Fine-Tuning Methods for Diffusion Models

Jan-Felix De Man[1][2750183]

Supervisor: Anil Yaman, Akshay Singh*

Vrije Universiteit Amsterdam, De Boelelaan 1105, 1081 HV Amsterdam

Abstract. In this study, we examine the efficacy of various fine-tuning methods for generative models with the goal of producing high-quality, specific object-oriented images. We specifically focus on Dreambooth, Textual Inversion, and Low Rank Adaptation (LoRA) fine-tuning methods for an image model fine-tuned to generate variations of a clothing brand's logo. Evaluations are performed using a custom scoring system that utilizes an object detector and Optical Character Recognition (OCR) system to evaluate how well the generated images match the desired prompts.

Our results indicate that the Dreambooth fine-tuning approach outperforms the others, displaying a 100% success rate in logo detection with an average confidence score close to 95%. Unfortunately, these results were imitations of the training set, since the overfitted model lost its ability to generate new images. Textual Inversion performs less effectively, demonstrating some conceptual understanding of the logo but failing to capture all the details accurately. The LoRA models score reasonably well but are notably less successful in generating an image with the logo consistently.

Our findings illustrate that fine-tuning a diffusion model is a tedious process, which often leads to a model that fails to generalize, or not learns at all. Results indicate that LoRA provides a good generalization at the cost of lower consistency.

1 Introduction

Image synthesis is a field with a lot of spectacular recent developments. Diffusion models, such as OpenAI's Dall-E 2 [13], allow users to generate an image from any text prompt. This ability will change the way individuals and companies look at graphic design, illustrations and pictures. Metyis, a consultancy with a drive for digital transformation, is interested in providing its clients with custom image generation models. These companies would then be able to generate their products within any kind of context, or to use these models to draw inspiration for new designs. In order to achieve this, Metyis would either have to train a

* Metyis, De Entree 69, 1101 BH Amsterdam

diffusion model of their own, or fine-tune a pre-trained diffusion model to include their desired item and style.

This work will focus on the latter. By using fine-tuning methods such as Dreambooth, Textual Inversion and Low Rank Adaptation, the objective is to establish an efficient fine-tuning approach that allows Metyis' clients to generate images, consisting mostly out of fashion items, within their specified style or to generate realistic images of their actual products. This could be of value by replacing human models in photo shoots, assisting in marketing campaigns, and to help in the creative flows of the designers. The following research questions will be addressed in this work:

1. How to evaluate the fine-tuning methods over a specific data set?
2. What are the characteristics of each fine-tuning method?
3. Which of these methods provide a generalizable and scalable solution?

It is hard to evaluate images that might only differ slightly. The stochastic nature of diffusion models, given that their generation process commences with random noise, further complicates the comparison between models, and the manual assessment involved presents a labor-intensive task. The methods of evaluation for such a task are still unclear. Common metrics like FID score are hard to apply to a limited data set and do not relate to our desired outcome. Therefore a new evaluation metric is proposed within the context of our data set. Since each of the methods are vastly different, finding their characteristics will give a lot of clarity to which method would be best for each use-case. Some methods are better at recreating the data set, while others excel at extracting elements from the data set and applying it to new contexts. It is important to decide on a relevant approach for each future use-case, as to not waste any valuable resources.

Finally, the goal of this work would be to provide a general approach that should work within most of the use-cases. In other words, the method that is able to extract and generalize over our training data will be selected. In the Related Work section one can find a small description and timeline of diffusion models in general, as well as an overview of the fine-tuning methods in their original context. The Methodology section goes over the diffusion processes and how the fine-tuning methods have been applied to Stable Diffusion in this work. The Experiment Design section will go over the data set used, the method of evaluation implemented in this work and the overall design of the experiment. The Results section presents the results of the experiment using our evaluation method. The Future Work section contains means of improving this work in future iterations while the Discussion section goes over the main points of discussion and concludes the work.

2 Related Work

Diffusion models are inspired by the laws of Non-equilibrium Thermodynamics, its first application in Deep Learning was proposed by Sohl-Dickstein et al. [20](2015). The idea behind is to retrieve the original state of an object, in this

case an image, after several changes have happened to it by means of a Markov chain. In the case of image synthesis diffusion models, the changes usually represent random Gaussian noise. This is called the noising process and happens stochastically. The denoising process then tries to retrieve the original picture from the fully noised picture. To achieve this, the model tries to predict which noise was added at every time step of the Markov chain. This method of unsupervised learning tries to estimate the mean and covariance for the Gaussian diffusion kernel. Originally, multi layer perceptrons were used for this task [20], but Ho et al. [4] (2021) introduced the use of UNet models [15].

As can be seen in Fig. 1, the UNet model consists out of several bottlenecks that aid with edge detection, these bottlenecks also have skip connections implemented to not lose any information. The denoising UNet model also contains encoder and decoder blocks, self-attention modules and sinusoidal time embeddings. The encoder-decoder structure is designed to capture hierarchical representation of the input data. Since not all regions have equal importance at every time step, self-attention mechanisms were introduced by Hu et al. [6]. These blocks perform two main operations: Squeeze and Excitation (SE). The squeeze operation reduces the spatial dimension by applying average pooling. The excitation operation applies fully connected layers to learn the importance feature of each channel and generate a channel-wise attention map. The input it receives is the input at time t of the denoising process and the time t itself.

