# Denoising Diffusion Probabilistic Models for Number Generation

George Witt

May 15 - 19 2023

**Abstract**

Denoising Diffusion Probabilistc Models (DDPM) are modified UNet models with added self-attention and residual connections trained to remove gaussian noise over a poisson noise application process. This technique has been used to great effect in image generation [1]. We implement and train a DDPM model based on existing hyperparameters to generate black and white images of numbers from the well-known MNIST dataset, with a modified UNet structure meant to run on lower-end GPUs [1, 2]. We demonstrate the effect of exponential smoothing on convergence time in the loss function, and demonstrate the construction of image features earlier in training compared to the standard model [3]. We conclude with a discussion of our implementation of the DDPM model. Our work is available online on github at this link `https://github.com/georgew79/PHYS486_FinalProject_Diffusers.git`

## 1 Introduction

Probabilistic models have historically suffered from a tractability and flexibility trade-off [4]. Models that are highly tractable are simple to fit, but lack the ability to fit arbitrary distributions in underlying data, and vice versa [1, 4]. The Denoising Diffusion Probabilistic Model (DDPM), however, avoids the trade-off completely. The model takes inspiration from nuclear thermodynamic processes in which one distribution is shifted to another through a forward 'diffusion' process [4].

In the DDPM model, a forward shift with applied noise from a known distribution is applied such that the DDPM model can learn the reverse transition [4]. This therefore means that the DDPM model can, in theory, revert given random noise following a similar distribution as to the one provided into data that resembles the starting values. This is a effectively a form of regression for the original starting values, which leads to raw generation in the case of truly random noise - not just noised data - provided to a model [1, 4].

## 2 DDPM Model

Let us now formally describe and derive attributes for the DDPM model. This follows the derivation in [1].

We begin by formally describing what is known as the 'forward diffusion process'. We state that this diffusion process follows a schedule of noise known as $\beta \equiv \beta_1, \beta_2, ..., \beta_T$,

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

The forward process gives us a resulting final noised version of our data at timestep $T$, with $q(\mathbf{x}_T|\mathbf{x}_0) = \prod_{i=1}^{T} q(\mathbf{x}_i|\mathbf{x}_{i-1})$, where $T$ is some arbitrary hyperparameter and the variance schedule is also a set of hyperparameters.

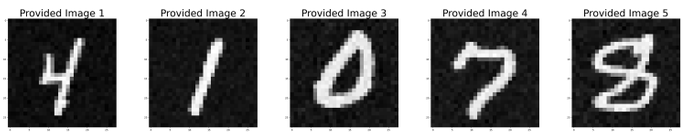
The reverse process $p(\mathbf{x}_0|\mathbf{x}_T)$ can be defined as follows

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

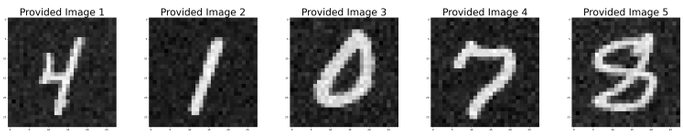
$$p(\mathbf{x}_0|\mathbf{x}_T) = \prod_{t=1}^{T} p(\mathbf{x}_{t-1}, \mathbf{x}_t)$$

as shown in [1]. The reverse process $p$ is therefore a function of $\theta$, the model parameters. In other words, $p \equiv p_\theta \equiv p(..)|(\theta)$.

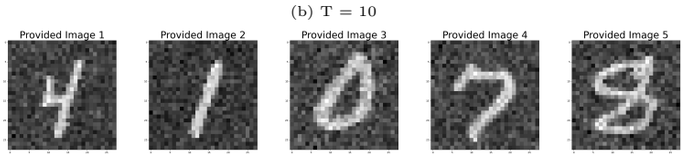
We now must describe the loss function. We begin with the loss function described with the variational bound on the negative log loss such that



