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

June 2, 2024

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, 2024. június 2.

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 szemantikus szegmentációs vezérlésen alapuló generatív modellek alkalmazásával nyújtson megoldást erre a korlátozásra. Ezzel a megközelítéssel tetszőleges önvezető domainbe tartozó jelenetek generálhatóak, ezáltal bővítve a hiányos adathalmazokat és javítva a modellek teljesítményét.

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 applying semantic segmentation mask-guided diffusion models. Through this approach, arbitrary self-driving scene setups can be produced, therefore enriching insufficient datasets and improving the performance of neural networks.

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, therefore also reducing labeling costs. The initial masks may be further modified (e.g. via basic image editing tools), providing even more unique inputs for the model. To illustrate the effectiveness of the method, even handmade drawings may be used as inputs.

In my thesis, I investigate the possibilities of using semantic segmentation mask-guided models, trained specifically for self-driving environment data generation. Two main distinctive approaches are explored: training a model from scratch and utilizing pre-trained large diffusion models. I leveraged the Berkeley Deep Drive dataset [7], which comprises traffic participant frames annotated with semantic segmentation. The outcomes validated the concept of using such masks for scene control, while highlighting the method's potential for scalability. Furthermore, the from-scratch implementation is designed to be compact and minimal, enabling everyday users to explore the generative domain and encouraging them to adapt this approach to address their unique challenges and ideas.[2]

[2]Repository available at `https://github.com/kajc10/semseg-guided-diffusion` (accessed: 2024.05.31.)

Chapter 2

Theoretical Background

2.1 Basic image augmentation methods

Basic data augmentation methods can typically be divided into two principal categories: photometric and geometric augmentations [8].

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; in other words, the mask remains unchanged (see Figure 1). Moreover, special attention must be paid to maintaining the realistic nature of the images; for example, human skin should not appear purple.

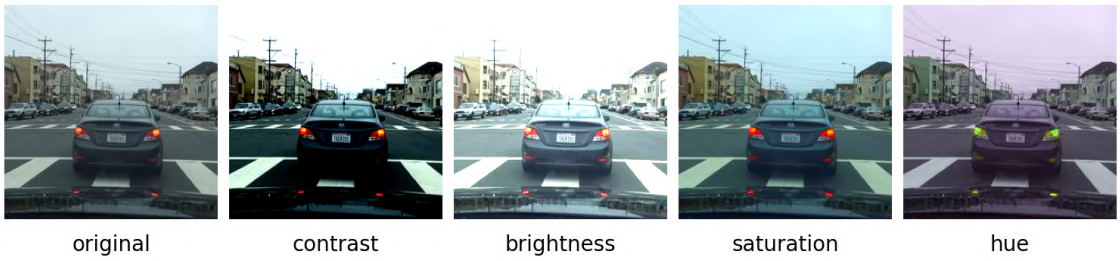


Figure 1: Color augmentation examples

On the other hand, geometric augmentation methods introduce structural changes to the image, thereby altering the composition and layout of the scene. Note that, in parallel, if available, relating segmentation masks have to be modified as well. Some commonly used geometric augmentations are resizing, cropping, flipping (both horizontal and vertical), and rotating (see Figure 2). 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. As before, special care must be taken to preserve the realism of the images, especially during flipping and rotating operations.

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 learning-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 Main types 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) [9], Variational Autoencoders (VAEs) [10], and diffusion models [11].

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 [12]. 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 [9] [13], which could constrain their utility in assignments that demand in-depth data exploration.

2.2.2 Variational Autoencoders

VAEs [14] 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 [15]. 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) [16].

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 [17].

While there have been developments that transition the diffusion process to the latent space of an autoencoder [18] / VQ-autoencoder [19] - commonly termed as 'Stable Diffusion' [20] - using expansive network architectures, and some recent advancements [21]

[22] have even moved away from the classical noise introduction strategy to predicting VQ (Vector Quantization) [23] tokens directly, the introduction in later Section 2.3 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 [16] (see in 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.

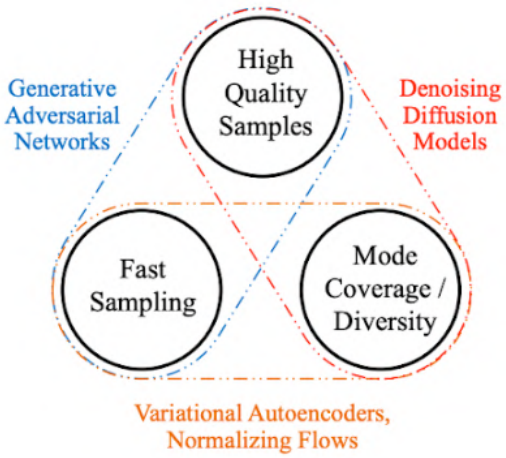


Figure 3: Generative learning trilemma[1]. Source: Figure 1 from 'Tackling the Generative Learning Trilemma with Denoising Diffusion GANs' [24]

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

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

- especially regarding mathematical derivations - the reader is encouraged to consult the original paper and supplementary explanatory blog posts [25][26][27]. I would like to highlight that the following technical summary is processing the content of an excellent video summary [28] about the topic.

DDPM's main approach originates from the concept proposed by the authors of *Deep Unsupervised Learning using Nonequilibrium Thermodynamics* [11], 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.

Although DDPM might initially appear complex, the underlying concept is relatively straightforward. 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 amount of noise present at that step. 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 in Figure 4. The distribution of the noised image can be described as:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \tag{2.1}$$

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

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

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 [29], a cosine 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 $(\beta_t I)$.

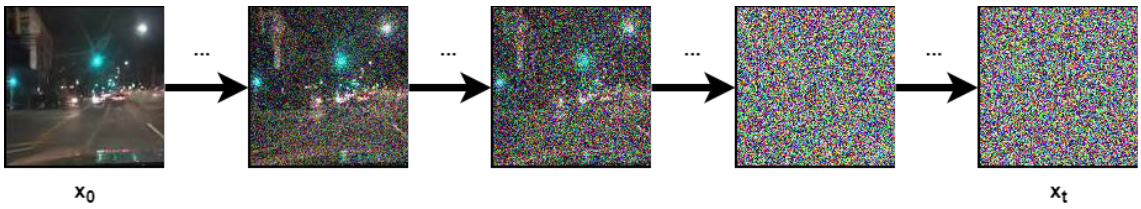


Figure 4: Forward diffusion

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

$$x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\varepsilon \tag{2.2}$$

Fortunately, this can be further derived into a closed-form formula. This way we can directly generate a 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:

$$\begin{aligned} \alpha_t &= 1 - \beta_t \\ \bar{\alpha}_t &= \prod_{i=1}^{t} \alpha_i \\ \varepsilon &\sim \mathcal{N}(0, I) \end{aligned} \tag{2.3}$$

The formula can be written as :

$$\begin{aligned} x_t &= \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\varepsilon \\ &= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\varepsilon \\ &= \sqrt{\alpha_t\alpha_{t-1}\alpha_{t-2}}x_{t-3} + \sqrt{1-\alpha_t\alpha_{t-1}\alpha_{t-2}}\varepsilon \\ &\dots \\ &= \sqrt{\alpha_t\alpha_{t-1}...\alpha_1\alpha_0}x_0 + \sqrt{1-\alpha_t\alpha_{t-1}...\alpha_1\alpha_0}\varepsilon \end{aligned}$$

$$=> \boxed{x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon} \tag{2.4}$$

[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, the Variational Lower Bound[4] is applied. For further mathematical derivation and explanation, see [27]. 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:

$$L_{simple} = E_{t,x_0,\varepsilon}[||\varepsilon - \varepsilon_\theta(x_t, t)||^2] \tag{2.5}$$

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

Figure 5 illustrates this process. The diagram shows how a noisy image (right) is processed through the diffusion model (U-Net) at timestep t to predict the noise, which (a portion of it) is then subtracted to yield a less noisy image (left) at t-1. This iterative process continues until the image is denoised. The arrows in the diagram indicate the flow of images through the model, highlighting the steps of noise prediction, removal and the iterative manner of the process.

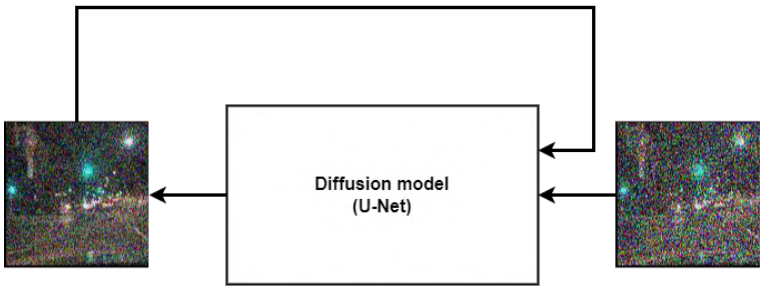


Figure 5: Reverse diffusion process. The right image is the noisy input, the U-Net model predicts the noise, a portion of it is subtracted to obtain a less noisy image on the left. This process is repeated iteratively.

[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

1: **repeat**
2: $x_0 \sim q(x_0)$
3: $t \sim Uniform(\{1, \dots, T\})$
4: $\epsilon \sim \mathcal{N}(0, I)$
5: Take gradient descent step on
$\nabla_\theta ||\epsilon - \epsilon_\theta \left(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, t\right)||^2$
6: **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**. The reformulated equation (2.4) can be applied as described above. Then the model takes x_t and t as input and predicts the noise that was added to the image. 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 at timestep t.

