BrandOff: Content Obfuscation of Copyrighted Branding or Trademarks in Images Using StyleGANv2

Prajwal Jadhav[1], Vaibhavi Lambole[2], Ajay Pandita[3], Govind Rajput[4], and Sharmila Wagh[5]

[1, 2, 3, 4] *Dept. of Computer Engineering, M.E.S. College of Engineering, India*

[5] *Professor, Dept. of Computer Engineering, M.E.S. College of Engineering, India*

Abstract

Logo design is an essential aspect of branding that can be a powerful symbol of a company's identity. However, the use of logos is heavily regulated and using copyrighted logos without permission can lead to significant legal consequences. This has led to growing interest in developing techniques for generating logos that are both unique and legally permissible. In this study, the authors explore the use of StyleGAN v2, a trailblazing generative model, to generate logos. The authors propose an approach that involves training the StyleGANv2 model on a dataset of existing logos to generate new logos that are visually similar but legally permissible. The authors then use YOLOv5, a cutting-edge object detection model, to detect well-known logos in images before obfuscating them with the generated logos. The authors conducted experiments on a dataset of images containing copyrighted logos and compared the original images to images with obfuscated logos. The results suggest that using generated logos to obfuscate well-known logos can be an effective means of avoiding copyright infringement while preserving the visual consistency of images. This research has important implications for the field of logo design and copyright law and demonstrates the potential of generative AI and object detection for creating unique and legally permissible logos for various applications.

Keywords— *generative adversarial network; styleGAN; styleGANv2; yolov5; copyright obfuscation*

1. Introduction

Logo is a symbol composed of words, images, and colors used to recognize a brand or product. Logos come in different sizes and shapes that range from simple text logos to precise logo marks. Copyright allows an entity to protect obvious, original, and replicable works such as films, music, photographs, books, software code, and websites. If one breaks copyright law even by accent, one can face large fines. [1][2] The primary identification of any company is its logo, a simple graphical symbol that is unique, simply recognizable and especially useful in branding efforts.

Recent advances in generative artificial intelligence (AI) have made it possible to generate high-quality images that are visually indistinguishable from the real images. [3][4] One particularly exciting application of generative AI is in logo generation, where AI algorithms are used to create unique and visually appealing logos. By training generative models on large datasets of existing logos, new logos can be created that are both visually consistent with existing branding and uniquely distinctive. This approach can potentially save designers and companies significant time and effort in the logo creation process, while also producing logos that are more personalized and memorable for customers. [5] [6]

The authors were contacted by Textmover Labs[1] for sponsorship. Textmover labs is a computer vision and image processing startup based in Arizona, USA. They work on various image processing products and one of the major projects for them was dealing with copyrighted branding or logos in images and other multimedia. Textmover Labs was looking for an accurate and fast way to obfuscate trademarked symbols from the images. The authors were given the freedom to choose the techniques that feel appropriate as there was no compulsion to use machine learning or non-machine learning methods.

The YOLO model for detecting objects is novel compared to previous techniques. [7]–[9] While earlier methods repurpose classifiers to perform detection, YOLO tackles object detection as a regression problem through spatially separated bounding boxes and associated class probabilities. By using a single neural network to predict bounding boxes and class probabilities directly from full images in one evaluation, the entire detection process can be optimized end-to- end for detection performance as a single network.[7] Furthermore, YOLO has many variants that balance speed, accuracy and efficiency and these parameters can be varied according to needs, such as when it is needed to detect pedestrians in real- time. [10]

In this paper, the authors propose a way to remove copyright information from images. It is done by using the recent version of yolo, that is YOLOv5 and StyleGANv2. Section 2 provides the literature survey in the field of object and more specifically logo detection. Section 3 gives more information about the system architecture of YOLO. Section 4 illustrates the methodology followed to carry out logo generation and obfuscation. It provides the steps followed in the data pre-processing and model training. Section 5 analyzes the results from the experiment and visualizes them. Finally section 6 concludes the paper and gives directions for future work.

2. Literature Survey

A seminal paper by Goodfellow, Pouget-Abadie, Mirza, et al. [5] ”Generative Adversarial Networks” proposed a new method for generating synthetic data that involves training two neural networks in a game-like fashion. The first network, called the generator, learns to generate synthetic samples that are indistinguishable from real samples, while the second network, called the discriminator, learns to distinguish between real and fake samples. The two networks are trained simultaneously, with the generator trying to fool the discriminator and the discriminator trying to correctly identify the real and fake samples. Through this adversarial process, the generator gradually learns to produce more realistic samples, while the discriminator becomes more skilled at identifying fake samples. This paper sparked a surge of interest in generative models, particularly in the field of image generation, and has led to numerous extensions and applications of the original GAN framework.

Karras, Laine, and Aila [6] present a novel architecture for generative models, known as StyleGAN. The StyleGAN framework builds upon the original GAN formulation by introducing a disentangled representation of the latent code used to generate images. This disentanglement allows for greater control over the style and appearance of the generated images, as well as enabling a progressive training approach that gradually adds more complexity to the generated images. The StyleGAN architecture has been shown to produce high-quality synthetic images with impressive realism, and has been applied to a wide range of image generation tasks, including face synthesis, artwork generation, and even logo design. [6]

Building upon the original StyleGAN architecture, Karras, Laine, Aittala, *et al.* [3] introduced a number of improvements to its training and synthesis processes. The paper proposes a new method for mapping the disentangled latent code to the image space, known as adaptive instance normalization, which improves the stability and diversity of the generated images. In addition, the paper presents an extensive empirical analysis of the StyleGAN architecture, identifying key factors that affect its performance and proposing several modifications to improve its stability and image quality. The improvements proposed in this paper have since been incorporated into the latest version of the StyleGAN architecture, known as StyleGAN2, and have led to significant advances in the quality and diversity of generated images.

