Style GAN condioned on Input Label

LoGANv2: Conditional Style-Based Logo Generation with Generative Adversarial Networks

Cedric Oeldorf
Department of Data Science and Knowledge Engineering
Maastricht University
Maastricht, The Netherlands
Email: cedric.oeldorf@gmail.com

Gerasimos Spanakis
Department of Data Science and Knowledge Engineering
Maastricht University
Maastricht, The Netherlands
Email: jerry.spanakis@maastrichtuniversity.nl

***Abstract*—Domains such as logo synthesis, in which the data has a high degree of multi-modality, still pose a challenge for generative adversarial networks (GANs). Recent research shows that progressive training (ProGAN) and mapping network extensions (StyleGAN) enable both increased training stability for higher dimensional problems and better feature separation within the embedded latent space. However, these architectures leave limited control over shaping the output of the network, which is an undesirable trait in the case of logo synthesis. This paper explores a conditional extension to the StyleGAN architecture with the aim of firstly, improving on the low resolution results of previous research and, secondly, increasing the controllability of the output through the use of synthetic class-conditions. Furthermore, methods of extracting such class conditions are explored with a focus on the human interpretability, where the challenge lies in the fact that, by nature, visual logo characteristics are hard to define. The introduced conditional style-based generator architecture is trained on the extracted class-conditions in two experiments and studied relative to the performance of an unconditional model. Results show that, whilst the unconditional model more closely matches the training distribution, high quality conditions enabled the embedding of finer details onto the latent space, leading to more diverse output.**

1. Introduction

Since their inception, Generative Adversarial Networks (GANs) showed the way to a whole new generation of neural network (NN) applications [1]. By enabling NN-based approaches to tasks that traditionally require human-sourced creativity, we have seen many incredibly interesting uses such as generating images from text [2], composing music [3], fashion design [4], illustrating animation characters [5] and even creating emojis from peoples faces [6].

Tackling a creative domain such as the illustrative design process can provide creative professionals with tools that both assist and augment their work. One such domain requiring vast amounts of creativity is that of brand/logo design. With most Americans exposed to 4,000 to 20,000 advertisements a day [1], companies are paying ever increasing attention to their branding. This puts pressure on designers to come up with aesthetic yet innovative and unique designs in an attempt to set their designs apart from the masses.

GANs could assist designers by either providing them with inspiration or by reducing the number of design iterations undergone with clients. A problem with GAN generated content is that samples are created from an unknown noise distribution known as the input latent code. In order to facilitate specification based content generation, the user must be able to shape this latent input code in such a way that it allows for an intuitive determination of the output's style.

The implementation of class conditions, which guide the network to produce output associated with the features of a specific class, could serve as a possible approach to this. One challenge lies in the fact that such a model would require logos to be sectioned into classes based on their visual characteristics whilst logo characteristics are hard to define.

Furthermore, the high degree of multi-modality of logos [7] increases the complexity of the task. Whilst being a problem that has been tackled in previous research [7], [8] which showed somewhat stable 32×32 pixel results, such a small resolution is undesirable for real use as most details are not identifiable. Considering that model complexity exponentially increases with image size, another challenge involves generating higher resolution and more detailed logos.

With the various aspects of the problem in mind, we engage with the following questions throughout this research paper:

- How can we extract high-quality and easily definable conditions from logos?
- Can we increase input/output image resolution whilst retaining stable training and sensible results?
- How does enforcing conditions on a GAN affect performance on multi-modal data?

1. https://www.forbes.com/sites/forbesagencycouncil/2017/08/25/finding-brand-success-in-the-digital-world/

Main Idea: Modify StyleGAN to Conditional StyleGAN and explore on the low resolution for LOGO image generation.

Both our data and python implementation are available via GitHub [2].

2. Background and Related Work

First experiments involved conditional GANs for logo synthesis and manipulation [7]. Results showed that including conditioning labels as input to a GAN has a positive effect on training stability as it promotes feature disentanglement, given that the synthetic labels are meaningful. State-of-the-art performance was achieved in terms of quantitative performance metrics by their Wasserstein GAN (WGAN) extended with an auxiliary classifier (WGAN-AC). However, they concluded that results looked more appealing in terms of human judgment when produced by their layer conditional deep convolutional GAN (DCGAN-LC).

Shortly thereafter, notable results were achieved through the use of colour as a condition within a WGAN-AC using gradient penalty [8]. Albeit achieving promising results, the paper confirmed that the conditions need to convey meaningful information that can guide the logo in feature disentanglement, else the generator sets back to a random state.

2.1. Progressive Training

The issue of GAN stability is tackled by adjusting the training methodology to start training on low resolution images and add higher resolution layers as training progresses [9]. Considering that low resolution images hold fewer modes and less class information [10], the network is enabled to learn large-scale patterns in the data and then pivot from learning coarse to progressively more fine detail as resolution increases [9]. Not only does this stabilize training, but also speeds it up through the fact that the network does not need to tackle the task of immediately mapping the intermediate latent space to high resolution imagery [9].

