

Generating and Interpreting Logo Representations Learned by StyleGAN2: Logos without Text

Arin Champati

Partner: Brendan Houle

Adviser: Dr. Olga Russakovsky

Abstract

Logo design is a tedious process in which clients and designers face creative roadblocks, communication difficulties, and possibility of plagiarism. This project aims to mitigate the aforementioned limitations in form of a tool where users can generate logo designs and instantaneously modify cardinal features, such as color, shape, and complexity. After training the state-of-the-art style-based GAN architecture, StyleGAN2 [13], the resulting latent space is explored to find semantically meaningful patterns. This paper will specifically discuss methodologies and limitations as they relate to quasi-conditional output control, projection of real logos, and feature direction discovery. Ultimately, proposed novel automatic feature labeling methods and projection techniques allow for future model-agnostic research in regards to logo generation and latent space exploration.

1. Introduction

There are many difficulties associated with logo design which limit efficiency in development; namely, clients and designers face creative hurdles, copyright risks, and communication barriers. Moreover, for a business attempting to create an original brand identity, a logo design offers consequences other than simply contributing its personality. Instead, certain qualities of a logo, such as ‘asymmetry’, ‘descriptiveness’, and ‘complexity’, tangibly affect customer and market evaluations of a business [8, 9]. For example, in a study designed to determine the impact of ‘descriptive’ vs. ‘non-descriptive’ logos (as determined by hand-labelers) on sales/willingness to buy, Luffarelli et al. found that, across multiple datasets (i.e. logos of Fortune 500 companies and logos of startups), a ‘descriptive’ logo has more positive impact on sales than a ‘non-descriptive’

logo [9]. Because of the significance of a logo’s design, in regards to its contribution to a business’s success, it is important that the creation process is seamless, such that few barriers are faced.

Consider a hypothetical client and designer in the process of creating a logo. In the current state of logo design, a designer drafts many iterations of logos based on the client’s general desires, and the client then narrows down based on their preference. This process is repeated until a final logo is decided upon. With this approach, a client is unable to accurately convey their wishes to the designer, and the designer must guess the client’s wishes via brute force, i.e. painstakingly designing multiple iterations of logos. In addition, the designer must be cautious to not resemble an existing logo to prevent copyright infringements.

This project aims to ameliorate the logo creation process, by mitigating the difficulties described before, in the form of a tool that allows users to generate logos that *do not exist* and to control features in those generated logos, for example, adjusting color and complexity (see Figure 1). My focus was to generate and control features for brand marks, or logos without text, like the Apple logo. My partner, Brendan Houle, focused on logos with text, such as the Walmart logo.

To accomplish this task, a Generative Adversarial Network (GAN), which is used to generate images that resemble the distribution of an input dataset, was trained. In general, GANs have excelled in the synthesis of naturally distributed, low-variation datasets, such as faces and cars [16]. However, domains with a higher degree of feature variability and complexity, such as logos, still pose difficulties in image generation. Thus, an additional goal of this project was to abstract high-variation difficulties associated with GANs from the users of the tool.

After discovering semantically interpretable feature directions via supervised and unsupervised methods and creating a robust system to provide control in generated logos, a study on a set of users showed that the brand mark creation tool was a success, with feature directions and generation-control implementations achieving high ratings.

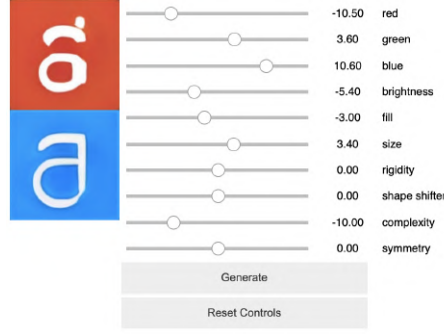


Figure 1: The tool's logo feature controls.

2. Problem Background and Related Work

2.1. Basic GAN

At a high level, a GAN attempts to approximate the underlying probability distribution, p_{data} , associated with the input data [3], where the approximated distribution is denoted as p_z . A GAN consists of two neural networks, named the generator and the discriminator. The generator takes a random noise vector, z , as input and generates samples, and the discriminator classifies if a given sample is real or fake. The networks play a mini-max game, where the generator minimizes the two-sample test ($p_z = p_{data}$), and the discriminator maximizes the objective ($p_z \neq p_{data}$) [3]. Simply put, the generator will try to fool the discriminator by generating images that resemble the input data, but at the same time, the discriminator will get better at classifying those images as fakes. A simple diagram of a GAN can be found in Figure 2¹.

Additionally, because the generator takes vectors z as input where each component is drawn from a Uniform distribution, it is trained to map *any* vector sampled from the same distribution to some reasonable output image. The resulting high dimensional vector space that is spanned by these vectors is referred as the latent space. Certain vector directions in the latent space may also correlate to semantically interpretable features, such as hair color or wrinkles in human faces (if the GAN was trained on images of human faces). A sample of such directions and their effect on the output can be found in Figure 3.

The problems with Basic GAN, especially with a high variation dataset, are that the model

¹Image Source: https://developers.google.com/machine-learning/gan/gan_structure

parameters tend to oscillate (referred to as instability) and never converge, and the model might suffer from mode collapse, where the generator only outputs a few modes it knows can beat the discriminator.

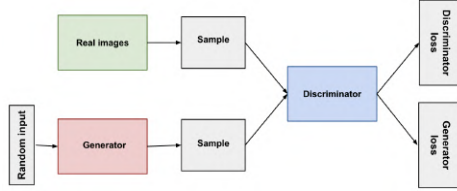


Figure 2: Basic GAN diagram.

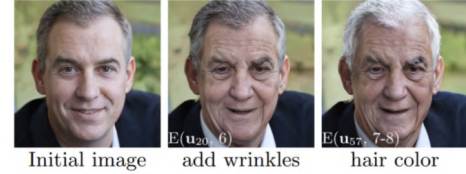


Figure 3: Interpretable vector directions [5].

2.2. Conditional GANs for Logo Generation

There exist prior works that have explored GAN architectures and methods specifically optimized for logo generation. One of the first experiments for GAN logo creation is the 2017 paper, “Logo Synthesis and Manipulation with Clustered Generative Adversarial Networks.” Sage et al. trained on a conditional version of iWGAN, an architecture that fuses auto-encoders and WGAN, which converts the discriminator from a classifier to a numerical score giver and uses Wasserstein loss [2, 15]. A conditional GAN introduces class labels to both the generator and discriminator, such that the output at test time may be controlled by condition. This is done for more stable training and faster convergence, as well as conditional control [15].

The authors also created a tool for logo generation - in this tool, users can change the color of a logo, mix styles between logo classes, and interpolate between logos [15]. However, each feature in the tool carries undesirable characteristics; namely, when shifting colors, shape often also changes (Figure 4) and mixing styles leads to fuzzy results (Figure 5). The limitations of this work may be explained by the entanglement of the latent space due to the architecture, low quality of unsupervised class labeling, training on logos with text, which increases variance in the training distribution, and non-robust feature direction learning methods.

