Corso di Laurea Magistrale in Informatica

Text-to-image diffusion models attacks in scarce resource environments and privacy issues

Relatore:

Prof. Stefano Ferrari

Laureando:

Federico Fiorio

Matricola: 08276A

Anno Accademico 2022/2023

dedicated to . . .

dedica

Contents

1 Introduction **9**

2 Background **11**

2.1 Text-to-image models 11

2.1.1 Autoencoders 12

2.1.2 Generative Adversarial Network (GAN) 14

2.2 Diffusion models 15

2.2.1 DDPM 15

2.2.2 Conditional models and guidance 21

2.2.3 U-net 23

2.2.4 DDIM 24

2.2.5 Latent diffusion models 26

2.3 Attacks on Diffusion models 28

2.3.1 Shadow models configuration in membership inference attack . . 29

2.3.2 Overfitting, sufficient but not necessary 31

2.4 Attack models 34

2.4.1 Support vector machine 34

2.4.2 Logistic regression 36

2.4.3 Multi layer perceptron with softmax 37

2.5 Differential privacy 38

3 Technologies and instruments **42**

3.1 Attacked models 42

3.1.1 Stable-diffusion v1-5 42

3.1.2 karlo v1 alpha 44

3.1.3 Dreamlike photoreal 2.0 45

3.1.4 Differentially private diffusion models 46

3.2 Features extracting model 46

3.3 Image captioning model 49

3.4 Huggingface 51

3.5 Python 52

3.6 Google colab 53

3.7 Sklearn 53

3.8 Datasets used 54

3.8.1 LAION 54

3.8.2 VGG-face 56

3.8.3 MS-COCO 56

3.8.4 CC12M 57

3.9 Downloading images and file formats 58

4 Experiments and innovations **59**

4.1 Main papers presentation 59

4.1.1 Membership Inference Attacks Against Text-to-image Generation Models 60

4.1.2 Dataset disclosure 62

4.1.3 Membership Inference Attacks From First Principles 63

4.2 Main flaw of MIA Against Text-to-image Generation Models paper . . . 65

4.3 Dataset search 65

4.3.1 Why the dataset can't be downloaded in its completeness 66

4.4 Innovations 67

4.4.1 No background faces MIA 68

4.4.2 Prompt attack 70

4.5 Implementation 71

4.6 FID 72

4.7 Privacy and DPDM 73

4.7.1 Bad setup for attack 74

4.8 Prevent attacks and further work 74

5 Results **77**

5.1 Stable-diffusion, LAION5v+ vs MS COCO, 200, 1000, 2000 images, attack II-S 78

5.2 Karlo v1 alpha, II-S attack, 1000 images used, CC3M vs MS COCO 80

5.3 Dreamlike photoreal 2.0, II-S attack, 1000 images used, LAION5v+ vs MS COCO 81

5.4 Stable-diffusion privacy attack (profile photos) with BG, 1200 images, LAION5v+ face vs VGG face 82

5.5 Stable-diffusion, privacy attack (profile photos) with NO BG, 1200 images, LAION5v+ face vs VGG face 83

5.6 Stable-diffusion, prompt attack 2000 images, LAION5v+ vs MS COCO 84

5.7 FIDs 85

5.8 Comparison with state-of-the-art 87

6 Conclusions **88**

Chapter 1

Introduction

This thesis wants to present possible membership inference attacks (MIAs) that can be done on big text-to-image diffusion models with no particular hardware or software constraints.

The state-of-the-art MIAs are currently more focused on creating shadow models and achieving results based on accuracy. In this thesis, I explore another state-of-the-art MIA not related to shadow models.

This attack was first introduced in Membership Inference Attacks Against Text-to-Image Generation Models [13].

I applied more consistent metrics than accuracy to it, as stated in Membership Inference Attacks From First Principles [33].

I also explored different ways to change and improve this attack; in particular, I noted that the attack is very efficient when background is removed from profile photos, being able to achieve a TPR of 92% with a FPR of 0.1%.

I also developed a new MIA, still based on the paper [13] but that retrieves prompts instead of images from the training set.

I also addressed how privacy is protected with the use of differential privacy and why it is not very widely used in today's solutions. I also gave new ideas for future work in order to implement a more consistent method against MIAs.

The thesis structure consists of a background on MIAs and diffusion models and the analytical and mathematical fundamentals in order to have a complete knowledge of the topic. This will explore other types of text-to-image techniques, more classical, and I will make clear what the architecture of diffusion models is and what the difference is between denoising diffusion probabilistic models (DDPMs), denoising diffusion implicit models (DDIMs), and latent diffusion models (LDMs).

I will explain how MIAs are performed, why they are performed, and which models are used to perform them.

After the background part, I will introduce the technologies and methodologies used, as well as the datasets.

The following part consists of the experiments that have been conducted and the various innovations.

To conclude, I will present the results and how they are related to state-of-the-art ones.

Chapter 2

Background

In this chapter I will talk about what there is behind diffusion models, how we got there and the mathematical resoning behind them.
I will explain text-to-image models as well as other concepts, like differential privacy and in general everything that is needed to better understand the experiments done.

2.1 Text-to-image models

Text-to-image models are a class of machine learning models that aim at generating images by describing text as input.
The main models used today for generating images are diffusion models, generative adversarial networks (GANs) [1], and variational autoencoders (VAEs) [2].
This thesis focuses on diffusion models, but I will simply explain how VAEs and GANs work and why they are becoming less used than diffusion models.

2.1.1 Autoencoders

Autoencoders [3] are composed of two main components: an encoder that maps data from a high-dimensional space to a lower latent space, and a decoder that maps data from the latent space back to the input space.
An intuition can be that if the autoencoder becomes good at reproducing input data, then it is possible to say that the model understood the intrinsic meaning of data, and thus the lower-dimensional latent space (which is the output of the encoder) is a good representation of data.
VAEs derive from autoencoders, the main difference between these two is that VAEs apply a stochastic process to generate the output, while autoencoders are deterministic.
The main reason autoencoders were introduced in the literature is to represent high-dimensional data with a low set of features, or, in other words, to compress data.
The encoder part of the architecture can be used alone to reduce the dimensionality of data, in this case, the decoder can be discarded.
On the other hand the decoder part of the architecture can be used without the encoder in order to generate data.

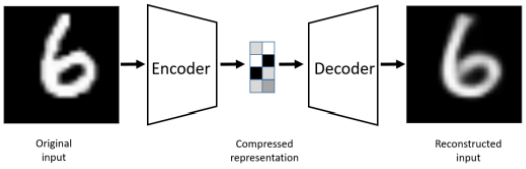

Figure 2.1: architecture of an autoencoder [3]

The main reason autoencoders do not sample well (generation) given a latent variable is that their latent space is not isotropic, and thus it's difficult to get good samples from it.
With VAEs, this latent space becomes closer to an isotropic space.
This is possible due to the stochastic process implemented in VAEs.
The encoder still compresses data, but the target is no longer a lower-dimensional space; now it is a variational distribution (multivariate gaussian distribution).
Typically, the variational distribution needs to be a unit normal (0 mean and 1 standard

deviation) so it is easier for the decoder to generate new samples.

The output of the encoder is given as input to the decoder, which will then map from the latent space to the input space (as in the autoencoder architecture).

VAEs have more stable training than GANs, but they produce worse results compared to them.

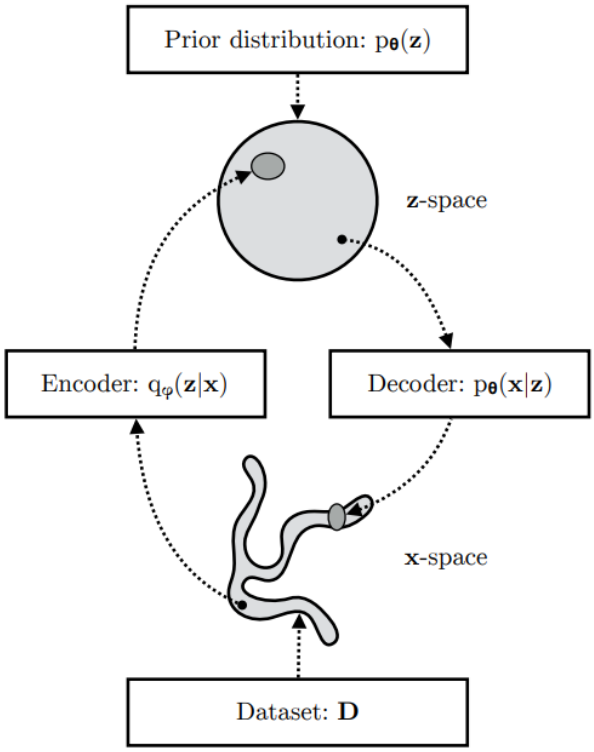

Figure 2.2: Image from the paper [4], a VAE learns the mapping betweem the x space, which is usually a complicated distribution and a latent space, z-space, which is usually a simpler distribution (sphere in the image).

The generative model learns a joint distribution $p_\theta(x,z) = p_\theta(z)p_\theta(x|z)$ with a prior distribution over latent space $p_\theta(z)$ and a stochastic decoder $p_\theta(x|z)$.

The stochastic encoder $q_\phi(z|x)$ is used instead of computing the intractable posterior $p_\theta(z|x)$ of the generative model

2.1.2 Generative Adversarial Network (GAN)

GAN is a machine learning architecture composed of two neural networks: the generator and the discriminator.

The architecture is trained as a zero-sum game, where the gain of one agent means the loss of the other.

The generator has the objective of creating photorealistic images to fool the discriminator, which tries to distinguish between real images and fake images from the generator.

These models have a 'long' history in the world of image generation, and their results are considered photo-realistic.

However, it is not all butterflies and rainbows. These models have a hard time in training, sometimes it results in suboptimal results, and a lot of hyperparameter fine-tuning and tricks are necessary to make it work.

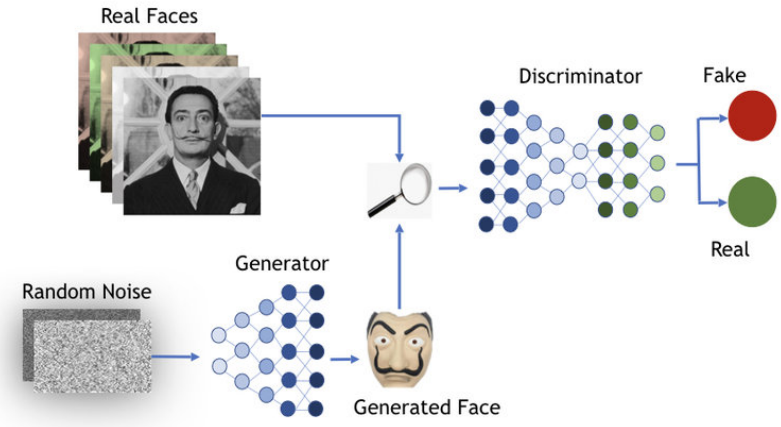

Figure 2.3: architecture of a GAN [5]

GANs are still competing in the state-of-the-art models for image generation, especially because the sampling is way faster than diffusion models.

2.2 Diffusion models

Diffusion models are a class of latent variable models, that are inspired by the diffusion process in thermodynamics [6][7].
In the context of thermodynamics, diffusion refers to the phenomenon where molecules move randomly from regions of high concentration to those of lower concentration until equilibrium is reached.
Denoising diffusion models leverage this same concept to reduce noise in signals or images by modeling noise as the random movement of particles, denoising algorithms aim to restore the original signal by simulating the diffusion process.
Diffusion models achieve state-of-the-art results beating also GANs [8].
The only downside of these models is the time it takes for sampling from them, however, some steps forward have been accomplished and some other approaches have been taken e.g. denoising diffusion implicit models (DDIM).

Many different architectures of diffusion models exist, I would like to make clear the difference between denoising diffusion probabilistic models (DDPMs) and denoising diffusion implicit models (DDIMs).

2.2.1 DDPM

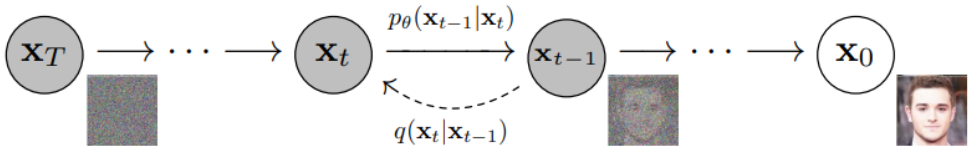

Figure 2.4: Graph model presentented in DDPM paper [6], representation of forward and reverse process

Formalization of DDPMs [6]:
Diffusion models are latent variable models.

In order to understand these models, three fundamentals steps needs to be formalized:

- Forward process;
- Reverse process;
- Training.

The forward process or diffusion process, is fixed to a Markov chain that gradually adds Gaussian noise to the data according to a variance schedule $\beta_1, ..., \beta_T$.
The forward process or diffusion process is formalized as:

$$q(x_{1:T}|x_0) := \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{2.1}$$

The following equation, represents the transition probability that adds Gaussian noise to the previous state x_{t-1} 2.4, with β_t being a schedule of increasing noise levels.
The noise schedule is typically annealed from a high value to a low value over the course of training.

$$q(x_t|x_{t-1}) := \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I}) \tag{2.2}$$

Where β_t indicates at each step the trade-off between information to be kept from the previous step and new noise to be added.
It is also possible to write:

$$x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\varepsilon_t \qquad \varepsilon_t \sim \mathcal{N}(0, \mathbf{I}) \tag{2.3}$$

Now the discretization process should be more clear and it is possible to show that every step of the chain can be generated from x_0:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{2.4}$$

where:

$$\alpha_t = (1-\beta_t) \qquad \text{and} \qquad \bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s = \prod_{s=1}^{t} (1-\beta_s)$$

From the Markov property, the probability of a forward chain is written as (same as the original formalization):

$$q(x_{0:T}) := q(x_0) \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{2.5}$$

The joint distribution $p_\theta(x_{0:T})$ is called the reverse process, and it is defined as a Markov chain with learned Gaussian transitions starting at $p(x_T) = \mathcal{N}(x_T; 0, \mathbf{I})$.
The reverse process is formalized as:

$$p_\theta(x_{0:T}) := p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t) \tag{2.6}$$

2.6 is also the probability of a specific backward trajectory where μ_θ and Σ_θ are two functions parametrised by θ which is want must be learned.
$p(x_T)$ is an isotropic gaussian distribution that does not depend on θ.
more formally:

$$p(x_T) = \mathcal{N}(x_T; 0, \mathbf{I})$$

The following equation represents a single step in the reverse process 2.4:

$$p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1};\ \mu_\theta(x_t,t),\ \sum_\theta(x_t,t)) \tag{2.7}$$

The previous state x_{t-1} is sampled from the learned Gaussian transition probability
($p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1};\ \mu_\theta(x_t,t),\ \Sigma_\theta(x_t,t))$), where $\mu_\theta(x_t,t)$ and $\Sigma_\theta(x_t,t)$
are learned parameters that map the current state x_t and time t to the mean and covariance of the transition probability.
In the original paper the covariance is fixed and only the mean is learned.

Training is performed by optimizing the usual variational bound on negative log likelihood:

$$\mathbb{E}[-\log p_\theta(x_0)] \leq \mathbb{E}_q \left[-\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} \right] = \mathbb{E}_q \left[-\log p(x_T) - \sum_{t \geq 1} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})} \right] =: \mathbb{L} \tag{2.8}$$

During optimization, the forward process is used to sample latent variables from the posterior distribution, and the reverse process is used to simulate trajectories of the Markov chain that start at the endpoint of the forward process and move backwards in time to estimate gradients with respect to the model parameters (θ).
The goal of training is to have reverse and forward process to follow the same distribution.

As formalized before, two different processes in diffusion models are present.
One forward process and one reverse process, both previously formalized.
In the forward process noise is added iteratively to an image sampling from a normal distribution.
This process is simply a preparation of the data for training.
The goal of this model is to try to understand how much noise was added to an image during each step.
If the model can do it, then a strong tool for generation is achieved.
In order to aim for this goal a reverse process was defined.
This last one starts from the last noisy image and tries to remove noise iteratively until the starting image is re-created.
To be trained and learn how to perform a good reverse process a loss function is defined.
This last one, let the model understand if the noise that it is predicting is similar to the true noise generated during forward process or if it is really far away from the truth.
Here the same problem from VAE was encountered, in order to train the model the negative log-likelihood is used, this is previously formalized as:

$$\mathbb{E}[-\log p_\theta(x_0)] \tag{2.9}$$

The probability of x_0 that depends on all other random variables need to be computed, this probability will need to track all other x_{t-1} variables which is not possible in practice.
The variational lower bound is used for this purpose (previously normalized as L).
After a bunch of simplifications the complexity of L can be further reduced and the final objective to minimize while training becomes:

$$L_{simple} = \mathbf{E}_{t,x_0,\varepsilon}\left[||\varepsilon - \varepsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon, t)||^2\right] \tag{2.10}$$

2.10 can also be written as:

$$L_{simple} = \mathbf{E}_{t,x_0,\varepsilon}\left[||\varepsilon - \varepsilon_\theta(x_t,t)||^2\right] \tag{2.11}$$

Where $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$ Which is a random step of a random trajectory in the training algorithm context.

