Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Telecommunications and Media Informatics

Diffusion-based object generation on images for self-driving environment

Master's Thesis

Author
Katica Bozsó

Advisor
dr. Bálint Gyires-Tóth
András Béres

December 8, 2023

Contents

HALLGATÓI NYILATKOZAT

Alulírott *Bozsó Katica*, szigorló hallgató kijelentem, hogy ezt a dolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2023. december 8.

Bozsó Katica
hallgató

Kivonat

Önvezető járműrendszerek esetében a deep learning folyamatok nagy mértékben támaszkodnak a kiegyensúlyozott és változatos adathalmazra - aminek összeállítása komoly kihívást jelent, hiszen némely minták rendkívül ritkán fordulnak elő, pl. különleges időjárási körülmények vagy pedig speciális objektumkompozíciók. Mély neurális háló alapú megoldások, kifejezetten a nem rég teret nyerő generatív hálózatok orvosolhatják ezen problémát. A területhez tartozó egyik legfontosabb fejlesztés a diffúzió alapú megközelítés, mely zajból állít elő új képeket. Túlnyomórészt ezen megoldások a 'text2image' metódust alkalmazzák, vagyis szövegbemenet segítségével teszik vezérelhetővé a generálási folyamatot. Előrehaladott képességeik ellenére azonban ezek a modellek még nem biztosítanak teljesen explicit kontrollt a generált tartalom felett, különösen olyankor amikor az objektumok relatív helyzetét kell meghatározni egy adott képen. A dolgozat célja az volt, hogy megoldást javasoljon erre a korlátozásra egy szemantikus szegmentáción alapuló vezérlési mechanizmus generatív diffúziós modellbe történő integrálásával, és demonstrálja annak hatékonyságát az autóipar területén. Ezzel a megközelítéssel tetszőleges önvezető domainbe tartozó jelenetek generálhatóak, ezáltal bővítve a hiányos adathalmazokat.

Abstract

In the autonomous mobility industry, deep learning pipelines are critically dependent on the balance and variety of training data. Achieving this balance is particularly challenging due to the scarcity of data in rare scenarios, such as unique weather conditions or specific traffic configurations. Deep learning-based methods, particularly those within the emerging field of generative AI, hold potential for advanced solutions. A key development in this domain is the diffusion-based approach, capable of generating novel images from a random noise distribution. Predominantly, these models utilize a 'text2image' methodology, enabling the generation of images via textual prompts. However, despite their advanced capabilities, these models do not yet provide complete explicit control over the generated content, particularly in terms of the relative positioning of objects within images. The goal of this thesis was to propose a solution to this limitation by integrating a semantic segmentation based control mechanism into a generative diffusion model and demonstrate its effectiveness on the automotive domain. Through this approach, arbitrary self-driving scene setups can be produced, therefore enriching insufficient datasets.

Chapter 1

Introduction

Deep neural networks have revolutionized image generation, finding widespread application in fields such as arts [1], entertainment, medical science [2], and the development of autonomous driving systems [3][4]. A particularly captivating branch of generative AI is the emergent diffusion-based [5] approach. This method hinges on training a model adept at noise prediction, capable of iteratively crafting images from a standard noise distribution during inference. As many advancements have aimed to enhance the controllability of image generation, cutting-edge solutions now mostly employ a 'text2image' approach, enabling systems to generate images guided by textual prompts [6].

However, two challenges stand out: advanced models are vast, potentially putting them out of reach for average users due to their training and deployment complexities. Conventionally, these models are accessible via online API[1]s and platforms, but such avenues rarely provide a transparent view of the inner workings or the ability to fine-tune on custom datasets. Secondly, while textual prompts are innovative, they are yet to offer complete explicit control over generation, especially when specifying the relative positioning of objects within an image. Although some fields might not prioritize this feature, many could significantly benefit from enhancements in this area.

Autonomous driving systems is a domain where there is an insatiable demand for diverse, and sometimes very specific training data. The collection process, especially for rare scenarios such as pre-accident object positioning, poses challenges not only in terms of cost, but also in feasibility, therefore acquiring such recordings could be a pivotal achievement. Textual prompts may help to generate images where the composition is simple, e.g. 'a red car at the cross-roads', but when there is a particular idea about the scene setup, e.g. 10 cars with multiple colors at different directions around likewise interacting humans, a pure text-based description is hardly applicable.

Semantic segmentation maps could mean a more reasonable alternative, since they possess a greater descriptive power on the pixel-level. The goal is to facilitate automotive data en-

[1] Application Programming Interface

richment through offering control by mask-guidance. Classes occuring on groups of image pixels may be influenced directly in this manner, while also enabling multiple utilization possibilities of the model: Companies typically possess at least a minimal amount of training data containing semantic segmentation masks. By the help of a mask-guided diffusion model, these could be used 'out-of-the-box', generating multiple versions (e.g. colors of cars change) of the same scene setup. The initial masks may be further modified (e.g. via basic image editing tools), providing even more uniqe inputs for the model. To illustrate the effectiveness of the method, even handmade drawings may be used as inputs.

In my thesis I introduce a semantic segmentaion mask-guided model, trained specifically for self-driving environment data generation. I leveraged the Berkeley Deep Drive dataset, which comprises traffic participant frames annotated with semantic segmentation. Outcomes validated the concept of using such masks for scene control, while highlighting the method's potential for scalability. Furthermore, my implementation is designed to be compact and minimal, enabling everyday users to grasp, experiment with, and adapt this approach to address their unique challenges and ideas in generative image modeling.[2]

[2]repository will be published later at `https://github.com/kajc10/semseg-guided-diffusion`

Chapter 2

Theoretical Background

2.1 Classical Image Augmentation Methods

Classical data augmentation methods can typically be divided into two principal categories: photometric and geometric augmentations [7].

Photometric augmentations - typically referred to as color augmentations - do not modify the intrinsic structure of the data, but nonetheless, produce a visually distinct output. Some of the more common ones in this category include contrast adjustments, brightness modifications, saturation enhancements, and hue shifts.

Although these augmentations provide diversity in the visual appearance of the data, they primarily act on the pixel values, therefore they do not facilitate any structural modifications to the underlying scene of the image.

Figure 1: Color augmentation examples

On the other hand, geometric augmentations are designed to address this limitation. These methods introduce structural changes to the image, thereby altering the composition and layout of the scene. Some commonly used geometric augmentations are resizing, cropping, flipping (both horizontal and vertical), and rotating. By employing these techniques, the objects' relative positions within the image can be altered, which can lead to more diverse scene compositions and enhance model generalization.

Figure 2: Geometric augmentation examples

In a standard dataloader, these augmentation functions can be applied with a probability factor p, allowing the model to learn from both the original and the modified data concurrently. Furthermore, the parameters governing these augmentations, such as the degree of rotation or the intensity of brightness change, can be fine-tuned to optimize the model's performance. Adjusting these parameters is often an empirical process, influenced by the nature of the dataset and the specific problem at hand.

For both photometric and geometric methods, it is worth noting that while they serve as a means of data enrichment, they do not inherently produce entirely new scene configurations. The primary advantage of using these methods is to reduce the model's tendency to overfit by providing variations of the same sample. To create explicit scene configurations or generate novel content, there is a need to pivot towards deep dearning-based methods, which are equipped with more advanced tools for such purposes.

In the subsequent sections, deep learning-based augmentation and generation methods will be addressed, focusing on their capabilities and potential to enrich datasets further.

2.2 Key Branches of Generative Models

Generative deep learning models are dedicated to understanding and replicating the inherent distribution of given data. By effectively learning the 'essence' of a dataset, these models are capable of generating new data samples that can be considered as drawn from the same distribution as the training data. The preservation of the intricate relationships and patterns in the original data can often lead to insightful and creative synthetic outputs. Among the diverse list of generative models for computer vision tasks, three branches are particularly noteworthy due to their distinct capabilities and broad applications: Generative Adversarial Networks (GANs) [8], Variational Autoencoders (VAEs) [9], and Diffusion Models [10].

2.2.1 Generative Adversarial Networks

GANs operate on the concept of a zero-sum game between two components: a generator and a discriminator. The generator creates synthetic data, while the discriminator evaluates this data against the real dataset. The aim is for the generator to produce data that

the discriminator cannot distinguish from the real dataset. This adversarial process drives the generator to create increasingly realistic data, useful for creating diverse and complex scenarios. However, the training of GANs can be challenging due to issues like training instability and the phenomenon known as mode collapse [11]. In this situation, the generator starts to produce a restricted range of samples, limiting its diversity. Furthermore, it is essential to note that GANs do not offer a direct representation of the data's density function [8] [12], which could constrain their utility in assignments that demand in-depth data exploration.

2.2.2 Variational Autoencoders

VAEs [13] assume that the data is generated by some latent (hidden) variables and aim to model the data distribution explicitly. A typical VAE comprises an encoder, which translates the input data into a latent space, and a decoder, which then recreates the data from this latent representation. For clarification, the 'autoencoder' phrase is used due to the model's encoder-decoder structure. Owing to their probabilistic nature, VAEs excel at managing uncertainty and creating new data efficiently. These models can be particularly advantageous when it is vital to understand the data and manipulate the latent variables [14]. However, due to some simplifying assumptions in their design, the samples generated by VAEs may lack the sharpness or realism found in those produced by Generative Adversarial Networks (GANs) [15].

