RBE 474X: Dreaming Data

Jakub Jandus, Hudson Kortus, Dexter Stark
Worcester Polytechnic Institute
Worcester, Massachusetts

***Abstract*—This project implements a denoising diffuions Model (DDPM) to generate synthetic images to augment the CIFAR-10 dataset. The synthetically generated images are used to train a simple multilayer perceptron (MLP) and we show these training images improve model accuracy. Later the team explores enhancing the DDPM using a nonlinear cosine scheduler and adjusting model hyperperameters with mediocre results.**

I. Introduction

Diffusion Models have been shown to have performance comparable, if not superior to more conventional image generation models such as GANs and VAE models. [1] Synthetically generated images can be powerful for dataset augmentation where generated images are added to the training dataset, increasing the size that the model can be.

In project P4: Dreaming Data, the goal was to generate synthetic data just like the CIFAR-10 dataset using a diffusion model. This is done by training the diffusion network on images that contain gradually noisier data. Just like a drop of color slowly diffuses in a glass of water. Upon image reconstruction, we begin with a random noise and a class label. Through an iterative process, the image is refined until it becomes fully denoised - this is analogous to starting with a glass of dyed water and progressively reversing the diffusion process to obtain the initial drop of color.

II. Diffusion Implementation

The implementation of the diffusion model within our code required that we complete the unfinished initialization, forward, and reverse process calculations. The forward process is where we train the model by giving it progressively noisier and noisier images, letting it learn the difference between an image and noise. In the reverse process the diffusion network is given noise and iteratively removes noise until a hallucinated image is generated.

III. Training

For the forward pass, a training image is iteratively corrupted with Gaussian noise, dictated by a scheduler variable β_t. In parts 1 and 2 β_t is controlled linearly, but in part 3 β_t is nonlinear which and will be expanded upon later. The linear forward process is shown in figure 1 where x_t is the next most corrupted image. To make the equation continuous, β_t is cumulatively multiplied to generate $\overline{\alpha}$, allowing for instantaneous calculation between corruption points. Finlay, to implement the corruption pass in code, figure 1 is rearranged to generate figure 2. Training is performed by taking the gradient

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

Fig. 1. Definition of normal distribution added at at each forward step

$$x_t = \sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\epsilon$$

Fig. 2. Continuous equation giving the noised image x_t given training image x_0

of the l2 loss of the difference between the ground truth added noise figure 2 and the predicted noise ϵ_θ.

In order to ensure that the data we generated had the greatest positive impact on the CIFAR10 dataset, experiments were conducted on the number of epochs that the model would have to go through. Originally, the model was ran over the course of 150 epochs. 4 While this created images that were recognizable as the classes they were trained to be, it was determined that the images were inaccurate enough that they needed to be remade. After increasing the number of epochs to 200, the images became much clearer and could easily be identified as their respective classes. 5 By providing data that is closer in comparison to the real life counterparts they are trying to represent, we can minimize the potential for classification errors in the future.

IV. Generation

Once the diffusion model learned to differentiate between an image and noise using the CIFAR-10 dataset, we could give it noise and it could hallucinate images. In the reverse pass Gaussian noise is generated and then given to the model. The model iteratively compares the image with the predicted noise and saves the result for the next pass. To keep steps small, the continuous β schedule $\overline{\alpha}$ is used to scale to output of the model as seen in figure 6.

V. Data Augmentation

Now that we are able to produce large amounts of synthetic data, we can use this to influence the dataset that we used to train the model. By adding a large amount of synthetic data to each class of the CIFAR-10 dataset, we can strengthen the data to create a model more effectively. To simplify the process of adding to this dataset, we reused an MLP model that we created for a past project that will be able to classify the images created and sort them into their respective categories within the dataset. To see the difference in the model before