In their research, Mino et al. developed an improved Wasserstein generative adversarial neural network (with gradient penalty) that can produce logos based on a palette of twelve different colors. This is an early illustration of how artificial intelligence may support designers in their imaginative work. [11] Oeldrof et. al. demonstrated a method to enhance the StyleGAN architecture by introducing a conditional extension that aims to improve the quality of low-resolution results and provide greater control over the generated output using synthetic class-conditions. The paper explores techniques for obtaining class-conditions, a process that is complicated by the challenge of defining visual features unique to logos. The conditional style-based generator architecture is used in two experiments to train on the obtained class-conditions and compared to an unconditional model. The study discovered that although the unconditional model was more similar to the training data, the high-quality conditions resulted in more intricate features being incorporated into the latent space, resulting in a more diverse output. [12]

In their work, Yang et al. create a collection of icons encompassing eight distinct categories. They also implement cutting- edge research and training methods such as an attention mechanism and spectral normalization, building upon the original StyleGAN. These approaches demonstrate that the model is capable of generating excellent quality icons. [13]. IconGAN, a conditional generator with two dual discriminators with orthogonal augmentations, is introduced by Chen et al. In order to standardize the feature space of both discriminators, they also employ a contrastive feature disentanglement method. [14]

Redmon et al. introduced a groundbreaking approach to object detection called YOLO (You Only Look Once) in a highly influential paper published in 2016. [7] Since then, multiple variations of YOLO have been developed. Compared to classifier-based systems, YOLO offers several advantages. It analyzes the entire image during testing, allowing its predictions to be informed by the global context within the image. Additionally, it produces predictions with just one network evaluation, in sharp contrast to R- CNN and Fast R-CNN, which require thousands of evaluations for a single image. As a result, YOLO is remarkably fast, performing over 1000 times faster than R-CNN and 100 times faster than Fast R-CNN. [8]

Redmon et al. developed two real-time detection systems, YOLOv2 and YOLO9000. YOLOv2 is considered cutting-edge and outperforms other detection systems in various detection datasets. Moreover, it can operate at different image sizes to achieve a balance between speed and accuracy. YOLO9000, on the other hand, is a real-time framework that can detect over 9000 object categories by optimizing both detection and classification simultaneously.[15]

The small footprint of YOLO makes it uniquely suitable for real-time computer vision applications. Moreover, Liu et al and Zhu et al. have created some light-weight versions of yolo. [16] [17] They are even more resource efficient and inexpensive to run than the original version. They have been used for detecting drone captured images [17] as well as detecting tomatoes [16].

3. Data Availability Statement

For this paper we used a dataset generated by Oeldrof et. al. which is created by removing all text-based images from the LLD-logo dataset [18] and extending the remaining logos with image based logos and illustrations scraped off of Google images [12]. The dataset used in this research paper consists of a collection of images and is available via GitHub

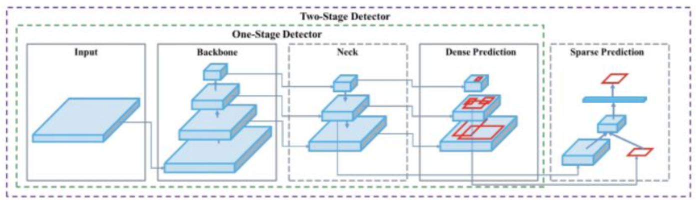

Figure 1: Anatomy of an object detector [20, Fig. 1]

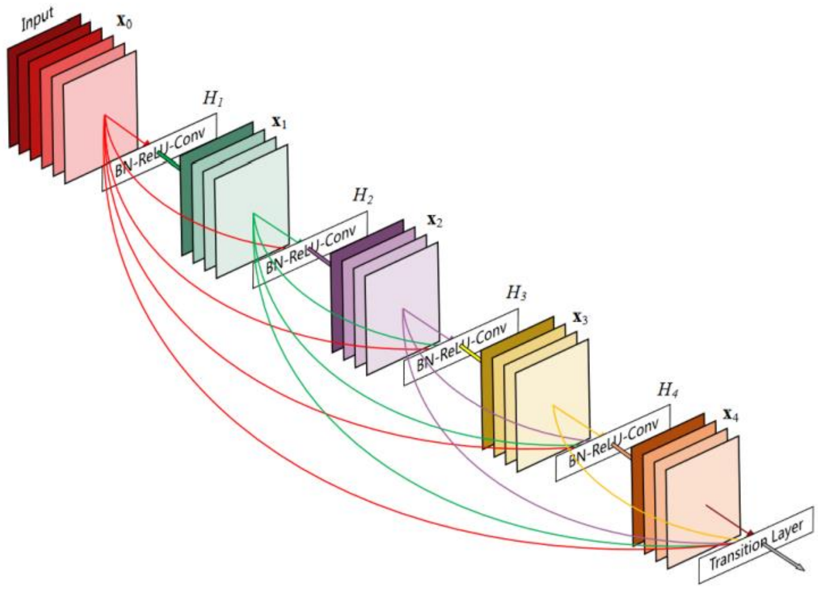

Figure 2: Layers in a DenseNet with connections between layers [22, Fig. 1]

