SAPIENZA
UNIVERSITÀ DI ROMA

Classifier-free Diffusion Guidance with a Robust Energy-Based Classifier

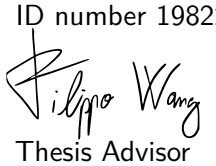
Facoltà di Ingegneria dell'Informazione, Informatica e Statistica

Corso di Laurea in Applied Computer Science and Artificial Intelligence

Candidate

Filippo Wang

ID number 1982262



Thesis Advisor

Prof. Iacopo Masi

Co-Advisor

Dr. Maria Rosaria Briglia

Academic Year 2023/2024

SAPIENZA
UNIVERSITÀ DI ROMA

Classifier-free Diffusion Guidance with a Robust Energy-Based Classifier

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica

Corso di Laurea in Applied Computer Science and Artificial Intelligence

Candidate

Filippo Wang
ID number 1982262

Thesis Advisor

Prof. Iacopo Masi

Co-Advisor

Dr. Maria Rosaria Briglia

Academic Year 2023/2024

Thesis defended on 22 July 2024
in front of a Board of Examiners composed by:

Prof. Tiziana Calamoneri (chairman)
Prof. Daniele Gorla
Prof. Angelo Monti
Prof. Iacopo Masi
Prof. Emanuele Panizzi
Prof. Manuela Petti
Prof. Gabriele Tolomei

Classifier-free Diffusion Guidance with a Robust Energy-Based Classifier
Bachelor's thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: wang.filippo02@gmail.com

In deepest gratitude to my mother and my brother,
who worked tirelessly to give me a better future.

Abstract

This thesis is the result of the work conducted at the OmnAI research lab based in Sapienza University of Rome from February 2024 to June 2024. The main objective is to study the generative properties of adversarially trained classifiers and exploit their robustness to make a diffusion model perform conditional sampling or, in other words, classifier guidance.

This research bridges an important gap in the field of Diffusion Models' conditional image generation since it is currently believed that it is not possible to use an off-the-shelf robust classifier to achieve classifier guidance. We challenge this belief by demonstrating that robust classifiers can effectively guide unconditional diffusion models without having to re-train a time-dependent classifier based on the DM's noise scheduler. In our work, we introduce novel approaches to successfully achieve this task by interpreting a Standard Adversarially Trained classifier as a generative Energy-Based Model, effectively achieving Classifier-free guidance in a Diffusion Model yet using a Robust Classifier.

Contents

Chapter 1

Introduction

Since the advent of artificial intelligence (AI), researchers have aspired to create machines capable of generating images as realistic as those perceived by the human eye. This pursuit involves developing algorithms that can produce new images by learning the probability distribution of data over existing ones. Over the years, the field has witnessed extraordinary progress, primarily fueled by advancements in machine learning (ML), most notably deep learning.

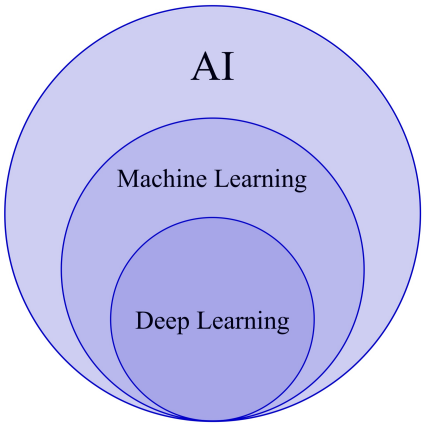


Figure 1.1. A Venn diagram showing the idea of the relation between AI, Machine Learning and Deep learning.

1.1 Discriminative vs Generative AI

AI models can be broadly categorized into discriminative and generative. Discriminative models, such as those used in image classification, focus on distinguishing between different data classes by learning decision boundaries ([1] Bishop 2006). Convolutional neural networks (CNN) ([2] O'Shea and Nash 2015) are excellent at tasks like object recognition; in particular, the creation of AlexNet by [3] Krizhevsky et al. 2012 revolutionized the way to approach image classification; it was one of the first significant deep-learning models to be trained using Graphics processing units (GPU). In fact, they were able to train a CNN so well that it broke the record for image classification and localization metrics, winning the ImageNet Large Scale Visual Recognition Challenge 2012;

Despite the great performance of Deep Learning Models, [4] Szegedy et al. 2014 discovered that Neural Networks (in particular discriminative models) suffer from adversarial attacks that demolish any state-of-the-art model's performance metrics. Adversarial attacks are small data perturbations, often undetectable by human perception. Therefore, robust classifiers were trained to resist these attacks.

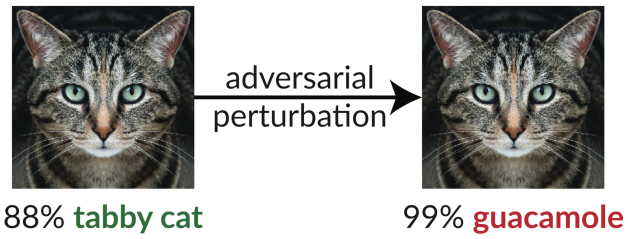


Figure 1.2. An imperceptible perturbation in the image fools the InceptionV3 network by Google into believing that a cat is guacamole. Image taken from [1].

Generative models aim to understand the underlying input data distribution to create new, synthetic data points. A groundbreaking innovation in this realm was the introduction of Generative Adversarial Networks (GAN) by [5] Goodfellow et al. 2014. GANs consist of two neural networks—a generator and a discriminator—that engage in a dynamic adversarial training process where the discriminator learns to recognize fake images from real images. In contrast, the generator tries to fool the discriminator during each training step by generating images that are indistinguishable from the real ones .

Recently, [6] Sohl-Dickstein et al. 2015 introduced diffusion models (DM), which transform pure Gaussian noise sampled from a standard normal distribution into complex and high-quality images by approximating the inverse of a stochastic differential equation (SDE) defined in the forward process.

After that, [7] Ho et al. 2020 introduced Denoising Diffusion Probabilistic Models (DDPM), setting top benchmarks in image synthesis by refining noisy images into clear ones through a series of denoising steps. [8] Dhariwal and Nichol 2021 showed that diffusion techniques can surpass GANs in generated image quality and diversity metrics such as Frechet Inception Distance (FID) ([9] Heusel et al. 2018) and Inception Score (IS).

Figure 1.3. An astronaut riding a horse generated by Dall-E 3 through ChatGPT-4o.

[1] https://github.com/anishathalye/obfuscated-gradients

Further enriching the landscape are Energy-Based Models (EBMs), which utilize energy functions to model data distributions. [10] Lecun et al. 2006 highlighted that EBMs provide robust learning algorithms for complex data distributions, adding to the diversity of generative modelling techniques. Meanwhile, Variational Autoencoders (VAEs), introduced by [11] Kingma and Welling 2022, combined neural networks with probabilistic graphical models to generate new data points from learned latent representations.

Recent years have witnessed astonishing advancements proven by models like DALL-E by OpenAI, Stable Diffusion by Stability AI and Midjourney, which can generate ultra-detailed and diverse images from textual descriptions exploiting the power of diffusion models, and pushing the frontiers of creativity and design. Moreover, recent innovations like Sora by OpenAI have extended these capabilities even further, enabling the generation of videos and expanding the horizons of AI in visual media. In figure 1.3, you can see an image we have generated using Dall-E 3; Nowadays, anyone can write a prompt to generate images that match similar quality from text-to-image models available online.

Although generative models are incredibly capable, they require massive amounts of training data and enormous computing power costs, making it prohibitively expensive to retrain them often.

1.2 Thesis Outline

The thesis begins by recalling some background theory of Diffusion Models and their evolution over time, starting from concepts rooted in Statistical Mechanics and Physics, such as discrete-time Markov chains and Langevin Dynamics. Next, we discuss the definition of score functions and their application in modeling the underlying input data distribution to gradually transform a random data point sampled from a standard Gaussian distribution into a realistic image. Following this, we explore DDPMs and DDIMs, which are practical implementations of Diffusion Models through the training of a time-dependent denoising neural network. We also study the theoretical derivation and practical application of conditional sampling, thanks to Classifier Guidance and Classifier-free Guidance. Subsequently, we explain how an adversarial classifier is trained using the Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD) attacks. We conclude the background chapter by looking into the process of converting a discriminative neural network into a generative energy-based model.

In the related work section, we review three papers that are closely linked to our research, highlighting how their discoveries provide valuable insights into the behaviour of robust models in a generative modeling scenario.

The fourth chapter of this thesis dives into the theoretical study and practical experiments of our research, with a primary focus on enabling a robust classifier to successfully guide a diffusion model.

Finally, we conclude with a brief summary of our work and outline our future research directions.

1.3 Contribution

To achieve classifier guidance using a pre-trained diffusion model, a separate time-dependent classifier must typically be trained on noisy images to successfully induce conditional image synthesis during the reverse diffusion process.
It is currently believed that:

> "*... an off-the-shelf adversarially trained robust classifier would not fit our purpose in this context...*"

as stated by [12] Kawar et al. 2023 in his proposed method section.

This thesis explores the generative properties of robust classifiers and how they can be utilized to successfully achieve classifier guidance. After reproducing the results of the pre-trained diffusion model and the robust classifiers, we analyze the connection between a classifier trained with standard adversarial training using ε-ball projected gradient descent attacks and their robustness to Gaussian noise added during the forward diffusion process.

We also examine the interpretation of the same classifiers when turned into energy-based models and how their gradients influence the conditional generation process. We find that the training ε-radius of a robust classifier is correlated with the quality of the images it helps to generate, with the more robust models leading to fewer hallucinations.

Additionally, we analyze and interpret the differences between the energy of the natural training data (in this case, CIFAR-10) and the conditionally sampled data, providing insights into the role of EBMs in generative modeling. In our experiments, we find that class-conditional energy is more informative than the ordinary class-conditional probability.

Throughout the research, we continuously experiment with different settings, providing the intuition behind certain methods as well as a formalized methodology to allow reproducibility. We argue how our choices are supported by existing literature and "grounded" intuition, with the goal of always achieving better outcomes. Along the experiments, we show class-conditioned samples generated by the relative method, proving that classifier guidance is actually achievable by means of a robust classifier.

Chapter 2

Background

Before going into the details of our study, let's establish the concepts needed for a comprehensive understanding of the material. This chapter provides formal definitions of the mathematical models and some intuitions behind them. But first, let's understand, in general, how a model learns to generate new images.

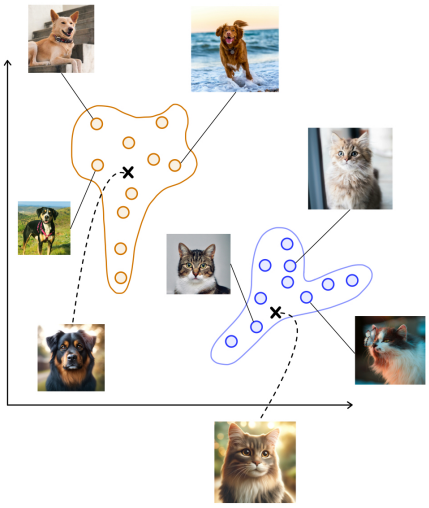


Figure 2.1. High-level illustration of sampling new images from a model distribution p_θ.

Assume we want to generate images of cats and dogs; these two classes of images each contain common features and underlying patterns. In machine learning, it is assumed that the data points (in this case, images of cats and dogs) are i.i.d., forming an unknown underlying probability distribution p, called **data distribution**; in general, a model's objective is to approximate the latter as accurately as possible by tuning its (usually) numerous parameters represented by θ, and so, shaping its own representation of "reality", the **model distribution** p_θ. The **dataset** is sampled from p, and is denoted as p_{data}. Therefore, p_θ is learned by approximating the distribution of the training set $X \sim p_{\text{data}}$, so that we can draw new images from the learned p_θ, and hopefully, they are realistic pictures of cats and dogs that do not actually "exist"!

We really want to put some emphasis on the last statement since it is relatively trivial for a model to generate images already present in the dataset; the true challenge of Generative Models is to create diverse and high-quality images that are

indistinguishable from real ones without requiring excessive computational power.

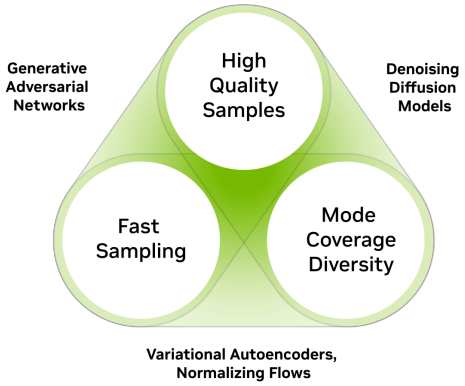


Figure 2.2. The generative trilemma. Image from Nvidia[1].