In “LoGANv2: Conditional Style-Based Logo Generation with Generative Adversarial Networks”, Oeldorf and Spanakis created a conditional version of StyleGAN [14]. StyleGAN approaches instability by progressively adding layers (also increasing resolution) during training



Figure 4: Shift to Blue by Sage et al. [15].

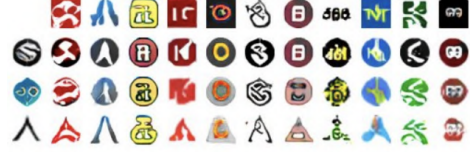


Figure 5: Style mixing by Sage et al. Logos in row 1 are applied as style to logos in column 1 [15].

[10, 12]. The intuition is that the model can first learn large-scale, coarse patterns in the data and later focus on finer details [10, 12]. StyleGAN additionally maps the input vector, z , to a higher dimensional vector, w , via an intermediate feed forward neural network [12]. Each dimension of w is then applied as style to its corresponding layer in the generator [12]. Oeldorf and Spanakis converted this architecture to support class conditioning, for the same reasons as Sage et al., i.e. to provide conditional control to the outputs and further reduce instability. Oeldorf and Spanakis also trained on logos that did not include text [14].

Their results showed that the unconditional StyleGAN model more closely represented the training distribution and produced more conservative samples, while their best conditional model was more ‘creative’, but represented the real distribution far less [14]. The conditional models additionally traded this creativity for higher frequency of ‘nonsensical’ outputs [14]. A comparison between outputted images of the unconditional StyleGAN model and conditional StyleGAN model can be found in Figures 6 and 7.

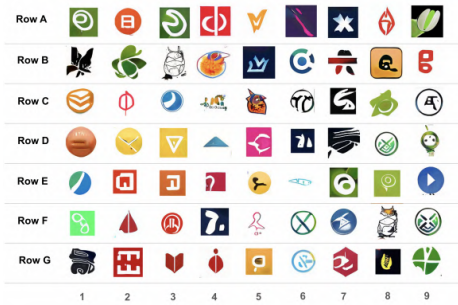


Figure 6: Unconditional StyleGAN Outputs [14].

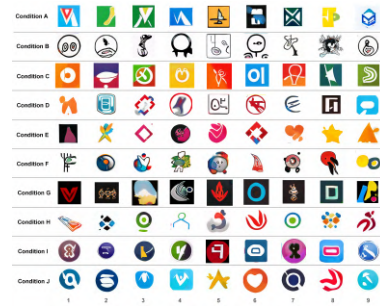


Figure 7: Conditional StyleGAN Outputs [14].

2.3. StyleGAN2

While StyleGAN provides high-quality high-resolution outputs, there are still some undesirable qualities. Namely, the normalization methods in StyleGAN result in noise in the feature maps of all

generated images, and these artifacts can be seen in some outputs [13]. Furthermore, because of progressive growing, in which each resolution's values are generated by its corresponding generator, one generator may have strong preference for a certain mode and may contribute to an unrealistic looking image [13]. For example, in a StyleGAN generated face, if the face is at an angle, the nose often points forward. To combat this, StyleGAN2 does not use progressive growing; instead, it utilizes skip connections between feature maps to make use of multiple scales of generation, while also maintaining the W space [13]. By implementing these changes, StyleGAN2 achieves state-of-the-art performance in multiple GAN evaluation metrics (i.e. Fréchet Inception Distance and Inception Score), far outperforming StyleGAN, while still allowing style control via its z to w mapping network [13].

3. Approach

The aim of constructing the logo creation tool was not to provide finalized logo designs; instead, the aim was to inspire creativity and improve illustrative communication. Thus, while generating realistic logos was desired, most emphasis was placed on the ability to quickly recast and modify a logo design. Therefore, the majority this project related to finding interesting, yet semantically interpretable directions in the latent space, such that users could alter the structure of a logo instantaneously. Note that the tool itself is a Python Notebook, which includes the implementations discussed in this paper, as well as implementations discussed in my partner's paper.

One of the most important distinctions between logo designs is the inclusion or emission of text. For this reason, in the tool, a separation must be made between brand mark generations and textual logo generations. In order to achieve this distinction and avoid entanglement, my partner and I decided that entirely separate models must be trained and separate directions must be learned. My focus was to discover directions for brand marks as I was intrigued by the possibility of learning impressive, yet serviceable shape and color related features. Inspired by each feature's measured impact on sales/willingness to buy, priority was placed on finding the latent space directions that represented size, symmetry, complexity, shape, and color [8, 9].

3.1. Data and Model

Given the intuition gained from previous works and the goals set forth above, StyleGAN2 was chosen as the architecture to train due to the ability to control resolutions of applied styles and the improvements over its predecessor, StyleGAN. The models were trained on 128x128 images (due to time and resource constraints) on two separate configurations of StyleGAN2, Configurations E and F. In both configurations, every proposed deviation from StyleGAN is applied - Configuration F is, however, a larger network than Configuration E (i.e. higher resolution layers contribute more to the generated outputs) [13].

StyleGAN2 was trained on truncated version of the Boosted Large Logo Dataset (BLLD) [14, 15]. The BLLD contains a collection of 40,000 non-text logos scraped from Twitter, as well as 15,000 logo-like images, as shown in Figure 8 [14]. The dataset still contained many logos with text, so I used Google’s Tesseract, an optical character recognition engine, to truncate it further [14, 17]. Thresholding by the detection confidence on binarized images to limit false positives, 700 logos that contained 2 or more characters were removed. A visualization of the removed logos can be found in Figure 9. The final dataset, referred to as the Truncated Boosted Large Logo Dataset (TBLLD), contains mainly brand-marks, logo-like images, and single character logos.



Figure 8: Additional logo-like images added to BLLD [14].



Figure 9: Subset of binarized text-logos removed from BLLD.

3.2. Quasi-Conditional Control and Robust Logo Selection

Because StyleGAN2 is not a conditional architecture, users are unable to specify the class of brand mark that is generated. Additionally, due to the high-variation in the TBLLD, many generated logos are noisy and uninspired. To account for these issues, the user is given the ability to generate

randomly or from an index of good logos, an index of logos by feature, or an index of real-life logos (i.e. they are given quasi-conditional control of generated outputs).

To reduce noisy outputs, a list of z vectors that map to realistic generations was curated (Table 8). Moreover, a side effect of automatic labeling, as explained in Section 4.2 (“Supervised Latent Space Exploration”), is that the labels themselves are class clusterings. So, for example, users can select green logos (Table 9 and Table 10) or complex logos (Table 12) from each respective index of labeled images. In addition, real-life logos can be projected into the latent space so that, for instance, a user can gain tangible inspiration from the Spotify logo (Table 14). Finally, given a generated logo, a user can choose from an index of ‘similar’ logos, that vary from the original by shape, color, or both.

