Controlling Poisson Flow Generative Model

Implementing a class conditional generative model

Ioannis Georgiades

Supervisors: Lydia Chen, Zilong Zhao

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 25, 2023

Name of the student: Ioannis Georgiades
Final project course: CSE3000 Research Project
Thesis committee: Lydia Chen, Zilong Zhao, Anna Lukina

Abstract

With the following paper we are planning to present and explore the possibilities of the the newly introduced Poisson Flow Generative Model (PFGM). More specifically, this work aims to introduce the Conditional Poisson Flow Generative Model (CoPFGM), which by extending the existing repository of the PFGM, it will be able to be trained in a way that allows for conditional image sampling. The work aims to provide a more modular solution that can be easily adjusted for multiple data sets, including custom, as well as datasets taken directly from large Python libraries such as PyTorch and TensorFlow. Our proposed CoPFGM consists of two steps: (i) modifying the input of underlying UNet and (ii) modifying the loss function. More specifically, for (i) we have augmented the input channels of every image with one-hot-like class conditional images, and about (ii) we are introducing an updated loss function which incorporates the Cross-Entropy Loss of the generated images during training. The proposed model is tested against 2 datasets, the MNIST, and the Dilbert dataset, the latter, consists of 1100 custom images of the faces of 6 characters taken from the Dilbert Comic-Strip. The proposed model will be tested and presented in the form of an Ablation Study, with which, we show the conditional behavior of the channel augmentation, and the image improvement in terms of class representation with the Cross-Entropy loss.

1 Introduction

Conditional generative models, compared to unconditional ones, as the name suggests, can generate/sample images that satisfy a certain condition. Generative models are a powerful class of models capable of learning the underlying distribution of the dataset. This allows for image generation that is similar to the training data. Conditional image a generation has various benefits, including and not limited to efficiency and quality. Efficiency as less time and resources are required during sampling process to get images from the desired class. Quality, due to the fact that conditional generative models can often generate higher-quality samples than unconditional models. This is because the conditioning information provides additional structure that can help the model to generate better samples. One popular example in the scope of conditional generative models are the *Conditional Generative Adversarial Nets (CGANs)*[1], which acted as the basis for a lot of subsequent models[2 3 4 5]. In a few words, CGANS, by incorporating the information of the label both at the generator as well as the discriminator, they were able to generate images that belong to a certain class with high quality.

Deep generative models are slowly becoming part of the everyday world. From the image[6 1 7 8] and text[9 10] up to and not limited to audio[11 12] generation, their quality can be compared to and sometimes exceed the product of professionals across different disciplines[13]. There are two types of generative models: deterministic, e.g., GANs, and probabilistic generative models, e.g., Diffusion[14 15 1] and Poisson Flow Generative Model. Even though GANs have been around for a longer time than Diffusion models, the latest has proven to beat GAN on Image Synthesis in both terms of quality, training and sampling time[16].

While other Diffusion models account for conditional image generation via embeddings or simply the use of labels, Poisson Flow Generative Model initial repository does not directly address such needs. Therefore, the purpose of this paper is to introduce the concept of conditional sampling images using the Poisson Flow Generative Model (PFGM)[6 17] and address the following questions.

1. How can we introduce the content of the label of every image to the model/underlying neural network?
2. How should the existing loss function needs to be modified in order to account for conditional image generation.
3. What changes need to be made in terms of the overall code structure to support training such a conditional model?

The rest of the paper is structured as follows. In Section 2, we will present all the related work and background information regarding this paper. Section 3 will analyze the Methodology, including the proposed solution in detail. In Section 4, the Evaluation of the findings and the results and a small discussion about the the whole project will be given. Following that, We will mention the Responsible Research related to our project, and lastly, we will make some conclusions and propose some future work for this project.

2 Background and related work

In this section we will go over some important ideas and concepts which later are going to be utilized for the methodology section. Including the PFGM model itself, and how is the Loss Function derived and what it represents, as well as the formula for the Cross-Entropy Loss.

2.1 Poisson Flow Generative Models Overview

The idea underlying the PFGMs[6] draws parallelism with its predecessor, the Diffusion Model[15]. While the latest uses principles in thermodynamics to add noise to images and then try to 'de-noise' them, PFGMs are treating the pixels as charged particles and by borrowing laws from electrostatics, are able to corrupt the images with random noise. More specifically, as the name suggests, using the Poisson Equation the authors were able to express the particle dynamics in a Poisson field, with both a theoretical and an empirical equation. Eventually by utilizing backward Ordinary Differential Equation solvers, they were able to generate samples from the estimated underlying distribution. Following, we dive deeper into the PFGM, especially how the loss the function is derived, which loss function will be updated in our proposed solution.