The variance is fixed in the original paper, however some modifications have happened in the literature.
A fixed variance is found to be suboptimal, the variance can be learned from the noise as well.
Nichol and Dhariwal [9] propose a hybrid objective for training both mean and variance using the weighted sum $L_{simple} + \lambda L_{vlb}$ where vlb means variational lower bound.
This new approach permits to have fewer sampling steps without a big loss in quality.

Before formalizing training and sampling algorithms like in the original paper [6] and try to describe them in a less mathematical way, I will point out the parameters that a DDPM needs to learn during training.
During training the parameters to learn are μ_θ and Σ_θ by minimizing the lower bound 2.8

Algorithm 1 Training

1: **repeat**
2: $x_0 \approx q(x_0)$
3: $t \approx Uniform(\{1,...,T\})$
4: $\varepsilon \approx \mathcal{N}(0,\mathbf{I})$
5: Take gradient descent step on:
$\nabla_\theta||\varepsilon - \varepsilon_\theta(x_t,t)||^2$
6: **until** converged

line 1: begin iterative process
line 2: draw batch of samples from training sample distribution, that is, perform forward process.

line 3: perform uniform timesteps from 1 to T.

line 4: Draw some noise with same dimensionality as input data.

line 5: Use parametrization trick to produce samples at timestep t and then give that sample to your model and learn what was the noise used at that timestep t via gradient descent.

line 6: until convergence, repeat iterative process.

In short,a noise prediction network is being trained (predict noise that was used at timestep t).

Algorithm 2 Sampling

1: $x_T \approx \mathcal{N}(0, \mathbf{I})$

2: **for** t = T, ..., 1 **do** :

3: $\mathbf{z} \approx \mathcal{N}(\mathbf{0}, \mathbf{I})$ if t > 1, else $\mathbf{z} = \mathbf{0}$

4: $x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \varepsilon_\theta(x_t, t) \right) + \sigma_t \mathbf{z}$

5: **end for**

6: **return** x_0

line 1: the image x_T is a noisy image drawn from normal distribution

line 2: begin for loop that lasts T steps

line 3: $\mathbf{z}$ represents noise drawn from a normal distribution, in the case of the last step, so if t > 1 no noise is added

line 4: the reverse process is used so, from last step T samples are drawn from normal distribution, then iteratively the sample is refined.

From sample x_t the noise predicted is removed and the normalized noise is added.

The normalization is based of the standard deviation of the sample preceding x_{t-1}.

The process is iterated and when it arrives at timestep 0 the final image is generated, this one has no more noise added to it.

line 5: the loop ends

line 6: return the sample.

2.2.2 Conditional models and guidance

I will now describe the concept of conditional model and diffusion guidance.
A conditional model is a model that takes in input, not just a sample but also a class label, or in the case of diffusion models for text-to-image generation, it will take a text embedding.
The text embedding will be given as input to the model in every step of the training to be conditioned on those embeddings.
The forward process of a conditioned diffusion model stays the same.
The reverse process and the variational bound can be formally written as the following, were **c** is the input on which we want to condition the model:

$$p_\theta(x_{0:T}|\mathbf{c}) := p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t,\mathbf{c}) \tag{2.12}$$

The following equation represents a single step in the reverse process 2.4:

$$p_\theta(x_{t-1}|x_t,\mathbf{c}) := \mathcal{N}(x_{t-1};\, \mu_\theta(x_t,t,\mathbf{c}),\, \sum_\theta(x_t,t,\mathbf{c})) \tag{2.13}$$

The variational bound can be re-written as:

$$\mathbb{E}_q\left[-\log p(x_T|\mathbf{c}) - \sum_{t\geq 1}\log\frac{p_\theta(x_{t-1}|x_t,\mathbf{c})}{q(x_t|x_{t-1},\mathbf{c})}\right] =: \mathbb{L} \tag{2.14}$$

The conditioning applied during training might not be enough for good sampling.
Two ways of guiding the sampling process were introduced in the literature: classifier guidance [8] and classifier-free guidance [11].
In the classifier guidance, an additional model is being used, a classifier.
This classifier will guide the diffusion model simply by taking the gradient on the prediction of the class of an image and using it to perturb the mean of the conditioned model.
In practice, the gradient of the two models are mixed.
By using this perturbation, in the lantent space the information is becoming closer to a situation in which the embedding of the text well represents the image.
In other words, a better spot is found to represent the two components (sample and embedding) in the latent space.

Given a diffusion model $(\mu_\theta(x_t), \sum_\theta(x_t))$, classifier $p_\phi(y|x_t)$ and a gradient scale s, an increasing s improves sample quality but reduces sampling diversity:

Algorithm 1 [8] Sampling with classifier guidance
input: class label y, gradient scale s 1: $x_T \leftarrow \mathcal{N}(0, \mathbf{I})$
2: **for** t = T, ..., 1 **do** :
3: μ, $\sum \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$
4: $x_{t-1} \leftarrow$ sample from $\mathcal{N}(\mu + s\Sigma\nabla_{x_t} \log p_\phi(y|x_t), \Sigma)$
5: **end for**
6: **return** x_0

The sampling (line 4) can be seen as taken from the following modified score:

$$\nabla_{x_t} [\log p(x_t|c) + \omega \log p(c|x_t)] \tag{2.15}$$

A larger ω gives a better overall quality but less diversity (the samples will be more similar between each other).
Given the above defitions, formally writing the classifier guided model should not be a surprise:

$$\tilde{p}(x_t|c) \propto p(x_t|c)p(c|x_t)^\omega \tag{2.16}$$

where the first term $p(x_t|c)$ is the score from the diffusion model and the second term $p(c|x_t)^\omega$ is the score from the classifier guidance.

In classifier-free guidance, the model's knowledge about its class conditioning is used as the classifier knowledge in the classifier guidance.
A conditioned and an unconditioned model are jointly trained.
The classifier-free guidance is sometimes referred to as: "implicit classifier".
The sampling from this model is fomalized as:

$$p(c|x_t) \propto p(x_t|c)/p(x_t) \tag{2.17}$$

The model 2.16 is similar to 2.17, but with the difference of not having a classifier.
In 2.17 the term $p(x_t|c)$ is referred to the conditional diffusion model and the term $p(x_t)$ is referred to the unconditional diffusion model.
The sampling from the classifier guidance model was defined as 2.15, now the modified score of the model is defined as:

$$\nabla_{x_t} [(1+\omega) \log p(x_t|c) - \omega \log p(x_t)] \tag{2.18}$$

Another intuition: imaging having the two points in the space, the one that predicts the noise of an unconditioned image and the one that predicts the noise of a conditioned image, it is possible to sort of find the best direction to push the prediction of noise by using these points.

2.2.3 U-net

Architectures are different in every model, some hyperparameters might change, maybe some layers or some blocks.
However, the usual architecture that is used for the reverse process is typically a U-net.
This architecture takes its name from the shape that it has.
A U-net is very similar to an autoencoder, it was first introduced [12] for image segmentation for biomedical images.
In the current literature, it is seen as a denoising autoencoder with a bottleneck for compressing information analogously to autoencoders.
A typical U-net has ResNet blocks, which try to solve the vanishing/exploding gradient problem by using skip connections.
A skip connection connects two activations of two layers by skipping some layers in between.
The advantage is that any unnecessary layer will be skipped by regularization.
Another important block that is used inside a U-net is usually attention, typically for text-to-image diffusion models.
The cross-attention mechanism helps to concatenate the embedding of the text (conditional model) with the embedding of the image, in order to find a sweet spot for both in

the latent space.

Some architectures also like to work with low resolution images and then upsample them to high quality resolution images via a super resolution procedure applied after or at the end of the U-net.

2.2.4 DDIM

The problem with DDPMs is that sampling is slow; GANs achieve way faster sampling than DDPMs because they do not apply an iterative process.

By making the DDPMs not based on a Markov chain, the number of iterations can be decreased and it is possible to achieve the same or better quality in sampling.

The original paper [7] defines a new family of forward processes indixed by σ:

$$q_\sigma(x_{t-1}|x_t,x_0) = \mathcal{N}\left(\sqrt{\alpha_{t-1}}x_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\frac{x_t-\sqrt{\alpha_t}x_0}{\sqrt{1-\alpha_t}}, \sigma_t^2\mathbf{I}\right) \tag{2.19}$$

in the family of functions the mean is chosen to ensure that the following is true for all t:

$$q_\sigma(x_t|x_0) = \mathcal{N}\left(\sqrt{\alpha_t}x_0, (1-\alpha_t)\mathbf{I}\right)$$

The previous $q_\sigma(x_t|x_0)$ is very important because now the same property of DDPMs is guaranteed.

This property permits to recreate noise in an image at step t with just the initial noise in the image at step 0.

However in 2.19 a property of DDPMs is lost.

The process is no longer Markovian, so an image at the step t cannot be defined just by the image at step t-1.

The single forward process, can be derived from the Bayes' rule:

$$q_\sigma(x_t|x_{t-1};x_0) = \frac{q_\sigma(x_{t-1}|x_t,x_0)q_\sigma(x_t|x_0)}{q_\sigma(x_{t-1}|x_0)} \tag{2.20}$$

σ determines the stochasticity of the process,

if $\sigma_t = \sqrt{(1-\alpha_{t-1})/(1-\alpha_t)}\sqrt{1-\alpha_t/\alpha_{t-1}}$ the definition matches the one of the DDPMs.

if $\sigma_t = 0$ then, the forward process becomes deterministic.
This indicates that the DDIMs are part of the same family of functions as the DDPMs.
Also the reverse process becomes determistic with a sigma of 0. This means that the same original noise leads to the same image.
The training objective is equivalent for any value of σ, this means that a model trained for the original DDPM process can be used for any process of the family.
the two generative formulas are different between DDPMs and DDIMs, but the model trained is the same for both approaches, because the training objective is the same for both.
As it is possible to see in 2.6 since the process is not Markovian, it is possible to skip some connections and sample with fewer steps.

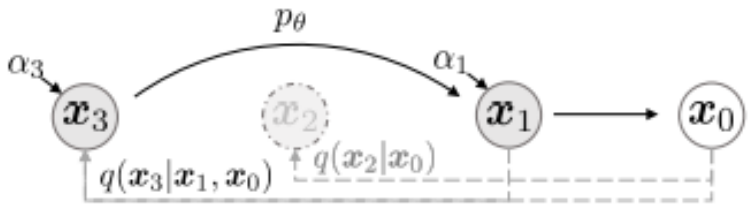

Figure 2.5: Graphical model for accelerated generation, where $\tau = [1,3]$ image from original paper [7]

The generative process defined in the paper is [7] $p_\theta(x_{0:T})$ where each $p_\theta^{(t)}(x_{t-1}|x_t)$ leverages knowledge of $q_\sigma(x_{t-1}|x_t, x_0)$, previously defined.
For some $x_0 \sim q(x_0)$ and $\varepsilon_t \sim \mathcal{N}(0, \mathbf{I})$, x_t can be obtained from the following formula, which is a linear combination of x_0 and noise ε, it can be applied also to DDPMs:

$$x_t = \sqrt{\alpha_t} x_0 + \sqrt{1-\alpha_t}\varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, I) \tag{2.21}$$

Then, the model $\varepsilon_\theta^{(t)}(x_t)$ attempts to predict ε_t from x_t without knowledge of x_0.
By rewriting 2.21 it is possible to predict x_0 given x_t.:

$$f_\theta^{(t)}(x_t) := \frac{(x_t - \sqrt{1-\alpha_t} \cdot \varepsilon_\theta^{(t)}(x_t))}{\sqrt{\alpha_t}} \tag{2.22}$$

The generative process with a fixed prior $p_\theta(x_T) = \mathcal{N}(0, \mathbf{I})$:

$$p_\theta^{(t)}(x_{t-1}|x_t) = \begin{cases} \mathcal{N}(f_\theta^{(1)}(x_1), \sigma_1^2 \mathbf{I}), & \text{if } t = 1 \\ q_\sigma(x_{t-1}|x_t, f_\theta^{(t)}(x_t)), & \text{otherwise} \end{cases} \tag{2.23}$$

where $q_\sigma(x_{t-1}|x_t, f_\theta^{(t)}(x_t))$ is defined from 2.19 with x_0 replaced by $f_\theta^{(t)}(x_t)$.
From $p_\theta(x_{1:T})$ in 2.23, it is possible to generate a sample x_{t-1} from a sample x_t with the following:

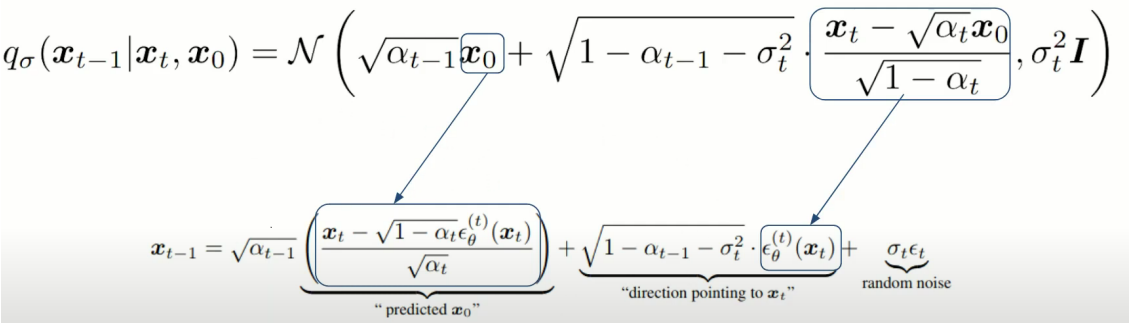

Figure 2.6: The terms from the family 2.19 are substituted in order to get a sampling formula for x_{t-1} given x_t in the general case.

The reverse process can be generalized to any subsample of steps and make it faster:

$$p_\theta(x_{0:T}) := p_\theta(x_T) \prod_{i=1}^{S} p_\theta^{(\tau_i)}(x_{\tau_{i-1}}|x_{\tau_i}) \times \prod_{t \in \bar{\tau}} p_\theta^{(t)}(x_0|x_t) \tag{2.24}$$

Most high-level features are similar, regardless of the generative trajectory.
In many cases, samples generated with only 20 steps are already very similar to others generated with 1000 steps in terms of high-level features, with only minor differences in details.

2.2.5 Latent diffusion models

This section is an introduction to latent diffusion models [16] (LDMs), they are used to make the inference and training steps in text-to-image generation less expansive than before.

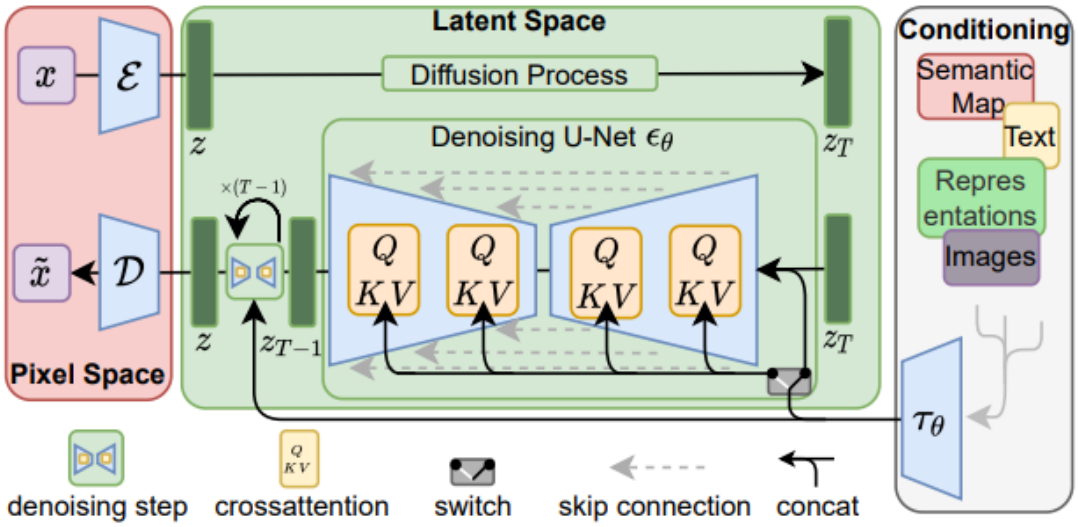

Figure 2.7: Visual explanation of LDMs found in the paper [16]

The following explanation is based on the paper[16]:

Starting from 2.10 which is the usual training objective for text-to-image diffusion models, the paper[16] arrives at a more appealing trainign objective specific for this architecture.

