TRAINING DIFFUSION MODELS

Deep learning - fall semester

Authors
Tamásy András - EA824Y
Takáts Bálint - PUWI1T
Hujbert Patrik - D83AE5

Contents

1 Introduction

In this project, we delved into the practical application and training of Denoising Diffusion Probabilistic Models (DDPMs), guided by the pioneering studies of Jaiming et al. [1] and Ho et al. [2]. Our investigative framework was anchored on the Caltech Birds dataset [3], a comprehensive collection featuring high-resolution images of common bird species found across the United States. This dataset, encompassing over ten thousand images spanning two hundred species, is enriched with detailed annotations and bounding boxes, offering a robust basis for our analysis.

Our approach involved a thorough experimental data analysis paired with strategic data augmentation to enrich the dataset further. Central to our methodology was implementing the renowned U-net architecture [4], enhanced with self-attention mechanisms. This allowed us to leverage the intricate patterns within the dataset effectively.

Throughout our model training process, we utilized Weights and Biases for meticulous experiment tracking. This not only provided us with real-time insights into the model's performance but also allowed for a streamlined and efficient iterative improvement process. Our work aims to contribute to the growing body of knowledge in machine learning, specifically in the application of DDPMs to complex image datasets.

The inspiration and the backbone of our code were built on the Keras documentation [5], which we rewrote in torch. We also built uppon the work of András Béres on generating flowers [6].

In the development of our project, we utilized advanced tools such as ChatGPT and Grammarly, which played a pivotal role in various aspects including code generation, visualization, grammar enhancement, effective phrasing, text production, and LaTeX formatting. These tools significantly enhanced the quality of our work, proving to be an indispensable component of the project.

2 Exploration

Our dataset, illustrated in Figure 1, exemplifies the diversity and richness of the Caltech Birds dataset. As shown in the figure, the dataset provides high-quality, detailed images of various bird species for our analysis.

To maintain consistency and focus on the primary subjects of our study, we preprocess the images from the Caltech Birds dataset. This preprocessing involves cropping the images using the provided bounding boxes to primarily focus on the birds, effectively minimizing background distractions. Subsequently, we resize the images to a uniform dimension of $64 \times 64 \times 3$. This standardization is crucial for our analysis, as it ensures that all images are of comparable scale and resolution, thereby maintaining the high quality of the dataset. Figure 2 shows a subsample of the processed images.

The preprocessing stage concludes with the standardization of the images. This critical step involves adjusting the pixel values of each image to have a mean of 0 and a standard deviation of 1.

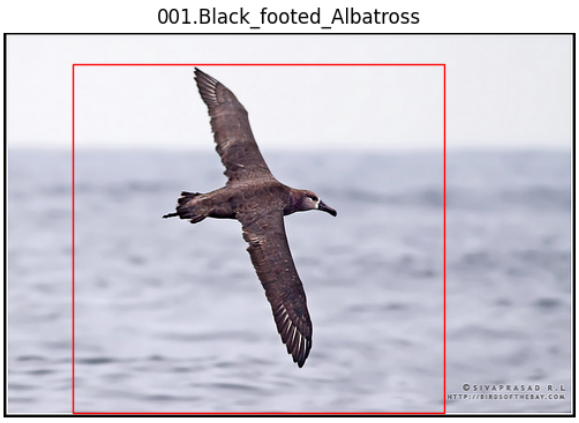

Figure 1: A sample from the Caltech dataset

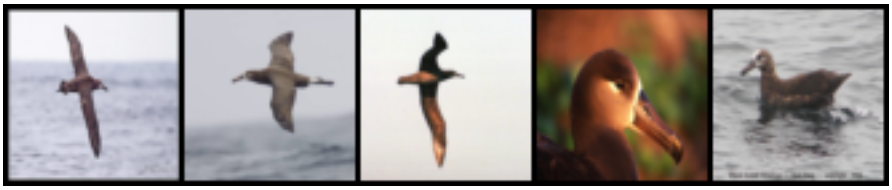

Figure 2: Four samples from the dataset after preprocessing.

3 Implementation

We compiled Table 1 with the most important files and notebooks to give a quick overview of the codebase.

File Name	**Description**
config.py	Contains all configuration settings and parameters required for the project.
create_metadata.py	Responsible for generating and saving metadata in the form of JSON files.
dataloader.py	Handles the loading of data for the diffusion model.
ddpm.py	Implements the Denoising Diffusion Probabilistic Model (DDPM), including core functionalities for training and sampling.
main.py	The main entry point of the application, handling command-line arguments and model initialization.
modules.py	Houses architectural modules for the U-Net model used in the diffusion process.
utils.py	Contains utility functions used across the project.

Table 1: Summary of Python and related files in the Project

We think that the most important parts of our codebase are the network definition, the metrics and the sampling process. For sampling, we use an algorithm based on Ho et al.'s work [2].

We train the network with the hyper-parameters described in Table 2:

Our training loss curves show a steady convergence shown in Figure 7 in the appendix. We also observe high GPU utilization throughout the trianing.

