Logo Generation Using Regional Features: A Faster R-CNN Approach to Generative Adversarial Networks

Aram Ter-Sarkisov[0000−0002−1300−6132] and Eduardo Alonso[0000−0002−3306−695X]

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

Abstract. In this paper we introduce Local Logo Generative Adversarial Network (LL-GAN) that uses regional features extracted from Faster R-CNN for logo generation. We demonstrate the strength of this approach by training the framework on a small style-rich dataset of real heavy metal logos to generate new ones. LL-GAN achieves Inception Score of 5.29 and Frechet Inception Distance of 223.94, improving on state-of-the-art models StyleGAN2 and Self-Attention GAN.

Keywords: Deep Learning · Generative Adversarial Networks · Logo Generation.

1 Introduction

Generative Adversarial Networks (GANs) were first introduced in [7]. They have gained a wide recognition in the Artificial Intelligence community due to their ability to approximate the distribution of real data by generating fake data. Recent advances include Progressive-Growing GANs, StyleGAN and StyleGAN2 that learn styles at different resolutions[14–16], Self-Attention GANs (SAGANs) that learn the connections between different spatial locations[29], CycleGANs and Pix2Pix GANs for unpaired style transfer[30, 12] and Wasserstein loss function[1].

Faster R-CNN and Mask R-CNN[24, 6, 9] are state-of-the-art open-source deep learning algorithms for object detection and instance segmentation that work in multiple stages, unlike single-shot models like YOLO[23].

Faster R-CNN first predicts regions containing objects based on overlaps (Intersect over Union, IoU) between fixed-size rectangles known as anchors and ground truth bounding boxes using Region Proposal Network (RPN). Then, it pools features from these areas by cropping and resizing corresponding areas in features maps. This is done using Region of Interest Pooling (RoIPool) to

construct fixed-size Regions of Interest (RoIs) containing rescaled regional features for each object (later replaced by more accurate Region of Interest Align, RoIAlign[9]). These local features are fed through fully connected (`fc`) layers to independently predict the object classes and refine bounding box prediction. In addition to this, Mask R-CNN segments objects' masks.

One of the new and challenging areas in GANs and neural style transfer is the creation of logos and fonts. This area includes style and shape transfer between fonts[3, 4], logo synthesis[26, 21, 19], transfer of style to font[2] and font generation[8]. A specific challenge in this area is disentaglement of content and style learning, often done through training of two different encoders and feature concatenation, as in [4], and separation of transfer of shape and texture (ornamentation), done through pretraining of the shape model and ornamentation model that takes the shapes and adds ornamentation[3]. Logo synthesis (style transfer), as in [21, 19, 26], also uses conditional input (random vector + sparse vector for the class).

We address the shortcomings of the state-of-the-art models, such as the size of the output, which in most cases is limited to 64x64 pixels. This size is sufficient for separate characters/glyphs or small logos, as readability does not suffer. For larger logos or words, model output must be upsampled. Another limitation we address is the size of the training data: we leverage Faster R-CNN's capacity to sample a batch of regional features in a single image to overcome the need for a large dataset.

In this paper we present a GAN model for generating logos of heavy metal bands. To the best of our knowledge, it would be the first GAN study that is focused on the generation of band logos. With respect to specifically heavy metal logos, recently, there were two related publications: in [28] style transfer model based on [5] was used to fuse the style of heavy metal bands logos, e.g. Megadeth and the content of corporate logos, e.g. Microsoft. In [25] the styling of heavy metal logos and its association with genre and readability are investigated.

Measured by Frechet inception distance[11], Inception score[27] and detection accuracy, the presented model confidently outperforms the state-of-the-art StyleGAN2 and SAGAN frameworks. Our contribution consists of the following:

- Local Logo GAN (LL-GAN) framework: training the Generator by comparing regional features extracted from the fake and real data using RoIAlign module in Faster R-CNN. Since loss is computed only on regional features, the Generator's parameters receive updates only from the region containing the logo in the real data. This model augments the baseline GAN framework, serving as an additional source of gradients for the Generator's parameters. Ground truth bounding box is used to determine positive RoIs in the fake image, therefore the Generator learns to output spatially-aware logos. A number of RoIs is sampled from each image using RPN and RoIAlign mod-

ules, which compensates for the sparsity of the data,

– Logo generator. The model is capable of generating style-rich heavy metal logos consisting of glyph-like structures that closely resemble real-life band logos without suffering from the mode collapse. This includes an augmentation of the DCGAN's model architecture[22] that allows for creation of large images (282×282),

– Style-rich metal band logos dataset. Images with heavy metal band logos were scraped from the internet and labelled at text level (bounding box around the band's logo). Each image contains a single-word logo, with a simple background (e.g. black or white) across 10 bands selected for the style of the logo. The dataset consists of 923 images and an equal number of bounding box coordinates of the logo.

