Abstract:

Generative modelling, a pivotal domain in deep learning, continuously evolves with innovative techniques striving to synthesize intricate data distributions. While established methodologies like Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) have made significant strides in this domain, the advent of diffusion models presents a novel approach to data generation. This research undertakes a comprehensive exploration into diffusion models, aiming to elucidate their foundational principles, provide a hands-on implementation in Python, and visually demonstrate the transformative phases of noise addition and reduction.

Our inquiry unveils the structured progression inherent to diffusion models: the systematic perturbation of real data into a state of near-randomness, followed by a reconstructive journey that commences from pure noise and culminates in data samples echoing the authentic dataset's characteristics. Through our Pythonic implementation and accompanying visual illustrations, we render a tangible account of an image's metamorphosis through these stages.

Highlighting both the theoretical constructs and practical implications of diffusion models, this research serves as a conduit for deep learning enthusiasts and practitioners to grasp the nuances and potentialities of this emerging generative paradigm.

Introduction:

Generative Adversarial Networks (GANs) have achieved impressive results across a plethora of applications. Yet, they are notoriously challenging to train, and their generated outputs can sometimes suffer from a lack of diversity. This is attributed to challenges such as mode collapse and vanishing gradients, among others. Concurrently, Variational Autoencoders (VAEs) possess a solid theoretical backbone, but designing an optimal loss function for them has proven to be elusive, often leading to suboptimal outputs.

A promising alternative emerges in the form of Diffusion Models. These models are grounded in probabilistic likelihood estimation techniques. The underlying principle of Diffusion Models centres around the concept of data degradation through noise. As information gradually diminishes due to noise interference, the intriguing hypothesis posited by these models is the potential to reverse this process. The aim? To extract and restore the original information from its noise-corrupted state.

This perspective shares some similarities with VAEs, especially in the optimization of an objective function by first projecting data into a latent space and then reconstructing it. However, the divergence is in the methodology. Instead of directly learning the data distribution, Diffusion Models focus on modelling a series of noise distributions within a Markov Chain framework. The goal is to "decode" data by methodically denoising it, in a step-by-step manner.

Applications and Intricacies of Diffusion Models:

Art & Design:

- Applications: Artists and graphic designers can utilize Diffusion Models to create novel artwork or intricate designs. The models can also suggest changes or evolutions to existing work, opening up new avenues of creativity.
- Intricacies: It requires careful calibration to ensure the generated designs align with human aesthetic sensibilities. Over-reliance on such tools might also diminish the authenticity of artistic work.

Medical Imaging:

- Applications: Diffusion Models can help in reconstructing medical images, enhancing resolution, or simulating potential disease progressions based on existing scans.
- Intricacies: Extreme precision is needed; minor errors can lead to misdiagnoses. Also, integrating such models into existing medical workflows requires rigorous validation and ethical considerations.

Entertainment and Gaming:

- Applications: Game designers and film producers can use these models to create dynamic environments, character designs, or even plot evolutions.
- Intricacies: The balance between creative input and machine-generated content must be maintained to avoid homogenization of content.

Scientific Visualizations:

- Applications: Scientists can use Diffusion Models for predicting and visualizing complex phenomena, such as weather patterns, molecular interactions, or astronomical events.

- Intricacies: Scientific accuracy is paramount. Over-relying on the model without validation can lead to incorrect conclusions.

Retail and Fashion:

- Applications: Brands can employ Diffusion Models to predict fashion trends or come up with novel product designs based on existing preferences.
- Intricacies: The fashion industry thrives on a mix of predictability and spontaneity. Models might skew too much towards existing trends, missing out on spontaneous shifts in consumer preferences.

Real Estate & Architecture:

- Applications: Architectural firms might use Diffusion Models to predict how buildings will age or to suggest novel designs based on existing architectural trends.
- Intricacies: Over-reliance on models might stifle architectural innovation. There's also a risk of uniformity in designs if multiple firms deploy similar models.

Financial Modelling:

- Applications: Financial analysts could employ Diffusion Models to visualize potential market movements based on vast datasets.
- Intricacies: Financial markets are notoriously unpredictable. Over-reliance on any predictive model, including Diffusion Models, can lead to significant financial missteps.

Problem Definition:

Generative modelling techniques have continually aimed to capture and replicate intricate data distributions. Over the years, deep learning has presented an array of approaches, each promising improved fidelity, versatility, and efficiency in generating synthetic data. While techniques like GANs and VAEs have demonstrated significant success, they come with inherent challenges such as mode collapse, training instability, and intricate latent space constraints.

In the backdrop of these challenges, the primary objective of our research is twofold:

Understanding Diffusion Models:

- Before application, it is imperative to decode the core principles governing any technique. For diffusion models, this means diving deep into its foundational logic and the algorithms underpinning its success. What differentiates this method from its predecessors?
- How does it leverage the iterative process of noise addition and subsequent reduction to generate data resembling the original dataset?