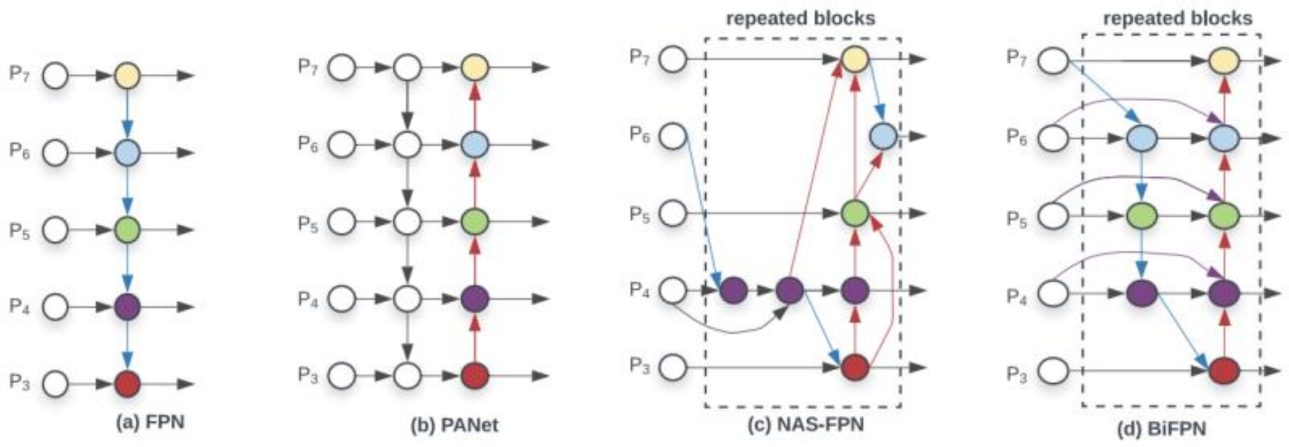

Figure 3: Feature network design. An additional bottom-up pathway is added on top of FPN [23, Fig. 2]

4. System Architecture

The architecture of our system can be divided into two distinct parts: logo detection pipeline based on YOLOv5 and logo generation component based styleGANv2.

4.1 YOLOv5 Architecture

The purpose of YOLOv5 is to perform object detection, which involves generating features from input images and using a prediction system to identify objects by drawing boxes around them and classifying them. The YOLO model was revolutionary since it was the first object detector to include bounding box prediction and class label prediction into an end-to-end differentiable network. [7][19]

Three essential parts make up the YOLO network: [20]:

1. **Backbone**: A convolutional neural network that joins and organizes different degrees of detail in picture data.
2. **Neck**: A group of layers used to combine and mix picture characteristics before sending them to the prediction step.
3. **Head**: Makes box and class predictions using the characteristics from the neck.

Both YOLOv4 and YOLOv5 employ the CSP Bottleneck technique to extract image features, which resolves problems with duplicated gradients in larger ConvNet backbones. This results in a smaller quantity of parameters and FLOPS, while preserving the same level of importance. This is crucial for the YOLO family, as rapid inference and a small model size are critical considerations. [7], [20]

The CSP models are founded on DenseNet, which aimed to link layers in convolutional neural networks with the following objectives: [21]:

- To mitigate the problem of the vanishing gradient, which arises when it’s difficult to backpropagate loss signals through a very deep network.
- To enhance feature propagation.
- To encourage the network to recycle features.
- To decrease the number of network parameters.

The PANet neck is a feature aggregation technique, also called the Path Aggregation Network, which is utilized in YOLOv4 and YOLOv5. It unifies multiple feature paths from the backbone network into a solitary feature representation, which is subsequently employed for object detection duties. The PANet neck has several benefits compared to traditional feature aggregation methods. For example, it allows for more efficient computation and faster inference times, while also providing a more robust and discriminative feature representation. Additionally, the PANet neck is designed to be flexible and easily adaptable to different network architectures, making it a versatile and powerful tool for object detection tasks. The PANet neck is a key component of the YOLOv4 and YOLOv5 object detection frameworks and plays a critical role in their performance and accuracy. The P_i mentioned in the below figure refers to individual feature layers in the CSP backbone. [23]

4.2 StyleGANv2 Architecture

Given that StyleGANv2 is built upon StyleGAN which in turn is built upon ProGAN, it is worth examining the ProGAN architecture first. ProGAN is composed of a Generator and Discriminator, as is typical in a GAN. The Generator generates images that appear to be ”real,” while the Discriminator distinguishes between real and fake images. Unlike a standard GAN, the ProGAN Discriminator analyzes the image at multiple scales. During training, the Discriminator takes in images at different resolutions, which are then combined to determine if the image is real or fake. Real images are subjected to a ”progressive downsampling process,” which produces lower-resolution versions of the original image. For instance, a 256x256 image is transformed into a list of images with sizes [256x256, 128x128, 64x64, 32x32, 16x16] and fed into the Discriminator. The Generator, on the other hand, creates images at the same resolutions and feeds them into the Discriminator. [24] The ProGAN architecture has demonstrated success in generating high- quality images, and StyleGAN has been built on this architecture to achieve an intuitive synthesis process. [6][24]

There are three changes in the StyleGAN Generator that make it different from ProGAN: A starting learnable constant, Mapping Network and Adaptive Instance Normalization (AdaIN) and Noise Injection for Stochastic Variation. [6]

4.2.1 Starting Constant

Instead of generating images from a latent code, StyleGAN starts with a constant learned image of size 4x4. This constant is gradually upscaled and new content is added to the image by applying style and noise at each block.