An overview of such a methodology can be seen in Figure 1. Training of both the generator G and the discriminator D starts at a low resolution of 4×4 pixels, but increases as training advances. In order to avoid a "shock" to the network when adding new layers, a transition phase outlined in the center of Figure 1 is implemented. During the transition phase, skip connections are introduced to assist the pass-over to a higher resolution. After a few iterations these connections are removed and training continues. It is important to note that no layers are frozen during the process, with the entire network remaining trainable throughout the process.

Whilst progressive training solves GAN stability for high resolution training to a large extent, the embedding of a complex data distribution onto a (relatively) small latent space results in unavoidable feature entanglement. That is, multiple features are mapped to single positions in the space. Such entanglement leads to significantly difficulty in controlling the output of the network, which, if we refer

2. https://github.com/cedricoeldorf/ConditionalStyleGAN

ProGAN concept:

Progressive growing. See mine Annotated ProGAN paper for concept.

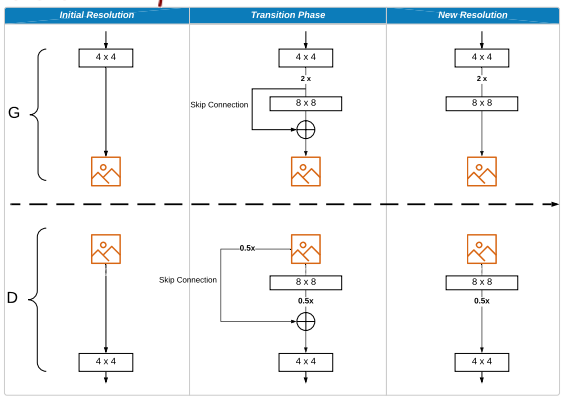


Figure 1. Progressive growing of the GAN architecture involves the addition of a higher resolution layers via a transition phase.

In this paper, they pass labels with noise z.

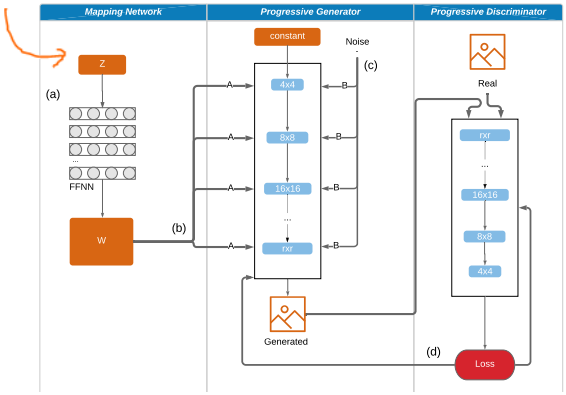


Figure 2. An outline of StyleGAN architectural elements. The mapping network transforms the initial latent vector z into w by feeding it through a feed-forward neural network (FFNN). The progressively grown generator takes both latent vector and noise inputs at all layers. The discriminator grows with the generator and returns a loss metric which is backpropagated to both networks.

to our problem statement, is something we would like to achieve. This brings us to an extension of this architecture presented in the following section.

2.2. Style-Based Generator

By combining progressive training, a mapping network and adaptive instance normalization, feature entanglement as mentioned in Section 2.1 can be combated with the StyleGAN architecture [11].

We outline the upcoming subsections by exploring the alphabetical markings on Figure 2. Marked as "(a)" in the figure, we start by describing the mapping network, which provides the input to the generator and serves as the baseline combatant to feature disentanglement. This is followed by marking "(b)", which describes how the output of the mapping network is fed into the progressive generator and "(c)" explores how stochastic variation can be introduced

Read annotated StyleGAN paper for more concept.

to the model. Marking "(d)" lies close to the loss function, which we explore for our specific case in Section 4.

2.2.1. Mapping Network. In order to gain independent control over individual low- to high-level features, a feed-forward neural network can be used as a mapping network. As opposed to feeding a random vector straight into the generator, the input is projected onto an intermediate latent space w by being fed through a mapping network [11]. This latent space allows controls of "styles" within convolutional layers at each resolution using Adaptive Instance Normalization explored in the next subsection .

2.2.2. Adaptive Instance Normalization. Traditionally in stabilization methods, each convolutional layer is accompanied by batch normalization [12], which eases training complexity. However, significant improvement concerning style transfer results was observed when instance normalization (IN) was first introduced [13].

Instance Normalization

$$IN(x) = \gamma(\frac{x - \mu(x)}{\sigma(x)}) + \beta \tag{1}$$

IN is seen in Equation 1, where $\mu(x)$ and $\sigma(x)$ represent scale and bias. It normalizes input with respect to a certain style which is defined by the parameters γ and β. In order to allow this equation to adapt to any given style, an adaptive affine transformation can be used.