2.2.3 Diffusion models

Diffusion models [5] represent another distinctive approach for data generation. Traditionally, these models propose a stochastic process to gradually transform the data distribution into a known distribution, typically Gaussian, through a sequence of small, noise-adding steps. This diffusion process can be reversed to generate new data samples. Diffusion models do not require an explicit likelihood function and can model complex data distributions, which offer a great deal of flexibility. Although they can be computationally intensive during the generation process, the diversity and quality of the data they generate could greatly enhance the robustness of deep learning models that are lacking diverse data [16].

While there have been developments that transition the diffusion process to latent space - commonly termed as "Stable Diffusion" [17] - using expansive network architectures, and some recent advancements [18] [19] have even moved away from the classical noise introduction strategy to predicting VQ (Vector Quantization) [20] tokens directly, this study adheres to the foundational noise prediction approach of the original diffusion models.

2.2.4 Generative learning trilemma

After addressing the potential methods, a design choise had to be made. The generative learning trilemma [21] (see on Figure 3) provides guidance in deciding which branch is most suitable for a given task. The three features taken into consideration are quality, diversity and speed. GANs are able to produce high quality samples fast, but lack diversity. VAEs excel at fast sampling and model coverage, but fail to produce high quality samples. Finally, diffusion models are adept at producing diverse, high quality samples, at the cost of speed.

The current goal is to aid automotive data enrichment, for which quality and diversity are crucial, but fast sampling is not, since the model will not be deployed to an edge device. After evaluating the possibilities based on the generative trilemma, the optimal choice was to use diffusion models.

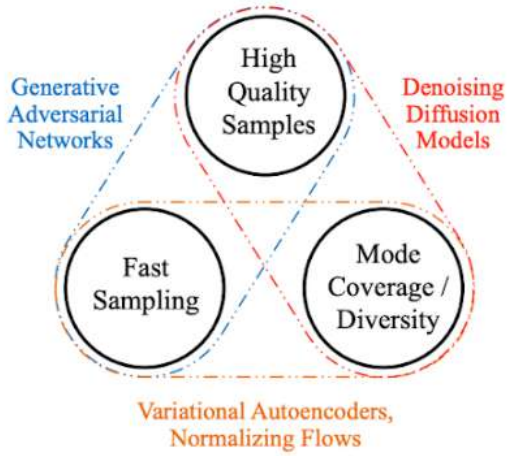


Figure 3: Generative learning trilemma[1]

2.3 Denoising Diffusion Probabilistic Models

Denoising Diffusion Probabilistic Models (DDPM) [5] constitute a popular basis for diffusion-based generative models, and they serve as the foundation for the experiments in my study. This subsection aims to present the primary concepts behind DDPM and to briefly explain its functioning. For a deeper understanding and more technical details - especially regarding mathematical derivations - the reader is encouraged to consult the original paper and supplementary explanatory blog posts [22][23][24]. I would like to highlight that the following technical summary is processing the content of an excellent video summary [25] about the topic.

[1] https://developer.nvidia.com/blog/improving-diffusion-models-as-an-alternative-to-gans-part-1/ accessed: 2023.11.01.

DDPM's main approach originates from the concept proposed by the authors of *Deep Unsupervised Learning using Nonequilibrium Thermodynamics* [10], who articulate their methodology as follows:

> 'The essential idea, inspired by non-equilibrium statistical physics, 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 of the data.'

Taking inspiration from these insights, the authors of DDPM demonstrate that such model can be used for effective data synthesis.

While DDPM may initially seem complex, the underlying idea is rather simple. An x_0 image is taken as input, and Gaussian noise is progressively added over a series of T steps, leading to a significantly distorted version. Subsequently, a neural network is trained with a specific objective: for a given noisy image at timestep t, it should identify the precise noise introduced during that step, allowing the image to move closer to its state from the preceding timestep $t-1$. Essentially, the network learns to predict the noise added at each specific timestep. The overarching aim is to systematically reverse the noise, moving from the heavily distorted image back to the original. Once the model is adeptly trained, the process can begin with an image that is purely noise. By consistently feeding this noisy image into the network and subtracting its denoising predictions, the noise steadily fades, revealing a synthesized image by the end of the iterations.

The following subsections offer further clarification on the mechanics of this forward and backward process.

2.3.1 Forward diffusion - adding noise

The forward process is responsible for gradually applying noise (sampled from a normal distribution) over lots of steps (authors used 1000, this is what I applied as well) to an x_0 data sample until it turns into complete noise. This can be formulated as a Markov chain[2] of T steps, illustrated on Figure 4. The distribution of the noised image can be described as:

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

Here t is a timestep (1-T), x_0 is a data sample from the real data distribution $q(x)(x_0 \sim q(x))$, β_t is variance (0-1) and I is the Identity matrix. The β_t variance can be fixed as a constant or scheduled over the T timesteps. In the original DDPM paper linear scheduler is used, increasing from $\beta_1 = 0.0001$ to $\beta_T = 0.02$. Based on literature [26], a cosine

[2] `https://en.wikipedia.org/wiki/Markov_chain` accessed: 2023.11.02.

scheduler is more effective, therefore I decided to rather use that. Note that $q(x_t|x_{t-1})$ is still a normal distribution defined by the mean $(\sqrt{1-\beta_t}x_{t-1})$ and variance $(\beta_t I)$.

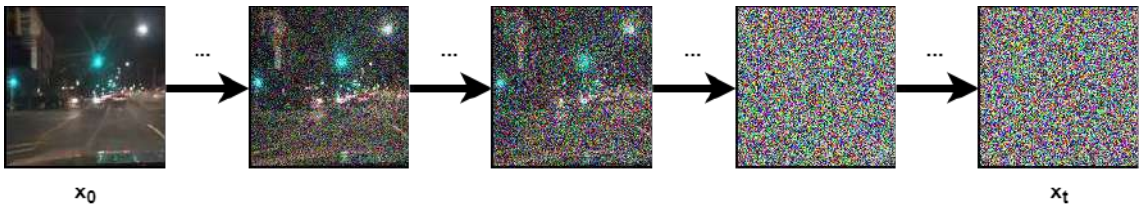


Figure 4: Forward diffusion

With the help of the so called 'Reparameterization trick'[3], a sampled image x_t can be expressed as:

$$x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\varepsilon$$

Fortunately, this can be further derived into a closed-form formula. This way we can directly generate noisy image for an arbitrary timestep t in a **single step**, thus making the process much faster. After defining α_t, $\hat{\alpha}_t$ and ε as:

$$\alpha_t = 1 - \beta_t$$

$$\hat{\alpha}_t = \prod_{i=1}^{t} \alpha_i$$

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

The formula can be written as :

$$x_t = \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\varepsilon$$

$$= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\varepsilon$$

$$= \sqrt{\alpha_t\alpha_{t-1}\alpha_{t-2}}x_{t-3} + \sqrt{1-\alpha_t\alpha_{t-1}\alpha_{t-2}}\varepsilon$$

$$...$$

$$= \sqrt{\alpha_t\alpha_{t-1}...\alpha_1\alpha_0}x_0 + \sqrt{1-\alpha_t\alpha_{t-1}...\alpha_1\alpha_0}\varepsilon$$

$$=> \boxed{x_t = \sqrt{\hat{\alpha}_t}x_0 + \sqrt{1-\hat{\alpha}_t}\varepsilon} \tag{2.1}$$

[3] https://theaisummer.com/latent-variable-models/#reparameterization-trick accessed:2023.11.02.

2.3.2 Reverse diffusion - removing noise

Noising an image is fairly simple via the closed-form formula. Doing the opposite and removing the noise is a more complicated task. Directly predicting x_0 could be an option, but authors found that this leads to worse sample quality than their other proposals. A normal distribution needs mean and variance ($\mathcal{N}(\mu, \sigma^2)$), but the variance can be fixed, therefore it is enough to **predict the mean of the Gaussian distribution at each timestep**. To make it computable, Variational Lower Bound[4] is applied. For further math derivation and explanation, see [24]. Mean Squared Error (MSE) can be computed between μ and predicted μ, however, the objective of predicting the mean can be reformulated into the objective of **directly predicting the noise**.

This way, a simplified objective (referred to as such in the original paper) can be written as:

$$\boxed{L_{simple} = E_{t,x_0,\varepsilon}[||\varepsilon - \varepsilon_\theta(x_t, t)||^2]} \tag{2.2}$$

When a network is capable of predicting the current noise at a given t, the process can be applied iteratively. Noise is predicted, then removed from the noisy image, thus forming a bit less noisy image. When starting from t=T, by t=1, a clean, novel image is created. See Figure 5.

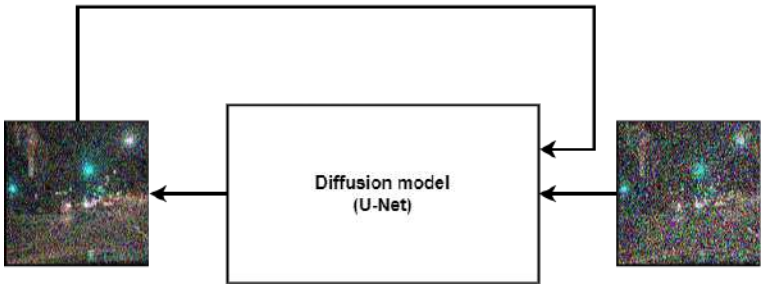


Figure 5: Reverse diffusion

[4] https://yunfanj.com/blog/2021/01/11/ELBO.html accessed: 2023.11.02.

