et al.*,[9] semantic segmentation is considered a pixel-wise classification problem, and a well-designed segmentation model should simultaneously encompass two tasks, classification and localisation. It was observed that these tasks are naturally "contradictory", as improving

Further author information: (Send correspondence to R.I.)
R.I.: E-mail: riad.ibadulla@city.ac.uk

AI and Optical Data Sciences V, edited by Ken-ichi Kitayama, Volker J. Sorger, Proc. of SPIE
Vol. 12903, 1290308 · © 2024 SPIE · 0277-786X · doi: 10.1117/12.3008618

one tends to diminish the other. This is because the classification model is insensitive to translation or rotation, while the localisation task should contain information regarding the appropriate coordinates in the output mask. For that reason, the classification models consist of pooling operations to extract the features at every scale. When having small 3x3 kernels on the deepest layers of the classification network, the kernel size - feature map resolution ratio is larger than on the shallow layers. Hence the features of the deeper layers can be affected by more pixels of the original image. This is why classification networks are mostly pyramid- or cone-shaped. Ideally, a barrel-shaped segmentation model would locate pixels of each class more precisely.

Although Peng *et al.*[9] proposed their own architecture, Global CNN, the well-established U-Net[1] can also address both problems simultaneously, where the contracting path solves the classification task, expanding path and skip connections support the localisation. However, it can be seen that most of the segmentation networks,[2,10] including U-Net, use an encoder-decoder structure or contain some piece of architecture for pulling the feature maps down in order to extract the features.[11] These networks simply inherit the successful structure of predecessor networks like AlexNet[12] or LeNet,[13] which are meant to do the classification task. It is important to keep in mind that higher-resolution feature maps and kernels are more suitable for segmentation tasks since high-resolution kernels have a higher effective receptive field than de-facto standard 3x3 kernels.

One of the key advantages of employing small kernels and cone encoder-decoder architectures is the speed of inference on CPU/GPU-based hardware. On the other hand, this acceleration in optical environments can be provided intrinsically by the optics. In previous work, we proposed the FatNet[14]conversion for classifier CNNs which reduced the number of channels and increased the kernel size and resolution of the feature map accordingly by keeping same number of parameters and pixels in each feature maps of the network. This conversion makes the network more suitable for 4f free-space optical acceleration.

In order to understand the reason behind it, it is worth looking at the principle of work in the 4f optical neural network accelerator. The 4f setup consists of an input laser, two convex lenses, and modulators. The idea is based on the Fourier transform properties of the convex lenses and performs the convolution operation based on the convolution theorem. Any convex lens projects a Fourier transform of the input object located on the front focal plane onto the back focal plane.[15] At this point, it can be pointwise multiplied by the kernel in the Fourier domain. After passing through the second lens, the multiplied output is converted back into the spatial domain and read by the camera. The process described above can perform the convolution operation using the 4f system, but in order to be able to apply the method to the convolutional neural network, it is essential to read the output of the 4f system, apply the activation function electronically and repeat the process. This causes the main bottleneck of the optical acceleration. Moreover, it is important to note that the resolution of the input and the kernel will not affect the system's frame rate. Hence, in order to maximise the utilisation of the system, the number of conversions to electronics should be reduced, but the resolution should be used to our advantage.

One of the obvious ways to utilise the high-resolution capabilities of the 4f system is to tile the inputs and kernels and perform convolutions in parallel, in other words, perform the batch tiling. According to Li *et al.*,[16] the high-resolution capabilities of 4f system can also be used to tile the channels and kernels, to perform several 2D convolution operations of one convolutional layer simultaneously. However, the FatNet algorithm ensures faster training in the 4f optical accelerator by reducing a number of channels and increasing the resolutions of the feature maps and kernels of CNNs, while relying on batch tiling. As the resolution is not an obstacle for the 4f optical accelerator, while fewer convolution operations mean fewer optics-electronics conversions. However, it can be assumed the FatNet conversion is even more suitable for segmentation tasks, which we have proposed in our work, and developed a Fat-U-Net, which is described in section 2.4.

Our previous work[14] was based on the conversion of the ResNet-18 into the FatNet. In this work, we demonstrate the possibility of expanding the FatNet further for segmentation tasks with U-Net, turning it into a Fat-U-Net. Notably, Fat-U-Net achieves a theoretical $\times 538$ faster inference when run on optical devices and $\times 37$ acceleration in inference compared to the results provided by U-Net on GPU.

Moreover, this work demonstrates the validity of the FatNet conversion algorithm. We trained other networks, called Intuitive Fat-U-Nets, with fewer channels and larger kernels, which did not adhere to the FatNet conversion principles. These networks were converted from U-Net based on the number of weights, without considering the number of pixels in each feature map. Despite this, none could outperform the original Fat-U-Net in terms of performance.

The performances of U-Net and Fat-U-Net implementations are compared using the Oxford IIIt pet dataset and HeLa cells dataset.

2. MATERIALS AND METHODS

2.1 Oxford-IIIT Pet

The first dataset analysed was the Oxford-IIIT pet dataset[17] developed by the Visual Geometry Group, consisting of 7,359 images and 37 pet categories, each containing approximately 200 images. These images exhibit significant differences in scale, pose, and lighting conditions. Each image is accompanied by ground truth annotations, including breed identification, head region of interest (ROI), and pixel-level trimap segmentation. Since our network is focused on segmentation, we are not taking the classes into account but focusing on the segmentation of the pets and backgrounds. The dataset provides a train-test split, where 3,680 images are designated for training and 3,669 for testing. Because the images are of different sizes, the images are resized to 160x160 in this work. The intensity of the channels of the dataset is normalised between 0 and 1, and the centring is performed with the mean of (0.485, 0.456, 0.406) and standard deviation of (0.229, 0.224, 0.225) for each RGB channel, respectively.