In the tool, a user is prompted to select two starting logos via the methods described above and is then presented the ability to mix shape and color styles between the logos, referred to as interpolation. In sum, the user is provided *limitless* control when selecting a logo to apply feature directions on, which should improve the perceived quality of the tool, while maintaining variation and semantic interpretability of generated outputs.

4. Implementation

4.1. Style Mixing

Each 512-dimensional latent vector, z , can be mapped to its corresponding 12x512-dimensional w vector via the feed forward mapping network, f , such that $f : z \rightarrow w$ [13]. w vectors allow resolution specific control when mixing styles between latent representations (throughout this paper, ‘resolution’ refers to the layer, or set of layers, of w on which a style is applied to). Consider applying the style of w_2 to w_1 . For every desired application resolution, w_2 can simply replace w_1 at those resolutions. Lower resolutions [0-7], generally change structural or shape-related features, while higher resolutions [8-11], generally change color palette. Shape mixing can be visualized in Figure 10, and color mixing in Figure 11, where each logo in the first row is applied as style to each logo in the first column.

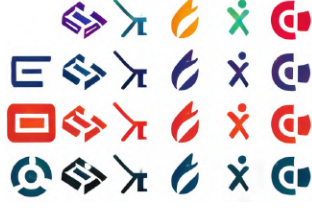


Figure 10: Shape style mixing.



Figure 11: Color style mixing.

4.1.1. Resolution Control in Interpolation A useful application of style mixing is interpolation, where styles are fractionally mixed. Consider w_1 and w_2 at a certain resolution, r . Then, for some α , interpolation over a resolution, r , where w_2 is applied as style to w_1 is defined as,

$$\hat{w}_r = w_{1r} \cdot (1 - \alpha) + w_{2r} \cdot \alpha \text{ for } r \in 0, 1, \dots, 11$$

The above equation can be applied separately to shape resolutions [0-7] with α_{shape} and color resolutions [8-11] with α_{color} . An example of interpolation can be found in Figure 12, where α_{shape} increases by row, and α_{color} increases by column (i.e. the top left and bottom right logos denote w_1 and w_2 respectively).



Figure 12: Shape and color interpolation.



Figure 13: Index of similar logos.

4.1.2. Resolution Control to Obtain Similar Logos Given a w vector, an index of similar latent representations can be obtained by applying multiple random directions, w_{random} , to shape resolutions [0-5] and color resolutions [8-11] separately. For higher variation between both shape and color, z_{random} can be applied directly to the corresponding z vector. Note that this method expands ‘Vicinity Sampling’, as originally described by Sage et al. [15], by introducing resolution specific control. Figure 13 demonstrates this method, where the leftmost column indicates the original logo.

4.2. Supervised Latent Space Exploration

This section explains the methods used to discover latent space directions that correlate to automatically labeled latent vectors. Each of these features were motivated in Section 3 (“Approach”).

4.2.1. Learning Pipeline 20,000 random z vectors were generated and mapped to images via the generator. Then, each vector’s corresponding image was labeled via automatic methods (explored later in this section) such that each latent vector, z , was associated with a binary class label, l . Logistic regression was trained on 16000 of these samples (4000 were left unseen as a test set) with z vectors as input and their associated labels as the true output. Because the class labels were often very unbalanced (for example there were around 3500 green logos and 16500 non-green logos in the labeled dataset), class weights, inversely proportional to the frequencies of each class, were utilized, so that the model did not learn to classify all samples as the negative class. ElasticNet regularization was additionally used to prevent overfitting by enforcing both low complexity and sparsity in the model [20].

Once the classifier was fit, the coefficients of features in the decision function, θ_z , were used as the direction associated the feature (after dividing by the norm to obtain the unit vector), such that,

$$feature\ direction = \frac{\theta_z}{||\theta_z||}$$

The coefficients θ_z indicate the effect of a one-unit change in a dimension of z on the log odds of being a certain feature. Intuitively, traveling along this direction in the latent space increases the log odds, and therefore the odds, of a latent vector representing that feature.

To apply the direction θ_z on a given sample w , the resolution of the feature associated with θ_z was first considered, i.e. low resolution (shape) or high resolution (color). Let b denote the desired beginning resolution applying θ_z to w , and let e denote the desired ending resolution for applying θ_z to w . Let r denote a particular resolution of w . Finally, let \hat{w}_a denote the layers of w in which θ_z is applied, and let \hat{w}_n denote the layers of w in which θ_z is not applied. Then, for some distance, d , and scale, s ,



Figure 14: Traveling along green direction with averaging method.

$$\hat{w}_a = w_r + (\theta_z \cdot d \cdot s), \text{ for } r \in \{b, \dots, e\} \quad \hat{w}_n = w_r, \text{ for } r \notin \{b, \dots, e\}$$

Note that while θ_z was learned in the Z space, it may still be applied to w layers due to the learned non-linear mapping network. Attempts were made to learn in the W space, however, the models performed poorly due to the increase in input dimensionality. Mapping θ_z to θ_w and applying particular resolutions of interest also led to subpar direction traversals. Through experimentation, the range of resolutions to apply θ_z on a given w for each feature were determined as: Color [8-11], Size [4-5], Complexity [0-3], and Symmetry [0-5].

An alternative approach is to label a sequence of latent vectors, take the average of those vectors, and divide by the norm, mirroring the method used by Sage et al., to obtain z_{avg} [15]. z_{avg} can be applied at the desired resolutions to obtain \hat{w}_a and \hat{w}_n (as described above). The problem with this approach is that it dramatically overfits on the training set and does not represent ‘extreme’ examples very well. By taking the mean vector, it heavily considers well represented vectors in the latent space. With logistic regression, the learned direction should extend into less represented, ‘extreme’ areas. An example of the shortcomings of the averaging method can be found in Figure 14, where green vectors were averaged and travelled along for a very short distance (applied to high resolution layers). As the latent vector gets more extreme, the logo becomes unintelligible, and the colorings become noisy.

4.2.2. Color by RGB Percentage Contribution The goal of this labeling technique was to determine if a logo was a specific color, i.e. red, green, or blue, by comparing each channel’s percentage contribution to the output. For a given sample, percentage contribution was determined as the sum of pixel values in one channel divided by the sum of pixel values in all channels. Let c denote a particular channel and s denote the sum of pixel values in that channel. Then, the total sum of

pixels across all channels is $T = s_1 + s_2 + s_3$. Additionally, for some channel, c , the percentage contribution to the output, p , can be calculated by $p = \frac{s}{T}$.

Let c_1 denote the color channel associated with the color to be labeled, and c_2, c_3 to denote the other color channels. Then,

$$p_1 = \frac{s_1}{T} \quad p_2 = \frac{s_2}{T} \quad p_3 = \frac{s_3}{T}$$

Let C denote the indicator variable that the image is labeled with color c_1 , then,

$$C = \begin{cases} 1 & \text{if } p_1 > p_2 \geq p_3 \\ 0 & \text{otherwise} \end{cases}$$

4.2.3. Color by HSV Maximum Quantity ‘Color by RGB Percentage Contribution’ sometimes produces an undesirable labeling, which may lead to entanglement in the learned direction. Namely, a higher percentage contribution of one color channel may not necessarily denote that the image is perceived as that color. Consider Figure 15, which represents a tan color with the red channel containing the highest percentage contribution to the output. With ‘Color by RGB Percentage Contribution’, this color would be classified as red, which is not desirable.