A trained model can be used for generating new samples, using Algorithm 2.

Algorithm 2 Sampling

1: $x_T \sim \mathcal{N}(0, I)$
2: **for** $t = T, \dots, 1$ **do**
3: $z \sim \mathcal{N}(0, I)$ if $t > 1$, else $z = 0$
4: $x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(x_t, t)\right) + \sigma_t z$
5: **end for**
6: **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:

$t > 1$:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, t)\right) + \sqrt{\beta_t}\epsilon \tag{2.6}$$

$t = 1$:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, t)\right) \tag{2.7}$$

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

In the equations above, ϵ is the noise term, α_t and β_t are parameters (pre-defined, not trainable) at timestep t, and $\bar{\alpha}_t$ is a cumulative product of α terms up to t. Equations (2.6) and (2.7) describe the update rules for x_{t-1} depending on the value of t. Equation (2.6) applies when $t > 1$, where extra noise ϵ is added, whereas Equation (2.7) applies when $t = 1$, where no extra noise is added.

For a more comprehensive understanding of the topic, see supplementary Code listings in the Appendix.

2.3.4 Classifier Free Guidance

In the field of generative diffusion models, producing specific outputs (controlled generation) is often challenging. Classifier Guidance [30] 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) [31] 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 2.3.5.

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. See Equation 2.8.

$$\tilde{\epsilon}_\theta(\mathbf{z}_\lambda, \mathbf{c}) = (1 + w)\epsilon_\theta(\mathbf{z}_\lambda, \mathbf{c}) - w\epsilon_\theta(\mathbf{z}_\lambda) \tag{2.8}$$

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, denoted by w. 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.

2.3.5 Guiding methods

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 [32], there is an effective fusion between vision and language models. CLIP can guide generative diffusion models by providing textual prompts. This type of guidance, known as classifier guidance, leverages the model's ability to measure the similarity between an image and a caption. By maximizing this similarity using its gradients, CLIP effectively steers the generative process to produce outputs that align closely with the provided textual description.

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.

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 apply another image as a source of inspiration or reference [33]. In this method, an input image or a portion of an image is provided to the model, which then modifies, enhances, or recreates it based on the learned distribution and the given reference. This technique is particularly useful in tasks like image-to-image translation, where the model can convert an input image from one domain to another. Examples include translating sketches to realistic photos, converting daytime images to nighttime scenes, or generating colorized versions of black-and-white photos.

Low-resolution images may also be used as conditioning inputs, especially when the goal is to produce upscaled or 'super-resolution' [34] 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 masks. While the foundational concept of using semseg (semantic segmentation) masks to guide generative models might have precedence in literature [35], 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.

2.4 Introduction to popular methods

2.4.1 Stable Diffusion

Stable Diffusion (SD) is a state-of-the-art generative model that has garnered significant attention for its ability to generate high-quality images from textual descriptions efficiently. This model is based on a latent diffusion model framework, which innovatively integrates the principles of VAEs and diffusion-based generative processes. Stable Diffusion excels in high-resolution image synthesis while maintaining a lower computational cost compared to other generative models such as GANs.

A key feature of Stable Diffusion is its encoder-decoder framework, which efficiently captures and recreates complex image details. Initially, the model compresses data into a latent representation using a Variational Autoencoder (VAE) [20]. The diffusion process then takes place within this latent space - utilizing a U-Net-based architecture - significantly reducing computational load. Finally, the model expands the latent representation back into the pixel space, creating high-quality imagery.

While Stable Diffusion has demonstrated high performance in both speed and quality, it still has limitations, particularly in controlling specific elements within the generated images. Although the model can generate images that closely align with text prompts, controlling the exact position, orientation, or interaction of objects within the image remains challenging (see Section 2.4.1.1). This limitation is particularly significant in domains requiring precise control, such as autonomous driving, where accurate depiction of traffic scenarios is crucial.

In the context of this thesis, exploring Stable Diffusion provides valuable insights into the capabilities and limitations of current state-of-the-art generative models in producing controlled imagery. By understanding these aspects, challenges associated with generating detailed and precise images - necessary for applications in self-driving environments - can be better addressed.

2.4.1.1 Limitations

The mentioned lack of controllability can be better understood on some minimal samples. Consider Figure 6 as the baseline image. Then see Figure 7 for an example generated using Stability AI's Stable Diffusion[6].

Figure 6: Reference image for comparison

Figure 7: Stability AI's Stable Diffusion - Generated image by prompt: 'view from inside car, grey car in front, red firetruck front right, high tree front left, tall buildings in the background'.

Despite the impressive quality seen in Figure 7, the generated image does not perfectly reflect the scene setup described in the prompt.

A human is likely to interpret images differently than an AI model, therefore an additional experiment can be conducted. Utilizing ChatGPT's[7] 'image2text' feature, it was asked to generate an input prompt for the 'text2image' generator models.

[6] `https://huggingface.co/spaces/stabilityai/stable-diffusion` (accessed: 2023.10.30)
[7] `https://chat.openai.com/` (accessed: 2023.11.01.)

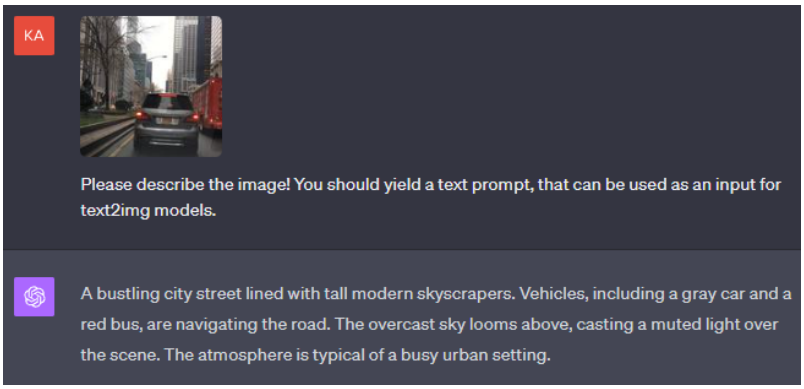


Figure 8: Input prompt generated via ChatGPT

Using the yielded prompt image synthesis was attempted again, but as before, an identical output could not be achieved, see Figure 9.

Figure 9: Image generated by 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.'

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.

2.4.2 ControlNet

ControlNet [35] is an advanced extension designed to enhance the functionality and precision of pre-trained Stable Diffusion models, enabling more directed and specific generative

outcomes. Traditional diffusion models, while powerful, often produce generalized results that lack the ability to incorporate specific directives or conditions. ControlNet addresses this limitation by allowing users to exert precise control over the generated images through new conditional inputs, such as human poses, edges, or depth maps, without modifying the original model, ultimately transforming simple prompts into highly controllable, detailed and structured visuals.

At its core, ControlNet employs a unique approach by creating two versions of a large diffusion model's weights: a 'trainable copy' and a 'locked copy'. The locked copy preserves the network's capabilities learned from analyzing billions of images, while the trainable copy is used to learn conditional control on specific tasks using targeted datasets. These two neural network blocks are connected through 'zero convolution' layers [35] — essentially 1×1 convolutions with both weight and bias initialized to zero. Starting with weights set to zero, they are gradually optimized during training. Moreover, zero convolution avoids introducing new noise to deep features, resulting in faster training times compared to initializing new layers from scratch. See Figure 10 for the building blocks of the composed architecture.

Overall, ControlNet significantly expands the creative possibilities of Stable Diffusion by allowing users to define specific attributes and structures in their generated images, with a shortened training time compared to fine-tuning. Whether for artistic purposes, precise image synthesis, or research applications, ControlNet offers a robust and flexible toolset for enhancing AI-driven image generation. Unlike training from scratch, when fine-tuned, ControlNet can leverage information not only from the training dataset but also from the data Stable Diffusion was originally trained on. Based on these advantages, ControlNet was chosen to be highly utilized in this thesis.

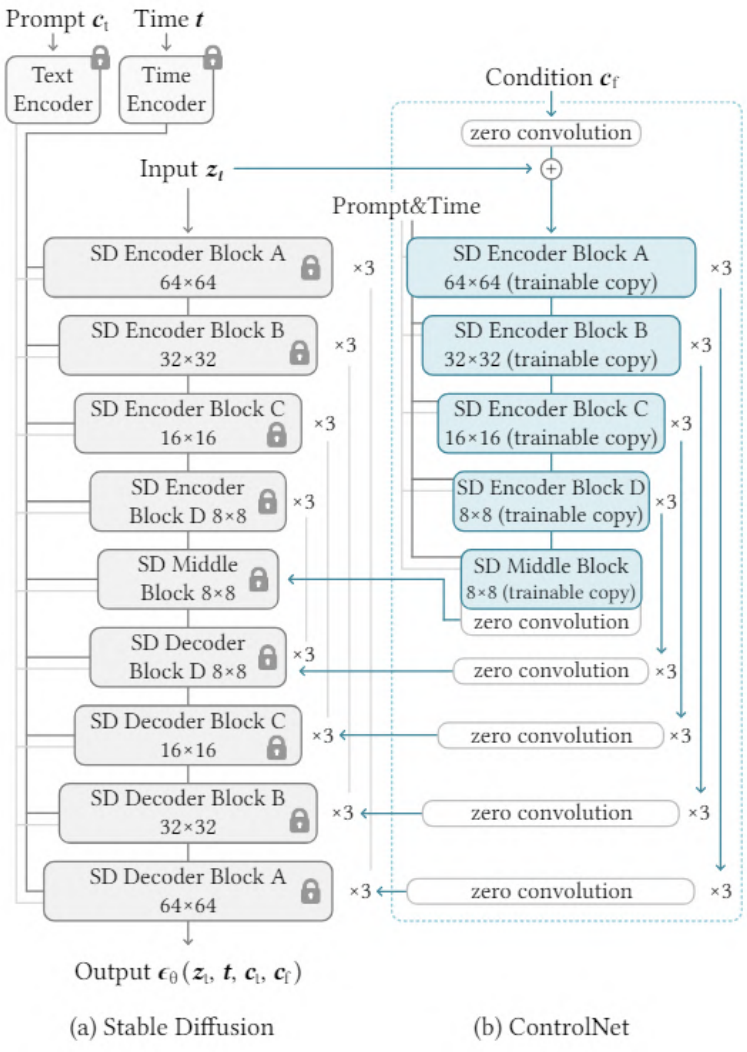


Figure 10: Stable Diffusion's U-Net architecture connected with a ControlNet on the encoder blocks and middle block. The locked, gray blocks show the structure of Stable Diffusion. The trainable blue blocks and the white zero convolution layers are added to build a ControlNet. (Figure 3 from [35])

