Deep Learning in Data Science Project Report

Custom Project: Image segmentation with Diffusion models

Jacques Fürst
Michal Andrzej Sitarz
Robert Skoglund

May 26, 2024

Abstract

Denoising Diffusion Probabilistic Models (DDPMs) mark a significant advancement in the development of the field of Computer Vision. Given their success in image generation, we aim to utilize the feature maps created by a trained DDPM as an insertion into a pixel classifier to perform image segmentation. Furthermore, we aim to evaluate the impact of different architectural add-ons on the DDPM, which were introduced in the Deep Learning community over the past years. Despite training a functional DDPM from scratch which can reconstruct and generate a good image, we found that the adjustments we made from the original research resulted in poor performance of the semantic segmentation task. Furthermore, we experimented with various design modifications to improve the DDPM's performance and better understand what makes it work well.

1 Introduction

Diffusion models, which were introduced in 2020 [5], are powerful tools to generate images, using the diffusion procedure that is based on nonequilibrium thermodynamics. Their introduction arguably marks a milestone in the field image synthesis - beating GANs in their image quality and diversity. Therefore, exploring the capabilities of these models beyond their traditional task of image synthesis is of interest.

In this work, we aimed to replicate the approach applied in the paper *Label-Efficient Semantic Segmentation with Diffusion Models* [1] and through it evaluate both different design choices in the evolution of the diffusion model architecture. Therefore, a basic diffusion model architecture was implemented and trained on the *Pascal Visual Object Class* (VOC) dataset [3]. Then, feature maps retrieved from the upscaling layers of the trained diffusion models UNet were used to train a pixel classifier [12] to see the performance of the base model on this specific task. Finally, the given base model was trained on the MNIST dataset [6] and modified with improvements from [2], [7] and [9] in order to determine the significance of the performance improvements of these changes.

2 Related work

In [5], the authors trained a Denoising Diffusion Probabilistic Model (DDPM) by reversing the gradual noising process to create high-quality synthetic images. The DDPM aims to learn the reverse process and reconstruct the noisy data, which is usually done with a UNet architecture that can capture contextual information of the data and preserve details. These models can be applied to a range of Computer Vision tasks, such as in [1], where it was used to segment images. They used the architecture proposed in [2], and trained a segmentation mask model with features from the DDPM, with a very small subset of the data.

Furthermore, a lot of research has been done on improving the diffusion models. For instance, an improved DDPM was proposed in [9], where they improve performance by adapting the UNet architecture of the diffusion model as well as adjustments to loss and the noise function of the diffusion process. The UNet architecture is also crucial to the training of Diffusion models. The base model from [5] relies on ResNet blocks. However, in [7] the authors propose a ConvNeXt block, which is an improvement on the ResNet block, and Swift Transformer. They altered the network by using depth-wise convolution, GELU activation, and layer normalization (and some other micro-design changes).

Finally, [4] proposed a metric, Frechet Inception Distance (FID), for evaluating the similarity of the images sampled by the diffusion models against the real images.

3 Data

In this project, two datasets were employed for our training: VOC and MNIST.

The first dataset is the *PASCAL Visual Object Classes (VOC) 2012* segmentation challenge dataset [3], containing 9993 images with labels for 20 image segmentation classes with the superclasses: 'Person', 'Vehicle', 'Animal', and 'Indoor'. Hence, this dataset was employed for the image segmentation task since it had labeled data available. Furthermore, the labeled objects were mostly fairly large and the images had a lower complexity than for instance the *CelebA* dataset [8] which was employed by the authors of [1]. This was of use to us since the limited computational resources at our disposal required us to reduce the dimensionality of the images we used for the training of the diffusion model somehow. We opted to reduce the images from 256x256 to 32x32 pixels, thus significantly reducing the image quality, but making training the diffusion model a more achievable task. The larger size of each class object in the dataset came in handy here since the downscaling of the images would not lead to as many class inconsistencies (since several pixels with different classes might be merged into one otherwise).

For the evaluation of the different diffusion architecture configurations, the MNIST dataset [6] was employed. It is a good standardized dataset containing a training set of 60,000 images and 10,000 test images. The images we used consisted of 32x32 pixels with one channel only (as compared to the usual three RGB channels) making it a good fit for training a model that yields comparable results after shorter periods of training, which supported the training procedure within the time constraints of this project.