Practical Implementation and Visualization:

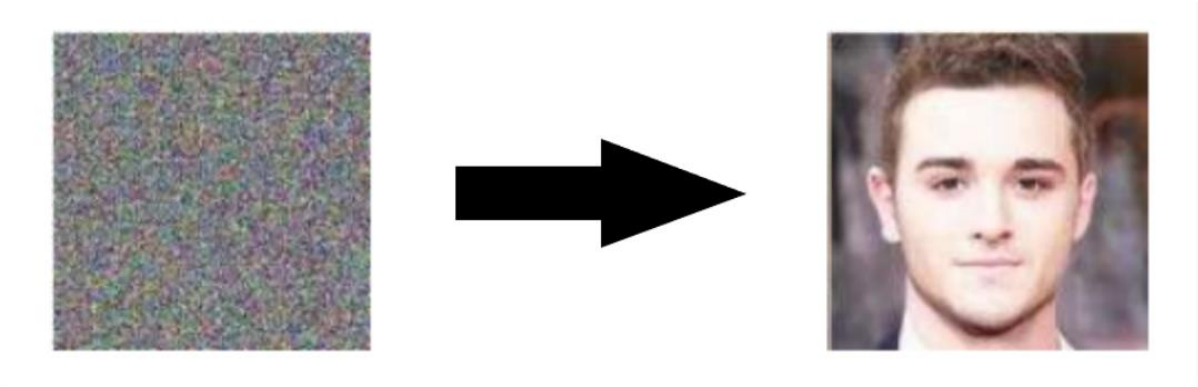
- Merely understanding a technique's theoretical underpinnings does not complete the research journey. The next pivotal step is the hands-on application.
- By implementing diffusion models in Python, we aim to translate theory into practice, providing a tangible toolset for the deep learning community.
- Furthermore, through visual demonstrations of images undergoing noise transitions, we aspire to provide an intuitive grasp of the model's operations, bridging the gap between abstract concepts and their real-world manifestations.

Working Process:

Diffusion Models operate within the realm of generative models, aiming to produce data that resonates with the characteristics of their training datasets. The primary operation unfolds in two stages:

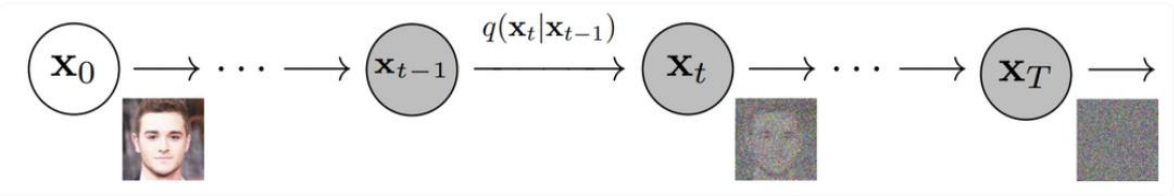
1. Noise Infusion: Initially, the training data undergoes a perturbation process where Gaussian noise is systematically introduced. This step degrades the original clarity of the data, embedding randomness into it.
2. Noise Reduction and Recovery: Post noise addition, the model's main task is to decipher the underlying patterns within the noisy data, aiming to reverse the perturbation. By learning this denoising mechanism, the model becomes adept at reconstructing the original data or, in essence, "removing" the added noise.

Upon completion of the training phase, generating new data with the Diffusion Model becomes intuitive. By feeding the model with randomly sampled noise, it leverages its learned denoising process to output data reminiscent of its training set.



Diffusion Models can be used to generate images from noise (adapted from source)

At its core, a Diffusion Model is defined as a latent variable model. What differentiates it is its utilization of a predefined Markov chain to map data into a latent space. Instead of directly projecting data, this model progressively infuses noise, leading to the formation of an approximate posterior denoted as $q(\mathrm{x}_{1:T}|\mathrm{x}_0)$, where $\mathrm{x}_1, \ldots, \mathrm{x}_T$ represent the latent variables sharing the same dimensional attributes as x_0. The following figure provides a visual representation, showcasing the Markov chain's application specifically to image data.



During the diffusion process, the image progressively evolves, eventually resembling pure Gaussian noise. The primary objective in training a diffusion model is to master this inverse transition – i.e., training $p_\theta(x_{t-1}|x_t)$. By retracing steps along this transformation chain, the model is empowered to synthesize new data.

Diffusion Models – Deep Dive

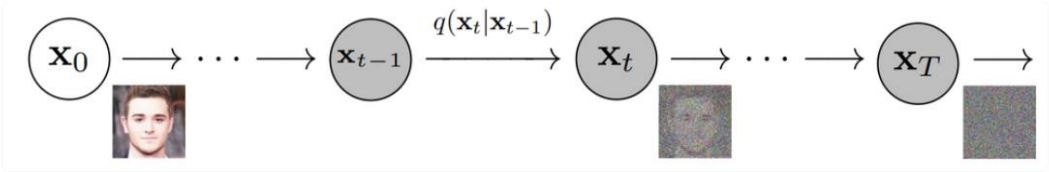
As outlined previously, the Diffusion Model operates in two distinct phases: the forward process (often termed as the 'diffusion process') where a piece of data, typically an image, undergoes incremental noise addition, and the reverse process (or 'reverse diffusion process'), where the noise-riddled data is reverted to a representation close to the original target distribution.