Chapter 3

Goal

Current state-of-the-art generative models, such as Stable Diffusion XL[1] and Midjourney v5[2], excel at text-driven guidance. However, their proficiency decreases when precise object positioning is required, which is critical in domains such as autonomous mobility. The generation of 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 currently offer.

The goal of this thesis is to address this challenge by applying semantic segmentation mask-guided models, specifically trained for automotive data generation. These models should not only grasp the intricacies of vehicular scenes but also provide a precision level that generic generative models struggle to achieve. The final goal is the creation of a synthetic dataset that effectively helps neural network performance on an Advanced Driver-Assistance System (ADAS) task.

The transformative potential of the proposed method will be demonstrated through a systematic series of experiments. There will be two main approaches:

1. A diffusion model with a U-Net [36] based architecture will be implemented and extended to enable control via semantic segmentation masks.
2. A ControlNet will be trained with mask guidance for a pre-trained Stable Diffusion model.

The two models will be trained and then evaluated both metrically and visually. Their utilization possibilities will be showcased through multiple examples:

1. Image generation with semantic segmentation mask guidance.
2. Image generation with manually modified masks.

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

3. Image generation via hand-made drawings.

Both methods will utilize a preprocessed traffic participant dataset. Novel images will be generated from these two distinct approaches. Generations from the more efficient solution should be used to demonstrate how newly synthesized images can improve a neural network's performance on an ADAS downstream task.

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

Chapter 4

Methods and Implementation

The entire work consists of three main phases: novel image generation, evaluation of generation quality and the assessment of the enriched dataset's effects on an ADAS task.

During the first phase, two different methods were employed for image generation: one was a conventional model implementation and training from scratch, while the other involved training a ControlNet for a pre-trained Stable Diffusion model. An enriched dataset can be constructed by inferring with either solution. See the combined phase 1 and 2 pipeline in Figure 11.

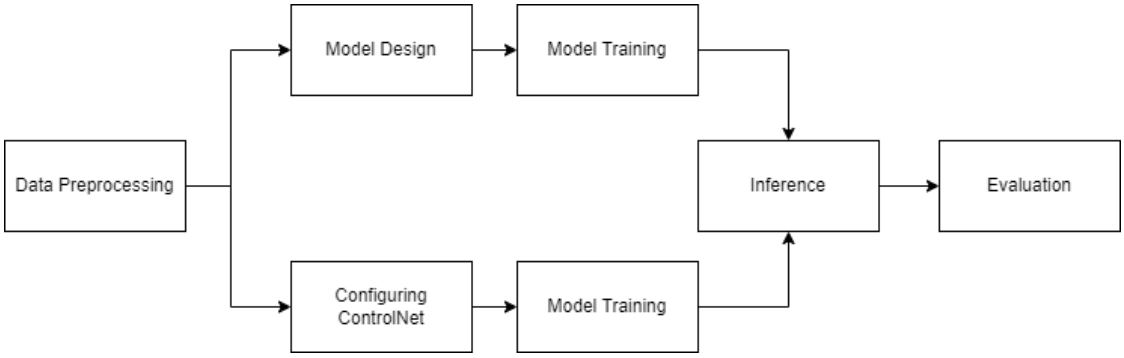


Figure 11: Implementation - combined phase 1 (data generation) and 2 (evaluation) pipeline

In the final third phase, an ADAS task was selected and trained on both a baseline dataset and an enriched one (containing the synthetic images). The pipeline in Figure 12 describes the process.

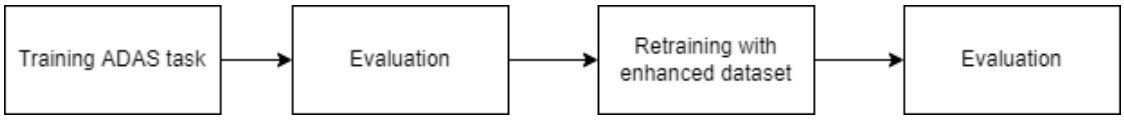


Figure 12: Implementation - phase 3 pipeline

4.1 Dataset

A comprehensive dataset is fundamental for any deep learning experiment. After careful consideration, the Berkeley Deep Drive (BDD) dataset [7] was selected and adjusted to meet the specific requirements of both methods used in this study. The from-scratch model operated only on lower resolution images, whereas the ControlNet could handle higher resolutions.

The BDD dataset is a public, diverse, large-scale urban dataset comprising 100K driving videos collected from more than 50K rides, resulting in over 100 million frames. Annotations are available for several tasks such as lane detection, object detection, and, most importantly for this thesis, semantic segmentation.

For this work, a total of 8,000 semantic segmentation annotations—stored as polygons in .json files—were processed. The original images, with a resolution of 1280x720, were unsuitable for model training in their native form. To address this, 720x720 center-crops were created and downscaled to 128x128 for the from-scratch model and 512x512 for the ControlNet model. The .json annotations were processed into semantic segmentation maps, and a colorbook containing class-color mappings for all 19 classes was created. The classes included are: bicycle, motorcycle, train, bus, truck, car, rider, person, sky, terrain, vegetation, traffic sign, traffic light, pole, fence, wall, building, sidewalk, and background. These mappings allow for the modification or creation of new masks.

After data preprocessing, 7,000 images with corresponding masks were prepared for training, while 1,000 pairs were reserved for testing. Samples from the dataset (center-cropped and resized) are shown in Figure 13. Even within this small subset, the diversity of the dataset is evident.

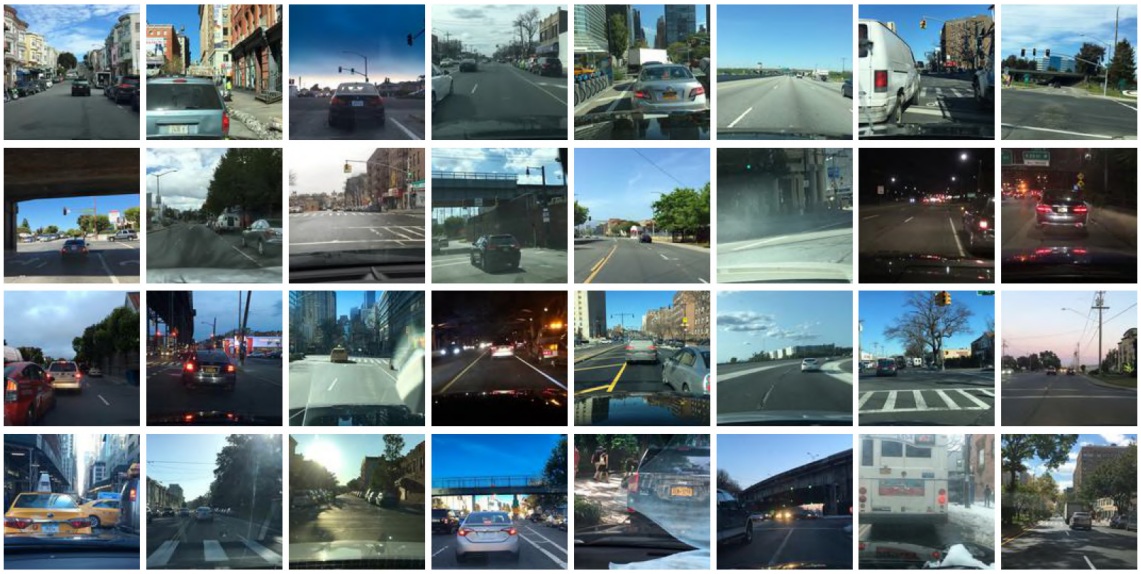


Figure 13: BDD100k samples

The colorbook, shown in Figure 14, is used for altering existing maps or creating new ones. It is stored as a .json file containing explicit class and corresponding RGB values.