4.2.2 Mapping Network

The latent vector *z* is used to produce the style of the image. It is first sampled from a predefined distribution and then mapped into an intermediate latent space *W* to produce *w*. The mapping network, which is implemented using an 8-layer MLP, produces an affine transformation on the intermediate latent code w to generate style $y = (y_s, y_b)$. This style is then fed into the Adaptive Instance Normalization (AdaIN) layer, followed by a convolution to generate new contents for the image.

4.2.3 Adaptive Instance Normalization (AdaIN)

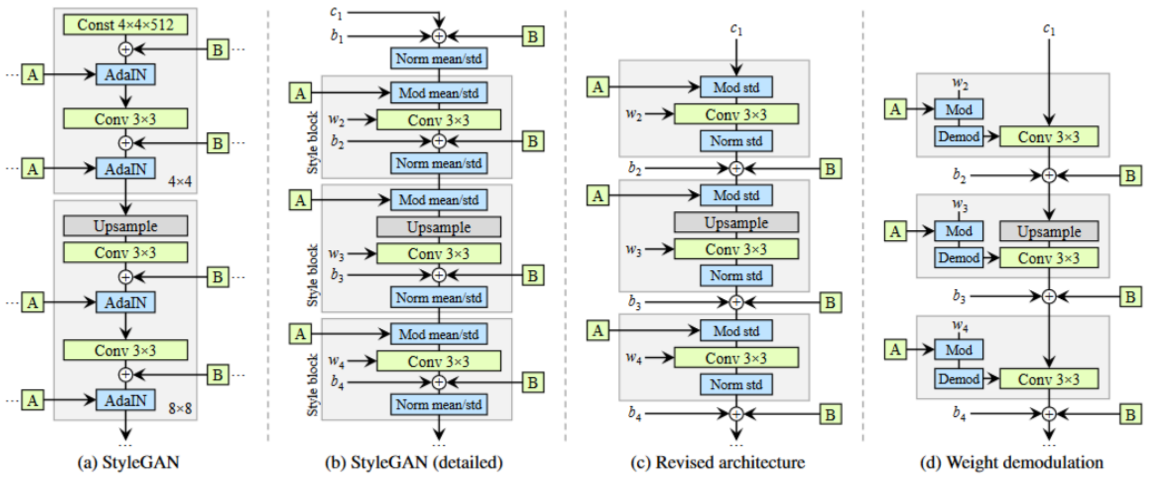

Figure 4: StyleGAN2 combines 3 modules into a single convolution [3]