During the forward process, the transitions in the sampling chain can be approximated by conditional Gaussians, provided the noise intensity remains relatively low. This approach,

when integrated with the Markovian premise, offers a straightforward parameterization for the diffusion process:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \prod_{t=1}^{T} \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$$

$\beta_1, \ldots, \beta_T$ refers to the variance schedule which can learned adaptively or predetermined. For an appropriately chosen variance schedule, when T reaches sufficiently large values, x_T tends to closely resemble an isotropic Gaussian distribution.



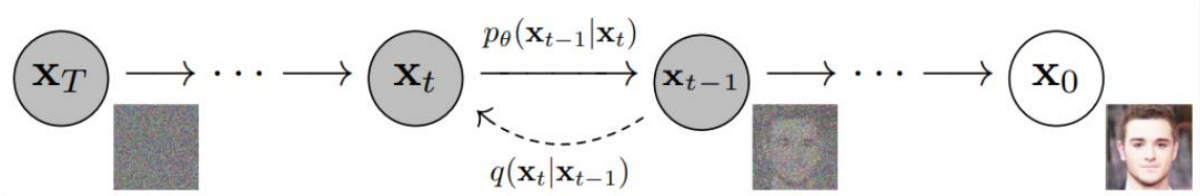
Given the Markov assumption, the joint distribution of the latent variables is the product of the Gaussian conditional chain transitions

As alluded to earlier, the true prowess of diffusion models is unveiled during the reverse process. During its training phase, the model is adeptly schooled to counteract the diffusion process, setting the stage for the generation of new data. Starting with an unadulterated Gaussian noise $p(\mathrm{x}_T) := \mathcal{N}(\mathrm{x}_T, 0, \mathrm{I})$, the model's learning process revolves around understanding the joint distribution $p_\theta(\mathrm{x}_{0:T})$ as:

$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := p(\mathbf{x}_T) \prod_{t=1}^{T} \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

Wherein the Gaussian transitions are influenced by time-dependent parameters that are subject to learning. It is paramount to underscore that, within the Markov framework, any reverse diffusion transition distribution is contingent solely on its immediate predecessor or successor, based on one's perspective.

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$



Training:

At its core, the training of a Diffusion Model revolves around pinpointing the reverse Markov transitions that amplify the likelihood of the dataset in use. In tangible training scenarios, this

translates to the minimization of the variational upper bound associated with the negative log likelihood.

$$\mathbb{E}[-\log p_\theta(\mathbf{x}_0)] \leq \mathbb{E}_q \left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] =: L_{vlb}$$

Our aim is to express the L_{vlb} using Kullback-Leibler (KL) Divergences. The KL Divergence serves as a non-symmetric statistical metric that quantifies the deviation of a probability distribution P from a reference distribution Q. The rationale behind this endeavour is the Gaussian nature of the transition distributions in our Markov chain. Utilizing KL divergences is particularly advantageous in this context since the KL divergence between two Gaussian distributions boasts a closed form.

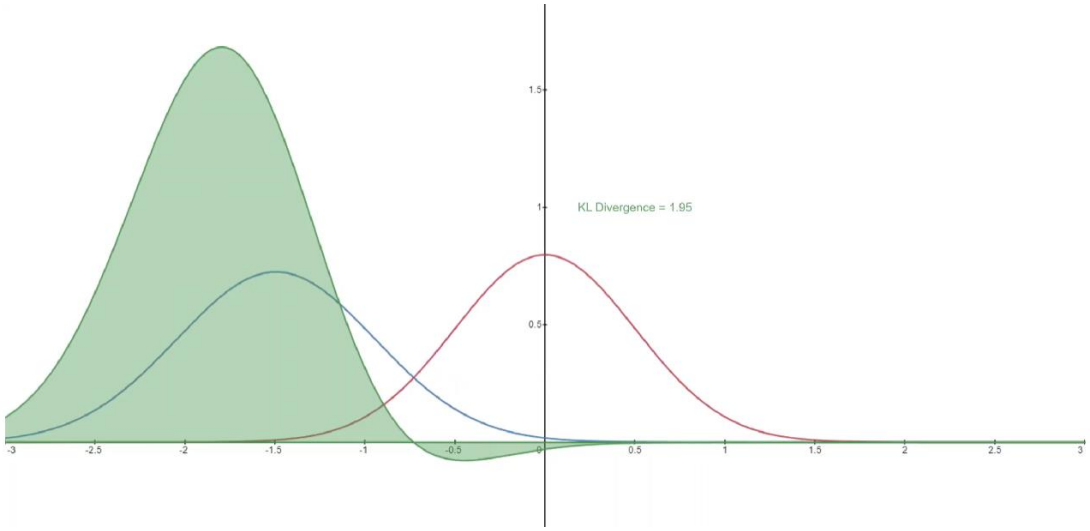
KL Divergence:

