One Shot Model For The Prediction of COVID-19 and Lesions Segmentation In Chest CT Scans Through The Affinity Among Lesion Mask Features

Aram Ter-Sarkisov
CitAI Research Center,
Department of Computer Science,
City, University of London
alex.ter-sarkisov@city.ac.uk

Abstract

We present a novel framework that integrates segmentation of lesion masks and prediction of COVID-19 in chest CT scans in one shot. In order to classify the whole input image, we introduce a type of associations among lesion mask features extracted from the scan slice that we refer to as affinities. First, we map mask features to the affinity space by training an affinity matrix. Next, we map them back into the feature space through a trainable affinity vector. Finally, this feature representation is used for the classification of the whole input scan slice.

We achieve a 93.55% COVID-19 sensitivity, 96.93% common pneumonia sensitivity, 99.37% true negative rate and 97.37% F1-score on the test split of CNCB-NCOV dataset with 21192 chest CT scan slices. We also achieve a 0.4240 mean average precision on the lesion segmentation task. All source code, models and results are publicly available on `https://github.com/AlexTS1980/COVID-Affinity-Model`.

Keywords: COVID-19, Computer Vision, Object detection, Object segmentation, Image classification,

1. Introduction

There are three main approaches to the early detection of COVID-19: reverse transcription polymerise chain reaction (RT-PCR), which is considered to be the golden standard for the COVID-19 diagnosis [1], chest x-ray (CXR) images and chest computer tomography (CT) scans. CXR is more rapid, and hence preferable at the times of high workload at radiological departments. CT scans are slower to obtain, but they are more accurate, because of the axial (slices of images), rather than the frontal angle of CXR. Both methods quickly became an active area of research in the Deep Learning (DL) community, with a large number of open-source datasets, such as CNCB-NCOV [2] and models, such as COVIDNet [3], JCS [4], COVIDNet-CT [5] and many other. 2D lung CT analysis can also be extended to 3D, as in [6], by combining a set of slices to obtain the full scan. In addition to RT-PCR, CXR and CT scans, datasets of lungs ultrasound scans are sometimes used, as in [7].

There is a number of DL approaches to COVID-19 classification: straightforward combination of an existing or custom-made feature extractor, see e.g. [8, 9, 10, 11, 12], augmentation of the classification module with the segmentation functionality, e.g. in [4, 13, 14], and a fusion of instance segmentation of lesions with the image-level classification, [15, 16]. In addition, several models segment instances of lesions, without

classifying the whole slice or scan. In such case medical practitioner has to infer the class of the image from the location and features of the lesions in order to classify the scan, see e.g. [17].

Recently a new set of segmentation models was introduced. In [18], authors introduce a dual-branch combination net that segments lungs and lesions in individual CT slices: it predicts the slices class and fuses all slices to classify the whole scan. In [19] the input in the model is a 3D CT scan which is sliced into a bag of separate images and patches of lesions to classify the severity of lung infection.

An example of an analytical rather that DL solution was introduced in [20]. It uses histogram normalization and several optimization tools to find multilevel thresholds for lesion segmenation in chest CT scans. In [21] an encoder-decoder model utilizing atrous convolution and attention was introduced for lesion segmentation in CT slices.

Although in most publications the reported model's accuracy is high, their real-life applicability is limited due a number of factors, such as a small dataset used for training and validation, and hence limited capacity to generalize to in-the-wild data. For example, [14, 8, 11] predict two classes (e.g. COVID-19 and Common Pneumonia) instead of three (COVID-19, Common Pneumonia and Control). Many models rely on a large training split (e.g. [5]) and a long list of data augmentation tricks (image cropping, shearing, jitter, etc) alongside a relatively small test split. At the same time, in [22] a much smaller dataset was used for training, due to the functionality of Mask R-CNN [23] that extracts mask feature maps and segments different types of lesions derived from regions of interest (RoIs). Also, in [24] good results were achieved using truncated ResNet18 and ResNet34 backbones.

In order to improve the performance and stability of the models, some of them use only regions in feature maps instead of entire feature maps (localization of features). In [25] an ensemble of majority-voting ResNet18 networks was trained on sets of randomly sampled and cropped patches from different areas of chest X-rays. In [4] a UNet pixel-level semantic segmentation model was used to extract image-level masks to augment the classifier's feature maps. Often a combination of segmentation and classification helps with deriving saliency maps that increases the explainability of the model's results, such as in [25, 4].

Novelty of our approach can be summarized in the following way:

1. Methodology: we repurpose lesion mask features that Mask R-CNN uses for lesion mask segmentation to classify the whole input chest CT scan slice. This is done by deriving multiple affinities, a type of association among lesion mask features, using trainable Affinity matrix and Affinity vector. Affinity representations are computed for all channels (feature maps) in RoIs, and then mapped back into the feature space to predict the class of the image.

2. Accuracy: for the classification problem, our best model achieves 93.55% COVID-19 sensitivity and 97.37% F1 score on test split. For lesion segmentation, our best model achieves mean average precision (mAP), main MS COCO criterion, of 0.424. These results improve on those reported in [22, 15] and most reported OS models, including Mask R-CNN

3. Domain adaptation: we finetune and evaluate our model to another large COVID-19 dataset, iCTCF. As with CNCB-NCOV, finetuning to iCTCF uses a small share of the training split, but achieves 95.73% COVID-19 sensitivity and 97.53% F1 score. This demonstrates the potential of our model to generalize to the unseen data.

The rest of the paper is structured as following: Section 2 discusses the datasets used for segmentation and classification problems, Section 3 introduces the model's methodology, Section 4 presents results and analysis of the experiments (including domain adaptation) and Section 5 concludes.

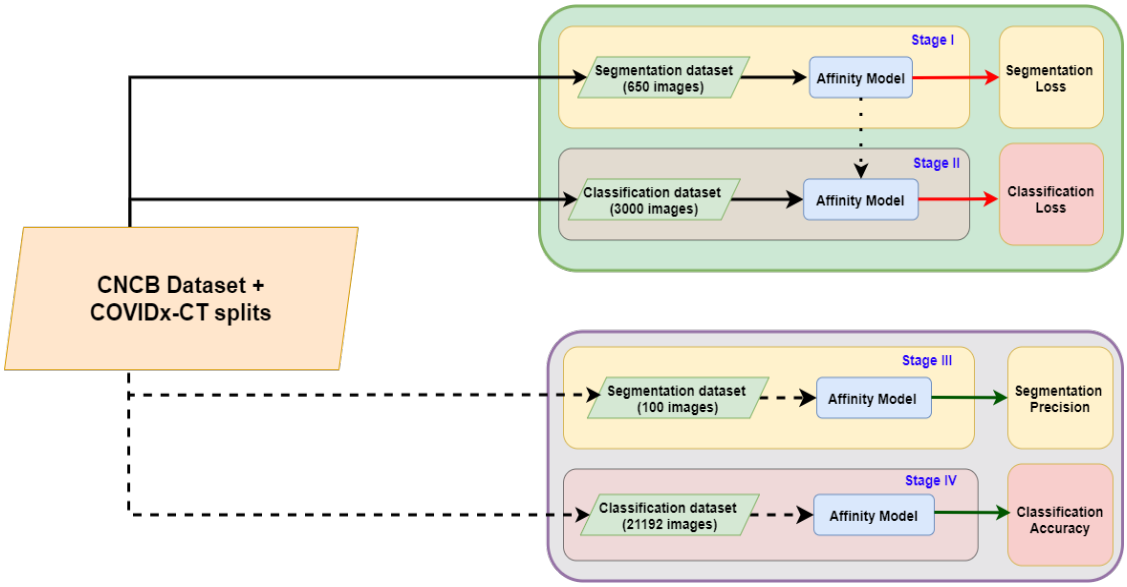


Figure 1: Overall flowchart of the algorithm. Normal black arrows: training data (images and labels), broken black arrows: test data (images and labels), dotted black arrow: weight copy, red arrows: loss computation, green arrows: model evaluation. Best viewed in color.