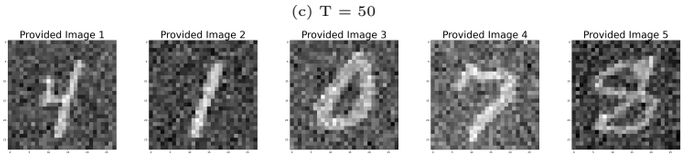

(a) T = 2



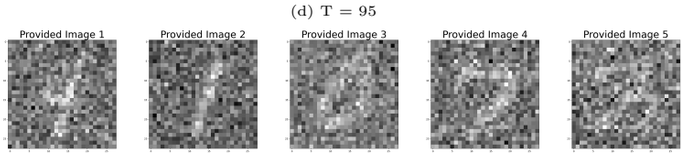

(b) T = 10



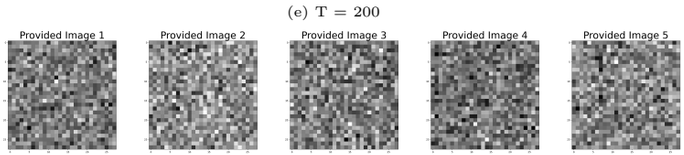

(c) T = 50




(d) T = 95




(e) T = 200




(f) T = 400

Figure 1: Example noising MNIST data over steps 2, 10, 50, 95, 200, 400.

$$L := \mathbb{E}(-\log(p_\theta(\mathbf{x}_T)) - \sum_{t \geq 1} \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}$$

After realizing the Gaussian shape of $q(\mathbf{x}_T|\mathbf{x}_0)$ it is possible to derive the loss function in terms of KL divergence, before finally showing

$$L \propto \mathbb{E}(||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}}\epsilon, t)||^2)$$

where $\alpha = 1 - \beta$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, and $\epsilon, \epsilon_\theta$ are the true added noise from the forward process and the predicted noise given $\theta$ respectively. This is useful as a simplified loss function, since the coefficients out front are constant.

This means that from the negative log loss, we have derived how we can simply apply the MSE to the predicted losses to improve the model's ability to 'denoise' the given data. We note that the derivation is continued and more complete in [1, 4]. The timestep derivations are given in [1], and used in our implementation at the given link.

# 3 DDPM Architecture

The DDPM architecture is built to predict the true noise applied to a data structure given noised data structures using the MSE, as we just derived. It must therefore take in batches of noised data, and reproduce batches of the noise itself, effectively splitting them. It is trivial from this to reproduce the original image, given our predictions of the noise are strong [1]. This is demonstrated in the 'sample from model' function in our provided implementation.

To illustrate this, we will work with images as our data structure. In other words, we will learn how to predict a starting image given a Gaussian noised image. This can later be applied to generating brand new images similar to a given dataset given conditioning on the particular dataset starting from random Gaussian noise.