2.1 From Statistical Mechanics to Deep Learning

A discrete-time Markov Chain (DTMC) is a sequence of random variables characterized by the **memorylessness** property, meaning that the probability of future states depends only on the current state; it is easily formalized as follows:

Let $\{X_n : n \geq 0\}$ be a positive recurrent Markov Chain, then $\forall n \geq 0$

$$\Pr\{X_{n+1} \mid X_n, X_{n-1}, \ldots, X_0\} = \Pr\{X_{n+1} \mid X_n\} \tag{2.1}$$

Another essential property of Markov Chains under certain conditions is **reversibility**, which was leveraged for generating images by [6] Sohl-Dickstein et al. 2015. The authors introduced a novel generative AI algorithm, where the primary objective was to transform a simple known distribution (e.g., a Gaussian distribution) into a target data distribution through a diffusion process. However, it's important to note that this work didn't directly build upon traditional Markov Chains; instead, a probabilistic model was explicitly defined as the endpoint of the Markov Chain. The idea of gradually converting one distribution to another is directly inspired by non-equilibrium statistical physics [13] Jarzynski 1997.

Furthermore, the formulation of this approach includes mathematical tools beyond Markov Chains. Concepts from physics, such as Langevin dynamics ([14] Langevin 1908), which is the stochastic realization of the Fokker-Planck equation, were utilized to define a Gaussian diffusion process that attains a distribution as its equilibrium. Additionally, the Kolmogorov forward and backward equations ([15] Feller 1949) were employed to prove that many forward diffusion processes can be described with the same functional form as their reverse processes.

[1] `https://www.nvidia.com/en-us/glossary/generative-ai`

2.2 Score-based generative modeling

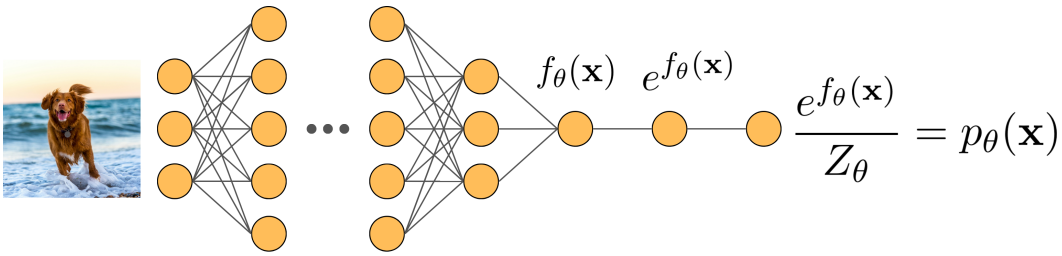


Figure 2.3. High-level illustration of a DNN mapping images to a probability measure.

It is extremely challenging to estimate the underlying probability distribution of high dimensional data accurately (such as images, audio, and video). However, we can build a deep neural network (DNN) that can learn complex probability distributions. Yet, it is not so trivial to make a DNN represent such distributions. Typically, a DNN is seen as a black box that, given a high-dimensional data point $\mathbf{x}$, computes a one-dimensional value $f_\theta(\mathbf{x})$. To make the function non-negative over the entire domain, we can take the exponential $e^{f_\theta(\mathbf{x})}$ and divide it by a normalizing constant Z_θ, thus correctly defining a probability density function (pdf) $p_\theta(\mathbf{x})$, where θ represents the weights of the model.

By definition, $Z_\theta := \int e^{f_\theta(\mathbf{x})}\, d\mathbf{x}$, and even in the discrete case, the computation of this constant term is #P-complete, which means that it is at least as hard as NP-complete problems, and thus intractable.

To overcome this issue, [16] Song and Ermon 2020 introduced the usage of score functions to build score-based models. Given a pdf $p_\theta(\mathbf{x})$, the score function (or simply score) is defined as:

$$s_\theta(\mathbf{x}) \triangleq \nabla_x \log p_\theta(\mathbf{x}) \tag{2.2}$$

On the other hand, score functions are computationally feasible and theoretically retain all the information of $p_\theta(\mathbf{x})$. In principle, one can compute the score function from the pdf by taking the gradient and, conversely, reconstruct the pdf by integrating the score function. The score function represents a vector field that indicates the direction in which the pdf increases most rapidly.

But how does it avoid computing the normalizing constant Z_θ? Thanks to the properties of logarithms, we can parametrize the score as the gradient of the energy-based model f_θ.

$$\begin{aligned} s_\theta(\mathbf{x}) &= \nabla_x \log p_\theta(\mathbf{x}) = \nabla_x \log \frac{e^{f_\theta(\mathbf{x})}}{Z_\theta} \\ &= \nabla_x f_\theta(\mathbf{x}) - \underbrace{\nabla_x \log Z_\theta}_{=0} = \nabla_x f_\theta(\mathbf{x}) \end{aligned} \tag{2.3}$$

[1] https://yang-song.net/blog/2021/score/

From here, to build a score-based model, we shall estimate the score function of the underlying density function by training a score model such that it minimizes the difference between the model's score and the true score, figure 2.4 shows an illustration of Score function (the vector field) and respective density function (the contour); it is easy to compare by subtraction the two vector fields since they lie in the same vector space, but since we don't have access to the true score, [17] Hyvärinen 2005 formulated an equivalent estimation method called **score-matching**, is formalized as follows:

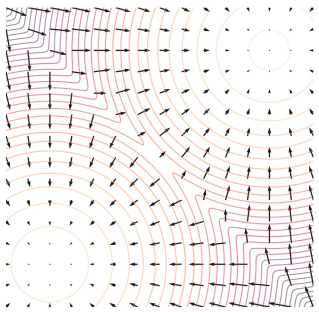


Figure 2.4. Score and pdf of a mixture of two Gaussians.[2]

given $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n | n > 0\} \sim p_{data}(\mathbf{x})$; learn $s_\theta(\mathbf{x}) \approx p_{data}(\mathbf{x})$

by minimizing the Fisher divergence

$$\frac{1}{2}\mathbb{E}_{p_{\text{data}}(\mathbf{x})}\left[\|\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - s_\theta(\mathbf{x})\|_2^2\right] \tag{2.4}$$

equivalent to

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x})}\left[\frac{1}{2}\|s_\theta(\mathbf{x})\|_2^2 + \text{trace}\left(\nabla_{\mathbf{x}} s_\theta(\mathbf{x})\right)\right]$$

$$\approx \frac{1}{n}\sum_{i=1}^{n}\left[\frac{1}{2}\|s_\theta(\mathbf{x}_i)\|_2^2 + \text{trace}\left(\nabla_{\mathbf{x}} s_\theta(\mathbf{x}_i)\right)\right] \tag{2.5}$$

where $\nabla_{\mathbf{x}} s_\theta(\mathbf{x})$ is the jacobian matrix of $s_\theta(\mathbf{x})$.

This "vanilla version" of score-matching becomes very computationally expensive as the input size grows due to the presence of trace $(\nabla_{\mathbf{x}} s_\theta(\mathbf{x}))$. Evaluating the trace of the Hessian matrix $H_{\log p_\theta}(\mathbf{x}) = \nabla_{\mathbf{x}} s_\theta(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^D$ requires D times more backpropagation passes compared to computing just $s_\theta(\mathbf{x})$ [18] Song et al. 2019. Thus, this method scales poorly with increasing input dimensions and network complexity.

We introduce Denoising Score Matching by [19] Vincent 2011, where we can estimate the score of corrupted data points:

$$\frac{1}{2n}\sum_{i=1}^{n}\|s_\theta(\tilde{\mathbf{x}}_i) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}_i \mid x_i)\|_2^2 \tag{2.6}$$

where $q_\sigma(\tilde{x}_i)$ represents x_i perturbed by some noise

and

$$\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \sim p_{data}(\mathbf{x}) \wedge \{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots, \tilde{\mathbf{x}}_n\} \sim q_\sigma(\tilde{\mathbf{x}}) \;\mid\; \tilde{\mathbf{x}}_i \sim q_\sigma(\tilde{\mathbf{x}}_i \mid \mathbf{x}_i)\ \forall i$$

If we choose a Gaussian smoothing kernel,

$$q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}} \mid \mathbf{x}, \sigma^2 I)$$

the loss further simplifies to

$$\mathcal{L}(\theta;\sigma) \triangleq \frac{1}{2n}\sum_{i=1}^{n}\left\| s_\theta(\tilde{\mathbf{x}}_i) + \frac{\tilde{\mathbf{x}}_i - \mathbf{x}_i}{\sigma^2}\right\|_2^2 \tag{2.7}$$

We can apply Stochastic Gradient Descent (SGD) to minimize the above-defined loss function. Since the objective is to model the score on natural data points, we want to choose a very small σ. However, if σ is too small, it causes the loss variance to explode.

Algorithm 1: Langevin Dynamics.

$\mathbf{x}_0 \sim \pi(x)$; `// Initialize from any prior distribution`
Choose ϵ and T such that $\epsilon \to 0$ and $T \to \infty$;
for $t \leftarrow 1, 2, ..., T$ **do**
  $\mathbf{z}_t \sim \mathcal{N}(0, I)$;
  $\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} + \frac{\epsilon}{2}\nabla_x \log p(\mathbf{x}_{t-1}) + \sqrt{\epsilon}\mathbf{z}_t$;

To generate new samples from a trained score function $s_\theta(\mathbf{x})$ we can leverage **Langevin Dynamics** by randomly sampling any data point in the relative vector space and iteratively let it "flow" towards high-density locations of $p(\mathbf{x})$ by following the vector field given by the gradients represented in $s_\theta(\mathbf{x})$. To avoid convergence to a single point, some level of Brownian motion is added at each step, making the process stochastic. It can be imagined like a particle taking small steps, following directions predicted by the score function, and at each step, it gets pushed by some invisible random force until it reaches an area where it is highly probable that natural images exist.

Algorithm 1 demonstrates an approach for sampling from $p(\mathbf{x})$ using $\nabla_x \log p(\mathbf{x})$. It guarantees that $\mathbf{x}^T \sim p(\mathbf{x})$, and since $s_\theta(\mathbf{x}) \approx \nabla_x \log p(\mathbf{x})$, we can easily replace $\nabla_x \log p(\mathbf{x})$ with our score model $s_\theta(\mathbf{x})$.

However, having chosen a small σ, the trained score function is accurate only in high-density regions. This results in inaccurate gradients in low-density regions, as the model is trained to minimize the loss on samples with little perturbation. By definition, these samples lie very close to the natural data points, so they belong to high-density regions.

To address this issue, we can train a score model $s_\theta(\mathbf{x}, \sigma_i)$ conditioned by the variance of the chosen corruption kernel. By selecting multiple noise levels $\sigma_1 < \sigma_2 < \ldots < \sigma_N$, the score model becomes accurate across the entire vector space, from low data-density regions to high data-density regions. Figure 2.5 provides a high-level visualization of a jointly trained score function. We call it Noise Conditional Score Network (NCSN), and to train the score model on multiple levels of noise jointly (i.e. $\forall\sigma \in \{\sigma_i\}_{i=1}^L : s_\theta(\mathbf{x}, \sigma) \approx \nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x})$).
We call this model the Noise Conditional Score Network (NCSN). To train the score model on multiple levels of noise jointly (i.e., $\forall\sigma \in \{\sigma_i\}_{i=1}^L : s_\theta(\mathbf{x}, \sigma) \approx \nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x})$)

we can redefine the objective function loss as

$$\mathcal{L}(\theta; \{\sigma_i\}_{i=1}^L) \triangleq \frac{1}{L}\sum_{i=1}^{L}\lambda(\sigma_i)\mathbb{E}_{p_{\sigma_i}(\mathbf{x})}\left[\|\nabla_{\mathbf{x}} \log p_{\sigma_i}(\mathbf{x}) - s_\theta(\mathbf{x}, \sigma_i)\|_2^2\right] \tag{2.8}$$

[3] `https://yang-song.net/blog/2021/score/`

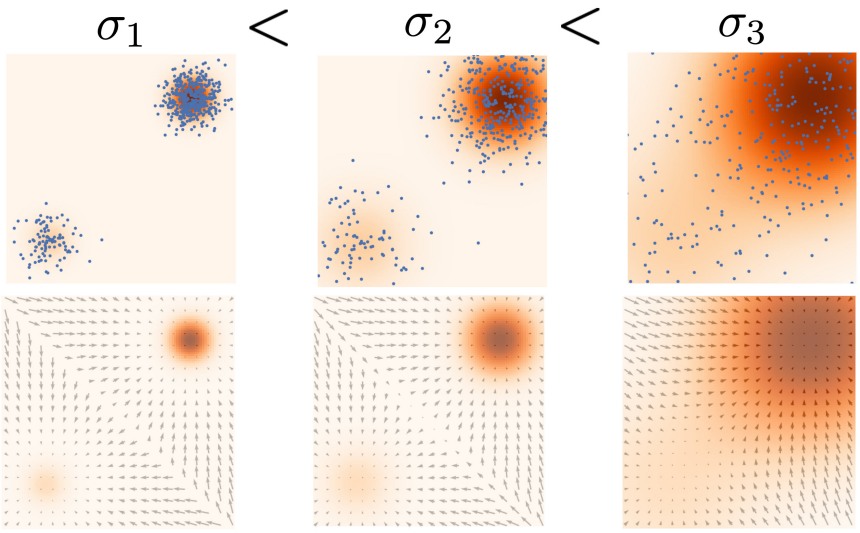


Figure 2.5. First row shows data points from a mixture of two Gaussians perturbed with three levels of Gaussian noise, and the **second row** visually shows the score model for each level of noise. Image taken from blog [3].

where $p_{\sigma_i}(\mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}} \mid \mathbf{x}, \sigma^2 I)$ is the probability density function of perturbed data points at noise level σ_i, and $\lambda(\sigma_i)$ is some positive weighting function.