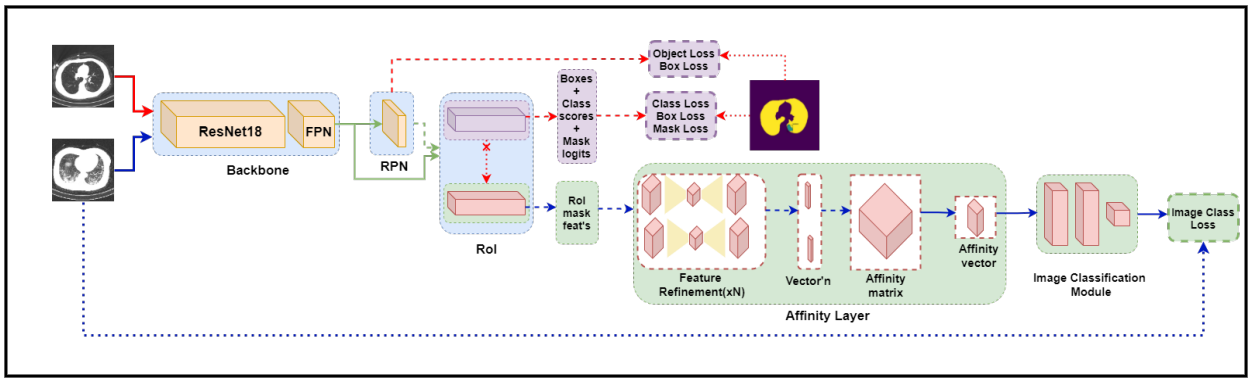


Figure 2: One Shot Affinity Model. Features from FPN (last backbone layer) are passed to RPN (Region Proposal Net) layer to predict raw bounding boxes and both of them are passed to the RoI layer to extract RoIs and predict class, box and mask of objects. Light blue background: Mask R-CNN layers shared by the segmentation and classification problems; purple background + blocks: RoI segmentation branch; light green background + red blocks: RoI classification branch, Affinity layer and image classifier; blocks with broken edges: loss computation. Normal arrows: features, broken arrows: batches, dotted arrows: labels, broken+cross arrow: weight copy from the segmentation to the classification branch. The details of the RoI segmentation and classification branches see Figure 3. Best viewed in color.

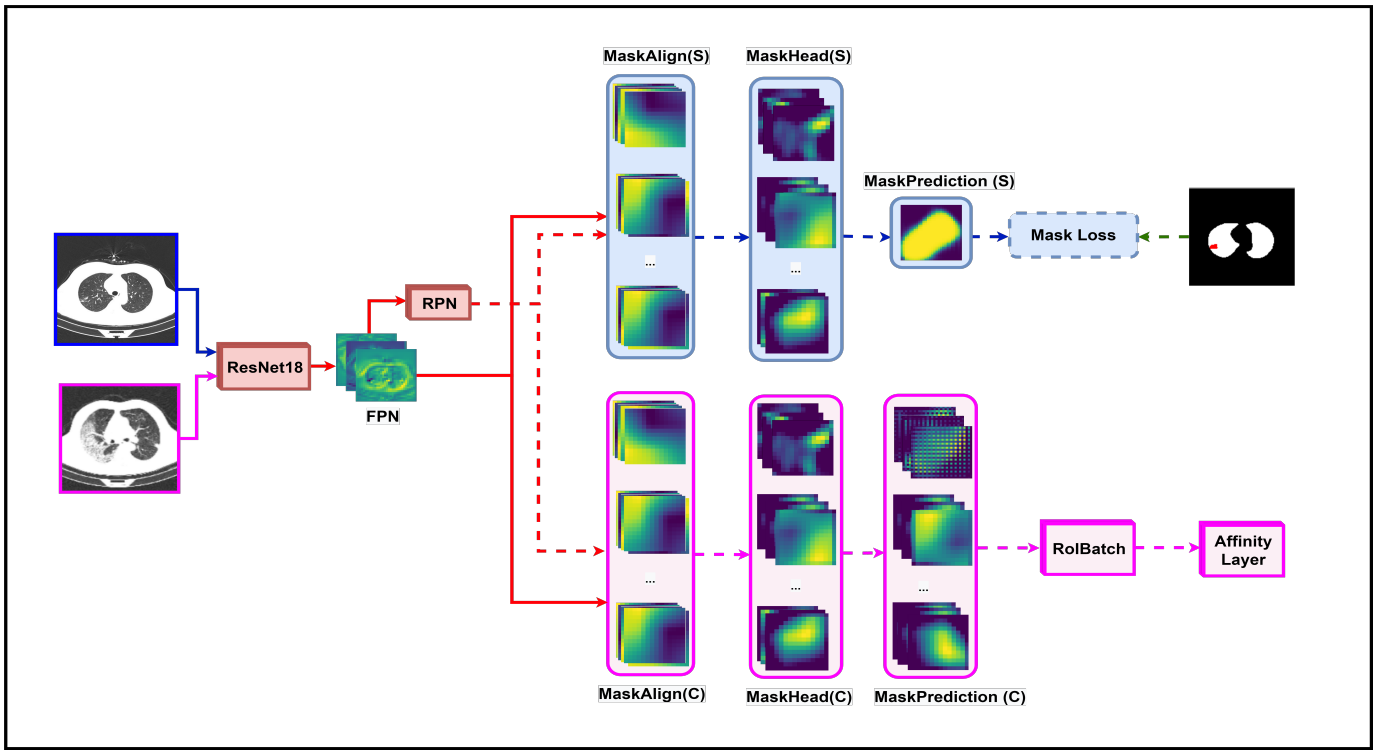


Figure 3: Region of Interest layer with segmentation and classification branches. For clarity and emphasis on masks, we omitted RoI box and label functionality. Backbone (ResNet18+FPN) and RPN functionality (red blocks) is the same for both stages: features (normal red arrows) and raw boxes (broken red arrows). In the segmentation stage (light blue background), mask loss (blocks with broken edges) is computed by taking prediction from MaskPrediction(S) (blue broken arrows) and gt labels (green broken line). At classification stage (light magenta background) batches (broken magenta arrows) are fed forward through the classification branch. MaskAlign(S) and MaskAlign(C): map raw boxes to FPN features to extract RoI (lesion) mask features. MaskHead(S) and MaskHead(C): mask feature refinement. MaskPrediction(S): mask features upsampling, predictions and binarization for loss computation, MaskPrediction(C): mask features upsampling. Note mask features in MaskAlign(S), MaskAlign(C) and MaskHead(S), MaskHead(C) are the same in both stages due to the parametrization thereof. Best viewed in color.

2. Data and Related Work

Raw chest CT scan data is taken from CNCB-NCOV repository, [2], `http://ncov-ai.big.ac.cn/download`. It is provided by CNCB-NCOV and is split into two subsets: experimental data taken from COVID-19 positive patients labelled at pixel level (clean lungs and lesion masks) that we use for the segmentation problem (750 images), and the data labelled at slice level that we use for the classification problem (over 104000 images). The former dataset contains a total of 3 positive (non-background): Ground Glass Opacity (GGO), Consolidation (C) and healthy lungs. The latter dataset contains 3 image-level labels: COVID-19, Common Pneumonia (CP) and Normal/Control.

Summary of the segmentation dataset is presented in Table 1. The presence of negative slices (slices that do not contain any lesions) in COVID-19-positive patients is explained by the specifics of the manifestation of COVID-19: slices closer to the top or bottom of the lungs tend to have smaller lesion manifestation than those in the middle.