2.2 HeLa Cells

The second case analysed a high-resolution dataset of HeLa cells observed with Serial Block Face Scanning Electron Microscopy. It consisted of $8192 \times 8192 \times 518$ voxels[18] from which a $2000 \times 2000 \times 300$ region of interest (ROI) with a single cell has been cropped.[19] The hand-segmented ground truth (GT) with four classes (background, cell, nuclear envelope, and nucleus) is publicly available only for the ROI[20] and GT can be generated with image-processing algorithms .[21] In this work, we focus only on the segmentation of the nucleus.

Since our version of Fat-U-Net was designed for 160x160 images, we have prepared patches of 160x160 from odd-numbered slices of the ROI with 50% overlap. Taking only half of the slices and accounting for the 529 patches within each slice, we generated 79,350 image pairs along with their corresponding ground truth masks. Before saving the dataset of patches, all patches were low-passed filtered with a Gaussian filter. For that reason, we perform the same Gaussian filtering every time when evaluating new data.

Performing the data split among the shuffled patches could potentially result in a training or test set biased towards a specific class due to the inclusion of an excessive number of background images. Therefore, we performed a train-test split on a per-slice basis. Slices $(1, 11, 21, \ldots, 281, 291)$ were set as test slices, and slices $(5, 25, 45, \ldots, 285)$ were set as validation slices. Originally the rest of the data was used for training. However, as the shallowest and deepest slices contain only background, this leads to data imbalance and the training slices were set only to the slices where the cell and the nucleus are fully visible in the middle of the ROI. With this strategy, the training slices were defined within the range of 97 to 183 with step 2, excluding slices ending with 1 and 5, which resulted in 26 slices and 13,754 patches, such as (97, 99, 103, 107, 109,..., 177, 179). Although the number of training patches may appear limited, it is sufficient for binary nucleus segmentation. In contrast to Karabag *et al.*,[21] we ensured that our model evaluation did not include any slices that were part of the training process.

No data augmentation was applied to the dataset in this study. However, normalisation was performed to scale the data values between 0 and 1. Furthermore, centring was conducted using the calculated mean and standard deviation values, which were determined to be 0.6379 and 0.0855, respectively.

2.3 U-Net

U-Net[1] is a CNN architecture initially developed for the segmentation of biomedical images. Its unique architecture, consisting of contracting and expanding paths, allows it to capture local and contextual information effectively, leading to impressive segmentation results. The contracting path of the network can be seen as the typical CNN used for classification. It consists of blocks of convolutional layers, activation functions and pooling operations for feature extraction at different scales.

The expanding path of the U-Net serves for the upsampling of the extracted features to reconstruct the segmentation mask of the input image. This is achieved using transposed convolution operations to upsample the

feature maps and concatenation with the corresponding feature maps of the same resolution from the contracting path. The main role of the skip connections is to conserve the spatial information that is lost during the pooling process in the contracting path. Our implementation of U-Net is shown in Figure 1(a). It contains five stages, and unlike the original implementation of U-Net by Ronnenberg *et al.*,[1] it does not require cropping of the feature maps when performing skip concatenation, as it only uses convolutions with the "same padding".

2.4 Fat-U-Net

The idea of *Fat* layers, i.e., layers where there is no reduction in size, was introduced in[14] for the conversion of the CNN for classification into a form which is more compatible with 4f free-space optical accelerators. The underlying principle of FatNet conversion is to maintain the constant number of trainable parameters and the pixels in each layer while increasing the resolution of feature maps and kernels and decreasing the number of channels in each layer. By making this conversion, the network takes full advantage of the high-resolution capabilities of the 4f system, thereby optimising its performance and efficiency in the context of free-space optical acceleration. Since the main bottleneck of the free-space 4f accelerator is the latency of the camera, the fewer convolution operations that the networks have, the fewer optic-electronics conversions are required. Eventually, the cone-shaped classifier convolutional networks turn into barrel-shaped networks with higher-resolution feature maps and high-resolution kernels, which sometimes reach the size of the feature maps making it a "Fat" Layer.

The original FatNet conversion, designed specifically for the classification task, maintains the same architecture as the original network until the feature maps are pooled down to the resolution with a number of pixels less than or equal to the number of classes. It is posited that when it comes to the FatNet conversion for the segmentation, pooling may be unnecessary, and the input resolution can be preserved throughout the entire network. Consequently, increasing the resolution of kernels while keeping the resolution of the feature maps constant would decrease the feature map-to-kernel resolution ratio, emulating the effect of pooling the feature maps without actual pooling implementation. This approach can significantly increase the inference time of the network run on the 4f free-space optical accelerator and hypothetically retains localisation accuracy even more effectively.

U-Net contracting		weights	pixels	New layers		FatU-Net adjusted	
Channels	kernel			Channels	kernel	Channels	kernel
3×64	3	1,728	1,638,400	3×64	3	3×32	5
64×64	3	36,864	409,600	32×32	6	32×32	6
64×128	3	73,728	819,200	32×32	9	32×16	12
128×128	3	147,456	204,800	16×16	24	16×16	24
128×256	3	294,912	409,600	16×16	34	16×8	48
256×256	3	589,824	102,400	8×8	96	8×8	96
256×512	3	1,179,648	204,800	8×8	136	8×10	122
512×512	3	2,359,296	51,200	10×10	160	10×10	160
512×1024	3	4,718,592	102,400	10×18	160	10×20	154
1024×1024	3	9,437,184	102,400	20×20	160	20×20	160

Table 1. Construction table for Fat-U-Net's first half out of the U-Net's contracting path.