[20] Song and Ermon 2020 empirically observed that it feels natural to choose $\lambda(\sigma) = \sigma^2$, and it is highly recommended by the authors to choose $\{\sigma_i\}_{i=1}^L$ as a geometric sequence such that $\frac{\sigma_1}{\sigma_2} = \cdots = \frac{\sigma_{L-1}}{\sigma_L} > 1$.

To generate new samples from a trained score function $s_\theta(\mathbf{x}, \sigma)$, [16] Song and Ermon 2020 propose a method called **Annealed Langevin Dynamics**.

Algorithm 2: Annealed Langevin dynamics.

Require: $\{\sigma_i\}_{i=1}^L$, ϵ, T
Initialize $\tilde{\mathbf{x}}_0$; `// from any fixed prior distribution e.g. uniform`
for $i \leftarrow 1$ **to** L **do**
  $\alpha_i \leftarrow \epsilon \cdot \frac{\sigma_i^2}{\sigma_L^2}$; `//` α_i `is the step size`
  for $t \leftarrow 1, 2, ..., T$ **do**
    Draw $\mathbf{z}_t \sim \mathcal{N}(0, I)$;
    $\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2} s_\theta(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i}\mathbf{z}_t$;
  $\tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$;
return $\tilde{\mathbf{x}}_T$;

Intuitively, for each level of noise σ_i, it samples from $p_{\sigma_i}(\mathbf{x})$ using naïve Langevin Dynamics, as defined in Algorithm 1, with progressively smaller step sizes α. In the final iteration, it (hopefully) samples from $p_{\sigma_L}(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$ when $\sigma_L \approx 0$.

Figure 1 is a visual example of the ability of Annealed Langevin Dynamics to recover the original density distribution.

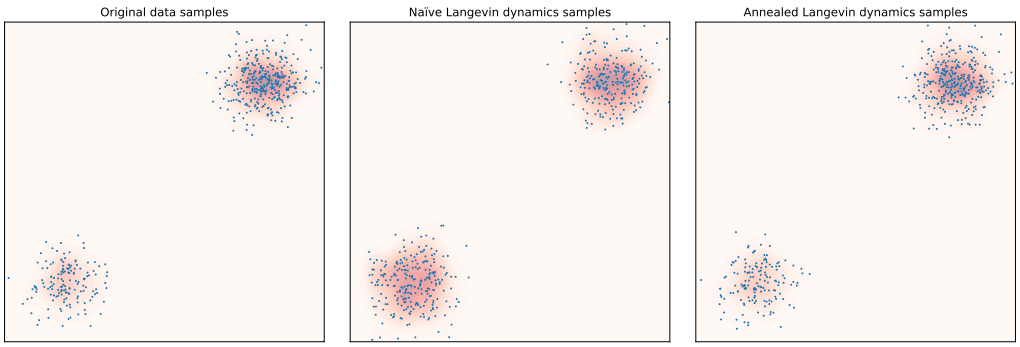


Figure 2.6. Simulation of sampling data points using Naïve Langevin dynamics vs Annealed Langevin dynamics.

2.3 Denoising Diffusion Probabilistic Models

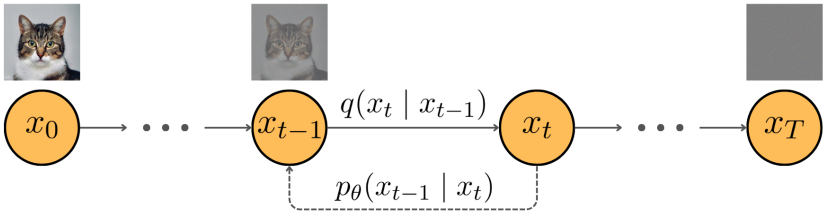


Figure 2.7. Forward and Reverse Diffusion chain.

[7] Ho et al. 2020 first revealed an explicit connection between diffusion models and score-based models. More specifically, NCSNs are a generalization of Diffusion Probabilistic Models since the variational lower bound used for training diffusion models is essentially equivalent to a weighted combination of score-matching loss functions.

From [6] Sohl-Dickstein et al. 2015:

> "*The essential idea [...] is to systematically and slowly destroy structure in a data distribution through an iterative forward diffusion process. We then learn a reverse diffusion process that restores structure in data, yielding a highly flexible and tractable generative model*".

In practice, we define a forward diffusion process starting from a natural image x_0 and gradually add Gaussian noise to it over T timesteps. Following this, during the reverse diffusion process, a U-Net is trained to predict the noise added at each timestep, effectively becoming a denoising model.

Formally, the forward diffusion process is formulated as a Markov Chain, where each transition depends only and only on the previous state. More specifically, a Markov diffusion kernel is applied at each timestep to gradually destroy the structure of the data.

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}) \tag{2.9}$$

where β_t is the diffusion rate set by a fixed scheduler.

Intuitively, x_t is x_{t-1} with the addition of some Gaussian noise with variance β_t. Instead of sampling t times during the forward process, we can use the reparametrization trick, $\mathcal{N}(\mu, \sigma) = \mu + \sigma \cdot \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$, to fast-forward the process in one go.

Let $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$

we can rewrite equation 2.9 as

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\epsilon$$

$$= \sqrt{\alpha}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon \tag{2.10}$$

and so we can directly sample $\mathbf{x}_t$ from $\mathbf{x}_0$ by

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \quad \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$$

$$= \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{2.11}$$

The reverse diffusion process involves sampling $\mathbf{x}_{t-1}$ from $\mathbf{x}_t$, meaning that given a noisy image, it predicts a less noisy version of it:

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)) \tag{2.12}$$

where μ_θ and Σ_θ are neural networks that respectively predict the mean and the variance of the time-dependent backward distribution conditioned on the previous image $\mathbf{x}_t$.

In practice, we can decide to fix $\Sigma_\theta = \sigma^2$ to avoid training a second (expensive) network

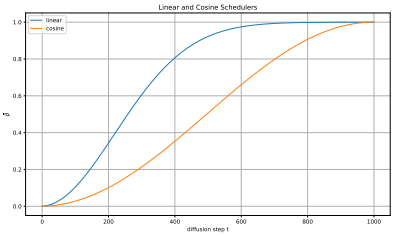
$$p(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma^2) \tag{2.13}$$

and train $\mu_\theta(\mathbf{x}_t, t)$ on the following loss function.

$$\mathcal{L} = \mathbb{E}_{t\sim U(0,T), \mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(0,I)}[||\epsilon - \epsilon_\theta(\mathbf{x}_t, t)||^2] \tag{2.14}$$

This elegant objective function is derived from a beautiful mathematical formulation that sets a variational lower bound on the (intractable) negative log-likelihood of the original image, $-\log(p_\theta(\mathbf{x}_0))$, using the reversed Kullback-Leibler Divergence. It minimizes the distance between the real noise added during the forward diffusion and the subtracted noise during the backward diffusion. Although it's not strictly about additions and subtractions, but it involves a weighted sum to maintain saturation and brightness balance.

(a) Samples from linear (top) and cosine (bottom) schedulers at linearly spaced timesteps from 0 to T. Image taken from [21] Nichol and Dhariwal 2021.



(b) Noise level at each timestep for Linear and Cosine Schedulers.

Figure 2.8. Comparison of Schedulers.

Algorithm 3: Classifier Guided Sampling.

Given : Classifier $p_\phi(y \mid \mathbf{x}_t)$
Input : class label y,
guidance strength s

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
for all t *from* T *to* 1 **do**
  $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
  $\mu \leftarrow \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) + s\sigma_t \nabla_{\mathbf{x}_t} \log p_\phi(y \mid \mathbf{x}_t)$;
  $\mathbf{x}_{t-1} \leftarrow \mu + \sigma_t \epsilon$;
return $\mathbf{x}_0$

Algorithm 4: Classifier-Free Guided Sampling.

Input : class label y,
guidance strength s

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
for all t *from* T *to* 1 **do**
  $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
  $\tilde{\epsilon} \leftarrow (s+1)\epsilon_\theta(\mathbf{x}_t, t, y) - s\epsilon_\theta(\mathbf{x}_t, t)$;
  $\mu \leftarrow \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \tilde{\epsilon} \right)$;
  $\mathbf{x}_{t-1} \leftarrow \mu + \sigma_t \epsilon$;
return $\mathbf{x}_0$

Figure 2.9. Pseudo algorithms of Classifier Guidance Sampling and Classifier-Free Guidance Sampling [4].

In a nutshell, we can easily write down formulas for the forward and backward diffusion processes as follows:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\epsilon \tag{2.15}$$

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) + s\sigma_t + \sqrt{1-\alpha_t}\epsilon \tag{2.16}$$

Regarding the choice of a scheduler, [7] Ho et al. 2020 initially used a linear scheduler. Later on, [8] Dhariwal and Nichol 2021 improved the model by switching to a cosine scheduler, which has the purpose of destroying the information in the image more slowly. Figure 2.8b compares the scheduler functions, and Figure 2.8a provides an example of how noise is gradually added using the two different schedulers. Clearly, the cosine scheduler retains information in the image for longer compared to the linear scheduler.

So far, we have been generating images from any class in the training set. How can we achieve class-conditional sampling? It turns out that it is relatively intuitive. A time-dependent classifier $p_\phi(y|x_t)$ can be trained, meaning it receives as input not only a (noisy) image but also the timestep t of the diffusion chain.

The gradients of $p_\phi(y|x_t)$ point in the direction that most increases the probability of x_t being classified as class y. Therefore, we can slightly modify the sampling

[4] `https://cvpr2022-tutorial-diffusion-models.github.io`

formula to deviate the generation towards class y.

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right) + s\sigma_t\nabla_x \log p_\phi(y|\mathbf{x}_t) + \sigma_t\epsilon \tag{2.17}$$

where $\sigma_t = \sqrt{1-\alpha_t}$, and s is the guidance strength. By tuning this value we trade diversity for quality.

The new conditional sample distribution is defined as

$$\tilde{p}_\theta(x_t|y) \propto p_\theta(x_t|y)p_\phi(y|x_t)^s \tag{2.18}$$

Training a second classifier (a neural network) is expensive. To avoid this, we observe by Bayes' theorem

$$p_\theta(y|x_t) \propto \frac{p_\theta(x_t|y)}{p_\theta(x_t)} \tag{2.19}$$

It follows

$$\tilde{p}_\theta(x_t|y) \propto p_\theta(x_t|y)\left[\frac{p_\theta(x_t|y)}{p_\theta(x_t)}\right]^s = \frac{p_\theta(x_t|y)^{s+1}}{p_\theta(x_t)^s} \tag{2.20}$$

By taking the log space of both sides

$$\log \tilde{p}_\theta(x_t|y) = (s+1)\log \underbrace{p_\theta(x_t|y)}_{\text{conditional}} - s\log \underbrace{p_\theta(x_t)}_{\text{unconditional}} + C \tag{2.21}$$

By doing so, we achieve **Classifier-free guidance** by training the Denoising U-Net both conditionally and unconditionally at the same time. This means that we set a probability p_{uncond} to train $\epsilon_\theta(\mathbf{x}, t, y)$ unconditionally at every iteration.

One of the most significant challenges associated with DDPMs is the considerable inference time required. While the forward process can be directly applied without delays from neural network usage, the reverse process is highly time-consuming, as it necessitates performing T queries to the same neural network. This constraint limits the common usage of the model since it requires substantial computing power to generate images in a feasible time.

A solution to this issue was developed by reconsidering the inference process to minimize the number of iterations required by the generative model. The key concept of this solution lies in examining the objective function expressed in expansive form:

$$L_\gamma(\epsilon_\theta) := \sum_{t=1}^{T} \gamma_t \mathbb{E}_{\mathbf{x}_0\sim q(\mathbf{x}_0),\epsilon_t\sim\mathcal{N}(0,I)}\left[\left\|\epsilon_\theta^{(t)}\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t\right) - \epsilon_t\right\|_2^2\right] \tag{2.22}$$

Here $\epsilon_\theta := \epsilon_\theta^{(t)}\big|_{t=1}^{T}$ represents a set of T functions, each reliant on a set of trainable parameters θ_t, aiming to approximate the noise added at timestep t of the forward process. Observing that these functions rely on the marginal $q(x_t|x_0)$ rather than the joint $q(x_{1,\dots,T} \mid x_0)$, a novel mathematical model for the inference procedure called Diffusion Denoising Implicit Models (DDIM) was proposed by [22] Song et al. 2022. This is a non-Markovian (and thus, deterministic) sampling procedure that does not require to retrain the Diffusion Model.