[1] https://stablediffusionweb.com/

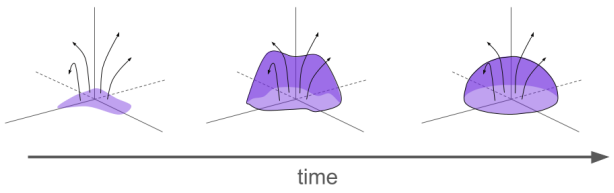

Figure 1: Illustration of the positive Poisson Field progression over time presented in O'Connor's article about explanation of PFGM[20].

2.2 Deeper dive into PFGMs

Poisson Equation

Poisson's equation, named after French physicist Siméon Denis Poisson[18], is a widely utilized elliptic partial differential equations in theoretical physics. It allows for calculating potential fields from known charge or mass distributions, subsequently enabling the derivation of corresponding electrostatic or gravitational fields.

$$\Delta\phi = f \tag{1}$$

where Δ is the Laplacian operator usually rewritten as $\nabla^2 \equiv \sum_{i=1}^{N} \frac{\partial^2}{\partial x_i^2}$, ϕ is called the *potential* and f the *source function*. The authors of the original paper utilize the equation in the scope of the electrostatic theory[19], where $f = -\rho(x)$ which is interpreted as the electric charge density. Therefore the equation 1 becomes:

$$\nabla^2\phi(\mathrm{x}) = -\rho(\mathrm{x}) \tag{2}$$

The paper that introduces PFGM[6], using an extension of the Green's function [2] also proves that:

$$\phi(\mathrm{x}) = \int G(\mathrm{x},\mathrm{y})\rho(\mathrm{y})d\mathrm{y} \tag{3}$$

$$G(\mathrm{x},\mathrm{y}) = \frac{1}{(N-1)S_{N-1}(1)} \frac{1}{\|\mathrm{x}-\mathrm{y}\|^{N-2}} \tag{4}$$

where $S_{N-1}(1)$ is a geometric constant representing the surface area of the unit (N-1) sphere.

Poisson Field

By defining the gradient (Poisson) field $\mathbf{E}$, as $\mathbf{E}(\mathrm{x}) = -\nabla\phi(\mathrm{x})$. $\mathbf{E}$ is the N-dimensional analog of the electric field. In other words, **Poisson Field** describes how forces behave in space, more specifically, in this case the electromagnetism forces from point charges. As the paper proves, if you treat a data distribution as charge distribution, then the Poisson Field over time it is uniformly distributed in a higher-dimensional hemisphere as is shown in figure 1. From 3, $\mathbf{E}(\mathrm{x})$ can be written as:

$$\mathbf{E}(\mathrm{x}) = -\nabla\phi(\mathrm{x}) = -\int \nabla_x G(\mathrm{x},\mathrm{y})\rho(\mathrm{y})d\mathrm{y} \tag{5}$$

$$\nabla_x G(\mathrm{x},\mathrm{y}) = -\frac{1}{S_{N-1}} \frac{\mathrm{x}-\mathrm{y}}{\|\mathrm{x}-\mathrm{y}\|^N} \tag{6}$$

[2]https://mathworld.wolfram.com/GreensFunction.html

So eventually, we want to train a model that is able to simulate the negative Poisson Field, that means by starting from the random noise in a higher dimensional space, it is able to gradually de-noise it and generate a picture that could have been taken from the underlying distribution.

Augmented Poisson Field

When working with a 2-dimensional space, Xu and his team, faced the issue of mode collapse, where all points/particles during the reverse process, went toward a singular point. To resolve the issue, they augmented an extra dimension to the original data, $\tilde{\mathrm{x}} = (\mathrm{x}, z)$, where (Eq. 5) becomes:

$$\mathbf{E}(\tilde{\mathrm{x}}) = -\nabla\phi(\tilde{\mathrm{x}}) = \frac{1}{S_N(1)} \int \frac{\tilde{\mathrm{x}}-\tilde{\mathrm{y}}}{\|\tilde{\mathrm{x}}-\tilde{\mathrm{y}}\|^{N+1}} \tilde{p}(\tilde{\mathrm{y}})d\tilde{\mathrm{y}} \tag{7}$$