The trained perceptual compression models ε and D play an important role 2.7.

They are able to compress and decompress data so that, the model can focus on only important semantic bit of information and also work in a lower-dimensional space.

The model takes advantage of image-specific inductive biases to build a more specific U-net with 2-D convolutional layers to further focus on important information inside images.

The objective for training is now focused on relevant bits:

$$L_{LDM} := \mathbb{E}_{\varepsilon(x),\varepsilon\sim\mathcal{N}(0,1),t}\left[||\varepsilon - \varepsilon_\theta(z_t,t)||_2^2\right] \tag{2.25}$$

Do not forget conditioning!

It can be achieved with a conditional denosing autoencoder: $\varepsilon_\theta(z_t,t,y)$

A domain-specific encoder τ_θ is introduced, it is able to project a conditioning (like

textual information) to an intermediate representation: $\tau_\theta(y) \in \mathbb{R}^{Mxd_r}$
The representation is mapped to intermediate layers of the U-net using cross-attention layers.
The conditional LDM objective becomes:

$$L_{LDM} := \mathbb{E}_{\varepsilon(x),y,\varepsilon\sim\mathcal{N}(0,1),t}\left[||\varepsilon - \varepsilon_\theta(z_t, t, \tau_\theta(y))||_2^2\right] \tag{2.26}$$

Summarizing: LDMs are powerful architectures biased towards text-to-image generation, they use powerful encoders to encode images and text and use attention layers in the U-net to process them.
The information is compressed and thus faster to compute, with the advantage of focusing only on specific details.
The final output is then decompressed from the latent space to the input space of pixels, this guarantee a fast processing but still an high-resolution image as result.

2.3 Attacks on Diffusion models

This section will be dedicated to the possible attacks that can be achieved on diffusion models.
I will not go into details of the attacks since every attack is different.
An attack depends on the type of architecture that is being attacked and if there is a white box or black box approach.
In a black box approach the settings of the model are not known; on the other hand, in a white box approach, the model hyperparameters are known.
The main types of attack that are known in the literature are [15]:

- Membership inference attacks: these attacks permit to understand if a sample is present or not in the training set;
- Model inversion attacks: these are designed to extract sensitive information about the training data or the model itself. By analyzing the output of a machine learning

model, attackers can infer information about the training data that was used to train the model;

- Attribute inference attacks: these attacks aim at reconstructing attributes of training samples;
- Extraction attacks: these attack aim at completely recover training samples.

Most work in the literature against diffusion models is done on membership inference attacks [13].
Some work has been done also to retrieve training samples [15].
An extraction attack is very dangerous for a text-to-image model.
In many cases, these models are very large and trained on a lot of web images; some of them might have some privacy or copyright issues.
An attacker might be able to steal sensitive information without anyone noticing it.
It is important to not underestimate these attacks.
Some precautions have been proposed; the most important one is the use of differential privacy.
It is a mathematical way to address privacy in ML training, but since there is not a specific threshold for defying what is at risk and what is not at risk, differential privacy is used in many cases in a form that is not optimal and thus is not sufficient to block maleficent attacks [14].
In other cases, differential privacy is not used because it is basically a trade-off between performance and privacy.
Some model creators prefer to achieve state-of-the-art results instead of defending personal information.

2.3.1 Shadow models configuration in membership inference attack

A common pipeline for membership inference attacks is used in the case of a black box setting; this setup requires the use of shadow models.

This approach is one of the only useful ones in a black box setting, but it requires a lot of resources to train and create these shadow models.
In this thesis, the aim is to be able to access a white box model or at least a part of the training set (not the full dataset) in order to compensate for the lack of resources.

The pipeline using shadow models to perform a black box membership inference attack is described in the paper [25].
An attacker that wants to perform a MIA on a specific model M knows the inputs to M and the outputs of M.
The user doesn't have direct access to the model, but through APIs.
Since access to the attacked model is very restricted, information about the training set needs to be learned not from the model itself but from the so-called shadow models.
Some assumptions need to be made anyway.
The type and architecture of the target model need to be known. In the case where the architecture is not known, since the model is accessed through an API, it will provide a very similar or identical model if the user asks to perform the same task. It's then possible to perform training using the same API.
Here are two possibilities:

- There is also a known data distribution that mimics the underling data of the training set;
- There is not a known data distribution that mimics the training set.

In the first case, it is already possible to start training the shadow models to become copies (or very similar) of the attacked model.
This shadow model will mimic the behavior of the attacked model on the shadow model's training set.
If there is no knowledge of the training data distribution, the solution is to use synthetic data.
If the API is very confident in its output on the synthetic data given as input, then, it

probably comes from the same distribution as the training set (**NOTE: in this case the attacked model is a classifier**).

Now there is everything needed to train as many shadow models as are necessary.

The label 'in' will be given to elements from the training sets of the shadow models, and the 'out' label will be given to elements from the test sets.

These data are then used to train an attack model, which is a simple binary classifier, which can then be queried to understand if a specific input was part or not of the original training set of the attacked model.

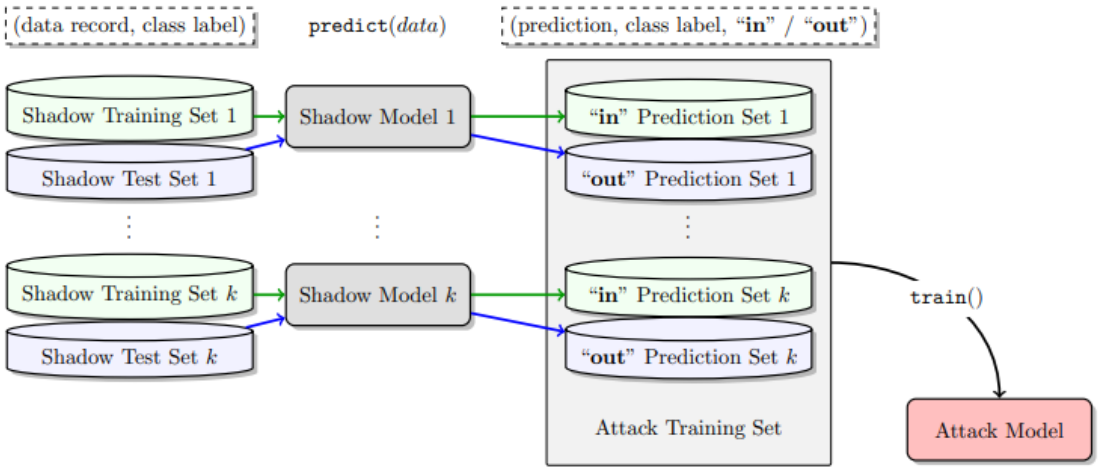

Figure 2.8: Image taken from paper[25]; shadow models setting.

2.3.2 Overfitting, sufficient but not necessary

In the paper from Bogdan Kulynych [26], vulnerability to membership inference attacks is discussed. The concept of vulnerability of an ML model to membership inference attacks is introduced in the same paper.

The vulnerability to MIAs is measured by the normalized advantage [27] of the adversary A over random guessing.

A ML model is said to overfit, or poorly generalize, when its average loss on the training set differs from its loss on new samples from the population.

Previous work [27], shows that while overfitting is sufficient for MIAs, it is not necessary.

The following image 2.9 shows that for the common notion of generalization, that is, average loss (area) on training set and test set, the model does not overfit.

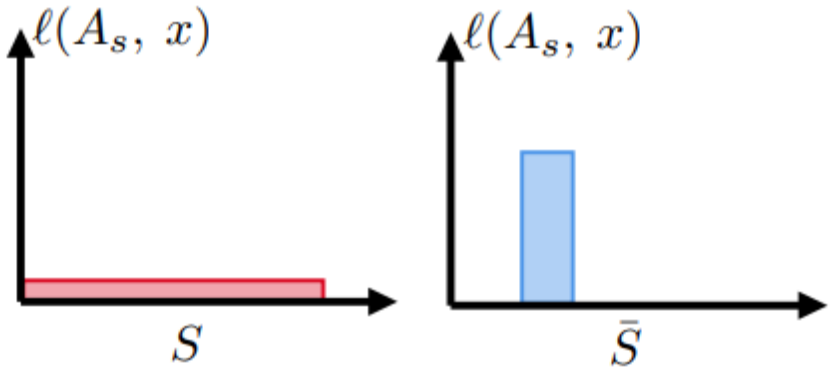

Figure 2.9: Image taken from paper[26]; on the left loss function on training set and on the right on test set.

The authors decided to define a more strict definition of generalization so that it can be a sufficient and **necessary** condition for MIAs.
It covers the difference in the distributions of any given property of a model on the training data and outside.
A property is defined as a function that takes as input a model and a sample and gives as output a numeric vector:

$$\pi(A_S, x) \tag{2.27}$$

Where x is a sample, A_S is a model and π is a property.
A property function can be, a loss function, gradient or prediction from the model.
To make a definition to be necessary for MIAs, the authors decided to look for the distributions of properties on examples x from training set and test set.
Definition of distributional-generalization:
For any property function $\pi(A_S, x)$

$$d(P_{x\sim S}(\pi(A_s, x)), P_{x\sim \bar{S}}(\pi(A_S, x))) \tag{2.28}$$

Where $d(a,b)$ is a measure of dissimilarity between probability distributions.
It looks at the distance between the distribution of data in the training set and in the

test set of any possible property, which implies also the loss function. This means that this definition is a more general definition of generalization where the standard one can be applied by using as property the loss function and the mean discrepancy function as measure of dissimilarity (d).

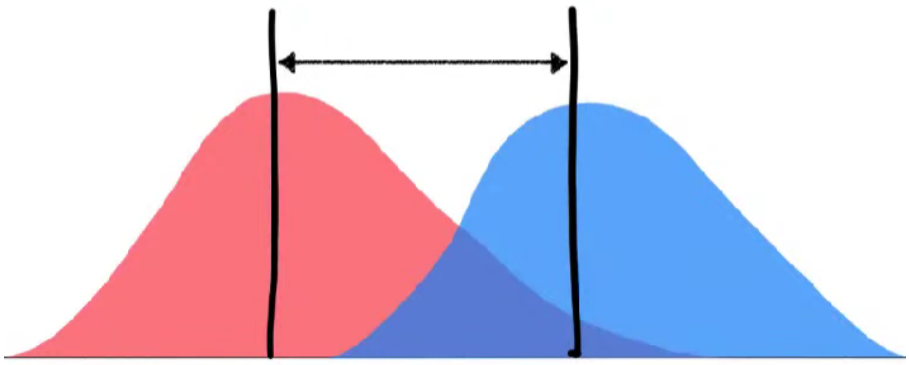

Figure 2.10: Image representing the distribution of any property $\pi(A_s, x)$ (on the x-axis) and how it differs from training and test set

Whereas standard generalization quanties how much the training algorithm tends to memorize the training dataset through the lens of its performance (loss).
Distributional generalization can do so (1) through the lens of other properties beyond losses, and (2) considering distributional information instead of only the difference between the means.
distributional generalization is able to capture any discrepancy in the population, whereas standard generalization does not.
To summirize:

- Poor standard generalization is sufficient for MIAs;
- Poor distributional generalization is sufficient and necessary.

2.4 Attack models

Every attack to machine learning models is built in a different and unique way, however in many cases there is a classification model that is trained and is used to perform the actual attack on a specific targeted model.
Usually these classification tools are binary classifiers and in the following subsections, with analytical references, I will explain the fundamental concepts behind the most used ones.

2.4.1 Support vector machine

Support vector machines (SVMs) in machine learning, are a type of supervised learning used for classification and regression.
In this thesis they have been used as a binary classifier to distinguish between what's **in** the training set of the attacked model and what's **out**.
The goal of SVMs is to create the best line or decision boundary that can separate n-dimensional space into classes, so that when a new data point needs to be classified, it is put in the right space.
The best decision boundary is an hyperplane in n-dimensions where n is > 2, in the case of binary classification, this boundary is simply a line.
The decision boundary needs to be created so that it has the maximum margin. The margin is the distance between the decision boundary and the data points.
By intuition, if the margin is big it means that the data is well separated, in the opposite case, with a small margin it is hard to distinguish between the classes.

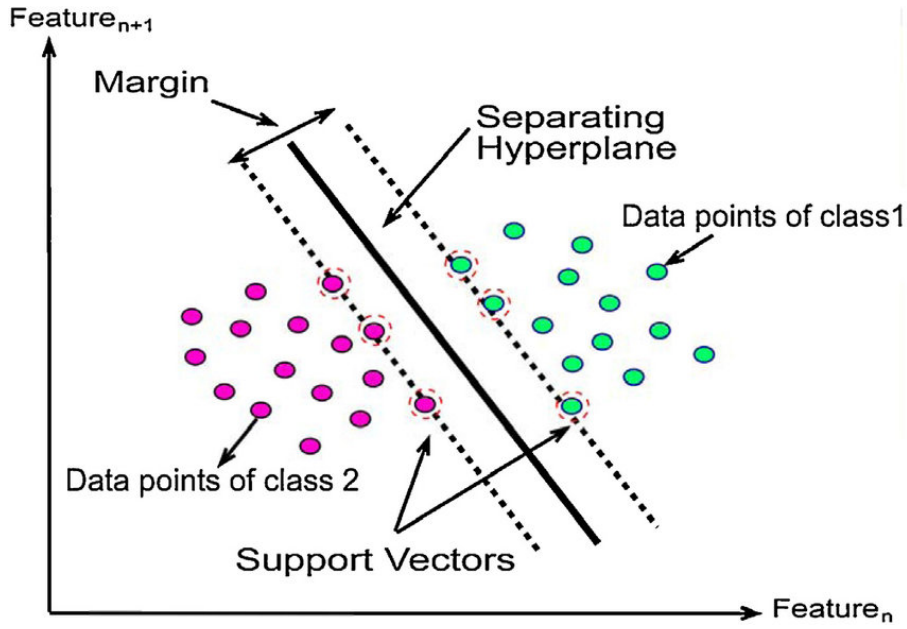

Figure 2.11: Support vector machines

In figure 2.11 it is possible to see the different components in this algorithm.
The support vectors are defined formally as:

$$\mathbf{wx} - b = +-1 \tag{2.29}$$

where the - 1 means that the support vector is on the left of the margin and + 1 means the opposite.
The decision boundary will have equation:

$$\mathbf{wx} - b = 0 \tag{2.30}$$

These equations are nothing but hyperplanes, that in 2 dimensions become lines.
The objective is maximize this margin.
By some simple linear algebra it is possible to take a point on the decision boundary and prove that the margin is equals to:

x on decision boudary;
How many unit vectors are needed to come from **x** and arrive at the support vector (in this case support vector on the right)?

$$\mathbf{w}(\mathbf{x}+k\frac{w}{||\mathbf{w}||})-b=1$$
$$k=\frac{1}{||\mathbf{w}||}$$

The margin is:

$$\frac{2}{||\mathbf{w}||} \tag{2.31}$$

Maximizing the margin means minimizing the denominator so $||\mathbf{w}||$.

This optimization problem can be written as:

$$min_{w,b}||\mathbf{w}||$$
$$\text{s.t. } y_i(\mathbf{w}\mathbf{x}_i-b)\geq 1$$
$$\forall i=1,...,N$$

The problem is solved by finding w,b that solve this minimization problem.

SVM is very helpful method if we don't have much idea about the data. It can be used for the data such as image, text, audio etc.

It can be used for the data that is not regularly distributed and have unknown distribution.

It also doesn't suffer from overfitting and outliers have not much importance.

2.4.2 Logistic regression

Logistic regression is a classifier that estimates the probability of an event to occur.

In regression analysis it's possible to infer the value of the dependent variable through the values of the independent variables.

The dependent variable in logistic regression is a dichotomous variable, it can take only two possible values.

In linear regression the dependent variable is a continuous one.

Since the values of the dependent variable need to be between 0 and 1 (for probability estimate), the logistic regression function will look like this:

$$f(z)=\frac{1}{1+e^{-z}} \tag{2.32}$$

2.32 is called logistic function. The value z is substituted with the linear regression function which is:

$$\hat{y} = b_1 x_1 + b_2 x_2 ... b_n x_n + b_0 \tag{2.33}$$

Where x is the n-dimensional input, **b** is the n+1 parameter vector where b_0 is the intercept of the line and the rest is part of the slope of the line.