Figure 15: Tan with red channel as highest percentage contribution.

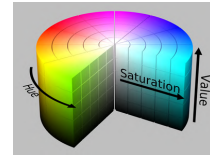


Figure 16: HSV Cylinder.

Instead, the image can be transformed to the HSV (Hue, Saturation, Value) color space, which attempts to mimic human perception of color. This color space considers color classes exclusively in the Hue channel, where each degree range (degrees can range from 0° to 360°) represents a different class. Additionally, the Value channel (range of 0 to 255), represents the brightness of a color. The

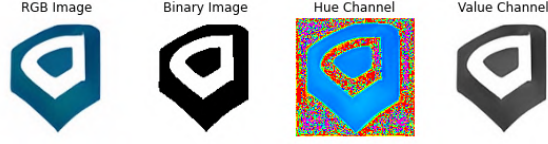


Figure 17: Hue channel noise for white pixels.

HSV cylinder is shown in Figure 16². HSV is thus desirable for automatic labeling of color as color class determination is reduced to selecting from a range of Hue degree values. Namely, for given a pixel, p , the following degree ranges of Hue H (or Value V), denote its color class,

Red: $H < 20^\circ$ or $340^\circ < H$

Green: $100^\circ < H < 140^\circ$

Blue: $220^\circ < H < 260^\circ$

Black: $V < 10$

Let q_1 denote the total quantity of pixels of the desired color class, and let q_2 , q_3 , and q_4 denote the total quantity of pixels of the other classes (including black). Let C denote the indicator variable that the image is labeled with the color associated with q_1 , then,

$$C = \begin{cases} 1 & \text{if } q_1 > q_2 \geq q_3 \geq q_4 \\ 0 & \text{otherwise} \end{cases}$$

It is important to note that the Hue channel was preprocessed to account for white pixels. Namely, because brightness is only controlled by the Value, a visually white pixel will have an associated Hue. To account for this, the images were binarized, and pixels with 0 values in the binary image were ignored when determining the quantity of pixels for each color class. Motivation for this preprocessing step can be visualized in Figure 17.

4.2.4. Size by Binary Thresholding The goal of this labeling technique was to determine if a logo has small size by thresholding on the percentage of white pixels in the image. The intuition is that, as long as the logo itself is not white (i.e. the background is white), then in general, an increasing number of white pixels may correlate to a smaller logo. To accomplish this labeling, the image was

²Image Source: https://en.wikipedia.org/wiki/HSL_and_HSV



Figure 18: 5-pixel corner summing regions on generated logo and binarized version.

first binarized. Then, let q_w denote the quantity of white pixels in the binarized image, and let q_b denote the quantity of black pixels in the binarized image. Let p_w denote the percentage of white pixels in the image, such that $p_w = \frac{q_w}{q_w + q_b}$. Finally, let S denote the indicator variable that the image is labeled as a small logo, then,

$$S = \begin{cases} 1 & \text{if } p_w > 0.85 \\ 0 & \text{otherwise} \end{cases}$$

Note, a threshold of 0.85 was chosen so that 15% of logos were labeled as small.

4.2.5. Size by Filled Corner Pixels The goal of this labeling is the inverse of ‘Size by Binary Thresholding’, namely, it aims to label logos with fully colored backgrounds. The intuition is that because the latent space has a roughly linear structure, the direction correlating to fully colored backgrounds might also change the size of the logo. To accomplish this labeling, the image was first binarized, and then, at every corner, the sum of binary pixels in a 5x5 area was calculated, as denoted by the corner squares in Figure 18. Let T denote the highest sum possible (i.e. 25), and let q denote the sum of binary pixels in all four square areas. Finally, let S denote the indicator variable that the image is labeled as a logo with a filled background, then,

$$S = \begin{cases} 1 & \text{if } \frac{q}{T} > 0.9 \\ 0 & \text{otherwise} \end{cases}$$

Note, a threshold of 0.9 was chosen to accommodate for logos that have *almost* completely filled backgrounds.

4.2.6. Size by Hough Circles and Filled Corner Pixels The mean images of the real dataset and a set of 20,000 generated logos resemble circles, suggesting that on average, logos are circular



Figure 19: Mean image of real dataset (left), and mean image of generated images (right).



Figure 20: Hough Circles method demonstration. Green circle denotes most central detection.

(Figure 19). Based on this observation, circles in the logos can be detected, and classes can be labeled based on a detected radius threshold.

The Circle Hough Transform was used to detect multiple circles at different radii, and the radius of the most centrally located circle was used to determine class labels. Some limitations of this method is that if the logo is very rigid, the detected circles may not accurately represent the size of the logo, as shown in Figure 20, where the green circle denotes the most central detection, and the red circles denote other detections. Moreover, any logo with a filled background should be determined as larger as explained in Section 4.2.5 (“Size by Filled Corner Pixels”).

Let r denote the radius of the most centrally detected circle, and let F denote the indicator variable that the image is labeled with a filled background (from Section 4.2.5). Let S denote the indicator variable that the image is labeled as a large logo, then,

$$S = \begin{cases} 1 & \text{if } r > 50 \text{ or } F = 1 \\ 0 & \text{otherwise} \end{cases}$$

Note that a radius of 50 was chosen as the threshold so that 20% of logos were identified as large or with filled backgrounds.

4.2.7. Structural Complexity by Blob Thresholding The goal of this labeling technique was to determine if a logo was complex by thresholding on the number of detected features. An elementary feature detection method, blob detection, was used to identify amorphous regions of a grayscale



Figure 21: Blob Thresholding method demonstration.

logo that share a common feature (brightness) and differ greatly from surrounding regions. Let q denote the quantity of blobs detected for a given image, and let CX denote the indicator variable that the image is labeled as a complex logo, then,

$$CX = \begin{cases} 1 & \text{if } q > 13 \\ 0 & \text{otherwise} \end{cases}$$

Note that a threshold of 13 was used so that 20% of logos were identified as complex.

4.2.8. Structural Complexity by Spatial Information Thresholding There exists a key limitation with ‘Structural Complexity by Blob Thresholding’, which mainly relates to the method’s underlying assumptions. More specifically, the number of detected blobs does not necessarily imply higher complexity, instead, it simply signifies that a logo has distinguishable regions. This problem can be visualized in Figure 21, where the red circles represent the detected blobs. Blob Thresholding would indicate that the left image is more complex than the right image, but this is not the case.