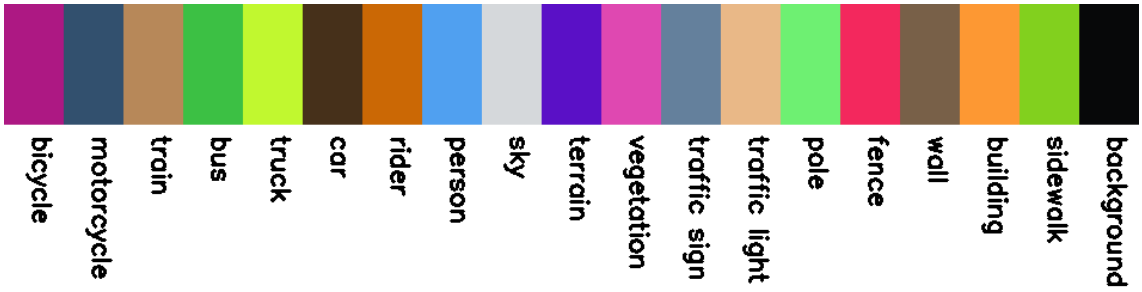


Figure 14: Generated colorbook

An example of an image and its corresponding semantic segmentation map is shown in Figure 15.

Figure 15: Example of image and corresponding semantic segmentation mapping

For transparency, the dataset summary is provided in Table 1.

Table 1: Summary of utilized datasets

Method	Set	Resolution	Number of Samples	Number of Classes
From-scratch	Train	128x128	7000	19
From-scratch	Test	128x128	1000	19
ControlNet	Train	512x512	7000	19
ControlNet	Test	512x512	1000	19

4.2 From-scratch diffusion model

4.2.1 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 omitted most of the Cross-Attention [37] modules which are present in a standard implementation. My architecture (see Figure 16) consisted of 3 downscaling blocks [downscale, Dou-

bleConv, DoubleConv], followed by 3 bottlenecks [DoubleConv], and finally 3 upscaling blocks [upscale, DoubleConv, DoubleConv]. To aid more advanced feature extraction and representation learning, two attention blocks were used in the network—both consisting of multi-head attention and linear layers combined with normalization. DoubleConv layers are defined as [Conv2d, GroupNorm, GELU, Conv2d, GroupNorm] [38] [39].

Timestep is integrated during the forward call via Transformer sinusoidal position embedding [37]. It is passed to the following blocks through an embedding layer [SILU, Linear] [40]: 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, thereby 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 'num_classes' (19) input channels and 128 output channels. The original input image is passed through a similar layer, but with 3 input and 128 output channels. It was essential to match the output channel numbers, so feature values can be **added** when conditional training is in effect. Concatenation would not have worked, since Classifier-Free Guidance takes unconditional steps as well, and when None labels are passed, a channel number mismatch would emerge. With only two Attention blocks, the total number of parameters was 90,816,131.

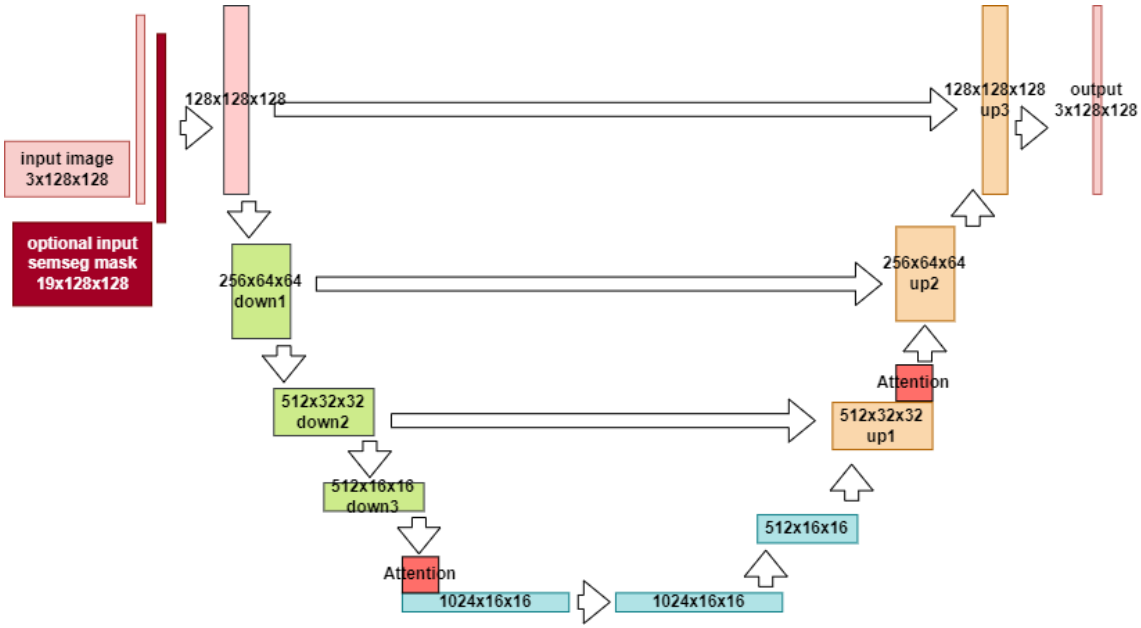


Figure 16: The utilized U-Net-like architecture

4.2.2 Hyperparameters

The model was trained on a dataset of 7,000 128x128 images using a single NVIDIA V100 32 GB GPU over a period of two and a half days with a batch size of 50.

An AdamW [41] optimizer with a learning rate of 0.0003 was employed. The noising steps were set to 1,000 for the diffusion process, as suggested in the original paper. Exponential Moving Average (EMA) [42] was also introduced and proved to be more stable than the basic model. While the training loss plateaued after approximately 300 epochs, indicating minimal gains in traditional loss reduction, the visual fidelity of the results continued to improve, peaking around 550 epochs. This suggests that the model benefits from extended training periods for subtle optimization beyond what is captured by the loss metric alone.

4.2.3 Inference

After loading the trained model, inference can be performed. Due to the nature of Classifier-Free Guidance (CFG), the trained model can be used in both an unconditional and conditional manner. The CFG scale, set to 3, interpolates between the unconditional prediction and the conditional prediction, weighting the conditional prediction three times more heavily, thus enhancing the model's adherence to the given condition while still maintaining some influence from the unconditional prediction. To test the results of the semantic segmentation (semseg) guidance experiment, masks need to be provided as conditioning inputs. In my implementation, this can be done effortlessly by specifying a mask path in a YAML configuration file. The only crucial requirement is to use the colors from the colorbook (whose path is also defined in the YAML file) and provide a 128x128 semseg mask. The dataloader handles the class mapping and one-hot encoding, and finally, the iterative denoising process will generate novel images using 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, making it more efficient to process multiple samples simultaneously.

Inference time was measured for multiple batch sizes, as shown in 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

4.3 Integration of ControlNet

4.3.1 Configuration and training

For integrating ControlNet, a pretrained Stable Diffusion model (specifically sd1.5) was used as the foundation. The ControlNet implementation from the Diffusers GitHub repository[1] was set up by following the detailed instructions provided, which include many configuration options, allowing the entire training process to be tailored according to one's system capabilities.

The necessary dependencies were installed using the requirements.txt file. A Hugging Face dataset was then created, consisting of a CSV file that contained paths to images, corresponding semantic segmentation maps (as conditioning images), and captions labeled as 'Traffic scene.'

The training process was initiated using the Accelerate library[2] with an AdamW8bit[3] optimizer for reduced memory usage. The learning rate was set to 1×10^{-5} to ensure stable training. Specific validation images and prompts were included to evaluate the model during training. The batch size was set to 16, and gradient checkpointing[4] was enabled for a more efficient memory utilization. Training, concluding at 20000 steps, took around 20 hours. Based on visual inspection, checkpoints starting from 6000 steps were acceptable, with 15000 steps being sufficient. Progress was tracked and reported using Weights & Biases (wandb)[5]. Additionally, the training was configured to resume from the latest checkpoint in case of interruptions.

[1] https://github.com/huggingface/diffusers/tree/main/examples/controlnet (accessed: 2024.05.29.)

[2] https://github.com/huggingface/accelerate, https://huggingface.co/docs/accelerate/en/index (accessed: 2024.05.22.)

[3] https://huggingface.co/docs/bitsandbytes/en/optimizers (accessed:2024.05.31.)

[4] https://huggingface.co/docs/transformers/v4.18.0/en/performance (accessed: 2024.05.31.)

[5] https://wandb.ai (accessed: 2024.05.22.)

4.3.2 Inference

For inference, the Diffusers pipeline was used again. The ControlNet model and the Stable Diffusion pipeline were loaded using the paths to the trained models. The UniPCMultistepScheduler[6] [43] - 'a training-free framework designed for fast sampling' - was configured for the inference process. Memory optimizations were also enabled by using 'xformers'[7], which reduces memory usage and computational load through efficient tensor operations, and model CPU offload, which shifts parts of the computation from the GPU to the CPU to balance the load and enhance overall performance.

The inference process involved processing each mask image in the input folder, generating corresponding output images based on a specified prompt ('Traffic scene. Berkeley Deep Drive style, high quality, extremely detailed texture.'), and saving the generated images to the output directory. Compared to the slow inference of the from-scratch model, which required 1000 steps, a single image was generated in just 20 inference steps, taking only 3 seconds. The 14000th checkpoint was selected for image synthesis, with the guidance scale set to 5. Additionally, a manual seed was used to ensure the reproducibility.

4.4 Hardware and software environment

GPU power is indispensable for any 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: 4 pcs 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. Conda environments were created, to which the packages necessary for the particular approach were added.