Finally, we used the Bedroom28 dataset from [11] to test our pixel classifier on a pre-trained model. The dataset is composed of 256x256 images of bedroom scenes with 30 most re-appearing classes.

4 Methods

The initial goal was to dive into using a diffusion model architecture for semantic segmentation as proposed in [1] and try various improvements. However, as it was mentioned in Section 3, we settled for using a different dataset, with smaller resolution, and thus, we needed to adjust the architecture and training of the diffusion model. Hence, we implemented a DDPM (see Figure 1) as proposed in [5] and trained on the VOC Segmentation data. Firstly, we complete the forward process, which can be described as:

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

where β_t is the variance scheduler that determines the amount of noise added at each step. We are teaching the UNet model to reverse the process and reconstruct the original data from noise, which can be defined as:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_\theta(x_t, t)).$$

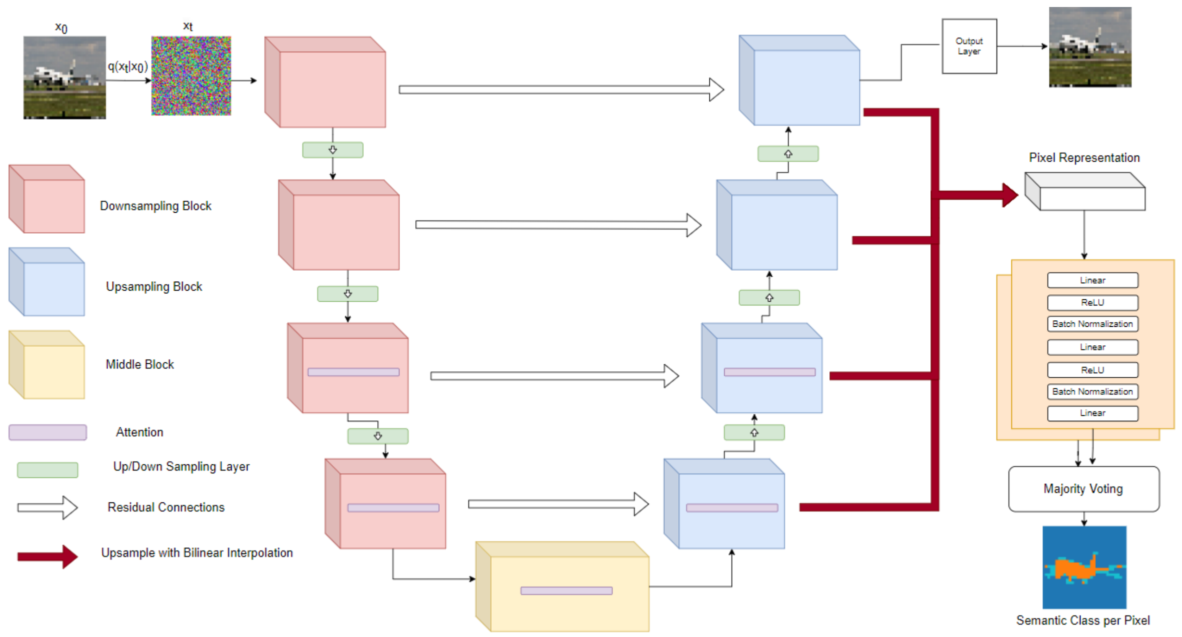


Figure 1: DDPM and Pixel Classifier Architecture

The baseline model was trained using MSE loss with 1000 diffusion steps which should make the noising transitions smoother and training more stable. When sending the image through the network, the timestep is accompanied, utilizing Sinusoidal Timestamp Embedding as in [10]. All the blocks are assembled by combining ResNet blocks, and after Down/Up sampling blocks are passed, Down/Up sampling layers are used. We added attention [10] into the lower layers to capture long-term dependencies, enhance feature representation and handle more complex patterns. Last notable feature of the UNet is the residual connections between all of the residual blocks in down and upsampling. Finally, the features are passed through a final convolutional layer to retrieve initial image dimensions.