The logistic regression function for predicting that the value of dependent value is 1 will look like this:

$$f(\hat{y}|x_1, x_2 ... x_n) = \frac{1}{1 + e^{-(b_1 x_1 + b_2 x_2 ... b_n x_n + b_0)}} \tag{2.34}$$

The parameter vector **b** can be learned through an optimization algorithm by defining a loss function.

A common case is to define an error of 1 when the prediction is wrong and 0 when the prediction is right and then update the parameters through gradient descent algorithm.

2.4.3 Multi layer perceptron with softmax

A multilayer perceptron (MLP) is a fully connected feedforward artificial neural network. This is a very wide topic but I will try to explain the main things without going too much into details.

A MLP has single units called neurons. Each one of them has different inputs, weights and biases.

Usually the weights are the most important thing about a network and it's what need to be tuned for learning.

Inputs, weights and biases are combined in a weighted sum that is then given as input to an activation function. The output of a k_{th} neuron will look like this:

$$y_k = \varphi(\sum_{i=0}^{n} w_{ki} x_i) \tag{2.35}$$

The x_0 input is assigned the value +1, which makes it a bias input with $w_{k0} = b_k$.

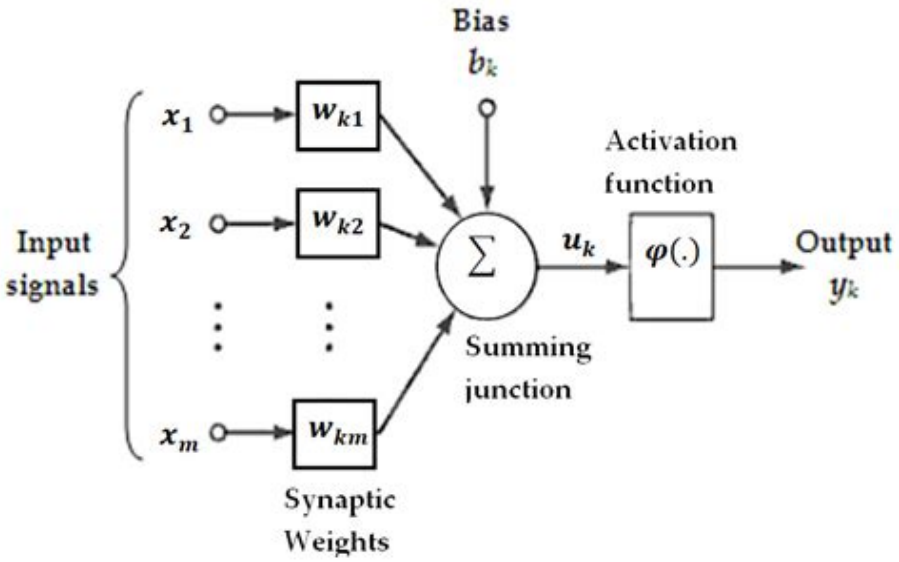

Figure 2.12: Artificial neuron

2.12 is a single neuron, many neurons can be combined to form layers and many layers can be combined to form a neural network.
3-layers fully connected feed forward NN is usually referred to MLP.
The activation functions are non linear and they exist of many types, their use is related to the network specific architecture.
For training these models the usual optimization algorithm that is used is gradient descent, which updates the weights of all neurons in all layers by backpropagating the signal from the output layer to the input layer.

In order to use a MLP as a binary classifier, the output layer needs to have a softmax activation function, it converts a vector of j real numbers into a probability distribution of j possible outcomes, in this case j = 2 (number of output classes).

2.5 Differential privacy

Differential privacy is a privacy framework, it is a mathematical way to define privacy.
Why is differential privacy needed?

It is the only mathematical way to define privacy, and other techniques like data anonymization don't really work.
A common example is the Netflix prize dataset.
In 2007, Netflix announced a prize of one million dollars to anyone who could increase the prediction rate of their recommendation system.
In order to work with real data and real cases, Netflix released a large dataset of users and their movie preferences.
The data were anonymized, and sensitive information like name and address were deleted in the publication of the data.
However, a study [17] revealed that anonymity is not enough to hide someone's identity.
In fact, some of the users' names were reconstructed, and their movies and political preferences were exposed to the world.
The attack consists of a linkage attack, in simple words, by the union of different databases all easy to access, anyone identity can be reconstructed even if their personal information is anonymized.

The key idea behind differential privacy is to learn from the general trend of data without releasing personal information about individuals.
The dataset should not be influenced by the presence of a specific individual, and the general trend of the data should be the same with or without a specific field of data.
After one outcome becomes much more likely after adding one particular individual's data, information is leaked.
On the other hand, if someone's information is added and the outcome remains similar, it is possible to say that the machine learning model is more confident because it has more data, but we are not leaking information about the individuals in the dataset.
It is possible to look at the log of the ratios of the two probabilities, before and after adding a specific field to have an idea of the privacy loss that is going on.
Differential privacy aims at reducing this privacy loss.
As formalized in the paper [18] a privacy loss for a mechanism M, a dataset D where D' is the same dataset with one more example, a set of outcomes S and a privacy budget ε is

defined as:

$$\log \frac{P(M(D) \in S)}{P(M(D') \in S)} \leq \varepsilon \tag{2.36}$$

A smaller privacy budget equals a more private dataset and a smaller learning effect on that same dataset, on the other hand a bigger budget means less privacy and more learning.

A more traditional way to define differential privacy is:

$$P(M(D) \in S) \leq e^{\varepsilon} P(M(D') \in S) + \delta \tag{2.37}$$

where δ is the failure probability.
δ needs to be much less than the probability of a given individual in the dataset.
It is useful to easily prove that the bound works.
The previous defition is also referred to as: (ε-δ)-differential privacy.
This is effective because now, it is proven that the dataset is not leaking no more of what the bound proves, no matter of the post-processing that an attacker can do.
How is this achievable? The answer is with noise.
By adding noise no outcome is more likely than another, for example in images the noise is translated in blurriness.
Also another way to perform this mechanism is by doing random sampling.
There is one another important thing that needs to be addressed, as the fundamental law of information recovery states, it is possible to erode privacy by asking many overly-accurate questions.
Thi can be easily represented with differential privacy by (the more queries you have, the higher the privacy loss): an algorithm (ε_1-δ_1) differentially private followed by an algorithm (ε_2-δ_2) differentially private, can be represented as: (ε_1+ε_2)-(δ_1+δ_2) differentialy private.

Differential privacy has been used in deep learning by modifying the gradient descent algorithm [19]. The cited paper modifies stochastic gradient descent to become differentially private (DP-SGD).

The idea is to limit privacy loss per gradient update by adding noise, so for each gradient update we have a guarantee to have (epsilon + delta) privacy.
This is true for each step of the gradient update, but what about at the end of the algorithm? This is computed by using a technique called the moments accountant.
Sampling with probability q and with T timesteps, the total differential privacy applied during DP-SGD is:

$$(Tq\varepsilon - Tq\delta) \tag{2.38}$$

There are other bounds that tighten this constraint, but the first one given, which is more naive, should give an idea of what is going on.

The moments accountant comes into play as the differential privacy is modeled as a normal random distribution; this gives the possibility to apply concentration inequalities that study the behavior of the distribution when moving away from the mean. It is possible to get a tighter bound by looking at higher moments like the variance (moment for X^2) and so on, up to X^λ.
Through the training, it keeps track of what bound is associated with each of the moments and chooses the tighetest one (from the first moment to the 32^{nd}moment).

This was how differential privacy is formalized and used in deep learning, we will see an application of this during further paragraphs of the thesis, and we will also see why it is not used in the context of text-to-image models.

Chapter 3

Technologies and instruments

In this chapter I will talk about the main technologies and instruments used, I will start by talking about the models that I attacked and why, and then I will go into more details of the instruments used.

3.1 Attacked models

The attacked models are the ones that receive the attack, an outsider will try to gather important information about their training set, this operation is done after the model is trained and is put into production.
I will go through the type of models attacked, training setup and then explain in details the attack in the next chapter.
The following information is related to the architecture of the models and how they were trained.

3.1.1 Stable-diffusion v1-5

Stable-diffusion is open-source latent diffusion model introduced in the paper [16].
It is the main model attacked in the tests of this thesis, the main reason is that the attack

developed needs to have at least a preview of the training set to be open source, more on this in the next chapter.
The background for these type of models is introduced in 2.2.5.
This model is very popular also because it is possible to run on a consumer GPU (easily on 12GB RAM).
Obviously a pre-trained model needs to be used, but the mechanism behind latent diffusion models gives the possibility to fine-tune and make inferences by using the weights of a pre-trained model on a consumer GPU, this is one of the reasons of this model popularity.
This thesis focuses on diffusion models, on low availability of resources and on a dataset which is at least in part open-source, so this model is the best one for this purpose.
The model is used in this paper as text-to-image generation and not for the inpainting capabilities.
A schema of the training steps done on the pretrained model (It can be found on the technical card of the model at
`https://huggingface.co/runwayml/stable-diffusion-v1-5`) is the following:

- stable-diffusion-v1-1: 237,000 steps at resolution 256x256 on laion2B-en.
  194,000 steps at resolution 512x512 on laion-high-resolution (170M examples from LAION-5B with resolution >= 1024x1024).

- stable-diffusion-v1-2: Resumed from stable-diffusion-v1-1. 515,000 steps at resolution 512x512 on "laion-improved-aesthetics" (a subset of laion2B-en, filtered to images with an original size >= 512x512, estimated aesthetics score > 5.0, and an estimated watermark probability < 0.5. The watermark estimate is from the LAION-5B metadata, the aesthetics score is estimated using an improved aesthetics estimator).

- stable-diffusion-v1-3: Resumed from stable-diffusion-v1-2 - 195,000 steps at resolution 512x512 on "laion-improved-aesthetics" and 10% dropping of the text-conditioning to improve classifier-free guidance sampling.

- stable-diffusion-v1-3: Resumed from stable-diffusion-v1-2 - 195,000 steps at res-

olution 512x512 on "laion-improved-aesthetics" and 10% dropping of the text-conditioning to improve classifier-free guidance sampling.

- stable-diffusion-v1-5 Resumed from stable-diffusion-v1-2 - 595,000 steps at resolution 512x512 on "laion-aesthetics v2 5+" and 10% dropping of the text-conditioning to improve classifier-free guidance sampling.

Aesthetic score is simply an indicator of how "beatiful" and aesthetic is an image. It is computed with a neural network that takes as input the CLIP embeddings of the images and gives as output an aesthetic score.

3.1.2 karlo v1 alpha

Karlo is a text-conditional image generation model based on OpenAI's unCLIP architecture.

The model's card can be found at

`https://huggingface.co/kakaobrain/karlo-v1-alpha`.

Karlo is a diffusion model but its architecture is different from stable-diffusion's.

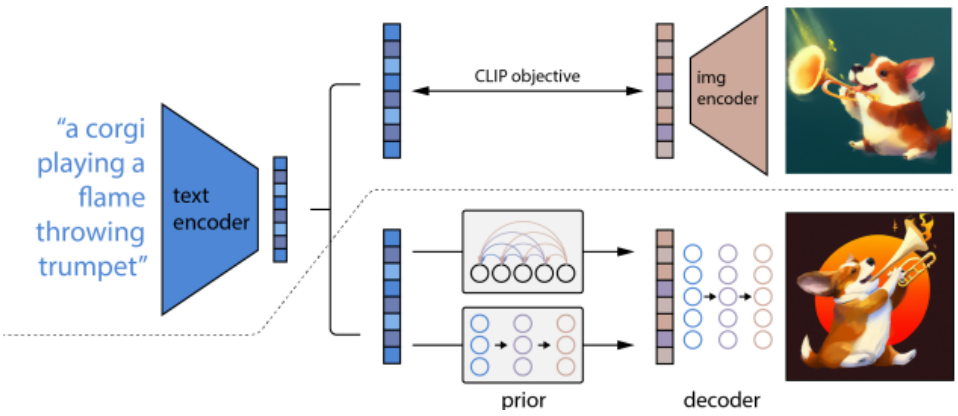

Figure 3.1: unClip architecture

The 3.1 image is taken directly from the paper here cited [20].

It is an overview of the unCLIP architecture.

The CLIP training process gives a joint representation space for images and text.

The CLIP embedding is then given to a diffusion prior in order to create an image embedding that will be used by the decoder for the image generation task.

Karlo does have a prior and a decoder, as stated in 3.1 but it also has super-resolution modules.
In specific, the standard SR (super-resolution) module trained by DDPM objective upscales 64px to 256px in the first 6 denoising steps based on the respacing technique.
Then, the additional fine-tuned SR module trained by VQ-GAN-style loss performs the final reverse step to recover high-frequency details.

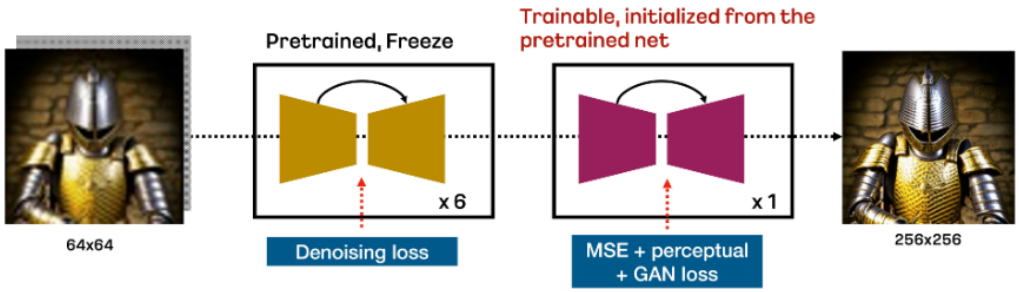

Figure 3.2: The image represents the SR modules of karlo [21]

Here a schema of how the model was trained:

- all components are trained from scratch on 115M image-text pairs including COYO-100M, CC3M, and CC12M;
- In the case of Prior and Decoder, ViT-L/14 provided by OpenAI's CLIP repository is used;
- In the case of the SR module, first it is trained by using the DDPM objective in 1M steps, followed by additional 234K steps to fine-tune the additional component.

3.1.3 Dreamlike photoreal 2.0

Dreamlike Photoreal 2.0 is a photorealistic model based on Stable Diffusion 1.5.
I opted for this model because it is a fine-tuned version of stable-diffusion model and it

works also with higher resolution images.
The idea is to try the same type of attack and see if fine-tuning a model is actually helpful for 'forgetting' the previous training sets on which it was trained.
The model card can be found at
`https://huggingface.co/dreamlike-art/dreamlike-photoreal-2.0`.
The training schema is pretty much the same as stable-diffusion 1.5 except from the fine-tuning, on which authors didn't go into details.

3.1.4 Differentially private diffusion models

These models come from the paper [22].
The authors applied differential privacy and DP-SGD for training these diffusion models, however the attack resulted in a failure, mainly because these models are not text-to-image models and so I couldn't replicate the previously done attack.
I will motivate the failure later in the chapter about Experiments.
These are the only diffusion models that I found trained on differentially private set-ups.
There are reasons why companies do not use differential privacy in text-to-image models and the main one is for quality reasons.
Differential privacy adds noise to the images which result in a lower quality product.
Since the aim of companies that produce text-to-image models is quality, they prefer to have less private models to achieve better performances and resolutions giving the users what they want.
Further considerations about this topic in the conclusions.

3.2 Features extracting model

VGG16 is a convolutional neural network (CNN) it was first proposed by K. Simonyan and A. Zisserman from Oxford University in the paper [23].

This model achieves 92.7% top-5 test accuracy on the ImageNet dataset which contains 14 million images belonging to 1000 classes.
VGG16 or VGGNet was the winner of the ILSVRC-2014 competition.
It improved Alexnet (another CNN) in the use of kernels, Alexnet has the kernel size of 11 for the first convolutional layer and 5 for the second layer, while VGG16 created a longer sequence of 3x3 kernels.
The main reason behind this it to avoid overfitting.
3x3 kernels are the smallest possible, smaller dimensions couldn't capture vertical and horizontal features.
Another benefit of using multiple smaller layers rather than a single large layer is that more non-linear activation layers accompany the convolution layers, allowing the network to converge quickly.

VGG16 is a 16-layers deep CNN, it has 138 million parameters, which is a big number even for today standards.
A lot of parameters don't always mean complexity, in fact, the architecture is pretty simple and has the right number of layers to avoid gradient vanishing.

For better understanding the network, here are some details about the components of it:

- VGGNet receives as **input** a 224×224 image.
  In order to have an input of 224x224 cropping the center of the image or resizing different images sizes is necessary.
- The **convolutional filters** of VGGnet use the smallest possible receptive field of 3×3.
  VGGnet also uses a 1×1 convolution filter as the input's linear transformation.
  VGGnet uses a stride of 1 to preserve spatial information (stride indicates how many pixels the kernel 'moves')

- The **Rectified Linear Unit Activation Function** (ReLU) component, ReLU is a linear function that with a matching output for positive inputs and a zero output for negative inputs.