2.3.3 Complete pipeline

The authors of DDPM provided the following algorithm to define the complete training pipeline:

Algorithm 1 Training

repeat
  $x_0 \sim q(x_0)$
  $t \sim Uniform(\{1, \dots, T\})$
  $\epsilon \sim \mathcal{N}(0, I)$
  Take gradient descent step on
    $\nabla_\theta ||\epsilon - \epsilon_0 \left(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, t\right) ||^2$
until converged

An image is sampled from the training dataset, along with timesteps and noise from a normal distribution. Noised variations of it are generated at each timestep t via **forward diffusion**. As described above, this can be formulated as: $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\varepsilon$. Then the model takes x_t and t as input and predicts the actual noise on the image between t and $t - 1$. MSE[5] loss is calculated between the predicted and the original noise. Through optimization for this loss, the model learns to predict current noise present in an image between t and $t - 1$ timesteps.

A trained model can be used for generating new samples, using the following algorithm.

Algorithm 2 Sampling

$x_T \sim \mathcal{N}(0, I)$
for $t = T, \dots, 1$ **do**
  $z \sim \mathcal{N}(0, I)$ if $t > 1$, else $z = 0$
  $x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(x_t, t)\right) + \sigma_t z$
end for
return x_0

Here $\epsilon_\theta(x_t, t)$ denotes the model's output for x_t (sampled from a normal distribution) and t. The output is the predicted noise on noisy image at timestep t.

Clarifying point 3-4) it should be noted that extra noise is not added when $t = 1$. Thus the denoising function has two forms:

t>1:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1 - \hat{\alpha}_t}}\epsilon_\theta(x_t, t)) + \sqrt{\beta_t}\epsilon$$

[5]Mean squared error - `https://en.wikipedia.org/wiki/Mean_squared_error` accessed: 2023.11.02.

t=1:

$$x_{t-1} = \frac{1}{\sqrt{\alpha}_t}(x_t - \frac{\beta_t}{\sqrt{1-\hat{\alpha}_t}}\epsilon_\theta(x_t, t))$$

To aid in better grasping the topic, supplementary Python code snippets have been included in the appendix, as referenced in Code Listing 6.

2.4 Classifier Free Guidance

In the field of generative diffusion models, producing specific outputs (controlled generation) is often challenging. Classifier Guidance [27] proposed an initial approach to tackle this problem: train a separate image classifier and use its gradients to guide the image generation process towards the desired output. However, this strategy incurs a significant computational overhead, since it requires training an extra model.

Classifier Free Guidance (CFG) [28] offers a more streamlined solution: without the need for supplementary networks, it enables precise image synthesis via a process known as conditioning. It circumvents the need for an additional image classifier by jointly optimizing a single neural network for dual tasks simultaneously. In this context, an unconditional model generates output based solely on the learned data distribution, without specific conditions or classes. In contrast, a conditional model produces output based on both the learned data distribution and specific conditioning. This conditioning can be integrated through a minimal architectural modification. An extra conditioning parameter is passed to the network as well - which may be drawn for varying fields, further explained at Section 2.5 `Guiding Techniques`.

Central to CFG is the bridging between the outputs of the conditional and unconditional models, a concept captured in the CFG paper's equation 6. This extrapolation allows for a seamless blend between the two modes, enhancing the model's versatility. The model is parameterized as $p_\theta(z|c)$, leveraging the same score estimator but incorporating the identifier c as a component of its input. For unconditional updates, c is set to 'None'. By randomly assigning c to the unconditional class identifier during the training phase, CFG concurrently masters the creation of both generalized samples of the distribution and controlled outputs. This duality empowers precise image synthesis with reduced computational needs.

A pivotal element in this process is the guidance weight. It modulates the balance between the conditional and unconditional outputs, ensuring the generated output aligns with the desired condition while maintaining authenticity. However, it is worth noting a potential drawback of CFG: during the inference step, two forward passes through the network are required (one unconditional and one conditional pass), which may have implications on efficiency and processing speed.

2.5 Guiding Techniques

Diffusion models offer various ways to control their outputs.

One of the most versatile and intuitive methods for guidance is using textual descriptions. With models like OpenAI's CLIP [29], there is an effective fusion between vision and language models. CLIP, for instance, can be used to guide the generative diffusion models by providing textual prompts. These prompts can vary from simple attributes like 'sunset' to complex descriptions like 'a tranquil beach during sunset with children playing.' Such models essentially learn the intricate relationship between visual and textual data, enabling more descriptive and customized image generation.

As previously touched upon in CFG, using class labels is another predominant method for guiding image synthesis. By associating a specific label with the image data during training, the model can then generate images corresponding to that class upon request. For instance, a model trained with labels like 'cat' or 'dog' can produce images of cats or dogs respectively when provided with the single class label.

Rather than using text or class labels, some techniques opt to apply another image as a source of inspiration or reference. In this method, an input image or a portion of an image is provided to the model, which then modifies, enhances, or recreates based on the learned distribution and the given reference. This technique is particularly useful in tasks like style transfer, image-to-image translation, or even in scenarios where the desired output is a variation of an existing image.

Low-resolution images may also be used as conditioning inputs, especially when the goal is to produce upscaled or 'super-resolution' [30] versions of the images. These low-resolution images serve as references during the generation process, aiding the model in understanding the fundamental structures and patterns of the original content. By using them as a baseline, the model is guided to enhance and refine details, ultimately resulting in a high-resolution image that retains the essence of the original while boasting superior clarity and definition.

Another promising technique for guiding the outputs of diffusion models is through semantic segmentation, referred to as 'semseg-mask conditioning'. While the foundational concept of using semseg (semantic segmentation) masks to guide generative models might have precedence in literature [31], its application in the automotive-specific domain remains largely unexplored. Moreover, the methods for incorporating such masks can vary significantly, and a standardized architecture has not yet been established. Hence, experimentation in this field remains both relevant and warranted.

Chapter 3

Goal

While current state-of-the-art models such as Stable Diffusion XL[1] and Midjourney v5[2] are adept at text-driven guidance, their proficiency wanes when it comes to precise object positioning — a critical factor in domains such as autonomous mobility. The need for generating detailed and accurate traffic scenes for deep learning algorithms in these applications demands a higher degree of control than what broad-spectrum generative models can offer.

The goal of this thesis is to address this challenge, via introducing a semantic segmentation mask-guided model, specifically trained for automotive data generation. The model should not only grasp the intricacies of vehicular scenes, but also provide a precision level that is challenging for generic generative models to match.

The transformative potential of the method should be demonstrated during a systematic series of experiments. A U-Net [32] based architecture will be implemented and extended to enable control via semantic segmentation masks. A public traffic participant dataset will be acquired and preprocessed. The model will be trained and its utilization possibilites will be showcased during multiple examples: 1) image generation with semantic segmentation mask-guidance 2) image generation with manually modified masks 3) image generation via hand-made drawings. All results shall be evaluated both visually and metrically.

Subsequent chapters will detail the essential steps to achieve the declared objectives, encompassing dataset preparation, baseline training, model design with guidance integration, and an in-depth evaluation of the results.

[1] `https://stability.ai/stable-diffusion` accessed: 2023.10.29.
[2] `https://docs.midjourney.com/docs/model-versions` accessed: 2023.10.29.

Chapter 4

Methods and implementation

Although this work builds around the semantic segmentation guidance, a simpler baseline was established in the early phases. As a result, this yielded two implementations, an unconditional model, that does not incorporate any guidance, only generating novel images from the data distribution, and the one that can be guided via semantic segmentation masks.

The pipeline on Figure 6 could be followed during both methods.

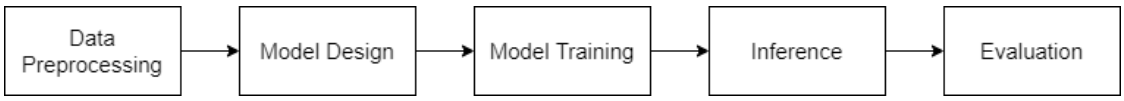


Figure 6: Implementation pipeline

4.1 Dataset

Having a comprehensive dataset establishes a base for any Deep Learning experiment. Berkeley Deep Drive [33] is a public, diverse large-scale urban dataset, in which 100K driving videos were collected from more than 50K rides, resulting in more than 100 million frames in total. Annotations are available for several tasks, such as lane detection, object detection, and for the main motive of my work, semantic segmentation as well.

There were a total of 8000 semantic segmentation annotations - stored as polygons in .json files - that needed to be processed for the current use-case. Also, the original images have a resolution of 1280x720, which is unfeasible for model training. To resolve these issues, 720x720 center-crops were created and downscaled to 128x128 resolution, while the json annotations have been also processed and saved as 128x128 semseg maps, supplied with a colorbook containing class-color mappings for all 19 classes, which are: bicycle, motorcycle, train, bus, truck, car, rider, person, sky, terrain, vegetation, traffic sign, traffic light, pole, fence, wall, building, sidewalk and background. Via these mappings, masks can be modified or even novel ones can be created.

At the end of the 'Data Preprocessing' step, 7000 training images with corresponding masks were ready for training, while 1000 pairs were left untouched for testing purposes.

Samples from the dataset (center-cropped and resized to 128x128) are presented on Figure 7. Even on this small subset, it can be observed how diverse the dataset is.