Given a set of training data $D = \{x_i\}_{i=1}^n$ the authors defined the *empirical version* of the Poisson field (Eq. (7)) as

$$\hat{\mathbf{E}}(\tilde{\mathrm{x}}) = c(\tilde{\mathrm{x}}) \sum_{i=1}^{n} \frac{\tilde{\mathrm{x}}-\tilde{\mathrm{x}_\mathrm{i}}}{\|\tilde{\mathrm{x}}-\tilde{\mathrm{x}_\mathrm{i}}\|^{N+1}} \tag{8}$$

where $c(\tilde{\mathrm{x}})$ is a constant used numerical stability. The PFGM is based on a neural network f_θ, that aims to calculate the negative normalized field for a huge batch of data by minimizing the following loss function:

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \|f_\theta(\tilde{\mathrm{y}}_i) - \mathbf{v}_{\mathcal{B}_L}(\tilde{\mathrm{x}})\|_2^2 \tag{9}$$

$$\mathbf{v}_{\mathcal{B}_L}(\tilde{\mathrm{x}}) = -\sqrt{N}\hat{\mathbf{E}}_{\mathcal{B}_L}(\tilde{\mathrm{x}}) / \|\hat{\mathbf{E}}_{\mathcal{B}_L}(\tilde{\mathrm{x}})\|_2 \tag{10}$$

where $\mathcal{B}$ is a small batch of images or 'samples.' taken from a larger batch $\mathcal{B}_L$.

2.3 Cross Entropy Loss

Classification is used in this work as well, to guide the model to learning to de-noise the perturbed images in a way that the 'estimated' image belongs to a certain class. But we are not stopping there, we want to 'punish' possible uncertainty, in other words, given a pre-trained classifier, we would like the model not only to de-noise the image in a way that belongs to a certain class according to the classifier , but also to do it with as much certainty as possible. For example a classification of an image that belongs to a class c with probability of 0.70 will be punished more compared to classification with a probability of 0.90. This is the reason we are using Cross-Entropy Loss, as explained in detail below.

Assuming that we have our input as x where x has size of $(batchSize, numClasses)$ where each image in the batch has associated probabilities of being in each class, with the requirement:

$$\forall i \in [1, batchSize] \sum_{j=1}^{Classes} x_{i,j} = 1 \tag{11}$$

Moreover, we assume y is the target vector, which has the same size as x with the only difference

$$\forall i \in [1, batchSize] \exists c \in Classes | y_{i,c} = 1 \tag{12}$$

$$\forall c' \neq c, y_{i,c'} = 0 \tag{13}$$

With (Eq. 11 and 12) in mind, can be defined *Cross Entropy Loss* as:

$$\mathrm{L}(x, y) = \frac{\sum_{n=1}^{N} l_n}{N} \tag{14}$$

$$l_n = -\sum_{c=1}^{C} w_c log \frac{exp(x_{n,c})}{\sum_{i=1}^{C} exp(x_{n,i})} y_{n,c} \tag{15}$$

In the scope of this research project, we assume equal weights $w_i = 1$ for each class as they are equally distributed. Moreover, implementation for such a function already exists, and we will be using the implementation provided by **PyTorch** for such calculation.

3 Methodology

We propose the Conditional Poisson Flow Generative Model, or **CoPFGM**, which relies heavily on the implementation of the existing PFGM[17], and introduces the concept of conditional training and sampling with a combination of two key ideas. First, we augment the channels of each image to include the information of the image's label. Secondly, we update the loss function to include both the existing loss function, as well as the Cross-Entropy Loss of the denoised images, which is calculated with the help of a classifier that is trained before the CoPFGM. Both these concepts will be explained in detail in this section, as well as an explanation of how we obtain the probabilistic model, which is used in finding the cross-entropy loss. At the end, we will be able to conditionally generate images of 6 characters from Dilbert's comic strips as shown in Figure 2, as well as handwritten digits from 0-9, as shown in Figure 3

Figure 2: Combined results of sampling from the CoPFGM trained in Dilbert's dataset for 25k steps, the 6 characters that were included in the dataset.

3.1 Overall Architecture of CoPFGM

Before we dive into the details, it is important for us to explain the updated structure of the proposed model. As shown in Figure 4 we are introducing two new components, the **Classifier** and the image **labels**, and we are updating two more, the **NCSN++**[21] and the **Loss Function**.