Adaptive Instance Normalization

$$AdaIN(x, y) = \sigma(y)(\frac{x - \mu(x)}{\sigma(x)}) + \mu(y) \tag{2}$$

Adaptive instance normalization (AdaIN), as shown in the above equation, takes an additional input of style y in order to replace the two parameters of IN with a scale $\sigma(y)$ and a bias $\mu(y)$ [14].

In context of the mapping network in section 2.2.1, the intermediate space w is transformed into the scale and bias of the AdaIN equation. Noting that w reflects the style, AdaIN, through it's normalization process, defines the importance of individual parameters in the convolutional layers. Thus, through AdaIN, the feature map is translated into a visual representation.

Achieve through the noise injection.

2.2.3. Stochastic Variation. As features are controlled by the mapping network and AdaIN, it was found that a random noise vector is not needed as initial input, but can be replaced by a constant [11]. However, in order to introduce stochastic variation into the model, which is traditionally done by inserting noise into the random input vector, it was proposed to add noise to each channel of the individual convolutional layers as seen in Figure 3 [11].

The noise inputs take the form of two-dimensional matrices sampled from a Gaussian distribution. These are then scaled to match the dimensions within the layer and applied to each channel [11]. This introduces variation within that feature space.

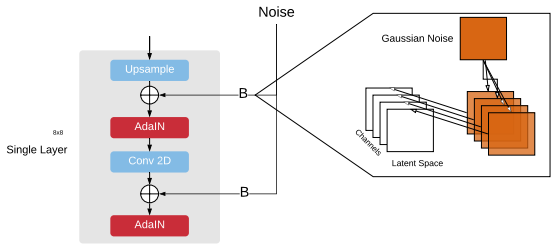


Figure 3. Inserting Noise into the Generator

2.2.4. Drawback: Limited control. In the scenario such as wanting to generate a specific type of logo for a client, we would want to give generator a specification by which it should synthesize a logo. Although the addition of the mapping network allows for increased control of style/features through W, this space is is quite large and likely imperfect. Each value of W would need to be mapped to a feature in order to allow for human control of the synthesized logo.

Why StyleGAN not able to generate a specific LOGO specified as input?

As a goal in this paper is to be able to generate logos based by some form of input, a possible solution to this problem is introduced as the model used in this this work in Chapter 4.

3. Data Preprocessing and Label Extraction

This section introduces both data and methods used to extract meaningful class-conditions from logos. There are three stages, data preprocessing in which the logos are cleaned and prepared, feature extraction in which we project the logos into a quantifiable space and clustering in which K-means clustering applied to the mentioned space.

3.1. BoostedLLD

The data used to train the models for our experiments is based off of the LLD-logo [7] dataset, which was preprocessed and boosted with additional images in order to be a more precise fit for our problem domain.

3.1.1. Large Logo Data. First introduced by Sage et al. (2018), the large logo dataset (LLD) comes in two forms, LLD-icon and LLD-logo. The icon version consists of 32px favicons, whereas the logo version consists of up to 400px logos scraped from twitter. Whilst previous research in logo synthesis [7], [8] made use of or focused on the icons set, we opted for the logo version as the higher resolution is in line with our research goals.

Whilst the LLD-logo set consists of 122,920 logos, many of these logos consist purely of text as seen in Figure 8.

Figure 4. Examples of text-based logos

As we would like to shift the focus of our model away from fonts and text, we decided to drop all text-based logos using the tesseract [15] open-source optical character recognition model. After dropping all images that had identifiable text on them, we were left with circa 40 000 logos.

3.1.2. Boosting. With the aim of diversifying and extending the remaining data after the preprocessing of the LLD-logo set, we scraped Google for new logo-like images. Keywords pertaining to topics such as nature, technology and illustrated characters were used in what resulted in circa 15 000 additional images for our training data. From the examples in Figure 5 we see that, whilst these images aren't official logos, they do carry the desired features of potential logos.

Figure 5. Examples of logos from boosting dataset

3.2. Label Extraction

In order to train our model in a conditional manner, we need class labels for our data. These can be extracted through clustering-based methods, of which the two we used are outlined below.

3.2.1. Object-Classification-Based Clustering. With one of our research goals being the extraction of human-interpretable characteristics from logos, we opted for a clustering method that makes use of Google Cloud Vision in the following three steps:

- Step 1: Using the Google Vision API[3], each logo is given 4 to 8 word labels that describe the contents. .
- Step 2: In order to bring the word labels into a quantitative space, we use a a pre-trained Word2Vec model[4] [16]. Having multiple words per logo, the midpoint between the word vectors of each individual logo is calculated and serves as the spatial representation for each logo.
- Step 3: K-means clustering [17] is used to partition the images into segments with the aim of having these represent unique visual properties within the logos. It is clear that, whilst there is some separation of visual characteristics, many logo styles appear in multiple clusters and it is not immediately clear what a cluster should represent.

The poor separation of logos is what brings us to our second clustering approach in the following section.

3. https://cloud.google.com/vision/
4. https://code.google.com/archive/p/word2vec/

Steps used to clustered objects around the different labels that is used to train the conditional style gan.