Instead, image complexity can be determined as a function of its compression ratio [19]. The intuition is that the less complex an image is, the more it can be compressed, vice versa. However, determining this complexity score would be very costly as each image must be compressed. Alternatively, spatial information (SI), specifically, the mean of spatial information (SI_{mean}), is a robust estimator for image complexity [19]. Let s_h and s_v denote a grayscale image that has been filtered with horizontal and vertical Sobel kernels, respectively. Then [19],

$$SI = \sqrt{s_h + s_v} \quad SI_{mean} = \frac{1}{P} \sum SI$$

Let CX denote the indicator variable that the image is labeled as a complex logo, then,

$$CX = \begin{cases} 1 & \text{if } SI_{mean} > 1000 \\ 0 & \text{otherwise} \end{cases}$$

Note that a threshold of 1000 was used so that 20% of logos were identified as complex.

4.2.9. Structural Symmetry by Split Comparison The goal of this labeling technique was to determine if a logo was symmetrical by splitting the image evenly over each axis and comparing binary pixel values for each pair of halves. Let q_h denote the quantity of pixels that matched between horizontally split halves, and let q_v denote the quantity of pixels that matched vertically split halves. Let T denote the total number of pixels in the image, and let SS denote the indicator variable that the image is labeled as a symmetric logo, then,

$$SS = \begin{cases} 1 & \text{if } \frac{q_h}{T} > 0.7 \text{ or } \frac{q_v}{T} > 0.7 \\ 0 & \text{otherwise} \end{cases}$$

Note that a threshold of 0.7 was chosen so that 15% of logos were identified as symmetric. Additionally, consider that this is a very elementary approach to symmetry as it only considers pixel similarity, instead of feature similarity, across halves. Logos with filled backgrounds or very small logos (i.e. with white backgrounds), are identified as symmetric, as naturally, most of these pixel values align. Thus, this method largely does not achieve its purpose.

4.3. Unsupervised Latent Space Exploration

While supervised latent space exploration allows for control and immediate interpretability of the learned directions, entanglement is introduced through imperfect labeling. Furthermore, there are features that cannot be labeled via automatic methods, especially structural features. These problems motivate unsupervised latent space exploration, in which the process of determining features is reversed; namely, the directions are first learned and then interpreted/labeled.

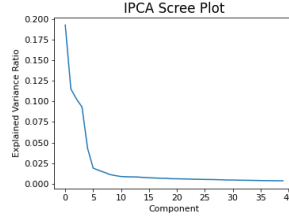


Figure 22: PCA Scree Plot.

4.3.1. PCA on Latent Space The intermediate learned latent space, W , is 12×512 dimensional, so it seemed beneficial to reduce the dimensionality and interpret each reduced component. This can very simply be done via Principal Component Analysis (PCA), where new characteristics (components) are constructed as linear combinations of the original data, such that components are uncorrelated and explain maximal variance. The implementation from Härkönen et al. was used, where a collection of 300,000 w vectors are generated, and PCA is applied on those vectors, such that a basis, V , for the latent space, W , is created [5]. Then, by varying each entry x_k in x , a vector of dimension intensities, a new vector \hat{w} may be obtained, such that [5],

$$\hat{w} = w + Vx$$

4.3.2. Interpreting Principal Feature Directions A scree plot comparing explained variance was used to identify how many of the first principal feature directions should be interpreted. As shown in Figure 22, features obtained by applying the basis at components 0-10 are sufficient as the explained variance drops significantly after the 5th component. In fact, the first 5 components alone explain 55% of the variation.

The first 10 components were explored in silos, where starting and ending resolutions, as well as distance traveled along the directions, were varied. Each component was heavily scrutinized, yet concluding multiple interpretable shape-related features was very difficult. Ultimately, only a few components carried interpretable meaning, as will be shown in Section 5.3.4 (“Unsupervised”).

4.4. Projection into Latent Space

Existing logos can be projected into the latent space via regression. In the official StyleGAN2 implementation, among noise regularization, the regression loss is dominated by ‘perceptual loss’ [13]. By encouraging high-level feature similarity, perceptual loss discourages pixel to pixel replicas so that the learned vector is well represented in the latent space [7, 13]. However, the difficulty with this approach, in regards to logo synthesis, is that VGG16 layers, trained on ImageNet, are utilized to extract features for comparison. Because ImageNet is a natural dataset and the extracted features are high-level, use on the unnatural logo dataset may result in undesirable, inconsistent behavior.

Instead, pixel to pixel loss can be reintroduced in the loss function, such that the projection exists in a well-represented region of the latent space, while still resembling the original logo. L1 distance will be used, instead of L2 distance, as it tends to encourage less fuzzy outputs in image-to-image translation [6]. Informally, let L_s denote the perceptual loss with noise map regularization as proposed by the original StyleGAN2 paper [13], let $I_{projected}$ denote the projected image, and let $I_{original}$ denote the original image, then the new loss, $L_{perceptual+L1}$, is,

$$L_{perceptual+L1} = L_s + |I_{original} - I_{projected}|$$

5. Results and Evaluation

5.1. Models

The Fréchet Inception Distance (FID) is used to measure the quality of generated images in comparison to the real distribution [4]. The mean and covariance of the generated and real distributions are estimated after embedding into an InceptionV3 feature space, and the Wasserstein Distance is used as a measure of the quality of generated samples [1, 4]. A lower FID denotes a closer resemblance of the generated distribution to the real distribution [4]. FID is a desirable (perhaps the most robust)

GAN evaluation metric as it aligns with human judgement, detects mode collapse, is sensitive to image distortions, and prefers high-fidelity generations [1].

FID scores for StyleGAN and Conditional StyleGAN, trained on the BLLD by Oeldorf and Spanakis [14], as well as StyleGAN2 Configuration E and Configuration F, trained on the TBLLD, can be found in Table 1. By the FID score, StyleGAN2 Configuration F far outperforms other methods in regards to output quality and similarity to the real distribution. The FID scores followed qualitative observation - in general, Configuration E generated much noisier outputs than Configuration F. For these reasons, StyleGAN2 Configuration F trained on the TBLLD was chosen for latent space exploration and was included in the final tool.

Table 1: Model FID Scores

	StyleGAN	Conditional StyleGAN	Config E	Config F
FID	47.5	101.9	34.7	28.6

5.2. User Study

A study was conducted on a set of 10 users, in which users were directed to use the tool and answer key questions. Perceived quality of outputs, learned directions, projections, and other implementations are otherwise difficult to capture via strictly quantitative measures. Results can be found in Table 2. ‘Generation Ratings’ denote the average score on a scale of 1 to 5 for each logo generation method, where 5 is a perfect rating. ‘Random’ denotes random generation, ‘Good Index’ denotes selecting from a curated list of good logos, and ‘Existing Index’ denotes selecting from a list of projected logos. Unsurprisingly, selecting from a list of good indices outperformed random generations. Additionally, the users perceived high resemblance for projection outputs. ‘Did the expected behavior occur?’ was asked for logo selection by feature as well as interpolation by resolution. The proportion of ‘yes’ responses are reported. Of the color selection methods, green performed the worst. Furthermore, shape related indices performed worse than color, in general, which indicates entanglement in the automatic labeling. Finally, ‘Was this feature helpful?’ reports the proportion of ‘yes’ responses for overarching implementations. Each major implementation was

deemed as useful for all users. Note that users were also asked to rate each feature direction and method, on the same 1-5 scale as used before, however, those values are reported for each feature within Section 5.3 (“Feature Directions”).