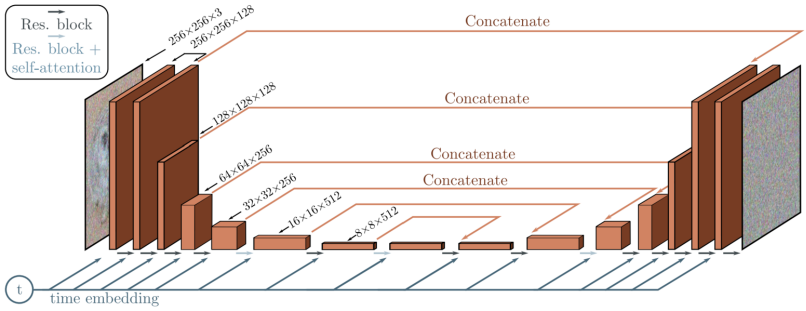

Fig. 1: UNet architecture for Diffusion Models

Classifier guidance, introduced by Dhariwal et Nichol [2] (2021), introduced conditional diffusion models, which allow to guide the denoising process by adding a label as a conditional. This was later extended to Classifier-Free Guidance, first introduced in GLIDE [10] (2022), allowing diffusion models to take in any text input as a conditional to guide the generation process.

Efficient fine-tuning of large neural networks is an increasingly popular research area. Since the rise of extremely large model architecture, such as LLMs or large diffusion models, there is more and more demand for transfer techniques to use these models for smaller and more specific tasks.

In the case of diffusion models, a popular approach for fine-tuning UNet architectures, known as *Dreambooth* as introduced by Ruiz et al. [16], has gained significant attention. The primary goal of Dreambooth is to incorporate new subjects into the model with minimal reliance on additional training images. This is achieved by embedding a new (unique identifier, subject) pair into the diffusion models' dictionary. Any uncommon token with a weak prior can serve as the unique identifier. Following the example of [16], let's assume $[V]$ as our chosen unique identifier. Consequently, the prompts for fine-tuning become as simple as *A* $[V]$ *dog*. The model (with frozen text encoder) is then trained on our data set. In order to prevent overfitting on the subject images, a class-specific prior preservation loss is introduced. This involves generating images of the subject noun (*dog*) using the original model. The paper is based on the Imagen diffusion model [18], which employs cascaded super-resolution diffusion models. First a 64×64 image is generated, which is later upsampled into a 256×256 image and then to 1024×1024.

A more efficient approach is the implementation of Low Rank Adaptation for Large Language Models (LLMs). Inspired by the observations of Li et al. [8] that the learned over-parameterized models reside on a low intrinsic dimension, Hu et al. [5] hypothesised that the change in weights during model adaption also has a low intrinsic rank. Full fine-tuning of LLMs can be described as updating the pre-trained weights Φ_0 to $\Phi_0 + \Delta\Phi$ by means of the conditional language modeling objective:

$$\max_{\Phi} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log(P_{\Phi}(y_t|x, y_{<t})) \tag{1}$$

It is important to note that Φ_0 and $\Phi_0 + \Delta\Phi$ have the same dimension, which, for some models, can be extremely large and thus very expensive. Not only does training consume a lot of resources, the fine-tuned model is also of large size and thus storage and deployment of such models can be challenging. That is why a parameter-efficient approach was proposed, where the task-specific parameter increment $\Delta\Phi = \Delta\Phi(\Theta)$ is further encoded by a smaller-sized set of parameters Θ with $|\Theta| \ll |\Phi_0|$. The objective can be rewritten as:

$$\max_{\Phi} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log(P_{\Phi}(y_t|x, y_{<t})) \tag{2}$$

where the trainable parameters $|\Theta|$ can be as small as 0.01% of $|\Phi_0|$. In practice, when considering a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, it is represented as $W_0 + \Delta W = W_0 + BA$ where $B \in \mathbb{R}^{d\times k}$, $A \in \mathbb{R}^{r\times k}$ with rank $r \ll \min(d, k)$. During training, W_0 is frozen and does not receive gradient updates, while A and B contain trainable parameters. The rank r thus represent the low intrinsic dimension, which is a parameter that can be tuned depending on the complexity of your task.

The key advantages of this approach are that these trainable parameters $|\Theta|$ are

smaller in size, allowing easy sharing and even swapping for different subtasks. The computational requirements for training these weights is considerably lower and there is no additional latency for inference when using these adaptations.

Textual Inversion, a method introduced by Gal et al. [3] aims to place new words into the tokenizer of the text embedding model used by diffusion models. In order to do this, a new pseudo-word is introduced, denoted by S_*, which can then be used to compose novel text prompts. In conditional image generation using diffusion models, during inference time the first stage usually consists out of creating a numerical embedding for our text input. The tokenizer splits the words into tokens which get mapped to a dictionary, where each entry has it's own vector that embeds its meaning. These embeddings are usually learned as part of the training process. Textual Inversion adds a new token S_* to the tokenizer and freezes the rest of the model, including the other parts of the text encoder. In this fashion, it tries to find the optimal embedding for our token S_*, such that the model would learn interpret said token and create new images that have this concept. A diffusion model can be described as:

$$L_{DM} := \mathbb{E}_{y,\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta\left(z_t, t, c_\theta(y)\right)\|_2^2\right], \tag{3}$$

where t is the time step, z_t is the noise at time t, ϵ is the unscaled noise sample, and ϵ_θ is the denoising network. The optimization goal for Textual Inversion likewise becomes:

$$v_* = \text{argmin}_v \mathbb{E}_{y,\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta\left(z_t, t, c_\theta(y)\right)\|_2^2\right] \tag{4}$$

The same training scheme for training diffusion models can thus be used, with the difference being that both c_θ and ϵ_θ are frozen and thus their parameters not trained. This allows to fine-tune diffusion models, without distorting output for other text prompts. In other words: if S_* is not called, the added token would not influence the output.