We define a trainable generative process $p_\theta(x_{0:T})$ where each $p_\theta(x_{t-1} \mid x_t)$ leverages the knowledge of $q_\sigma(x_{t-1} \mid x_t, x_0)$. Intuitively, given a noisy observation x_t, we

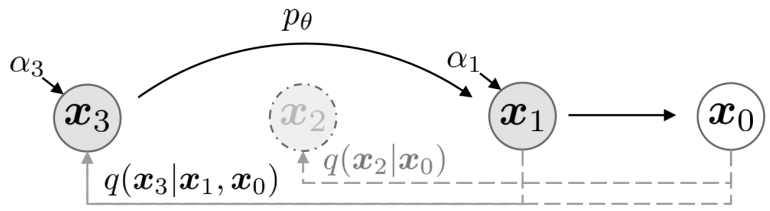


Figure 2.10. Graphical model for accelerated generation, where $\tau = [1, 3]$. image taken from [22] Song et al. 2022

first predict the corresponding x_0, and then use it to obtain a samples x_{t-1} through the reverse conditional distribution $q_\sigma(x_{t-1} \mid x_t, x_0)$.

Thus, we can express x_t as a linear combination of x_0 and a noise variable ϵ:

$$x_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, I) \tag{2.23}$$

The model then tries to predict ϵ_t from x_t, without explicit knowledge of the term x_0. However, rewriting the previously defined function, the term x_0 can be expressed as a function of x_t, defined as the denoised observation approximation function, which predicts x_0 given x_t.

By defining this approximation, a unique form of the generation process is provided:

$$p_\theta^{(t)}(x_{t-1} \mid x_t) = \begin{cases} \mathcal{N}(f_\theta^{(1)}(x_t), \sigma_1^2 I) & \text{if } t = 1 \\ q_\sigma(x_{t-1} \mid x_t, f_\theta^{(t)}(x_t)) & \text{otherwise} \end{cases} \tag{2.24}$$

The generic x_{t-1} sample can now be defined as:

$$\mathbf{x}_{t-1} = \sqrt{\alpha_{t-1}}\left(\frac{\mathbf{x}_t - \sqrt{1-\alpha_t}\epsilon_\theta^{(t)}(\mathbf{x}_t)}{\sqrt{\alpha_t}}\right) + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\epsilon_\theta^{(t)}(\mathbf{x}_t) + \sigma_t(\eta)\epsilon_t \tag{2.25}$$

where $\epsilon_t \sim \mathcal{N}(0, I), \alpha_0 := 1$
and $\sigma_t(\eta) = \eta\sqrt{(1-\alpha_{t-1})/(1-\alpha_t)}\sqrt{1-\alpha_t/\alpha_{t-1}}\ \forall\ t \in \{1, ..., T\}$ such that $\eta \in \mathbb{R}_{\geq 0}$

By providing this sampling generation definition, a general form for representing the entire generation process is established, including DDPM's. This can be achieved by imposing $\eta = 1$. This adjustment renders the forward process Markovian once again, reverting the generative process to DDPM.

Finally, to reduce the number of timesteps required during reverse diffusion without retraining our model, we set a different sub-sequence of timesteps τ from $[1, \ldots, T]$. By setting $\eta = 0$, the sampling becomes deterministic, resulting in "pure" DDIM. By tuning η, we can achieve different levels of stochasticity, corresponding to the amount of Gaussian noise added at each timestep. Figure 2.10 provides a simple visual example.

DDIM can perform fast sampling because it does not rely on small β_t steps (i.e., the amount of noise removed at each step).

Table 2.1. CIFAR10 and CelebA image generation measured in FID. $\eta = 1.0$ and $\hat{\sigma}$ are cases of DDPM (although [7] Ho et al. 2020 only considered $T = 1000$ steps, and $S < T$ can be seen as simulating DDPMs trained with S steps), and $\eta = 0.0$ indicates DDIM. Table courtesy of [22] Song et al. 2022.

	CIFAR10 (32 × 32)					CelebA (64 × 64)				
η/S	10	20	50	100	1000	10	20	50	100	1000
0.0	**13.36**	**6.84**	**4.67**	**4.16**	4.04	**17.33**	**13.73**	**9.17**	**6.53**	3.51
0.2	14.04	7.11	4.77	4.25	4.09	17.66	14.11	9.51	6.79	3.64
0.5	16.66	8.35	5.25	4.46	4.29	19.86	16.06	11.01	8.09	4.28
1.0	41.07	18.36	8.01	5.78	4.73	33.12	26.03	18.48	13.93	5.98
$\hat{\sigma}$	367.43	133.37	32.72	9.99	**3.17**	299.71	183.83	71.71	45.20	**3.26**

2.4 Adversarially trained Classifiers

Training Classifier Neural Network F_ϕ on Dataset X can be defined as

$$\min_{\phi} \mathbb{E}_{(\mathbf{x},y)\sim X}\Big[\mathcal{L}\left(F_\phi(\mathbf{x}), y\right)\Big] \tag{2.26}$$

where $\mathcal{L}$ is a loss function (e.g. cross-entropy loss).

Models trained as in Equation 2.26 are subject to adversarial attacks ([23] Madry et al. 2019, [4] Szegedy et al. 2014), where adversarial examples are misclassified by F_ϕ. Let's define such data.

Given $(\mathbf{x}, y) \in X$, we say $\mathbf{x}^*$ is an adversarial example if $F_\phi(\mathbf{x}^*) \neq y$ and $\mathbf{x}^*$ is perceptually similar to $\mathbf{x}$ to a human observer. Therefore, F_ϕ is an ε-robust classifier if for any $(\mathbf{x}, y) \sim \mathcal{D}$, where $\mathcal{D}$ is the underlying distribution of data,

$$y = F_\phi(\mathbf{x}) = F_\phi(\mathbf{x} + \delta) \quad \text{for any} \quad \|\delta\| \leq \varepsilon \tag{2.27}$$

To address this issue, [23] Madry et al. 2019 updated Equation 2.26, defining a new training problem for **Robust Classifiers**

$$\min_{\phi} \mathbb{E}_{(x,y)\sim X}\left[\max_{\delta\in S} \mathcal{L}\left(F_\phi(\mathbf{x}+\delta), y\right)\right] \tag{2.28}$$

where $S = \{\delta : \|\delta\|_p \leq \varepsilon\}$ is dependent on a fixed L_p norm.

In practice, Eq. 2.28 represents a Minmax optimization problem, learning parameters ϕ such that the classifier performs well on both clean and adversarial data up to a certain extent. By increasing the robustness radius ε, the model slightly sacrifices its accuracy on clean data in exchange for a substantial increase in accuracy on perturbed data [24] Tsipras et al. 2019.

Minmax optimization problems are often difficult to solve or even intractable, but [23] Madry et al. 2019 provides an effective approximation that makes it tractable.

Fast Gradient Sign Method (**FGSM**), defined by [25] Goodfellow et al. 2015, is an algorithm for L_∞-norm adversarial attacks and is described by the following formula:

$$\mathbf{x}^* = \mathbf{x} + \epsilon \cdot \operatorname{sign}\Big(\nabla_x \mathcal{L}(F_\phi(\mathbf{x}), y)\Big) \tag{2.29}$$

where sign is a function that returns the sign of the entries, encoding the direction of the gradient. Note that the gradient is computed with respect to $\mathbf{x}$. In this equation, $\mathbf{x}$ represents the original input, y is the true label, ϵ is the perturbation magnitude, and $\mathcal{L}$ is the loss function used to train the classifier F_ϕ.

FGSM works by perturbing the input data in the direction of the gradient of the loss function. This perturbation is scaled by ϵ, which controls the strength of the attack. The goal is to find the smallest possible perturbation that can change the classifier's prediction.

Projected Gradient Descent (**PGD**) can be seen as an iterative generalization of the FGSM algorithm [23] Madry et al. 2019. It is used to generate adversarial examples by iteratively perturbing the input. The iterative procedure is shown below:

$$\mathbf{x}_i^* = \text{clip}_\epsilon\Big(\mathbf{x}_{i-1}^* + \alpha \cdot \text{sign}\left(\nabla_{\mathbf{x}_{i-1}^*}\mathcal{L}(F_\phi(\mathbf{x}_{i-1}^*), y)\right)\Big) \tag{2.30}$$

Here, clip_ϵ denotes the function that projects its argument to the surface of $\mathbf{x}$'s ϵ-ball, and α is the step size. This algorithm is able to produce strong adversarial data that have a high probability of fooling a model; for this reason, it is usually used as a benchmark for the robustness of neural networks.

PGD iteratively refines the adversarial example by applying small perturbations, ensuring that each step remains within the ϵ-ball around the original input. This process continues until the perturbation successfully alters the classifier's prediction or reaches a predefined number of iterations. This method is known for generating some of the most challenging adversarial examples, making it a crucial tool for evaluating and enhancing the robustness of classifiers.

Both of these attacks can be targeted or untargeted. In targeted attacks, we choose the class we want the model to classify the image as. In untargeted attacks, the model will likely misclassify the image into the closest class in terms of the distance between decision boundaries in high-dimensional space.

Recent studies have highlighted a significant connection between robustness to Gaussian noise and adversarial robustness. The method of randomized smoothing, as detailed in [26] Cohen et al. 2019, provides a framework for transforming any classifier into one that is certifiably robust against adversarial perturbations through the application of Gaussian noise. Their method is mathematically backed by proven theorems. Specifically, Theorem 1 in their work states that for a randomized smoothed classifier g, there exists a certified radius R within which the classifier's prediction is guaranteed to be stable,

$$R = \frac{\sigma}{2}\left(\Phi^{-1}(\underline{p_A}) - \Phi^{-1}(\overline{p_B})\right) \tag{2.31}$$

where σ is the standard deviation of the Gaussian noise, Φ^{-1} denotes the inverse CDF of the standard normal distribution, and $\underline{p_A}$ and $\overline{p_B}$ are the lower and upper bound probabilities assigned by the base (non-smoothed) classifier f to the top two classes (confidence-wise), respectively. This suggests that robustness to Gaussian noise and adversarial robustness are intrinsically linked, providing a promising direction for deep learning models that leverage this strong connection.

2.5 Classifiers as Generative Energy Based Models

In general, image classification problems involving K classes are handled by training a model $F_\phi(\mathbf{x}) : \mathbb{R}^{H\times W\times C} \to \mathbb{R}^K$, giving in output a vector of size k. This vector is often passed through the Softmax function in order to have a proper probability distribution. This process can be formalized as

$$p_\phi(y \mid \mathbf{x}) = \frac{\exp(F_\phi(\mathbf{x})[y])}{\sum_{k=1}^{K} \exp(F_\phi(\mathbf{x})[k])} \tag{2.32}$$

[27] Grathwohl et al. 2020 showed that a generative energy-based model is hidden inside a standard classifier, meaning that we can re-interpret the logits of F_ϕ to compute $p_\phi(\mathbf{x}, y)$ and $p_\phi(\mathbf{x})$.

We know that for any generative model,

$$p_\phi(\mathbf{x}) = \sum_y p_\phi(\mathbf{x}, y) = \frac{\sum_k \exp(F_\phi(\mathbf{x})[y])}{Z(\phi)}. \tag{2.33}$$

By marginalizing any $y \in K$,

$$p_\phi(\mathbf{x}, y) = \frac{\exp(F_\phi(\mathbf{x})[y])}{Z(\phi)} \tag{2.34}$$

By definition of conditional probability,

$$p_\phi(y \mid \mathbf{x}) = \frac{p_\phi(\mathbf{x}, y)}{p_\phi(\mathbf{x})} = \frac{\exp(F_\phi(\mathbf{x})[y])Z(\phi)}{\sum_k \exp(F_\phi(\mathbf{x})[y])Z(\phi)} \tag{2.35}$$

the intractable normalizing constants cancel out;
and finally, by going into log space we define the classifier score as a combination of energy models

$$\log p_\phi(y \mid \mathbf{x}) = \log p_\phi(\mathbf{x}, y) - \log p_\phi(\mathbf{x}) \tag{2.36}$$

$$= F_\phi(\mathbf{x})[y] - \text{LogSumExp}_y(F_\phi(\mathbf{x})[y]) \tag{2.37}$$

$$= E_\phi(\mathbf{x}) - E_\phi(\mathbf{x}, y) \tag{2.38}$$

where

$$E_\phi(\mathbf{x}) = -\log p_\phi(\mathbf{x})$$

$$E_\phi(\mathbf{x}, y) = -\log p_\phi(\mathbf{x}, y)$$

Thus, [27] Grathwohl et al. 2020 have found a generative model hidden within every standard discriminative model!

Chapter 3

Related work

3.1 Robustness is at odds with Generation

Figure 3.1. adversarial examples from smoothed DNNs at different smoothing scales σ. image taken from [28] Kaur et al. 2019

In recent years, a strong connection between Adversarially Trained (AT) models and generative modeling has been discovered. The findings of [28] Kaur et al. 2019 strongly support the hypothesis that Perceptually Aligned Gradients (PAG) are a general property of robust classifiers. Specifically, given any natural image, if we iteratively take gradient steps that maximize the score of any fixed targeted class y of a robust classifier, the result is perceptually aligned with human perception. In contrast, performing the same experiment with a vanilla-trained classifier ($\sigma = 0$) does not yield perceptually aligned results.

This property highlights the ability of robust classifiers to capture and reflect significant generative features in their gradient space, thereby bridging the gap between adversarial robustness and generative capabilities.