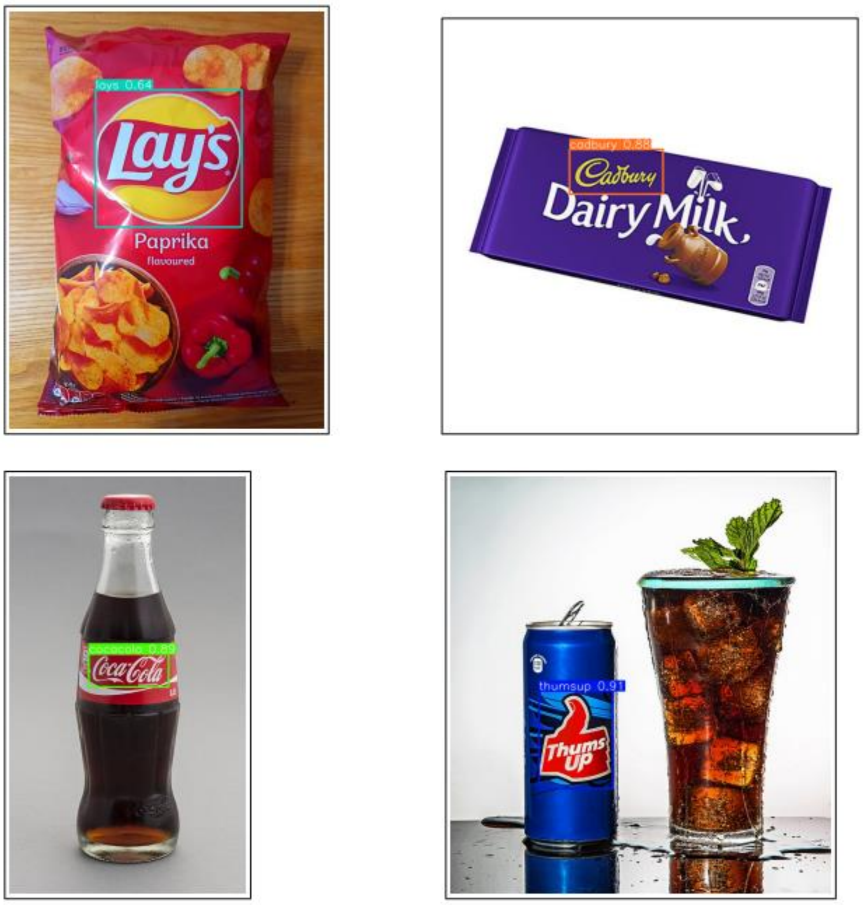

Figure 5: Results of detection of different logos using YOLOv5

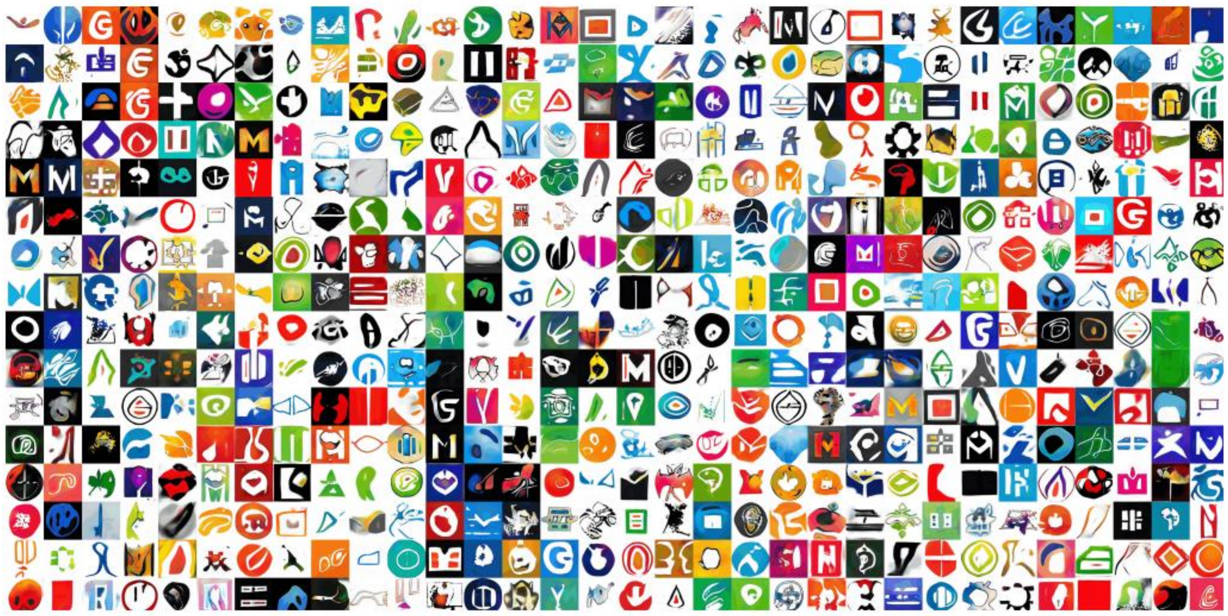

Figure 6: Sample 1 of generated logos after completion of training

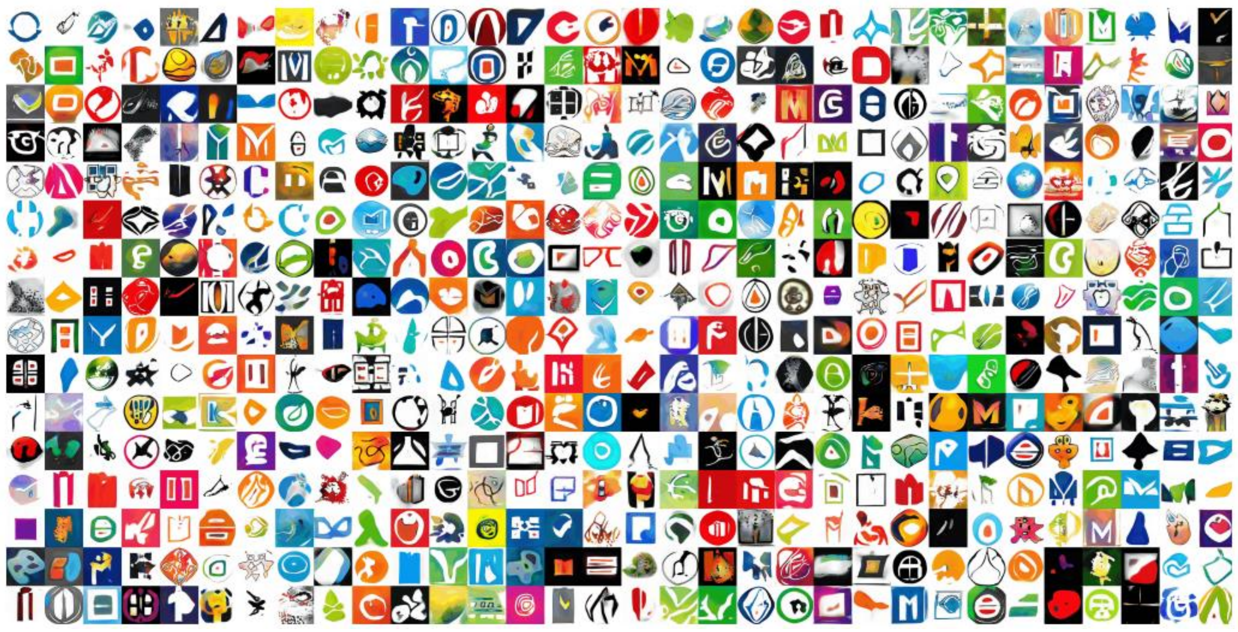

Figure 7: Sample 2 of generated logos after completion of training

The Mapping network produces the latent code ’*w*’, which is then passed through a learned affine transform and an AdaIN layer. To create a style, the affine transform employs two linear layers with scale = y_s and bias = y_b.

$$\mathrm{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i} \frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i}$$

Where x is the output feature map of the previous layer. The AdaIN first standardizes each channel x_i to have ”zero mean” and ”unit variance” and then applies y_s and y_b . This implies that the style y determines the statistics of the feature map for the next convolutional layer. In other words, y_s represents the standard deviation and y_b represents the mean. The style determines which channels will have a greater impact in the next convolution.