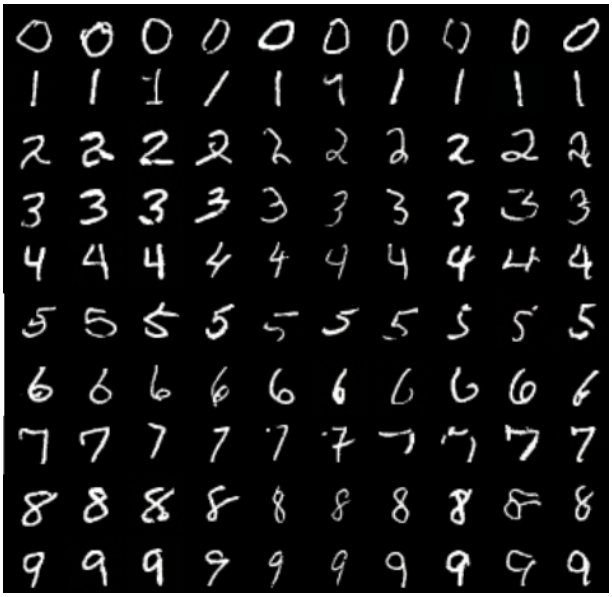

Figure 3: Combined results of sampling from the CoPFGM all the numbers with the model trained on the MNIST dataset for 50k steps.

So starting from the **Classifier**, it is a model that is trained with the same dataset **before** the CoPFGM and it is capable given a new image to probabilistically classify it to the classes of the dataset. Where probabilities for all classes for a single image should add up to 1. To train such classifier easily we would need the **labels** of every input image.

Next, the **NCSN++**[21] is the U-net mainly used in the original paper[6] to calculate the expected negative Poisson Field (see Eq. 5) and then later used for sampling purposes, the exact update will be explained in 3.2. It is important to mention the relation with the **Perturbed samples** part, which, even though it is not modified in our the proposed solution is directly related to the U-net itself. Empirically, the forward Poisson process on an image can be illustrated by adding seemingly random colored pixels. Depending on how much we 'allowed' the forward process to run, we can get from little up to a lot of noise, as shown in Figure 6. *Visually, the purpose of the **NCSN++** is to be able, given an image with added noise, to estimate which pixels were added as 'noise.'*

The **Loss Function** is a function that initially given the output of the above mentioned **NCSN++**, it produced a loss, which our model is trying to minimize over time. Our proposed model would use the outcome of the classifier as well, to provide an updated loss function, which includes the Cross-Entropy loss. Which then our optimizer will try to minimize by changing the parameters of the **NCSN++**. We will go over the updated loss function later in this section.

3.2 Augmenting input channels with label information

The first step to the proposed solution is the introduction of the label information into the neural network architecture that is used to find the noise for a given image. The authors of the original paper adopt the NSCN++/DDPM++ architectures[22]

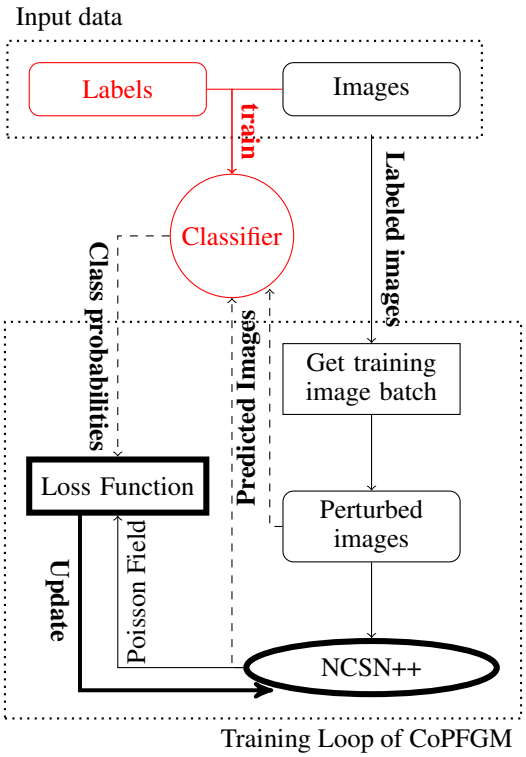

Figure 4: Updated code structure regarding training CoPFGM, which includes the classifier and updated loss function.