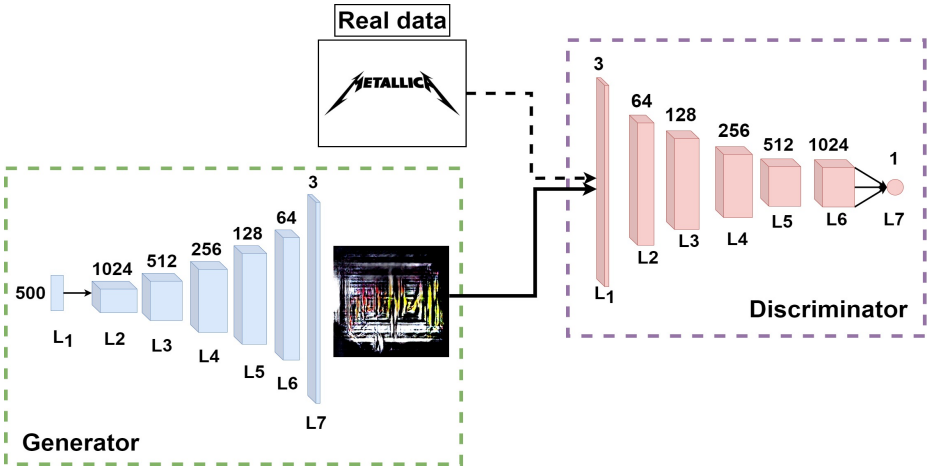

Fig. 1: DCGAN+ framework. Details of the architecture of both models is presented in Table 2. Values in each module in the number of feature maps in the Convolution (Discriminator) or Transposed Convolution (Generator) models. Normal arrows: features and fake data, broken arrow: real data.

2 Our Approach

Model sizes and structures are compared in Table 1.

2.1 DCGAN+ framework

DCGAN+ is an augmentation of the DCGAN architecture[22] that enables generation of larger images in a single shot. The main idea behind the architecture is the selection of the right rate of upsampling and downsampling of feature

maps in each model (kernel size, stride, padding). Figure 1 and Table 2 provide a summary of the models' architectures. This solution successfully addresses the problem of the size of the generated logo, as we increase it from at most 64×64, as in [26] to 282×282.

Table 1: Comparison of sizes of the frameworks. G: generator, D: discriminator, F: Faster R-CNN.

Framework	Number of Parameters	Structure of the framework
DCGAN+	43.83M + 3.93M	G + D
LL-GAN	43.83M + 3.93M + 41.43M	G + D + F
StyleGAN2 [16]	84.69M (Total)	G + D
StyleGAN2 w/attention [16]	85.87M (Total)	G + D
SAGAN [29]	8.1M + 4.92M	G + D
DCGAN [22]	3.5M + 2.7M	G +D
Faster R-CNN [10]	41.80M	F

Table 2: DCGAN+ framework. G: Generator, D: Discriminator

Model	Block	Depth	Kernel	Stride	Pad
G	L_1(Input)	500	0	0	1
	L_2	1024	8	2	0
	L_3	512	4	2	0
	L_4	256	4	2	1
	L_5	128	4	2	1
	L_6	64	2	2	1
	L_7(`tanh`)	3	2	2	1
D	L_1(Input)	3			
	L_2	64	4	2	1
	L_3	123	3	2	1
	L_4	256	3	2	1
	L_5	512	3	2	1
	L_6	1024	3	2	1
	L_7(`fc`)	1	-	-	-

2.2 LL-GAN framework

Overall framework is presented in Figure 2. Generator and Discriminator are the same as in DCGAN+. One of the key contributions of this paper is the use of local features from the RoIAlign stage in Faster R-CNN to compute style loss. We use the ground truth bounding box around the band logo to extract one RoI from the real data, skipping the RPN stage. For the fake data, RPN predicts raw boxes passed on to RoIAlign that uses these predictions to extract RoI features and outputs B positive predictions (i.e. RoI box predictions that have IoU with the ground truth box greater than a pre-defined threshold), each of fixed size $H \times W \times C$. Each RoI's height and width are hyperparameters, and depth C is determined by the depth of the FPN feature map, see [18].