With the baseline DDPM, we were able to extend it to include a pixel classifier which will be used for semantic segmentation, according to [1, 12]. When passing the data through the UNet, the feature maps can be extracted, hopefully encoding some low-level information about the images. For every image, features are extracted from all upsampling blocks (B = {6,7,8,9}), upscaled with bilinear interpolation to fit the image size, at three timesteps (t = {50, 150, 250}), concatenated and flattened (resulting in 1536 channels). We had to slightly diverge from [1], as it was a little bit unclear what block 12 refers to, but we tried to replicate the timesteps as their investigation showed that these should work the best. The pixels are then used to train an Ensemble of Multi-Layer Perceptrons as proposed in [12], with the majority voting of pixels.

Finally, we implemented various extensions to the baseline DDPM architecture, as we hoped that improving the diffusion model would conjointly improve the semantic segmentation. The first addition we chose was an improved cosine noise schedule for the Diffusion process, which was advertised in [9] to improve model performance, especially on smaller image sizes, which made it a valuable choice for our approach. It functions by adjusting the distribution with which the noise is added throughout the diffusion process by flattening its curve and setting minimum and maximum values for the beta schedule to prevent adding too little/much noise at the beginning and end of the process, respectively. Secondly, we added a new architecture for the residual block as introduced in [7], as we wanted to find out if it will capture the spatial and channel-wise information more effectively than ResNet blocks (see Appendix A). Thirdly, we added adaptive group normalization (AdaGN), which should improve the FID of the diffusion model and is defined as:

$$AdaGN(x) = \gamma_{G_i}(\frac{x_{ij} - \mu_{G_i}}{\sqrt{\sigma^2_{G_i} + \epsilon}}) + \beta_{G_i},$$

where γ and β are adaptive scale and shift parameters, μ and σ^2 are mean and variance over the group channels G_i. This seemed like a good choice to improve the performance, since normalization is performed throughout the whole UNet, and this should make more adaptable and make better generalizations.

5 Experiments

There were three main investigations that we conducted: **(1)** training a baseline DDPM model on the VOC dataset and synthesizing images; **(2)** extending (1) to incorporate the pixel classifier for semantic segmentation on VOC data; **(3)** testing extensions and modifications on the DDPM from (1) to synthesize better images on the MNIST dataset.

5.1 DDPM training on VOC Dataset

The final version of the baseline model was trained on 1000 epochs. We performed some small tests to test the outputs that the DDPM produces, such as using a small subset of the data, running for longer/shorter, and using a simpler version of the UNet (having less residual connections between blocks, no attention, and a shallower network). However, to obtain results better than pure noise when reconstructing the images on the VOC dataset, we had to implement the architecture described in Section 4. The resulting sampled images look very reminiscent of the original dataset, albeit slightly darker, but we concluded that we successfully managed to synthesize images based on the training dataset (see Figure 2).

5.2 Pixel Classifier on VOC Dataset

The pixel classifier was trained using our DDPM. Initially, we used 20 images (as advertised in [1]) from the VOC validation set to ensure the generality of our model performance. Afterward, due to poor results, we tried to train on more images, tried to change the learning rate, and utilized batching with shuffling to try and tackle the overfitting. However, the pixel classifier doesn't work perfectly on unseen data (see Figure 3) = the validation loss stagnates and the performance on simple masks is sub-optimal. Some