We merged clean lungs and the background, and two types of lesions into one, therefore segmentation labels contain only one 'positive' class: lesions. Train/validation/test splits for the classification model in Table 2 are taken from COVIDx-CT [5]. This is done for the sake of comparability with other models

Table 1: Summary of the CNCB-NCOV [2] segmentation dataset. Only positive images (images with lesions) are used to train the model. Both positive and negative slices are used to evaluate the model.

Split	Positive	Negative	Total
Train/Val	475	175	650
Test	71	29	100

Table 2: Summary of the CNCB-NCOV[2] classification dataset and COVIDx-CT splits.

Split	COVID-19 patients	CP patients	Normal patients	Total patients	COVID-19 slices	CP slices	Normal slices	Total slices	Ratio Train/Test
Train[5]	300	420	144	864	12520	22061	27201	61782	2.9153
Train-ours	10	10	10	30	1000	1000	1000	3000	**0.1415**
Validation	95	190	47	332	4529	7400	9107	21036	-
Test	116	125	52	293	4346	7395	9450	21192	-

and consistency of the results, as the negative images for positive classes (i.e. lesion-free slices taken from COVID/CT-positive patients) were removed from the final data by the developers of COVIDx-CT. Instead of using the full train split, we randomly sampled a small subset (second row in Table 2). Only 3000 images in total (1000 per class) were used to train all models reported in Section 4.4. Validation and test splits were used in full (21036 and 21192 images resp.). We used the remainder of the training data (over 58000 images) for the domain adaptation studies, see Section 4.5.

2.1. Comparison To Other Solutions

Some of the best results in the available literature, e.g. [5, 25, 3] and many other depend on a number of dataset augmentation tricks. In addition, in [5], the ratio of train to test split 2.9153 (last column in Table 2) is very high, i.e. the test split is much smaller than the train split, which makes our solution, with ratio of 0.1415 more robust. On top of it, we did not use any of the following data balancing/augmentation tricks implemented in many other studies:

1. Image enhancement (histogram equalization),
2. Image manipulation (rotation, jittering, random bounding boxes, shearing, cropping, etc),
3. Class-based resampling (class balancing).

While these tricks are legitimate and widely used, they obfuscate the insight, to what extent the reported results are due to the model's efficiency rather than these data tricks. Apart from the subtraction of the global mean and division by the global standard deviation, we applied no other data manipulations to either dataset, which is one of the strengths of our approach. Additional statistical analysis of the data in Tables 1 and 2 was presented in [5, 22, 16].

The selected CNCB-NCOV dataset is preferable to most other publicly available COVID-19 datasets:

- Radiopedia (`www.radiopedia.org`) medical segmentation dataset contains scans from 9 patients (CNCB-NCOV:150 patients)
- UCSD dataset[13] contains 746 images across two classes (COVID vs Non-COVID),
- COVID-CTset dataset[26] contains 12058 images across two classes (COVID vs Normal),
- SARS-COV-2-CT-Scan dataset[27] contains 2481 images across two classes (COVID vs CP).

For the domain adaptation study in Section 4.5 we used the iCTCF dataset ([28, 29]) with 13676 images, and the rest of the CNCB-NCOV training data that was not used for the training of the model in Section 4

3. Methodology

The overall protocol of the algorithm is visualized in Figure 1: light green background is the training protocol, light magenta background is testing protocol. Stages I and III: segmentation training and testing, Stages II and IV: classification training and testing.

The overall architecture of the Affinity model is presented in Figure 2. Backbone (feature extractor + FPN), RPN and segmentation branch in the RoI layer are inherited from Mask R-CNN. Augmentations of its architecture for the classification problem in this study are classification branch in the RoI layer, Affinity layer and image classifier. Augmentation of the RoI layer with the classification branch was first introduced in [15].

For the architecture of the RoI layer see Figure 3. Segmentation branch (box and class functionality omitted for clarity) solves the problem of detection, classification and segmentation of objects in CT images, and is trained using instance-level gt data (boxes, class labels and masks). Classification branch has the same architecture as the segmentation branch. Its weights are not trainable; instead, they are copied from the segmentation branch each training iteration.

For the detailed discussion of Mask R-CNN, [30], in the context of COVID-19 segmentation, see [22, 15], especially for the difference between training and evaluation (test) stages and weight copy from segmentation into classification branch. Here we will only explain the concept of lesion (RoI) mask features, which is critical to our study.

3.1. RoI Mask Features for Lesion Segmentation

One of the essential concepts of instance segmentation, are local (regional) features. RoI layer accepts a batch of raw box candidates from RPN and global (image-level) features from FPN, and, using an algorithm called RoIAlign, maps raw RPN boxes to global FPN features, to crop, resize, and extract two sets of batches with RoI (regional) features: one with box+class features, the other with mask features. RoI box and class features are used to predict box coordinates and object classes, RoI mask features are used to segment masks of each object.

In Figure 3, for better clarity, we omit RoI box+class functionality, and show only RoI mask features. Lesion mask features for object segmentation are extracted from MaskAlign(S), and passed through MaskHead(S) and MaskPrediction(S) (light blue background). RoI batch of mask features has dimensionality $b \times C \times H \times W$, where b is the batch size, C is the number of channels, H, W are its height and width. Finally, in MaskPrediction(S) object masks are predicted and binarized, compared to gt labels to compute and backpropagate segmentation loss.

3.2. Lesion mask features for image classification

The first novelty of our solution is repurposing RoI mask features for image classification. As shown in Figure 3, classification branch (light magenta background) has the same architecture and the functionality as the segmentation branch (light blue background). MaskAlign(C) extracts RoI batch, passes it through MaskHead(C) and MaskPrediction(C). Unlike MaskPrediction(S), MaskPrediction(C) does not extract object masks, so its output is a batch of RoI mask features with dimensionality $\beta \times C \times H \times W$, where β is the size of this batch, and the remaining variables are the same as defined in Section 3.1. Batch of RoI mask features is the only output of the RoI classification branch, that is accepted by the Affinity layer.

The novelty of this approach cf. [15] is the replacement of RoI box and class confidence scores with RoI mask features. The advantage of the new approach is explained by the fact that RoI mask features contain a more detailed and rich information about the objects. In [15] the output of the classification branch has

dimensionality $\beta \times 5$ (batch size, 4 box coordinates + confidence score). Unlike [15], where only β is a hyperparameter, in this study all four dimensionality values are hyperparameters.

From now on we will refer to lesion (RoI) mask features as RoIs, for simplicity and clarity.

3.3. Affinity Layer

Main contribution of our study is using RoIs for image classification. This is done in the Affinity layer in three stages: first, the RoIs are accepted, refined, and vectorized. Next, trainable Affinity matrix and Affinity vector compute affinities (associations) among RoIs, and output a single feature vector. Image class is predicted from this vector in the image classifier.

3.3.1. RoI mask feature refinement

The objective of this stage is to obtain features relevant to the classification problem by refining them is several stages. Although these stages are similar to MaskHead and MaskPredictor layers in RoI classification branch, the key difference is that RoI refinement layer is trainable wrt image class loss.

In this stage, RoIs' dimensionality is downsampled from $\beta \times C \times H \times W$ to $\beta \times C \times H/2 \times W/2$, then upsampled back to $\beta \times C \times H \times W$ a total of N times: $N \times (Conv2D \rightarrow BatchNorm2D \rightarrow ReLU \rightarrow ConvTranspose2D)$ layers. The kernel size of $ConvTranspose2D$ is the same as $Conv2D$, which guarantees that the output of the refinement stage has the same dimensionality as the input, $\beta \times C \times H \times W$.