3.2 Image Synthesis with a Single Robust Classifier

[29] Santurkar et al. 2019 have shown that the features learned by a single, off-the-shelf, adversarially trained classifier are sufficient for a wide range of image synthesis tasks, such as those shown in Figure 3.2.

These results alone prove that robust classifiers have incredible generative capabilities, and their findings indicate that adversarial robustness may have benefits beyond security and reliability. We believe they have paved the way for developing better generative models thanks to the properties of robust DNNs.

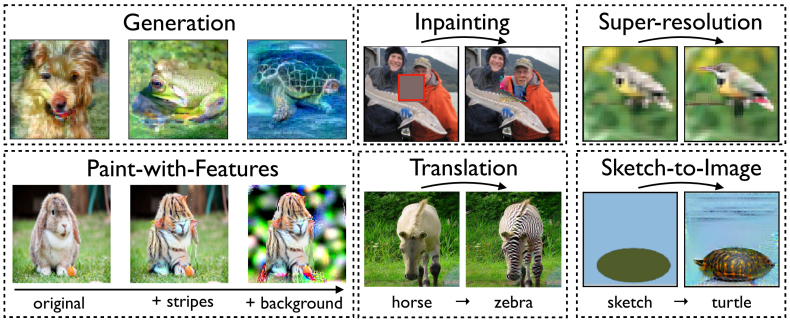


Figure 3.2. Image synthesis and manipulation tasks performed using a single (robustly trained) classifier. image taken from [29] Santurkar et al. 2019

3.3 Robust Classifier Guidance

Class conditional diffusion synthesis can be achieved in a classifier-free manner by conditioning a Denoising Diffusion Probabilistic Model (DDPM) on the class, effectively training a denoising model $\epsilon_\theta(\mathbf{x}, t, y)$. Another approach is to explicitly train a tailored time-dependent classifier $p_\phi(y \mid \mathbf{x}_t, t)$ to incorporate its gradients in the reverse diffusion process, thereby "encouraging" the image to be classified as a fixed y. This constraints the final output to be sampled from the class-conditional input space distribution. This process can be summarized by Equation 2.17.

[12] Kawar et al. 2023 has developed a method to train a novel AT classifier through PGD attacks to perform robust-classifier guidance since, as the paper states:

> "*...an off-the-shelf adversarially trained robust classifier would not fit our purpose in this context. This is due to the fact that in the diffusion process, the classifier operates on intermediate images x_t, which are a linear mixture of an ideal image and Gaussian noise. Furthermore, this mixture is also a function of t, which requires the classifier model to be time-dependent.*"

This thesis is based on the belief that this statement might not be completely true, and on the belief that an off-the-shelf AT robust classifier can be used for successfully performing guidance in Diffusion Models.

The guidance is performed by using the gradients of a model $h_\phi(\mathbf{x}_t, t) = \{\log p_\phi(j \mid \mathbf{x}_t, t)\}_{j=1}^{C}$, where p_ϕ is trained as follows: pick a noisy image $\mathbf{x}_t$, and perform a PGD adversarial attack on it by applying early stopping, halting the attack as soon as the classifier misses. This approach allows their model to be robust against both Gaussian noise (already achieved) and adversarial noise (novel).

Chapter 4

Experiments and Results

4.1 Introduction to the Main Idea

The main idea behind our research lies in strong hypotheses from the existing literature about the connection between adversarially trained robust classifiers, generative models, and energy-based models. Current methods of Classifier-free Guidance leverage Bayes' theorem (Equation 4.1) to rewrite the conditional probability distribution of the input space as a combination of generative scores, all coming from the generative model described by θ. By doing so, we can train a conditional DM $p_\theta(\mathbf{x} \mid y)$ without needing to train a separate time-dependent classifier.
Let $p_\theta(\mathbf{x})$ be an unconditional DM, and $p_\phi(y \mid \mathbf{x})$ a classifier trained on the same dataset as p_θ

$$\nabla_x \log p_\theta(\mathbf{x} \mid y) = \nabla_x \log p_\theta(\mathbf{x}) + s\nabla_x \log_\phi(y \mid \mathbf{x})$$

$$= \nabla_x \log p_\theta(\mathbf{x}) + s\left[\nabla_x \log p_\theta(\mathbf{x}, y) - \nabla_x \log(p_\theta(\mathbf{x}))\right]$$

$$= (1-s)\nabla_x \log p_\theta(\mathbf{x}) + s\nabla_x \log p_\theta(\mathbf{x}, y) \tag{4.1}$$

We believe that conditional sampling can be achieved by combining an unconditional Diffusion Model, specifically DDPM or DDIM, with an **off-the-shelf** robust classifier. Current literature strongly suggests that robust classifiers possess generative properties, and their gradients are highly informative about the input space. Therefore, these gradients can effectively guide the diffusion model towards the desired class. We can write a similar equation to 4.1 by considering the classifier:

$$\nabla_x \log p_\theta(\mathbf{x} \mid y) = \nabla_x \log p_\theta(\mathbf{x}) + s\nabla_x \log p_\phi(y \mid \mathbf{x})$$

$$= \nabla_x \log p_\theta(\mathbf{x}) + s\left[\nabla_x \log p_\phi(\mathbf{x}, y) - \nabla_x \log p_\phi(\mathbf{x})\right]$$

$$= \nabla_x \log p_\theta(\mathbf{x}) - s\nabla_x \log p_\phi(\mathbf{x}) + s\nabla_x \log p_\phi(\mathbf{x}, y) \tag{4.2}$$

Furthermore, we observe that we can turn the classifier score into Energy-Based Models (EBMs) as described in Section 2.5.

$$= \nabla_x p_\theta(\mathbf{x}) + s[\nabla_x E_\phi(\mathbf{x}) - \nabla_x E_\phi(\mathbf{x}, y)] \tag{4.3}$$

Algorithm 5 shows the steps to follow in order to unconditionally sample images by having a trained diffusion denoiser ϵ. Algorithm 6 instead shows the classifier guidance version of it by considering the classifier score $\nabla_x \log p_\phi(y \mid \mathbf{x}_t)$. We will make use of these implementations for our experiments.

Algorithm 5: 100 timesteps DDIM Sampling

Input : Brownian motion η

$\tau = [\tau_0 = 0, 10, 20, 30, ..., 1000 = \tau_S = T]$;

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;

for all *s from S to* 1 **do**

  $t \leftarrow \tau_s$;

  $t' \leftarrow \tau_{s-1}$;

  $\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;

  $\hat{\epsilon} \leftarrow \epsilon_\theta(\mathbf{x}_t, t)$;

  $\hat{\mathbf{x}}_0 \leftarrow \frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}}$; `// Estimate` $\mathbf{x}_0$

  $\mu \leftarrow \sqrt{\bar{\alpha}_{t'}}\hat{\mathbf{x}}_0 + \sqrt{1-\bar{\alpha}_{t'}}\hat{\epsilon}$;

  $\mathbf{x}_{t'} \leftarrow \mu + \sigma_t(\eta)\varepsilon$;

return $\mathbf{x}_0$

Algorithm 6: 100 timesteps Classifier Guided DDIM Sampling

Given : Classifier $p_\phi(y \mid \mathbf{x}_t)$

Input : class label y, guidance strength s, Brownian motion scale η

$\tau = [\tau_0 = 0, 10, 20, 30, ..., 1000 = \tau_s = T]$;

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;

for all *s from S to* 1 **do**

  $t \leftarrow \tau_s$;

  $t' \leftarrow \tau_{s-1}$;

  $\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;

  $\hat{\epsilon} \leftarrow \epsilon_\theta(\mathbf{x}_t, t) - \sqrt{1-\bar{\alpha}_t} s \nabla_{x_t} \log p_\phi(y \mid \mathbf{x}_t)$; `// gradient step`

  $\hat{\mathbf{x}}_0 \leftarrow \frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}}$;

  $\mu \leftarrow \sqrt{\bar{\alpha}_{t'}}\hat{\mathbf{x}}_0 + \sqrt{1-\bar{\alpha}_{t'}}\hat{\epsilon}$;

  $\mathbf{x}_{t'} \leftarrow \mu + \sigma_t(\eta)\varepsilon$;

return $\mathbf{x}_0$

Figure 4.1. Guided and Unguided version of the implementation of DDIM-sampling

4.2 Model Selection and Evaluation

We have decided to use the **diffusers**[1] library available on HuggingFace. Specifically, we utilize their pre-trained models of DDPM and DDIM, as well as a PyTorch implementation[2] of [7] Ho et al. 2020 by Google. For the pre-trained robust classifiers, we chose to use the **robustness**[3] package by [30] Engstrom et al. 2019. The classifiers are trained with Standard Adversarial Training (SAT) PGD on ℓ_2 norm with ε-constraint = [0/255, 0.25/255, 0.5/255, 1.0/255] where $\varepsilon = 0/255$ trains a non-robust classifier. From now on, we will omit the denominator by just writing ε = [0, 0.25, 0.5, 1.0]. All experiments are made on a single NVIDIA GTX 1060 with 6GB of VRAM.

We re-evaluated the classifiers to make sure they perform as expected; table 4.1 shows the test-accuracies on 20-steps PGD-attacks with step size = $2.5 * \frac{\varepsilon_{test}}{20}$

ε-test \ ε-train	0.0	0.25	0.5	1.0
0.0	**95.25%**	92.77%	90.83%	81.62%
0.25	8.65%	81.21%	**82.40%**	75.53%
0.5	0.28%	62.29%	**70.17%**	68.63%
1.0	0.00%	21.18%	40.48%	**52.72%**
2.0	0.00%	0.53%	5.22%	18.59%

Table 4.1. CIFAR10 L2-norm (ResNet50) 20-steps pgd

Our approach to conditional sampling is highly inspired by [8] Dhariwal and Nichol 2021's official codebase[4], and we directly adopt their diffusion models' metrics evaluation procedure.

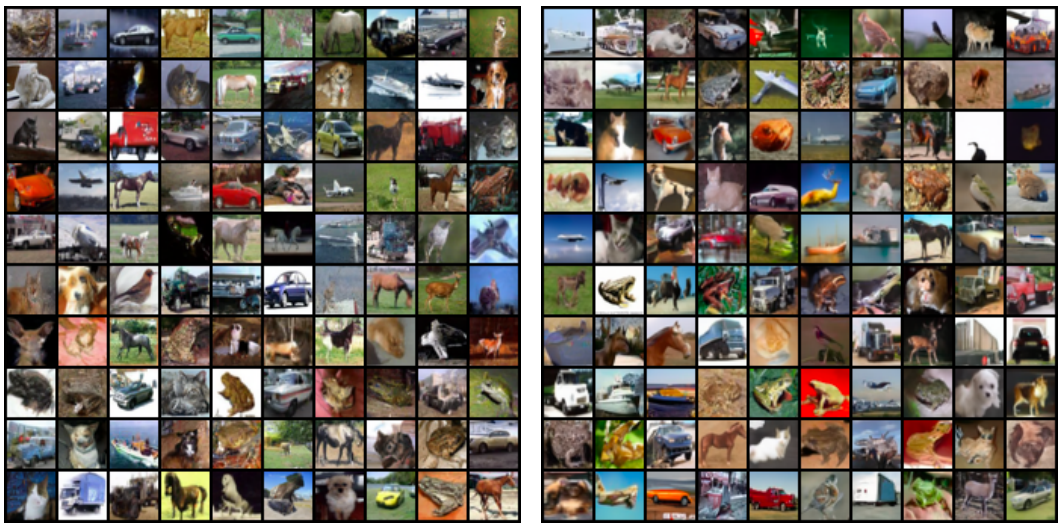


Figure 4.2. Unconditionally generated images from DDPM (left) and DDIM (right) sampling methods

We work with the CIFAR-10[5] dataset, a well-known benchmark in the machine

[1] https://huggingface.co/docs/diffusers
[2] https://huggingface.co/google/ddpm-cifar10-32
[3] https://github.com/MadryLab/robustness
[4] https://github.com/openai/guided-diffusion
[5] https://www.cs.toronto.edu/~kriz/cifar.html

learning community. It was chosen due to its manageable size and the availability of extensive benchmarking results.

After having checked the classifiers' performance, we evaluate the pre-trained Diffusion model's metrics, thus setting the baseline for conditional generation. Here is a brief high-level description of the metrics involved:

- **Inception Score (IS)** measures the quality and diversity of generated images. Higher is better.
- **Fréchet Inception Distance (FID)** evaluates the distribution divergence between the generated images and real ones. Lower is better.
- **Precision** assesses the fraction of "realistic" generated images.
- **Recall** measures the diversity of generated images by evaluating how many real images are covered by the generated ones.

	IS ↑	FID ↓	Precision ↑	Recall ↑
DDPM	8.68	16.49	0.68	0.59
DDIM	8.40	14.79	0.64	0.59

Table 4.2. Evaluation metrics for unconditional guidance using DDPM and DDIM on CIFAR-10 test set, with 1000 and 100 inference steps respectively

The evaluation is performed on 10,000 images sampled from each method against 10,000 different images from the CIFAR-10 test set.