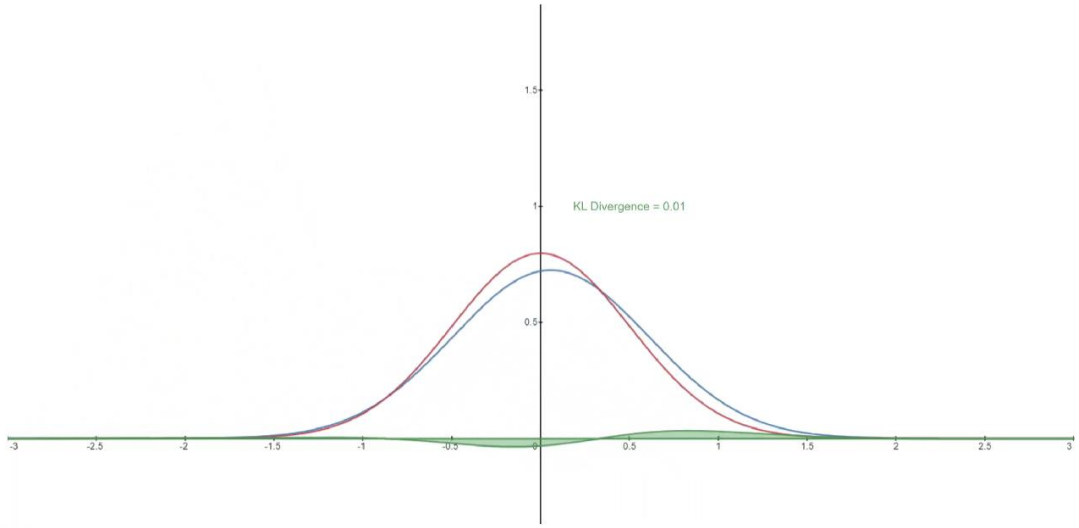
The mathematical notation to represent KL Divergence is

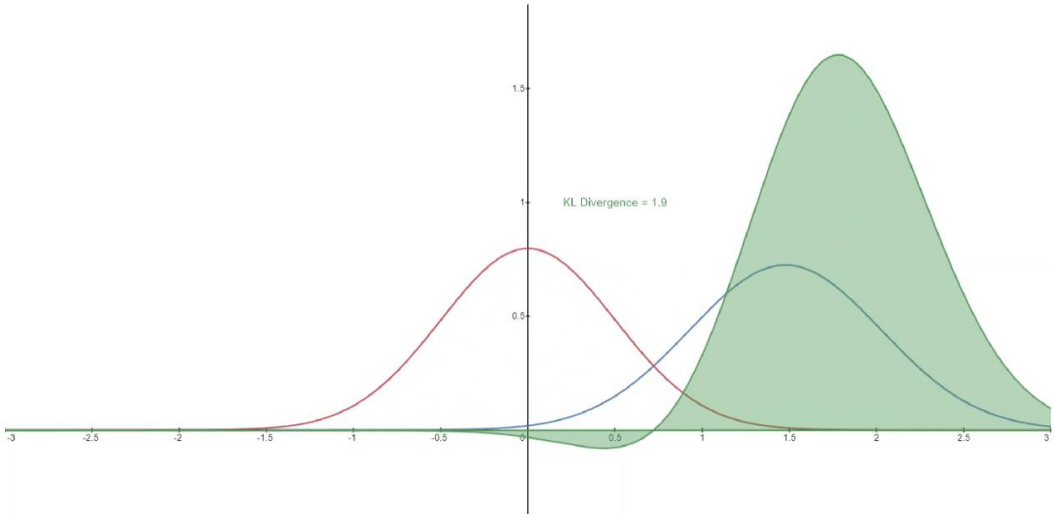
$$D_{\mathrm{KL}}(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx$$

The double bars indicate that the function is *not* symmetric with respect to its arguments.

In the illustration provided, the blue curve depicts distribution P, while the red curve represents the reference distribution Q. The function characterizing the KL divergence is shown by the green curve, derived from the integral definition mentioned earlier. The aggregate area beneath this green curve quantifies the KL divergence of P from Q at any specified instance, and this measure is also numerically presented alongside.







As touched upon earlier, there exists a viable methodology to rephrase L_{vlb}, predominantly utilizing the KL divergences:

$$L_{vlb} = L_0 + L_1 + ... + L_{T-1} + L_T$$

Where,

$$L_0 = -\log p_\theta(x_0|x_1)$$

$$L_{t-1} = D_{KL}(q(x_{t-1}|x_t, x_0) \,||\, p_\theta(x_{t-1}|x_t))$$

$$L_T = D_{KL}(q(x_T|x_0) \,||\, p(x_T))$$

By imposing a condition on x_0, within the forward process posterior of L_{t-1}, we attain a more manageable form. This structure ensures that all KL divergences are juxtapositions between Gaussian distributions. Such a configuration allows for precise computation of these divergences using closed-form solutions, sidestepping the need for Monte Carlo estimations.