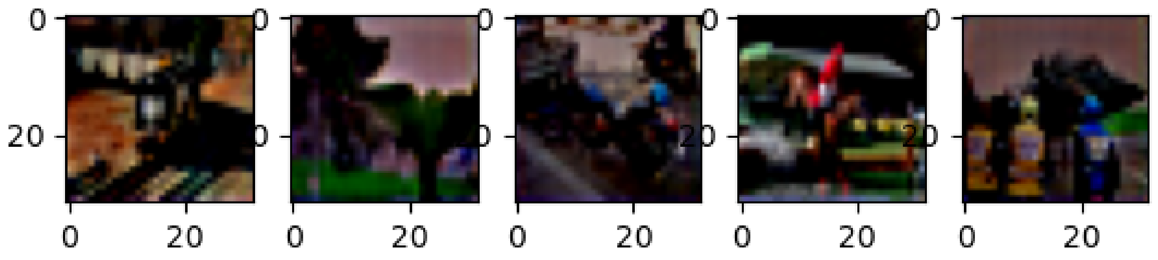
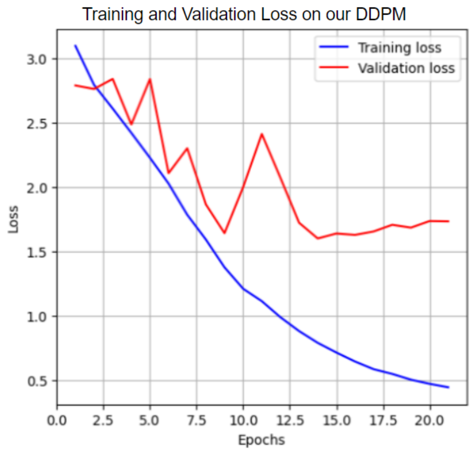
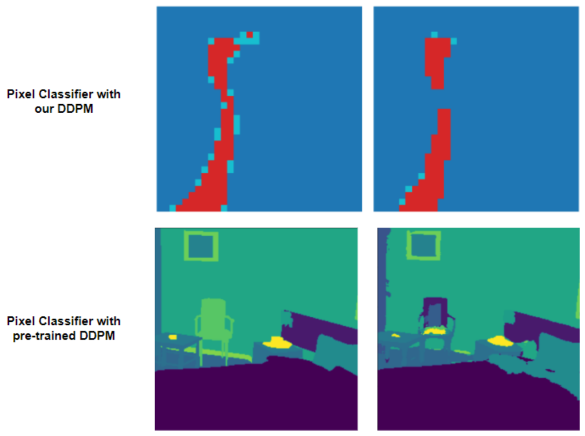


Figure 2: Sampled images from the baseline DDPM model

possible problems could be caused due to differences from [1]: smaller resolution images, modifications to the UNet architecture, a dataset that has more sparse labels (an extensive background and a small number of classes per image, which we tried to tackle by gathering all the pixels and randomizing the order), and using fewer features for classification. As I final experiment, we tested our implementation of the Pixel Classifier with a pre-trained DDPM [11]. We extracted the model and features using their pipeline (and used the Bedroom28 dataset for consistency with their code), and after a short training and just a small ensemble, we got promising semantic masks (see Figure 3 (b)). Therefore, we can conclude that our DDPM and our pixel classifier work well separately, however, more care needs to be taken when putting them together to create a semantic segmentation model.



(a)


Pixel Classifier with our DDPM
Pixel Classifier with pre-trained DDPM

(b)

Figure 3: **(a)** Loss whilst using our model; **(b)** Predicted labels for our (top) and [11] (bottom) model.

5.3 DDPM extensions on MNIST

We decided to test the DDPM modifications on an easier dataset (MNIST); we trained it for 50 epochs and tested the performance by visually inspecting the generated images and computing the FID. The FID score might not be the best metric to compare it to other research, as it was trained on images of size $299 \times 299 \times 3$. However, it can still serve as a valuable metric for comparing the differences between our

Experiments	**FID score (↓)**
DDPM	47.081
DDPM with Adaptive Group Normalization	49.303
DDPM with ConvNext blocks	44.001
DDPM with Cosine Noise Scheduling	46.541
DDPM without attention	52.476

Table 1: FID based on model type

models (as we run all of them in a standardized fashion). We sampled from each model three times and averaged the scores calculated from all three samples to get our results, which are summarized in Table 1.

Looking at the results, the most impactful changes to the model appear to be using the ConvNeXt architecture and removing the attention from the model. As expected, it improves model performance, as it is meant to capture more information than ResNet blocks, thus, leading to a lower FID score on the generated images. On the other hand, the removal of attention worsens the model's performance. Looking at the adaptive group normalization, it appears to worsen the model performance which could be due to the different architecture we employ as compared with the authors and the adjusted image size we train on. The improved noise scheduling, whilst improving the model performance a little bit, does not have a significant impact altogether. It might be that the simplicity of the dataset and rather short training duration (50 epochs) lessen its impact.