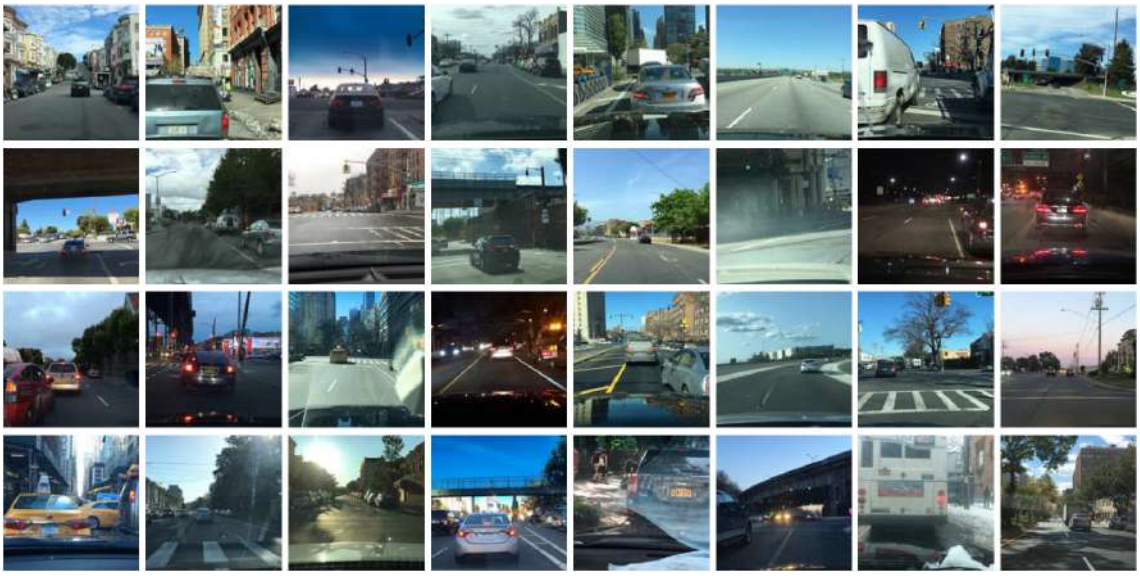


Figure 7: BDD samples

The colorbook presented on Figure 8 may be used for altering already existing maps or creating new ones. It is also stored as a json file containing explicitly the classes and corresponding RGB values. Since these values are read runtime, mappings can be even modified if necessary.

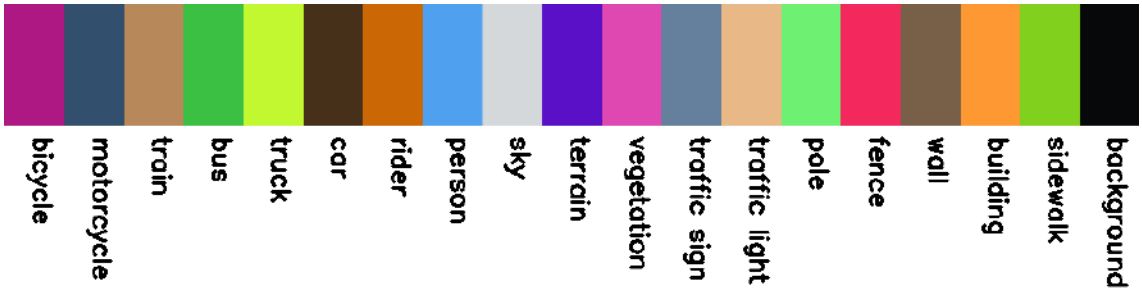


Figure 8: Generated colorbook

Finally, a concrete example of an image-semseg pair is showcased on Figure 9. It has to be noted that during data loading, the colors are mapped to classes which are then one-hot encoded[1] although this will be addressed further in a later section.

[1]https://medium.com/hackernoon/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008f accessed: 2023.10.30

Figure 9: Example of image and corresponding semseg mapping

For the sake of transparency, see the dataset summary on Table 1.

Table 1: Summary of used datasets

name	set	resolution	number of samples	number of classes
BDD	train	128x128	7000	19
BDD	test	128x128	1000	19

4.2 Model Design

Typically a U-Net is used for diffusion models. This is what I applied as well, although I had to strive for a relatively low computational cost architecture, therefore I ommited the Cross-Attention [34] modules which are present in a standard implementation. My architecture (see Figure 10) consisted of 3 downscaling blocks [downscale, DoubleConv, DoubleConv], followed by 3 bottlenecks [DoubleConv] and finally 3 upscaling blocks [upscale, DoubleConv, DoubleConv]. DoubleConv layers are defined as [Conv2d, GroupNorm, GELU, Conv2d, GroupNorm] [35][36]. Timestep is integrated during the forward call via Transformer sinusoidal position embedding [34]. It is passed to the following blocks through an embedding layer[SILU, Linear] [37] : down1, down2, down3, up1, up2 and up3. Up to this point this model is identical to a model designed for unconditional training.

The incorporation of the semantic segmentation mask is the feature that makes my implementation unique. The masks are one-hot encoded, therefore creating a more meaningful format for the neural network. When having 19 classes, the shape of an input tensor is Bx19x128x128 (following a [B,C,H,W] order).

The semantic segmentation mask is passed through a simple DoubleConv block ([Conv2d, GroupNorm, GELU, Conv2d, GroupNorm]) - with having 'num_classes' (19) input channels, and 128 output channels. The original input image is passed through a similar layer, but with having 3 input and 128 output channels. It was essential to match the output channel numbers, thus this way feature values can be **added** when a conditional training is in effect. Concatenation would have not worked, since Classifier Free Guidance takes un-

conditional steps as well, when None labels are passed, a channel number mismatch would emerge. Without any Attention blocks, the total number of parameters was 88,842,371.

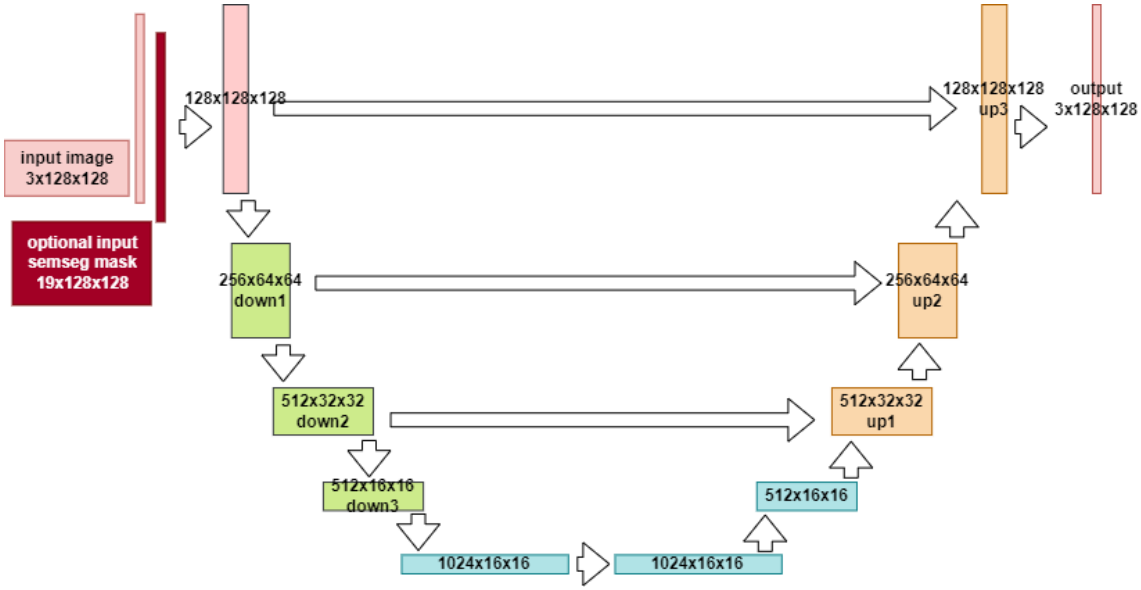


Figure 10: The utilized U-Net-like architecture

4.3 Training

4.3.1 Hardware and software environment

GPU power is indispensable for sufficient training. A Docker container, running on a DGX station containing 4 NVIDIA V100 cards (used 1) was provided by the university. Access was ensured through SSH-connection.

The exact software and hardware setup:

- System: Ubuntu 18.04.6 LTS
- CPUs: 40 pcs Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz
- GPUs: 4pcs Tesla V100-DGXS-32GB (1 used)
- CUDA version: 11.7
- Memory: 257866 MB

VS Code served as a development environment, which bridged the gap between the cluster and my local machine. A conda environment was created for the installations, to which the packages neccessary for Diffusion were added.

The code was prepared in Python language, heavily relying on the PyTorch library. Wandb was used as a logger tool, which enabled to continuously monitor the progress of the time-

taking diffusion trainings. It provided insight about the training loss (Figure 11) and also about the current visual capabilities of the model (Figure 12).