3.3.2. RoI vectorization

Next, we vectorize RoIs, i.e. we transform the batch of RoIs from dimensionality $\beta \times C \times H \times W$ to $\beta \times C \times 1 \times 1$ in order to simplify affinity computation and reduce processing time. This is done in three steps: in the first two steps, RoIs' H and W are halved, in the last step they are downsampled to $1 \times 1 : 3 \times Conv2D \rightarrow ReLU \rightarrow BatchNorm2D$. H, W must be divisible by 2 twice. Kernel sizes of the first two $Conv2D$ layers are 2×2, and the kernel size of the third one is divisible by the height and width of the features after the first two. These layers are also trainable wrt image loss. After these operations the dimensions of RoI batch is $\beta \times C \times 1 \times 1$, which we simplify to $\beta \times C$.

We denote each RoI vector $\mathbf{x}_k, 1 \leq k \leq \beta$, and the expression (value) in j^{th} channel in the k^{th} RoI is $x_{k,j}$. Matrix of the RoI batch is $\mathbf{X}$ with dimensions $\beta \times C$. Matching RoI vectorization to the original RoI mask features logic in the RoI layer, each value $x_{k,j}$ is a 'summary' of the j^{th} feature map in the k^{th} RoI, obtained by downsampling the feature map from $H \times W$ to 1×1.

3.3.3. Affinity Matrix $\boldsymbol{W}$

This is the key novelty of the whole model, in which we derive a set of affinities among RoIs. For each channel j in RoIs there exists a set of C^* different affinities. Each value in this set is the strength of the corresponding affinity in channel j.

We refer to this stage as semi-supervised learning, because none of the β RoIs is labelled in any way. Only the input image has a label (COVID-19, CP, Normal, see Figure 2). Therefore, the weights in the Affinity matrix and Affinity vector are updated only using gradieants from the image loss.

Each vector $\mathbf{x}_k$ has C channels, so, by composing the vectors row-wise, the RoI matrix $\mathbf{X}$ can be written as a matrix size $\beta \times C$ of row vectors $\mathbf{x}_k$, see Equation 1. Rows are RoIs, and each column is a vector of values in the j^{th} channel, $\hat{\mathbf{x}}_j$. We introduce a trainable Affinity matrix $\mathbf{W}$, which consists of C^* column vectors $\mathbf{w}_r$, Equation 2, so the dimensions of $\mathbf{W}$ are $\beta \times C^*$. The purpose of $\mathbf{W}$ is to map RoIs from feature

space to affinity space, therefore, each $\mathbf{w}_r$ maps RoIs to the r^{th} subspace of affinities.

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_\beta \end{bmatrix} \tag{1}$$

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 : \mathbf{w}_2 : \ldots : \mathbf{w}_{C^*} \end{bmatrix} \tag{2}$$

Our objective is to learn C^* affinities for each of C channels across all RoIs. To do so, we have to take a dot-product between each $\mathbf{w}_r$ and $\hat{\mathbf{x}}_j$, Equations 3 and 4, therefore, $\mathbf{W}$ is transposed. Each term $w_{r,k}x_{k,j}$ in the sum in Equation 4 is the strength of r^{th} affinity in the j^{th} channel in k^{th} RoI. Therefore, dot-product $y_{r,j}$ is the strength of the r^{th} affinity in the j^{th} channel across all RoIs. We refer to $\mathbf{Y}$ as affinity representation matrix.

$$\mathbf{Y} = \mathbf{W}^T \otimes \mathbf{X} = \begin{bmatrix} \mathbf{y_1} : \mathbf{y_2} : \ldots : \mathbf{y}_\mathrm{C} \end{bmatrix} \tag{3}$$

$$y_{r,j} = \mathbf{w}_r \circ \hat{\mathbf{x}}_j = \sum_{k=1}^{\beta} w_{r,k}x_{k,j} \tag{4}$$

Matrix $\mathbf{Y}$ consists of C column vectors $\mathbf{y}_j$, so the dimensionality of $\mathbf{Y}$ is $C^* \times C$. Example of the affinity representation matrix is given in Figure 4. Although $\mathbf{Y}$ could be used as an image class predictor, collapsing it into a vector will speed up the computation.

3.3.4. Affinity Vector $\boldsymbol{v}$

Once affinity representation matrix $\mathbf{Y}$ is obtained, we take its matrix-vector product with the trainable affinity vector $\mathbf{v}$ with C^* elements. Vector $\mathbf{v}$ scales the affinities, i.e. weighs their effect, and maps them back into the feature space, Equations 5-7.

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{C^*} \end{bmatrix} \tag{5}$$

$$\mathbf{z} = \mathbf{Y^T} \odot \mathbf{v} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_C \end{bmatrix} \tag{6}$$

$$z_j = \mathbf{y}_j \circ \mathbf{v} = \sum_{r=1}^{C^*} y_{j,r}v_r \tag{7}$$

We transpose $\mathbf{Y}$ and take dot-product of each $\mathbf{y}_j$ with $\mathbf{v}$ to get vector of feature representations $\mathbf{z}$, Equation 6. Each term in the sum in Equation 7, $y_{j,r}v_r$ is the weight of the r^{th} affinity in channel j. Matching this expression to the RoI mask feature logic in the RoI layer, z_j in Equation 7 is a value that expresses features in j^{th} channel across all RoIs, adjusted for C^* affinities among these features. Vector $\mathbf{z}$ is the only output of the Affinity layer.

Table 3: Average Precision on the segmentation data test split (100 images). Best results in bold.

# Affinities	AP@50%IoU	AP@75%IoU	mAP
8	0.590	0.389	**0.424**
16	0.571	0.352	0.391
32	**0.614**	0.382	0.395
64	0.603	**0.414**	0.422
Mask R-CNN (head only) [22]	0.511	0.301	0.298
Mask R-CNN (full)[22]	0.565	0.413	0.352

3.4. Image Classification Module and Loss Functions

Feature vector $\mathbf{z}$ is the input in the final image classification module with two fully connected layers and the output class logits layer with 3 neurons (one neuron per class).

The model solves a segmentation and classification problem simultaneously in one shot, using a linear combination of two loss functions, Equation 8.

$$L_{Total} = L_{SEG} + L_{CLS} \tag{8}$$

$$L_{CLS} = -t_k^{Cl} \log \sigma(h_k(\mathbf{s})) - \sum_{k \neq Cl} (1 - t_k^{Cl}) \log(1 - \sigma(h_k(\mathbf{s}))) \tag{9}$$

$$t_k^{Cl} = \begin{cases} 1 \text{ if k=Cl} \\ 0 \text{ else} \end{cases}$$

In Equation 8 L_{SEG} loss is the same as in Mask R-CNN [30]: two losses in RPN (object/background+raw boxes), and three in RoI (boxes, classes and masks). Equation 9 is per-class binary cross-entropy. This means that we compute 3 image class loss values. Here σ is the sigmoid function, $h_k(\mathbf{s})$ is a logit value of the k^{th} neuron for some linear input map $\mathbf{s}$, Cl is the correct class of the input image.

4. Experimental results

4.1. Training Protocol

Each iteration, the presented model is trained in two stages: segmentation stage and classification stage.

Segmentation stage. Gradient computation and weight updates are switched off for the classification branch in the RoI layer, Affinity layer and image classifier. Segmentation loss is computed for the output in RPN and the segmentation branch in RoI. Weights in the backbone, RPN and RoI segmentation branch are updated using gradients from L_{SEG}. Weights from RoI segmentation branch are copied into the RoI classification branch.

Classification stage. Gradient computation and weight updates are switched off for RPN and RoI (both branches). Image-level loss is computed using Equation 9 for the output of the image classifier. Weights in backbone, Affinity layer and image classifier are updated using gradients from L_{CLS}.