Table 2: User Study

Generation Ratings		Did the expected behavior occur?		Was this feature helpful?	
Random	3.8	Red Index	1.0	Similar Logos	1.0
Good Index	4.7	Green Index	0.7	Style Interpolation	1.0
Existing Index	4.3	Blue Index	0.9	Feature Directions	1.0
		Filled Background Index	0.9		
		Complexity Index	0.7		
		Symmetric Index	0.5		
		Color Interpolation	0.9		
		Shape Interpolation	1.0		

5.3. Feature Directions

The F1 Score on the held-out test set is reported for each fitted model. F1 Score is the harmonic mean of Precision and Recall and aims to seek balance between both the measures, which makes it more suitable for class imbalances, as opposed to accuracy. Precision, Recall, and F1 Score are defined as follows, where TP, FP, and FN denote the number of true positives, false positives, and false negatives respectively,

$$Precision = \frac{TP}{TP + FP} \quad Recall = \frac{TP}{TP + FN} \quad F1\ Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

A balance between Precision and Recall was desired because the learned directions are meant to be travelled along in positive *and* negative directions - i.e. false positives and false negatives are equally important, to avoid entanglement. However, the F1 Score simply provides insight into how well the direction was learned, which does not necessarily lend to the quality of the direction itself.

Instead, to provide insight into the perceived quality of the learned direction, two additional metrics are reported: Hand Label Score and User Study Avg. Rating. Hand Label Scores were determined by generating 100 pairs of logos for a given feature and method. Each pair consisted of

the original generated image, I , and the resulting image, \hat{I} , after traveling 20 units in the positive feature direction. Then, the percentage of pairs in which the expected changes occurred, when comparing I to \hat{I} , is reported. However, the Hand Label Score introduces my own biases into the evaluation. For example, my perception of complexity may differ from another user. So, the User Study Average Rating, which averaged a feature and method’s scores as determined by 10 users (as described in Section 5.2 (“User Study”)) is also a reported metric.

Additionally, for each feature and method, tables of visualizations are included in Section 9 (“Appendix”), which include hand selected traversals in both positive and negative directions, where the leftmost image indicates the starting logo, and the rightmost image indicates the farthest traversal along the learned vector. For supervised features, visualizations for the automatic labelings are also included.

5.3.1. Color In addition to the metrics described in Section 5.3 (“Feature Directions”), the Average Mean Squared Error per channel is reported in order to interpret color direction entanglement. To obtain these values, 5000 pairs of images for a given color direction were generated. Then, the MSE was obtained between each channel of the original image, $I(x, y, z)$, and resulting image, $\hat{I}(x, y, z)$, after traveling 20 units in the positive color direction, such that,

$$\begin{aligned} MSE_R &= \frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N [I(i, j, 0) - \hat{I}(i, j, 0)]^2 \\ MSE_G &= \frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N [I(i, j, 1) - \hat{I}(i, j, 1)]^2 \\ MSE_B &= \frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N [I(i, j, 2) - \hat{I}(i, j, 2)]^2 \end{aligned}$$

The per channel MSE for every image was calculated, as determined by the formulas above, and averages of these values are reported. Even though image reconstruction is not involved, Average MSE per channel provides intuition on the differences between I and \hat{I} in regards to color. For example, for a non-entangled direction that correlates to red, $Avg. MSE_R$ is expected to far exceed $Avg. MSE_G$ and $Avg. MSE_B$.



Figure 23: Blue direction traversal (top) and Green direction traversal (bottom).

The quantitative results comparing color methods can be found in Table 3, and visualizations for RGB Percentage Contribution and HSV Maximum Quantity can be found in Table 9 and Table 10 respectively. Based on these results, Red, Green, and Brightness via HSV Maximum Quantity and Blue via RGB Thresholding are included as the color directions for the final tool. Each of these chosen directions outperformed their counterparts in all metrics except for F1 Score.

Table 3: Color Methods Comparison

	RGB Percentage Thresholding			HSV Maximum Quantity			
	Red	Green	Blue	Red	Green	Blue	Brightness
Avg. MSE_R	0.084	0.099	0.068	0.157	0.077	0.076	0.007
Avg. MSE_G	0.031	0.030	0.023	0.015	0.049	0.035	0.006
Avg. MSE_B	0.042	0.108	0.112	0.038	0.130	0.096	0.006
F1 Score	0.69	0.63	0.70	0.66	0.62	0.62	0.68
Hand Label Score	0.82	0.63	0.86	0.87	0.79	0.74	0.88
User Study Avg. Rating	4.7	3.9	5.0	5.0	4.6	4.2	4.7

Blue learned by HSV Maximum Quantity is entangled with Red, as shown by the Average MSE values. This might be explained by its automatic labeling, where there are multiple dark blue and purple samples included (Table 10). Peculiarly, Green for both methods underperformed, with the Average MSE values (and general observation) indicating heavy entanglement with blue. This problem can be visualized in Figure 23, where the top image indicates a traversal in the Blue direction learned by RGB Percentage Contribution, and the bottom indicates a traversal in the Green direction learned by HSV Maximum Quantity. In both cases, the logo becomes light blue. Ultimately, in general, the color directions were rated very highly and fulfilled their purpose.

5.3.2. Size The metrics comparing size methods can be found in Table 4, and visualizations for each method can be found in Table 11. Based on the results, size learned by Hough Circles is included in the final tool as it outperforms its counterparts in all metrics.



Figure 24: Size by Hough Circles traversal.

Note that while the size of the logos did generally change, the directions were slightly entangled - the underlying structure of the original logo was sometimes affected. This can be visualized in Figure 24, which depicts a traversal with the Hough Circles method, where the logo loses its center rectangle as the size increases. However, the Hough Circles method generally affected the underlying features the least, as reflected in the User Study Avg. Rating.

Table 4: Size Methods Comparison

	Binary Thresholding	Filled Corner Pixels	Hough Circles
F1 Score	0.71	0.78	0.80
Hand Label Score	0.73	0.75	0.82
User Study Avg. Rating	4.0	4.0	4.5

5.3.3. Structural Complexity and Symmetry The metrics comparing complexity and symmetry methods can be found in Table 5, and visualizations for each method can be found in Table 12. Based on the results, complexity learned by Spatial Information is included in the final tool as it outperforms its counterpart in all metrics. The Blob Detection method labeled many images with filled backgrounds as complex, which introduced size entanglement in the learned direction. This can be visualized in Figure 25, where the top image indicates a traversal on the Spatial Information direction, and the bottom indicates a traversal on the Blob Thresholding direction. The top traversal increases complexity without affecting size, and the bottom traversal increases size.

Additionally, the shortcomings regarding automatic labeling of symmetry, as discussed in Section 4.2.9 (“Structural Symmetry by Split Comparison”), resulted in a subpar learned direction, as reflected by its low ratings.