3 Methodology

Diffusion models have a couple of processes that are important to understand in order to fully grasp the effect of the fine-tuning methods. In this section the mathematical workings of the diffusion processes will be explained, as well as how the fine-tuning methods were implemented, and in the case of LoRA, were adapted to use for diffusion models.

3.1 Diffusion Processes

The diffusion process consists out of two main steps, the noising process followed by the denoising process. In the physical context, diffusion would only relate to the noising process, but in the context of generative deep learning, the key is to restore the original asset from a completely diffused object. The reverse diffusion

process is therefore very powerful, as a completely diffused object could be as much as, or as little as, random noise. Drawing random noise is an easy task, therefore the denoising module in diffusion models are often referred to as the generator, as it can generate assets out of practically nothing.

Forward Diffusion Process The noising process, or the forward diffusion process, of a data point sampled from a real data distribution $x_0 \sim q(x)$ can be defined as adding a small amount of Gaussian noise to the sample in T steps, producing a sequence of noisy samples $x_1, ..., x_T$. The step sizes are controlled by a pre-defined variance schedule $\{\beta_t \in (0,1)\}_{t=1}^{T}$, such that:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \tag{5}$$

$$q(x_{1:T}|x_0) = \Pi_{t=1}^{T} q(x_t|x_{t-1}) \tag{6}$$

This can be more easily described, in the setting of image generation, by adding a small noise to every pixel for T steps. The multiplication represents the odds that one would end up with that certain noised image.

Reverse Diffusion Process The noising process can be reversed, thus sampling from $q(x_{t-1}|x_t)$, reconstructing the true sample from a Gaussian noise input $x_T \sim \mathcal{N}(0, I)$. In other words, this process tries to turn the noised image back into the input picture. $q(x_{t-1}|x_t)$ can only be estimated by fitting it on the entire data set, which is infeasible. We need to learn a model p_θ to approximate these conditional probabilities in order to perform the reverse diffusion process.

$$\mathrm{p}_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \textstyle\sum_\theta(x_t, t)) \tag{7}$$

$$p_\theta(x_{0:T}) = p(x_T)\Pi_{t=1}^{T} p_\theta(x_{t-1}|x_t) \tag{8}$$

A reparameterization trick of the forward diffusion process, in the form of:

$$x_t = \sqrt{a_t}x_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1} \tag{9}$$

$$\sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\epsilon}_{t-2} \tag{10}$$

with $\alpha_t = 1 - \beta_t$ and $\bar{\alpha} = \Pi_{i=1}^{t}\alpha_i$, allows us to get to the final step with one calculation. Stochastic differential equations are typically *ill-posed*, meaning that a solution might not exist or is not unique, which makes them practically intractable. The reverse diffusion process is only tractable by assuming Gaussians at every time step, instead of combinations thereof. This assumption allows us to only estimate the mean and variance of the Gaussian distribution.

In other words; if the denoising process is described as:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sum_\theta(x_t, t)) \tag{11}$$

then we would like to learn μ_θ to predict $\tilde{\mu} = \frac{1}{\sqrt{(\alpha_t)}}(x_t - \frac{1-a_t}{\sqrt{(1-\bar{a}_t)}}\epsilon_t)$.
Since x_t is available as input at training time, we can reparameterise the Gaussian noise term instead; meaning that instead of predicting the Gaussian distribution at every step, we predict the added noise. This simplified loss, introduced by Ho et al. [4], not only simplifies the process, but also leads to better results and faster convergence.

A preliminary objective In VAEs you have ELBO loss, a bound on the true log likelihood:

$$-L_{\text{VAE}} = \log p_\theta(x_0) - D_{\text{KL}}(q_\phi(z|x)) \leq \log p_\theta(x) \tag{12}$$

Apply to diffusion:

$$-\log p_\theta(x_0) \leq \mathbb{E}_{q(x_{0:T})}[-\log \frac{p_\theta(x_{o:T})}{q(x_{1:T}|x_0)}] = L_{LVB} \tag{13}$$

$$L_{LVB} = L_T + L_{T-1} + \ldots + L_0 \tag{14}$$

where $L_T = D_{KL}(q(x_T|x_0))||p_\theta(x_T))$, $L_t = D_{KL}(q(x_t|x_{t+1}, x_0)||p_\theta(x_t|x_{t+1}))$ and $L_0 = -\log p_\theta(x_0, |x_1)$