4.2.4 Stochastic Variation with Noise Injection

The StyleGAN Generator generates small details in an image, known as stochastic variations, which do not change the overall context of the image, such as hair placement, smile angle, stubble, and freckles. To generate such variations, Noise Injection is added to the StyleGAN generator before AdaIN layers. The noise added to the feature map has a small scale of variance compared to the feature map and zero mean. As a result, the overall context of the image is preserved, and the statistics of the feature map remain the same. This allows the Generator to learn how to generate new content on the given style while reserving its capacity to learn how to generate stochastic variations.

Although StyleGAN is a high-performing image generation model, it has a problem with artifacts in the generated images, which can become more pronounced at higher resolutions, particularly at 64x64 resolution. Research has identified two primary factors responsible for these artifacts: the architecture design of StyleGAN and Progressive Growing. [3][6]

In analyzing StyleGAN’s performance, researchers discovered that there was a flaw in the architecture design that caused artifacts to appear in the generated images. They believe that the Adaptive Normalization layers, which control the feature map statistics, unintentionally create these artifacts. These layers are an essential component of each block in the StyleGAN generator, allowing localized features within the network. However, they also put pressure on the generator to generate more details, which can result in information leaks to the next block, causing spikes that dominate the feature map statistics.

To address this issue, StyleGAN2 simplifies the architecture by baking the modulation and normalization components into a single convolution layer. This change relaxes the statistical restriction while removing the artifacts without affecting FID (Fréchet Inception Distance) scores. Additionally, StyleGAN2 removes progressive growth by simplifying the MSG- GAN architecture. [3]

5. Methodology

Just like the system architecture section this section is divided into two parts: first the authors give the methodology used to create a logo detector and then we state the method used for generating new logos.

5.1 Methodology for Logo Detection

The novel, high-performance YOLOv5 is used for detecting logos in images. First, a custom dataset using different brand images was created. 11 different logos from companies from various domains were selected. These companies are: Airtel, Amul, Cadbury, Capital One, Citibank, CocaCola, Dell, HSBC, Intel, Lays, LG, Pepsi, Red Bull and ThumsUp. The intention was to include logos from well-known companies from domains such as beverages, banking, electronics, etc. All the images were either .jpeg or .png format. In total, there were 157 images.

Next the images were uploaded to an online service called Roboflow. Roboflow is a web application which makes it easy to annotate images with bounding boxes.

After all the images are annotated, the images are pre-processed. Pre-processing is done to train, validate and infer the same image elements. Preprocessing has two main steps::

1. Auto Orient: Auto Orientation removes the EXIF data from the image so you can see the displayed image. They are saved to disk in the same way.
2. Resize : Resize resizes the image with the ability to scale to the desired set of dimensions. Annotations are scaled. With the Stretch to option, each image is 640x640.

Next, small additions were made to each image. Image magnification can increase the generalization of model performance by increasing the variety of training examples for the model. The following amplification methods were used: rotation and random noise.

1. Random Rotate: Randomly rotates the image clockwise or counterclockwise by a user-selected angle. The maximum image rotation angle was 3◦.
2. Random Noise: Adds salt-pepper noise to your image. Added up to 3% random noise.

After applying all these steps, additional images were generated and the total count of images became 409. The dataset was split into training, validation and test parts with 92%, 4% and 3% of total images respectively.

After generating the dataset using the above approach, the authors started training the model. As stated above, YOLO version 5 was used for training. The platform, Google Colab was used to train the model. The motivation behind choosing Google Collab was that it provides a free GPU and many people can work on the training process collaboratively.

The GPU used was NVIDIA Tesla T4 with 16GB GDDR6 memory. In total 12.7 GB RAM and

78.2 GB Disk space was available. Next, the dataset was trained on the YOLOv5 model with batch size 32 and epoch of 100. YOLOv5s was selected as the base model.

5.2 Methodology for Logo Generation

For logo generation, the state-of-the-art StyleGANv2 model by NVIDIA is used. It is based on generative adversarial network concepts. StyleGANv2 must be trained on a wide variety of images for it to produce unique and diverse logos. Finding an appropriate database is a crucial aspect of creating generative AI models. Luckily, a logo image dataset has been developed by Oeldrof et al, which is used to train our models for experiments. This dataset is derived from the LLD-logo dataset and has been preprocessed and augmented with extra images to better suit our problem domain. [12]

1. **Large Logo Dataset:** Sage, Agustsson, Timofte, et al. [18] initially presented the large logo dataset (LLD) in two varieties: LLD-icon and LLD-logo. LLD-icon has 32px favicons, while LLD-logo contains logos of up to 400px taken from Twitter. [18] LLD-logo has 122,920 logos, but a significant number of them only have text. Consequently, using optical character recognition, these images are removed from the dataset.
2. **Boosting:** Oeldrof and colleagues scraped Google to obtain fresh logo-like images. They employed keywords associated with subjects like nature, technology, and illustrated characters, which yielded about 15,000 more images to use for training data. [12]

Next, the authors pre-processed the dataset which included checking that all the images have 3 channels and transforming an image from 1 channel to 3 channels. Also, as StyleGANv2 requires all images to be of the same dimensions, all the images from the dataset were resized to 128x128 pixels.

Using this dataset, the StyleGANv2 model was trained on a machine with NVidia V100 GPU, 10 GBs of RAM and intel i7. The hyperparameters used were: batch size = 8, gamma value = 10, Generator learning rate = 0.0002, Discriminator learning rate = 0.0002. It took 2.5 days for the training to complete.