6 Conclusion

In conclusion, we investigated the DDPM architecture and its application to semantic segmentation. The steps we took were: 1. training a baseline DDPM for image synthesis, 2. applying the DDPM in the context of semantic segmentation by incorporating a pixel classifier, and 3. exploring modifications to DDPM architecture to improve image synthesis, namely (i) adaptive group normalization, (ii) ConvNext blocks, (iii) cosine noise scheduling, and (iv) ablating attention. Training the baseline DDPM generated relatively realistic images considering the smaller image size, with the architectural modifications having a varying performance on the FID. The highest performance boost is attributed to the ConvNext blocks, decreasing the FID, followed by cosine noise scheduling. Adaptive group normalization and ablating attention resulted in worse performance. The pixel classifier had a mediocre performance, but we determined that the MLP itself worked fine, but the problem lied in the feature maps we used. Therefore, having more time, and better resources, we would probably have managed to produce better results.

References

[1] Dmitry Baranchuk et al. “Label-efficient semantic segmentation with diffusion models”. In: *arXiv preprint arXiv:2112.03126* (2021).

[2] Prafulla Dhariwal and Alexander Nichol. “Diffusion models beat gans on image synthesis”. In: *Advances in neural information processing systems* 34 (2021), pp. 8780–8794.

[3] Mark Everingham et al. *The PASCAL Visual Object Classes (VOC) Challenge*. Accessed: 2024-05-20. 2010. URL: http://host.robots.ox.ac.uk/pascal/VOC/.

[4] Martin Heusel et al. “Gans trained by a two time-scale update rule converge to a local nash equilibrium”. In: *Advances in neural information processing systems* 30 (2017).

[5] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.

[6] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. *The MNIST Database of Handwritten Digits*. Accessed: 2024-05-20. 1998. URL: http://yann.lecun.com/exdb/mnist/.

[7] Zhuang Liu et al. “A convnet for the 2020s”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 11976–11986.

[8] Ziwei Liu et al. “Deep Learning Face Attributes in the Wild”. In: *Proceedings of International Conference on Computer Vision (ICCV)*. Dec. 2015.

[9] Alexander Quinn Nichol and Prafulla Dhariwal. “Improved denoising diffusion probabilistic models”. In: *International conference on machine learning*. PMLR. 2021, pp. 8162–8171.

[10] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[11] yandex-research. *ddpm-segmentation*. https://github.com/yandex-research/ddpm-segmentation/tree/master. 2022.

[12] Yuxuan Zhang et al. “Datasetgan: Efficient labeled data factory with minimal human effort”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 10145–10155.

Appendix A

Figure 4 summarizes the main differences between the ConvNeXt and the Residual blocks. Firstly, the depth-wise convolution with kernel of size 7 is used. Secondly, layer normalization is used over batch normalization, and only once, as they argue normalizing less yields better results. Then they use a piece-wise convolution that increases the number of channels four-fold. In our implementation we only doubled it, due to memory requirements. Afterward they use GELU activation, and once again they use only one activation function, followed by another convolution layer to bring the dimensions down again. Finally, they utilize the residual connection, just as is done in the ResNet block. The ConvNeXt block is meant to be superior to ResNet blocks, and when the paper was published it was used to outperform the Swift transformer which was SOTA at the time.

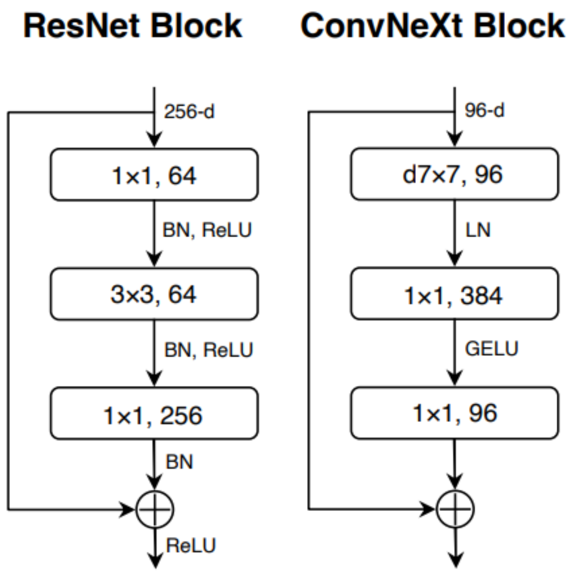


Figure 4: Differences between ConvNeXt and Residual blocks [7].