3.2 Latent Diffusion Models

Latent Diffusion Models (LDMs), introduced by Rombach, Blattmann et al. [14], enabled training on limited computational resources while retaining the quality and flexibility, by applying them in the latent space of pre-trained auto-encoders. Encoder $\mathcal{E}$ and decoder $\mathcal{D}$ ensembles, as seen in Fig. 2, were first trained to compress an input image $x \in \mathbb{R}^{H\times W\times 3}$ into a latent representation $z \in \mathbb{R}^{h\times w\times c}$, such that $z = \mathcal{E}(x)$ and $\tilde{x} = \mathcal{D}(z) = \mathcal{D}(\mathcal{E}(x))$. The auto-encoder in Stable Diffusion has a reduction factor of 8. An image of shape $(3 \times 512 \times 512)$ becomes a latent of size $(3 \times 64 \times 64)$. This requires $8 \times 8 = 64$ times less memory [14]. Cross-attention layers were introduced into the model architecture, allowing powerful and flexible generation for conditional inputs, such as text or bounding boxes. This leads to high-resolution synthesis in a convolutional manner.
The rest of the architecture of Stable Diffusion, Fig. 2, consists out of UNet architecture as the denoiser, and a CLIP text embedding model to process the conditional input.

3.3 Fine-Tuning Methods

Since the scope of this work is limited in both time and resources, the fine-tuning will be done on the pre-trained LDM called Stable Diffusion [14], which is the open-source discussed above.

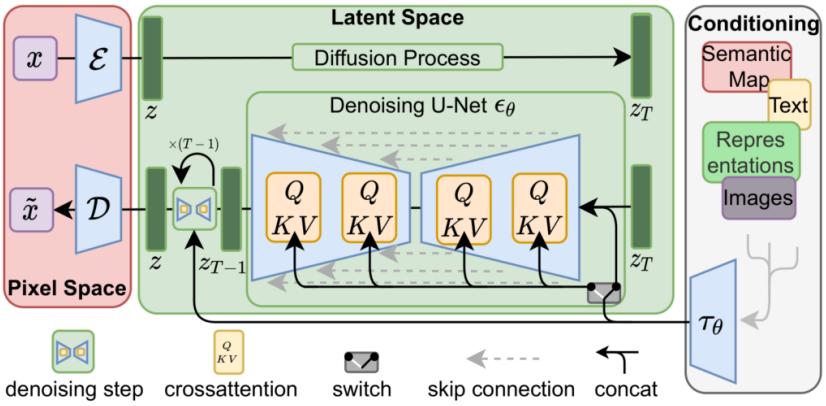

Fig. 2: Architecture of the LDM, with encoder $\mathcal{E}$ and decoder $\mathcal{D}$, and the attention condition implementation.

Source: [14]

Dreambooth Our implementation of the Dreambooth model in the context of Stable Diffusion involves a constrained approach wherein the majority of the model's parameters are held constant during training. Particularly, we only allow the parameters of the denoising UNet to undergo adaptation, which is named ϵ_θ in Fig. 2. In the scope of this project, no prior loss preservation mechanics were implemented. This will make this method prone to overfitting and might cause the model to take over characteristics of our data set, even when generating prompts that do not include our unique identifier. The output of this fine-tuning method are UNet weights fine-tuned on our data set, and have a size of 2.13GB. As outlined in section 2, our fine-tuning process employed a series of simple, identical prompts. The specifics of these prompts and the rationale behind their selection are further elaborated upon in section 4.3. A learning rate of $1e-5$ is recommended in [16], in this work we experimented with $1e-5$, $2e-5$ and $5e-5$.

Textual Inversion The implementation of Textual Inversion is by alternating the embedding for a certain token. In Stable Diffusion, this text encoder is the CLIP ViT-L/14 [11]. It is a model that measure the cosine similarity between images and their captions. In this context, CLIP, denoted as the condition layer in Fig. 2, is used to embed the text prompt such that it can serve as input for the denoising UNet. A learning rate of $1e-4$ is proposed in [3], we performed experiments with the learning rates $1e-3$, $1e-4$, $5e-4$, $1e-5$, $2e-5$ and $1e-6$.

Low Rank Adaptation In the context of diffusion models, Low Rank Adaptation (LoRA) is commonly employed in the UNet module of the network [17]. As

illustrated in Figure 1, this module comprises several building blocks. Previous work [5] suggests that achieving good results often involves tuning the attention layer. In LoRA, the attention layer is modified by introducing injectable weights of a specific dimension, denoted as r. Note that $B \in \mathbb{R}^{d \times k}$, $A \in \mathbb{R}^{r \times k}$, where d is the rank of the attention layer in which the weights will be injected. A higher r thus means more granular control over the influence of your LoRA embeddings.

The choice of r allows for fine-tuning the attention mechanism according to specific requirements and performance objectives. In this work, a lot of experiments were conducted in order to investigate the effect of the rank of the embeddings and the ideal learning rate. The models were trained in combination with Textual Inversion, with variable learning rates for the text encoder. Note that a low learning rate for the text encoder, thus applying Textual Inversion, has little result by itself, this process aids to give better embeddings to your new token. The hyperparameters that were experimented with in this work can be seen in Table 1. A total of $5^3 = 125$ LoRA models were trained.

unet_lr	**text_encoder_lr**	**network_dim**
5e-6	5e-6	4
1e-4	1e-4	8
1e-5	1e-5	16
5e-5	5e-5	32
1e-6	1e-6	64

Table 1: Grid Search Parameters for Fine-tuning using Low Rank embeddings.