4.2. Implementation details

The model was trained using Adam optimizer with a fixed learning rate of $1e-5$ and regularization hyperparameter $1e-3$. Image batch size was set to 1 for both problems. RoI mask features batch size β was set to 8 to minimize computation time (also, increasing it did not lead to a noticeable improvement in the

accuracy). Hence, the dimensionality of the batch of RoI mask features (output of the classification branch) is $8 \times 256 \times 28 \times 28$ (256: number of channels in FPN, 28: hyperparameter for height and width of RoIs). The input in the Affinity layer after feature refinement and vectorization is 8×256. We experimented with four C^* values: $8, 16, 32, 64$ (further increase of C^* did not improve the model's performance). The number of stages in the feature refinement N was set to 3.

Each model was trained for 100 epochs, which took about 14 hours. All experiments were carried out on a single GPU with 8Gb of VRAM. Backbone for all models was ResNet18+FPN. All other hyperparameters for training and evaluation of both stages were taken from [22, 15].

Segmentation and classification results presented in Sections 4.3 and 4.4, clearly demonstrate three observations:

1. Increase in the number of affinities among RoI mask features does not necessarily translate into the progress in accuracy both in segmentation and classification, across all metrics
2. This observation is also true for each particular model: for example, models that achieve high classification accuracy do not match it in segmentation, The opposite is also true.
3. The new solutions improve on benchmark models in segmentation for all metrics, and in classification for all metrics except common pneumonia

4.3. Segmentation Results

For the details of the segmentation precision metrics, see [31], as we used average precision (AP) at two Intersect over Union (IoU) threshold values: AP@50%IoU and AP@75%IoU, and mean AP (mAP) across 10 IoU thresholds between 50% and 95% with a 5% step (main MS COCO metric).

We merged clean lungs with the background, and GGO with C, so there is only one 'positive' class (lesions). At test (evaluation) time the model outputs a pre-defined set of box and class predictions with softmax confidence scores, and binarized masks for each predicted object. All predictions with a confidence score exceeding a pre-set threshold are output alongside their boxes, classes and masks. We set this confidence threshold to 0.05, and the maximum number of object predictions to 100, see Figure 6 for the visualization.

A very import hyperparameter is mask threshold, which is a value used for mask binarization: mask logits (MaskPrediction(S) in Figure 3) are converted into a binary mask: all pixels exceeding the threshold are mask, the rest are background. We set this threshold to 0 (which corresponds to 0.5 threshold for sigmoid).

Precision is computed using overlaps between predicted (binarized) object masks and gt masks. If the overlap exceeds the IoU threshold (e.g. 50%), and the class is predicted correctly, the prediction is a True Positive. If either of these conditions fails, or the object has a better prediction, it is a False Positive. If a lesion is not predicted, it is a False Negative. Once all predictions are obtained, a Precision-Recall curve is constructed to calculate the average precision for the input image.

In Table 1 there is a large number of negative images, i.e. images without any lesions. At training stage, they are discarded. At test stage, we use the following rule: if the input image is negative, and the model doesn't predict any lesions therein, its precision is 1. In case there are no objects and at least one prediction, or there are objects and no predictions, precision is 0.

For the rest of hyperparameters and computation details see
`https://github.com/AlexTS1980/COVID-Affinity-Model` and [22, 15].

Segmentation results are presented in Table 3. The model with the number of affinities $C^* = 8$ achieved a 0.424 mAP, which is well above the MS COCO average. So far, we could not identify a different COVID-19

open-source lesion segmentation model and reimplement it in a similar way to add to the benchmark comparison. Instead, we add results for Mask R-CNN from [22] that was trained with the same hyperparameters. We report the full model (all weights were trainable) and 'heads' only (RPN and RoI trainable, backbone frozen with weights pretrained on ImageNet).

From the results in Table 3 we can infer two things: first, our model improves on both Mask R-CNN models across all three criteria. For mAP, our best model ($C^* = 8$) outperforms the full model by 0.072, and the weakest one ($C^* = 16$) by 0.039. The smallest gap between our best model and the full Mask R-CNN is observed for AP@75%IoU: $C^* = 64$ outperforms it only by 0.001.

Next, none of the Affinity model architectures consistently achieves top result across all criteria. For AP@50%IoU model with $C^* = 32$ outperforms the next best ($C^* = 64$) by 0.011. For AP@75%IoU model with $C^* = 64$ outperforms the next best ($C^* = 8$) by 0.024. For mAP, model with $C^* = 8$ outperforms the next best ($C^* = 64$) by 0.002.

In Figure 6 we present examples of segmentation outputs. CT scan slice in Figure 6a is taken roughly from the middle section of the scan. There is a single large lesion (GGO) at the bottom of the left lung (mask on the right). The model correctly predicts, classifies and segments the lesion with a high softmax confidence of 99%. The map in the middle shows the same prediction with sigmoid intensity of the mask prediction.

Figure 6b displays the scan from the lower lobe, with two small lesions areas in the left lung. Although both are accurately detected and classified with softmax confidence of 85% and 75%, mask segmentation is not very accurate. This is clear from the middle image with the sigmoid intensity of the predicted masks that is concentrated mostly on the left of the first predicted box (prediction with confidence of 0.85%), and starts to fade quickly outside of it. The same is true for the second prediction (confidence of 0.75%). As a result, after binarization, most of the mask remains unpredicted. In Figure 6c the slice (upper lobe) contains only clean lungs. The 'prediction' is a background with a confidence of 0.001%.

4.4. Classification Results

To compute classification accuracy, we used per-class (c) sensitivity, precision and class-adjusted F_1 score, Equations 10-12.

$$Sens(c) = \frac{TP}{TP + TN} \tag{10}$$

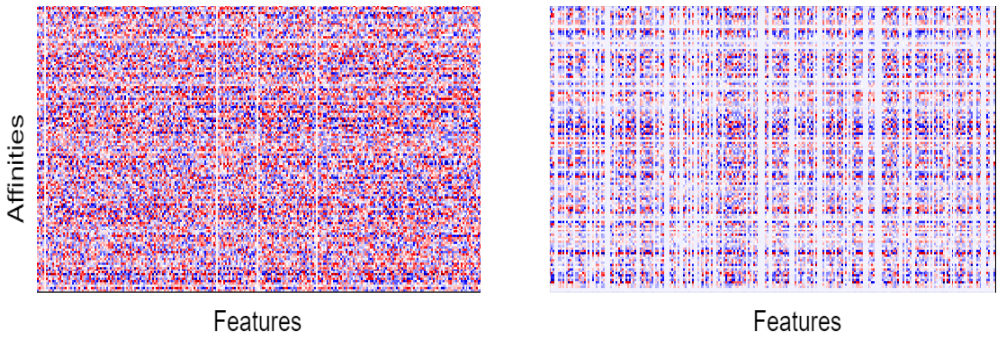
$$Prec(c) = \frac{TP}{TP + FP} \tag{11}$$

$$F_1 score = 2 \sum_{c=1}^{C} w(c) \times \frac{Sens(c) \times Prec(c)}{Sens(c) + Prec(c)} \tag{12}$$

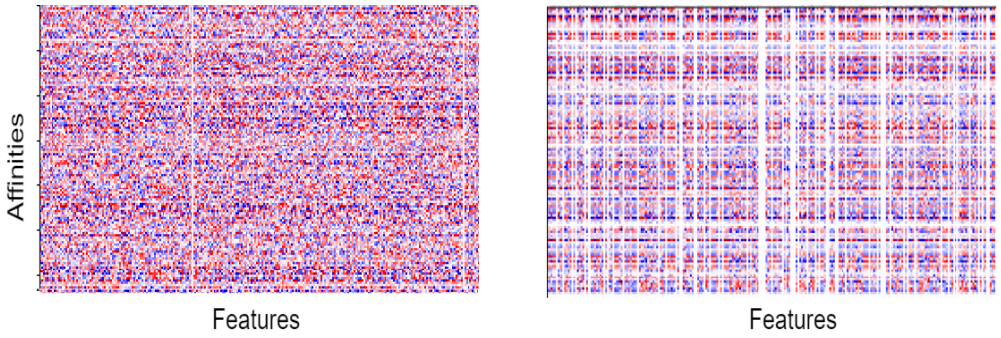
where $w(c)$ is the share of each class in the test split. Details of the test split are in Table 2. The share of COVID-19 in the test split is 22%, the share of CP is 35%, and the share of Normal is 43%.