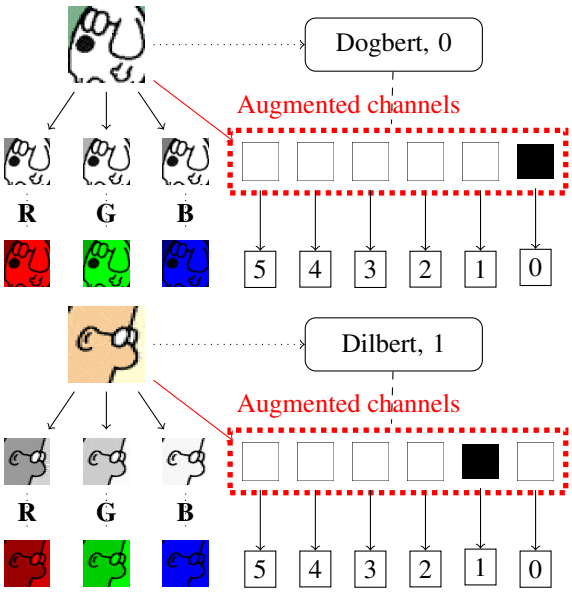

Figure 5: Visualization of the augmented channels for images of Dogbert and Dilbert. All channels are grids, with values ranging from 0 up to 1; for the RGB channels, we are showing both the grayscale interpretation and the actual color. For the augmented channels that correspond to either only white or black pixel grids.

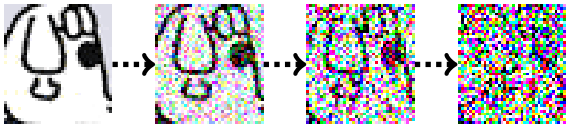

Figure 6: Perturbed image of Dogbert (character in Dilbert's comics) at different not adjacent time steps.

as the backbones for their model. One downside of this architecture is that it only supports images with square size and dimensions to be the power of 2. Which, unfortunately, rules out a lot of ways used by Conditional GANs[1] like to append the information on the label in the form of one-hot tensor [3] at the end of the existing image since that would be something that changes the image dimensions and the network could not use the input. Having this restriction in mind, as well as the flexibility of NSCN++ to accept an arbitrary number of channels for every image, the proposed model augments extra N channels to the existing (usually 1 or 3) channels of the image. Where N is the number of classes that exist in the dataset that we are seeking to conditional sample images from.

But the question remains, given the augmented channels, now how the label's information is incorporated. Before answering that question is important to understand the meaning of a channel for an image and what it represents. Every channel of an image creates its grid of values ranging from $[0, 1]$ where RGB channels, for example, are scaled- down values of the corresponding RGB value of each pixel. Now regarding the last N channels, we are utilizing a similar the idea as the one-hot encoding, with the only difference instead of a single one, we have a single channel with a grid full of 1s with the same size as the original image and the rest augmented channels are just 0s; see Figure 5 for a visual example of the method mentioned above.

3.3 Updating Loss Function with Cross Entropy Loss

We want to guide the neural network to denoise the images towards a certain outcome. For that purpose, we will use the cross-entropy loss, as explained in 2.3, where we defined the Cross-Entropy loss $\mathrm{L}(\mathrm{x}, \mathrm{y})$ between the predicted probabilities per class per image as x and the actual labels y in a one-hot-like format, one for every image. Moreover, we are denoting the initial loss function with $\mathcal{L}(\theta)$ (see Eq. 9) where θ represents the parameters of the NCSN++, mentioned before. With these equations in mind, we propose an updated loss function defined as follows:

$$L'(\theta) = (\beta + \mathrm{L}(\hat{x}, \hat{\lambda})) \cdot \mathcal{L}(\theta) \tag{16}$$

$$\hat{x} = \mathcal{P}(\{\tilde{y}_i + f_\theta(\tilde{y}_i)\}_{i=1}^{|\mathcal{B}|}) \tag{17}$$

Where $\mathcal{P}(\mathrm{x})$ returns a tensor with length equal to the number of classes and each element is the the associated probability that the image belongs to the corresponding class (see Figure 4 at the arrow going from the Classifier to the Loss function). The $\{\tilde{y}_i + f_\theta(\tilde{y}_i)\}_{i=1}^{|\mathcal{B}|}$, are simply all the predicted images formed by adding the negative Poisson field as the U-net

[3] https://en.wikipedia.org/wiki/One-hot

estimates it, and the initially perturbed images. While $\hat{\lambda}$ is the 2-dimensional tensors that contains the true classes of every image in the batch. We also introduce the hyper-parameter β which is used to avoid over-fitting the model with just the cross entropy loss. Fine-tuning this parameter, unfortunately, so far can only be done empirically.