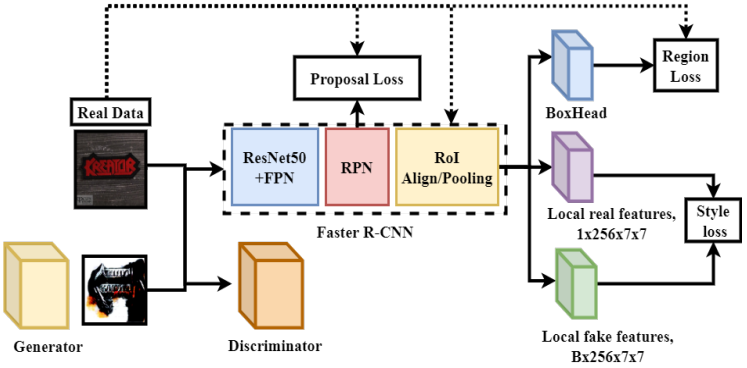

Fig. 2: LL-GAN framework. Normal arrows: features, dotted arrows: box coordinates, broken line box: Faster R-CNN.

Feature loss is computed between B positive RoIs from the fake and the single RoI from the real data (ground truth region). The number of RoIs varies from image to image, but on the average grows as the fake data increasingly resembles the real data.

Each of C feature maps extracted from the real data is vectorized, i.e. an i^{th} feature map is converted into a vector with $H \cdot W = HW$ elements which we refer to as $\mathcal{F}_i^r$. Dot-product is computed between each (i, j) pair of vectorized feature maps to obtain matrix $\mathcal{G}^r$ with dimensionality $C \times C$ (i.e. each (i, j) element in $\mathcal{G}^r$ is a dot product of the vectors $\mathcal{F}_i^r$ and $\mathcal{F}_j^r$), see Equation 1.

$$\mathcal{G}_{i,j}^r = \mathcal{F}_i^r \otimes \mathcal{F}_j^r \tag{1}$$

For each k^{th} RoI extracted from the fake data, we also compute Gram matrix $\mathcal{G}^{k,f}$, Equation 2, where $\mathcal{F}_i^{k,f}$ is an i^{th} vectorized feature map in the k^{th} RoI. Therefore $\mathcal{G}_{i,j}^{k,f}$ is the dot-product between each (i, j) pair of vectorized feature maps in k^{th} RoI, $\mathcal{F}_i^{k,f} \otimes \mathcal{F}_j^{k,f}$.

$$\mathcal{G}_{i,j}^{k,f} = \mathcal{F}_i^{k,f} \otimes \mathcal{F}_j^{k,f} \tag{2}$$

Equations 1 and 2 compute correlation between regional features, which represents the style. The normalized style loss of k^{th} RoI, D_k is computed using L_2 distance between $\mathcal{G}^r$ and $\mathcal{G}^{k,f}$ elementwise, Equation 3. Finally, we sum B normalized RoI losses, Equation 4.

$$D_k = \frac{\sum_{i=1}^{C} \sum_{j=1}^{C} \left(\mathcal{G}_{i,j}^r - \mathcal{G}_{i,j}^{k,f}\right)^2}{(2 \times H \times W)^2} \tag{3}$$

$$L^S = \frac{\sum_{k=1}^{B} D_k}{B} \tag{4}$$

The main idea of computing style loss using Equations 1 - 4 is to train the Generator to evolve features that approximate the distribution of the real logos, and in the same region as in the real data. The first requirement (style) is satisfied by Equations 1 and 2, the second one (spatial awareness) by the RoIAlign functionality: by backpropagating loss extracted from a region in the fake data, Generator learns to evolve region-aware logos. Total loss in this framework is computed using Equation 7.

$$L^D = \mathbf{E}_{x\sim p(x)} \log D(x) + \mathbf{E}_{z\sim p(z)} \log(1 - D(G(z))) \tag{5}$$

$$L^G = \mathbf{E}_{z\sim p(z)} \log D(G(z)) \tag{6}$$

$$L_{Total} = L^G + L^D + L^S \tag{7}$$

Equations 5 and 6 are the usual Discriminator and Generator losses, both computed using binary cross-entropy, for the real data x and fake data z, except that Generator loss maximizes the loss function instead of minimizing it, see Section 4 for details. L^S is the style loss in Equation 4.