Throughout the research, we have mostly used DDIM sampling since the pre-trained model we downloaded was just as good as the DDPM version, but the inference speed was approximately 20x faster.

4.3 Gradient Accumulation

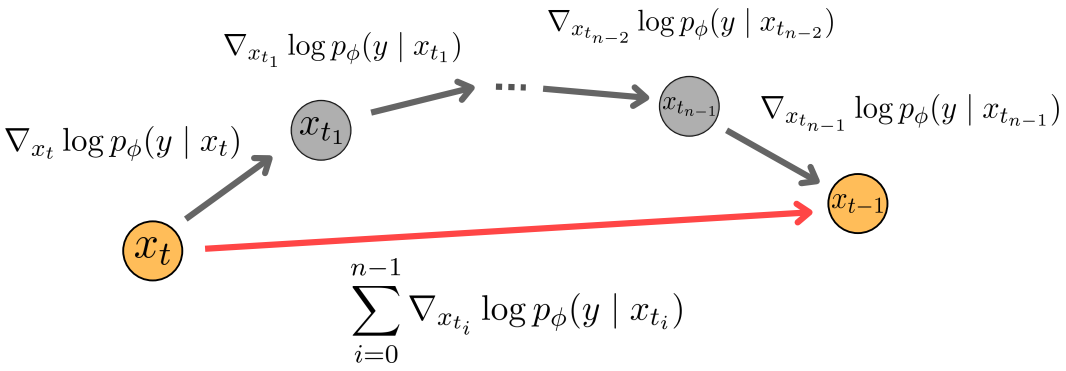


Figure 4.3. High-level illustration of the gradient cumulation process.

In the beginning, we suspected that in low-density regions (i.e., when the image is close to being pure noise), the classifier score isn't accurate, and thus, it is not contributing well to the model, as the model's accuracy on such data is close to random guessing. So we have decided to "activate" the guidance procedure only after a certain amount of timesteps (e.g. 30%) by first letting the unconditional

DM denoise the image and making the data point flow to a higher density region. Together with this, we have cumulated the gradient, meaning that at each timestep t, instead of using $\nabla_{x_t} \log p_\phi(y \mid x_t)$, we iteratively sum n times the gradients of intermediate images to (hopefully) generate images from the wanted class. Figure 4.3 shows a high-level illustration of the process. In essence, line 7 in Algo 6 becomes:

$$\hat{\epsilon} \leftarrow \epsilon_\theta(\mathbf{x}_t, t) - \sqrt{1-\bar{\alpha}_t}\, s \sum_{i=0}^{n-1} \nabla_{x_{t_i}} \log p_\phi(y \mid \mathbf{x}_{t_i})$$

where $x_{t_0} = x_t$

By employing the above-mentioned approach, we achieved highly saturated images that, while not entirely natural-looking, were clearly guided towards the desired class. Figure 4.4 illustrates samples guided towards the "car" class, with the guidance activating after 20% of the steps.

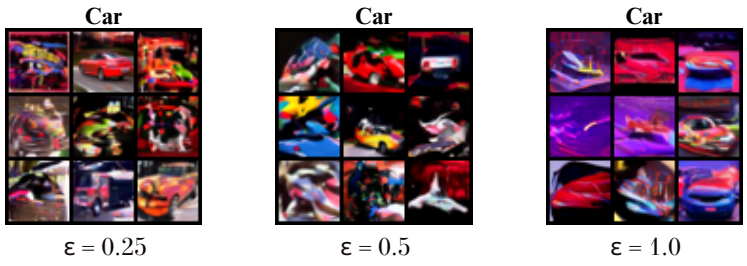


Figure 4.4. "cars" sampled by using 3 different classifiers, from less robust (left) to most robust (right) with guidance scale $s = 10$ and $\eta = 1$. Guidance activation after 20% of steps

4.4 Trade-offs Between Robustness and Natural Accuracy

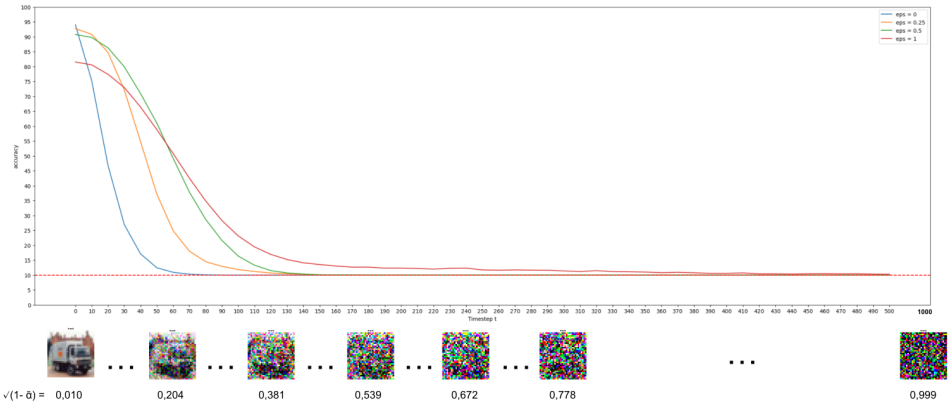


Figure 4.5. Accuracy of AT classifiers at multiple levels of noise. The x-axis shows examples of how the images appear at each respective timestep. The plot is cut at timestep 500 since the accuracies drop to 10% (i.e. random guessing). The evaluation was performed on the CIFAR-10 test set at intervals of 10 steps.

Before continuing with the experiments, we aimed to analyze the accuracy of the classifiers on images at various noise levels. To accomplish this, we con-

ducted an accuracy evaluation on the entire CIFAR-10 test set at timesteps $t = [0, 10, 20, 30, \ldots, 1000]$ for the non-robust model ($\varepsilon = 0$) and for the robust models ($\varepsilon = [0.25, 0.5, 1.0]$). Figure 4.5 shows the accuracy of AT classifiers at multiple levels of noise. Notice that after approximately 150 timesteps in the forward process, the average accuracies of all models drop to the baseline value. For the CIFAR-10 dataset, that is equal to 10%, which is the random guessing accuracy rate. The latter is expected as the images become too noisy to be recognizable. An exception to this pattern is the most robust model ($\varepsilon = 1.0$), which can still retrieve some information about the ground truth class from images that appear as complete noise to humans. The most explicit pattern of the plot is that the robustness of the model is correlated with its natural data accuracy, confirming once again the findings of [24] Tsipras et al. 2019

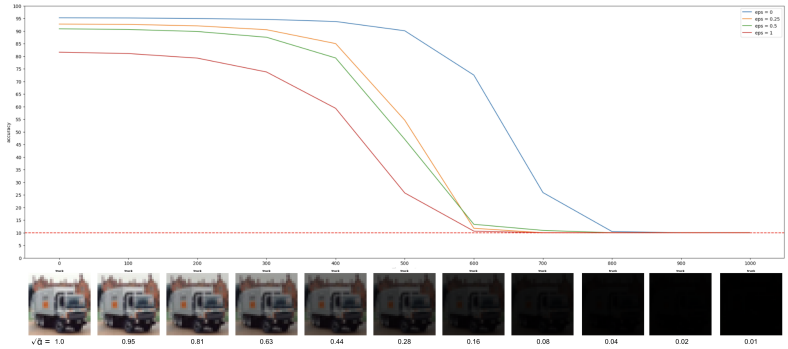


Figure 4.6. Accuracy of AT classifiers at different darkness levels, scheduled by $\sqrt{\bar{\alpha}_t}$, which represents the weight given to the natural image at timestep t in Eq. 2.23. The x-axis shows examples of how the images appear at each respective timestep. The evaluation was performed on the CIFAR-10 test set at intervals of 100 steps.

While conducting various experiments, we came up with an unexpected finding; we evaluated the classifier accuracy on CIFAR-10 test images by applying the "darkness" signal applied to x_0 in the forward diffusion process to x_t, that is $\sqrt{\bar{\alpha}_t}$. The radius ε of the robustness of the models is inversely proportional to the mean accuracy at each timestep. This is very strange since you would normally think that the more robust models perform better in very dark images, but we found that it is completely the opposite: less robust models are more robust to darkness.

4.5 Tuning the Guidance strength

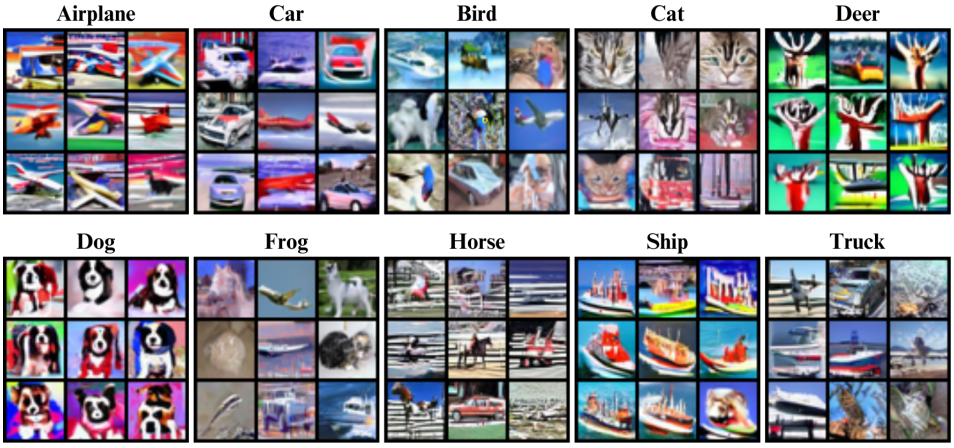


Figure 4.7. Class Conditional sampling by using $\varepsilon = 1.0$ classifier , $\eta = 0.5$, $s = 30$ for first 50% of timesteps, $s = 11$ for last 50% of timesteps

We empirically find that our previous hypothesis about guidance is most likely incorrect, indicating that it is essential to provide strong guidance when the data point is in low-density regions of the input space. So, we have dropped the idea of turning off the guidance at the beginning of the sampling process, and we have tried to start with a relatively high value for the gradient strength scalar s and drop it to a much lower value after a certain timestep. Furthermore, we stop using the gradient accumulation method since we found that the results are highly correlated with images sampled by only adjusting the guidance strength s. Figure 4.7 shows some examples of class conditional sampling using the above-mentioned approach. We can clearly see some improvements since the images appear to have fewer hallucinations and more natural-looking images. When the model fails to generate samples from the desired label, it usually produces images resembling some other class instead of nonsense. A common drawback that we find when using a strong guidance scale is getting highly saturated images and thus not conforming to reality.

If the guidance strength is too strong, the classifier score will drive the noise towards the class mode, overfitting the classifier's knowledge of the input space. Figure 4.8 shows a 100-step DDIM generation process of a truck, starting from top left (x_T) to bottom right (x_0). It is clear that only after ∼10 timesteps the high frequencies of a truck start to appear. It suggests that hyperparameter tuning is essential for successful conditional sampling. If the scale is too small, it fails to guide towards the intended class, thus generating unconditionally. If the guidance strength is too large, it overfits the class mode.

Figure 4.8. high guidance scale, 100-step DDIM conditional generation process of a truck.

4.6 Class-conditional Energy is more informative

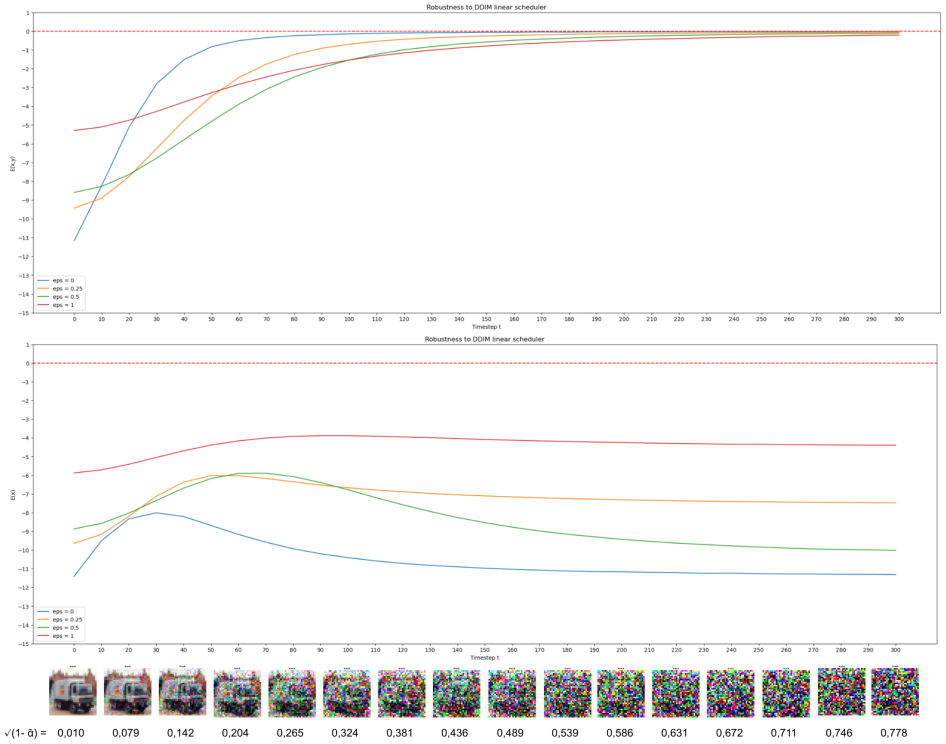


Figure 4.9. average E(x,y) and E(x) at multiple levels of noise. The plot is cut at timestep 300 since they converge. The evaluation was performed on the CIFAR-10 test set at intervals of 10 steps.