Since the original FatNet was designed for classification, only the contracting path of the U-Net was converted into the FatNet. Table 1 presents the Fat-U-Net equivalent of the FatNet construction table, as described in.[14] The table is used to compute the number of weights per layer, excluding the bias and the number of pixels per layer. The algorithm ensures the convolutional layers with the same number of input and output channels within convolutional blocks have an equal number of input and output channels after the conversion too. Upon completing the conversion of the contracting path of the U-Net into FatNet, the path was mirrored to generate the "expanding path", and the kernel sizes were recalculated to match the number of weights from the original layers. Since the so-called expanding path of the Fat-U-Net does not actually require upsampling, we have replaced the deconvolution operations with the simple 3x3 convolutions as illustrated in Figure 1(b).

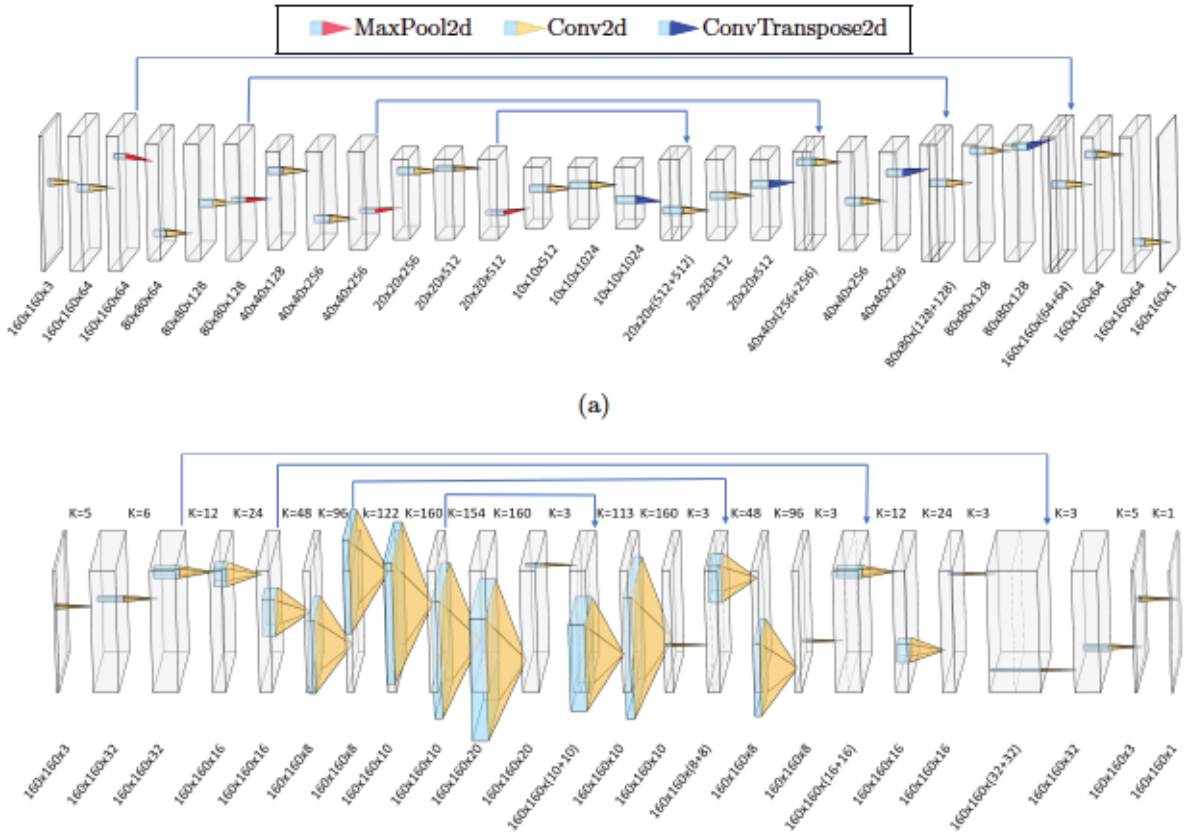


Figure 1. Graphical representation of our implementation of U-Net and Fat-U-Net architectures. (a) U-Net architecture, with all kernel sizes 3x3, MaxPool with kernels size of 2x2 and deconvolution operations with a kernel size of 3x3. (b) Fat-U-Net architecture derived from our implementation of U-Net, with the varying kernel sizes indicated as K at each layer.

3. EXPERIMENTS AND RESULTS

U-Net and the Fat-U-Net equivalent were implemented and tested in two segmentation tasks of the Oxford IIIt pet and HeLa cells. Performance was assessed by pixel-wise Accuracy, Intersection over Union (IoU), and Dice Score:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$IoU = \frac{TP}{TP + FP + FN} \tag{2}$$

$$DiceScore = \frac{2TP}{2TP + FP + FN} \tag{3}$$

Inference time was also measured on GPU and theoretically calculated for the optics to demonstrate the acceleration on a 4f free-space optical device. Overall if our U-Net implementation contains 3,833,984 convolution operations, its Fat-U-Net equivalent contains only 7,123. Since the resolution does not affect the speed of inference

in the 4f free-space optics, the inference of Fat-U-Net in optics will be **538 times faster** than U-Net if both run in optics. This acceleration is possible with only a small sacrifice in performance, as seen in Tables 3, 4.