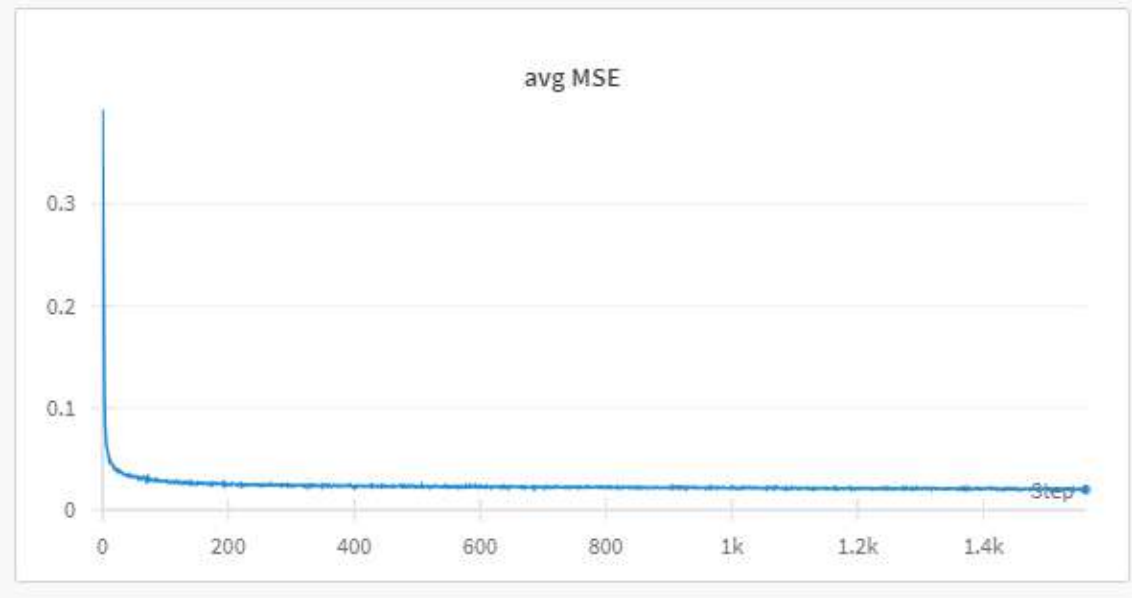


Figure 11: The MSE loss did not decrease significantly after after 200 epochs, but visual results turned out to be more satisfying at later iterations

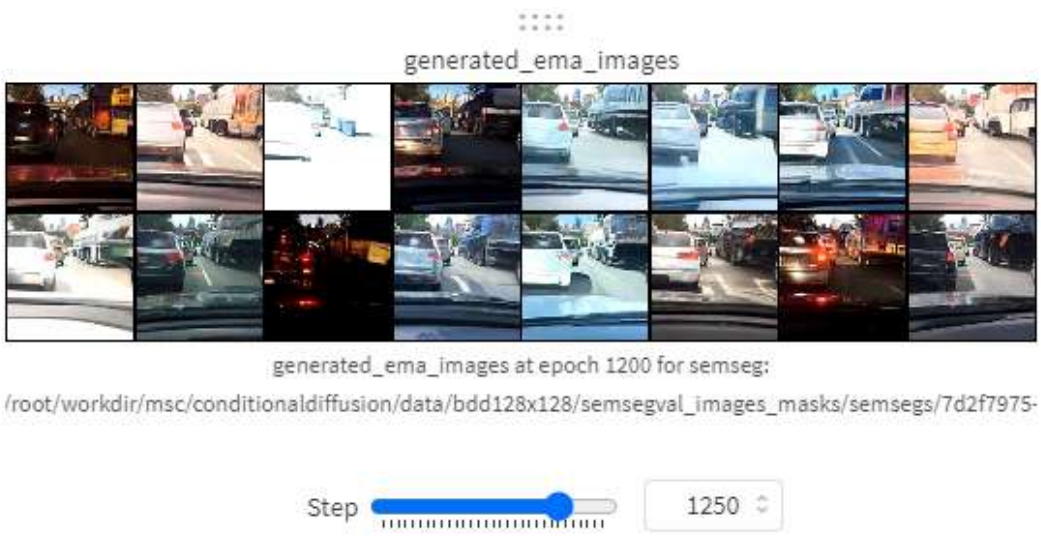


Figure 12: A batch of generated images at a given step of the training process

4.3.2 Hyperparameters

For the training dataset, containing 7000 128x128 images, trainings took approximately 7 days, with a batch-size of 64 and a total epochs of 1300 on a single NVIDIA V100 GPU.

AdamW was used as an optimizer and a learning rate of 0.0003 was set. Noising steps were set to 1000 for the diffusion, just as suggested in the original paper. Exponential Moving Average (EMA) [38] was also introduced and proved to be more stable then the basic model.

4.4 Inference

After loading the trained model, inference can be issued. Due to the nature of CFG, the trained model can be used in an unconditional and conditional manner as well. In order to test the results of the semseg-guidance experiment, masks need to be passed as conditioning. In my implementation, this can be done effortlessly via providing a mask path in a yaml configuration file. The only crucial rule is to use the colors of the colorbook (whose path is also defined in the yaml) and provide a 128x128 semseg mask. The dataloader handles the class mapping and one hot encoding, finally the iterative denoising process will yield novel images with the help of the trained model. Denoising a single sample, with conditioning enabled, takes approximately 1 minute and 15 seconds. Due to batching, the processing time is not directly proportional to the number of samples, acquiring multiple samples at the same time is more efficient.

Inference time was measured for multiple batch sizes, see them at Table 2.

Table 2: Inference times for different batch sizes (1000 steps)

samples	**conditional time (mm:ss)**	**unconditional time (mm:ss)**
1	01:15	01:11
2	01:35	01:31
4	02:19	02:15
8	03:47	03:41
16	04:58	04:48
32	08:58	08:42
64	21:12	20:47

Chapter 5

Results

For testing purposes, 1000 image-semseg pairs were set aside. Utilizing these segmentation masks, 1000 novel images were generated via the trained mask-guided model. These serve as a basis for evaluation. Although an additional unconditional baseline model was trained, results and evaluation focuses on the semantic segmentation-guided model, as it encompasses all features and advantages of the unconditional model as well.

5.1 Metric-based Evaluation

Although visual validation in this field is compelling, metric based evaluation is indispensable as well. In this section multiple possibilities will be addressed for a comprehensive view. SSIM, FID and KID are the standard metrics in the realm of generative models, therefore investigating their yielded results was expected.

5.1.1 SSIM

The Structural Similarity Index (SSIM) [39] is a metric designed to gauge the perceived quality of an image when compared to an original reference image. The SSIM is based on three comparison measurements: luminance, contrast, and structure. Its formula is defined as:

$$\mathrm{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{5.1}$$

Where:

- x and y are the two images being compared.
- μ_x is the average of image x.
- μ_y is the average of image y.

- σ_x^2 is the variance of image x.
- σ_y^2 is the variance of image y.
- σ_{xy} is the covariance of x and y.
- c_1 and c_2 are constants given by:

$$c_1 = (k_1 L)^2 \tag{5.2}$$
$$c_2 = (k_2 L)^2 \tag{5.3}$$

Where:

- L is the dynamic range of the pixel-values (usually $2^{\text{number of bits per pixel}} - 1$).
- $k_1 = 0.01$ and $k_2 = 0.03$ are commonly used values.

The SSIM metric it is typically calculated using a sliding Gaussian window of size 11x11, this is what I applied as well. The possible values range from -1 to 1: -1 meaning perfect anti-correlation, 0 indicating no similarities, and 1 perfect similarity.

For all 1000 image and corresponding generations the SSIM was calculated. The overall average was 0.6818, indicating a satisfactory level of similarity. To further investigate the results, I logged the lowest and highest value pairs to separate folders. The saved results made it clear that the lower values were caused by the day-night shifts. This is understandable, since the metric is based upon contrast and luminance, a good structure is not enough to yield high results.

Figure 13: Low SSIM value pair example 1

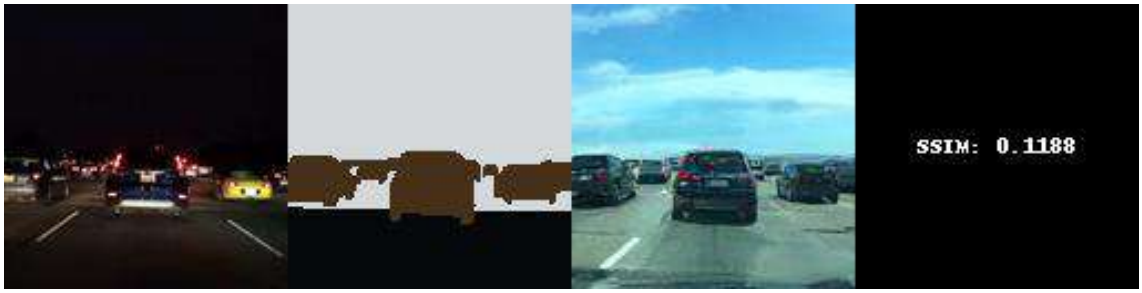


Figure 14: Low SSIM value pair example 2

High value pairs turned out to be more close color-wise. They also did alter the scene, but with introducing rather smaller modifications - like color of a single car.

Figure 15: High SSIM value pair

5.1.2 FID

The Fréchet Inception Distance (FID) [40] is a metric designed to evaluate the quality of generated images by comparing the statistical distribution of features extracted from a pre-trained Inception network [41] (think of these as embeddings). Essentially, it calculates the Fréchet distance between two multivariate Gaussian distributions, one from the generated images and one from real images. A lower FID score indicates that the two sets of images are more similar in terms of their statistics.

Given two sets of images, real and generated, the FID is computed as:

$$\text{FID}(x, g) = \|\mu_x - \mu_g\|^2 + \text{Tr}(\Sigma_x + \Sigma_g - 2(\Sigma_x \Sigma_g)^{0.5}) \tag{5.4}$$

Where:

- x represents the feature vectors of the real images and g represents the feature vectors of the generated images.
- μ_x and μ_g are the means of the feature vectors for the real and generated images, respectively.
- Σ_x and Σ_g are the covariance matrices of the feature vectors for the real and generated images, respectively.
- Tr stands for the trace of a matrix.

FID was calculated by the torchmetrics library[1], using a pre-trained Inception V3[2] network, resulting in an average of 0.4533. It is important to note that lower FID values indicate better image quality and more similarity to the real dataset, but the scale is not strictly linear.

[1] `https://torchmetrics.readthedocs.io/en/stable/image/frechet_inception_distance.html` accessed: 2023.11.02.

[2] `https://pytorch.org/hub/pytorch_vision_inception_v3/` accessed: 2023.10.30.