- The VGG network's **hidden layers** use ReLU instead of Local Response Normalization like AlexNet.
  The latter increases training time and memory consumption with little improvement to overall accuracy.

- **A pooling layer** follows several convolutional layers.
  Pooling is fundamental to decrease the number of features on which the network is working with, it decreases the dimensionality of the outputs of the convolutional layers.

- VGGNet includes three **fully connected layers**.
  The first two layers each have 4096 channels, and the third layer has 1000 channels, one for every class.

In the following 3.3 image, the depth of the configurations increases from the left (A) to the right (E).
The number of parameters also increases with the depth of the network.
The last layer is a soft-max since the network is used for classification purposes, in my case, I used it for feature extraction, so the last layer is not a softmax but it is represented by the FC-1000 layer so that the output is a 1000 vector of features.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 3.3: Image taken from [23], different setups for VGGnet

3.3 Image captioning model

BLIP is a vision-language pre-trained (VLP) model, it was first introduced by Salesforce research[24].

BLIP is a model that is able to perform various multi-modal tasks including:

- Visual Question Answering;
- Image-Text retrieval (Image-text matching);
- Image Captioning.

However I only used it as image captioning tool.

This model is an answer to the previously not optimal configurations of models in the VLP world:

- Previously studied encoder-based models are less straightforward to directly transfer to text generation tasks.
  and encoder-decoder architectures haven't been successfully used in VLP tasks.
- From data perspective, state-of-the-art models are trained on web scraped images with text-images pairs not always of the best qualities.
  In the BLIP paper it is shown that noisy web text is suboptimal for vision-language learning.

BLIP is a multimodal mixture of encoder-decoder, a unified vision-language model which can operate in one of the three functionalities:

- **Unimodal encoder**
- **Image-grounded text encoder**
- **Image-grounded text decoder** The decoder is trained with a language modeling (LM) loss to generate captions given images.

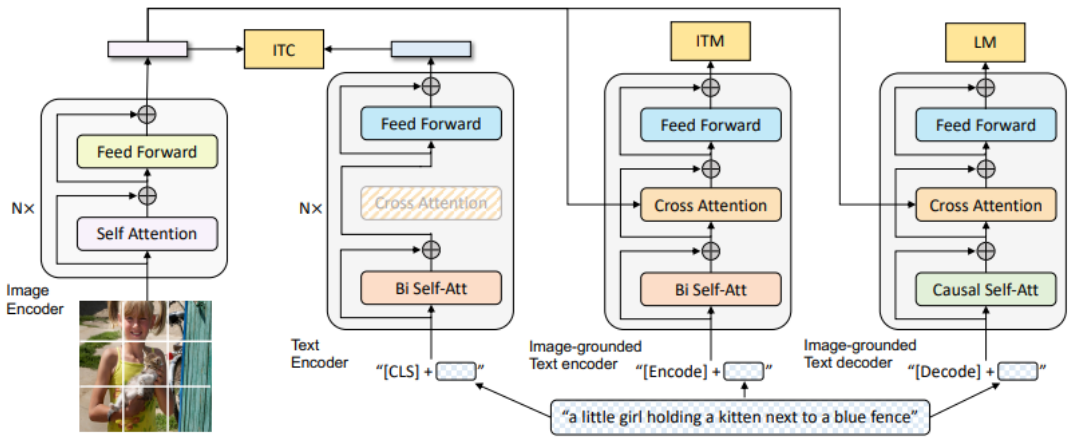

Figure 3.4: Image taken from [24], Blip architecture and its different modules

This model can be seen as a dynamic way to achieve different state-of-the-art results in different VLP tasks.
I will not go in further details for this model, for any further reading, please refer to the paper [24]

3.4 Huggingface

Hugging Face, is an American company that develops tools for building applications using machine learning.
It is well-known for its machine learning models that people freely upload on their website (`https://huggingface.co/models`) so that they can be used by other machine learning enthusiasts.
Hugging Face offers models, datasets, spaces and computational power for machine learning purposes.
It is also well-known for its incredible transformers library for natural language processing.
Hugging face hosts all kind of models that people create, from computer vision to multimodal models.
I personally used this framework in order to import the previously described models in: 3.1, 3.2, 3.3.
I was able to simply import them in my working environment thanks to the easy to use libraries.

Here is an example of how to import stable-diffusion model:

```
from diffusers import DiffusionPipeline,
DPMSolverMultistepScheduler
import torch

repo_id = "runwayml/stable-diffusion-v1-5"
```

```
pipe = DiffusionPipeline.from_pretrained(repo_id,
torch_dtype=torch.float16, revision="fp16")

pipe.scheduler = DPMSolverMultistepScheduler.
.from_config(pipe.scheduler.config)
pipe = pipe.to("cuda")
```

The previous code is pretty simple; it imports the needed libraries and the needed model by taking them from the repo hosted on hugging face, then the pipeline is brought to 'cuda', which means that the model is going to be run on the graphical card, which will make inference way faster than on the CPU.
Models like this are usually run only on the GPU because of the Single Instruction, Multiple Data (SIMD) architecture.
The workload of instructions on a GPU can be divided and run in parallel on multiple cores; this makes GPUs preferable to CPUs.

3.5 Python

Python is a high-level programming language. It emphasizes code readability by the use of indentation.
Thanks to the support to many programming paradigms and its simplicity in instructions, Python is one of the most used programming languages today.
The use of this programming language is very extensive in this thesis due to the many machine learning libraries that are implemented with it.
Some examples are Tensorflow, Pytorch, Sklearn and the more common ones for analytics and mathematics Pandas and Numpy.

3.6 Google colab

Google Colab is an executable webpage that resembles Jupyter notebook.
In Colab anyone can write python code and run it through the cell-system, in fact, there are various cells each one contains a piece of code which is then run to make the results available for the other cells.
Google Colab is extremely useful because it gives GPU, CPU and RAM for free while using it.
In my case I had necessity to use it, since local RAM was not enough to run stable-diffusion model for inference.
Thanks to Colab or tools similar to this, the attack performed is easy to reproduce without hardware limitations.

3.7 Sklearn

scikit-learn (sklearn) is a free software machine learning library for the Python programming language.
It features many ML common tasks like:

- Classification;
- Regression;
- Cleaning and transforming data;
- model selection;
- clustering;
- Dimensionality reduction.

This library was very useful for the thesis because it implements the attack models 2.4.

The possibilities for this library are many, it is very high-level and does not have much room for modifications like Pytorch has, however I did not require any particular modification to the common classification models so it was the best choice.

3.8 Datasets used

In this section I will talk about which datasets I have used to perform attacks, these are not random but are specific for text-to-image models and training sets of attacked models.

3.8.1 LAION

LAION (acronym for Large-scale Artificial Intelligence Open Network) is a German non-profit which makes open-sourced artificial intelligence models and datasets.
It is known for releasing big datasets used in high-profile text-to-image models like stable-diffusion and Imagen.

The data is taken from Common Crwal, a dataset of web scraped pages.
The developers looked for the <img> tag and used the content of the alt attribute as caption.
Not matching captions were identified using CLIP. The datasets do not contain images themselves but URLs that point to them, so researchers have to download them.

LAION-aesthetic v2 5+

LAION-aesthetic v2 5+ is a subset of LAION-5B. The developers decided to use an MLP to evaluate aesthetic scores of the images, initially trained on simulacra-aesthetic-captions dataset.
They found good results and continued with a version 2 of the MLP and that's why this dataset is called version 2.

The 5+ means that the aesthetic score predicted from the MLP is 5 or higher.
These are the subsets of data with 5+:

- 600M image-text pairs with predicted aesthetics scores of 5 or higher;
- 12M image-text pairs with predicted aesthetics scores of 6 or higher;
- 3M image-text pairs with predicted aesthetics scores of 6.25 or higher;
- 625K image-text pairs with predicted aesthetics scores of 6.5 or higher.

Since this thesis focus is having low budget on resources, I personally only used 1k images from the first 600M subset.

LAION-aesthetic v2 5+ faces

LAION-aesthetic v2 5+ faces is a subset of the first 600M aestetic score with 5+, where only faces were extracted by hand by me.
The faces are around 300, and every face detector could extract the same images that I used.
I did it by hand for two reasons:
The first one is that I didn't need many images, and the second one is to have only good quality profile pictures.
I only used 300 images because I wanted to emphasize the privacy issues with these models, so I used a very little dataset and performed an attack with it.

LAION-400M

LAION-400M [29] is an open source dataset. It is a large scale dataset meant for research only and not for any real world application, mainly because not all the data is being checked.

The image are taken from Common Crawl, the images and texts were filtered using OpenAI CLIP by calculating the cosine similarity between the text and image embeddings and dropping those with a similarity below 0.3.
0.3 is an heuristic threshold, the developers came up with it by confronting the similarity of texts and images to real people and register their feedback.
Even if this dataset is not explicitly mentioned in the training set of stable-diffusion, all images from previous datasets come from the same source which is common crawl and some subsets of images might end up in different datasets.

3.8.2 VGG-face

VGG face dataset [30] was created for faces recognition purposes.
It has over 2.6M images to be able to train a model that can recognize faces in images or videos.
I only used around 300 images from this dataset as I needed two different face datasets to perform the membership inference attack on images at risk of privacy.
The other dataset is LAION-aesthetic v2 5+ faces mentioned before.
Faces are perhaps the most common sensitive and easy-to-get information about any human, they assure almost full recognizability and thus they are a privacy concern.

3.8.3 MS-COCO

The MS COCO [31] (Microsoft Common Objects in Context) dataset is a large-scale object detection, segmentation, key-point detection, and captioning dataset. The dataset consists of 328K images.
I used around 1k images from this dataset.
I mainly used it to carry on the membership inference attack, using LAION and COCO as the two main datasets in order to distinguish between 'in' and 'out' samples.
COCO is a top choice for this attack because:

- Very popular dataset with proven scientific reliability;
- It is used for both object-detection and image-captioning, it means that there are captions as in LAION and that there are many different objects and images as in LAION (many objects for detection);
- It was used in Membership Inference Attacks Against Text-to-image Generation Models paper [13] which is the scientific paper that gave me the idea for writing this thesis.
  I can compare the results on MS-COCO directly with that paper.

3.8.4 CC12M

Conceptual 12M [28]is a dataset with around 12 million image-text pairs meant to be used for vision-and-language pre-training.
It is larger and covers a much more diverse set of visual concepts than the Conceptual Captions (CC3M).
It was used in the creation of one of the attacked models of this thesis 3.1.2. I extracted around 1k images from this dataset.
One problem with this dataset is that the images are extracted from the web and filtered in a similar way to LAION; thus, it's hard to tell if there are any overlaps between the two datasets, and in fact, the FID measure on the images of these two datasets confirms this possibility.
Even though VGG-face was created by scraping the web, the main difference is that VGG-face only concerns faces and thus is less likely to have exact copies of faces with respect to LAION-aesthetic v2 5+ faces than two images that represent possibly everything (like in CC12M and LAION).
Even if two images are not exact copies, there is the possibility that they are very similar, whereas if we consider only faces, it is less likely to happen.
In conclusion, it might not make sense to make an attack on images generated on LAION and images generated on CC12M since they might have a very similar distribution and the model might not be able to understand which images are 'in' and 'out' of the training

set.

3.9 Downloading images and file formats

The datasets used can all be categorized as big data datasets. These are typically downloaded as smaller compressed files that contain only a small part of the dataset.
Typically, the format for the compression is the .parquet.
Apache Parquet is an open-source, column-oriented data file format designed for efficient data storage and retrieval. It provides efficient data compression and encoding schemes with enhanced performance to handle complex data in bulk. I used an application called TAD that extracted the parquet file and converted it to CSV so I could import it in Python and look for the URLs of the images in the dataset online and download them automatically.
Another option is to directly import the parquet file in Python if it's not too big by using pandas.read_parquet function.

Chapter 4

Experiments and innovations

In this chapter, I will present the main experiments and innovations.
First, I need to talk more specifically about the papers that I used the most, then I will explain how I continued the work done in these two papers.
I will focus on the attacks that I have done, the datasets used for these attacks, and reproducibility.
At the end of the chapter, I will talk about privacy and how DPDM impacted my research.
The results of the experiments will be shown in the next chapter.

4.1 Main papers presentation

The papers that I am about to present are the ones that I used the most for this thesis; many others are cited and were used, but these two were the ones from which I took the most inspiration

4.1.1 Membership Inference Attacks Against Text-to-image Generation Models

Membership Inference Attacks Against Text-to-image Generation Mode [13] was released on October 3, 2022.
It is the only paper that didn't use shadow models to perform a membership inference attack; thus, it was the only reproducible attack in a scarce resource environment that I could produce, I took inspiration from one of their attacks, and then I tried to make some improvements.
I will describe which parts of the paper were the most useful and how they impacted my work.

The main attack

A text-to-image generation model M can map a text caption t to the corresponding image x.
To construct a text-to-image generation model M, one needs to collect a huge amount of data pairs (t, x) to construct the training set D.
The model is then optimized via minimizing a predefined loss function.
Goal of the adversary understand if a specific image x was used in the training set of the target text-to-image model M_{target}
Adversary's knowledge The adversary only queries a candidate image x without its corresponding text caption to infer the membership.
Even if datasets contain image-text pairs, the previous statement is more realistic.
The adversary also has black box access to the model to attack and a small subset from the member training data of target model $D_{member\,subtarget}$.
The adversary can then obtain another small set of local non-member data
$D_{non-member\,local}$.
The two small datasets are put together to create another dataset
$D_{auxiliary} = \{x_1 \cup x_2 : x_1 \in D_{member\,subtarget}, x_2 \in D_{non-member\,local}\}$ that can be used for

training of a binary classifier that will be the attack model A.

With $D_{auxiliary}$ I can perform an attack in a scarce resource environment without the need of levereging shadow techniques 2.3.1.

The attack used is referred in the paper [13] as II-S.
The intuition behind this attack is that the reconstruction error for an image that belongs to the training set is less than that of an image that doesn't belong to the training set.
To perform the attack pipeline, another tool needs to be used.
The adversary only holds the image x and in order to create another image x', the adversary needs a captioning model to generate a caption t for the image x.
then the generated caption t is fed into the target text-to-image generation model M_{target} to obtain a generated image x'. In this way, the query image is connected to the generated image explicitly.
Attack II-S, also applies a pre-trained visual language model to extract the embeddings of the generated image and its corresponding query image, respectively, and then measures the distance between them.
Finally, the adversary feeds the distances into the attack model to distinguish between members and non-members 4.1.

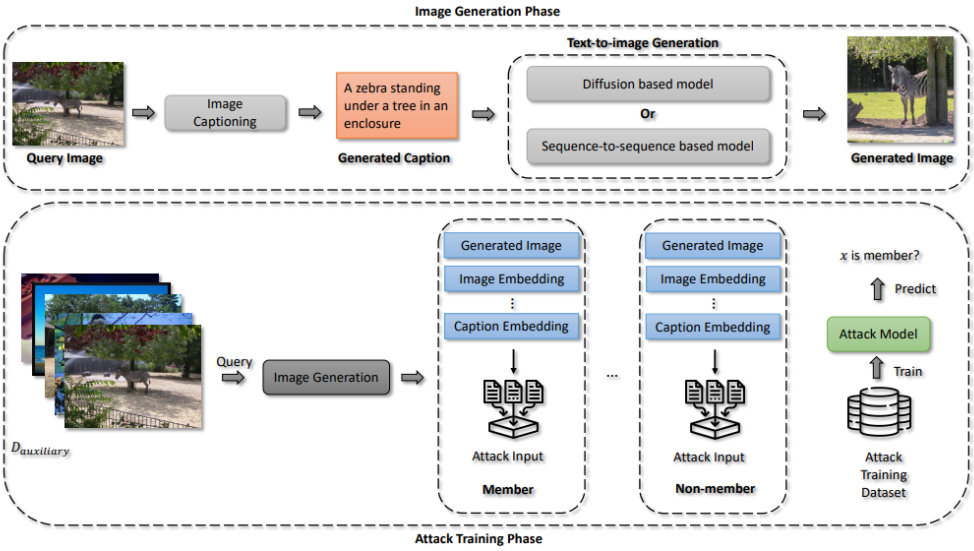

Figure 4.1: Image taken from paper [13], overview of the pipeline attack, generation of image and attack phase

In the paper, many attacks are presented, however II-S seems to be the most promising one.
The intuition for this model is based on semantic, since the embeddings from the visual language model capture the elements of the image.