6. Results

The purpose of conducting these experiments and evaluations is to address the research questions regarding the ability to produce diverse logos and evaluating the performance of YOLOv5 on a custom logo dataset. This section gives the results and analysis of the two components of our architecture: YOLOv5 logo detector and logo generation using StyleGANv2.

6.1 Experimental Analysis of YOLOv5 Logo Detection

The purpose of this Results section is to present the findings of this research on logo detection using the YOLO (You Only Look Once) object detection algorithm.

Metric	**Score**
Precision	0.83
Recall	0.54
maP50	0.82

Table 1: Results of YOLOv5 logo detector model

Table 1 shows the evaluation of the YOLOv5 logo detector. The results indicate that the YOLO model performed well in detecting logos in a dataset of images.

As a test data, the authors ran the application of obfuscating logos and firstly detected the brand logo in the image. Figure 5 shows the result of detection of logos.

6.2 Evaluation of Logo Generation using StyleGANv2

6.2.1 Quantitative Evaluation

Assessing the performance of a GAN through quantitative measures can be challenging, especially for logo synthesis, as logo quality is a subjective metric. Fréchet Inception Distance (FID), which is commonly used in GAN-related papers, is a metric used to evaluate the quality of generated images. Mathematically it can be described as:

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

where μ represents the mean and ∑ the covariance of the embedded layer of both *x*, the training data, and *g*, the generated data. Table 2 shows the evaluation of the YOLOv5 logo detector.

Metric	Score
FID	186.8
Discriminator Loss	0.4724
Generator Loss	2.633

Table 2: Results of quantitative analysis of logo generation

6.2.2 Qualitative Evaluation

To evaluate the quality of the results subjectively, two aspects need to be examined: Image quality and diversity. Some of the samples of generated logos are shown in Figure 6 and Figure 7

Image Quality: The logos that are generated are consistently of high quality and appear to be stable. Most of the designs are simple, with the rare occurrence of logos.

Diversity: The generated logos show a lot of similarity in terms of shape and color with slight variations. Nonetheless, a handful of distinct logos stand out.

6.3 Results of Logo Obfuscation

As stated, the original intent of this research endeavor was to see how logo generation can be used to obfuscate logos and trademarks from images. The generator from the trained StyleGANv2 model can be used to generate new and unique logo images. These images then in turn can be used to

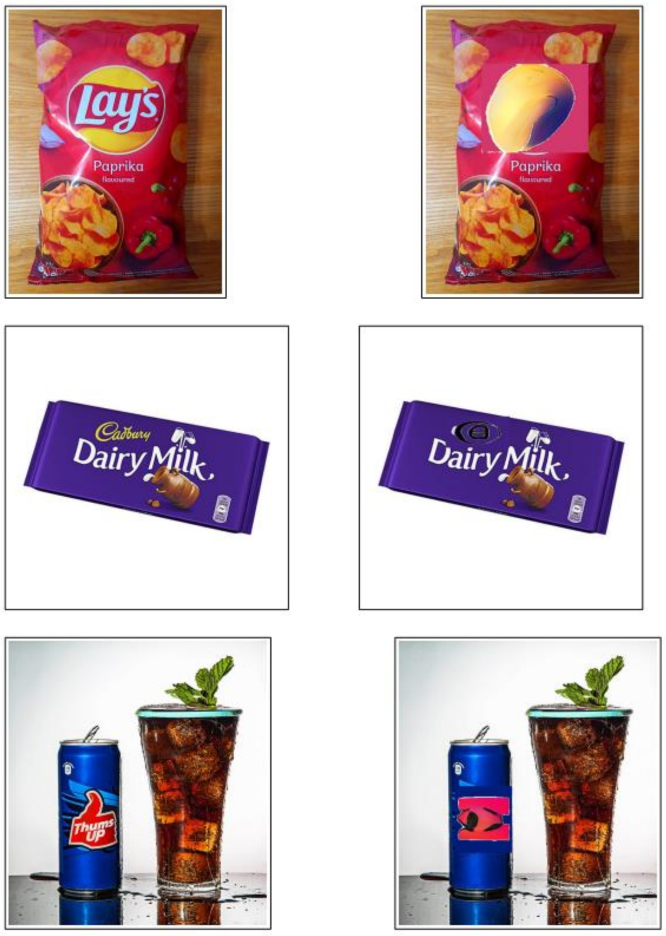

Figure 8: Original logos and their obfuscated counterparts, side-by-side.

obfuscate real logos from images. This process can be understood as three distinct steps:

1. The original logo (e.g. Pepsi logo) is detected using YOLOv5 and the co- ordinates of the bounding box are retrieved.
2. Using a pre-trained StyleGANv2 model logo is generated.

3. The region of the original logo in the image is replaced with this newly generated logo with the help of bounding box coordinates.

Python libraries openCV and Pillow are used to achieve this replacement of a region of an image using another image. The results of such obfuscation are shown in Figure 8.

7. Conclusion & Future Work

In this paper, the authors have explored the potential of using StyleGAN v2 to generate logos for the purpose of obfuscating well-known logos from images to avoid copyright infringement. It is also shown that the generated logos have a high degree of variability and can be used to create effective obfuscations of well-known logos.