5.1.3 KID

Kernel Inception Distance (KID) [42] is another metric that provides a measure of the similarity between two sets of images. KID computes the similarity in feature space (also utilizing embeddings from an Inception model). While different kernel functions can be applied, KID is most commonly associated with a polynomial kernel, though variants using other kernels, like Gaussian, exist. One of the key advantages of KID is that it provides an unbiased estimate of the population Maximum Mean Discrepancy (MMD).

Given two sets of images, real and generated, the KID is typically computed as:

$$\mathrm{KID}(x, g) = \mathbb{E}[\kappa(x, x')] + \mathbb{E}[\kappa(g, g')] - 2\mathbb{E}[\kappa(x, g)] \tag{5.5}$$

Where:

- x and x' are independent sets of feature vectors extracted from real images.
- g and g' are independent sets of feature vectors extracted from generated images.
- $\kappa(\cdot, \cdot)$ is the kernel function.

For the polynomial kernel commonly used with KID, the kernel function is defined as:

$$\kappa(a, b) = (a^T b + c)^d \tag{5.6}$$

Where c is a constant, often set to 1, and d is the degree of the polynomial, frequently chosen as 2.

However, if one were to use the Gaussian kernel, it is defined as:

$$\kappa(a, b) = \exp\left(-\frac{\|a - b\|^2}{2\sigma^2}\right) \tag{5.7}$$

Where σ is the kernel width.

It is important to note that when samples come from the same distribution, MMD (and thus KID) is expected to be close to zero, indicating that the two sets of images are similar in the feature space.

I used the KID implementation from the torchmetrics library[3]. Default values were used, except for 'subset_size', which was set as 100. Degree of the polynomial kernel function was 3 and KID value for the dataset was calculated as 0.0081.

[3] `https://torchmetrics.readthedocs.io/en/stable/image/kernel_inception_distance.html`, accessed: 2023.11.01.

5.1.4 Pixel accuracy and IoU

Although SSIM, FID, and KID are standard methods for generative model evaluation, due to the sensitivity of SSIM and FID and for better coverage, a fourth method was intoduced. A pretrained semantic segmentation network was deployed to produce masks for both original and generated images.

Defining P as the correctly classified pixels and T as the total pixels, Pixel Accuracy is given by:

$$\text{Pixel Accuracy} = \frac{P}{T} \tag{5.8}$$

For Intersection over Union (IoU), where I represents the intersection and U the union of the predicted and ground truth segmentations:

$$\text{IoU} = \frac{I}{U} \tag{5.9}$$

Using a pretrained Mask2Former[4] model (trained on Cityscapes[43] at a different resolution), the evaluation on 128x128 resolution BDD images revealed an average IoU of 0.3706 and a notably impressive pixel accuracy of 0.8162. The achieved pixel accuracy means that the model classified over 81% of the pixels correctly, underscoring the diffusion model's ability to authentically reproduce the broader segmentation structures. While the IoU, influenced by resolution and dataset variations, might seem modest, the commendable pixel accuracy provides a testament to the mask-guided model's overall effectiveness in crafting structurally coherent images.

Subsequent sections will present visual evidence further affirming the experiment's success.

5.2 Visual Evaluation

5.2.1 Test masks

In this subsection some examples of the 1000 test images are presented.

Figure 16: Test masks

[4] `https://huggingface.co/facebook/mask2former-swin-large-cityscapes-semantic` accessed: 2023.11.01.

Figure 17: Generated images based on the corresponding input masks on Figure 16

Out of this set of samples, I would point out last picture (bottom row, most right). The human figure is translucent. This is due to the imbalanced data, unfortunately the model was lacking human references. On the rest of the samples it can be seen that the model had no problems with vehicle generation. Further examples at Appendix A.0.30.

5.2.2 Modified test masks

Although there were varying scene setups in the test set, custom controllability can be better showcased via modifying those masks.

An example of the initial mask, the original image, and the generated images before any alteration:

Figure 18: Original image, mask and generated image

After modifying the mask via adding an extra car, the structure of the generation changed as expected, the object was inserted into the appropriate place.

Figure 19: Modified mask and generated image

It should be noted that the lines changed; however, since this was not annotated on the semseg mask, it is a natural behaviour. This observation could pave the way for potential improvements. For instance, a custom parameter might be introduced to adjust the model's tendency to preserve the original color nature of the photo. While this modification offers potential for other use-cases, the current implementation remains a robust tool for data enrichment.

5.2.3 Hand painting

This approach is among the more intriguing ones. Users can create new images by designing hand-drawn masks in a basic image editor, leveraging the established colorbook. A streamlined pipeline has been developed to effortlessly produce images from these handcrafted sketches.

Figure 20: Rough hand painting

While the generated images might lack the intricate details seen in outputs from precise semantic masks, the enhanced user control and hands-on experience make up for it.

5.2.4 Comparison with larger models

While larger 'text2image' based generative models can produce visually stunning results at higher resolutions, guiding them with prompts does not always achieve the desired outcomes.

Consider the following as the baseline image:

Figure 21: Reference image for comparison

An example generated using Midjourney's v5 model[5] is given below. The prompt provided, based on the reference image, was: 'view from inside car, grey car in front, red firetruck front right, high tree front left, tall buildings in the background, realistic –v 5.0'

[5] `https://docs.midjourney.com/docs/model-versions` accessed: 2023.10.30

Figure 22: Generated using Midjourney's v5 model

Despite the impressive quality seen on Figure 22, the generated image does not perfectly reflect the scene setup described in the prompt.

Similarly, results from Stability AI's Stable Diffusion[6] showcased the same challenges. The used prompt was: 'view from inside car, grey car in front, red firetruck front right, high tree front left, tall buildings in the background, realistic'. I also intentionally selected the 'ads-automotive' styling in the options in hope of more specialized results.

Figure 23: Generated using Stability AI's Stable Diffusion

[6] `https://huggingface.co/spaces/stabilityai/stable-diffusion` accessed: 2023.10.30

A human might interpret images differently than an AI model, therefore I conducted an additional experiment. I utilized ChatGPT's[7] new 'image2text' feature, and asked it to generate an input prompt for the 'text2image' generator models.

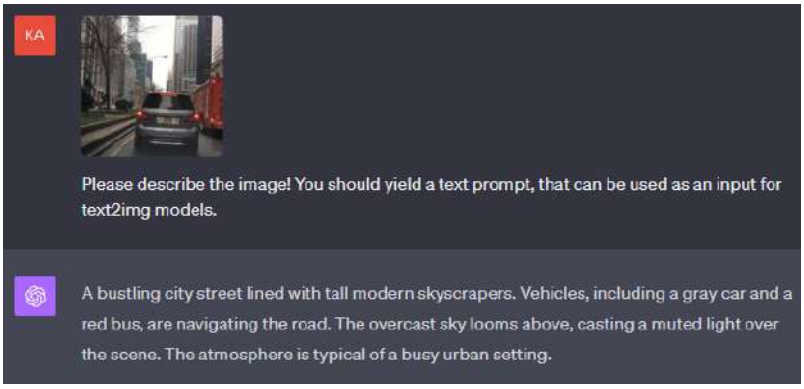


Figure 24: Input prompt generated via ChatGPT

Using the yielded prompt: *'A bustling city street lined with tall modern skyscrapers. Vehicles, including a gray car and a red bus, are navigating the road. The overcast sky looms above, casting a muted light over the scene. The atmosphere is typical of a busy urban setting.'*, image synthesis was attempted again, but as before, could not achieve an identical output:

Figure 25: Generations via AI generated prompts
(Midjourney left, StableDiffusion right)

The examples highlight that relying solely on prompts with large, generalized models does not offer the fine-grained control and accuracy that a specialized semseg-guided model can provide.

[7] `https://chat.openai.com/` accessed: 2023.11.01.

5.2.5 Generating unconditional samples

Due to the nature of Classifier Free Guidance, a segment of the training was carried out without labels, enabling the model to familiarize itself with the intrinsic patterns of the dataset. Samples queried without mask-labels yielded randomised outcomes, exhibiting somewhat more blurriness compared to samples queried from the unconditional baseline model.

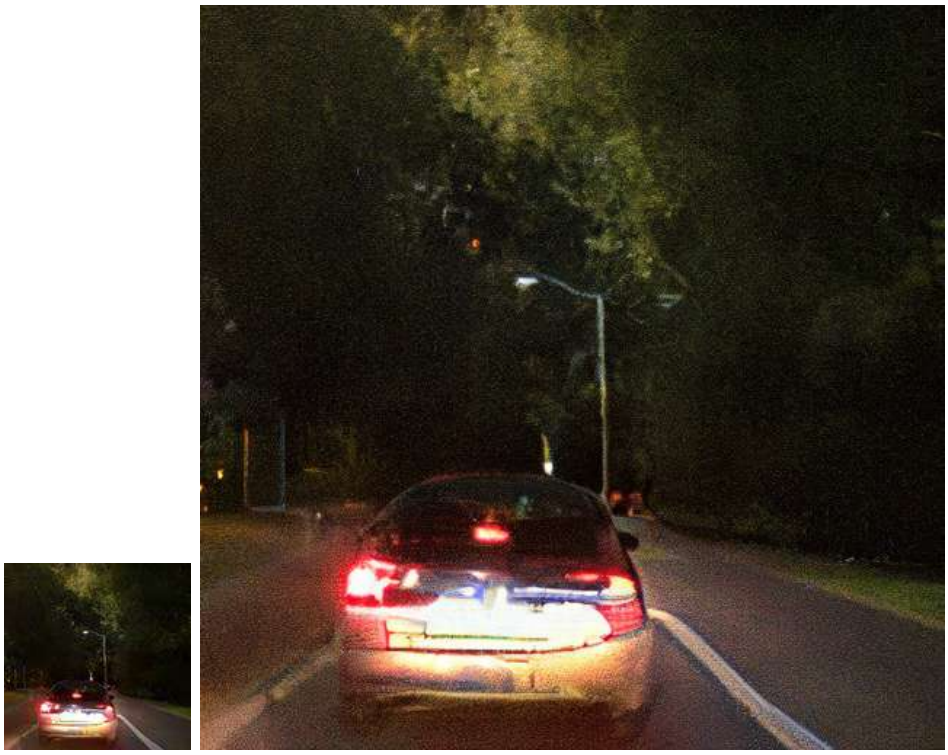
Figure 26: Samples queried without mask-guidance

Despite the unsharp results, it is evident that the model effectively captured the essence of the automotive dataset.