3.4 Obtaining the Probabilistic Model for Cross Entropy Loss

In the subsection 3.3 we used the $\mathcal{P}(\mathrm{x})$ function to retrieve the probabilities that the image x belongs to all the available classes of the dataset, probabilities that, according to the subsection 2.3 they should add up to 1 for every image; in other words, the following relation should hold.

$$\forall i \in [1, batchSize] \sum_{j=1}^{Classes} x_{i,j} = 1$$

Such functionality requires the implementation and training of a classification network/model that, simply by only accepting the denoised image, would output the corresponding probabilities. To implement and use such a model, a small modification on the code structure and 'pipeline.' needed to be made, as shown in Figure 4. More specifically, **before** training the CoPFGM model, we created and trained a classifier which, by using relatively simple architecture and minimal training time, we achieved a model with very high accuracy with minimum 90 percent accuracy. Last but not least, to ensure that the outcome probabilities indeed sum up to 1 for every image, we made use of the **Soft-max** function which is defined as [4]:

$$Softmax(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)} \tag{18}$$

The model used for the Dilbert dataset is represented in much detail in Appendix A, keep in mind that this classifier was specifically implemented to take as input 32x32 images with 6 classes on the dataset. Such model needs to be re-implemented and adjusted to the corresponding dataset if an arbitrary dataset needs to be utilized.

4 Experimental Setup and Results

In this section, we will go over the the environment used for the project, as well as some preparation is needed before running a custom dataset and train a new CoPFGM model. Furthermore, we are going to present the results of our model that was trained on two datasets.

4.1 Environment and preparation

This project was executed on the Google Colab Pro+ platform, leveraging a 40GB GPU and a CUDA-enabled virtual environment, enabling effective training within hours. This paper's related code can be run in any CUDA- powered Torch environment following the instructions provided in the project repository (https://github.com/GeorgiadeG/CoPFGM).

The repository contains configurations and code for two datasets, MNIST and a custom dataset of manually labeled faces from an American Comic Strip, Dilbert. To recreate results with a custom dataset, necessary considerations must be made. The MNIST dataset is directly accessible from the TensorFlow Datasets library, while the custom dataset requires additional data pre-processing.

The project's code can serve as a reference for training a dataset directly from the TensorFlow library or a custom dataset with images and labels. However, if a custom dataset is used, be aware of the complex configuration files involved regarding the data, training phase, sampling, and the neural network. It's suggested to refer to both the PFGM paper[6] and the Score-Based Generative Modeling[21] paper for understanding the hyper-parameters values. This paper primarily reuses the CIFAR-10 dataset's configuration file due to its compatibility with 32x32 images. However, the performance may vary with other models using the same configuration file.

4.2 Evaluation Metrics

When it comes to evaluation, we will use the FID/Inception score. The Fréchet Inception Distance (FID) and Inception Score are metrics used to assess the quality of the generated/sampled images. The Inception Score uses a pre-trained Inception model to classify generated images, then calculates a score based on the predicted label distribution's diversity and confidence. A higher score indicates a better image quality and variety. On the other hand, the FID compares the statistical properties of generated images and real images in the Inception model's feature space. A lower FID score implies that the real and generated images are more alike, indicating better generative model performance.

4.3 Sampling

Once the model is trained, we would like to generate images conditioned on a certain class. The authors of the PFGM paper[6] explain in detail how they make use of Ordinary Differential Equations (ODEs) to map the base distribution to a data distribution, and then use a reverse ODE solver to sample images using the same denoising network NCSN++ (see Figure 4). Now all that is left for us to do, is to reuse the code for sampling from the previous repository, and pass the information of the desired label into the U-net itself in the same way as we did during the training. Then, we are able to generate images from any given class with a single command. Refer the CoPFGM repository for more details.