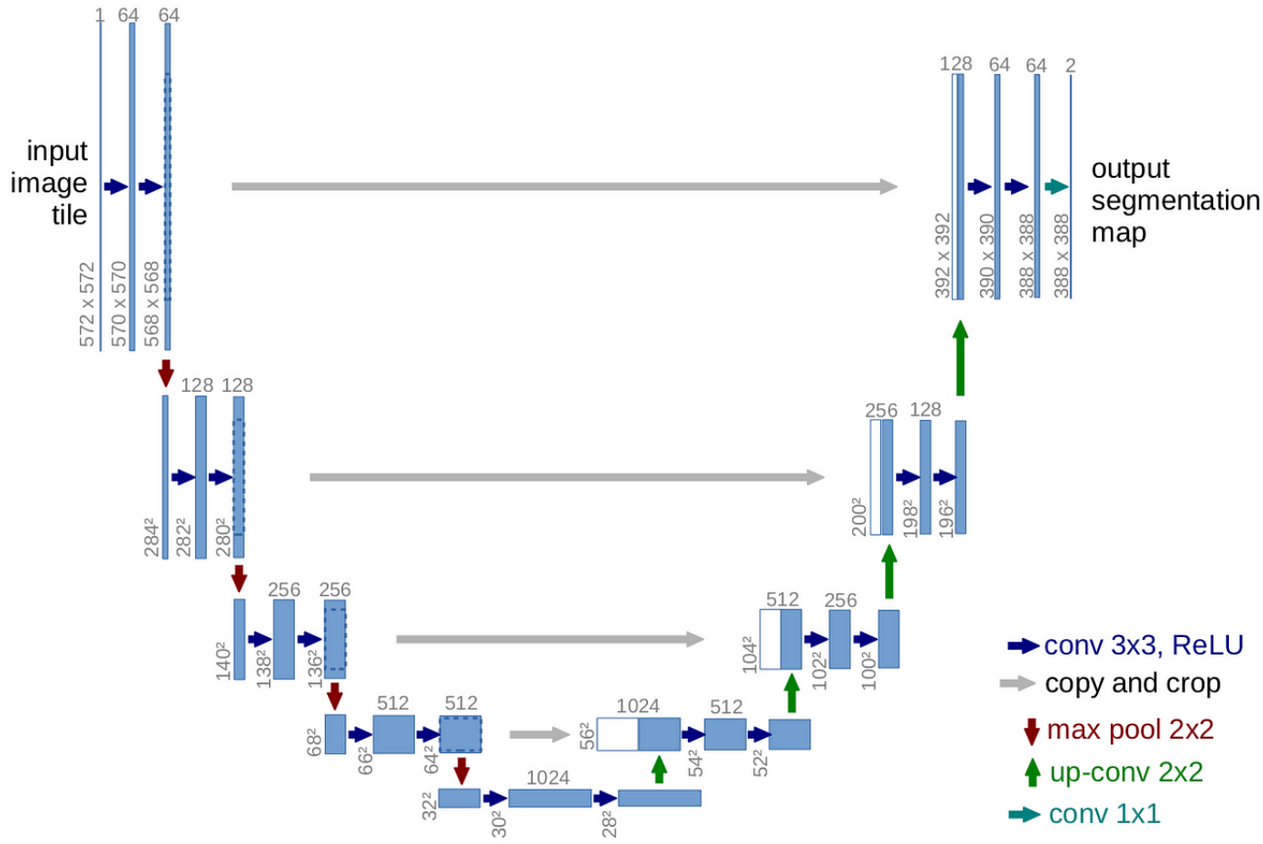



Figure 2: Standard UNet architecture diagram [2]

At a high level, the DDPM architecture consists of the following.

1. A series of 'down' blocks that decrease image size while casting into larger channel space through a series of convolutional layers and pulling.
2. Self-attention layers that separate the down blocks.
3. A 'bottom' series of convolution layers.
4. A series of upsampling blocks that return to the original image and channel size, coupled with residual connections from the matching down layers of the network.

This DDPM architecture is a modified form of the UNET architecture, shown in [1, 2]. The primary modification is the self-attention, with multihead attention layers. We further modifiy the architecture by reducing its size from 4 to 2 down and up sections as to lower the required VRam for training. While the current DDPM models were trained on arrays of high VRAM GPUs, we only have access to a standard NVIDIA GPU with 8 GB of VRam for training [1].

To compensate for the memory deficiency, we utilize the following model layers. Please see the provided implementation at `https://github.com/georgew79/PHYS486_FinalProject_Diffusers.git` for descriptions of the following layers.

Please note that the embedding dimensions used for the time embedding is 512, the same as its length.

1. Double convolution: 28 x 28 image to 28 x 28 image with 1 channel to 64 channels.
2. Downward block: Double convolution from 28 x 28 image to 14 x 14 image with 64 channels to 128 channels.
3. Self attention: Self attention with size 14.
4. Downward block: Double convolution from 14 x 14 image with 128 channels to 256 channels and 7 x 7.
5. Self attention: Self attention with size 7

6. Bottom convolution: Keeps size and channels, passing through intermediary layers of 512 channels.

7. Upward block: Upsample to 14 x 14, add in residual cross connection from the 2nd downward block and decrease channels from 256 to 128.

8. Self attention: Self attention with size 14

9. Upward block: Upsample to 28 x 28, add in residual cross connection from the 1st downward block and decrease channels from 128 to 28.

10. Convolutional layer: Keep 28 x 28, but decrease channels from 28 back to 1.

# 4 Exponential Moving Average

The EMA (Exponential Moving Average) technique for improving the training of the DDPM model is a very simple modification [3]. Effectively, after some warmup time, the idea is to update the weights of an external 'EMA' model using a weighted combination of the current weights and the new weights [3]. The idea is to limit the changes in the EMA model over time to prevent jumps from being large.

The equation, given a $\beta$ update proportionality factor, is as follows [3].

$$w_{new} = \beta w_{old} + (1 - \beta) w_{base,new}$$

Where $w_{old}$ are the old EMA model parameters, and $w_{base,new}$ the freshly updated parameters of the base (standard) model.