$$\nabla_\theta = \|\epsilon - \epsilon_\theta(\mathbf{x_t}, \mathbf{t})\|^2$$

Fig. 3. Gradient used for training forward pass

Fig. 4. Images generated from Gaussian noise at 150 epochs

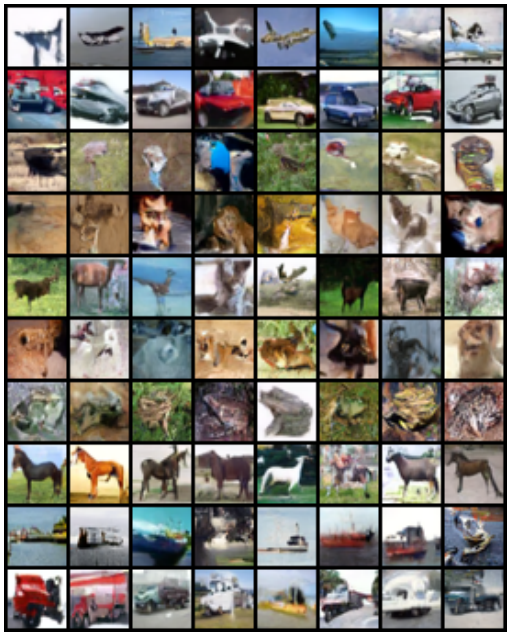

Fig. 5. Images generated from Gaussian noise at 200 epochs

and after augmentation, we took the confusion matrices from each scenario and compared them along with their overall accuracies. The model created prior to augmentation8 had an accuracy of 44%, while the augmented model9 had an accuracy of 46.47%. We found that this increase in accuracy

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

Fig. 6. Reverse pass equation predicting noise in image

was reasonable given the parameters. While these two tests were run at the same learning rate, our other trials showed that we should expect an even greater improvement in accuracy if the learning rate were to be raised. In seeking alternate ways to further increase the accuracy, it was hypothesized that replacing the MLP model with a CNN model would produce positive results. Unfortunately we were unable to test this hypothesis, as it was determined that we would not be able to properly implement and test this in the required time.

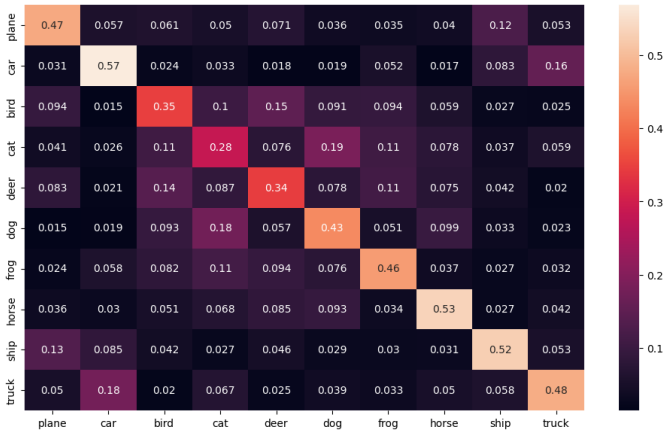

Fig. 7. Confusion matrix from P1 mlp model

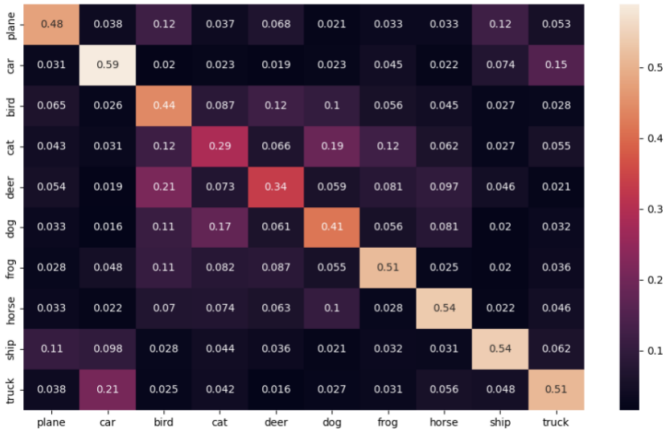

Fig. 8. Confusion matrix after augmentation

When deciding how large of a batch to create for the dataset, we settled on using 100 as it was easily divisible by the 5000 images we would need for each classification. Originally the batch size was 80, however, we realized that by increasing it to 100, we would be simplifying our options for sample sizes.