4 Experiment Design

4.1 Data set

It was decided to stick to one specific logo for the scope of this project. This way, it would be possible to provide a POC that we could generate new clothing items with said logo or use it to create creative assets. To find a large enough data set, all images on the Hugo Boss product pages were scraped if the product name included *logo*. Out of the thousands of images that were gathered this way, 150 contained a red *HUGO* logo, which was picked to be our data set. The 49 images in which the logo was large enough, make up the data set for fine-tuning the diffusion model. Fig. 3 shows four pictures of the data set, with their corresponding labels. All the pictures have been hand-labeled and our unique token *red_hugo_logo* was in all of these labels. Since diffusion models work with square images, transparent margins were added to all images, which are ignored by our model's set-up.

Fig. 3: Four example pictures from the training data, with prompts *black shorts with a red_hugo_logo on it* (top left), *a black sweatshirt with a red_hugo_logo on the chest* (top right), *a male model wearing a black t-shirt with red_hugo_logo on it* (bottom left), *a black hat with a red_hugo_logo on the front*(bottom right).

4.2 Evaluation

The performance of image generation models is often measured using the Frechet Inception Distance (FID). This metric finds its root in the Inception Score (IS), introduced by Salimans et al. [19]. IS utilizes the Inception convolutional neural network [21] trained on ImageNet, for which generated images with meaningful objects should have low label entropy, meaning that they could only belong to a few object classes. Meanwhile there should be enough diversity, meaning that a high amount of object classes should be predicted. The pitfall is that IS does not take the training data into account. Often the data set contains objects not present in ImageNet, and the relation between the data set and the generated images is not captured. Therefore FID was introduced. The FID takes the Inception model's intermediate layer activations from both real and generated

images, assumes these activations are Gaussian distributed, and then calculates the Frechet distance between these two Gaussian distributions. The more similar these distributions are, the more similar the data distributions of real and generated images and thus the better the generative model is. The aim of this work is to achieve good output with limited data, as well as generating our one subject only, which is the reason why the FID score is discarded as a possible metric.

Since the nature of our task, we want our output to resemble the input, or the logo displayed in the input, as much as possible. Inspired by IS, an object detection model can be used to detect whether the generated images do in fact contain the logo. The 150 images with the our red logo described in 4.1, together with high-quality generated images from the first model and social media content portraying the red *HUGO* logo have all been labeled and used to train the YOLO v8 object detection model [7]. On top of that, several regularization images have been added in the form of badly generated logos. In this manner, the model became less prone to detecting badly generated logos, as often a red box was enough to be detected as one of our logos. Image augmentation was implemented using small rotation, zoom crops and brightness alterations. This increased the size of our data set from 210 images to 630 images. A final mAP score of 0.995 at a confidence threshold of 0.5 was achieved after training the model using the set-up above. As can be seen on the left-hand side of Fig. 4, the object detection model manages to recognize the logo in generated samples. From the images on the right, however, it is clear that it is prone to classifying object with faulty spelling, or even red-colored rectangles in general.

Experimentally, it was found that our object detection model is still very prone to classifying badly generated images when the spelling was similar, or worse, plain wrong. Therefore another metric for evaluation was implemented in the form of Optical Character Recognition (OCR). EasyOCR by JaidedAI [12] was used to recognise and evaluate the text generated in the images.
The final metric is a combination of the confidence score assigned by the object detector and whether correct spelling of HUGO was found in the picture. Concretely, with a threshold of 0.70, which experimentally we have found to be the best value, the maximum confidence score per picture was taken (if any logo was detected). The maximum is to ensure that if there was a good logo in the picture and a bad logo, we would not punish the model since one good logo was generated. Using OCR, a score of 0 or 1 was assigned per picture, indicating whether the word *hugo* was detected in the picture. Alterations in the form of capitalised letters and 0's instead of o's were also accepted.

4.3 Prompts

The prompts that were used during training differ slightly between Dreambooth and the other two methods. For the first, all images were labeled as *a red_hugo_logo logo*, in accordance to [16]. For LoRA and Textual Inversion, all the images were labeled by hand. All of the labels included our unique identifier, followed by the class noun: *red_hugo_logo logo*. Some examples can be seen on

Fig. 4: Four examples of the YOLO object detection on generated images.

Fig. 3. These labels serve as the text prompt during training. During the training process, images were sampled every 3 epochs, using the following prompts:

```
A red_hugo_logo
A male model wearing a blue red_hugo_logo sweater
A female model wearing a green red_hugo_logo t-shirt
The latest red_hugo_logo products
A billboard with the red_hugo_logo
The new red_hugo_logo fragrance perfume
```

5 Results

During training, the six prompts described in section 4.3 were sampled and used for evaluation after every three training epochs. For LoRA models, eight images per prompt (per three epochs) were generated. None of the pictures displayed

are cherry-picked.
According to our evaluation method, Dreambooth fine-tuning trumps the other models by a large margin, its best 2 models have the object detector detect the logo in every single sample, in our first model even with a average confidence score of almost 95%, as can be seen in Table 2. It also leads the charts following our OCR evaluation. In contrast, Textual Inversion model's samples have a very low OCR score, it's best model scoring only 2 out 6 correctly spelled images. Although the same model scored decently on the object detector evaluation, for which 5 out of the 6 images had a logo detected. The LoRA models score relatively well on the OCR evaluation, but their samples failed to generate a similar logo at a lower rate, compared to both Dreambooth and Textual Inversion. In the best samples, it detected 38 out of 48, which is a lot lower when compared to the 100% success rate using Dreambooth.