4.4 Results

Now, it is time to go over the results of our model after running it with the two above mentioned datasets, MNIST and Dilbert. To evaluate the results better, and show that both of the main parts of our proposed solution work, we will go over the results in the format of *Ablation Study*, for both datasets, we will have to evaluate 3 different models, one is the CoPFGM itself, as proposed in the previous section, the second the model will have the channel augmentation but not the updated loss function lastly a model without the channel

[4] https://pytorch.org/docs/stable/generated/torch.nn.Softmax.html

Model	Dataset	FID score *	Inception score*
CoPFGM	Dilbert	1.4930×10^2	2.5904
CoPFGM-Old loss/without classifier	Dilbert	1.4608×10^2	2.9499
CoPFGM-Old U-net/without channel augmentation	Dilbert	1.0077×10^2	2.8267
CoPFGM	MNIST	3.1364×10^2	1.7850
CoPFGM-Old loss/without classifier	MNIST	3.0943×10^2	1.8149
CoPFGM-Old U-net/without channel augmentation	MNIST	2.1500×10^2	2.8150

Table 1: Performance metrics of different models for both datasets. Both the FID * and Inception * score was calculated as the average score from all the classes of the corresponding dataset.

augmentation but with the updated loss function. Note: The implementation for such models was made for evaluation purposes and it will not be included in the repository.

4.5 Ablation Study Results

As is already shown in Figures 2 and 3 and more in detail in Appendix B, our proposed model proved that it can generate images conditioned on a certain class from the underlying dataset, with an **average** sampling time of 5 32 by 32 images per second. But that leaves someone to wonder, what do the 2 main parts of CoPFGM contribute exactly to the outcome? Therefore, that is the reason we are performing an Ablation study to illustrate with figures and metrics how each idea affects the plain PFGM model, leading the way to the proposed CoPFGM.

Generally from Table 1 we can see no model significantly outperforming the others, besides slightly better metrics, both FID and Inception score for both datasets when we excluded the data augmentation. One possible explanation might be due to the reduced complexity of the model since the input of the NCSN++ is tripled for Dilbert and almost quadrupled for the MNIST dataset. But what about sampled images? What images can we generate from all these 3 models? As can be seen for a lot of examples in Figure 7 we can derive some general comments about these models.

Firstly, it is rather obvious that without augmenting channels, we can see no change in behavior in between different sampled classes, as there is no way of 'informing' the NCSN++, what class we are currently training, and which class we want to sample from. But besides that we can see since we are using the updated loss, which partially incorporated the Cross-Entropy loss, the quality of the sampled images especially for the MNIST dataset, is better then the other two models since we guide the model to generate images that have high probability of belonging to any class.

Secondly, it is also evident that the conditional behavior appears with just the channel augmentation, which, indeed makes sense. We are giving information about the class of each image during training, and we are sampling by passing the same information around. On the other hand, it is noticeable more in MNIST and a little less in the Dilbert dataset, the generated images even though they still resemble the conditioned class, they are still not quite good to be confidently classified into the same class.

5 Responsible Research

The foundation of robust and effective scientific research, is not only based on the method used or the results that were obtained, but also on the principles and standards for good research practices. This paper obeys the principles set by the Netherlands Code of Conduct [5] such as scrupulousness, transparency, and independence. which are dictating not only our procedural choices but also our underlying ethos. By adhering to this code of conduct, we aim to contribute to the scientific community, with a paper that is not only technically correct, but also ethically principled.

5.1 Scrupulousness

The term scrupulousness, among other things, refers to the attention to detail and dedication to maintain high standards of accuracy. Our paper explicitly defines and explains the previous related work and the steps taken for the proposed solution. All related work is properly referenced, and by using scientific methods, we were able to pay close attention and exercise the best possible care in designing and reporting our research.

5.2 Transparency

The Netherlands Code of Conduct defines Transparency as "*...ensuring that it is clear to others what data was the research-based on how the data were obtained, what and how the results were achieved...*". In our paper, we properly referenced the source of both of the related datasets, MNIST and Dilbert, even though for the latter one the the dataset is not currently public. Furthermore, in the methodology and the experimental setup section we are properly explaining how the resulting images were generated and sampled.

One of the main branches of transparency is the reproducibility of the project and the results. Usually, when it comes to Artificial Intelligence and Deep learning, recreating the same results is rare due to the randomness that is introduced during the training phase. This paper tries to address