Captioning and features extraction

The paper [13] shows that denoising steps, captioning tools and feature extraction tools have negligible effects on the final results.
Many different models for these purposes have been tried out: CLIP, ClipClap and BLIP. The results are consistent with all the different kind of attacks and with the different tools used. Attack II-S still have the advantage over the others, thus, remains the one on which is worth investigating more.
The model attacked in the paper are LDM and DALL-E mini, however I expect to have similar behaviour with the models that I have tested.
Since these variables have already been tested I decided to not re-do the same experiments on the chosen attacked models of this thesis.

4.1.2 Dataset disclosure

The dataset disclosure is referred to $D_{member\,subtarget}$. I do not fully agree with what is said in the paper [13] that the attack uses a black box access to the model; the reason is simple: there is access to the training dataset of the model.
It is true that the target model is just used as it was in a black box setting; however, black box also refers to the information that we know about the model, in this case there is knowledge about private information from the training process. Thus, I find the setting not to be a black box but more of a white box or a sort of gray box setting, in between the full black box setting and the fully disclosed white box setting.
A small part of the training set is not something guaranteed; fully black box settings without any training set preview must be ruled out from II-S attack.

The reason is that without some reference to members, it is impossible to distinguish between what's in the training set and what's not.
This is very important to specify since it is not written in the paper and might be confused with the previous black box setting defition.
The attack is still useful; it can be used with models on which there is an open-source dataset, like stable-diffusion, or on models where only a tiny part of the dataset is disclosed or previewed.

Some may argue that with an open-source training set, we can perform a MIA without the use of a binary classifier. It is, in fact, possible to know if an image is part or not of the training set just by searching for it in the fully revealed training set.
It is possible, but remember that we are in an environment with scarce resources.
A search through some billions of data is not an easy task; it requires a lot of memory capability and a lot of computational power for the search per sè.
Thus another important factor must be considered, if an attacker wants to perform a MIA, the training set must be downloaded and in order to download billions of data in a reasonable time, massive clusters of hardware are required, making the attack not so easy to produce.
Since the search is impossible in a scarce resource environment, I decided that the attack pipeline from the paper [13] is still useful for this setup, however, with the clarifications stated before in this section.

4.1.3 Membership Inference Attacks From First Principles

Membership Inference Attacks From First Principles [33] explains how accuracy should not be used a metric of assessment for MIAs; it also introduces a new MIA called LIRA, which is not important for this thesis.
MIAs are very often evaluated using average-case "accuracy" metrics that fail to characterize whether the attack can confidently identify any members of the training set.
In the paper a new type of assessment is advised: measure the true positive rate at a low false positive rate (e.g. $< 0.1\%$); with this metric it was found that most prior attacks

perform poorly when evaluated in this way.

MIAs have become the standard to test if a model is private due to their simplicity. However in many papers they are tested using average-case success metrics like accuracy or ROC-AUC.
However, privacy is not an average case metric, and should not be evaluated as such.

This paper re-examine the MIAs effectiveness and their evaluation of TP at low FP rate.
The most important thing when performing attacks on privacy is that If a membership inference attack can reliably violate the privacy of even just a few users in a sensitive dataset, it has succeeded. And conversely, an attack that only unreliably achieves high aggregate attack success rate should not be considered successful.
Metrics like AUC are not correlated with low-FP rate, thus, they are not a real indicator of the success of the attack.
Main reasons why accuracy does not work:

1. **Balanced accuracy is symmetric**, That is, the metric assigns equal cost to false-positives and to false-negatives, false positives reduce the attack efficacy while false negatives are just an indication of an unsuccessful extract;

2. **Balanced accuracy is an average-case metric**, but this is not what matters in security.

   Two attacks with same accuracy might be very different. One attack might perfectly recognize a subset of 0.1% and recognize 0.5% the other cases and the other attack might recognize 0.5005 all the given cases. While the first attack is very potent, the second one is useless and nonetheless they have the same accuracy.

A natural choice for attacks measure is to consider the tradeoff between the true-positive rate (TPR) and false-positive rate (FPR).
Intuitively, an attack should maximize the true-positive rate (many members are identified), while incurring few false positives (incorrect membership guesses).

For the same reason accuracy does not work, also AUC does not work, since the AUC averages over all false-positive rates, including high error rates that are irrelevant for a practical use.
The final takeaway is to test an attack with TP rates at low FP rates; This is emphasized by the log-log ROC representation.
This way there is more emphasis on the low FP part of the attack which is the most important thing to consider.

4.2 Main flaw of MIA Against Text-to-image Generation Models paper

In this paper some good intuitions were made, however the first thing that catches the eye when looking at results is accuracy.
Accuracy have been used in all results of the attacks of the paper, it is said to be the most used metric and so to compare with other prior attack, the creators also used it in their experiments. Accuracy does not tell if an attack is good or bad, so it's not a good indicator of the effectiveness of the attacks.
The attacks have to be tested again in order to understand their true effectiveness.
Further in results I will show what I obtained through the attack II-S of this paper.

4.3 Dataset search

It is possible to search through an enormous dataset through CLIP [32] search.
CLIP (Contrastive Language-Image Pre-Training) is a neural network trained on a variety of (image, text).
CLIP matches the performance of the original ResNet50 on ImageNet "zero-shot" without using any of the original 1.28M labeled examples, overcoming several major challenges in computer vision.

Clip search works in this way:

1. Create embeddings of the dataset you want to search in;
2. Embed the query image/text through CLIP;
3. Retrieve the similar images to the query image through KNN algorithm.

An example of such search con be performed on the LAION dataset through the following link: `https://rom1504.github.io/clip-retrieval/`.
The main problem is that to run a similar pipeline there needs to be a backend that contains all the data from the dataset.
All the images need to be downloaded, stored and the embeddings of each one computed, then the search algorithm can be performed.
These operations are impossible in a scarce resource environment, thus, this is not a viable option.

4.3.1 Why the dataset can't be downloaded in its completeness

Considering a consumer setup with an optic fiber internet connection, it is realistic to have around 65 Mpbs for download.
Images in large datasets are not high-quality ones, usually 200x200 to 800-1400, they vary from 10 Kb to 100 Kb.
Consider that for downloading a dataset we are actually downloading URLs and text pairs not images.
LAION 5B has around 40 TB of data with just URLs and text data.
Considering a 65 Mbps downloading speed it will take:

- 5 412 981 seconds
- 90 216,35 minutes
- 1 503,61 hours

- 62,65 days

Now consider a very optimistic time of 0.3 second for searching an image via the URLs through the internet and download that image.
Considering 5 billion of images we get 1.500.000.000 billions of seconds which is:

- 25 000 000 minutes
- 416 666 hours
- 17 361 days

If we do not consider the searching time but 350 TB of images, gained by considering the average image of 70 kb.
The result is still around 548 days.
Which is of course, a lot less, but is also unrealistic.

Considering 350 TB of data and 548 days to download all the images, the processing time for getting results is another obstacle, this is the proof of why it is not possible to perform a search via CLIP on a big dataset on a local machine.
The attack through searching is thus ruled out.

4.4 Innovations

In this chapter I will discuss the main innovations that I bring to MIAs with this thesis.
A lot of work has been done also in re-trying the prior literature attacks but with proper evaluation metric, however in this chapter I will focus on completely new approaches: I have created two new ways to intend the attack II-S of the paper [13].

4.4.1 No background faces MIA

Why faces?

Faces can be considered a private asset that people are not willing to give for sale.
Many people would not agree to sell their profile images for research or commercial purposes, even though this is what happens today with images released on social media and websites.
Images of faces are considered private because it is possible to recognize a human being among billions of others with that given image/images.
Face recognition is one of the most concerning things in today's era. The main reason is that governments can use this information for surveillance of their populations.
People are becoming more aware of the importance of their privacy, however billions of face images are floating around the internet every day, this is why it might be interesting being able to recognize if an image was used or not in the training process of a text-to-image generation model.

The attack

The attack follows the same principle of the main attack that can be summarized in figure 4.1.
In the main paper [13] the researchers also tried to recognize faces with the MIA described in chapter 4.1.1.
Profile images usually have a background, this one can represent different environments, when the difference between embeddings of the original image and generated image is performed, the information regarding backgrounds is redundant and is preferred to only have features of the faces themselves.
The background is considered redundant because the caption generated via BLIP usually captures the main features of the subject but doesn't really capture the background information; the result is that after the generation, the original and generated images will have different backgrounds with different features.

By removing backgrounds of the original profile image and the generated profile image (after the generation) the redundant information is eliminated and the feature extracting model can retrieve more representative information by using the same images.
Results 5 confirm this intuition, with an improved performance on overall ROC and TPR with low FPR.

Why apply only this process to faces?

The reason is simple, usually profile pictures have a distinct foreground and background, this is not guaranteed for other type of images that can be found in any text-to-image dataset.
In the following images it is clear why in non-profile pictures the background removal is not an easy task and also it might be an important part of the picture and its removal will cost important information.

Figure 4.2: LAION-5v+, no profile picture

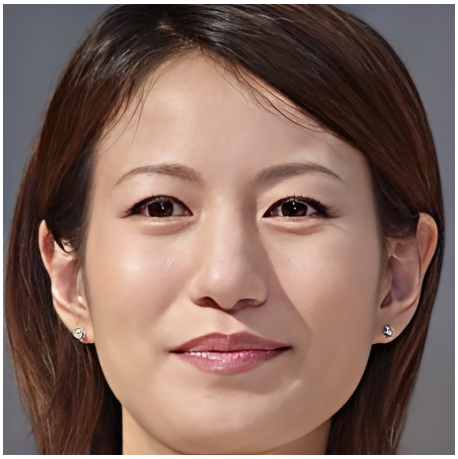

Figure 4.3: VGG faces, profile picture

Removing backgrounds from images different than profile pictures can be challenging and not easy to automate, while with faces is pretty easy to achieve and many python libraries can already do it with no effort.

4.4.2 Prompt attack

The idea behind this attack is similar to II-S [13], but instead of considering differences between the original image and the generated image to understand if the original image is part of the training set, This attack focuses just on the generated images.
The pipeline 4.1 stays pretty much the same; for creating the classifier, I will still need to have a part of the training set disclosed.
The idea is to get captions from the images through BLIP or by using the original captions and then generate new images with those captions.
After the images have been generated, I will get the embeddings of the images and use them to train the classifier.
This time the embeddings are not the difference between the original and generated images but are just the ones referring to the generated images of the 'in' training set and 'out' training set images.
After the classifier is trained, the attack will consist of giving the model an image, and it will say if the image was part or not of the training set. In this case, it's possible that the image is not a perfect copy of an image in the training set since we are speculating on just the generated images.
The attack is helpful, because instead of focusing on the image, it is possible to get the caption from the image using the same captioning model used for training and get an idea of a prompt that is in the training set.
Of course, it is very unlikely that the caption will be a perfect copy of one of the captions in the training set; however, it is possible that the caption will be similar or will generate similar content to one in the training set. In this way, we can disclose not the images in the training set, but the captions that generate similar content to the ones in the training set. In other words, we discover a caption that has same value of a caption in the training set.

In 4.4 and 4.5 is possible to see two images that have a pretty similar representation given the caption from the original dataset and the one generated by BLIP.
This should give an idea of how similar image generation can be with similar prompt,

and why the attack on prompts is helpful.
The images were generated from stable-diffusion-v1-5.
Having two prompts that generate very similar images is equivalent of having discovered the real prompt from the dataset, giving same privacy concerns as in the case of discovered images in the training set.
The cosine similarity on the two prompts 4.4 and 4.5 is : 0.0 and thus, should not be a good indicator on how similar content the prompts generate.

To better understand if two prompts generate the same content, the best idea is to compare the two images that the two different prompts generate using some measures like FID, this way it is possible to directly compare two generated images and link them to their prompts.

Figure 4.4: Originial prompt:
Rusty Bell on Old Lodge Building,
Searcy County, Arkansas

Figure 4.5: BLIP prompt:
rusty bell hanging from a brick wall
with a brick wall behind it

4.5 Implementation

link to my GitHub `https://github.com/fedeflowers/Thesis`.
The code is a big notebook file with some descriptions of the main parts of the code.
It is not intended to run sequentially but to run only the parts that are needed for the case.

I think it's a good way to for new people to learn more about the subject and test themselves what I have achieved.

4.6 FID

The Fréchet inception distance (FID) is a metric used to assess the quality of images created by a generative model.
Unlike the earlier inception score (IS), which evaluates only the distribution of generated images, the FID compares the distribution of generated images with the distribution of a set of real images ("ground truth").

Rather than directly comparing images pixel by pixel the FID compares the mean and standard deviation of the deepest layer in Inception v3.
These layers are closer to output nodes that correspond to real-world objects such as a specific breed of dog or an airplane and further from the shallow layers near the input image.
The embeddings used for computing FID in this paperwork are the ones coming from VGGnet.

Some FIDs between datasets have concerning values; for example, I obtained a lower FID for the CC dataset, which was not part of the training set in the case of stable-diffusion. With a deeper analysis, it is possible to note that some images that are part of LAION dataset are also part of the CC dataset, mainly because the way the data was gathered is pretty similar in the two datasets, making it very hard for the model to distinguish between images from the dataset and images outside the datasets.
There could be similar cases with other datasets, so before trying an attack, always verify with FID that the two sources of data have distinguishable distributions; otherwise, the attack will not be as powerful.
FID is a good indicator of how different two datasets for image generations are; the greater the difference in FID, the better the results of the attack.
The most important thing is to not have duplicate images in the 'in and 'out' datasets.

4.7 Privacy and DPDM

Differential private diffusion models [22] are a good way to test the use of differential privacy in diffusion models.
There are, however, some important things to note:

1. The sampling from DPDM was obtained through the published github: `https://nv-tlabs.github.io/DPDM/` which refers to the work done in the corresponding paper;

2. The models are diffusion models, but they are not text-to-image ones, so I could not recreate the attack II-S of the paper [13];

Since the models are trained with differential privacy, the final outputs of the models are blurred images, which make comparison with the training set data almost impossible. This is for sure a good trait of differential privacy for diffusion models; however, the blurriness, which is a good trait for privacy, is also a big downside for the usage of the model.
Imagine text-to-image models with outputted blurry images; the users will be very unsatisfied, mainly because the images are unusable.

The blurriness comes from the noise that is added through differential privacy, making it impossible to recognize a specific instance inside the dataset.
This follows the main principle of differential privacy, which is that the dataset should maintain the same effectiveness with or without a specific instance.
While differential privacy is a good solution for privacy issues, I would say that it is not applicable in the case of text-to-image models. The main reason is that the outputs come with too much reduced utility.
Researchers should come up with more effective ways to preserve privacy in these models, which should not impact the final outputs; in the current state, the trade-off between privacy and performance is too heavy, and companies will always choose performance.

4.7.1 Bad setup for attack

The attacks explained in this thesis could not be made on DPDM.
The models are not text-to-image models, and thus they do not receive a prompt as input. The sampling is random given the training images; it is possible to download the training datasets since they are open source; however, it's impossible to understand given an original image how the model will generate it as output through the diffusion process; so it's impossible to apply a difference in embeddings between the original and generated image.
All these reasons made the MIA setup for this thesis impossible.
This does not mean that the models are unattackable, but just that the setup is bad for the purposes of this thesis.

4.8 Prevent attacks and further work

In this chapter I will try to explain what was published in the paper Privacy preserving generative framework for images against membership inference attacks [34], which might be a good alternative to the simple differential privacy; The paper in fact utilizes synthetic data to train a model which should be able to prevent MIAs or at least be more robust against them.
The simple intuition is that if the model is trained on synthetic data which resemble the original distribution, but they do not contain strictly private information, since the data is completely synthetic.

In 4.6, it is possible to observe the proposed system model, which aims to safeguard privacy while training a machine learning model to defend against membership inference attacks. Let's go through the process in two phases.