Fig. 3: Examples of logos used in the training data overlaid with bounding box and score predictions by Faster R-CNN. Best viewed in color.

3 Dataset construction and labeling

To train LL-GAN models, dataset must have labels consisting of bounding boxes around logos (one box per image). Therefore, dataset construction consists of three stages: first, we scrape the logos from the internet and manually labelled a small portion of it. Next, we train Faster R-CNN on a labelled text and logo ICDAR dataset, to predict boxes around words, and finetuned it to the labelled portion of the metal logo data. Finally, we use this model on the remaining scraped data to label each metal logo with the bounding box.

3.1 Raw dataset

Our real dataset consists of 923 images of varying sizes. Each image contains a heavy metal band's logo, predominantly with a neutral (e.g. black or white) background. This was done in order to prevent the generator from learning

background features and instead focus on the logo style and semantics. Ten bands were selected purely for the style of their logos: Anthrax, Kreator, Manowar, Megadeth, Metallica, Motorhead, Sepultura, Slayer, Slipknot, Sodom. The sizes of images vary between 50x50 and 512x1024 pixels, with the majority about 200x200. Examples with the overlaid bounding boxes are presented in Figure 3. This is a very challenging dataset, for two reasons: it is very small, and it is rich in style (specific styles of heavy metal logos/fonts) and weak in content, because each image contains only a single logo, there's a limited number of observations for each logo. As we explained in Section 2 and show in Section 4, the ability of Faster R-CNN to learn and extract regional features from a single image addresses this challenge.

3.2 Faster R-CNN Logo Detector

To detect boxes around text in logos, we finetuned the out-of-the-box Faster R-CNN model from Torchvision v0.3.0 library with ResNet50 backbone feature extractor and FPN pretrained on MS COCO 2017 to ICDAR Focused Scene Text (ICDAR-FST2013), [13] dataset that contains 223 images of street signs for 100 epochs. This model was trained to detect separate words in various contexts. Next, we fintetuned it for 500 epochs to a portion of the metal logo dataset. The model predicts only two classes (object vs background) per RoI, and we capped the number of candidates in RPN stage at 1024 and also used a slightly larger RPN anchor generator (5 anchor sizes between 16 and 256 and 5 scales, between 0.25 and 2, a total of 25/location), learning rate of $1e-5$, regularization hyperparameter (weight decay) of $1e-2$ and Adam optimizer with $\beta_1 = 0.9, \beta_2 = 0.99$. Other important hyperparameters (positive/negative box thresholds, RoI dimensions, RoI batch size, heads sizes) were the same as in the baseline Torchvision model. First, this model was used to label the rest of the metal logo data for experiments in Section 4.Then, in Section 5, this model was used to detect logos produced by generators in all LL-GAN frameworks and to evaluate the accuracy of outputs of all generators and produce results in Table 4.

4 Experiments

4.1 DCGAN+ framework

We trained both Generator and Discriminator in the DCGAN+ framework from scratch with a learning rate of $1e-4$ and weight regularization coefficient of $1e-3$ for both models using Adam optimizer [17], batch size of 128 and binary cross-entropy loss for 1000 epochs. This took about 6 hours on a GPU with 8Gb VRAM. Following the recommendations in [7] and Pytorch GAN tutorial, Discriminator is updated using real and fake data (1 iteration). Then, the fake data is relabelled as real and the Generator is updated by computing loss using real labels. This is done to avoid premature convergence.

4.2 LL-GAN framework

For LL-GAN we used the pretrained weights and the same architecture for the Generator and Discriminator from DCGAN+. Only Generator and Discriminator were trained, all Faster R-CNN weights trained in Section 3 remained frozen, since the logo detector model was specifically trained to detect single logos anywhere. Real and fake data is processed differently by the logo detector. From the real data, only single RoI regional features with dimensions $C \times H \times W$ is extracted and vectorized, Equation 1, using ground truth bounding box, hence RPN stage is skipped, and no gradients are computed. Fake data is fed forward through the whole framework (see Figure 2), RoI features are extracted and vectorized, Equation 2 for the loss, Equations 3-7 and gradient computation.