They explained two clustering mechanism to cluster the scraped logos with labels. So, that that labels used to train conditional StyleGAN generator which generates output based on input labels. Think as conditional GAN.

Main Diff to StylGAN is to introduce condn labels with noise and pass to mapping network to generate W.

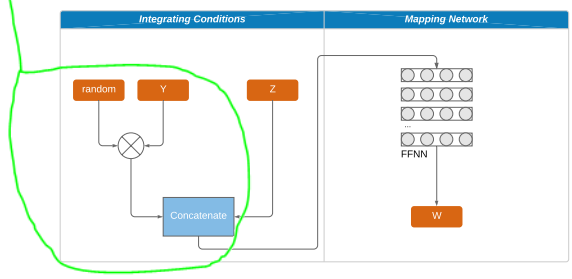


Figure 6. Producing a conditional latent vector for the StyleGAN architecture

3.2.2. ResNet Embedding Layer. Our second clustering approach follows the same steps as the first but with a different labelling technique. It makes use of the last max pooling layer of a pre-trained VGG16 [18] network in order to project input images into a 512-dimensional feature space. Samples taken from the clusters show that certain clusters represent certain visual characteristics, more so than in section 3.2.1.

Main Concept

4. Conditional Style-Based Generator

Recalling from Section 2.2.4, the unconditional StyleGAN architecture only allows for limited control over the produced output. Taking a step towards lifting this limitation, we introduce multiple extensions to the StyleGAN architecture which when combined turn it into a class-conditional model.

Such conditions would not only allow the specification of a style preceding output generation, but might also assist the model in learning a larger number of complex modes [7]. Considering that many related research efforts make use of very structured image domains where images across the distribution share, for example, locality of landmarks [11], [19], we pay special attention to easing learning within the by nature highly multi-modal logo domain.

4.1. Architecture

Motivated by the drawbacks in Section 2, we propose using a conditional framework for the StyleGAN architecture. There are two major differences between the conditional and unconditional StyleGAN architectures: (a) integrating class-conditions into intermediate latent space and (b) updating the loss function to take class-conditions into consideration.

Think of as GAN-> Conditional GAN

4.1.1. Intermediate Latent Space. Recalling that in the StyleGAN architecture, a random vector is fed through a mapping network, which in turn produces the input vector for the generator. The incorporation of conditions into this process takes place just before the mapping network as outlined in Figure 6.

Condn with input to form intermediate latent space, W

We can see how the latent embedding W depends on both the random input and the class-conditions. The

Idea is: Integrate One-hot encoded Label with the noise Z and pass to the mapping network to form intermediate latent vector W which contains label information.

process is run in mini-batches of size n, which is how we define the process mathematically below.

We introduce two matrices in Equation 3. Firstly, $Z \sim \mathcal{N}(\mu, \sigma^2)$, which is a matrix comprised of the initial latent space of size $n \times d$, where n is the size of the mini-batch and d is the length of the matrix. Secondly, Y, which is a matrix of the one-hot-encoded conditions of size $n \times c$, where c is the number of classes we introduce.

$$Z = \begin{bmatrix} l_{11} & l_{11} & \cdots & l_{1d} \\ l_{21} & & & \\ \vdots & & & \\ l_{n1} & l_{n2} & \cdots & l_{nd} \end{bmatrix}; Y = \begin{bmatrix} y_{11} & y_{11} & \cdots & y_{1c} \\ y_{21} & & & \\ \vdots & & & \\ y_{n1} & y_{n2} & \cdots & y_{nc} \end{bmatrix} \tag{3}$$

W, which is the matrix holding the final latent input vectors, is calculated in Equation 4. Y is multiplied with matrix $R \sim \mathcal{N}(\mu, \sigma^2)$ of size $n \times c$ and concatenating the result with Z. Figure 6 clarifies it.

$$W = f([Y \times R]^\frown Z) \tag{4}$$

R is random vector of normal distribution. The feed-forward neural network, which has also been defined as out mapping network, is represented as f in Equation 4. The result, W ends up having the same dimensions as L.

4.1.2. Updated Loss Function. Updating the WGAN loss function, the used loss function shows the discriminator considering the conditioning labels paired with respective output/training data [20]. Equation 5 represents this updated loss, where y are the class-conditions.

$$\nabla_{\theta D} \left[f_{\theta D}(x|y) - f_{\theta D}(G(w|y)) \right] \tag{5}$$

The architecture makes use of a gradient penalty term. Combining the term with equation 5, we are left with a conditional WGAN-GP loss function as seen in equation 6 below:

Loss of WGANGP with conditional input

$$\nabla_{\theta D} \underbrace{[f_{\theta D}(x|y) - f_{\theta D}(G(w|y))]}_{\text{Discriminator Loss}} + \underbrace{\lambda[(|| \nabla_{\hat{x}} D(\hat{x}|y)||_2 - 1)^2]}_{\text{Gradient Penalty}} \tag{6}$$

Where $\hat{x}$ is uniformly sampled along straight lines between pairs of real and generated data points [21].