Model Choice:

Having laid the mathematical groundwork for our objective function, we now embark on a series of decisions regarding the implementation of our Diffusion Model. For the forward process, our primary task is to determine the variance schedule, which typically exhibits incrementing values as the process unfolds.

Conversely, for the reverse process, our focus shifts to the selection of the Gaussian distribution parameterization and the corresponding model architecture(s). It's worth emphasizing the inherent versatility of Diffusion Models; the principal stipulation is the consistent dimensionality between the model's input and output. The intricacies of these selections will be delved into in the subsequent sections.

Forward Process and L_T

As highlighted previously, a pivotal decision in the forward process is the determination of the variance schedule. Specifically, we designate them as time-dependent constants, sidestepping their learnable nature. One might opt for a linear schedule ranging from $\beta_1 = 10^{-4}$ to $\beta_T = 0.2$, or even consider a geometric series.

Irrespective of the specific values selected, the constancy of the variance schedule ensures that L_T remains invariant concerning our ensemble of learnable parameters. This invariance facilitates its exclusion from training considerations.

Reverse Process and $L_{1:T-1}$

Let us delve into the determinations essential for outlining the reverse process. As previously indicated, we characterized the reverse Markov transitions utilizing a Gaussian framework.

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

It becomes imperative to elucidate the functional forms of μ_θ or Σ_θ. While intricate parameterizations for Σ_θ do exist, our approach is to align it with

$$\boldsymbol{\Sigma}_\theta(x_t, t) = \sigma_t^2 \mathbb{I}$$
$$\sigma_t^2 = \beta_t$$

In essence, we predicate on the belief that the multivariate Gaussian is a composite of independent Gaussians, all bearing consistent variance. Notably, this variance is not static but can evolve over time. We synchronize these variances to mirror those in our forward process variance schedule.

The new formulation of Σ_θ,

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I})$$

Which transforms into

$$L_{t-1} = D_{KL}(q(x_{t-1}|x_t, x_0) \,||\, p_\theta(x_{t-1}|x_t))$$

And further into

$$L_{t-1} \propto ||\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)||^2$$

In this context, the primary constituent of the difference emerges as a linear amalgamation of x_t and x_0, influenced by the variance schedule denoted as β_t. It is pertinent to note that the precise structure of this function does not hold paramount importance for our current discourse.

Central to our discussion is the implication that the most direct parameterization of μ_θ seeks to forecast the diffusion posterior mean. Intriguingly, findings suggest that training μ_θ to anticipate the noise component at specific timesteps produces superior outcomes. To be specific, consider

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right)$$

Where,

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

Such considerations steer us towards an alternative loss function.

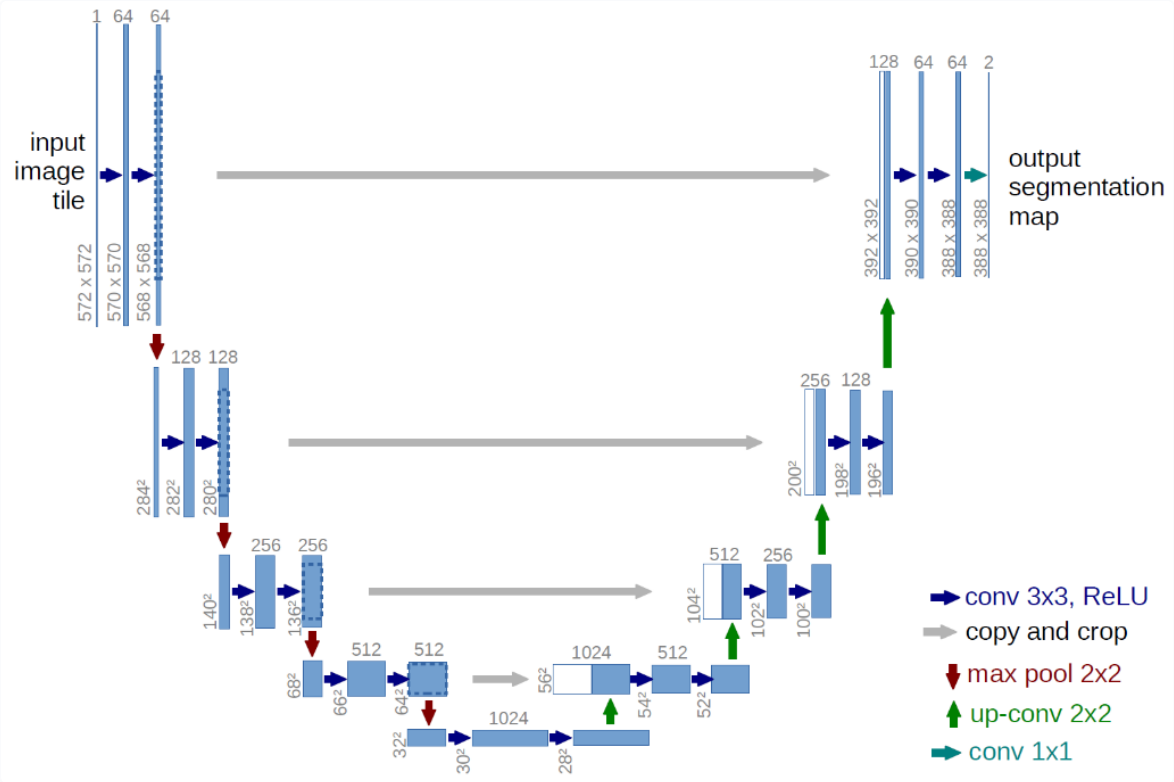
$$L_{\text{simple}}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}} \left[\left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2 \right]$$

This function has demonstrated attributes like enhanced training stability and superior performance.