Also, RoI module, during processing of fake images, always appends the ground truth bounding box coordinates to the list of RoIs. The reason for that is that early in training, Generator cannot output high-quality logos, and therefore Faster R-CNN will not be able to find good RoIs anywhere in the fake data. As a result, the number of positive RoIs (B in Equation 4) varied from image to image, but overall increased due to the improvement in the work of the Generator. In addition to the baseline LL-GAN framework that uses Equation 7 loss function, we experimented with a number of tricks:

- In addition to style loss in Equation 4, we added detection loss from fake data. Ground truth bounding box coordinates were taken from the real logo that was used to train the Generator. This added two more loss functions: raw boxes in RPN and refined boxes in RoI,

- Extend ground truth bounding boxes around logos to add more context when computing the Generator's loss. We experimented with different values and found 20 pixels in each direction the optimal number for the tradeoff between context and background noise.

- Compute L_2 loss between backbone features extracted from real and fake data, similar to content loss in neural style transfer [5]. Features were taken from all outputs of FPN layers. Therefore, in addition to B RoIs from which we compute L^S, we add the loss from features extracted from the whole image. The objective of adding this loss is to improve the Generator's ability to output a more neutral, e.g. black, background.

- Full model: we combine base model and all three extensions

We trained in total five frameworks (baseline + three augmentations + full model). Each framework was trained for 500 epochs, using Adam optimizer($\beta_1 = 0.9, \beta_2 = 0.999$), regularization parameter (weight decay) of $1e-3$. Hyperparameters of Faster R-CNN logo detector were the same across all frameworks, and shared most of them with the pre-trained logo detector, including the size of the RoIs, $H = 7, W = 7, C = 256$. Since logo generation is a very spatially sensitive

task, we used different thresholds for positive and negative candidates both at RPN and RoIAlign stages: the positive threshold was 0.9 and negative 0.1.

4.3 StyleGAN2

StyleGAN[15] and StyleGAN2[16] are the state-of-the art GANs that can learn different styles and generate high-quality large images, this includes training on small dataset (<5000 images). We trained StyleGAN2 on our data to generate images size 256×256, using high truncation $\psi = 1$ coefficient(no gradient averaging), augment the data by 25%, with the learning rate of $1e{-}4$ for both Generator and Discriminator, Adam optimizer ($\beta_1 = 0.5, \beta_2 = 0.999$), self-attention mechanism [29] and batch size of 4 (maximum possible for this image size on the GPU with 8Gb of VRAM. We trained each model (with and without attention modules) for 100000 steps ($\sim$ 100 epochs), which took about 72 hours, but we noticed that after about 20000 steps the model starts to overfit and exhibits a strong mode collapse. We therefore report the best result for each model (20000 steps for the StyleGAN2 with attention and 15000 for StyleGAN2 without attention).

4.4 Self-Attention GANs

We also train SAGAN, [29], with spectral normalization[20] and Hinge loss function. We used the recommended hyperparameters: latent dimension size 128, batch size of 64, Generator learning rate $1e-4$, Discriminator learning rate $4e-4$ and Adam optimizer ($\beta_1 = 0, \beta_2 = 0.9$). Generator's architecture consists of 7 modules ($ConvTranspose2D + BatchNorm + ReLU$, each equipped with a spectral transformer. Self-attention module is added to block 3 with 256 feature maps and map size of 16×16. The model outputs images size 256×256. SAGAN framework was trained for 300000 iterations ($\sim$ 330 epochs). Training was stopped due to the obvious mode collapse.

5 Evaluation of Results

Examples of outputs of all models are presented in Figure 5. In Table 3 we report FID and IS scores, in Table 4 we report quality and detection results for all models. The best results are bold+italicized, second best bold and third-best italicized. For FID score, we used the layer with 2048 maps, for IS scores we split the sample into either 1 or 10 subsets. Each model generates 512 images which are processed by Faster R-CNN logo detector. If it predicts a logo with confidence score exceeding the pre-defined threshold of 0.75, the detection is considered to be a True Positive (TP), otherwise it is a False Positive (FP). The assumption of this test is that a good Generator would output images that contain exactly single identifiable logo. If the detector predicts more than one logo in a single image with confidence exceeding this threshold, all predictions other than the best-scored one are counted as FPs. If it predicts no logos at all, it is also counted as an FP. Detection rate is defined as $\frac{TP}{TP+FP}$, average confidence is averaged over all detections, including those below the threshold.