Phase I: To start, the data owner builds an extractor and a reconstructor using source data.
These components help create two sets of data: "recovered data" ($\tilde{D}$) and the original

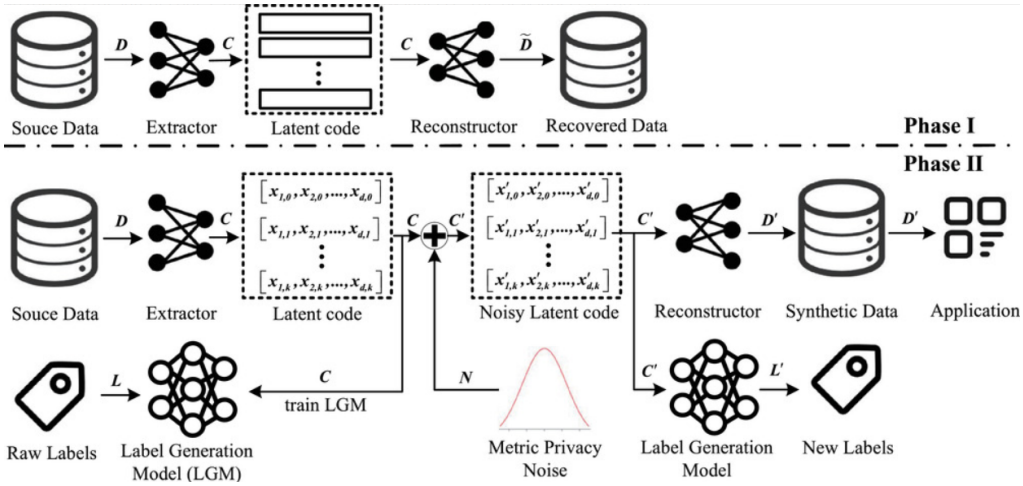

Figure 4.6: Image taken from paper [34], overview of the privacy preserving methods for creating a robust model against MIAs

data (D). The objective is to make these sets as similar as possible.

Phase II: Moving forward, the following steps are taken.
First,the latent code (C) is obtained from the extractor using the original data (D). Then, some noise (N) is added to this latent code (C) while ensuring privacy, resulting in a new "noisy" latent code (C').

Next, the reconstructor is used to generate synthetic data (D') based on the noisy latent code (C'). Additionally, if the original data (D) had labels (L), it is possible to train a label generation model (LGM) using the latent code (C) and the corresponding raw labels (L).
This trained LGM can predict new labels (L') from the noisy latent code (C').

Finally, it is possible to use the synthetic data (D') to train new models while preserving privacy and protecting against membership inference attacks.

More specifically, the data owner first trains a fundamental VAE model M_G locally using the data source D.
The VAE model maps the original data to the latent space vector through the encoder and then maps the latent space vector back to the original data through the decoder.
Because the decoder can rebuild the data in original data through the latent space vector,

it shows that the latent space vector contains enough original data information.
N is added to the latent vector generated by model M_G and the data owner leverages the decoder of M_G to reconstruct data D'.

The noise (N) added to the latent code (C) is used to re-create private data in the latent space, so differential privacy is also used in this pipeline.
In the case of DPDMs, the noise added was so heavy that the outputs were unusable, since in this new case the noise is added to the process of reconstruction, it should make sense that at the end of the reconstruction process the images will not be blurred but the added noise will be enough to not reconstruct the same original images.

Even if the results presented in the paper [34] are encouraging, the research only used accuracy to test their models and attacks, which as I pointed out before, is not a good enough metric. Also, the models attacked are not text-to-image models but classic CNNs, so some further research can be done having privacy against MIAs as main focus. It might also be possible to make the pipeline work without adding noise but just using synthetic data.

Chapter 5

Results

The following are the results of the attacks explained in the previous chapters, there is also a comparison of FIDs in order to explain the success of the prompt attack and the II-S attack.

The subsets of images ’in’ and ’out’ as sample to the attack are exactly half, so in the case of 200 images, 100 belongs to ’in’ training set images and 100 belongs to ’out’ training set images.
the test size is always 40% and training size is 60%.

5.1 Stable-diffusion, LAION5v+ vs MS COCO, 200, 1000, 2000 images, attack II-S

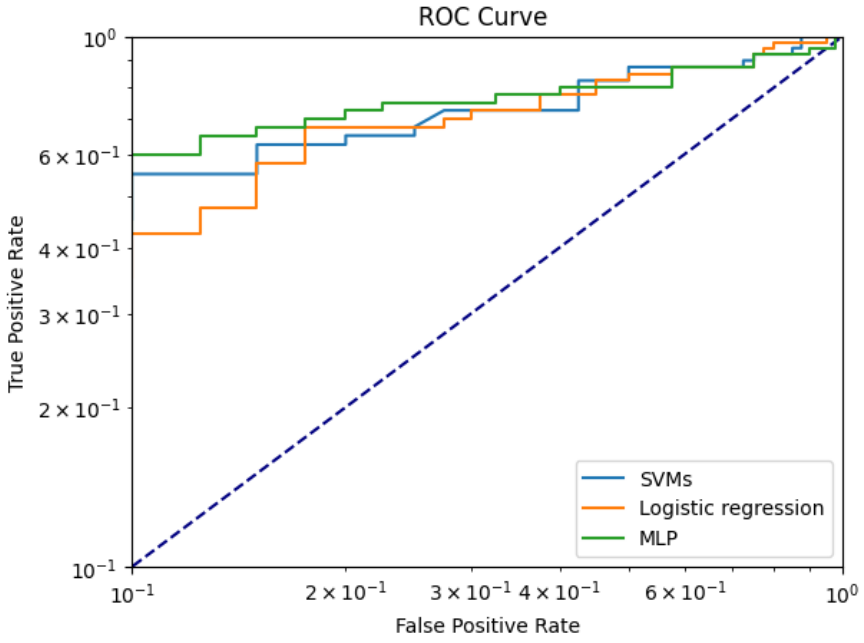

Figure 5.1: Log ROC curve of LAION vs COCO 200 images, stable-diffusion

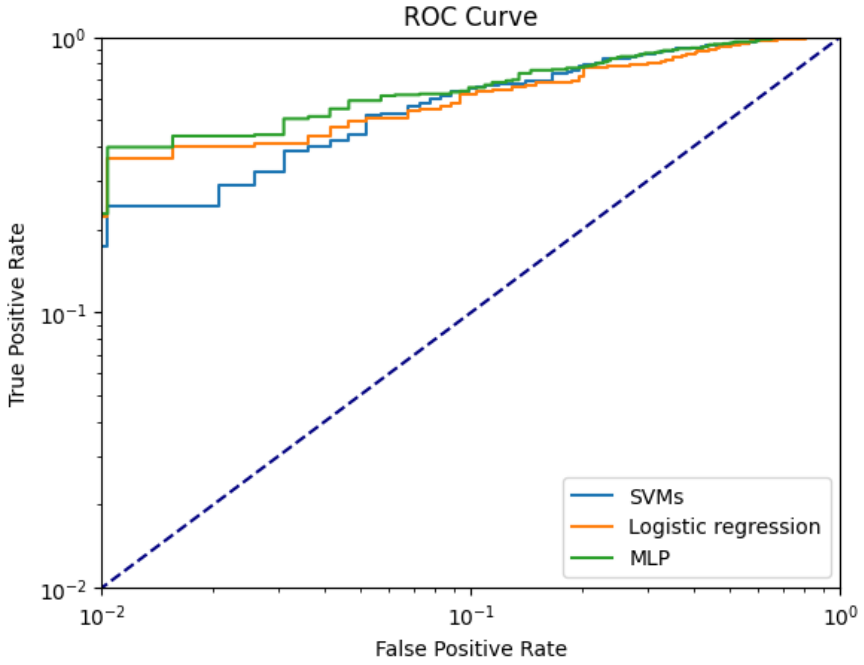

Figure 5.2: Log ROC curve of LAION vs COCO 1000 images, stable-diffusion

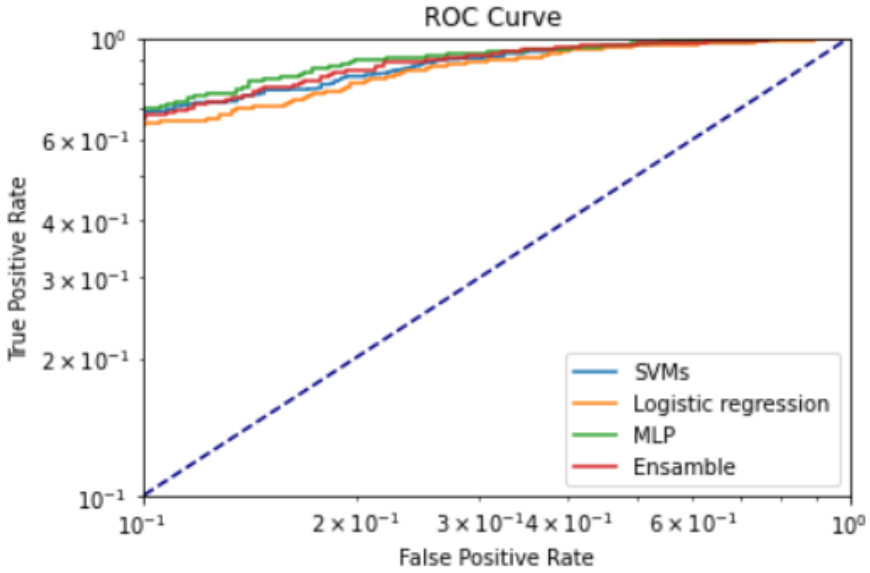

Figure 5.3: Log ROC curve of LAION vs COCO 2000 images, stable-diffusion

TPR with low FPR in 5.1 is:

- Tpr for fpr <= 0.1 in svm is: 0.55
- Tpr for fpr <= 0.1 in logistic regression is: 0.425
- Tpr for fpr <= 0.1 in MLP: 0.6

TPR with low FPR in 5.2 is:

- Tpr for fpr <= 0.1 in svm is: 0.6504
- Tpr for fpr <= 0.1 in logistic regression is: 0.6165
- Tpr for fpr <= 0.1 in MLP: 0.6553

TPR with low FPR in 5.3 is:

- Tpr for fpr <= 0.1 in svm is: 0.6898
- Tpr for fpr <= 0.1 in logistic regression is: 0.6451

- Tpr for fpr <= 0.1 in MLP: 0.6997
- Tpr for fpr <= 0.1 in Ensemble (hard voting): 0.6650

The information is summarized in the following table:

TABLE of TPR of different classifiers with FPR <= 0.1

	200 images	1000 images	2000 images
SVM	0.55	0.6504	0.6898
Log. Regr.	0.425	0.6165	0.6451
MLP	0.6	0.6553	0.6997

5.2 Karlo v1 alpha, II-S attack, 1000 images used, CC3M vs MS COCO

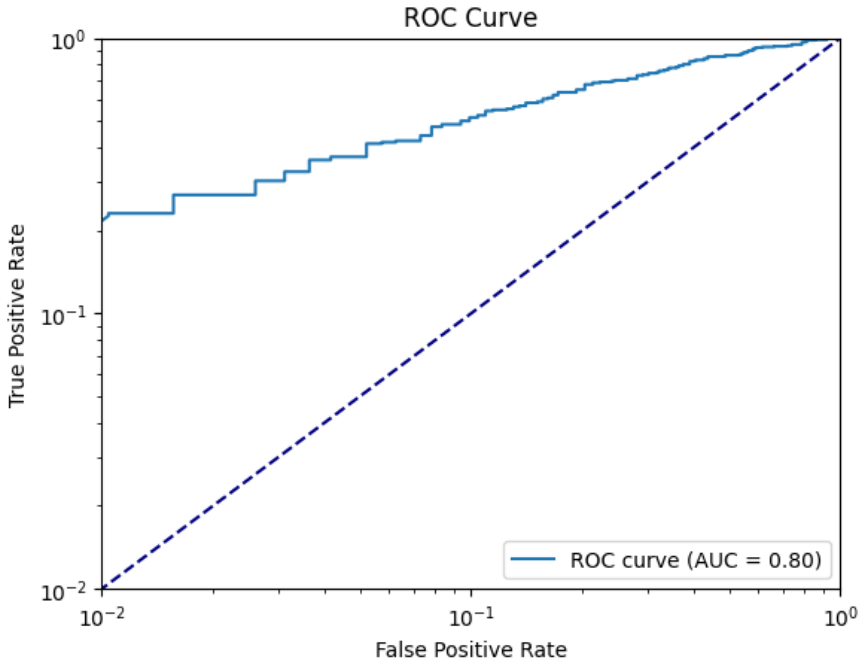

Figure 5.4: Log ROC curve of CC3M vs MS COCO, 1000 images, karlo

TPR with low FPR in 5.6 is:

- Tpr for fpr <= 0.1 using MLP is: 0.5144

5.3 Dreamlike photoreal 2.0, II-S attack, 1000 images used, LAION5v+ vs MS COCO

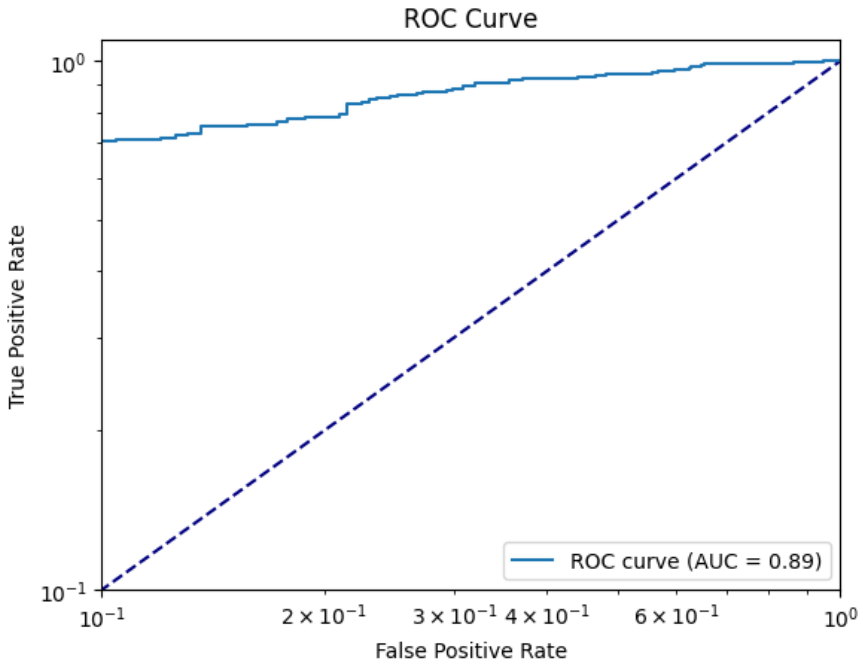

Figure 5.5: Log ROC curve of LAION5v+ vs MS COCO, 1000 images, dreamlike photoreal 2.0

TPR with low FPR in 5.6 is:

- Tpr for fpr <= 0.1 using MLP is: 0.7033

5.4 Stable-diffusion privacy attack (profile photos) with BG, 1200 images, LAION5v+ face vs VGG face

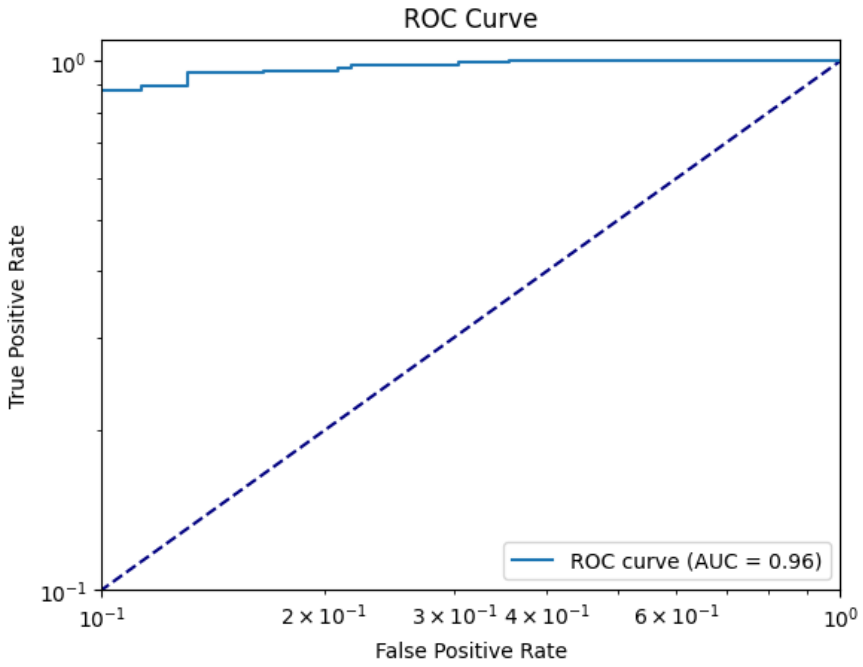

Figure 5.6: Log ROC curve of LAION5v+ face vs VGG face, 1200 images, stable-diffusion

TPR with low FPR in 5.6 is:

- Tpr for fpr <= 0.1 using MLP is: 0.8805

5.5 Stable-diffusion, privacy attack (profile photos) with NO BG, 1200 images, LAION5v+ face vs VGG face

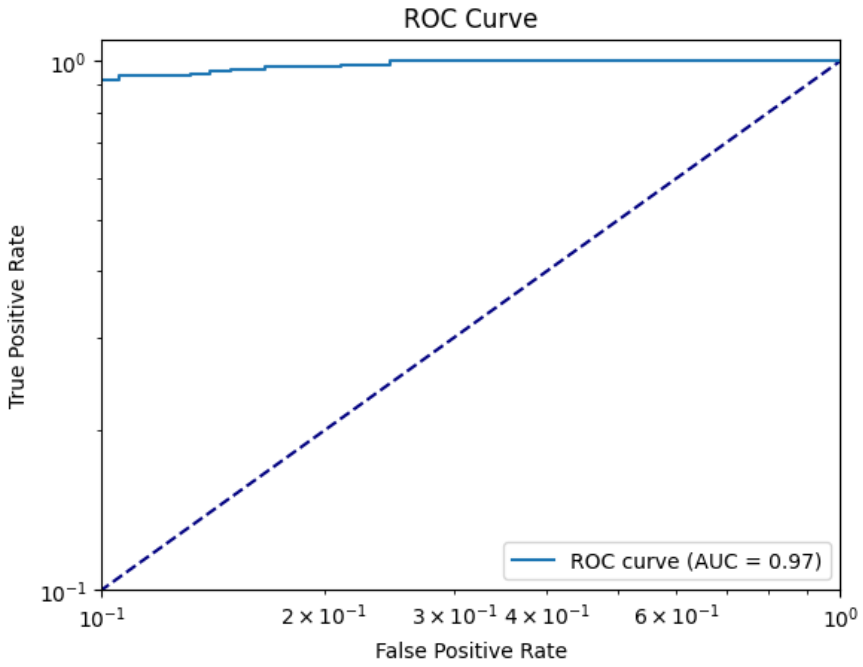

Figure 5.7: Log ROC curve of LAION5v+ face vs VGG face, NO Background, 1200 image, stable-diffusion

TPR with low FPR in 5.7 is:

- Tpr for fpr <= 0.1 using MLP is: 0.9230

The removal of background brought a 0.04 increase in TPR with fpr <= 0.1 using just 1200 images.