We have measured the inference time of Nvidia A100 with both U-Net and Fat-U-Net, and compared the results to the calculated theoretical inference time on 4f optical accelerator based on Li *et al.*[16] The results are shown in Table 2 for batch sizes of 1, 32, and 144. The batch size of 144 was chosen because it is the maximum possible batch size for the 4f system with 4k resolution, if batch tiling is applied.

Based on the results in Table 2, at the batch size of 144, the acceleration of inference of Fat-U-Net with 4f optics, compared to U-Net run on high-end GPU, is **37 times**.

Model and device	Batch 1	Batch 32	Batch 144
U-Net (Optics)	1920.00	59.900	13.300
Fat-U-Net (Optics)	3.46	0.108	**0.024**
U-Net (GPU)	4.55	0.894	0.883

Table 2. Inference time in milliseconds of U-Net and its Fat-U-Net equivalent model per image with different batch sizes run on 4f accelerator and Nvidia A100. The frame rate for 4f system was approximated at 2 MHz, and Nvidia A100 GPU was measured experimentally.

3.1 Oxford IIIt pet

Training of the Oxford IIIt pet dataset used the Adam optimiser; the learning rate was set to 1e-4 with a batch size of 16 and a number of epochs of 250. The training data went through augmentation during training, by random shift, scale, rotation, RGB shift, random brightness and contrast. We have used the BCEWithLogitsLoss of PyTorch, which combines Binary Cross-Entropy loss with the sigmoid layer. We have ensured that our U-Net results adhered to state-of-the-art standards before converting them into the Fat-U-Net and conducting the comparison of evaluation metrics between Fat-U-Net, its backbone U-Net, and previous research employing the Oxford IIIt pet dataset as a benchmark (Table 3).

Model	**Accuracy (%)**	**IoU (%)**	**Dice Score (%)**
U-Net (our implementation)	**95.33**	89.32	**94.33**
Fat-U-Net (ours)	93.40	85.08	91.87
SEU-Net[22]	-	$\approx$ 77.00	-
ICNet[23,24]	90.79	75.12	-
ConRec (20% of dataset) [25,26]	-	-	90.00
U-Net (as per Sundarrajan *et al.*)[27]	-	33.30	46.40
U-Net+VGG16[27]	-	89.40	94.20
U-Net+InceptionV3[27]	-	**91.60**	91.50

Table 3. Comparison of the evaluation results of the accuracy, mIoU, and Dice score of U-Net and its Fat-U-Net equivalent along with other works for Oxford IIIt pet.

We have visualised the predicted mask on the data for both U-Net and Fat-U-Net in Figure 2, to understand where the segmentation is excellent, where it is unacceptable, and where Fat-U-Net outperforms U-Net or vice versa.

3.2 HeLa cells

The Adam optimiser was used to train HeLa nucleus segmentation as well, with the inclusion of a weight decay set at 1e-4. We have applied two dropout layers with a probability of 50% to the beginning and end of the bridge section of U-Nets. The learning rate was set to 1e-3, with a batch size of 32 and a number of epochs of 20. The loss function remained the same, BCEWithLogitsLoss, which combines binary cross-entropy loss and the sigmoid layer.

Both U-Net and Fat-U-Net models were evaluated with four scenarios: (1) the complete set of odd slices ranging from 1 to 300, (2) the middle-range of odd slices (150-200) where the nucleus is visible, and (3)-(4) then

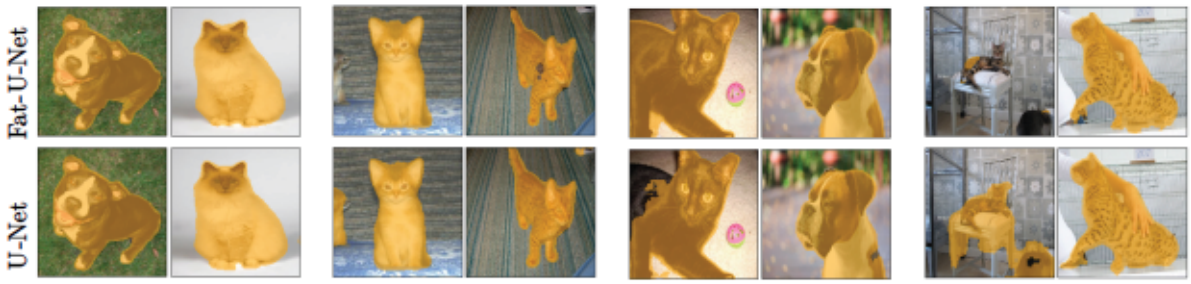


Figure 2. Qualitative results of Oxford IIIt pet dataset. (a) Examples of perfect segmentation by both algorithms. (b) Examples of U-Net performing better than Fat-U-Net. (c) Examples of Fat-U-Net outperforming U-Net. (d) Bad segmentation examples by both algorithms

Model	**Acc.(all) (%)**	**IoU(all) (%)**	**Acc.(150-200) (%)**	**IoU(150-200) (%)**
U-Net	95.71	66.32	99.59	**97.15**
U-Net (Test data)	**95.75**	**66.59**	99.57	97.11
Fat-U-Net	95.31	64.27	99.42	96.05
Fat-U-Net (Test data)	95.42	64.83	99.43	96.25
4 stage U-Net[21]	93.46	51.38	**99.66**	97.12

Table 4. Performance comparison of our implementation of five staged U-Net, its Fat-U-Net equivalent, and a four staged U-Net implemented in.[21] Evaluating Accuracy and IoU Metrics Across the entire dataset and 150-200 range for all odd and test slices that have not participated in the training process.

repeating the same strategies for the test slices (See Table 4). The results of the first two scenarios, which include both training and validation slices, were comparable to the results of the work of Karabag *et al.* [21]

Since GT was only available for the ROI, which is one cell of the larger $8192 \times 8192 \times 518$ datasets, qualitative tests were performed training on one cell and testing in an adjacent cell as demonstrated in Figure 3. Moreover, the qualitative tests were also performed on the segmentation of the larger original image of 8192×8192 containing all the cells (Figure 4).