5. Experimental Design & Results

The architecture undergoes three core experiments.

- Experiment 1: StyleGAN architecture (no conditions)
- Experiment 2: StyleGAN conditioned on object-classification-based labels
- Experiment 3: StyleGAN conditioned on ResNet-feature-based labels

These experiments and their evaluation aim to tackle the posed research questions of whether we can generate higher resolution logos and how both the low- and high-quality conditions extracted in Section 3 affect model performance. We first describe each experiment and their respective model outputs and then go deeper into the results analysis by performing both a qualitative and quantitative evaluation in which all models are compared. The major traits we are looking for are: image quality, diversity and homogeneity within a class-condition. Image quality and diversity are explored both qualitatively and quantitatively. Homogeneity within a class-condition will be explored visually and only applies to experiment 2 and 3.

5.1. Quantitative Evaluation

Based on Inception Score and Frechet Inception Score

Using a quantitative measure for analyzing the performance of a GAN is especially difficult for a domain such as logo synthesis, as the quality of a logo is a subjective metric. The inception score makes use of Google's Inception classifier [22] in order to measure the diversity of objects within generated images. If generated images hold a diverse set of recognizable objects, the inception score will be high.

Inception Score

$$IS(G) = exp(E_{x \sim p_g} D_{KL}(P(y|x)||p(y))) \tag{7}$$

Mathematically, the score is computed as the exponential of the KL-divergence D_{KL} between distributions $p(y|x)$ and $p(y)$.

Whilst this was shown to correlate well with subjective human judgment of image quality [13], the scores outcome relies heavily on objects being present in the data.

This score was extended with the proposed Frechet Inception Distance [23], which has been prominently used in many of the latest GAN related papers [7], [9], [11]. The difference lies in that the distribution statistics are extracted from the feature space of an intermediate layer as opposed to from the output layer. Mathematically it can be described as:

$$FID(x, g) = ||\mu_x - \mu_g||_2^2 + Tr(\sum_x + \sum_g - 2(\sum_x \sum_g)^{1/2}) \tag{8}$$

where μ represents the mean and $\sum$ the covariance of the embedded layer of both x, the training data, and g, the generated data. We want to minimize this distance, thus the smaller FID, the higher the quality of generated images. The fact that the FID does not make use of object classification, makes it a good score for this paper, as logos often by nature do not contain classifiable objects.

Taking the elements in equation 8 into account, we see the calculated score will represent both how similar the logos are to the training data in terms of quality, but also in terms of produced diversity captured by the covariance.

	Frechet Inception Distance
Experiment 1 (no conditions)	47.4694
Experiment 2 (object label conditions)	71.6898
Experiment 3 (ResNet feature conditions)	101.9211

TABLE 1. FID SCORES FOR EVERY EXPERIMENT. A LOW FID SCORE IN OUR CASE IMPLIES THAT THE SYNTHESIZED LOGOS AND REAL LOGOS CARRY CLOSELY-RELATED VISUAL FEATURES.

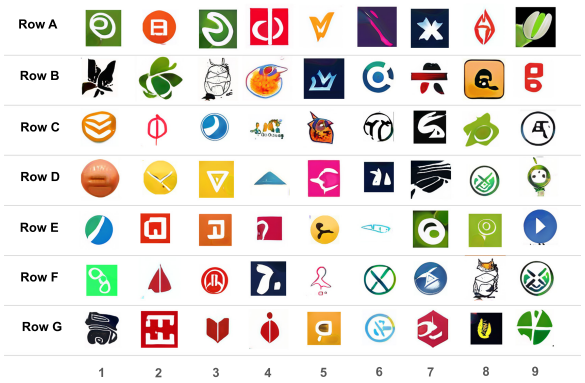


Figure 7. Experiment 1: Synthesized logos

5.2. Qualitative Evaluation

In order to subjectively measure the quality of the results, we not only view the raw results, but also analyze how well the latent embedding captured the training distribution. This is done through the truncation trick introduced below.

5.2.1. Truncation Trick. Within the distribution $p(x)$ of the training data, some areas of the sampling space can be of low density due to lack of representation in the training data [11]. Previous papers [24] have shown that sampling from a truncated space can improve the quality of generated results. Mathematically, this is achieved by calculating the center of mass in our latent vector W as:

$$\bar{w} = E_{z \sim P(z)}[f(z)] \tag{9}$$

this point represents a type of mean generated image of the learned distribution. By increasing ψ in equation reftruncation2, we specify how far we would like to deviate from this center of mass. The higher the value, the more we move towards the fringes of the distribution.

$$w' = \bar{w} + \psi(w - \bar{w}) \tag{10}$$

5.3. Experiment 1: No Conditions

In order to set a baseline with which to compare our conditional models with, a model was trained that does not make use of any conditions. This represents the StyleGAN architecture as it was presented in the paper it was proposed in [11]. Synthesized logos are presented in Figure 7.