5.6 Stable-diffusion, prompt attack 2000 images, LAION5v+ vs MS COCO

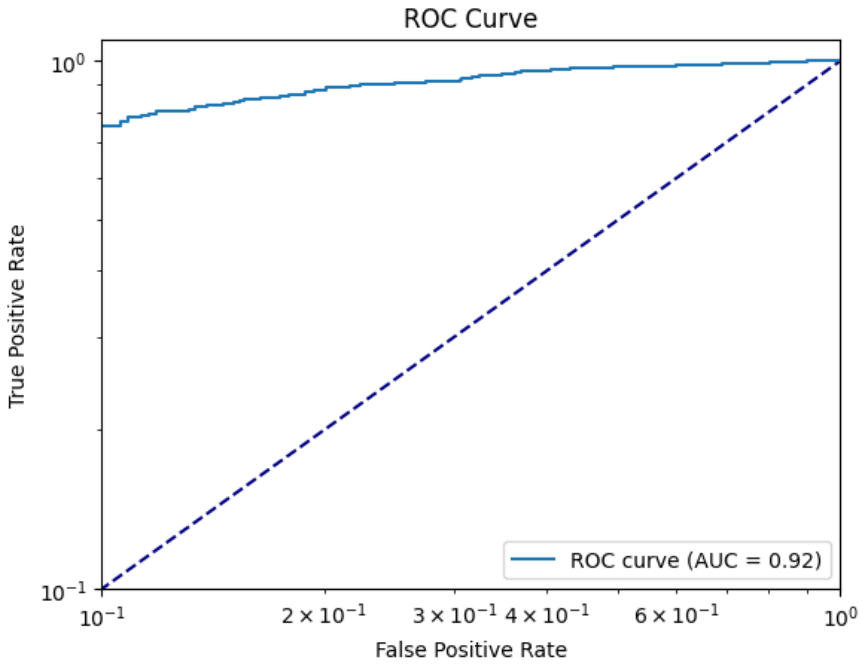

Figure 5.8: Log ROC curve of LAION5v+ vs MS COCO 2000 image, stable-diffusion

TPR with low FPR in 5.7 is:

- Tpr for fpr <= 0.1 using MLP is: 0.7518

In this attack is importat to consider also the FID between the images generated with stable-diffusion from the original prompt of the training set (1st generation) and the images generated from the prompt captioned directly from the 1st generation images (2nd generation).
To understand if the FID between the two generation of images is small and thus, if the prompt generate similar content between the two generations, the images need to belong to a similar distribution. If that is the case, the prompts from the original training set generate similar content to the prompts extracted with BLIP on the classified 'in' images of the attack.

The FID obtained on 500 1st generation images and 500 2nd generation images is: **3736.850**.

Compared to the FIDs in 5.7, it is pretty small, this confirms the attack utility.

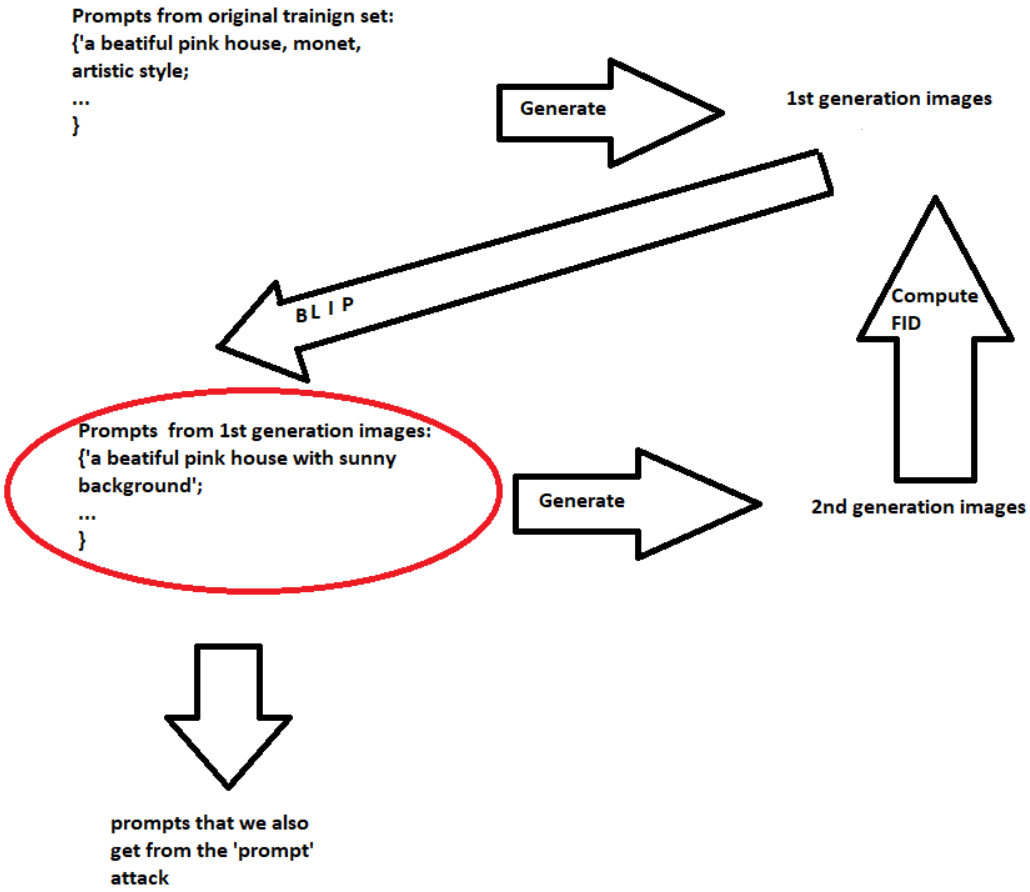

Figure 5.9: A schema on the validation of the 'prompt' attack

This attack is also useful to gather more prompts to generate synthetic data and create shadow models in order to perform other types of attack.

5.7 FIDs

FID between the original images of the various training sets and the corresponding generated images with stable-diffusion, 500 images sample:

	FID
COCO	4192.455
VGG	5874.106
LAION	4018.502
CC	3568.233

The smaller the FID and the better the model recognize the images as part of its training set.

As mentioned before I had some trouble with CC dataset, probably since the methods of extraction are very similar to the LAION ones and there are some overlapping of images, thus is difficult to perform a MIA against CC dataset.

If I decided to use more images as sample the FID would have decreased, however it is observable except for CC that with the LAION dataset stable-diffusion is able to create images more similar to the originals since it was its training set.

5.8 Comparison with state-of-the-art

It is hard to have direct comparison with state-of-the-art MIAs, since the model attacked and dataset can change heavily.
In this section I will compare the results obtained with others state of the art results taken from [33].

TPR @ 0.1% FPR

	C-10	C-100	WT103	Others
Yeom et al.[35]	0.0%	0.0%	0.1%	-
Shokri et al.[25]	0.3%	1.6%	-	-
Jayaraman et al.[36]	0.0%	0.0%	-	-
Song and Mittal. [37]	0.1%	1.4%	-	-
Sablayrolles et al. [38]	1.7%	7.4%	1.0%	-
Long et al. [39]	2.2%	4.7%	-	-
Watson et al. [40]	1.3%	5.4%	1.1%	-
Carlini et al. [33]	8.4%	27.6%	1.4%	-
Attack II-S	-	-	-	**69%**
Privacy attack, no BG	-	-	-	**92%**
Prompt attack	-	-	-	**75%**

Chapter 6

Conclusions

I have presented how to achieve state-of-the-art results with MIAs in a scarce resource environment.

I re-implemented the II-S attack presented in the paper [13] with a focus on log scale ROC and TPR at 0.1% FPR, achieving 69,97% of TPR on the stable-diffusion model.

I tried the same attack on different diffusion models, including Karlo v1 alpha, which follows a different architecture and dataset of stable-diffusion I obtained a TPR of 51,44% and on Dreamlike photoreal 2.0, which is a fine-tuned version of stable-diffusion, I obtained a TPR of 70.33%.

I brought an improvement in results on the attack against profile images from a TPR of 88.05% with the methodology of the paper [13] to a TPR of 92.30% with my new method.

I was able to develop an alternative MIA with a focus on the content that prompts generate, with a TPR of 75.18% on a low FID between the generated images from the original prompt and the prompts gathered from the attack.

I hope this research will be useful in better understanding the foundations of MIAs and how to develop them even without much resources, and I also hope to have brought new ideas for future work; in particular, I am expecting new research in the privacy field with a focus on how to protect against MIAs since the current always cited differential privacy does not solve all the issues alone, but it can be very useful in the context of the genera-

tion of synthetic training sets.

The limitations of this work are also its strenghs.

The MIAs can be performed in a scarce resource environment; however, if a part of the training set needs to be disclosed, it is then impossible to perform the same attack in a complete full-box setting.

More generally, there are many open questions that I hope will be investigated further, including differential privacy in text-to-image models and how to prevent MIAs.

Bibliography

[1] Ian J. Goodfellow, J. Pouget-Abadie, M.ehdi Mirza, B. Xu, D. Warde-Farley, S. Ozair†, A. Courville, Y. Bengio: Generative Adversarial Nets

[2] Diederik P. Kingma, M. Welling: An Introduction to Variational Autoencoders

[3] Dor Bank, Noam Koenigstein, Raja Giryes: Autoencoders, `https://arxiv.org/pdf/2003.05991.pdf`

[4] D. P. Kingma; M. Welling: An Introduction to Variational Autoencoders

[5] G. Bhaskar, D. Indira, C. Albert, T. Michael, B. Magdy: An Empirical Analysis of Generative Adversarial Network Training Times with Varying Batch Sizes

[6] J. Ho, A. Jay, P. Abbeel: Denoising Diffusion Probabilistic Models

[7] J. Song, C. Meng, S. Ermon: DENOISING DIFFUSION IMPLICIT MODELS

[8] P. Dhariwal, A. Nichol: Diffusion Models Beat GANs on Image Synthesis

[9] A. Nichol and P. Dhariwal: Improved denoising diffusion probabilistic models.

[10] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskeve,r M. Chen: GLIDE: Towards Photorealistic Image Generation and

Editing with Text-Guided Diffusion Models

[11] J. Ho, and T. Salimans, Classifier-free diffusion guidance.

[12] O. Ronneberger, P. Fischer, and T. Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation

[13] Y. Wu; N. Yu; Z. Lil; M. Backes; Y. Zhang: Membership Inference Attacks Against Text-to-image Generation Models

[14] B. Jayaraman and D. Evans: Evaluating Differentially Private Machine Learning in Practice

[15] N. Carlini; J. Hayes; M. Nasr; M. Jagielski; V. Sehwag; F. Tramer; B. Balle; D. Ippolito; E. Wallace: Extracting Training Data from Diffusion Models

[16] R. Rombach; A. Blattmann1; D.Lorenz; P.Esser; B. Ommer: High-Resolution Image Synthesis with Latent Diffusion Models

[17] A. Narayanan; V. Shmatikov: Robust De-anonymization of Large Datasets (How to Break Anonymity of the Netflix Prize Dataset

[18] R. Danger Differential Privacy: What is all the noise about?

[19] M. Abadi; A. Chu; I: Goodfellow; H. B. McMahan; I. Mironov; K. Talwar; L. Zhang: Deep Learning with Differential Privacy

[20] A. Ramesh; P. Dhariwal; A. Nichol; C. Chu; M. Chen: Hierarchical Text-Conditional Image Generation with CLIP Latents

[21] D. Lee, J. Kim; J. Choi; J. Kim; M. Byeon; W. Baek;S. Kim: Karlo-v1.0.alpha on COYO-100M and CC15M

[22] T. Dockhorn; T.Cao1; A. Vahdat; K. Kreis: DIFFERENTIALLY PRIVATE DIFFUSION MODELS

[23] K. Simonyan; A. Zisserman: VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

[24] J. Li; D. Li; C. Xiong; S. Hoi: BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation

[25] R. Shokri; Marco Stronati; C. Song; V. Shmatikov: Membership Inference Attacks Against Machine Learning Models

[26] B. Kulynych; M. Yaghini; G. Cherubin; M. Veale; C.Troncoso: Disparate Vulnerability to Membership Inference Attacks

[27] S. Yeom; I. Giacomelli; M. Fredrikson; S. Jha; Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting

[28] To Recognize Long-Tail Visual Concepts S. Changpinyo; P. Sharma: N. Ding: R. Soricut: Conceptual 12M: Pushing Web-Scale Image Text Pre-Training

[29] C. Schuhmann; R. Vencu; R. Beaumont; R. Kaczmarczyk; C. Mullis; A. Katta; T. Coombes; J. Jitsev; A. Komatsuzaki: LAION-400M: Open Dataset of CLIP-Filtered 400 Million Image-Text Pairs

[30] Omkar M. Parkhi; A. Vedaldi; A. Zisserman: Deep Face Recognition

[31] T. Lin; M. Maire; S. Belongie; L. Bourdev; R. Girshick; J. Hays; P. Perona; D. Ramanan; C. L. Zitnick; P. Dollar: Microsoft COCO: Common Objects in Context

[32] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, Ilya Sutskever: Learning Transferable Visual Models From Natural Language Supervision

[33] N. Carlini; S. Chien; M. Nasr; S. Song; A. Terzis; F. Tramer: Membership Inference Attacks From First Principles

[34] Ruikang Yang; Jianfeng Ma;Yinbin Miao; Xindi Ma: Privacy-preserving generative framework for images against membership inference attacks

[35] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In 2018 IEEE 31st Computer Security Foundations Symposium (CSF), pages 268–282. IEEE, 2018.

[36] Bargav Jayaraman, Lingxiao Wang, David Evans, and Quanquan Gu. Revisiting membership inference under realistic assumptions. In Proceedings on Privacy Enhancing Technologies (PoPETs), 2021.

[37] Liwei Song and Prateek Mittal. Systematic evaluation of privacy risks of machine learning models. In 30th USENIX Security Symposium (USENIX Security 21), 2021

[38] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, Yann Ollivier, and Herve J ´ egou. White-box vs black-box: Bayes ´ optimal strategies for membership inference. In International Conference on Machine Learning, pages 5558–5567. PMLR, 2019.

[39] Yunhui Long, Lei Wang, Diyue Bu, Vincent Bindschaedler, Xiaofeng Wang, Haixu Tang, Carl A Gunter, and Kai Chen. A pragmatic approach to membership inferences on machine learning models. In 2020 IEEE European Symposium on Security and Privacy (EuroSP), pages 521–534. IEEE, 2020

[40] Lauren Watson, Chuan Guo, Graham Cormode, and Alex Sablayrolles. On the importance of difficulty calibration in membership inference attacks. arXiv preprint arXiv:2111.08440, 2021.