All qualitative evaluation was performed for both U-Net and Fat-U-Net, for comparison purposes.

3.3 Validity of FatNet

To demonstrate the efficacy of the FatNet conversion, we have trained alternate networks with fewer channels and larger kernels. These networks, which we call Intuitive Fat-U-Nets, deviate from the FatNet conversion formula by focusing only on the number of weights and not considering the pixel count in each feature map. Three versions of Intuitive Fat-U-Net were designed and shown in Table 5.

Among all networks, Intuitive Fat-U-Net 1 is the closest to the original U-Net as the channels in the bottleneck rise up to 128, with the largest kernel size being 24. However, even Intuitive Fat-U-Net 1 performed worse than the Fat-U-Net as it can be seen in Table 6. While the Intuitive Fat-U-Net 3, the closest network to the Fat-U-Net with the largest kernel size, 153, performed the worst of all networks.

4. DISCUSSION

In this study, we successfully demonstrated that the FatNet conversion of in silico networks to optical devices is more efficient for segmentation tasks than for classification. For comparison, Ibadulla *et al.*[14] reported an acceleration of 8.2 times for the ResNet-18, if ResNet-18 and FatNet run on the optical device. This work shows a remarkable 538 faster inference of Fat-U-Net compared to the U-Net under the same conditions and $\times 37$ acceleration in inference compared to the results provided by U-Net on GPU. Moreover, from Table 2 it can be seen that the GPU Nvidia A100, being one of the best hardware accelerators, outperforms optical accelerator of

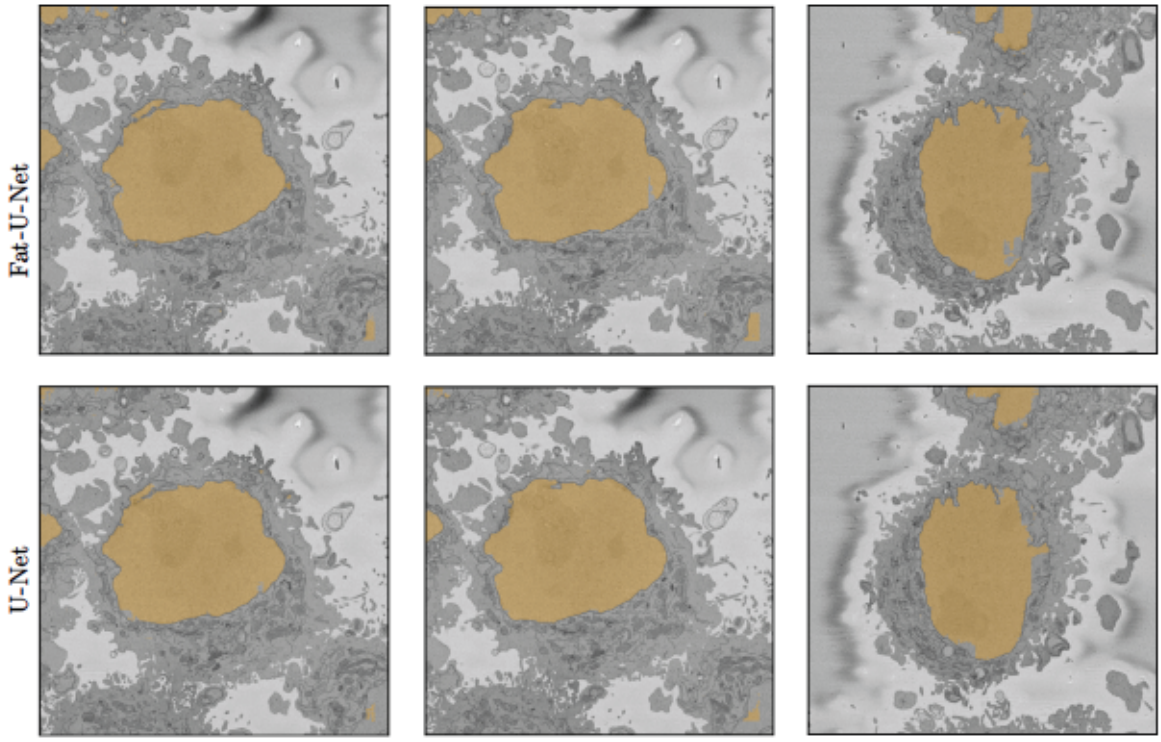


Figure 3. Qualitative results of HeLa dataset. (a) train slice 119 (b) test slice 121 (c) Unseen cell, taken from the larger field.