In Section 2.5, Equation 2.38, we learned that given a data point $\mathbf{x}$, we can write $\nabla_x \log p_\phi(y \mid \mathbf{x}) = \nabla_x E_\phi(\mathbf{x}, y) - \nabla_x E_\phi(\mathbf{x})$. We have empirically found that by dropping $E_\phi(\mathbf{x})$ and guiding only with $E_\phi(\mathbf{x}, y)$, the quality of samples improves. Intuitively, we can interpret $E_\phi(\mathbf{x}, y)$ as the direction pointing towards the class-conditional density, while $E_\phi(\mathbf{x})$ acts as an in-distribution measure, since in general, it is inversely proportional to $p_\phi(\mathbf{x})$ for any given $\mathbf{x}$. Since we are generating mainly from a trained Diffusion Model, we believe that this value is interfering with the DM representation of the input space encoded in the parameters of the Denoising UNet $\epsilon_\theta(\mathbf{x}_t, t)$

4.7 From Large Leaps to Incremental Steps

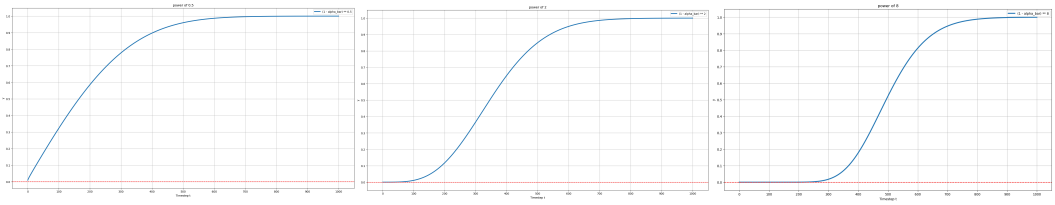


Figure 4.10. Plots three different schedulers: the standard $(1-\bar{\alpha}_t)^{0.5}$ (left), $(1-\bar{\alpha}_t)^2$ (center), and $(1-\bar{\alpha}_t)^8$ (right). Each plot should be read from right to left as it illustrates the strength of guidance during the reverse diffusion process.

In section 4.5, we tried to reduce the guidance scale factor abruptly. In this section, instead, we begin to smoothly control the guidance scale by distinguishing the classifier score scheduler from the denoising scheduler.

The main idea is to modify the gradient strength scheduler so that the guidance is strong at the beginning and gradually decreases towards the end. Normally, the scheduler would be $\sqrt{1-\bar{\alpha}_t}$. To achieve the desired effect, we can raise $(1-\bar{\alpha}_t)$ to a different power, making the schedule resemble a sigmoid curve. The higher the exponent, the more pronounced the effect. Figure 4.10 shows an example of how the scheduler changes as we change the power coefficient.

To provide a flexible experiment setting, we introduce a new sigmoid-based scheduling function (4.4). This function allows us to easily modify the shape of the sigmoid curve by adjusting three key parameters: y_0, t_0, and k. The parameter y_0 determines the horizontal asymptote, effectively setting the starting value of the scheduler. The parameter t_0 specifies the midpoint of the sigmoid, allowing us to shift the curve along timesteps. Lastly, k controls the steepness of the curve, allowing us to set how quickly the guidance changes from strong to weak. By tuning these parameters, we can easily explore the impact of different schedulers on the classifier guidance.

$$f(t) = (1-y_0) \cdot \frac{1}{1+e^{-k(t-t_0)}} + y_0 \tag{4.4}$$

The experiments led to surprisingly good-looking images; in particular, we can notice less saturation, so the images look more natural. Figure 4.11 shows an example by using a custom scheduler.

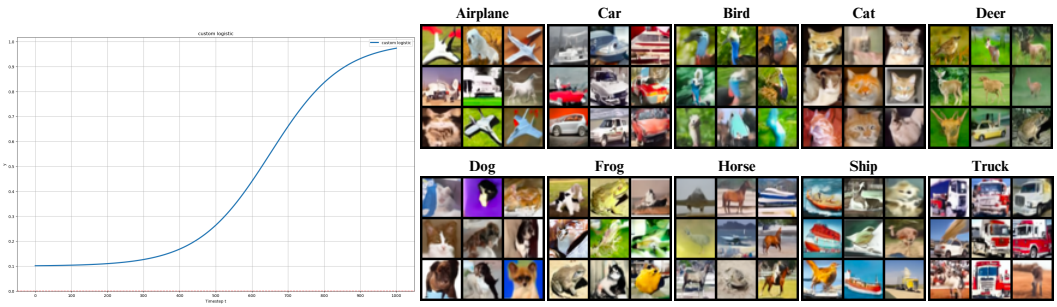


Figure 4.11. Guidance scheduler with parameters $t_0 = 650,\ y_0 = 0.1,\ k = 0.01$ (left) and images sampled with the same scheduler guided by using $\varepsilon = 1.0$ classifier, $\eta = 1$, $s = 12$

4.8 Dual Optimization of E(x,y) and ΔE(x)

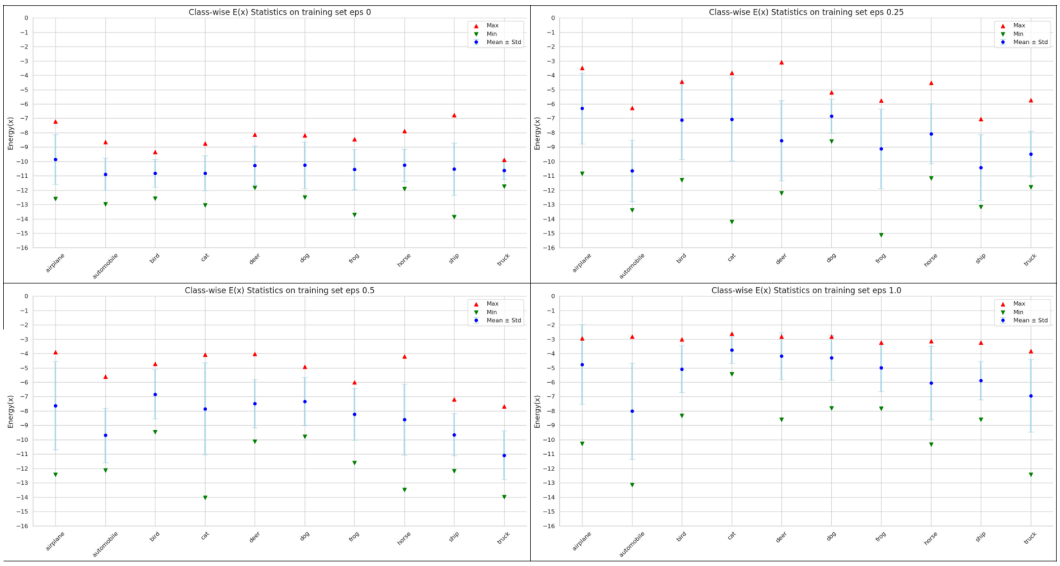


Figure 4.12. Average classifiers' E(x) statistics divided by class

In this section, we present a new objective function that brings back $E_\phi(\mathbf{x})$. Figure 4.12 presents the class-wise statistics of $E_\phi(\mathbf{x})$ from all our classifiers evaluated on the CIFAR-10 training set. It is evident that the models trained on larger ε-radius have greater average energy compared to lower ε-radius models. Notice also that both intra-class and inter-class variations follow the same reasoning. This behaviour is expected, as robust models tend to lose the correct perception of natural data while gaining awareness of adversarial data. A similar behaviour is found by analyzing $E_\phi(\mathbf{x}, y)$ of the training set.

Throughout the experiments, we observed that the average $E_\phi(\mathbf{x})$ of the generated images was significantly lower compared to the energy of the natural images from the CIFAR-10 training set. Intuitively, this could mean that the classifier believes that the generated images are more likely to belong to the training set than the training set images themselves.
To address this issue, we define the energy divergence:

$$\Delta E_\phi(\mathbf{x}) = |E_\phi(\mathbf{x}^*)[y] - E_\phi(\mathbf{x})| \tag{4.5}$$

where $E_\phi(\mathbf{x}^*)[y]$ is the fixed, class-wise average energy of the classifier described by ϕ, computed on the training set images having label y (i.e. the blue dots in Figure 4.12). So, the new objective function is defined as:

$$\mathcal{L} := E_\phi(\mathbf{x}, y) + \lambda \Delta E_\phi(\mathbf{x}) \tag{4.6}$$

where λ is a weighting constant.

By replacing the classifier's score with $\nabla_{x_t}\mathcal{L}$, we are able to conditionally sample images having a lower $\Delta E_\phi(\mathbf{x})$ on average.

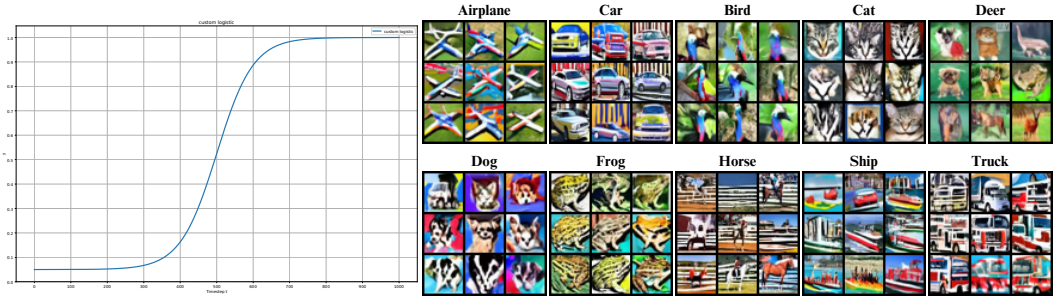


Figure 4.13. Guidance scheduler with parameters $t_0 = 500$, $y_0 = 0.05$, $k = 0.02$ (left) and images sampled with the same scheduler guided by using $\varepsilon = 1.0$ classifier, $\eta = 0.8$, $s = 4$. $E_\phi(\mathbf{x}, y)$ score

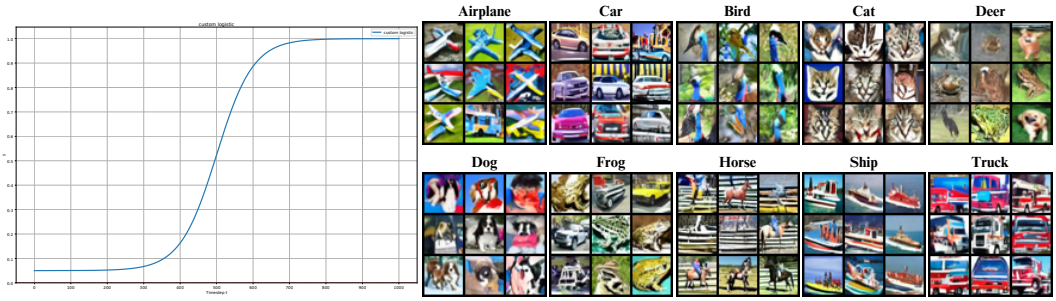


Figure 4.14. Guidance scheduler with parameters $t_0 = 500$, $y_0 = 0.05$, $k = 0.02$ (left) and images sampled with the same scheduler guided by using $\varepsilon = 1.0$ classifier, $\eta = 0.8$, $s = 4$. $\mathcal{L}$ score, $\lambda = 0.15$

The comparison between Figure 4.13 and Figure 4.14 shows the saturation difference between the images sampled with the two different methods. Even if the qualitative perception is very similar, the energy computed on the generated images by guiding with the objective function 4.6 is clearly lower. Figure 4.15 illustrates the comparison between the average energy computed on the above-mentioned images. The red area is an average energy indicator across all classes.

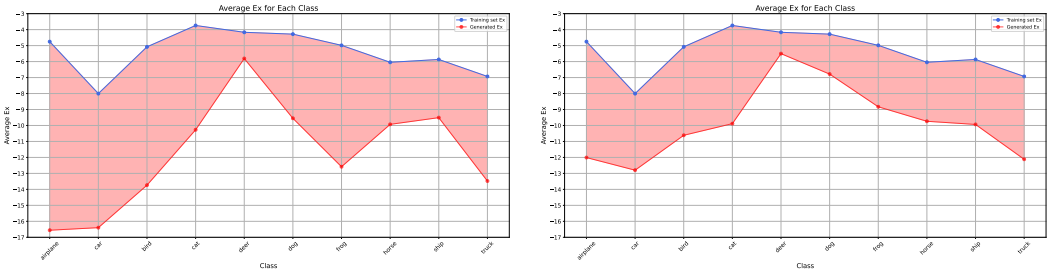


Figure 4.15. Comparison plot between $E_\phi(\mathbf{x}^*)$ (blue line) and $E_\phi(\mathbf{x})$ (red line) of samples in Fig. 4.13 (left) and Fig. 4.14 (right)

4.9 PGD-based Classifier Guidance

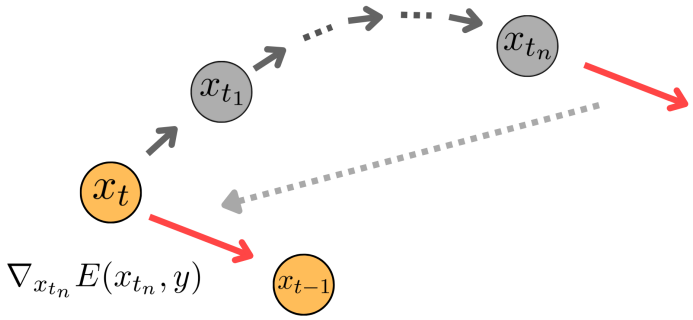


Figure 4.16. High-level illustration of PGD-based guidance process.