All codes were prepared in Python language, heavily relying on the PyTorch library[8]. Wandb was used as a logger tool, which enabled to continuously monitor the progress of the time-taking diffusion trainings.

[6] https://huggingface.co/docs/diffusers/en/api/schedulers/unipc (accessed: 2024.05.29.)
[7] https://huggingface.co/docs/diffusers/en/optimization/xformers (accessed: 2024.05.29.)
[8] https://pytorch.org (accessed: 2024.05.22.)

Chapter 5

Results

Besides demonstrating how semantic segmentation mask-guidance offers explicit control, improving an ADAS subtask must be the overall long-term goal. Therefore, a comprehensive evaluation of the from-scratch implementation, ControlNet model, and its impact on ADAS is expected as well. The generative quality of the models is assessed by reserving 1,000 image-semseg pairs from the BDD dataset. These pairs were used to generate novel images via the mask-guided models, which were trained on the remaining dataset. The from-scratch model operates at a resolution of 128x128 pixels, and its inference is relatively slow, while ControlNet operates at a higher resolution of 512x512 pixels and is significantly faster. It should also be noted that ControlNet generations were queried by the prompt 'Traffic scene. Berkeley Deep Drive style, high quality, extremely detailed texture.', without specifying any daytime condition.

The evaluation process included a comparison of the models based on several metrics: SSIM (Structural Similarity Index Measure) [44], FID (Fréchet Inception Distance) [45], KID (Kernel Inception Distance) [46], Pixel Accuracy, and IoU (Intersection over Union). These metrics provided a quantitative measure of the generative quality of the models. Additionally, visual evaluations were conducted to supplement the metric-based assessments, involving the examination of images generated from standard, modified, and hand-painted segmentation masks.

Despite lower metric values, ControlNet demonstrated superior performance in terms of speed, visual evaluations, diversity, and practical application. Moreover, it offers significantly higher resolution, which is crucial for ADAS tasks. Consequently, ControlNet was selected for further analysis in improving the ADAS subtask. Subsequent sections conclude a detailed analysis of generative quality using both metrics and visual assessments, highlighting limitations and examining the impact of the ControlNet images on an ADAS downstream task.

5.1 Objective evaluation

Table 3: Overall metric results for from-scratch and ControlNet models

Model	SSIM	FID	KID	Acc	IoU
From-scratch	0.6588	1.0636	0.0105	0.8170	0.3507
ControlNet	0.6079	1.8791	0.0279	0.7353	0.3116

Although visual assessments provide significant insights into model performance, a robust evaluation must also incorporate metric-based analyses. In this section multiple possibilities are 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 critical for understanding the nuanced differences between the two models. Table 3 provides a summary of the overall metric results for the from-scratch and ControlNet models, which are explained and further assessed in this section.

5.1.1 SSIM

The Structural Similarity Index (SSIM) [44] 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:

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

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:

$$\begin{aligned} c_1 &= (k_1 L)^2 \\ c_2 &= (k_2 L)^2 \end{aligned} \tag{5.2}$$

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 separately for the two models. The overall average for the from-scratch model was 0.6588, and for the ControlNet 0.6079, both indicating a satisfactory level of similarity.

To further investigate the results, the lowest and highest value pairs were logged 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. See Figure 17, 18, 19 and 20, each representing an original image, its corresponding mask, a generated image, and the respective SSIM value. High value pairs turned out to be more close color-wise. They also altered the scene, but by introducing relatively small modifications - like changed the color of a single car, see Figure 21 and 22.

Figure 17: From-scratch model low SSIM value pair example 1

Figure 18: From-scratch model low SSIM value pair example 2

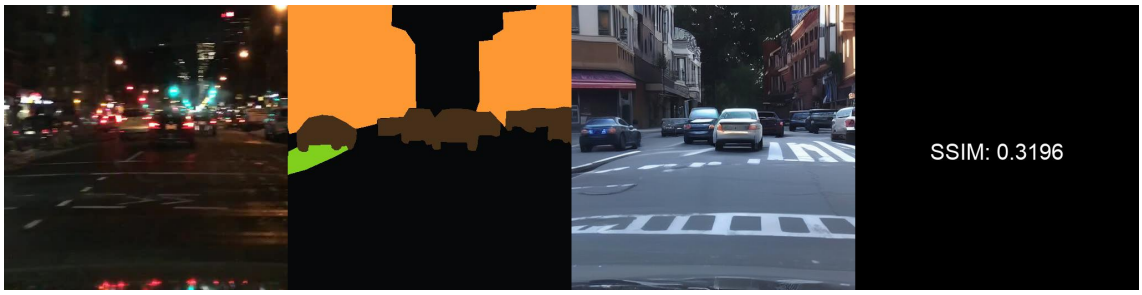


Figure 19: ControlNet model low SSIM value pair example 1

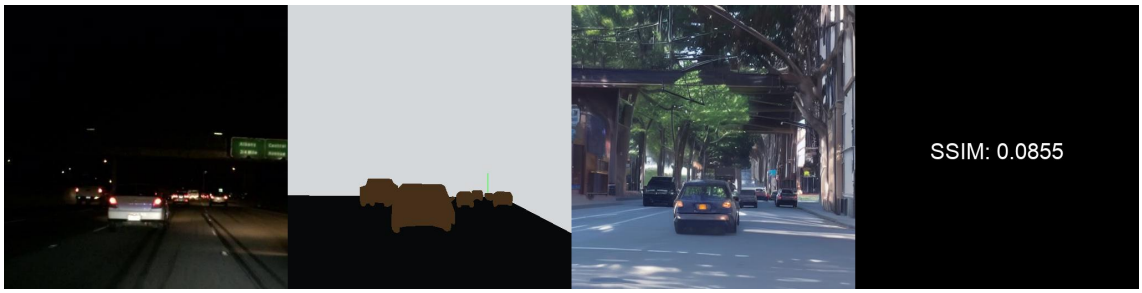


Figure 20: ControlNet model low SSIM value pair example 2

Figure 21: From-scratch model high SSIM value pair

Figure 22: ControlNet model high SSIM value pair

5.1.2 FID

The Fréchet Inception Distance (FID) [45] 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 [47] (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:

$$\text{FID}(x, g) = \|\mu_x - \mu_g\|^2 + \text{Tr}(\Sigma_x + \Sigma_g - 2(\Sigma_x \Sigma_g)^{0.5}) \tag{5.3}$$

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 1.0636 for the from-scratch model and 2.2753 for the ControlNet. 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.

5.1.3 KID

Kernel Inception Distance (KID) [46] 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:

$$\text{KID}(x, g) = \mathbb{E}[\kappa(x, x')] + \mathbb{E}[\kappa(g, g')] - 2\mathbb{E}[\kappa(x, g)] \tag{5.4}$$

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.

[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.)

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

$$\kappa(a, b) = (a^T b + c)^d \tag{5.5}$$

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:

$$\kappa(a, b) = \exp\left(-\frac{\|a - b\|^2}{2\sigma^2}\right) \tag{5.6}$$

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 from-scratch model's dataset was calculated as 0.0105 and 0.0279 for the ControlNet's.

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 introduced. 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:

$$\text{Pixel Accuracy} = \frac{P}{T} \tag{5.7}$$

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

$$\text{IoU} = \frac{I}{U} \tag{5.8}$$

Using a pretrained Mask2Former[4] model (trained on Cityscapes [48] at a different resolution), the evaluation on 128x128 resolution BDD images revealed an average IoU of 0.3507 and a notably impressive pixel accuracy of 0.8170. The achieved pixel accuracy

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

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

means that the model classified over 81% of the pixels correctly, underscoring the diffusion model's ability to authentically reproduce the broader segmentation structures. The same experiment was issued for the 512x512 ControlNet generations, resulting in an average IoU of 0.3116 and an imposing pixel accuracy of 0.7353 While the IoU, influenced by resolution and dataset variations, might seem modest, the commendable pixel accuracy provides a testament to both mask-guided models' overall effectiveness in crafting structurally coherent images.

Subsequent sections will present visual evaluations to further affirm the success of the methods.

5.2 Subjective evaluation

5.2.1 Test masks

Images were synthesized by both models for the 1000 test masks, in this subsection a selection of these examples are presented.

First, the from-scratch generation set is inspected (Figure 24). Notably, the last image (bottom row, far right) shows a translucent human figure. This issue is due to imbalanced data, as the model lacked sufficient human references. Otherwise, the model performed well with vehicle generation in the other samples. Further examples can be found in Appendix A.0.36.