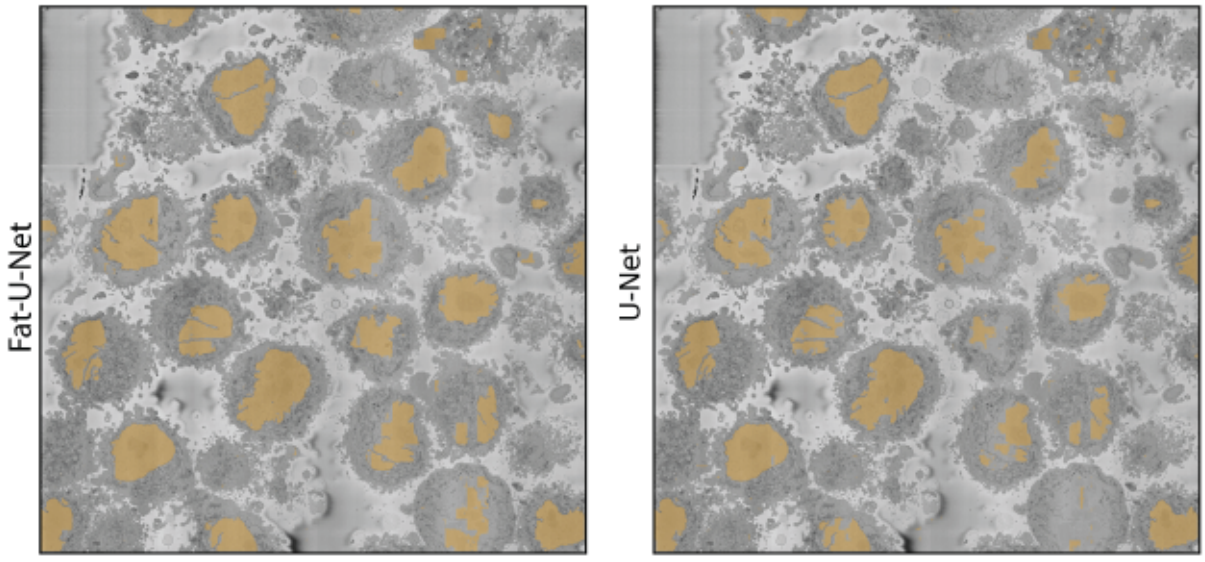


Figure 4. U-Net and Fat-U-Net segmentation results on 8192×8192 images.

2 MHz frame rate when running U-Net, but stays slow for all batch sizes when compared to Fat-U-Net run on optical accelerator. Given that the 4f optical device is meant to accelerate only the convolution operations, it is intuitive that fully convolutional networks like U-Net are ideally suited for the 4f accelerators, as they do not even need any amendments of the dense layers, as required in the classification.

With the speed advantages of Fat-U-Net established, our next objective was to validate its performance. We

Layer	Intuitive Fat-U-Net 1		Intuitive Fat-U-Net 2		Intuitive Fat-U-Net 3	
	Channels	Kernel	Channels	Kernel	Channels	Kernel
Conv block 1	$3 \rightarrow 8$	8	$3 \rightarrow 4$	12	$3 \rightarrow 4$	12
	$8 \rightarrow 8$	24	$4 \rightarrow 4$	48	$4 \rightarrow 4$	48
Conv block 2	$8 \rightarrow 16$	24	$4 \rightarrow 8$	48	$4 \rightarrow 8$	48
	$16 \rightarrow 16$	24	$8 \rightarrow 8$	48	$8 \rightarrow 8$	48
Conv block 3	$16 \rightarrow 32$	24	$8 \rightarrow 16$	48	$8 \rightarrow 10$	61
	$32 \rightarrow 32$	24	$16 \rightarrow 16$	48	$10 \rightarrow 10$	77
Conv block 4	$32 \rightarrow 64$	24	$16 \rightarrow 32$	48	$10 \rightarrow 16$	85
	$64 \rightarrow 64$	24	$32 \rightarrow 32$	48	$16 \rightarrow 16$	96
Conv block 5	$64 \rightarrow 128$	24	$32 \rightarrow 64$	48	$16 \rightarrow 20$	121
(bottleneck)	$128 \rightarrow 128$	24	$64 \rightarrow 64$	48	$20 \rightarrow 20$	153
DeConv 1	$128 \rightarrow 64$	3	$64 \rightarrow 32$	3	$20 \rightarrow 16$	3
Conv block 6	$128 \rightarrow 64$	24	$64 \rightarrow 32$	48	$32 \rightarrow 16$	96
	$64 \rightarrow 64$	24	$32 \rightarrow 32$	48	$16 \rightarrow 16$	96
DeConv 2	$64 \rightarrow 32$	3	$32 \rightarrow 16$	3	$16 \rightarrow 10$	3
Conv block 7	$64 \rightarrow 32$	24	$32 \rightarrow 16$	48	$20 \rightarrow 10$	77
	$32 \rightarrow 32$	24	$16 \rightarrow 16$	48	$10 \rightarrow 10$	77
DeConv 3	$32 \rightarrow 16$	3	$16 \rightarrow 8$	3	$10 \rightarrow 8$	3
Conv block 8	$32 \rightarrow 16$	24	$16 \rightarrow 8$	48	$16 \rightarrow 8$	48
	$16 \rightarrow 16$	24	$8 \rightarrow 8$	48	$8 \rightarrow 8$	48
deconv4	$16 \rightarrow 8$	3	$8 \rightarrow 4$	3	$8 \rightarrow 4$	3
Conv block 9	$16 \rightarrow 8$	24	$8 \rightarrow 4$	48	$8 \rightarrow 4$	48
	$8 \rightarrow 3$	24	$4 \rightarrow 3$	55	$4 \rightarrow 3$	55
segmenter	$3 \rightarrow 1$	1	$3 \rightarrow 1$	1	$3 \rightarrow 1$	1

Table 5. Comparison of the architectures of the Intuitive Fat-U-Nets. Unlike a Fat-U-Net, which is converted using a FatNet algorithm for the conversion, these intuitive networks were developed manually by choosing smaller channel sizes and computing the new kernel sizes without taking into account the number of pixels in the feature map.