Hyperparameter	Value
batch_size	1
epochs	200
lr	0.0003
beta_start	0.0001
beta_end	0.02
noise_steps	1000
ddim_sample_step	5

Table 2: Key Hyperparameters for the Model

Completing our training on the Google Cloud Platform (GCP) was a tough achievement, especially considering the prevalent GPU shortage that posed significant challenges. Fortunately, we were able to obtain access to a P100 GPU, which we utilized to the fullest extent possible under the circumstances.

We logged the whole training duration to Wandb and we also created periodic samples from the network. Figure 3 illustrates the output after 1000 steps.

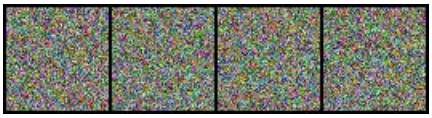

Figure 3: Random noise at the beginning of training.

Training the network took surprisingly long. Even after 100.000 steps were not perfect. This is illustrated in Figure 4. These samples are from our initial training run for Milestone 2. The reason training took so long is that there was a bug in our code. After fixing it training became much faster and our results were way better too.

Figure 4: Samples after 100.000 steps.

The Backward Denoising Process works by removing noise from the original image over 1000 steps. However, the denoised image is only saved at every 100th step in Figure 5. This results in 10 snapshots of the noise reduction process. Therefore, we observe the noise removal over these 10 steps.

For our longest train, we let the network learn for over 28.000 steps. Figure 6 presents our results. To reach this level of accuracy we had to train our network for over 8 hours on a P100 GPU on the GCP.

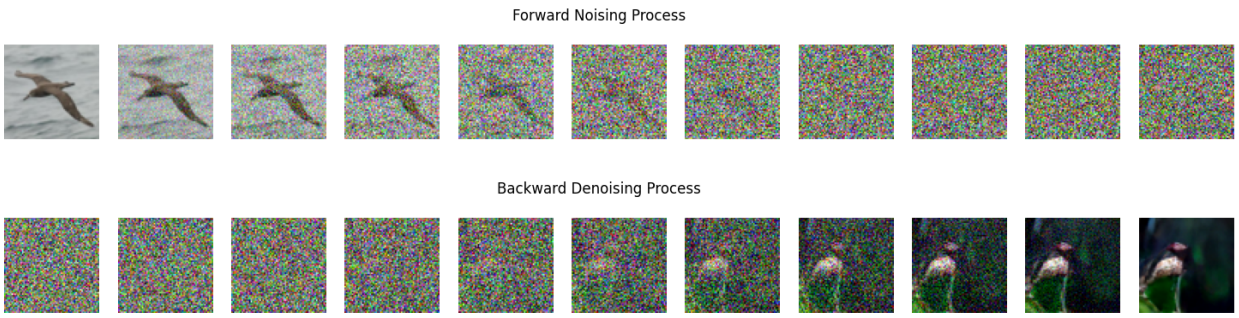

Figure 5: Forward and backward noising process

Figure 6: Best training run after 28.000 steps.

References

[1] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," 2022.

[2] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020.

[3] C. W. at al., "Caltech birds," Tech. Rep., 2011.

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.

[5] Keras, "Denoising diffusion implicit models." [Online]. Available: https://keras.io/examples/generative/ddim/

[6] B. András, "Clear diffusion in keras." [Online]. Available: https://github.com/beresandras/clear-diffusion-keras

4 Appendix

Training logs.

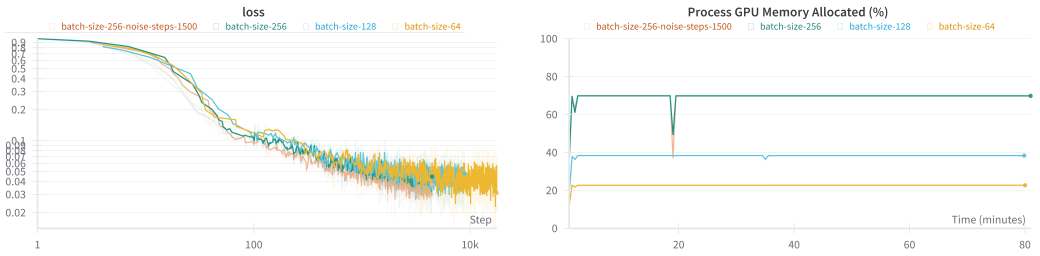

Figure 7: Loss with respect to different Hyperparameters and GPU memory utilization with different batchsizes.

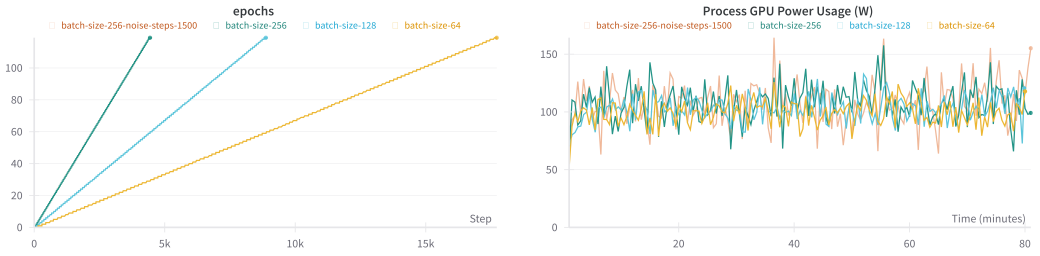

Figure 8: Steps with different batchsizes and power usage.