method	**parameters**	**average(ocr, yolo)**	**mean_confidence_score**	**mean_ocr_score**
DB	lr0_00002	0.724898001	0.949796001	0.5
DB	lr0_00005	0.706286823	0.912573646	0.5
DB	lr0_00001	0.634524385	0.602382104	0.666666667
LoRA	UNet5e-06TE0_0001dim16	0.594578506	0.709990345	0.479166667
LoRA	UNet5e-05TE5e-06dim16	0.587723781	0.696280895	0.479166667
LoRA	UNet0_0001TE0_0001dim8	0.550702949	0.622239232	0.479166667
TI	lr_0_001	0.545065025	0.756796718	0.333333333
TI	lr_0_0005	0.291357418	0.582714836	0
TI	lr_1e-06	0.21148403	0.42296806	0

Table 2: The best 3 models per fine-tuning method and the results of their evaluation. Dreambooth (DB), Low Rank Adaptation (LoRA) and Textual Inversion (TI) model's best epoch per parameter combination are displayed.

5.1 Dreambooth

The results of the Dreambooth fine-tuning approach depict our logo quite well. What can be seen in Fig. 5 is that the model tries to replicate pictures from the data set, instead of generating an image resembling the prompt. The first picture is very similar to one of the images of a sock in the data set, while the two people generated on the images to the right of it very much resemble human models in the data set images. The colors of the prompted clothing, a blue sweater for the second image, and green t-shirt for the third image, are displayed as black, which is the only color for these items in the data set. Intermediate checkpoints provided better results, as depicted in Fig. 6. Prompts that are still related to the data set, such as the color variations for the sweater and T-shirt, provide very good results. The rightmost picture, depicting the prompt *The new red_hugo_logo fragrance perfume*, clearly does not show perfume, but rather tries to replicate an image from the data set.

Fig. 5: Sample images of the Dreambooth model with learning rate 2e-5 at epoch 21. The prompts correspond to the prompts described in 4.3. Its evaluation is displayed in Table 2 as the first DB model.

Fig. 6: Sample images of the Dreambooth model with learning rate 1e-5 at epoch 12. The prompts correspond to the prompts described in 4.3. Its evaluation is displayed in Table 2 as the third DB model.

5.2 Textual Inversion

The samples relating to the results displayed in Table 2 as the best entry for TI, can be seen on Fig. 7. There are indeed two images that display the logo, or at least the text. It is clear that the model has an overall idea of what the logo should look like, but the white font in the leftmost picture, demonstrate the method fails to capture the details fully. Note that the data set only featured red logos with black font.

Fig. 7: Samples from the Textual Inversion model with learning rate 0.001 after 15 epochs of training.

From the models that were trained using textual inversion, only the one with learning rate 0.001 seemed to converge. Other learning rates lead to results that have a hint of the logos, at best. The second best samples, displayed in Fig. 8, show that the model was not yet able to successfully capture the logo.

5.3 Low Rank Adaptation

As can be seen in Table 3, according to our evaluation method, the best model has a high learning rate for the text encoder, meaning it performs textual inver-

Fig. 8: Samples from the Textual Inversion model with learning rate 0.0005 after 27 epochs of training.

Model	Epoch	Normalized_score	Mean confidence score	Mean ocr score
UNet5e-06TE0_0001dim16	3	0.976360172	0.709990345	0.479166667
UNet5e-05TE5e-06dim16	21	0.962657516	0.696280895	0.479166667
UNet5e-06TE0_0001dim16	6	0.92610986	0.724964845	0.416666667
UNet5e-05TE5e-06dim8	24	0.894806246	0.693645711	0.416666667
UNet0_0001TE0_0001dim8	3	0.888652542	0.622239232	0.479166667
UNet5e-05TE5e-06dim32	18	0.875298851	0.695878553	0.395833333
UNet5e-05TE5e-06dim64	27	0.869633995	0.690210889	0.395833333
UNet5e-06TE5e-05dim4	18	0.860923018	0.703245501	0.375
UNet5e-06TE0_0001dim16	9	0.852577357	0.607896071	0.458333333
UNet5e-06TE5e-05dim16	6	0.833038893	0.653597644	0.395833333

Table 3: The 10 best performing models according to our normalized score.