Image quality Produced logos appear to be stable and consistently of high quality. The majority of the designs

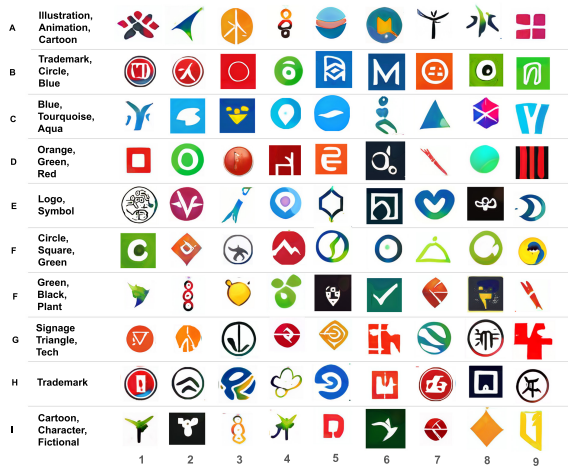


Figure 8. Experiment 2: Synthesized Logos

are very simple, with logos such as in B1 and B2 being rare. This could be an indication that some modes were not embedded on the latent space. According to the FID in Table 5.3, logos produced in experiment 1 are shown to be closest to matching the distribution of the training data.

Diversity We see many shape and colour repetitions with minor tweaks. However, a few truly unique designs are found such as the pattern in G1 and the character-like being in D10 of Figure 7. Additionally, when comparing this experiments behavior over different truncation values in Figure 10, we see that the fringes of the learned embedding aren't very stable, implying that some low density modes were dropped. Based on this we can gather that output was "conservative" but in line with the models goals.

5.4. Experiment 2: Object Classification Based Conditions

For this experiment, conditions based on the vectorized and clustered object classification labels were used. As was pointed out in Section 3.2.1, these conditions do not show much visual separation and often embody similar styles across multiple conditions. The model produced subjectively good-looking logos as seen in Figure 8. However, it did suffer from slow learning, which might be indicative of low-quality conditions acting as a sort of regularizer.

Image quality Many examples are not as clearly defined nor as sharp as in the previous case. Occasionally, such as in B2,E6 and F7, we find logos with fine very fine details, which is indicative of the model having embedded a few complexities of the distribution. However, such fine details quickly seem to form nonsensical output such as E1, and most logos follow a general square or circle trend. Considering that the FID score in Table 5.3 is much higher, we gather that the images do not match the training distribution well, however, through this the generated results are fairly unique.

Figure 9. Experiment 3: Synthesized Logos

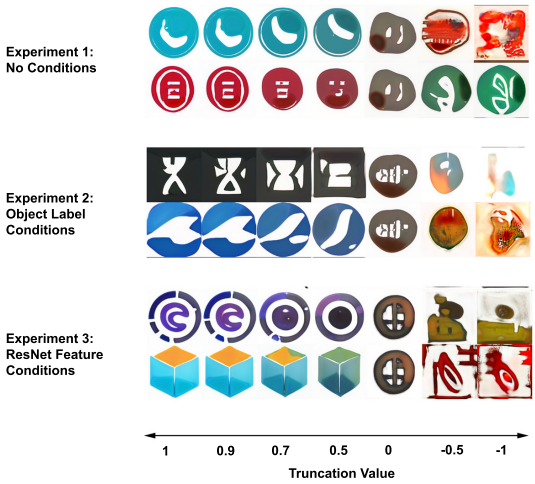


Figure 10. Experimental results over different truncation values

Diversity Whilst we do find a lot of repetition such as in G8 and H9, the seem to generally be quite diverse compared to experiment 1. A possible reason for this is that through the conditions, the embedding of the complexities within the training data distribution becomes simpler.

Homogeneity The lack of visual separation within the conditional classes of this experiment expectantly resulted in almost no homogeneity within the conditions. We do see weak trends such as "blue" in row C, leading us to believe that with improved class conditions we would see strong visual representation of these in each row.

5.5. Experiment 3: ResNet Feature Conditions

The third experiment made use of conditions based on the clustered output of an embedded ResNet layer. These conditions were of significant higher quality compared to those of Experiment 2, as pointed out in Section 3.2.2. The high quality labels seem to support the models learning by feeding it meaningful information. We display synthesized logos in Figure 9.

Image Quality Whilst it is surely the most creative of the three models described, the logos produced in this experiment see both clearly defined, high-quality examples (e.g. C5,C8 or D9), but also incoherent shapes that don't seem to resemble anything we see in the training data (e.g. F7 or H1). If we take into account that the FID of experiment 3 is more than double that of experiment 1, it can be deduced that finer output detail leads to quick divergence from the training distribution. Unfortunately the FID metric cannot take the visual creativity of the output into account, a property at which in our opinion experiment 3 excelled.