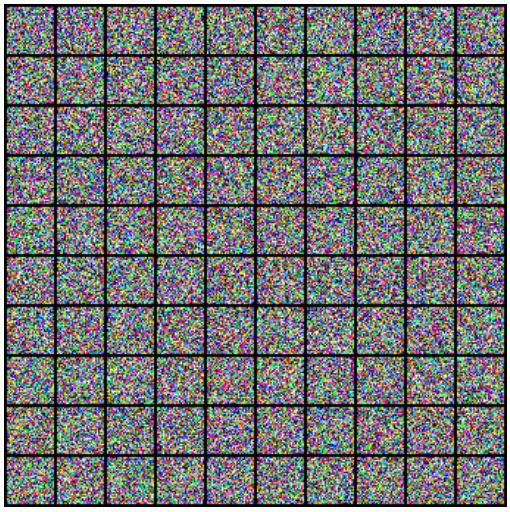

Fig. 9. Gaussian noise generated for default parameters, 200 epoch, and a batch size of 100

Fig. 10. Images generated using default parameters, 200 epoch, and a batch size of 100

With this, it is easier to have each batch produce the same number of images, as there will be no remainder at the end.

VI. Cosine Schedule and Parameter Ablation

As a further step to improve upon our data synthesis, the linear β_t schedule within diffusion model was replaced with a cosine schedule. [2] A cosine schedule takes small steps early on in the corruption process and then speeds up later in the process. When an image is noised through a linear schedule, the original image is usually unrecognizable after half of the steps, causing the last few steps to be redundant. When the model learns a little from each step rather than getting multiple readings of just noise, it is able to produce images of much higher quality with less training.

In order to fully understand the benefits of using a cosine scheduler, the model was trained once with the original linear scheduler, and again using the cosine scheduler. Early iterations such as figures 11 and 12, were created using channel multiplications of [1,2,3,4] and [1,2,2,2]. After several iterations, and changing parameters such as channel multiplications, channels, and grad clip results continued to be poor. 15 uses a channel multiplication of [1,1,2,2,4,4] which resulted in slightly better results. These poor results seemed consistent with our peers who also had insufficiently tuned their hyperperameters.

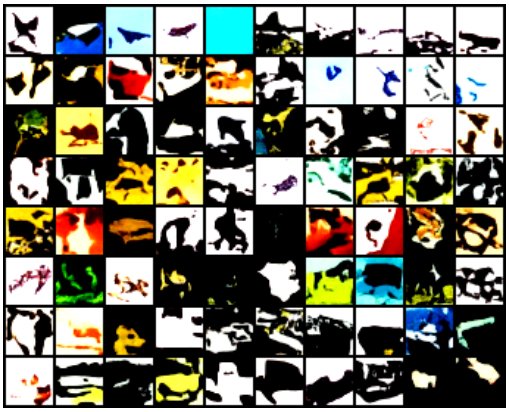

Fig. 11. First iteration of cosine schedule

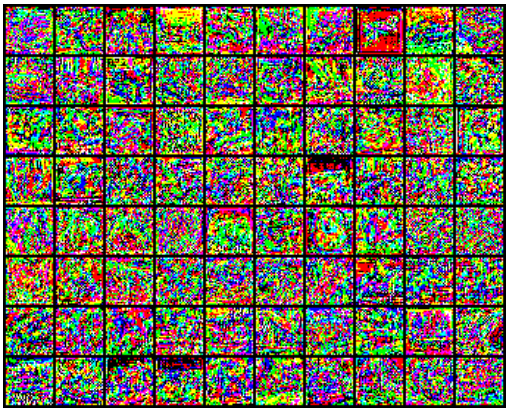

Fig. 12. Second iteration of cosine schedule

VII. Conclusion

Diffusion models are a great means of generating a large amount of training data. After conducting tests on the effectiveness of a model that includes said training data, we were able to determine that the additional data can certainly be of help when training a model, provided that the optimal parameters and models are included in the process. As for the inclusion of a cosine schedule, its usefulness is also dependent on being subjected to the proper conditions, as having ill suited hyperparameters can result in data that is far from usable.