5.3.4. Unsupervised Only two interpretable directions were discovered after applying PCA on the latent space. The metrics for these directions can be found in Table 6, and visualizations for each method can be found in Table 13. Note that visualizations for a direction named ‘Shape Shifter’ are also provided, however, metrics are not reported due to its entangled behavior.



Figure 25: Complexity by Spatial Information traversal (top) by Blob Detection traversal (bottom).

Table 5: Structural Complexity and Symmetry Methods Comparison

	Blob Thresholding	Spatial Information	Symmetry
F1 Score	0.64	0.64	0.75
Hand Label Score	0.48	0.62	0.22
User Study Avg. Rating	3.9	4.4	2.9

The ‘Fill’ direction, obtained by applying component 0 at resolutions 6 and 7, reduces the logo to an outline if traversing in the negative direction and colors any white pixels in the positive direction, as demonstrated in Figure 26, where the middle image indicates the starting logo. This direction almost always acts as intended, especially in the negative direction, confirmed by its Hand Label Score of 0.94. The ‘Rigidity’ direction, obtained by applying component 2 at resolutions 0 through 4, makes logo features more circular in the negative direction and more rigid in the positive direction, as demonstrated in Figure 27, where the middle image indicates the starting logo. The ‘Shape Shifter’ direction, obtained by applying component 0 at resolutions 0 through 5, changes the logo type, increases size, and increases complexity. Likely none of these directions could have been learned by supervised methods, and the directions are especially robust for shape features.

Table 6: Unsupervised Methods

	Fill	Rigidity
Hand Label Score	0.94	0.74
User Study Avg. Rating	4.7	4.6

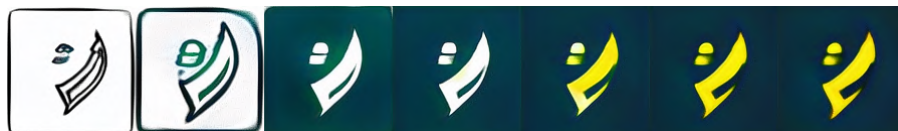


Figure 26: Fill traversal in both directions.



Figure 27: Rigidity traversal in both directions.

5.4. Projection

The objectives of projection include resembling the original image as well as existing in a well-represented region of the latent space (to allow for predictable feature direction traversals). To measure the quality of latent space representation, the Mean Euclidean Distance from $w_{average}$ is reported. $w_{average}$ is computed by mapping many random z vectors to w and taking the average of these vectors [13]. For context, 20,000 random w have a Mean Euclidean Distance of 65.63 from $w_{average}$.

Additionally, the Mean Image-to-Image MSE is reported. This metric is slightly misleading as a loss function that aims to minimize MSE (L2) will perform the best. Moreover, it does not accurately describe the overall similarity between two images. To account for this, Mean Structural Similarity Index (Mean SSIM) is reported, which measures perceptual differences between logos by comparing contrast, luminance, and structure [18]. It is often used to quantify data compression loss, where values close to 1 indicate high similarity.

For the purposes of comparison and context, metrics are reported on 23 projections that use (in combination with noise map regularization [13]) L1 loss, L2 loss, perceptual loss, and perceptual loss with added L1 loss. The comparisons can be found in Table 7. Interestingly, the loss function proposed in this project (perceptual + L1) outperforms the default StyleGAN2 implementation (perceptual) in all metrics. Perceptual + L1 projections feature crisp edges and accurate colors, which can be visualized in Table 14.

Often times, the perceptual loss method does not generate an image that closely resembles the original logo, as visualized in Figure 28, where a projection of the TikTok logo with perceptual and perceptual + L1 methods are compared. Note that the perceptual + L1 method not only closely resembles the original image, but its w representation is half the distance to $w_{average}$ (263.7)

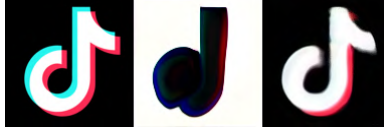


Figure 28: TikTok original image (left), perceptual loss projection (middle), and perceptual + L1 loss projection (right).

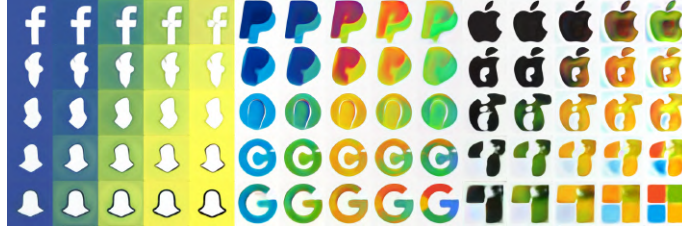


Figure 29: Interpolations on perceptual + L1 loss projections.

than the default method (421.6). Additionally, only some projections allow for non-noisy latent space traversal. For example, consider interpolations between projected logos in Figure 29. The interpolation between Facebook and Snapchat provides legible intermediate logos, however, Apple and Microsoft provide turbulent intermediate logos. In general, projections with highest euclidean distance from $w_{average}$ produced the worst latent space traversals.

Table 7: Projection Methods Comparison

	L1	L2	Perceptual	Perceptual + L1
Mean Euclidean Dist. from $w_{average}$	387.3	467.5	328.1	312.1
Mean Image-to-Image MSE	0.0080	0.0058	0.1014	0.0082
Mean SSIM	0.88	0.77	0.73	0.90

6. Conclusion

6.1. Summary

The principal objective of this project was to create a functional tool that allows users to quickly generate and iterate through brand mark designs. When considering contributions by my partner, Brendan Houle, the complete tool allows for generation and feature control for logos both with and without text. Through means of quantitative analysis, and more importantly, a user study, the tool and its constituent implementations were determined as helpful and meaningful.

Several key contributions were made during the process of constructing the brand mark portion

of the tool. Namely, novel automatic feature labeling techniques for brand mark images were introduced, which enabled pseudo-conditional control of generated outputs as well as effective discovery of latent space directions. Additionally, a novel projection loss function was introduced that improves the projection results in regards to visual similarity and latent space representation. As side effects, the project also produced the Truncated Boosted Large Logo Dataset (TBLLD) as well as trained StyleGAN2 models. In sum, a user can generate logos by condition or project existing logos into the latent space, generate similar logos, mix styles between logos, and control color, size, complexity, symmetry, rigidity, fill, and shape all with the click of a button or movement of a slider.

6.2. Limitations and Future Work

In general, automatic labeling of color, as well as color feature directions, outperformed shape labeling and directions. Subjective structural features, like complexity, were more difficult to quantify, which led to entanglement in labeling. Additionally, while size, complexity, and symmetry feature directions provided somewhat reasonable output displacement, major identifying features were sometimes lost during traversal. Unsupervised methods indeed did discover more robust structural features, but only a few principal components could be semantically interpreted and presented in the final tool. Because this project demonstrates that logo feature directions can be successfully learned via logistic regression, it may prove beneficial to hand label structural features. While the feature directions will likely still be entangled, they may outperform the methods used in this paper.