Classification accuracy results for each attempted number of affinities C^* are presented in Table 4. For the benchmark comparisons, ResNet18 was trained and tested on our data from scratch, COVIDNet-CT[5] model was evaluated using the model available online. Results for COVIDNet-CT reported in Table 4 are different from the ones in [5]. Results for DarkCOVIDNet [33] were taken from `https://github.com/muhammedtalo/COVID-19`. All other results were taken from the respective publications.

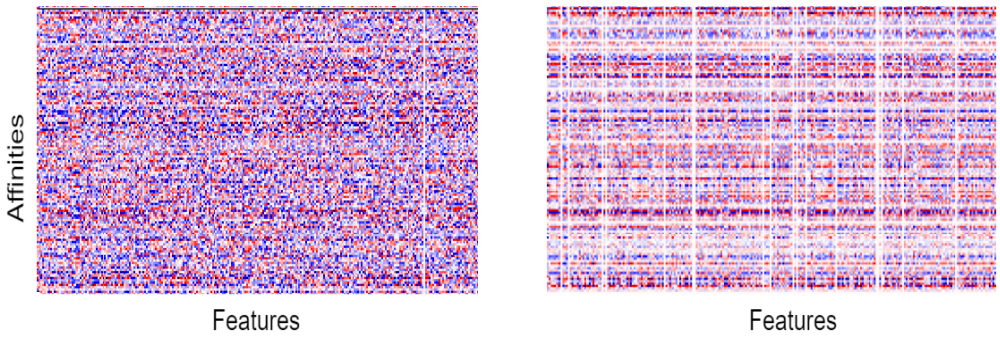
Overall, the model with $C^* = 16$ produces the largest number of top results. It achieves the highest COVID-19 sensitivity of 93.55%, Negative class sensitivity of 99.47% and F1 score of 97.37%. Affinity



(a) Negative



(b) Common Pneumonia



(c) COVID-19

Figure 4: Affinity representation, $\mathbf{Y}$ matrix in Equation 3 with dimensionality $C^* \times C$ for COVID-19, CP, and Normal classes. Each value in the matrix is $y_{r,j}$, Equation 4. Left column: affinities after 1 epoch, right column: affinities after 100 epochs. Color gamma is normalized between 255 (red, strong positive affinity) and -255 (blue, strong negative affinity). Best viewed in color.

Table 4: Class sensitivity and F1 scores on the COVIDx-CT test split (21191 images). Best class results in bold.

Model	COVID-19	CP	Negative	F_1score	Test set size
Ours (8 affinities)	92.34%	97.74%	99.13%	97.26%	
Ours (16 affinities)	**93.55%**	96.93%	**99.47%**	**97.37%**	21191
Ours (32 affinities)	90.93%	97.08%	99.06%	96.71%	
Ours (64 affinities)	93.50%	96.57%	96.85%	96.05%	
COVID-CT-Mask-Net [22]	90.80%	91.62%	91.10%	91.50%	21191
ResNet50 [13]	85.90%	-	-	88.10%	746
COVNet [9]	90.00%	-	-	89.04%	434
LightCNN [32]	88.23%	-	-	84.56%	392
ResNet18	92.59%	96.25%	92.03%	93.61%	21191
DarkCOVIDNet[33]	85.71%	85.22%	93.57%	89.73%	224
COVIDNet-CT[5]	81.70%	**98.74%**	98.95%	95.34%	21191
VGG16[28]	85.00%	97.00%	98.00%	-	19685

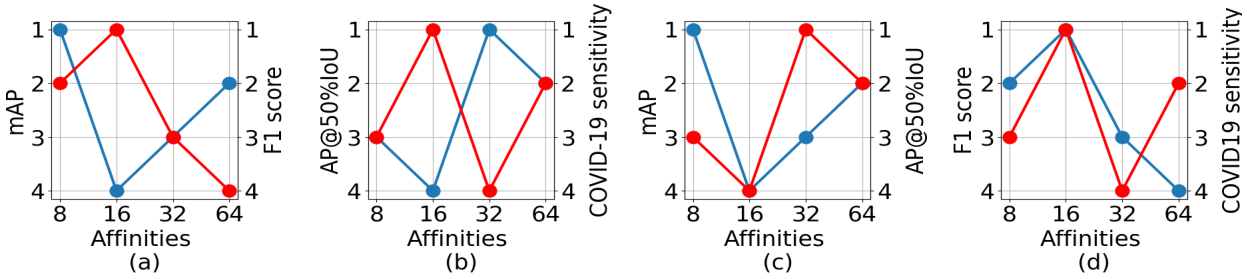


Figure 5: Rank of each model for two criteria. Blue: left axis, red: right axis. Figure 5(a): mAP vs F1 score, Figure 5(b): AP@50%IoU vs COVID-19 sensitivity, Figure 5(c): mAP vs AP@50%IoU, 5(d): F1 score vs COVID-19 sensitivity. Best viewed in color.

models outperform the best benchmark models in all classes, except CP: Model with $C^* = 16$ outperforms ResNet18 by 0.95% for COVID-19, COVIDNet-CT by 0.51% for Normal, and COVIDNet-CT by 2.03% in F1 score. The main difference from segmentation results, is that $C^* = 16$ consistently achieves top results, but does worse than other models in segmentation problem.

From the comparison of the models' performance for both problems we can observe that the actual number of affinities cannot be used as a predictor of the model's success. This observation is visualized in Figures 5(a)-(d), in which we plot each model's rank for each criterion. First, we plotted mAP vs F1 score (Figure 5(a)) and AP@50%IoU vs COVID-19 sensitivity (5(b)). Possibly the only consistent increase can be observed for F1 score for 16 and greater number of affinities. Next, we plot mAP vs AP@50%IoU (two main segmentation criteria, 5(c)) and F1 score vs COVID-19 sensitivity (two main classification criteria, Figure 5(d)). We justify these sets of pairs of metrics by the fact that mAP and F1 score are the main metrics for, resp. segmentation and classification, and AP@50%IoU and COVID-19 are next important. Same logic applies to the other two pairs.

Affinity representation matrix $\mathbf{Y}$ for the model with $C^* = 64$ and 3 different classes of input images is visualized in Figures 4a-4c after training for 1 (left) and 100 (right) epochs (in the latter case the input image is always correctly classified). Each element $y_{r,j}$ of the matrix is the strength of the r^{th} affinity in the j^{th} channel. The structure of the matrix in Figure 4a, Negative class, displays a greater presence of inactive (pale/white) affinities, but several affinities are consistently negative across all channels. Figures 4b, Common Pneumonia, and 4c, COVID-19, are similar in the way that both of them have a large number of both strong and weak affinities, and a large number of affinities is consistently strong across all channels.

Table 5: Class Sensitivity results on the COVIDx-CT left-out train split (58782 images) and F1 score. Best results in bold.

# Affinities	COVID-19	CP	Negative	F_1score
8	97.97%	97.22%	99.33%	98.31%
16	98.30%	97.10%	**99.43%**	98.38%
32	**98.69%**	**98.18%**	99.16%	**98.41%**
64	98.37%	96.78%	97.25%	97.31%

Table 6: Summary of the iCTCF-CT[28, 29] classification dataset.

Split	COVID-19	Negative	Total
Train/Val	300	300	600
Test	3701	9275	12976

Also, a large number of affinities change sign and magnitude across channels.