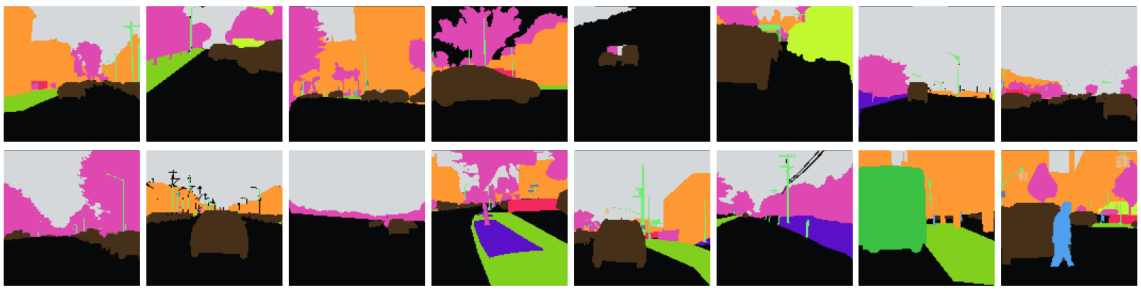


Figure 23: Test masks (128x128 scaling)

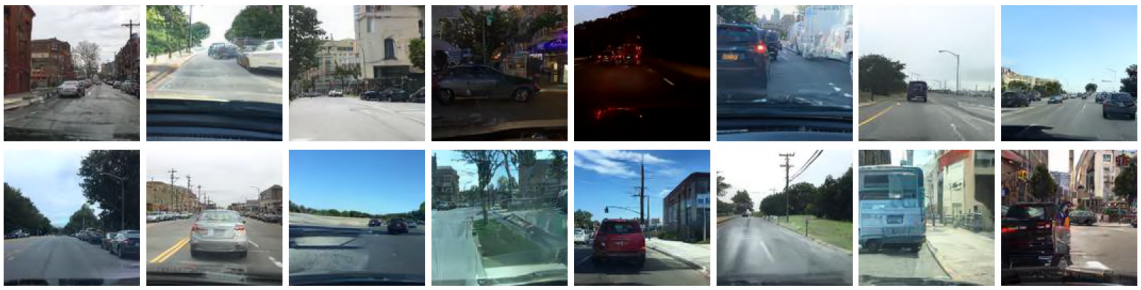


Figure 24: From-scratch model - Image generations based on the corresponding input masks in Figure 23

Images for the ControlNet model were queried in a similar manner, see Figure 25 and Figure 26. An important factor was the checkpoint selection. Earlier checkpoints tended

to be more similar to the BDD domain, but the semantic segmentation control was not fully operational, causing objects not to perfectly fill the masks. Conversely, later checkpoints lost the BDD style and relied solely on the Stable Diffusion domain. A potential solution - besides optimal checkpoint selection and guidance scale setup - could involve the finetuning of the Stable Diffusion model preceding the ControlNet training. This approach could adapt the BDD style before ControlNet training.

Figure 25: Test masks (512x512 scaling)

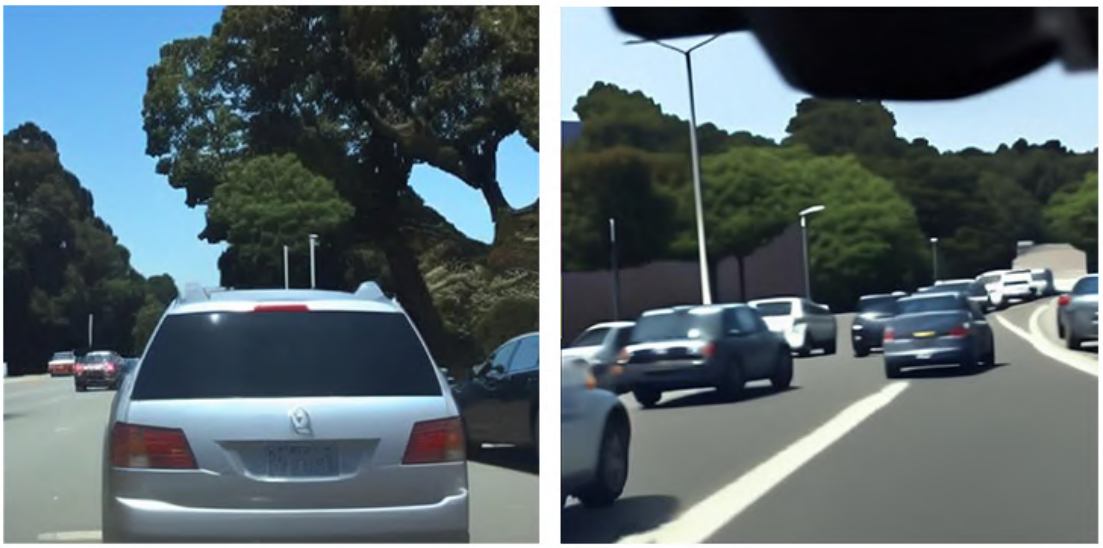


Figure 26: ControlNet model - Image generations based on the corresponding input masks in Figure 25

Further examples can be found in Appendix A.0.37, A.0.38.

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 image (from-scratch model) before any alteration is presented in Figure 27.

Figure 27: Original image, mask and generated image (from-scratch model)

After modifying the mask via adding an extra car, the structure of the generation using the from-scratch model changed as expected, the object was inserted into the appropriate place (Figure 28).

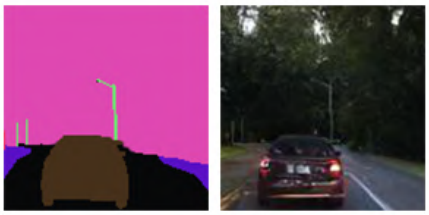


Figure 28: From scratch model - Modified mask and generated image

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.

The experiment was repeated on an 512x512 resolution mask for the ControlNet model. Results were similar, the car was placed to the appropriate position (see Figure 29).

Figure 29: ControlNet - generated image

As a side note, ControlNet's additional prompt-guidance mechanism can specify textural aspects of the scene, which could also be utilized to define lane configurations when necessary.

5.2.3 Hand painting

This approach is particularly intriguing as it enables users to 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. See Figure 30 and Figure 31 for the results.

Figure 30: From-scratch model - Rough hand painting

Figure 31: ControlNet model - Rough hand painting

Although the generated images may not exhibit the intricate details observed in outputs derived from precise semantic masks, the increased user control and interactive experience compensate for this limitation.

5.3 ADAS task performance improvement

Following a thorough evaluation of two distinct approaches, it became clear that while the from-scratch approach performed better metrically, the ControlNet model excelled in terms of resolution, visual quality, speed, ease of integration, and training time. Consequently, due to these advantages, the ControlNet model was selected for further development.

To facilitate training, a synthetic dataset comprising 7,000 novel images was generated using the base prompt 'Traffic scene. Berkeley Deep Drive style, high quality, extremely detailed texture.'

To evaluate the impact of the synthetic dataset on ADAS, a DeepLabv3 [49] semantic segmentation network was trained on the original dataset of 7000 images (training split), followed by retraining with an augmented set that included the 7000 additional synthesized images. These new images enriched the diversity and complexity of the training data. The key focus of this evaluation was to quantify improvements in the semantic segmentation network's performance. Post augmentation, the validation mean IoU of the network improved from 37.5% to 39.2%, demonstrating the effectiveness of using synthesized imagery in enhancing the robustness and adaptability of ADAS systems. I believe that significant

improvements can be achieved in the future by carefully examining the weaknesses of the dataset. By specifically generating images to address class imbalances or by producing an order of magnitude more synthetic images, the training dataset can be substantially augmented.

Chapter 6

Discussion

6.1 Metric results

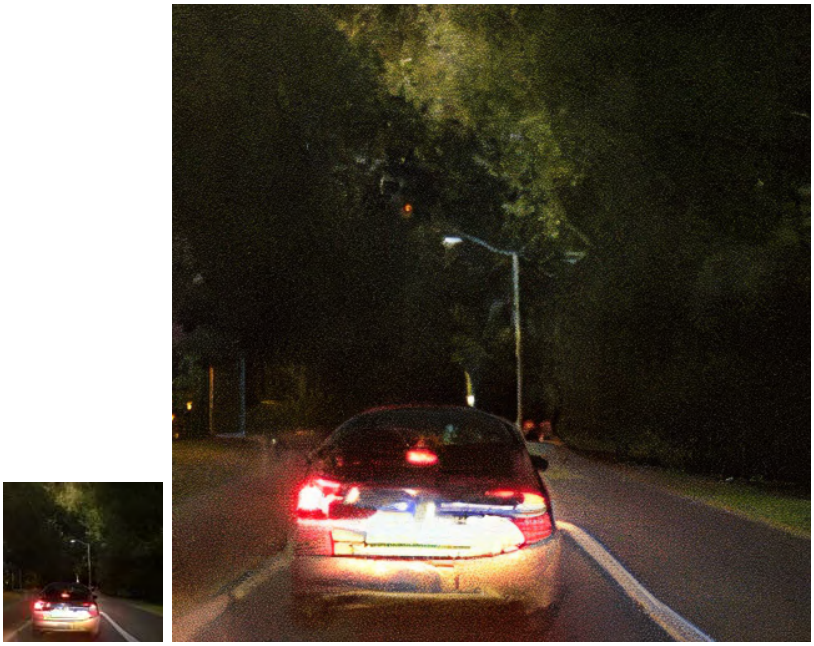
From-scratch models yielded slightly better results. This can be attributed to the fact that the from-scratch model was trained only on the training data, and the sequences were not strictly separated into exclusive train and test splits due to the nature of the original dataset. This issue could be improved by using a dataset where data selection has been performed with great precision, separating it into distinct training and validation semseg sets. Consequently, the model may have overfitted to certain scene setups, accurately guessing the original daytime conditions. On the other hand, ControlNet leverages a larger dataset from the pre-trained Stable Diffusion model, which has been exposed to a diverse range of images. This extensive pre-training allows ControlNet to generalize better to various scenarios. Additionally, its 'Traffic scene. Berkeley Deep Drive style, high quality, extremely detailed texture.' prompt did not provide specific guidance regarding the potential daytime of the generation, therefore the queried images were generated predominantly during the day by default.