The authors have also demonstrated the effectiveness of using YOLOv5 for detecting well-known logos from images before obfuscating them. This ensures that only the relevant logos are obfuscated, while leaving other parts of the image untouched. Our results suggest that this approach could be a valuable tool for individuals and companies who want to use images containing well-known logos without running afoul of copyright laws. The logo generation can also be used by artists for inspiration to create unique logos.

In the future, there are several potential directions for further work in this area. Firstly, the use of different GAN architectures can be explored, such as BigGAN or StyleGAN v3, to generate logos with even higher resolution and more realistic details. We can also investigate the use of other datasets or data augmentation techniques to improve the diversity and quality of the generated logos. Secondly, we can explore the use of the generated logos for other applications, such as logo retrieval or logo synthesis. Logo retrieval can be used to match a given logo with a similar logo from a database, while logo synthesis can be used to create new logos that combine the characteristics of multiple logos.

References

[1] E. N. Digital, “Adidas logo row: Luxury designer thom browne earns a stripe in trademark battle with adidas,” Times Now, Jan. 14, 2023. [Online]. Available: https:
[2] //www.timesnownews.com/business-economy / companies / adidas - logo- row-luxury-designer-thom-browne- earns - a - stripe - in - trademark - battle - with - adidas - article - 96987291 (visited on 01/14/2023).
[3] S. Butler, “Lidl wins high court case against tesco over blue and yellow logo,” The Guardian, Apr. 19, 2023. [Online]. Available: https://www.theguardian. com / business / 2023 / apr / 19 / lidl - wins - high - court - case - against - tesco- over- blue- and-yellow- logo (visited on 04/19/2023).
[4] T. Karras, S. Laine, M. Aittala, J. Hellsten,
[5] J. Lehtinen, and T. Aila, “Analyzing and improving the image quality of stylegan,” 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 8107–8116, 2019.
[6] A. Brock, J. Donahue, and K. Simonyan, “Large scale GAN training for high fidelity natural image synthesis,” in International Conference on Learning Representations, 2019. [Online]. Available: https://openreview.net/forum?id=B1xsqj09Fm.
[7] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, et al., “Generative adversarial nets,” in NIPS, 2014.
[8] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4396–4405, 2018.
[9] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” 2016 IEEE Conference on *Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.

[10] J. Redmon and A. Farhadi, Yolov3: An incremental improvement, 2018. DOI: 10. 48550 / ARXIV . 1804 . 02767. [Online]. Available: https : / / arxiv . org / abs / 1804.02767.

[11] A. Alaei and M. Delalandre, "A complete logo detection/recognition system for document images," in Proceedings - 11th IAPR International Workshop on Document Analysis Systems, DAS 2014, Apr. 2014.DOI: 10.1109/DAS.2014.79.

[12] W. Lan, J. Dang, Y. Wang, and S. Wang, "Pedestrian detection based on yolo network model," in 2018 IEEE International Conference on Mechatronics and Automation (ICMA), 2018, pp. 1547–1551. DOI: 10 . 1109 / ICMA.2018.8484698.

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

[14] C. Oeldorf and G. Spanakis, "Loganv2: Conditional style-based logo generation with generative adversarial networks," 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), pp. 462–468, 2019.

[15] H. Yang, C. F. Xue, X. Yang, and H. Yang, "Icon generation based on generative adversarial networks," Applied Sciences, 2021.

[16] Y. Chen, Z. Pan, M. Shi, H. Lu, Z. Cao, and W. Zhong, "Design what you desire: Icon generation from orthogonal application and theme labels," Proceedings of the 30th ACM International Conference on Multimedia, 2022.

[17] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," null, 2017. DOI: 10.1109/cvpr.2017.690.

[18] G. Liu, J. C. Nouaze, P. L. Touko Mbouembe, and J. H. Kim, "Yolo-tomato: A robust algorithm for tomato detection based on yolov3," Sensors, vol. 20, no. 7, 2020, ISSN: 1424-8220. DOI: 10 . 3390 / s20072145. [Online]. Available: https:// www.mdpi.com/1424-8220/20/7/2145.

[19] X. Zhu, S. Lyu, X. Wang, and Q. Zhao, Tph-yolov5: Improved yolov5 based on transformer prediction head for object detection on drone-captured scenarios, 2021. DOI: 10 . 48550 / ARXIV . 2108 . 11539. [Online]. Available: https : / / arxiv.org/abs/2108.11539.

[20] A. Sage, E. Agustsson, R. Timofte, and L. Van Gool, Lld - large logo dataset - version 0.1, https://data.vision.ee.ethz. ch/cvl/lld, 2017.

[21] T. Diwan, G. Anirudh, and J. V. Tembhurne, "Object detection using yolo: Challenges, architectural successors, datasets and applications," Multimedia Tools and Applications, Aug. 2022, ISSN: 1573-7721. DOI: 10 .1007 /s11042 - 022 - 13644 - y. [Online]. Available: https://doi.org/ 10.1007/s11042-022-13644-y.

[22] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," ArXiv, vol. abs/2004.10934, 2020.

[23] C.-Y. Wang, H.-Y. Mark Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "Cspnet: A new backbone that can

[24] enhance learning capability of cnn," in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2020, pp. 1571–1580. DOI: 10. 1109/CVPRW50498.2020.00203.

[25] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2261–2269. DOI: 10 . 1109 / CVPR . 2017.243.

[26] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in

2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2020, pp. 10 778–10 787. DOI: 10 . 1109 / CVPR42600 . 2020 . 01079. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR42600.2020.01079.

[27] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," ArXiv, vol. abs/1710.10196, 2017