# 5 MNIST Dataset

We perform our training on the full MNIST (Modified National Institute of Standards and Technology) dataset [5]. This dataset consists of black and white 28 x 28 handwritten digits from mixed NIST datasets [5]. It includes a total of 60,000 training images, and 10,000 test images. We employ all MNIST as passed with the dump file [5].

This dataset is standard in 'initializing' image models, and providing a proof of concept for the success and viability of the image model. It is akin to the 'CartPole' problem for Reinforcement Learning, or a 'Hello World' of sorts for machine learning.

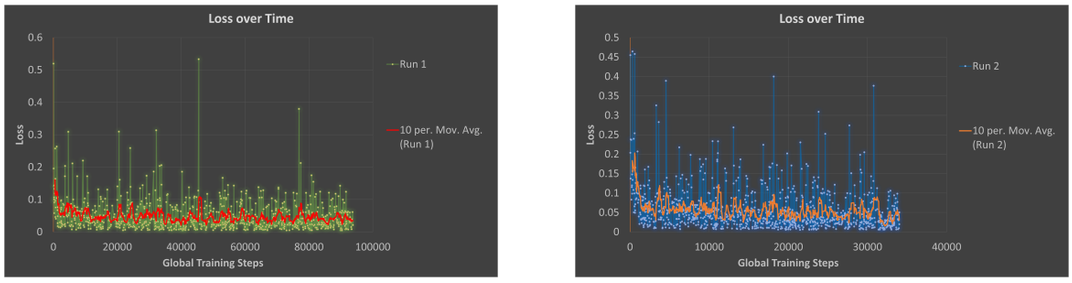
# 6 Training and Results

We present our loss curves over training for 2 runs of the DDPM architecture. We also present results from various stages in training of 4 standard DDPM models, and 2 being trained with the EMA approach. Overall we observe qualitatively strong performance from all models, though the strongest convergence time in the EMA model as expected [1,3]. We note that the later runs had to be amended in length for time after clear convergence was shown in order to collect a larger number of runs for the final submission.

We note that each training epoch is performed as 938 batches of size 64 over the full MNIST dataset. We further note our hyperparameters: learning rate $= 2 * 10^{-4}$, 512 noise steps, linear noise variance schedule from 0.0001 to 0.02, and an EMA combination parameter $\beta$ of 0.992. These values are all chosen based on recommendations from [1], and various short experiments with $\beta$. We choose the GELU and SiLU activation functions for comparison with existing implementations like (`https://github.com/dome272/Diffusion-Models-pytorch/blob/main/modules.py`). Activation function choices are not reported in [1, 3], and so we choose the GELU and SiLU to ensure comparability to existing implementations. The GELU activation is employed in the self attention step, and SiLU in the generation of the noise timestep embeddings for use in the UNet.

One may observe that all models qualitatively show the beginnings of structure by the 10th epoch. This includes the central white 'handwritten' feature, on a dark background. The EMA model, however, is the only model to directly have developed numerical features by this point in training. See figure 5, for example, in which it is clear to see that early in


Figure 3: Full loss curves over time for 2 different models. The first run is trained alone as its own model, and was therefore able to run for 6 hours. The second run was training in tandem with an EMA model, and could therefore only run for about 2 hours. The epochs are 100 and 35 respectively for runs 1 and 2. EMA model loss curves were further intensive memory wise to generate, and are not provided.




(a) Run 1

(b) Run 2

Figure 4: Two separate loss curves

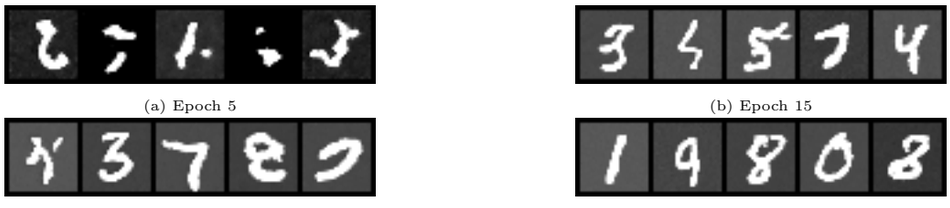
(a) Standard epoch 5

(b) EMA epoch 5

Figure 5: Comparison of early training, epoch 5, between two other versions of the EMA and standard model in training. Notice the total lack of structure in the standard model as compared to the EMA model this early in training.

training (epoch 5) no central features had yet developed in the standard model while the EMA model is well along with training its development of the numbers in the center.