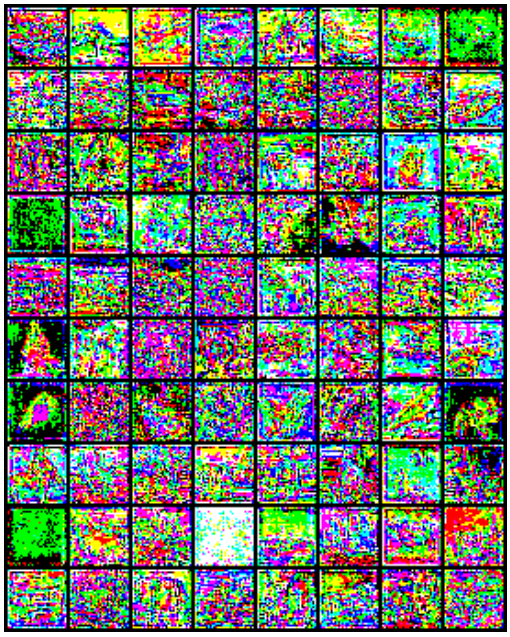

Fig. 13. Fourth iteration of cosine schedule

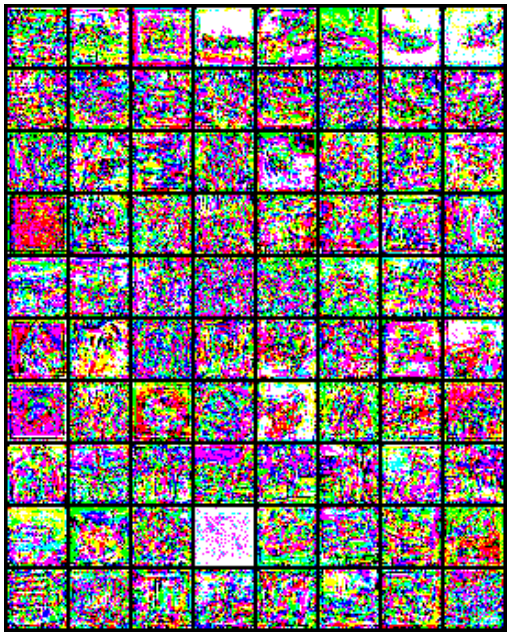

Fig. 14. Sixth iteration of cosine schedule

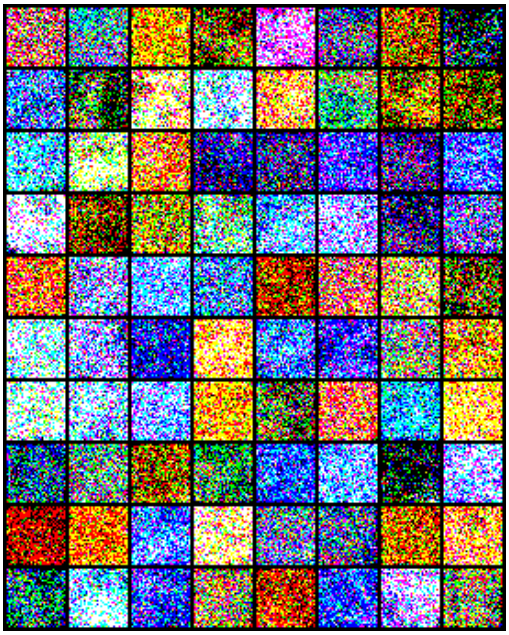

Fig. 15. Seventh iteration of cosine schedule

REFERENCES

[1] A. Nichol and P. Dhariwal, "Improved Denoising Diffusion Probabilistic Models," arXiv:2102.09672 [cs, stat], Feb. 2021, Available: https://arxiv.org/abs/2102.09672

[2] J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models." Available: https://proceedings.neurips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf

[3] Outlier, "Diffusion Models — Paper Explanation — Math Explained," www.youtube.com, Jul. 06, 2022. https://www.youtube.com/watch?v=HoKDTa5jHvg (accessed May 22, 2023).