6.2 Limitations

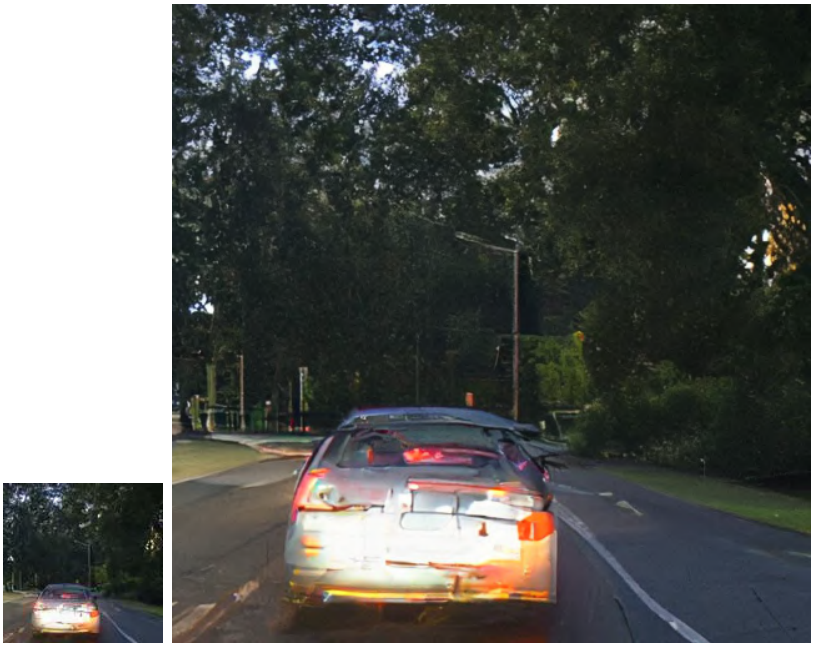
The most conspicuous difference between the two methods is the resolution. An effort was made to upscale the from-scratch model generations and therefore bring closer the two results. Directly scaling up the model's architecture is not a viable approach due to the prohibitive computational demands. Instead, advanced deep learning techniques were leveraged, a pretrained super-resolution model[1] was employed to upscale the generated images. The overarching scene composition remained unchanged, but refined details did not scale up accordingly (see Figure 32). Potentially, a model explicitly trained on the

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

BDD dataset might offer enhanced outcomes, but exploring that avenue remained beyond the current study's scope.



a)



b)

Figure 32: From-scratch model synthesized images and their 4x upsampled versions

Another important aspect is that the ControlNet model is able to generate images via extra prompt guidance, leveraging information also from the pre-training data. The from-scratch model was not armed with such abilities. However, if the checkpoint is not selected carefully, semantic segmentation control might not yet be in full effect, and details can be

lost. When the goal is to produce images that otherwise could not be recorded this is not a limitation, however the most optimal solution is when multiple images can be produced for a single label, therefore saving the costs of re-labeling. Figure 33 demonstrates the production of varying style images from a single mask - leveraging information from the pre-trained Stable Diffusion model - and also points out how some minor details are lost in the background.

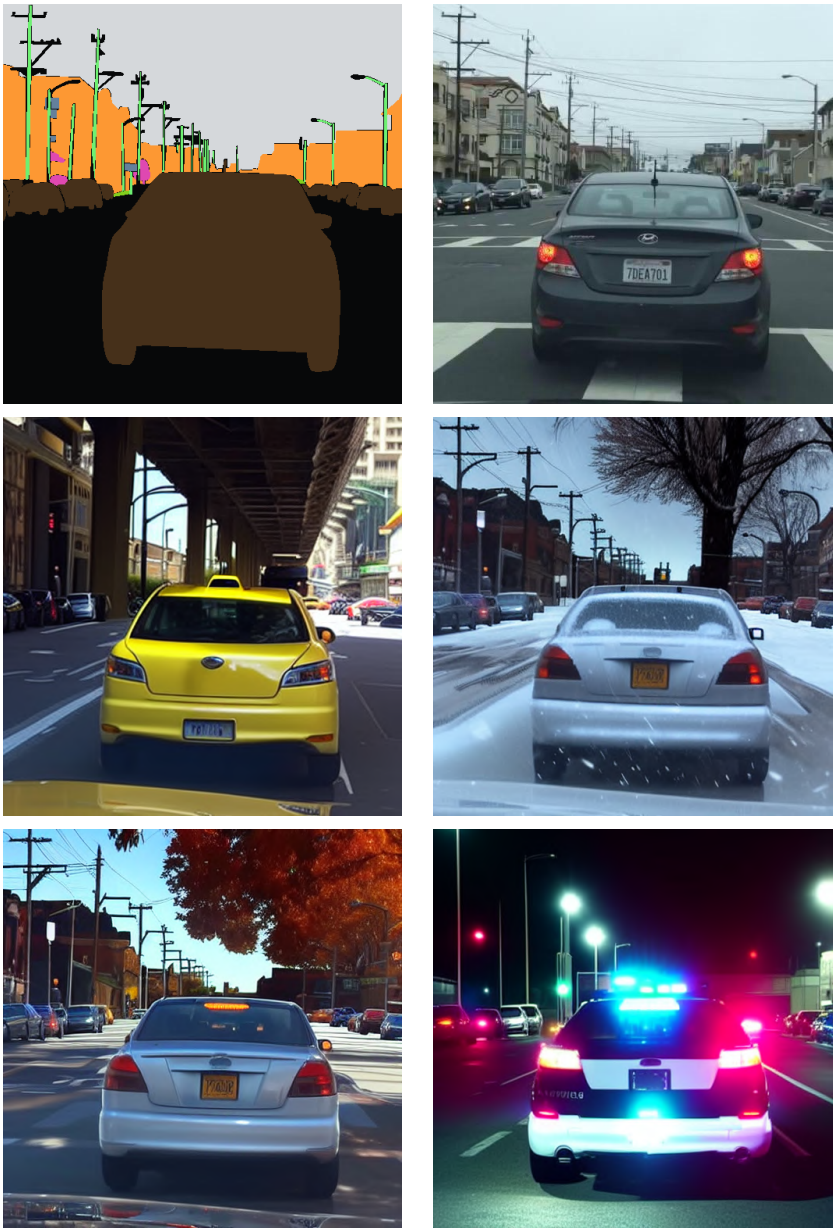


Figure 33: Variations of the original image via the following prompt conditioning: middle-left: 'Yellow taxi. Traffic scene.', middle-right: 'Snowy weather. Traffic scene.', bottom-left: 'Autumn weather. Traffic scene.', and bottom-right: 'Police car. Traffic scene.'

It is also important to address the drawbacks of using extra pre-training data. One significant issue is the occurrence of so-called hallucinations, which are most common

in objects that typically contain textual information in the real world. For example, as shown in Figure 34, meaningless text can sometimes appear. To mitigate this problem, automated quality checking can be employed. This involves using scripts or tools, such as other trained deep learning models, to filter out flawed generations. For instance, a LLaVA [50] model could be used to detect and remove images containing anomalous text. See Figure 35 for such attempt.

Figure 34: ControlNet hallucination example.

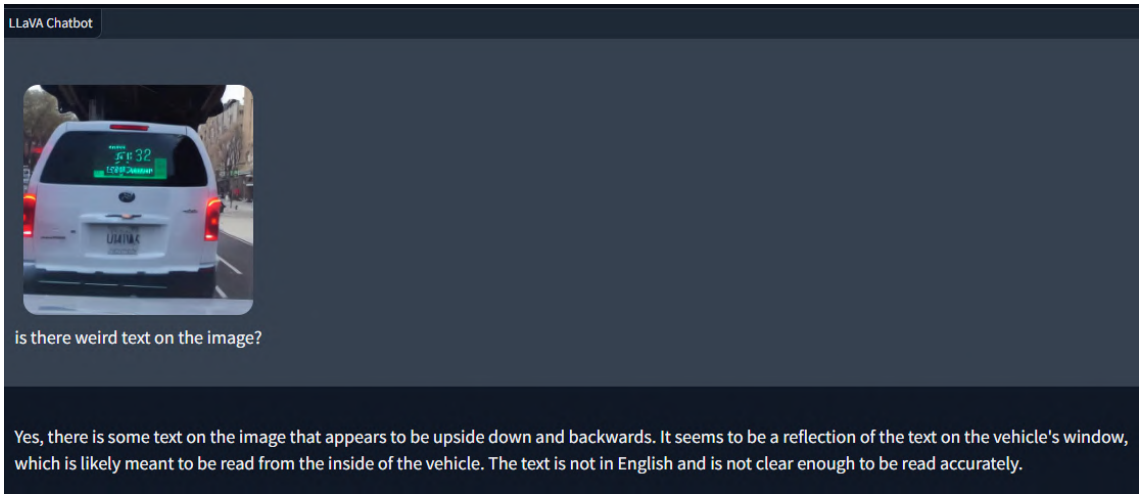


Figure 35: Utilizing LLava [50] for detecting hallucinations automatically on generated image.

6.3 From-scratch vs ControlNet model

This section outlines the main differences between the from-scratch and the ControlNet model, focusing on resolution, inference speed, performance on rare objects, training details, and other relevant aspects.

The from-scratch model operates at a resolution of 128x128 pixels, which limits the level of detail, fidelity, and most importantly, the industrial applicability of the generated images. Higher resolution image generation would have required significantly greater resources and out-of-the-box deep learning-based upscalers could not compensate this drawback ei-

ther. Training from scratch is inherently a time-consuming and resource-intensive process. In contrast, the ControlNet model functions at a higher resolution of 512x512 pixels (and can be further upscaled), producing more detailed and realistic images. Additionally, ControlNet benefits from a pre-training phase, significantly reducing the time and resources needed to achieve a reasonable level of performance.