Network Architecture

Before delving deep into the intricacies of our loss function, we need to establish the structural foundation for our model. It is worth emphasizing that the only structural constraint we impose on this model is the equivalence of input and output dimensions.

Considering such constraints, it is logical that image-based Diffusion Models predominantly employ architectures reminiscent of U-Nets.



Reverse Process Decoder and L_0:

Journeying through the reverse process entails numerous transformations, all governed by continuous conditional Gaussian distributions. Once this reverse cascade concludes, we are left with the task of synthesizing an image characterized by integer pixel values. This task demands a mechanism to ascertain discrete (log) likelihoods corresponding to every conceivable pixel value spanning the image.

Achieving this necessitates positioning the concluding transition in the reverse diffusion sequence as an autonomous discrete decoder. To evaluate the likelihood of a specific image x_0 considering x_1, we initially advocate for data dimension independence:

$$p_\theta(x_0|x_1) = \prod_{i=1}^{D} p_\theta(x_0^i|x_1^i)$$

Here, D stands for the data's dimensionality, with the superscript i denoting the extraction of a specific coordinate. The subsequent challenge is to gauge the probability of individual integer values for specific pixels, based on the distribution of potential values for the analogous pixel in the marginally noised image at the time $t = 1$. Represented as

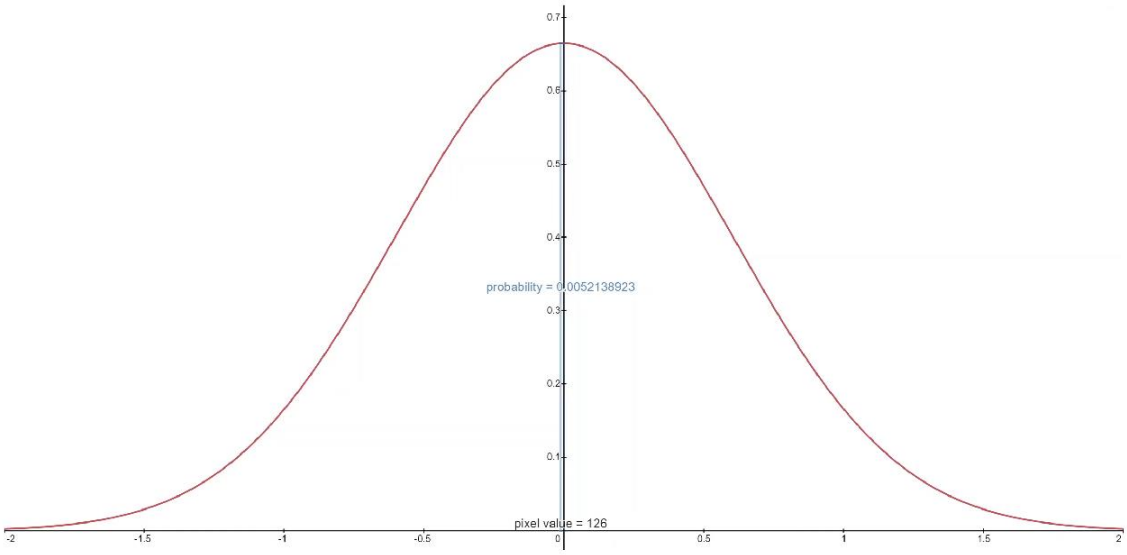
$$\mathcal{N}(x; \mu_\theta^i(x_1, 1), \sigma_1^2)$$