4.5. Ablation Study and Domain Adaptation

4.5.1. CNCB-NCOV Dataset (left-out part of the train split)

A total of 58782 images from the train split (see Table 2) were left out of the main investigation.

Classes breakdown in this split do not deviate substantially from the test and validation splits: normal class is 45.14%, CP is 36.24%, COVID-19 is 18.16%. Table 5 reports the models' accuracy. Overall, these results are comparable to those in the test split, see Table 4. The model with $C^* = 16$ achieves the highest Negative class sensitivity of 99.43%, the model with $C^* = 32$ achieves the highest COVID-19 sensitivity of 98.69%, CP sensitivity of 98.18% and F1 score of 98.41%. With respect to F1 score, compared to the results in Table 4, it improves $C^* = 8$ result by 1.04%, $C^* = 16$ by 1.00%, $C^* = 32$ by 1.70% and $C^* = 64$ by 1.26%.

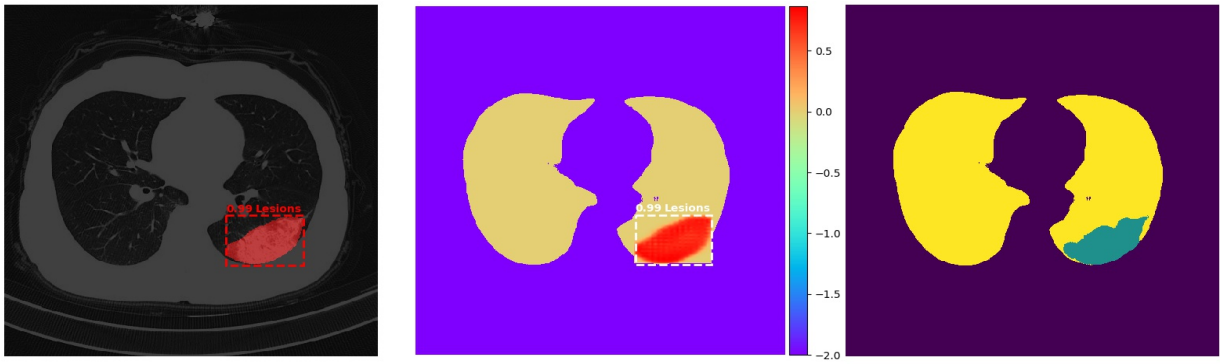
4.5.2. iCTCF Dataset

One of the challenges of fast and reliable COVID-19 diagnosis is the generalization of models to other datasets. According to [34], there is a number of pitfalls that prevent DL models from being used in the real clinical environment. These issues include both methodological flaws, and the reproducibility of the results. Although publications tend to report optimistic results, achieved by specific models, their methodology often suffers from the lack of the details of the data manipulation, training protocol, and also from various forms of data bias (labels, experimental settings, control groups, etc).

Most importantly, many datasets are either small, or suffer from a severe form of class imbalance in test splits, which inflates the models' accuracy, which does not translate into the ability to generalize to external datasets, or the data available in the real clinical environment. The use of varying, and often unclear evaluation metrics prevents the direct comparison of the models. Also, many of the models are not publicly available (e.g. on Github), which diminishes the reproducibility of results even more. Most publications do

Table 7: Accuracy results on the iCTCF-CT test split (12976 images). Best results in bold.

# Affinities	COVID-19	Negative	F_1score
8	95.73%	**98.30%**	**97.53%**
16	97.81%	97.39%	97.51%
32	**99.35%**	83.27%	87.88%
64	97.92%	91.46%	93.30%
VGG16[28]	97.00%	85.47%	-



(a) Slice 73

(b) Slice 103

(c) Slice 23

Figure 6: Segmentation of chest CT scan slices from test set (see Table 1). Left column: detection of lesions (bounding box + softmax class confidence + mask). Masks are binarized by using the threshold of 0.5 for pixel-level sigmoid values for each lesion, see [22] for details. Central column: detection of lesions with pixel-level sigmoid mask scores before binarization. Right column: ground truth.

not address any of these challenges, regardless of the chosen methodology or datasets.

To overcome some of these shortcomings, we extend our findings to a large open-source dataset iCTCF-CT [28], `http://ictcf.biocuckoo.cn` with two classes: Negative and COVID-19. The dataset details are summarized in Table 6. We split the data randomly into 600 training and validation and 12976 test images (note the class imbalance in the test split) achieving the test to train+validation split ratio of 21.62. We finetuned each model from the weights trained on CNCB-NCOV data for 10 epochs, which took about 15 minutes on the same GPU with 8Gb VRAM. No changes were applied to the models' architecture except the final classification layer, replacing 3 neurons with 2. The results on iCTCF presented in Table 7, confirm that Affinity models adapt quickly and achieve high accuracy on the external data after a very short finetuning.

Comparison to the result in [29] is mostly favorable. The original paper did not report F1 score; VGG16 achieved high COVID-19 sensitivity of 97.00%, and low Negative sensitivity of 85.74%, i.e. the solution suffered from high false positive predictions. Two models: $C^* = 16, 64$ improve on the base one across both classes: the former by 0.81% for COVID-19 and 11.92% for Negative. The latter outperforms it by 0.92% for COVID-19 and 5.98% for Negative. The one with $C^* = 32$ achieves top COVID-19 sensitivity of 99.35% thus beating the baseline by 2.35%, but it is also our only algorithm that has true negative rate lower than the baseline. Finally, the smallest model, $C^* = 8$ is the only one with COVID-19 sensitivity lower than the baseline by 1.27%. At the same time, it achieves both highest true negative rate and top F1 score of 97.53%.

Our models have the potential to be trained on a mix of datasets, which can further improve their ability to generalize to the new data and the real clinical environment. This will be an important avenue of our future work in this area.

4.6. Limitations of the methodology

After careful analysis of the results across three datasets and two problems, we can clearly identify one serious limitation of the presented Affinity methodology. By comparing results in Tables 3-5 and 7, despite the fact that Affinity models overall outperform benchmark models, we can point out, that none of the attempted Affinity models consistently achieved top results across all datasets, problems and precision criteria. The second problem is that models that achieved top results for CNCB-NCOV classification, did not perform well on CNCB-NCOV segmentation or iCTCF (e.g. $C^* = 16$) compared to other affinity models. The model that achieved top mAP and high F1 accuracy in iCTCF data, output weaker results across all classification criteria (e.g. $C^* = 8$).

This limitation motivates further study of the effect of affinity representation on the performance of the Affinity model, as affinity representation is the main step in the computation of features for image classification.

5. Conclusions

In this paper we presented a novel methodology for a one-shot solution of lesion segmentation and COVID-19 classification problems.

Main methodological novelty of the solution is affinity, a type of associations among Region of Interest mask features. The output of the affinity computation is used to predict the class of the whole image. Using a small fraction of the CNCB-NCOV training data and without any complicated data manipulation, we achieved strong results both in segmentation and classification problems. Domain adaptation studies on iCTCF dataset further validated the strength and the potential of our solution to generalize to the new data. Thorough investigation of the structure and the effect of the Affinity representation on the model's accuracy and generalization will be the focus of our future research.

All source code, models and results are publicly available on
https://github.com/AlexTS1980/COVID-Affinity-Model

References

[1] X. Xie, Z. Zhong, W. Zhao, C. Zheng, F. Wang, J. Liu, Chest ct for typical 2019-ncov pneumonia: relationship to negative rt-pcr testing, Radiology (2020) 200343.

[2] K. Zhang, X. Liu, J. Shen, Z. Li, Y. Sang, X. Wu, Y. Zha, W. Liang, C. Wang, K. Wang, et al., Clinically applicable ai system for accurate diagnosis, quantitative measurements, and prognosis of covid-19 pneumonia using computed tomography, Cell (2020).