Furthermore, in Figure 10, experiment 3 clearly retains the highest image quality of the three models at a truncation value of 1. This is in line with our expectations, where the high-quality conditions enabled the model to better learn the complexities of the training data distribution with respect to each class. This is furthermore confirmed by looking at a truncation value of 0, which represents the "average" logo produced by each network. Experiment 3's average holds arguably finer detail compared to the others.

Diversity Even within each condition, the model consistently produces a diverse set of examples. This is indicative of the model being capable of learning even the high level features of the training data distribution.

Homogeneity As expected, using visually coherent class-conditions shows a high magnitude of homogeneity of style within the conditions. We derive from this that the conditions aided style separation within the latent space, allowing for the mapping of specific conditions to features commonly found within each.

6. Conclusion

In this paper, we experimented with a conditional extension to the popular StyleGAN architecture in order to control the output of logo generation. We showed that meaningful conditions can help the model capture modes that might not be captured by an unconditional model. High-quality conditions inserted into the latent code proved to be a viable approach to controlling the output of a synthesis network.

We found that extracting easily definable classes from logos poses a challenge. Whilst the method involving the clustered output of ResNet layer embedding provided us with visually distinct groupings, we saw within the object-classification based method that creating automated language-labels does not result in descriptions that define the visual characteristics of logos. Moreover, both conditional and unconditional StyleGAN architectures enabled stable training for a resolution 4 times larger compared to previous research. We attribute this outcome to the fact

that progressive training simplifies the distributions to be embedded on the latent code. Furthermore, experiments have shown that the introduction of class-conditions in the model enables it to learn a larger number of modes in the context of highly multi-modal data. It became apparent that the meaningful separation of data into classes eases learning for the model. Having been fed high-quality information, the model is faster to shift its focus onto high level features, resulting in much more detailed logo synthesis. Lastly, a trade-off between the robustness and detail of generated output was observed. The conditional models were shown to produce more unique logos with fine features, but at the cost of also producing some nonsensical output. The unconditional model however, which generated mostly very simple but realistic logos, did not see the same amount of nonsensical output. We additionally proposed a new data set that shifts the focus of the LLD collection away from text-based designs and more towards illustrations. Although the size of the data set was significantly reduced, we believe that the higher quality played a significant factor in the success of our models.

In future, it would be desirable to head towards a text-input based architecture, especially when considering the use-cases of the logo domain. Keeping our approach in mind, a possible scenario might include replacing the class-conditions input with word vector representations. However, whilst the challenge of producing meaningful word labels from logos still goes unsolved, first word-based logo generation experiments could be run using easy-to-identify characteristics such as colour and shape. If successful, this would offer a very simple and straightforward way of controlling the latent embedding, and with that the architecture's output.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[2] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," *arXiv preprint arXiv:1605.05396*, 2016.

[3] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "Midinet: A convolutional generative adversarial network for symbolic-domain music generation," *arXiv preprint arXiv:1703.10847*, 2017.

[4] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1857–1865.

[5] Y. Jin, J. Zhang, M. Li, Y. Tian, H. Zhu, and Z. Fang, "Towards the automatic anime characters creation with generative adversarial networks," *arXiv preprint arXiv:1708.05509*, 2017.

[6] Y. Taigman, A. Polyak, and L. Wolf, "Unsupervised cross-domain image generation," *arXiv preprint arXiv:1611.02200*, 2016.

[7] A. Sage, E. Agustsson, R. Timofte, and L. Van Gool, "Logo synthesis and manipulation with clustered generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5879–5888.

[8] A. Mino and G. Spanakis, "Logan: Generating logos with a generative adversarial neural network conditioned on color," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2018, pp. 965–970.

[9] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[10] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2642–2651.

[11] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," *arXiv preprint arXiv:1812.04948*, 2018.

[12] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[13] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6924–6932.

[14] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1501–1510.

[15] R. Smith, "An overview of the tesseract ocr engine," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2. IEEE, 2007, pp. 629–633.

[16] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[17] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[18] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[19] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[20] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[21] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems*, 2017, pp. 5767–5777.

[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[23] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in Neural Information Processing Systems*, 2017, pp. 6626–6637.

[24] M. Marchesi, "Megapixel size image creation using generative adversarial networks," *arXiv preprint arXiv:1706.00082*, 2017.

LoGANv2 Summary

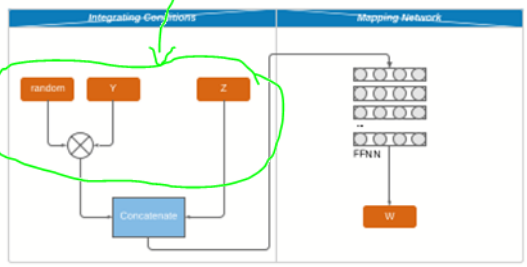
Prepared By: Sushant Gautam
Part of 30 Day GAN paper Reading
https://github.com/sushant097/30-Days-GANs-Paper-Reading

Extension of StyleGAN with Conditional labels as input.