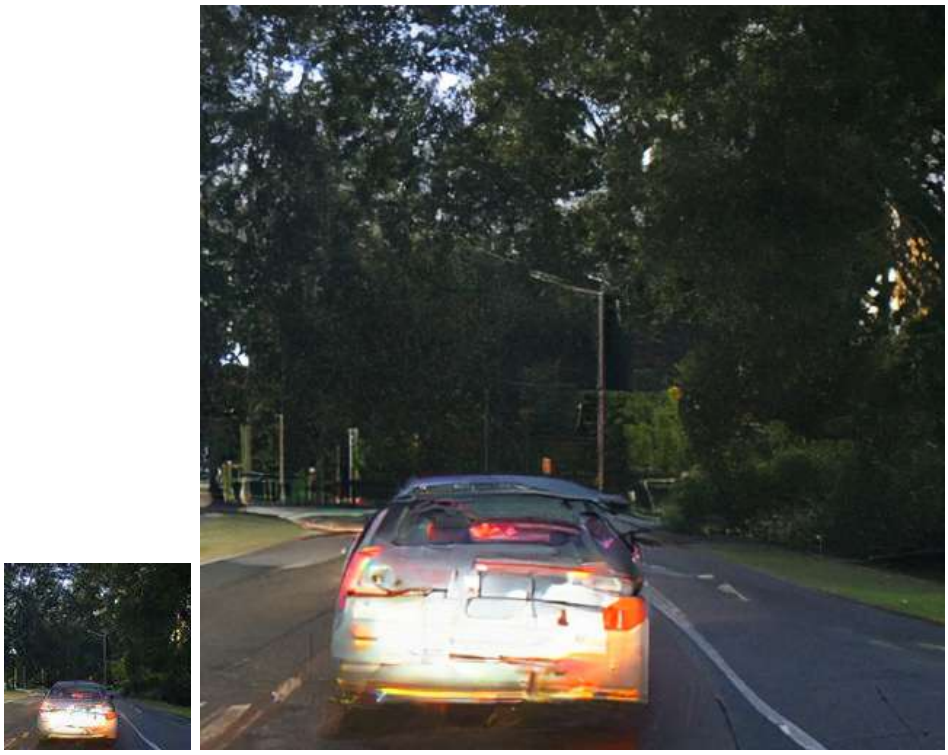
5.2.6 Upscaling results

It is worth highlighting a supplementary experiment conducted during the study. While 128x128 image generations adequately demonstrate a model's capability in capturing the essence of a dataset, human perception often finds higher resolutions aesthetically more pleasing. Directly scaling up the model's architecture is not a viable approach due to the prohibitive computational demands. Instead, leveraging advanced deep learning techniques for this purpose seems more practical. For this experiment, I employed a pretrained super-resolution model[8] to upscale the generated images. The overarching scene composition remained unchanged, but refined details did not scale up accordingly. Potentially, a model explicitly trained on the BDD dataset might offer enhanced outcomes, but exploring that avenue remained beyond the current study's scope.

[8] `https://huggingface.co/docs/diffusers/api/pipelines/stable_diffusion/upscale` accessed: 2023.10.30



Example A



Example B

Figure 27: Synthesized images and their 4x upsampled versions

Figure 27 shows that while the upscaled images retain the overall scene, they lack sharpness and detail in accordance with the resolution. This is likely attributable to the upscaler's training on a different data domain. A diffusion-based upscaler, specifically trained on the

BDD dataset, could potentially enhance even imperfect low-resolution inputs, resulting in more lifelike upscaled imagery.

5.2.7 Limitations

While my model has been adeptly trained to generate objects such as cars, vegetation, buildings, and the sky at arbitrary locations, its performance falters when generating certain objects like human figures. This limitation can be attributed to the insufficient training samples of such classes. As illustrated in Figure 28, the model correctly identifies the spatial position for the human object, but fails to provide an accurate texture representation.

Figure 28: Challenges in human figure generation

To further understand the model's limitations, I conducted an analysis of the class distribution within the training data. Figure 29 provides a visual summary of the pixel-wise occurrence of each class. Utilizing this information, the model's performance can potentially be enhanced by supplementing it with more training samples from underrepresented classes.

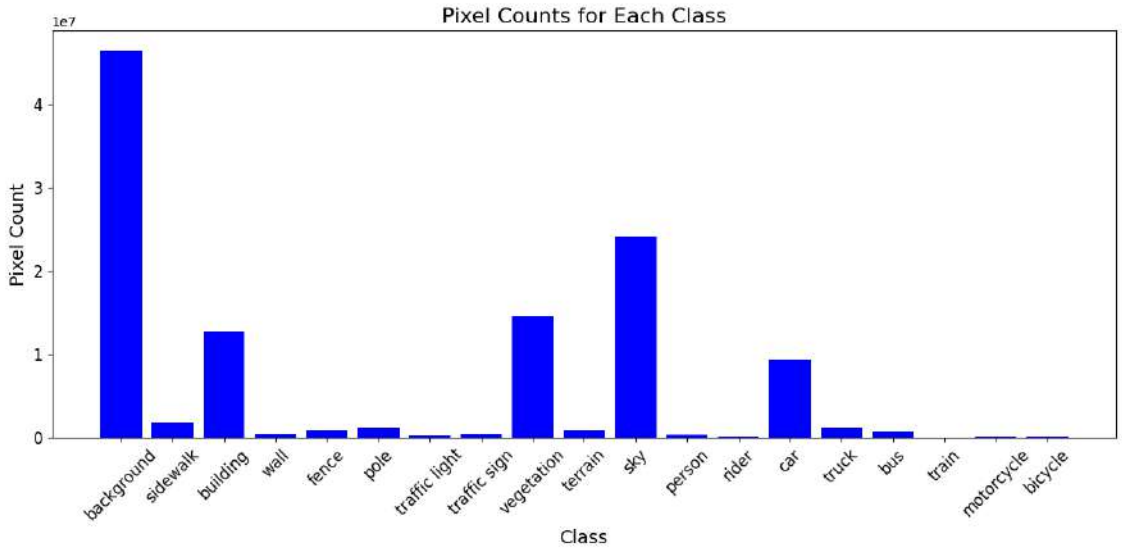


Figure 29: Class distribution analysis in training data

Chapter 6

Summary and future work

Throughout my work this semester, a series of systematically planned steps were undertaken. Datasets and model architecture for two distinct training types were prepared. Semantic segmentation mask-guidance was integrated by modifying a standard U-Net-like architecture. Using a DDPM-based diffusion model, I managed to train an unconditional baseline and also a semantic mask-guided model on a 128x128 resolution. An efficient pipeline was developed, streamlining data processing, training, and testing for each setup.

I have demonstrated that semantic segmentation masks offer clear control over scene generation. This versatility was highlighted through various applications: mask-guided, modified mask-guided, and hand-painted drawing-guided generation. In addition to visual evidence, I conducted multiple metric-based evaluations, affirming the stability of my method. While the resolution of the images generated might not rival that of larger, cutting-edge models, the degree of control surpasses what is typically achieved by mere text prompting. This claim is bolstered by a comparison with leading text-guided models for a specific object-compositioning task.

Looking forward, there are multiple avenues for improvement. One promising direction is incorporating Vector Quantized (VQ) encodings. As highlighted by [18], noising the encoded tokens and directly predicting the denoised versions (instead of noise) could provide both speed and scalability benefits. At the architectural level, while Attention mechanism was previously excluded due to its computational overhead, reintegrating it alongside VQ encodings could enhance generations, without imposing a drastic increase in computational demands. Finally, given the evident potential for visual improvement through upscaling, developing a dedicated diffusion model tailored to the BDD dataset for this specific goal could substantially enhance the quality, yielding more captivating and visually appealing outcomes. These are all intriguing possibilities that I look forward to implementing during the next semester for my final thesis.

Acknowledgement

This thesis has been supported by Continental Automotive Hungary Ltd.

I would like to extend my gratitude to Dr. Bálint Pál Gyires-Tóth and András Béres for their guidance and expertise, which significantly contributed to my work.

Bibliography

[1] A.-S. Maerten and D. Soydaner, "From paintbrush to pixel: A review of deep neural networks in ai-generated art," *arXiv preprint arXiv:2302.10913*, 2023.

[2] I. U. Haq, "An overview of deep learning in medical imaging," 2022.

[3] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[4] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," 2017.

[5] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[6] C. Zhang, C. Zhang, M. Zhang, and I. S. Kweon, "Text-to-image diffusion model in generative ai: A survey," *arXiv preprint arXiv:2303.07909*, 2023.

[7] N. E. Khalifa, M. Loey, and S. Mirjalili, "A comprehensive survey of recent trends in deep learning for digital images augmentation," *Artificial Intelligence Review*, vol. 55, 03 2022.

[8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[9] D. P. Kingma, M. Welling, *et al.*, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019.