Inference speed is another critical factor in practical applications. The from-scratch model is relatively slow in generating images due to its less optimized algorithm (DDPM) and architecture. Conversely, the ControlNet model demonstrates significantly faster inference speeds, even at larger resolutions. This improvement is partly due to ControlNet's integration within the Hugging Face ecosystem, where the availability of state-of-the-art optimizations and libraries makes it a superior implementation compared to my from-scratch model. This ecosystem allows ControlNet to be seamlessly integrated into existing workflows, facilitating faster development.

Regarding generative capabilities, the from-scratch model is built solely on my limited training data and relies exclusively on semantic segmentation guidance. In contrast, ControlNet leverages both pre-training information and prompt guidance. Therefore, ControlNet can be guided towards scene setups that were not part of the training dataset. While ControlNet's ability to hallucinate can introduce unexpected elements, this behavior can be monitored through an automatic quality checking workflow.

It is worth noting that the from-scratch model could be extended to a phase where it is first trained on another large dataset and then also aided by text guidance. However, given that ControlNet comes in a usage-ready state with a complete, well-built ecosystem, such a development would be an infeasible overhead in today's world, where producing results rapidly is a top priority.

In summary, the ControlNet model offers several advantages over the from-scratch model, including higher resolution, faster inference speed, better performance on rare objects, and more efficient training. Additionally, ControlNet's integration into the Hugging Face ecosystem makes it a superior choice for rapidly generating automotive data and improving ADAS tasks.

Chapter 7

Summary and Future Work

Throughout my work, a series of systematically planned steps were undertaken. Datasets and models for two distinct training types were prepared. Semantic segmentation mask-guidance was achieved by modifying a standard U-Net-like architecture and by training a ControlNet model. Using my custom DDPM-based diffusion model and the tailored ControlNet-based approach, I managed to train models for generating automotive data. An efficient pipeline was developed, streamlining data processing, training, and testing for each setup. Results were evaluated metrically and visually, and the effect of the synthetic data on the ADAS task was assessed.

The generation of a synthetic dataset that successfully improved an ADAS downstream task is a noteworthy achievement. This accomplishment provides valuable insights for industrial applications.

Looking forward, there are still avenues for improvement. One promising direction is to fine-tune or use LoRA [51] training on a Stable Diffusion Model prior to ControlNet training. This would help the base model better adapt to the specific data domain. Additionally, resolutions could be increased by opting for another base model, such as SDXL [52]. Finally, it would be beneficial to extend the overall pipeline with automated quality checking, ensuring that only high-fidelity generations are added to the synthetic dataset.

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 invaluable guidance and expertise, which not only significantly contributed to this work but also greatly influenced my academic journey.

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] A. S. Lundervold and A. Lundervold, "An overview of deep learning in medical imaging focusing on mri," *Zeitschrift für Medizinische Physik*, vol. 29, no. 2, pp. 102–127, 2019.

[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," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1907–1915, 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] 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," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2636–2645, 2020.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

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

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

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

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

[17] 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.

[18] W. H. L. Pinaya, S. Vieira, R. Garcia-Dias, and A. Mechelli, "Autoencoders," in *Machine learning*, pp. 193–208, Elsevier, 2020.

[19] A. Van Den Oord, O. Vinyals, *et al.*, "Neural discrete representation learning," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[20] 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.

[21] D. Rampas, P. Pernias, and M. Aubreville, "A novel sampling scheme for text- and image-conditional image synthesis in quantized latent spaces," *arXiv preprint arXiv:2211.07292*, 2022.

[22] P. Pernias, D. Rampas, M. L. Richter, C. Pal, and M. Aubreville, "Würstchen: An efficient architecture for large-scale text-to-image diffusion models," in *The Twelfth International Conference on Learning Representations*, 2023.

[23] 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.

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

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

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

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

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

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

[30] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *Advances in Neural Information Processing Systems*, vol. 34, pp. 8780–8794, 2021.

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

[32] 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.

[33] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1125–1134, 2017.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] Y. Wu and K. He, "Group normalization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 3–19, 2018.

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

[40] 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.

[41] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[42] 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.

[43] W. Zhao, L. Bai, Y. Rao, J. Zhou, and J. Lu, "Unipc: A unified predictor-corrector framework for fast sampling of diffusion models," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[44] J. Nilsson and T. Akenine-Möller, "Understanding ssim," *arXiv preprint arXiv:2006.13846*, 2020.

[45] 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.

[46] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton, "Demystifying mmd gans," *arXiv preprint arXiv:1801.01401*, 2018.

[47] 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.

[48] 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.

[49] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[50] H. Liu, C. Li, Y. Li, and Y. J. Lee, "Improved baselines with visual instruction tuning," *arXiv preprint arXiv:2310.03744*, 2023.

[51] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[52] D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Müller, J. Penna, and R. Rombach, "Sdxl: Improving latent diffusion models for high-resolution image synthesis," *arXiv preprint arXiv:2307.01952*, 2023.

Appendix

Code listings

Preparing alpha (α), alpha_hat ($\bar{\alpha}$), beta (β):

```
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
```

Supplementary images

Figure A.0.36: Test masks and corresponding generations (From-scratch model)

Figure A.0.37: Test masks and corresponding generations (ControlNet model)

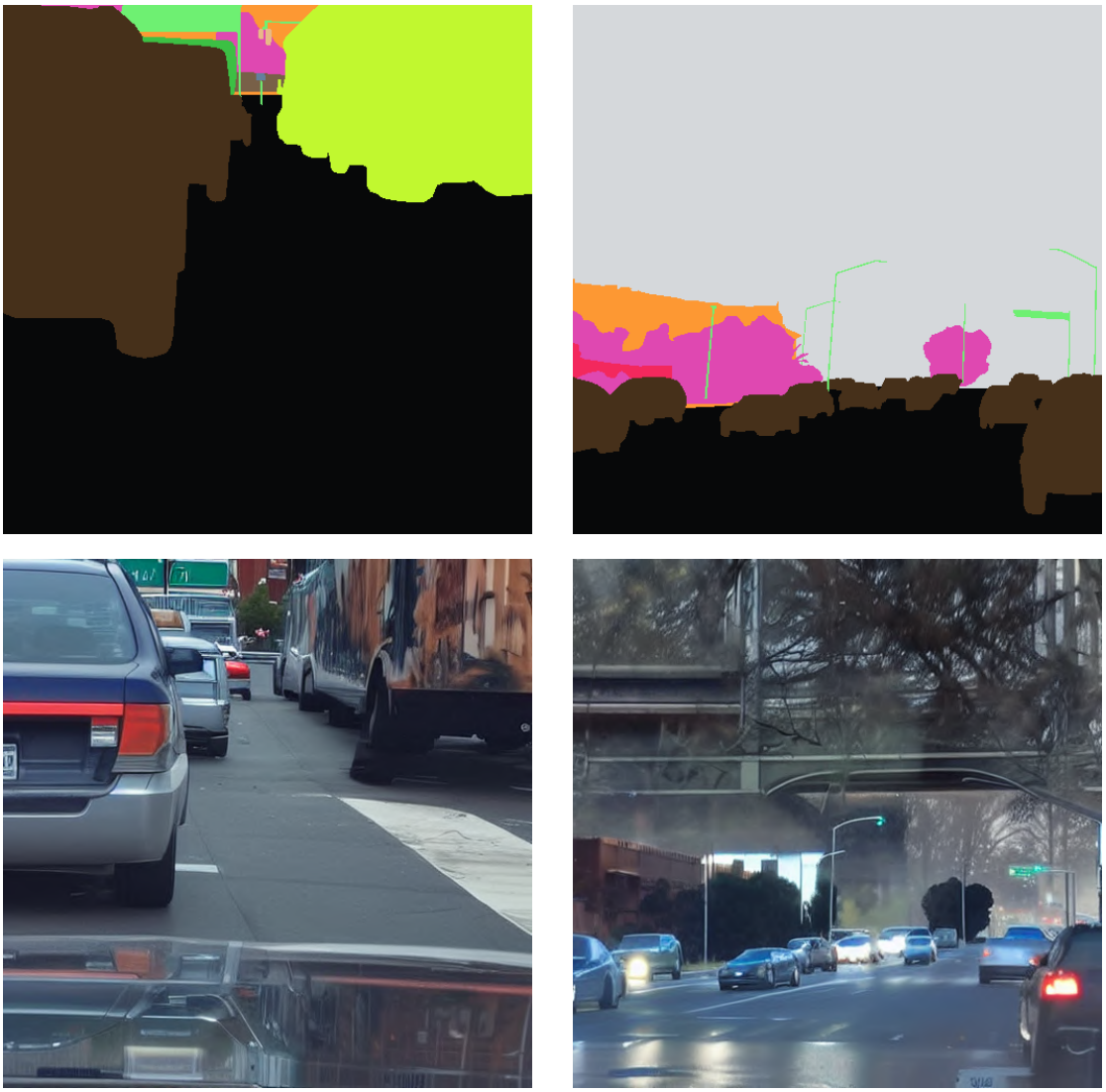


Figure A.0.38: Test masks and corresponding generations (ControlNet model)