sion at a high learning rate and thus converges quite quickly (our best model at epoch 3). Note that our TI models with this learning rate did not even make it to the top 3. This makes it clear that a model can converge very quickly with a low learning rate for the LoRA embeddings and a high learning rate for the text encoder. It is hard to measure the actual impact of these learning rates separately, but when comparing **UNet5e-06TE5e-05dim16** with **UNet5e-06TE0_0001dim16** at epoch 6, (the third and last entry in Table 3), the higher learning rate for the text encoder improved the score for object detection with more than 7%, and it also detected correct spelling for 20 out of the 48 images, which is one more than for the model with a low learning rate for the text encoder. As can be seen in Table 3, a rank of $r = 16$ seems to trump the other models. When inspecting the training loss of **UNet5e-05TE5e-06dim16**, displayed in Fig. 9 and its samples in Fig. 10, it seems that the model starts overfitting after around 24 epochs. The logos become more distorted, mostly in the sense that the text becomes bolder, and faces start to deform. Similar patterns can be seen in other sampled images when a certain sweet spot has been passed. When comparing the rank of the LoRA embeddings, as seen in Fig. 11, it can be seen that the model overfits more easily on lower dimensional embeddings. This is especially clear when comparing the images for *A billboard with the red_hugo_logo*. In the lower ranks, the model started overfitting already, while in the highest rank, rank 64, it looks like the model is still learning the logo. The other pictures at the highest rank also do not show distortion, apart

Model	best_dim	2nd best	3rd best
UNet_lr5e-06TE_lr0_0001	16 (0.97)	8 (0.80)	4 (0.79)
UNet_lr5e-05TE_lr5e-06	16 (0.96)	8 (0.89)	32 (0.88)
UNet_lr0_0001TE_lr0_0001	8 (0.89)	4 (0.80)	16 (0.77)
UNet_lr5e-06TE_lr1e-05	64 (0.83)	16 (0.73)	8 (0.73)
UNet_lr5e-06TE_lr5e-05	16 (0.83)	64 (0.74)	4 (0.74)

Table 4: The 5 best performing learning rates according to our normalized score and their best ranks.

from where it is still learning how to generate the logo, but the human models are still realistic and of high quality.

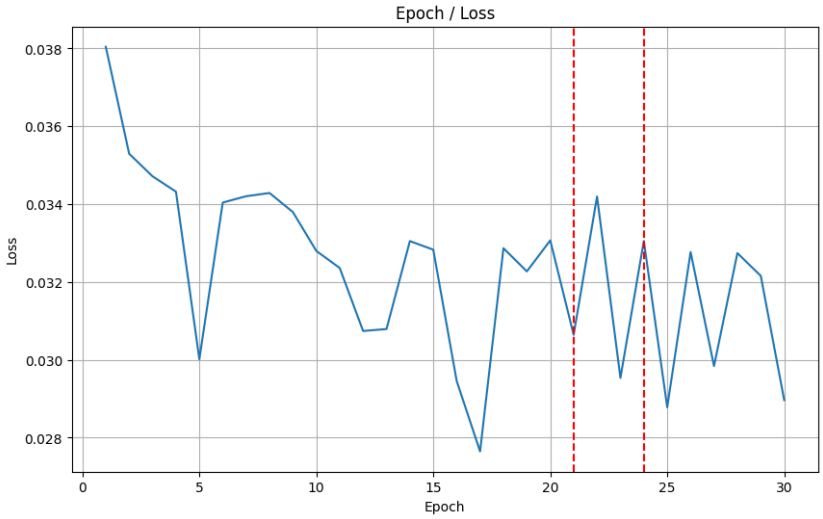

Fig. 9: The average training loss of model **UNet5e-05TE5e-06dim16** plotted per epoch. Red lines indicating the 21st and 24th epoch.

Fig. 10: Sampled images per 3 epochs of model **UNet5e-05TE5e-06dim16**. Every column corresponds to the prompts described in 4.3.

6 Limitations and Future Work

While it is shown that these fine-tuning methods can perform well for one data set, it is not clear whether these methods and parameters would work equally well on other data. Therefore, more experiments need to be carried out using other data sets in order to conclude these methods and findings are generalizable. Furthermore, prior loss preservation [16], adding more images to the data set of the same label (e.g. pictures of other logos), for Dreambooth fine-tuning could reduce the risk of overfitting. Although output for this method suggested that it recreated images from the training data with prompts that were not related to our logo at all. Experiments using inference methods such as Attend-and-Excite [1] and Composable Diffusion [9] have been carried out in order to increase promptual accuracy in the output.These inference methods could aid in guiding

Fig. 11: Sampled images after 30 epochs of model **UNet5e-05TE5e-06** with different ranks. Every column corresponds to the prompts described in 4.3.

the model better to follow the input prompt. This could aid in consistency, for LoRA models, and even reduce the effects of overfitting. The time limitation have diverted our focus towards the fine-tuning methods. Attend-and-Excite, however, delivered promising results at the expense of more compute. The evaluation method implemented in this work does not take the into account how the prompt relates to the output pictures, leading a highly overfit model to the best method according to our evaluation. A third evaluation metric could be added, such that these promptual failures would be accounted for. One such method could be to use the CLIP model to regress the image over possible prompts for each image [11]. If we take the example of the blue sweater, the input for the CLIP model could be the image, and the prompts in the form of *['a blue sweater', 'a black sweater', 'a white sweater', 'a red sweater']*. The model would compare the image for every prompt, and output the most relevant prompt. A score could then be calculated, either as a pass or fail, or using the confidence score of the correct output. This would allow to penalize the methods for promptual failures.

7 Discussion

Fine-tuning large neural networks is a delicate and hard assignment. The nature of diffusion models and their implementation of random noise, definitely do not facilitate that process. A second issue that arises is how to evaluate the output. As is clear from this work, even visually comparative methods do not regard for model overfitting and generalization.