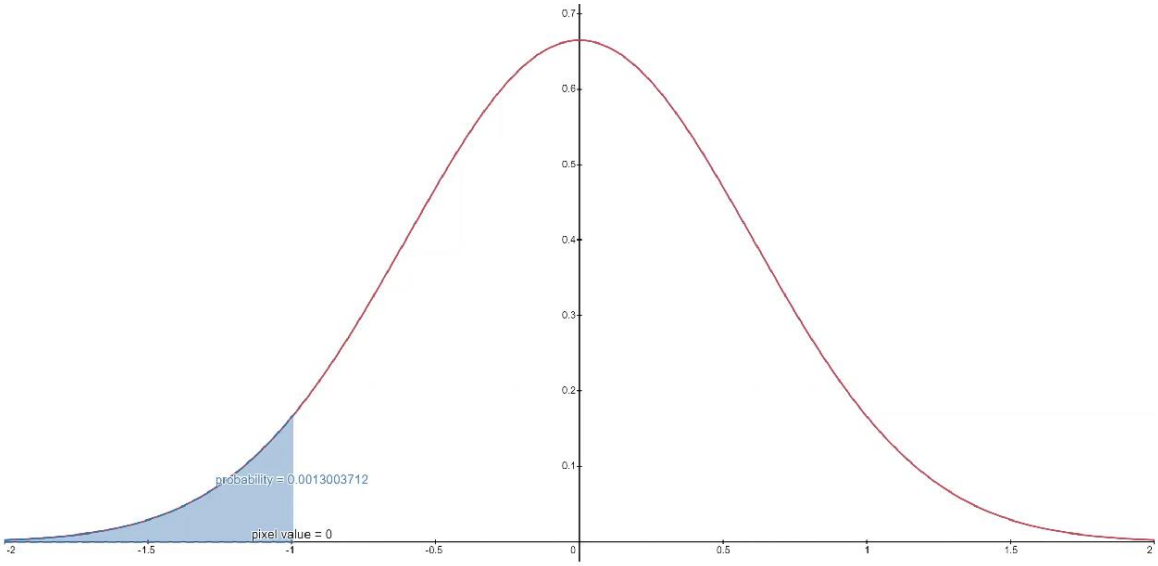
This approach pivots on pixel distributions for $t = 1$ which are sourced from the following multivariate Gaussian. Given its diagonal covariance matrix, this Gaussian can be segmented into a compilation of univariate Gaussians, each pertaining to a distinct data dimension.

$$\mathcal{N}(x; \mu_\theta(x_1, 1), \sigma_1^2 \mathbb{I}) = \prod_{i=1}^{D} \mathcal{N}(x; \mu_\theta^i(x_1, 1), \sigma_1^2)$$

For our image data, we operate under the presumption that they consist of integer values ranging from $0,1,\ldots,255$ – a standard for RGB images. These integers undergo a linear scaling transformation to fit within the range $[-1,1]$. This continuous range is then discretized into what we refer to as "buckets". When considering a specific scaled pixel value x, its bucket is bounded by the range $[x - 1/255, x + 1/255]$. In order to deduce the probability associated with a particular pixel value x, given the univariate Gaussian distribution of the respective pixel in x_1, one needs to calculate the area beneath the Gaussian curve that lies within the confines of the bucket centred at x.

Visually representing this, consider a graph where each bucket's area indicates the corresponding probabilities for a Gaussian distribution centred at mean 0. Within the context of our study, this translates to a distribution boasting an average pixel value of $255/2$ (or, half the brightness level). The red curve on this graph signifies the distribution of a certain pixel at the t=1 juncture. Meanwhile, the enclosed areas beneath this curve reveal the likelihood of the equivalent pixel value at the t=0 moment.





For any pixel at t=0, the corresponding value of $p_\theta(x_0|x_1)$ is obtained by multiplying these pixel values. This methodology can be neatly represented using the below equation.

$$p_\theta(x_0|x_1) = \prod_{i=1}^{D} p_\theta(x_0^i|x_1^i) = \prod_{i=1}^{D} \int_{\delta_-(x_0^i)}^{\delta_+(x_0^i)} \mathcal{N}(x; \mu_\theta^i(x_1, 1), \sigma_1^2) dx$$

This is further broken down as

$$\delta_-(x) = \begin{cases} -\infty & x = -1 \\ x - \frac{1}{255} & x > -1 \end{cases}$$

and

$$\delta_+(x) = \begin{cases} \infty & x = 1 \\ x + \frac{1}{255} & x < 1 \end{cases}$$

Using this expression for $p_\theta(x_0|x_1)$, the last component of L_{vlb} that is not defined in terms of a KL Divergence can be deduced as

$$L_0 = -\log p_\theta(x_0|x_1)$$

Final Objective:

Building on the insights from the previous section, it has been determined that predicting an image's noise component at specific timesteps yields optimal outcomes. The final objective used in this context is represented as:

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}[\, \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\|^2 \,]$$

Experimentation:

To gauge the practical performance and versatility of Diffusion Models, a comprehensive series of experiments were executed. This section delves into the methodologies adopted, the datasets used, and the critical observations derived from these experimental procedures.

Experimental Setup:

- All experiments were facilitated by Google Colab, taking advantage of Tensor Processing Units (TPUs) and TensorFlow-GPUs. The overarching motivation to employ this high-performance computational environment stemmed from the expected intensive computational demands of the Diffusion Model.

Experiment 1: Base Model on a Single Image

- The experiment focused on the basic Diffusion Model. A single image was taken as the subject, and it underwent both the diffusion and reverse diffusion processes.

Experiment 2: Oxford Flowers 102 Dataset

- Venturing into more intricate datasets, the second experiment leveraged the Oxford Flowers 102 dataset. This dataset is celebrated for its diversity, containing approximately 8,000 natural images of flowers. Notably, the official splits of this

dataset were found to be imbalanced, with a disproportionate number of images in the test split. To rectify this and ensure a more balanced training set, new splits were created (80% train, 20% validation) using the TensorFlow Datasets slicing API.