StyleGAN can generate images of various styles but not of specific labels. So, this paper addresses that limitation by introducing conditional one-hot encoded labels along with noise. So, the transformed intermediate space W should contain label information. Similarly, after affine transformation, style vectors should also contain label information that controls the output of the generator to produce specific labels along with various styles.

Main Difference

In this paper, they pass labels with noise z.



LoGANv2

Figure 6. Producing a conditional latent vector for the StyleGAN architecture

Add Condn label in Input

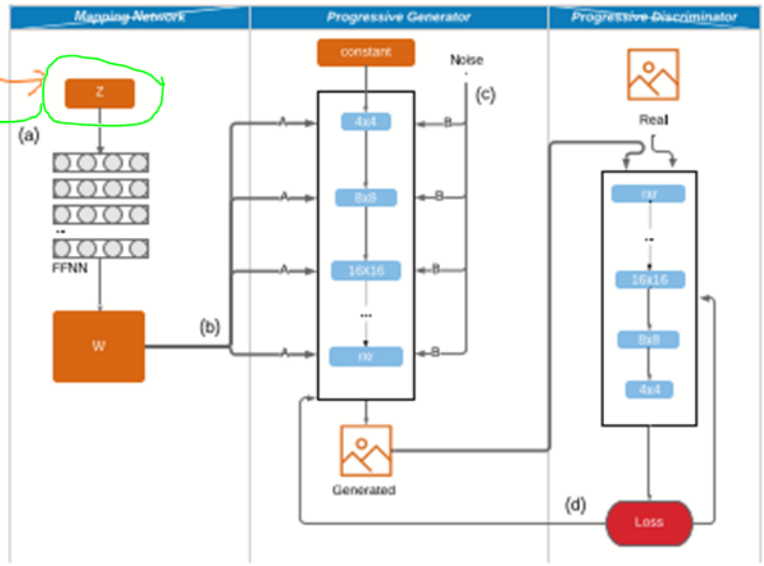


Figure 2. An outline of StyleGAN architectural elements. The mapping

Let's go through the architecture:

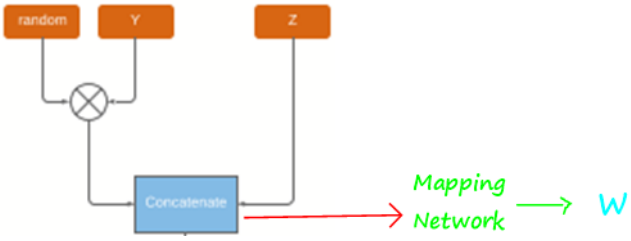
First, they convert labels to one-hot encoded vectors (Y) of size Batch_Size(N) x Channel(C).

They take the noize Z ~ N(μ, σ^2) (Normal Distribution) which is a matrix comprised of the initial latent space of size Batch_Size(N) x Latent_Dim(d).

$$Z = \begin{bmatrix} l_{11} & l_{11} & \cdots & l_{1d} \\ l_{21} & & & \\ \vdots & & & \\ l_{n1} & l_{n2} & \cdots & l_{nd} \end{bmatrix}; Y = \begin{bmatrix} y_{11} & y_{11} & \cdots & y_{1c} \\ y_{21} & & & \\ \vdots & & & \\ y_{n1} & y_{n2} & \cdots & y_{nc} \end{bmatrix} \quad (3)$$

Also, they take Random vector(R) of normal distribution. Multiply it to Y just to add some randomness in labels. Y is multiplied with matrix R ~ N (μ, σ^2) of size NxC and concatenating the result with Z . W , which is the matrix holding the final latent input vectors, is calculated in Equation 4.

$$W = f([Y \times R]^\frown Z) \quad (4)$$



Rest is the same as StyleGAN architecture.

Loss Function:

Also another main difference is in Loss Function. They use WGANGP (WGAN with gradient penalty) loss with conditional label.

Updating the WGAN loss function, the used loss function shows the discriminator considering the conditioning labels paired with respective output/training data. Equation 5 represents this updated loss, where y are the class-conditions

$$\nabla_{\theta D} \left[f_{\theta D}(x|y) - f_{\theta D}(G(\tilde{w}|y)) \right] \qquad (5)$$

The architecture makes use of a gradient penalty term. Combining the term with equation 5, we are left with a conditional WGAN-GP loss function as seen in equation 6 below:

Loss of WGANGP with conditional input

$$\nabla_{\theta D} \underbrace{[f_{\theta D}(x|y) - f_{\theta D}(G(w|y))]}_{\text{Discriminator Loss}} + \underbrace{\lambda[(|| \nabla_{\hat{x}} D(\hat{x}|y)||_2 - 1)^2]}_{\text{Gradient Penalty}} \qquad (6)$$

θ is the respective parameters to be update.

Data used:

They used LLD-logo dataset. They remove fonts containing logos using tesseract. For greater they collect further more logo images through google images scraping.

For Label Extraction, they perform clustering methods. Since they should be clustered and label is assigned that is actually needed to train conditional StyleGAN.

Outputs:

Figure 8. Experiment 2: Synthesized Logos