Model	Oxford IIIt pet		HeLa cells	
	Acc	IoU	Acc	IoU
Intuitive Fat-U-Net 1	92.71	83.71	99.08	93.90
Intuitive Fat-U-Net 2	89.39	77.84	97.95	87.58
Intuitive Fat-U-Net 3	89.18	76.98	98.45	89.75
Fat-U-Net	**93.40**	**91.87**	**99.43**	**96.25**

Table 6. Other "Large kernel/Few Channel" architectures in comparison with Fat-U-Net.

initially trained the U-Net to state-of-the-art standards before converting it to Fat-U-Net. Our U-Net implementation is marginally outperformed only by networks with pre-trained VGG16 and Inception V3 contracting paths (Table 3). As our implementation was trained from scratch, we believe it met the required standards before conversion. Fat-U-Net sacrificed only 1.93% in pixel accuracy, 4.24% in IoU, and 2.46% in Dice score. These results compare favourably to classification problems, where the accuracy drop was 6%.

Qualitative results in Figure 2 reveal that U-Net and Fat-U-Net exhibit distinct behaviour in various scenarios. Figure 2(a) is the demonstration of the perfect segmentation by both algorithms in instances where pets are clearly visible against a monochromatic background. Interestingly, in Figure 2(b), U-Net outperforms Fat-U-Net by segmenting a background cat, which is not part of the ROI. However, we can see the advantage of Fat-U-Net in Figure 2(c), where it has perfectly segmented both animals, in contrast to U-Net, which incorrectly classified some pixels of the cat and dog as background.

Our evaluation of Fat-U-Net for HeLa cell nucleus segmentation proved successful. Compared to the 4-staged U-Net,[21] our 5-stage U-Net implementation demonstrated marginally better performance on middle-range slices

and achieved a 14.94% higher IoU for all slices. It is important to consider that the ground truth for ROI cells includes only the segmentation of the central cell, excluding adjacent cells. Nevertheless, both U-Net and Fat-U-Net managed to segment these nuclei even with noisy ground truth data (Figure 3). Consequently, the segmented mask outperforms the ground truth on side slices (non-150-200), resulting in a lower IoU for all slices compared to middle-range slices. After converting to Fat-U-Net, the performance loss was smaller than in the Oxford IIIt pet dataset evaluation, at approximately 1% for middle-range slices and 2% for all slices. For the large images, Fat-U-Net provided better results than U-Net as can be seen in the cells on the bottom right.

To assess Fat-U-Net's performance in the original optical setup, it was trained using the 4f simulator. While this simulator does not completely replicate the real optics' performance, it demonstrated comparable results in the training for Hela Cells segmentation. Notably, it achieved an IoU of 95.58% on test slices 150-200 and 65.34% on all test slices.

5. CONCLUSION AND FUTURE WORK

In our research, we have successfully extended the application of FatNet conversion to the task of segmentation, by adapting the U-Net architecture for use with free-space optical accelerators. We have achieved 538 times fewer convolution operations in Fat-U-Net compared to U-Net, meaning 538 times faster inference when both networks run with the optical accelerator and 37 times faster inference compared to U-Net run on GPU. Both networks were evaluated across the Oxoford IIIt pet dataset and HeLa cell nucleus segmentation, on which we have achieved state-of-the-art performance. When it comes to the performance loss, the maximum loss was 4.24% in the test IoU for the Oxford IIIt pet dataset and 1.76% in the test IoU of HeLa cells nucleus segmentation, making the FatNet transformation even more preferable than the classification.

As this research primarily focuses on Fat-U-Net conversion, future work could investigate segmentation using only the contracting path of Fat-U-Net, to explore the advantages of high-resolution kernels in detail. Hypothetically, a U-Net with an extensive receptive field like in Fat-U-Net would not require skip connections. However, our experiments with U-Net and Fat-U-Net without skip connections yielded unsatisfactory results, even after removing the 3x3 convolutions that replaced transposed convolutions. A possible explanation is that Fat-U-Net maintains the U-Net architecture, and instead of pooling down feature maps, it increases kernel resolution, resulting in a feature map/kernel ratio similar to U-Net. Therefore, future work will include investigating the possibility of the enhancement of the effective receptive field by dropping the skip connections.

REFERENCES

[1] Ronneberger, O., Fischer, P., and Brox, T., "U-Net: Convolutional Networks for Biomedical Image Segmentation," (May 2015). arXiv:1505.04597 [cs].

[2] Badrinarayanan, V., Kendall, A., and Cipolla, R., "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **39**, 2481–2495 (Dec. 2017).

[3] Farabet, C., Couprie, C., Najman, L., and LeCun, Y., "Learning Hierarchical Features for Scene Labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **35**, 1915–1929 (Aug. 2013).

[4] Waldrop, M. M., "The chips are down for Moore's law," *Nature News* **530**, 144 (Feb. 2016). Cg_type: Nature News Section: News Feature.

[5] Lin, X., Rivenson, Y., Yardimci, N. T., Veli, M., Luo, Y., Jarrahi, M., and Ozcan, A., "All-optical machine learning using diffractive deep neural networks," *Science* **361**, 1004–1008 (Sept. 2018). Publisher: American Association for the Advancement of Science.