Experiment 3: MNIST Dataset

- The third experiment transitioned to a more conventional dataset – the MNIST dataset. This dataset is a staple in the machine learning community, containing handwritten digits, and served as an ideal benchmark for the Diffusion Model.

Performance Analysis:

- Collating insights from all experiments, the Diffusion Model emerged as a potent tool in data generation. Nevertheless, computational demands remained a consistent challenge, especially with more intricate datasets. These insights underscore the model's potential while also pointing towards avenues for optimization.

Conclusions from Experiments:

- The experiments affirmed the prowess of Diffusion Models in diverse scenarios, ranging from singular images to complex, natural datasets. However, the computational intensity necessitates continuous exploration for optimization strategies, making the models more amenable for large-scale or real-time operations.

Results:

The subsequent segment is dedicated to outlining the findings and observations drawn from each of the previously detailed experiments. Here we discuss both the qualitative and quantitative outcomes and how they align with our initial objectives and hypotheses.

Experiment 1: Base Model on a Single Image

- Processing Time:
  Training showcased computational demands. While 1,000 epochs took 2 hours on a CPU, it was limited training to 300 epochs, which completed in under an hour, to save time.
- Quality of Output:
  The reduced epochs led to a trade-off in image clarity. Though the image was identifiable, additional training epochs would likely enhance the resolution and reduce ambiguities.

- Efficiency vs. Quality:

  Balancing computational efficiency with desired output quality became a clear challenge in this experiment.
- Key Takeaway:

  The Diffusion Model's performance on the image underscored the need for powerful computing resources when working with more extensive training epochs. Finding the optimal spot between training time and quality remains critical.

Experiment 2: Oxford Flowers 102 Dataset

- Training and Image Quality: The model was subjected to a rigorous training regimen spanning 50 epochs. The duration of this training phase varied considerably based on the computational resource:

  T4 GPU: Approximately 2 hours

  A100 GPU: Roughly 30 minutes

  Notably, this extensive training yielded high-quality image generations, indicating the model's potential in rendering sophisticated datasets with clarity and fidelity.
- Key Takeaway: While the model's performance was commendable in terms of output quality, there is a stark disparity in training durations between different GPUs. This reinforces the significance of hardware selection in determining both efficiency and performance.

Experiment 3: MNIST Dataset

- Performance Speed: The primary observation from this experiment was the time-consuming nature of the process. As each timestep had to be looped through individually, the overall computation was slow-paced, even with optimized computational resources.
- Key Takeaway: The base model, though effective in its functionality, demands considerable computational time for a single image. This calls for further optimizations and more efficient implementations, especially if the model is to be scaled for broader datasets.

Concluding Remarks on Results:

The experiments underscored both the capabilities and limitations of the Diffusion Model. The quality of generated images, especially from the Oxford Flowers 102 dataset, is testament to

the model's potential. However, the computational demands, as evidenced by the long training durations and the iterative nature of the process, point towards the necessity for model and computational optimizations.

Conclusion:

Throughout our exploration of Diffusion Models, we have seen both their immense potential and the challenges they present. These models are powerfully generative, capable of creating high-fidelity reproductions and novel images, especially evident in our experiments with the Oxford Flowers 102 dataset and MNIST.

However, with their strengths come complexities. The computational demands of these models are considerable. In our single image test and the MNIST dataset experiment, it became evident that optimizing efficiency without compromising the quality is a tightrope we must navigate. While advancements in hardware, such as GPUs and TPUs, have mitigated some of these challenges, they are not entirely absolved.

Another significant observation was the trade-offs associated with model parameterization and dataset choices. Imbalances in the Oxford Flowers dataset splits presented challenges, necessitating a fresh partitioning approach. Similarly, our decision to limit epochs in the MNIST experiment highlighted the continual interplay between time, computational resources, and quality.

Key Takeaways:

- Versatility and Power: Diffusion Models offer a versatile and potent tool in the field of generative modelling.
- Computational Demands: High computational costs, both in terms of time and resources, need addressing for broader and efficient application.
- Dataset Integrity: Ensuring balanced and representative datasets is crucial for achieving best possible outcomes.

As we move forward, with advancements in technology and deeper insights into model optimizations, Diffusion Models' horizon looks promising. Their potential applications, from art and design to more nuanced scientific visualizations, make them an exciting field of study and development.

References:

https://medium.com/@vedantjumle/image-generation-with-diffusion-models-using-keras-and-tensorflow-9f60aae72ac

https://keras.io/examples/generative/ddim/

https://www.assemblyai.com/blog/diffusion-models-for-machine-learning-introduction/

https://arxiv.org/pdf/2006.11239.pdf?ref=assemblyai.com

https://github.com/lucidrains/denoising-diffusion-pytorch

https://github.com/mikonvergence/DiffusionFastForward