[10] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*, pp. 2256–2265, PMLR, 2015.

[11] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," 2016.

[12] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[13] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[14] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

[15] Z. Xiao, K. Kreis, and A. Vahdat, "Tackling the generative learning trilemma with denoising diffusion gans," *arXiv preprint arXiv:2112.07804*, 2021.

[16] S. Azizi, S. Kornblith, C. Saharia, M. Norouzi, and D. J. Fleet, "Synthetic data from diffusion models improves imagenet classification," *arXiv preprint arXiv:2304.08466*, 2023.

[17] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

[18] D. Rampas, P. Pernias, and M. Aubreville, "A novel sampling scheme for text- and image-conditional image synthesis in quantized latent spaces," 2023.

[19] P. Pernias, D. Rampas, M. L. Richter, C. J. Pal, and M. Aubreville, "Wuerstchen: An efficient architecture for large-scale text-to-image diffusion models," 2023.

[20] E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini, and L. V. Gool, "Soft-to-hard vector quantization for end-to-end learning compressible representations," *Advances in neural information processing systems*, vol. 30, 2017.

[21] Z. Xiao, K. Kreis, and A. Vahdat, "Tackling the generative learning trilemma with denoising diffusion gans," *arXiv preprint arXiv:2112.07804*, 2021.

[22] L. Weng, "What are diffusion models?," *lilianweng.github.io*, Jul 2021. Available: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/".

[23] S. Karagiannakos and N. Adaloglou, "Diffusion models: toward state-of-the-art image generation," 2022. Available: https://theaisummer.com/diffusion-models/.

[24] Steins, "Diffusion model clearly explained!," December 2022. Available: https://medium.com/@steinsfu/diffusion-model-clearly-explained-cd331bd411664a56.

[25] Outlier, "Diffusion models | paper explanation | math explained," June 2022. Available: https://youtu.be/HoKDTa5jHvg.

[26] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *International Conference on Machine Learning*, pp. 8162–8171, PMLR, 2021.

[27] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *Advances in neural information processing systems*, vol. 34, pp. 8780–8794, 2021.

[28] J. Ho and T. Salimans, "Classifier-free diffusion guidance," *arXiv preprint arXiv:2207.12598*, 2022.

[29] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, pp. 8748–8763, PMLR, 2021.

[30] W. Yang, X. Zhang, Y. Tian, W. Wang, J.-H. Xue, and Q. Liao, "Deep learning for single image super-resolution: A brief review," *IEEE Transactions on Multimedia*, vol. 21, pp. 3106–3121, dec 2019.

[31] L. Zhang, A. Rao, and M. Agrawala, "Adding conditional control to text-to-image diffusion models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3836–3847, 2023.

[32] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp. 234–241, Springer, 2015.

[33] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," 2020.

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[35] Y. Wu and K. He, "Group normalization. arxiv," *arXiv preprint arXiv:1803.08494*, 2018.

[36] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[37] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *Neural networks*, vol. 107, pp. 3–11, 2018.

[38] D. Busbridge, J. Ramapuram, P. Ablin, T. Likhomanenko, E. G. Dhekane, X. Suau, and R. Webb, "How to scale your ema," *arXiv preprint arXiv:2307.13813*, 2023.

[39] J. Nilsson and T. Akenine-Möller, "Understanding ssim," *ArXiv*, vol. abs/2006.13846, 2020.

[40] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.

[41] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[42] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton, "Demystifying mmd gans," 2021.

[43] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223, 2016.

Appendix

Preparing alpha, alpha hat, beta:

```
def cosine_noise_schedule(self):
    t = torch.linspace(0, 1, self.noise_steps)
    return reversed(self.beta_start + (self.beta_end - self.beta_start)
           * (1 + torch.cos(torch.tensor(np.pi) * t)) / 2)

self.beta = self.cosine_noise_schedule().to(device)
self.alpha = 1. - self.beta
self.alpha_hat = torch.cumprod(self.alpha, dim=0)
```

Noising function (closed form):

```
def noise_images(self, x0, t):
    sqrt_alpha_hat = torch.sqrt(self.alpha_hat[t])[:, None, None, None]
    sqrt_one_minus_alpha_hat = torch.sqrt(1 - self.alpha_hat[t])[:, None, None, None]
    e = torch.randn_like(x0)
    return sqrt_alpha_hat * x0 + sqrt_one_minus_alpha_hat * e, e
```

Algorithm 1 - training:

```
for epoch in range(args.epochs):
    for batch_idx, (images, labels) in enumerate(tqdm(dataloader)):
        images = images.to(device)
        t = diffusion.sample_timesteps(images.shape[0]).to(device)
        x_t, noise = diffusion.noise_images(images, t)
        predicted_noise = model(x_t, t)
        loss = mse(noise, predicted_noise)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

Algorithm 2 - sampling:

```
model.eval()
with torch.no_grad():
    x = torch.randn((n, 3, self.img_size, self.img_size)).to(self.device)
    for i in tqdm(reversed(range(1, steps)), position=0):
        t = (torch.ones(n) * i).long().to(self.device) #timestep.. used for indexing
        predicted_noise = model(x, t)
        alpha = self.alpha[t][:, None, None, None]
        alpha_hat = self.alpha_hat[t][:, None, None, None]
        beta = self.beta[t][:, None, None, None]
        if i > 1: #if not last, add noise
            noise = torch.randn_like(x)
        else:
            noise = torch.zeros_like(x)
        x = 1 / torch.sqrt(alpha) * (x - ((1 - alpha) / (torch.sqrt(1 - alpha_hat))) * predicted_noise)
        + torch.sqrt(beta) * noise

    model.train()
    x = (x.clamp(-1, 1) + 1) / 2
    x = (x * 255).type(torch.uint8)
    return x
```

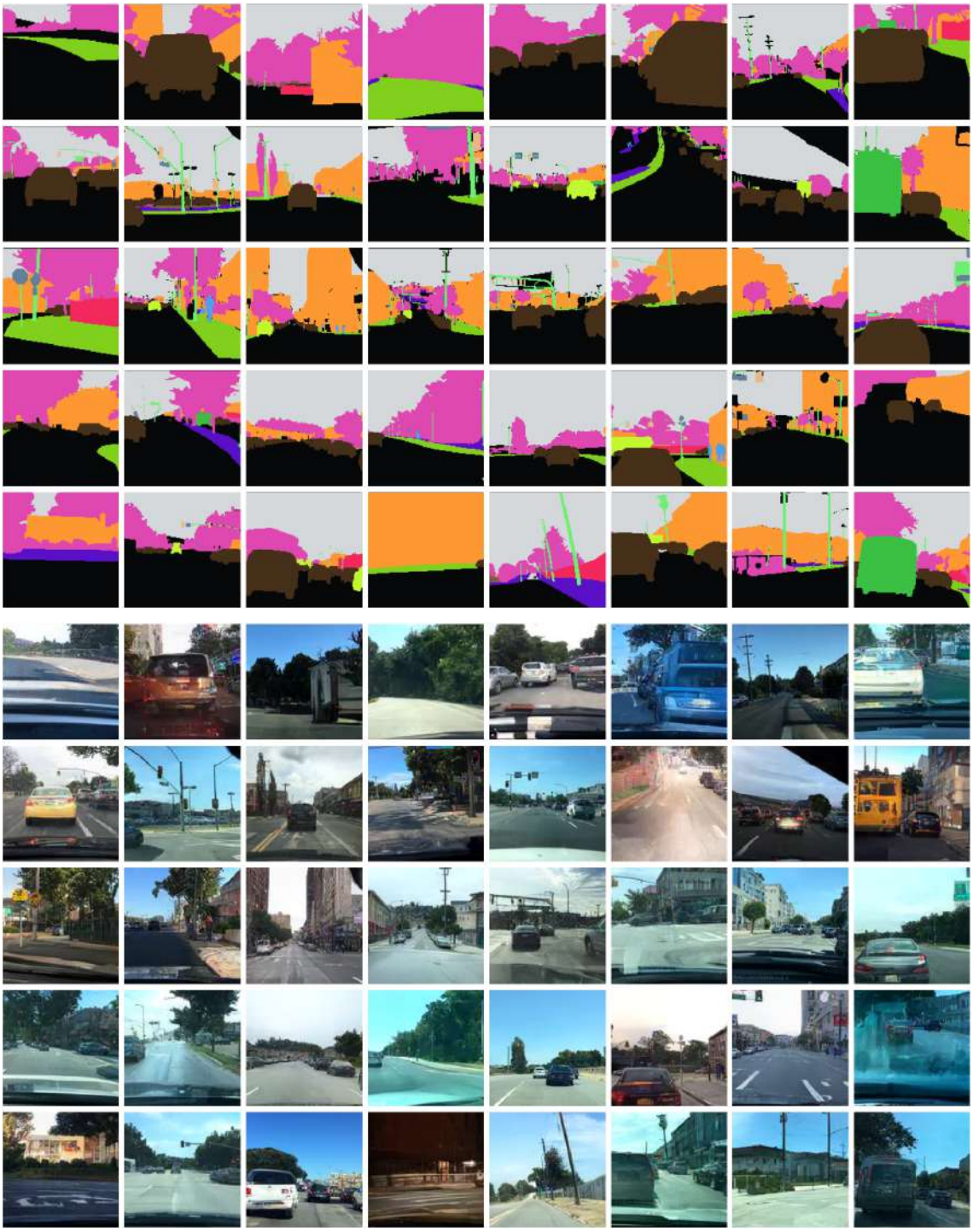


Figure A.0.30: Test masks and corresponding generations