5.1 DCGAN+ and LL-GAN

DCGAN+ achieves the best FID score of 220.155, in which it confidently outperforms far more sophisticated state-of-the-art models. It also achieves the third-best results across all other scores. The baseline model is capable of producing high-quality realistic logos in the style of heavy metal bands without overfitting to any particular feature. Among its weaknesses are the inconsistency in glyph stlye, both in terms of color and background noise, see Figures 4 and 5. In particular, some logos are red and yellow and consist of thin vertical lines. Vanilla LL-GAN model achieves the best IS scores of 6.339 and 5.292 and outputs highly detectable logos with high confidence. Most logos generated by the

Table 3: Comparison of models' performance-Quality. Italicized+bold: best, bold: second-best, italicized: third-best

Framework name	FID	IS(1)	IS(10)
DCGAN+	***220.155***	*6.023*	*5.105*
LL-GAN	**223.948**	***6.339***	***5.292***
+ FRCNN loss	271.030	5.705	4.947
+ extended boxes	247.181	5.753	4.901
+ backbone features	*237.752*	4.590	4.095
full	249.694	**6.232**	**5.150**
StyleGAN2 ($\psi = 0.6$)	329.026	2.840	2.766
StyleGAN2 ($\psi = 1.0$)	354.873	2.497	2.433
+attention	328.859	2.356	2.298
SAGAN	283.554	3.581	3.394

vanilla model are very realistic, resemble real glyphs, are consistent in colors (mostly red and white, as in the training data), and do not experience mode collapse. Also LL-GAN with all three augmentations perform well, producing IS scores of 6.232 and 5.150. In Figure 4 we placed outputs from DCGAN+ and different LL-GAN models that output logos with similar features side-by-side to highlight the advantages of our approach. The same features produced by LL-GAN generators are more homogeneous in color and shape, the background contains fewer geometric artefacts and is more consistent and neutral. Metrics discussed in this section confirm that this consistency does not come at the cost of lower variance in the output.

5.2 State-of-the-art models

StyleGAN2 is capable of producing logos with very consistent structures, but due to the size of the dataset suffers from mode collapse. This is reflected in the highest detection score of 0.687 and low FID and IS scores: the generated structures are consistent enough to be classified as a logo, but do not resemble the training data and are very similar. SAGAN also suffers from mode collapse.

By comparing results in Tables 3 and 4 and Figure 5 to the models' architectures and sizes in Table 1, LL-GAN models are comparable in size to StyleGAN2, but their Generators output more interesting logos.

Table 4: Comparison of models' performance-Detection. Italicized+bold: best, bold: second-best, italicized: third-best

Framework name	Detection Rate	AvgConf
DCGAN+	*0.670*	*0.739*
LL-GAN	**0.674**	**0.746**
+ FRCNN loss	0.640	***0.827***
+ extended boxes	0.666	0.707
+ backbone features	0.622	0.701
full	0.590	0.638
StyleGAN2($\psi = 0.6$)	0.554	0.670
StyleGAN2($\psi = 1.0$)	***0.687***	0.684
+attention	0.578	0.569
SAGAN	0.561	0.600

6 Conclusion

Generation of logos is a challenging problem that is becoming increasingly more popular in deep learning community. In this paper we presented a novel framework that fuses Faster R-CNN and GANs for generating large (282x282) heavy metal logos. The model was trained on a small style-rich dataset of real-life band logos. Results achieved by LL-GAN confidently outperform the state-of-the-art models trained on the same dataset, and we intend to explore the capacity of Faster R-CNN detector to extract and learn from regional features further. The advantages of our approach include:

- The novel idea of training the Generator using losses extracted from regional features in the real and fake data using Faster R-CNN.
- Computation of the style loss (Gram matrix) on regional features. This allows to use correlation between features in the fake and real data to transfer style from real to fake data, and construct samples from every image.
- The use of bounding boxes to determine the size of the RoIs in the fake data. Changing this size can improve results, e.g. by creating a more stable background.

Also, we would like to address certain limitations of the presented solution:

- Dataset and scope. All models were trained on a small dataset collected specifically to create logos in a particular style. We are confident this approach can be scaled to more general problems (e.g. logo stylization, style

Fig. 4: Comparison of DCGAN+ (left) and LL-GAN output (right). First row: DCGAN+ vs LL-GAN, second row: DCGAN+ vs LL-GAN(+backbone features), third row: DCGAN+ vs LL-GAN (full), fourth row: DCGAN+ vs LL-GAN(+FRCNN losses). The obvious weakness of DCGAN+ that LL-GAN fixes is the lack of shape (glyphs are made up of thicker, shorter features without gaps) and color (all glyphs in the logo have the same color) consistency. Each row used the same Generator input. Best viewed in color.