[6] Sui, X., Wu, Q., Liu, J., Chen, Q., and Gu, G., "A Review of Optical Neural Networks," *IEEE Access* **8**, 70773–70783 (2020).

[7] Miscuglio, M., Hu, Z., Li, S., George, J. K., Capanna, R., Dalir, H., Bardet, P. M., Gupta, P., and Sorger, V. J., "Massively parallel amplitude-only Fourier neural network," *Optica* **7**, 1812–1819 (Dec. 2020). Publisher: Optica Publishing Group.

[8] Li, B., Ersoy, O. K., Ma, C., Pan, Z., Wen, W., and Song, Z., "A 4F optical diffuser system with spatial light modulators for image data augmentation," *Optics Communications* **488**, 126859 (2021).

[9] Peng, C., Zhang, X., Yu, G., Luo, G., and Sun, J., "Large Kernel Matters – Improve Semantic Segmentation by Global Convolutional Network," in [*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*], 4353–4361 (2017).

[10] Noh, H., Hong, S., and Han, B., "Learning Deconvolution Network for Semantic Segmentation," in [*Proceedings of the IEEE International Conference on Computer Vision*], 1520–1528, IEEE, Santiago, Chile (2015).

[11] Long, J., Shelhamer, E., and Darrell, T., "Fully Convolutional Networks for Semantic Segmentation," in [*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*], 3431–3440 (2015).

[12] Krizhevsky, A., Sutskever, I., and Hinton, G. E., "ImageNet Classification with Deep Convolutional Neural Networks," in [*Advances in Neural Information Processing Systems*], **25**, Curran Associates, Inc. (2012).

[13] LeCun, Y., Jackel, L. D., Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Muller, U. A., Sackinger, E., Simard, P., and others, "Learning algorithms for classification: A comparison on handwritten digit recognition," *Neural networks: the statistical mechanics perspective* **261**(276), 2 (1995).

[14] Ibadulla, R., Chen, T. M., and Reyes-Aldasoro, C. C., "FatNet: High-Resolution Kernels for Classification Using Fully Convolutional Optical Neural Networks," *AI* **4**, 361–374 (June 2023). Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

[15] Culshaw, B., "The Fourier Transform Properties of Lenses," in [*Introducing Photonics*], 132–135, Cambridge University Press, Cambridge (2020).

[16] Li, S., Miscuglio, M., Sorger, V., and Gupta, P., "Channel Tiling for Improved Performance and Accuracy of Optical Neural Network Accelerators," *ArXiv* (2020).

[17] Parkhi, O. M., Vedaldi, A., Zisserman, A., and Jawahar, C. V., "Cats and dogs," in [*2012 IEEE Conference on Computer Vision and Pattern Recognition*], 3498–3505 (June 2012).

[18] Peddie, C. J., Jones, M. L., and Collinson, L. M., "Serial Block Face SEM of HeLa cell pellet with 10 nm pixels and 50 nm slices (benchmark dataset)," (May 2019). 10.6019/EMPIAR-10094.

[19] Peddie, C. J., Jones, M. L., and Collinson, L. M., "Cropped regions from Serial Block Face SEM of HeLa cell pellet with 10 nm pixels and 50 nm slices (benchmark dataset)," (Aug. 2020). 10.6019/EMPIAR-10478.

[20] Karabağ, C., Jones, M., and Reyes-Aldasoro, C. C., "Multiple Nuclei HeLa cell ground truth images with four labels (nuclear envelope, nucleus, rest of the cell, and background) for deep learning architecture training.," (Mar. 2022). doi.org/10.5281/zenodo.6355622.

[21] Karabağ, C., Ortega-Ruíz, M. A., and Reyes-Aldasoro, C. C., "Impact of Training Data, Ground Truth and Shape Variability in the Deep Learning-Based Semantic Segmentation of HeLa Cells Observed with Electron Microscopy," *Journal of Imaging* **9**, 59 (Mar. 2023). Number: 3 Publisher: Multidisciplinary Digital Publishing Institute.

[22] Sangalli, M., Blusseau, S., Velasco-Forero, S., and Angulo, J., "Scale-Equivariant U-Net," in [*33rd British Machine Vision Conference 2022, London, UK*], {BMVA} Press, London, UK (Nov. 2022).

[23] Edwards, J. and El-Sharkawy, M., "uICNet: Lightweight Image Segmentation," in [*2022 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*], 99–104 (Oct. 2022).

[24] Zhao, H., Qi, X., Shen, X., Shi, J., and Jia, J., "ICNet for Real-Time Semantic Segmentation on High-Resolution Images," in [*Computer Vision – ECCV 2018*], Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y., eds., *Lecture Notes in Computer Science*, 418–434, Springer International Publishing, Cham (2018).

[25] Dippel, J., Lenga, M., Goerttler, T., Obermayer, K., and Höhne, J., "Transfer Learning for Segmentation Problems: Choose the Right Encoder and Skip the Decoder," (July 2022). arXiv:2207.14508 [cs].

[26] Dippel, J., Vogler, S., and Höhne, J., "Towards Fine-grained Visual Representations by Combining Contrastive Learning with Image Reconstruction and Attention-weighted Pooling," (Feb. 2022). arXiv:2104.04323 [cs].

[27] Sundarrajan, K., Rajendran, B. K., and Balasubramanian, D., "Fusion of Ensembled UNET and Ensembled FPN for Semantic Segmentation," *Traitement du Signal* **40**, 297–307 (Feb. 2023).