In regards to projection, the results in this paper indicated that adding the pixel-to-pixel loss improved latent space representation and image similarity, albeit for a small sample size and for a very specific dataset. More projections should be made and evaluated for this model as well as for a naturally distributed application, such as human faces. This will more accurately reveal the plausibility of this proposed loss function.

Finally, not stated previously in this paper, is that the StyleGAN2 models experienced partial mode

collapse, in which the generator learned some repeating modes that could fool the discriminator (i.e. because the discriminator could easily tell between real and fake images). This behavior can be explained by dataset limitations. The TBLLD contained around 65,000 high-variation images, which makes it difficult for the generator to produce realistic samples. While mirror imaging was used to increase the dataset to be 130,000 samples, partial mode collapse still occurred. To combat this, StyleGAN2 with Adaptive Discriminator Augmentation (ADA), can be trained [11]. ADA enables success with limited data by adaptively augmenting the discriminator's view of the data distribution, which requires the generator to produce indistinguishable samples from that augmented view. This technique achieved one of the best FID scores ever reported, but more importantly, it increases the breadth of GAN applications by reducing the need for large datasets [11]. Training on StyleGAN2 with ADA will not require any codebase changes and will likely have immediate positive impact. Additionally, most implementations in this project are model agnostic (i.e. projection loss, automatic labeling, feature direction learning), thus, comparing the efficacy of different GAN architectures in regards to logo synthesis and latent space exploration is feasible and encouraged.

7. Acknowledgements

I would first like to thank Dr. Olga Russakovsky. Taking her Computer Vision class introduced me to machine learning, and it is thanks to her that I have achieved some depth of knowledge and passion in this field. I truly appreciated the advice and insights I was given by Dr. Russakovsky, as well as the other students, in COS IW07. I looked forward to attending seminar each week because of the wonderful community that was built. Finally, thank you to my partner, Brendan Houle, for accompanying me on this intellectual journey. Your constant encouragement pushed me to work hard and explore the limits of my knowledge, which resulted in quite a successful end product, but more importantly, wonderful memories.

8. Honor Code

This paper represents my own work in accordance with university regulations.

/s/ Arin Champati

References

- [1] A. Borji, “Pros and cons of gan evaluation measures,” 2018. Available: <https://arxiv.org/abs/1802.03446>
- [2] Y. Chen, Q. Gao, and X. Wang, “i{wgan}: an autoencoder {wgan} for inference,” 2020. Available: <https://openreview.net/forum?id=HJg6VREFDH>
- [3] I. J. Goodfellow *et al.*, “Generative adversarial networks,” 2014. Available: <https://arxiv.org/abs/1406.2661>
- [4] M. Heusel *et al.*, “Gans trained by a two time-scale update rule converge to a local nash equilibrium,” 2018. Available: <https://arxiv.org/abs/1706.08500>
- [5] E. Härkönen *et al.*, “Ganspace: Discovering interpretable gan controls,” 2020. Available: <https://arxiv.org/abs/2004.02546>
- [6] P. Isola *et al.*, “Image-to-image translation with conditional adversarial networks,” 2018. Available: <https://arxiv.org/abs/1611.07004>
- [7] J. Johnson, A. Alahi, and L. Fei-Fei, “Perceptual losses for real-time style transfer and super-resolution,” 2016. Available: <https://arxiv.org/abs/1603.08155>
- [8] A. S. Jonathan Luffarelli and H. Yang, “The visual asymmetry effect: An interplay of logo design and brand personality on brand equity,” 2018. Available: <https://journals.sagepub.com/doi/10.1177/0022243718820548>
- [9] M. M. Jonathan Luffarelli and A. Mahmood, “Let the logo do the talking: The influence of logo descriptiveness on brand equity,” 2018. Available: <https://journals.sagepub.com/doi/10.1177/0022243718820548>
- [10] T. Karras *et al.*, “Progressive growing of gans for improved quality, stability, and variation,” 2018.
- [11] T. Karras *et al.*, “Training generative adversarial networks with limited data,” 2020. Available: <https://arxiv.org/abs/2006.06676>
- [12] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” 2019. Available: <https://arxiv.org/abs/1812.04948>
- [13] T. Karras *et al.*, “Analyzing and improving the image quality of stylegan,” 2020. Available: <https://arxiv.org/abs/1912.04958>
- [14] C. Oeldorf and G. Spanakis, “Loganv2: Conditional style-based logo generation with generative adversarial networks,” 2019. Available: <https://arxiv.org/abs/1909.09974>
- [15] A. Sage *et al.*, “Logo synthesis and manipulation with clustered generative adversarial networks,” 2018 *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018. Available: <http://dx.doi.org/10.1109/CVPR.2018.00616>
- [16] D. Saxena and J. Cao, “Generative adversarial networks (gans): Challenges, solutions, and future directions,” 2020. Available: <https://arxiv.org/abs/2005.00065>
- [17] R. Smith, “An overview of the tesseract ocr engine,” in *Proc. Ninth Int. Conference on Document Analysis and Recognition (ICDAR)*, 2007, pp. 629–633. Available: <https://research.google/pubs/pub33418/>
- [18] Z. Wang *et al.*, “Image quality assessment: from error visibility to structural similarity,” *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004. Available: <https://ieeexplore.ieee.org/document/1284395>
- [19] H. Yu and S. Winkler, “Image complexity and spatial information,” in *2013 Fifth International Workshop on Quality of Multimedia Experience (QoMEX)*, 2013, pp. 12–17. Available: <https://ieeexplore.ieee.org/document/6603194>
- [20] H. Zou and T. Hastie, “Regularization and variable selection via the elastic net,” *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005. Available: <http://www.jstor.org/stable/3647580>

9. Appendix

The tool can be found at:

<https://colab.research.google.com/github/bnhoule/stylegan2-logo-tool/blob/main/FinalTool.ipynb>

Table 8: Randomly generated logos vs. index of good logos

Random	Good

Table 9: Visualizations for Color by RGB Percentage Contribution.

	Red	Green	Blue
Labeling			
Positive			
Negative			

Table 10: Visualizations for Color by HSV Maximum Quantity.

	Red	Green	Blue	Brightness
Labeling				
Positive				
Negative				

Table 11: Visualizations for Size Methods.

	Binary Threshold	Filled Corner	Hough Circles
Labeling			
Positive			
Negative			



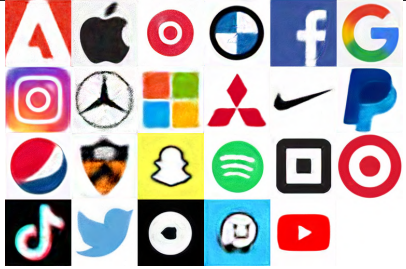
Table 12: Visualizations for Complexity and Symmetry Methods.

Complexity			Symmetry
	Blob Detection	Spatial Information	Binary Split
Labeling			
Positive			
Negative			

Table 13: Visualizations for Unsupervised Methods.

	Fill	Rigidity	Shape Shifter
Positive			
Negative			

Table 14: Visualizations for Projection Methods.

Original	
	
L1	L2
	
Perceptual	Perceptual + L1