In this Section, we present a novel method to approach classifier guidance. During each iteration of the reverse denoising diffusion process, we perform a variation of an-steps PGD attack on the image $\mathbf{x}_t$, maximizing the joint energy $E(\mathbf{x}, y)$ where y is a fixed guidance label.

At each step $i \in \{1, \dots, n-1\}$, the intermediate gradient is first normalized to have l_2-norm equal to 1. Then we multiply it by a fixed step size γ and subtract it from $\mathbf{x}_{t_i}$ to get $\mathbf{x}_{t_{i+1}}$. The idea is to use the classifier score on the image $\mathbf{x}_{t_n}$ as guidance for $\mathbf{x}_t$ since it should be better informed about the direction towards the class distribution. Figure 4.16 shows a high-level visual example of the approach; keep in mind that thinking about high-dimensional spaces is tough for our brains because we're so used to thinking in three dimensions. Our intuition, which works well in our 3D world, often lets us down when we try to understand more complex, higher-dimensional concepts. Nevertheless, the illustrations help us explain what we are thinking about.

This procedure can be formalized by applying Algorithm 8 at each step in the reverse diffusion process.

Algorithm 8: PGD-based guidance score

Given : Classifier $p_\phi(y \mid \mathbf{x})$, step size γ, # of attack steps n, timestep t

for $i \leftarrow 0$ ***to*** $n-1$ **do**

  $\mathbf{x}_{t_{i+1}} \leftarrow (1+\gamma)\mathbf{x}_{t_i} - \gamma \frac{\nabla_{\mathbf{x}_{t_i}} E_\phi(\mathbf{x}_{t_i}, y)}{\left\| \nabla_{\mathbf{x}_{t_i}} E_\phi(\mathbf{x}_{t_i}, y) \right\|_2}$;

Score $= \nabla_{\mathbf{x}_{t_n}} \log E_\phi(\mathbf{x}_{t_n}, y)$; `// replaces` $\nabla_{x_t} \log p_\phi(y \mid \mathbf{x}_t)$ `in algo 6`

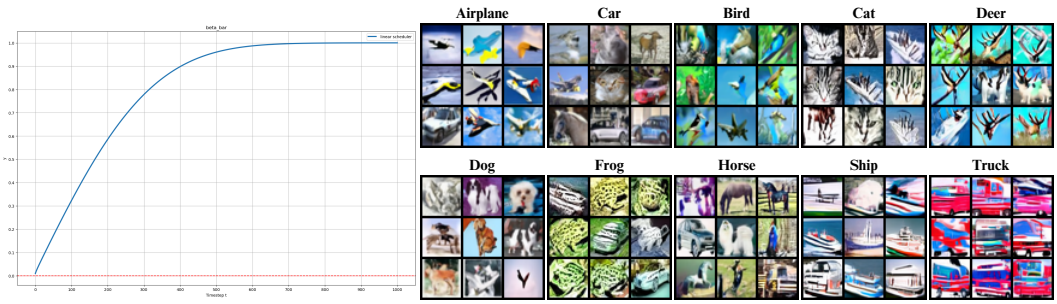


Figure 4.17. Linear guidance scheduler (left) and images sampled with the same scheduler guided by means of 100 steps PGD-based guidance, with $\gamma = 0.01$. Samples are not generated by using common parameters

Figure 4.17 shows some images sampled with this method. Although it doesn't always generate from the desired class, overall, the colours look more natural, and the images are more diverse between them.

We believe that exploring further PGD-based guidance could lead to interesting results. By refining this approach, we anticipate achieving more accurate class conditioning and generating images with greater fidelity and diversity.

Chapter 5

Conclusions

5.1 Summary of Findings

In this thesis, we have explored how to use an off-the-shelf adversarially trained classifier to achieve conditional sampling in unconditional diffusion models. Our research has demonstrated that classifier guidance is indeed possible by using a SAT classifier, showcasing the power of robust models in generative tasks. We have started the research by searching some pre-trained Diffusion Models and diverse AT classifiers.

We have reproduced the DM's intended behaviour and we tested its evaluation metrics as well as the qualitative perception of the unconditionally generated images. Thus, we have settled the baseline to compare our class-conditioned samples.

After that, we analyzed the discriminative models' behaviour against Gaussian noise, typically found in DDPMs and DDIMs, while also highlighting their unexpected vulnerability to darkness. This allowed us to think about new methods to use best their generative features to successfully drive samples towards the desired class. Furthermore, we have turned discriminative models into generative Energy-Based Models (EBMs). We used the energy functions to improve the guidance process thanks to empirical experiments. We have interpreted the meaning of $E_\phi(\mathbf{x})$ and $E_\phi(\mathbf{x}, y)$ in a generative setting, giving us some intuitions about the hidden generative properties of a classifier.

A significant contribution of this work is the statistical analysis of classifier metrics against the entire CIFAR-10 test set across a simulated forward diffusion process, treating classifier logits as probabilities as well as energy functions. This analysis provided a deeper understanding of the connection between accuracy and energy and how they change over the course of a diffusion process. The behaviour of the accuracy was intuitively predicted, but the $E(\mathbf{x})$ function had some unexpected outcomes.

We introduced a novel sigmoid-based function to study the contribution of guidance strength across all timesteps, and we have tried different methods to improve the generation quality while maintaining class conditioning.

5.2 Limitations

Our research was constrained by a few limitations. The limited GPU and computing power available restricted our ability to retrain diffusion models. Thus,

we have worked with a pre-trained diffusion model that was far from state-of-the-art evaluation metrics.

5.3 Future Research

Looking ahead, we plan to expand our research across different datasets such as CelebA and ImageNet. Moreover, we plan to investigate the gradients of robust classifiers and how to enhance their contribution towards diffusion models in order to achieve good metrics in classifier-guided samples.

To provide theoretical insights, we shall mathematically study the connection between the adversarial robustness of a classifier and its accuracy on predicted score functions in low-density regions of the input space.

We believe that these efforts will refine our approach and improve the overall quality and effectiveness of classifier guidance by using an off-the-shelf robust classifier.

Bibliography

[1] Christopher M. Bishop. *Pattern recognition and machine learning*. Information science and statistics. Springer, New York, 2006. ISBN 978-0-387-31073-2.

[2] Keiron O'Shea and Ryan Nash. An Introduction to Convolutional Neural Networks, December 2015. URL http://arxiv.org/abs/1511.08458. arXiv:1511.08458 [cs].

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://papers.nips.cc/paper_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.

[4] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, February 2014. URL http://arxiv.org/abs/1312.6199. arXiv:1312.6199 [cs].

[5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks, June 2014. URL http://arxiv.org/abs/1406.2661. arXiv:1406.2661 [cs, stat].

[6] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics, November 2015. URL http://arxiv.org/abs/1503.03585. arXiv:1503.03585 [cond-mat, q-bio, stat].

[7] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models, December 2020. URL http://arxiv.org/abs/2006.11239. arXiv:2006.11239 [cs, stat].

[8] Prafulla Dhariwal and Alex Nichol. Diffusion Models Beat GANs on Image Synthesis, June 2021. URL http://arxiv.org/abs/2105.05233. arXiv:2105.05233 [cs, stat].

[9] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium, January 2018. URL http://arxiv.org/abs/1706.08500. arXiv:1706.08500 [cs, stat].

[10] Yann Lecun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu-Jie Huang. *A Tutorial on Energy-Based Learning*. January 2006.

[11] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes, December 2022. URL http://arxiv.org/abs/1312.6114. arXiv:1312.6114 [cs, stat].

[12] Bahjat Kawar, Roy Ganz, and Michael Elad. Enhancing Diffusion-Based Image Synthesis with Robust Classifier Guidance, March 2023. URL http://arxiv.org/abs/2208.08664. arXiv:2208.08664 [cs].

[13] C. Jarzynski. Equilibrium free energy differences from nonequilibrium measurements: a master equation approach. *Physical Review E*, 56(5):5018–5035, November 1997. ISSN 1063-651X, 1095-3787. doi: 10.1103/PhysRevE.56.5018. URL http://arxiv.org/abs/cond-mat/9707325. arXiv:cond-mat/9707325.

[14] Paul Langevin. Sur la théorie du mouvement Brownien, C. R. Acad. Sci., 146, 530. 1908. URL https://rcin.org.pl/dlibra/doccontent?id=194893. Publisher: CNRS.

[15] W. Feller. On the Theory of Stochastic Processes, with Particular Reference to Applications. In *Proceedings of the [First] Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 403–433. University of California Press, January 1949.

[16] Yang Song and Stefano Ermon. Generative Modeling by Estimating Gradients of the Data Distribution, October 2020. URL http://arxiv.org/abs/1907.05600. arXiv:1907.05600 [cs, stat].

[17] Aapo Hyvärinen. Estimation of Non-Normalized Statistical Models by Score Matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005. ISSN 1533-7928. URL http://jmlr.org/papers/v6/hyvarinen05a.html.

[18] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced Score Matching: A Scalable Approach to Density and Score Estimation, June 2019. URL http://arxiv.org/abs/1905.07088. arXiv:1905.07088 [cs, stat].

[19] Pascal Vincent. A Connection Between Score Matching and Denoising Autoencoders. *Neural Computation*, 23(7):1661–1674, July 2011. ISSN 0899-7667. doi: 10.1162/NECO_a_00142. URL https://ieeexplore.ieee.org/document/6795935. Conference Name: Neural Computation.

[20] Yang Song and Stefano Ermon. Improved Techniques for Training Score-Based Generative Models, October 2020. URL http://arxiv.org/abs/2006.09011. arXiv:2006.09011 [cs, stat].

[21] Alex Nichol and Prafulla Dhariwal. Improved Denoising Diffusion Probabilistic Models, February 2021. URL http://arxiv.org/abs/2102.09672. arXiv:2102.09672 [cs, stat].

[22] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising Diffusion Implicit Models, October 2022. URL http://arxiv.org/abs/2010.02502. arXiv:2010.02502 [cs].

[23] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks, September 2019. URL http://arxiv.org/abs/1706.06083. arXiv:1706.06083 [cs, stat].

[24] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness May Be at Odds with Accuracy, September 2019. URL http://arxiv.org/abs/1805.12152. arXiv:1805.12152 [cs, stat].

[25] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples, March 2015. URL http://arxiv.org/abs/1412.6572. arXiv:1412.6572 [cs, stat].

[26] Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified Adversarial Robustness via Randomized Smoothing, June 2019. URL http://arxiv.org/abs/1902.02918. arXiv:1902.02918 [cs, stat].

[27] Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your Classifier is Secretly an Energy Based Model and You Should Treat it Like One, September 2020. URL http://arxiv.org/abs/1912.03263. arXiv:1912.03263 [cs, stat].

[28] Simran Kaur, Jeremy Cohen, and Zachary C. Lipton. Are Perceptually-Aligned Gradients a General Property of Robust Classifiers?, October 2019. URL http://arxiv.org/abs/1910.08640. arXiv:1910.08640 [cs, stat].

[29] Shibani Santurkar, Dimitris Tsipras, Brandon Tran, Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Image Synthesis with a Single (Robust) Classifier, August 2019. URL http://arxiv.org/abs/1906.09453. arXiv:1906.09453 [cs, stat].

[30] Logan Engstrom, Andrew Ilyas, Hadi Salman, Shibani Santurkar, and Dimitris Tsipras. Robustness (python library), 2019. URL https://github.com/MadryLab/robustness.

Acknowledgements

I want to thank my advisor, Iacopo Masi. Your constant availability and support have been invaluable throughout this journey. To me, you are like a superhero.

I am also profoundly thankful to all my university colleagues. Your continuous support has been the backbone of my undergraduate experience. Whenever one of us needed help, we were always there for each other. I am extremely lucky to have found such wonderful friends during my bachelor's studies.

To all my friends, thank you for your belief in me. Even if I sometimes fail to show my appreciation, please know that your support from behind the scenes means the world to me.

A special thank you goes to Emma. Over the past three years, we've faced both wonderful and challenging times. I believe it couldn't have been better, as true growth comes from experiencing both the highs and the lows. I hope our journey never ends, growing and maturing together.

And most importantly, I want to express my deepest gratitude to my mother and brother. Your faith and support have made it possible for me to pursue higher studies. I am truly grateful for everything you have done for me. I love you. I am who I am only because of you.