[3] L. Wang, A. Wong, Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images, arXiv preprint arXiv:2003.09871 (2020).

[4] Y.-H. Wu, S.-H. Gao, J. Mei, J. Xu, D.-P. Fan, C.-W. Zhao, M.-M. Cheng, Jcs: An explainable covid-19 diagnosis system by joint classification and segmentation, arXiv preprint arXiv:2004.07054 (2020).

[5] H. Gunraj, L. Wang, A. Wong, Covidnet-ct: A tailored deep convolutional neural network design for detection of covid-19 cases from chest ct images, arXiv preprint arXiv:2009.05383 (2020).

[6] Z. Han, B. Wei, Y. Hong, T. Li, J. Cong, X. Zhu, H. Wei, W. Zhang, Accurate screening of covid-19 using attention based deep 3d multiple instance learning, IEEE Transactions on Medical Imaging (2020).

[7] S. Roy, W. Menapace, S. Oei, B. Luijten, E. Fini, C. Saltori, I. Huijben, N. Chennakeshava, F. Mento, A. Sentelli, et al., Deep learning for classification and localization of covid-19 markers in point-of-care lung ultrasound, IEEE Transactions on Medical Imaging (2020).

[8] C. Butt, J. Gill, D. Chun, B. A. Babu, Deep learning system to screen coronavirus disease 2019 pneumonia, Applied Intelligence (2020) 1–7.

[9] L. Li, L. Qin, Z. Xu, Y. Yin, X. Wang, B. Kong, J. Bai, Y. Lu, Z. Fang, Q. Song, et al., Artificial intelligence distinguishes covid-19 from community acquired pneumonia on chest ct, Radiology (2020).

[10] T. Yan, P. K. Wong, H. Ren, H. Wang, J. Wang, Y. Li, Automatic distinction between covid-19 and common pneumonia using multi-scale convolutional neural network on chest ct scans, Chaos, Solitons & Fractals 140 (2020) 110153.

[11] Y. Song, S. Zheng, L. Li, X. Zhang, X. Zhang, Z. Huang, J. Chen, H. Zhao, Y. Jie, R. Wang, Y. Chong, J. Shen, Y. Zha, Y. Yang, Deep learning enables accurate diagnosis of novel coronavirus (covid-19) with ct images, medRxiv (2020).

[12] S. Yazdani, S. Minaee, R. Kafieh, N. Saeedizadeh, M. Sonka, Covid ct-net: Predicting covid-19 from chest ct images using attentional convolutional network, arXiv preprint arXiv:2009.05096 (2020).

[13] J. Zhao, Y. Zhang, X. He, P. Xie, Covid-ct-dataset: a ct scan dataset about covid-19, arXiv preprint arXiv:2003.13865 (2020).

[14] A. Amyar, R. Modzelewski, H. Li, S. Ruan, Multi-task deep learning based ct imaging analysis for covid-19 pneumonia: Classification and segmentation, Computers in Biology and Medicine 126 (2020) 104037.

[15] A. Ter-Sarkisov, Single-shot lightweight model for the detection of lesions and the prediction of covid-19 from chest ct scans, medRxiv (2020). doi:10.1101/2020.12.01.20241786.

[16] A. Ter-Sarkisov, Detection and segmentation of lesion areas in chest CT scans for the prediction of COVID-19, medRxiv (2020). doi:10.1101/2020.10.23.20218461.

[17] D.-P. Fan, T. Zhou, G.-P. Ji, Y. Zhou, G. Chen, H. Fu, J. Shen, L. Shao, Inf-net: Automatic covid-19 lung infection segmentation from ct images, IEEE Transactions on Medical Imaging (2020).

[18] K. Gao, J. Su, Z. Jiang, L.-L. Zeng, Z. Feng, H. Shen, P. Rong, X. Xu, J. Qin, Y. Yang, et al., Dual-branch combination network (dcn): Towards accurate diagnosis and lesion segmentation of covid-19 using ct images, Medical image analysis 67 (2021) 101836.

[19] K. He, W. Zhao, X. Xie, W. Ji, M. Liu, Z. Tang, Y. Shi, F. Shi, Y. Gao, J. Liu, et al., Synergistic learning of lung lobe segmentation and hierarchical multi-instance classification for automated severity assessment of covid-19 in ct images, Pattern recognition 113 (2021) 107828.

[20] A. Oulefki, S. Agaian, T. Trongtirakul, A. K. Laouar, Automatic covid-19 lung infected region segmentation and measurement using ct-scans images, Pattern recognition 114 (2021) 107747.

[21] X. Zhao, P. Zhang, F. Song, G. Fan, Y. Sun, Y. Wang, Z. Tian, L. Zhang, G. Zhang, D2a u-net: Automatic segmentation of covid-19 ct slices based on dual attention and hybrid dilated convolution, Computers in biology and medicine (2021) 104526.

[22] A. Ter-Sarkisov, COVID-CT-Mask-Net: Prediction of COVID-19 from CT Scans Using Regional Features, medRxiv (2020). URL: https://github.com/AlexTS1980/COVID-CT-Mask-Net. doi:10.1101/2020.10.11.20211052.

[23] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[24] A. Ter-Sarkisov, Lightweight model for the prediction of COVID-19 through the Detection and Segmentation of lesions in chest ct scans, medRxiv (2020). doi:10.1101/2020.10.30.20223586.

[25] Y. Oh, S. Park, J. C. Ye, Deep learning covid-19 features on cxr using limited training data sets, IEEE Transactions on Medical Imaging (2020).

[26] M. Rahimzadeh, A. Attar, S. M. Sakhaei, A fully automated deep learning-based network for detecting covid-19 from a new and large lung ct scan dataset, medRxiv (2020). URL: https://www.medrxiv.org/content/early/2020/06/12/2020.06.08.20121541. doi:10.1101/2020.06.08.20121541.

[27] E. Soares, P. Angelov, S. Biaso, M. H. Froes, D. K. Abe, Sars-cov-2 ct-scan dataset: A large dataset of real patients ct scans for sars-cov-2 identification, medRxiv (2020). URL: https://www.medrxiv.org/content/early/2020/05/14/2020.04.24.20078584. doi:10.1101/2020.04.24.20078584.

[28] W. Ning, S. Lei, J. Yang, Y. Cao, P. Jiang, Q. Yang, J. Zhang, X. Wang, F. Chen, Z. Geng, et al., Open resource of clinical data from patients with pneumonia for the prediction of covid-19 outcomes via deep learning, Nature biomedical engineering (2020) 1–11.

[29] W. Ning, S. Lei, J. Yang, Y. Cao, P. Jiang, Q. Yang, J. Zhang, X. Wang, F. Chen, Z. Geng, et al., ictcf: an integrative resource of chest computed tomography images and clinical features of patients with covid-19 pneumonia (2020).

[30] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask r-cnn, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 2961–2969.

[31] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick, Microsoft coco: Common objects in context, in: European conference on computer vision, Springer, 2014, pp. 740–755.

[32] M. Polsinelli, L. Cinque, G. Placidi, A light cnn for detecting covid-19 from ct scans of the chest, arXiv preprint arXiv:2004.12837 (2020).

[33] T. Ozturk, M. Talo, E. A. Yildirim, U. B. Baloglu, O. Yildirim, U. R. Acharya, Automated detection of covid-19 cases using deep neural networks with x-ray images, Computers in biology and medicine 121 (2020) 103792.

[34] M. Roberts, D. Driggs, M. Thorpe, J. Gilbey, M. Yeung, S. Ursprung, A. I. Aviles-Rivero, C. Etmann, C. McCague, L. Beer, et al., Common pitfalls and recommendations for using machine learning to detect and prognosticate for covid-19 using chest radiographs and ct scans, Nature Machine Intelligence 3 (2021) 199–217.