Generally, since the whole denoising UNet is trained during Dreambooth fine-tuning, the model tends to overfit on the training data. The images resemble the data set with a high accuracy, but regardless of the prompt will recreate pictures from this data set. This leaves the model unusable for abstraction and

defeats the purpose of training such models. Regularization techniques, such as prior preservation loss could aid with this shortcoming, by adding picture that have the same label, but are visually different (e.g. other logos) to the training data. We believe, however, that this would still lead to models unable to produce vastly different images.

Textual Inversion is computationally cheap and converges fast over high learning rates. This is great for quick iterations of ideas and to grasp a general concept. On the other hand, it lacks detailed capturing of subjects, so this might not be the preferred fine-tuning method for every application. In the industry, it could serve a good role to help with imagery inspired by the data set, which could be very useful for aiding in creative processes.

LoRA fine-tuning seems to be in between those two methods. It can converge very quickly in combination with reasonable learning rate for the text encoder, and with the combined UNet embedding layer it provides more detailed results. The output is less consistent in actually generating a logo for every sample, but this seems to be a nice trade-off when compared to the scalability, expense of training and the quality of the output. The added weights to the model have enough impact to be able to faithfully generate the subject, while not being too dominant over the rest of the model weights which leads to a general model that is still able to create images vastly different to the data set.

An effective evaluation metric needs to be decided upon depending on the data set. Object detection is an acceptable evaluation method, but if the data is limited this might exclude this as an option. As mentioned in section 6, CLIP classification can be implemented for prompt-image comparison, but manual work is needed to come up with non-matching prompts. Other evaluation methods can be used depending on the nature of the data.

This work proves that you do not need a large amount of data in order to fine-tune a pre-trained model, great results were achieved with as little as 50 images. In summary, Dreambooth is a great approach if the desired output is very similar to the data available and there are not compute or memory limitations, Textual Inversion is able to capture a subject conceptually and requires less compute and has quick convergence, LoRA with the right training parameters offers a scalable solution without interfering with the general model's capabilities, at the expense of consistency.

References

1. Chefer, H., Alaluf, Y., Vinker, Y., Wolf, L., Cohen-Or, D.: Attend-and-excite: Attention-based semantic guidance for text-to-image diffusion models (2023)
2. Dhariwal, P., Nichol, A.: Diffusion models beat gans on image synthesis. CoRR **abs/2105.05233** (2021), https://arxiv.org/abs/2105.05233
3. Gal, R., Alaluf, Y., Atzmon, Y., Patashnik, O., Bermano, A.H., Chechik, G., Cohen-Or, D.: An image is worth one word: Personalizing text-to-image generation using textual inversion (2022). https://doi.org/10.48550/ARXIV.2208.01618, https://arxiv.org/abs/2208.01618
4. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. CoRR **abs/2006.11239** (2020), https://arxiv.org/abs/2006.11239

5. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: Lora: Low-rank adaptation of large language models (2021). https://doi.org/10.48550/ARXIV.2106.09685, https://arxiv.org/abs/2106.09685
6. Hu, J., Shen, L., Albanie, S., Sun, G., Wu, E.: Squeeze-and-excitation networks (2019)
7. Jocher, G., Chaurasia, A., Qiu, J.: YOLO by Ultralytics (Jan 2023), https://github.com/ultralytics/ultralytics
8. Li, C., Farkhoor, H., Liu, R., Yosinski, J.: Measuring the intrinsic dimension of objective landscapes. CoRR **abs/1804.08838** (2018), http://arxiv.org/abs/1804.08838
9. Liu, N., Li, S., Du, Y., Torralba, A., Tenenbaum, J.B.: Compositional visual generation with composable diffusion models (2023)
10. Nichol, A., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., Chen, M.: GLIDE: towards photorealistic image generation and editing with text-guided diffusion models. CoRR **abs/2112.10741** (2021), https://arxiv.org/abs/2112.10741
11. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning transferable visual models from natural language supervision (2021)
12. Rakpong, K.: Jaided AI: EasyOCR demo, https://www.jaided.ai/easyocr/
13. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., Chen, M.: Hierarchical text-conditional image generation with clip latents (2022)
14. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. CoRR **abs/2112.10752** (2021), https://arxiv.org/abs/2112.10752
15. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. CoRR **abs/1505.04597** (2015), http://arxiv.org/abs/1505.04597
16. Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M., Aberman, K.: Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation (2022). https://doi.org/10.48550/ARXIV.2208.12242, https://arxiv.org/abs/2208.12242
17. Ryu, S.: Low-rank adaptation for fast text-to-image diffusion fine-tuning. https://github.com/cloneofsimo/lora
18. Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S.K.S., Ayan, B.K., Mahdavi, S.S., Lopes, R.G., Salimans, T., Ho, J., Fleet, D.J., Norouzi, M.: Photorealistic text-to-image diffusion models with deep language understanding (2022)
19. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans (2016)
20. Sohl-Dickstein, J., Weiss, E.A., Maheswaranathan, N., Ganguli, S.: Deep unsupervised learning using nonequilibrium thermodynamics (2015). https://doi.org/10.48550/ARXIV.1503.03585, https://arxiv.org/abs/1503.03585
21. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions (2014)