All in all, we report the loss curves for 2 of the runs in figures 3 and 4, a specific example of the EMA advantage for early training in figure 5, sample standard model training outputs in figure 6, and a comparison between EMA and standard model training over 30 epochs in figure 7.



(a) Epoch 5

(b) Epoch 15

(c) Epoch 35

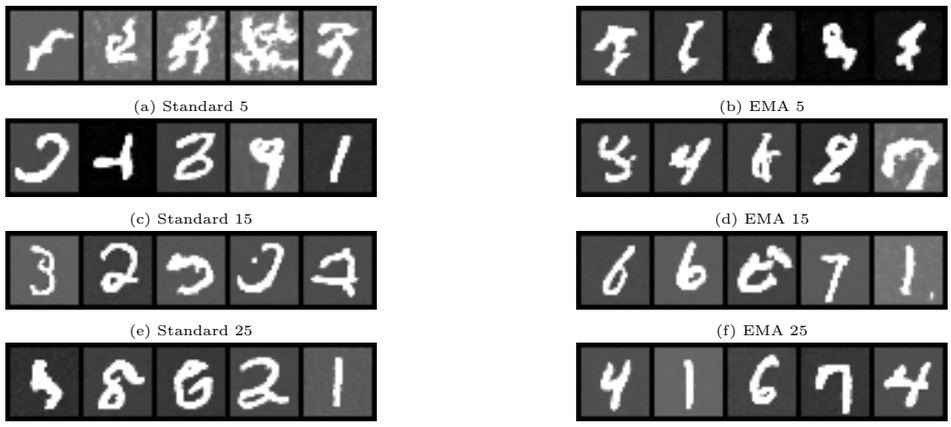
(d) Epoch 65

Figure 6: Sample output from the long standard model training over 6 hours, from epochs 5, 15, 25, and 65, to show progression of training initially and then the quality of final results.

This improvement in convergence time - manifested in faster realization of the image structure earlier in training - for EMA is as expected from outstanding literature [1, 3].

# 7 Conclusion

Overall, we implement and present the training of a DDPM model in python with the Pytorch framework. We provide a brief derivation of the mathematics behind the model, including a derivation of the MSE beginning from the log loss of a probabilistic model. We



(a) Standard 5 (b) EMA 5

(c) Standard 15 (d) EMA 15

(e) Standard 25 (f) EMA 25

(g) Standard 30 (h) EMA 30

Figure 7: Sample output from the shorter trained standard model and the co-trained EMA model, from epochs 5, 15, 25, and 30, to show progression of training initially and then the quality of final results.

further qualitatively demonstrate the effect of EMA training on improving the visual fidelity of the generated digits earlier in the training process than for the standard DDPM. We note that future work can be applied to using this model for more complex and larger image generation (requires better GPUs available), classifier free guidance for text-based generation, or even to domains other than image generation such as the original thermodynamic denoising applications [6, 4]. We provide a open-source implementation on github at this link `https://github.com/georgew79/PHYS486_FinalProject_Diffusers.git`

# 8 References


[1] Ho, J., Jain, A., and Abbeel, P. (2020, December 16). Denoising Diffusion Probabilistic models. arXiv.org. https://arxiv.org/abs/2006.11239

[2] Ronneberger, O., Fischer, P., and Brox, T. (2015, May 18). U-Net: Convolutional Networks for Biomedical Image Segmentation. arXiv.org. https://arxiv.org/abs/1505.04597

[3] Nichol, A., and Dhariwal, P. (2021, February 18). Improved denoising diffusion probabilistic models. arXiv.org. https://arxiv.org/abs/2102.09672

[4] Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. (2015, November 18). Deep unsupervised learning using nonequilibrium thermodynamics. arXiv.org. https://arxiv.org/abs/1503.03585

[5] L. Deng, "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]," in IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 141-142, Nov. 2012, doi: 10.1109/MSP.2012.2211477.

[6] Ho, J., amp; Salimans, T. (2022, July 26). Classifier-free diffusion guidance. arXiv.org. https://arxiv.org/abs/2207.12598