(a) DCGAN+

(b) LL-GAN

(c) LL-GAN + extended boxes

(d) LL-GAN + Faster R-CNN loss

(e) LL-GAN + backbone features

(f) LL-GAN (full)

(g) StyleGAN2 ($\psi = 1$)

(h) StyleGAN2 ($\psi = 1$) + attention module

(i) SAGAN

Fig. 5: Examples generated by the models presented in the paper overlaid with bounding boxes predicted by the Faster R-CNN logo detection (+confidence score). Three last images for StyleGAN2 and StyleGAN2+Attention models were obtained using mixing regularities, see [16] for details. All DCGAN+ and LL-GAN images are 282×282, all other models are 256×256. Best viewed in color.

transfer, conditional logo creation) and larger datasets.

– Disentaglement and fusion of style and content. Disentanglement of style from content is active area of research in the font generation community[4, 3]. In this paper we only used a single Generator for the logo generation. This result can be improved both by augmenting the architectures, and fusing the style and content datasets.

References

1. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein gan. arXiv preprint arXiv:1701.07875 (2017)
2. Atarsaikhan, G., Iwana, B.K., Uchida, S.: Contained neural style transfer for decorated logo generation. In: 2018 13th IAPR International Workshop on Document Analysis Systems (DAS). pp. 317–322. IEEE (2018)
3. Azadi, S., Fisher, M., Kim, V.G., Wang, Z., Shechtman, E., Darrell, T.: Multi-content gan for few-shot font style transfer. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7564–7573 (2018)
4. Gao, Y., Guo, Y., Lian, Z., Tang, Y., Xiao, J.: Artistic glyph image synthesis via one-stage few-shot learning. ACM Transactions on Graphics (TOG) **38**(6), 1–12 (2019)
5. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2414–2423 (2016)
6. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 580–587 (2014)
7. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014)
8. Hayashi, H., Abe, K., Uchida, S.: Glyphgan: Style-consistent font generation based on generative adversarial networks. arXiv preprint arXiv:1905.12502 (2019)
9. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 2961–2969 (2017)
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
11. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: Advances in neural information processing systems. pp. 6626–6637 (2017)
12. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1125–1134 (2017)
13. Karatzas, D., Shafait, F., Uchida, S., Iwamura, M., i Bigorda, L.G., Mestre, S.R., Mas, J., Mota, D.F., Almazan, J.A., De Las Heras, L.P.: Icdar 2013 robust reading competition. In: 2013 12th International Conference on Document Analysis and Recognition. pp. 1484–1493. IEEE (2013)
14. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196 (2017)

15. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4401–4410 (2019)
16. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8110–8119 (2020)
17. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
18. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2117–2125 (2017)
19. Mino, A., Spanakis, G.: Logan: Generating logos with a generative adversarial neural network conditioned on color. In: 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA). pp. 965–970. IEEE (2018)
20. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. arXiv preprint arXiv:1802.05957 (2018)
21. Oeldorf, C., Spanakis, G.: Loganv2: Conditional style-based logo generation with generative adversarial networks. arXiv preprint arXiv:1909.09974 (2019)
22. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)
23. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 779–788 (2016)
24. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)
25. Rijken, G.J., Cutura, R., Heyen, F., Sedlmair, M., Correll, M., Dykes, J., Smit, N.: Illegible semantics: Exploring the design space of metal logos. arXiv preprint arXiv:2109.01688 (2021)
26. Sage, A., Agustsson, E., Timofte, R., Van Gool, L.: Logo synthesis and manipulation with clustered generative adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5879–5888 (2018)
27. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans. In: Advances in neural information processing systems. pp. 2234–2242 (2016)
28. Ter-Sarkisov, A.: Network of steel: Neural font style transfer from heavy metal to corporate logos. arXiv preprint arXiv:2001.03659 (2020)
29. Zhang, H., Goodfellow, I., Metaxas, D., Odena, A.: Self-attention generative adversarial networks. In: International Conference on Machine Learning. pp. 7354–7363 (2019)
30. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE international conference on computer vision. pp. 2223–2232 (2017)