[5] https://www.nwo.nl/en/netherlands-code-conduct-research-integrity

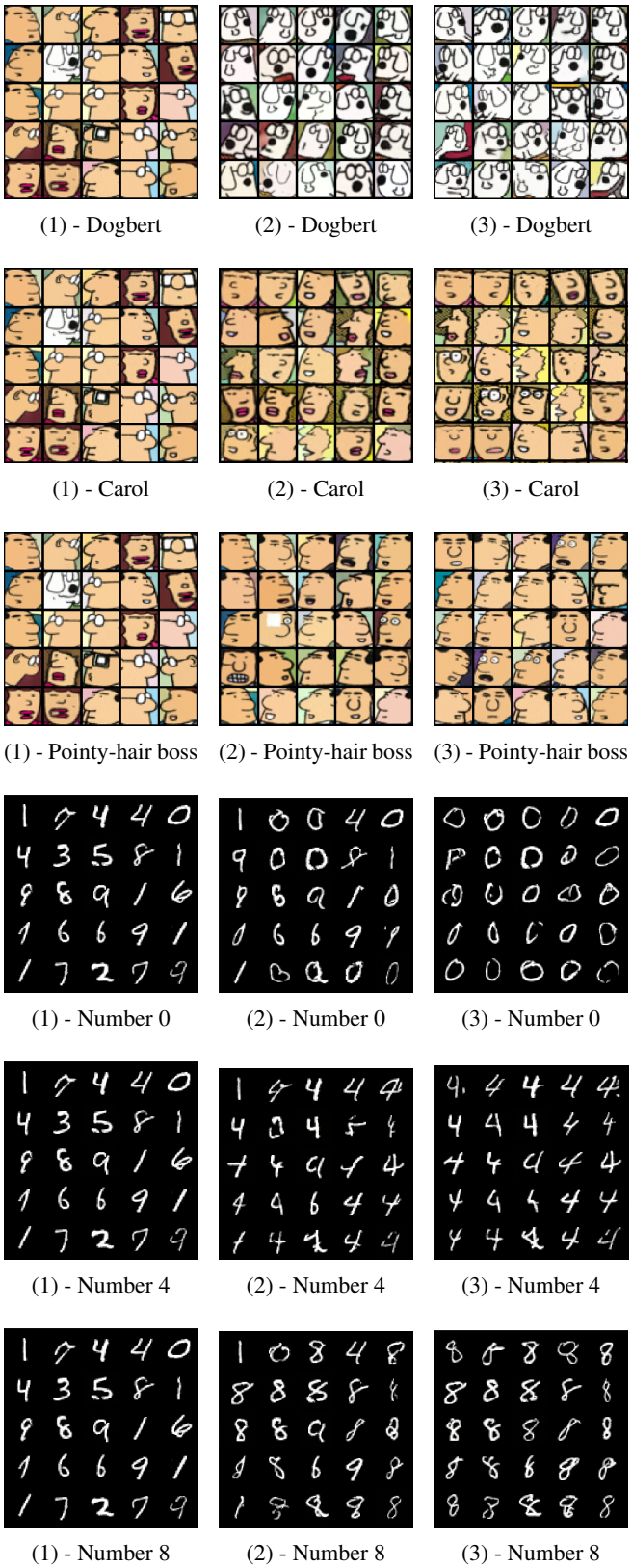

Figure 7: Samples from the Ablation Study. Each column represents a model, from left to right, we have (1) CoPFGM without Channel Augmentation, (2) CoPFGM without Cross Entropy Loss, and (3) the proposed CoPFGM. Each row represents one of the many classes of each dataset (3 random classes per dataset)

such an issue by providing a public repository of the implementation of the proposed solution, as well as an explanation on how to use two different kinds of datasets. Datasets that can be already provided by a Python library, such as Tensorflow or PyTorch, or even a custom dataset. The same results might not be reproducible but at least another CoPFGM model can be generated that can properly and accurately generate conditioned images, assuming proper setup by the user.

5.3 Independence

The term independence is usually associated with the concept of confirmation bias, which is guiding our decisions both for the design and assessment of the data in order to get any positive results. Decisions that can deviate from properly scientific and scholarly standards, which would lead to a false and unethical research without contributing to the scientific community. This report adheres to Independence such all the design choices were guided by adjusted techniques that were used in similar models and architectures, which is making our choice of method independent and scientifically correct.

6 Discussion

In this section, we will discuss stuff related to the results, as well as the bigger picture of this research. We will go over an interpretation of the results as well as the limitations that are related to this research.

6.1 Interpretation of the results

Summarizing the results of the Ablation Study, we can introduce the conditional sampling to the original PFGM, with just the channel augmentation, and enhanced by the addition of the Cross-Entropy loss to the Loss Function. Concluding that a combination of both of these ideas would lead to the creation and training of a model based on PFGM that is able to conditionally sample from each class distribution. As it showed from the sampled images in Figure 7, it is possible with appropriate fine-tuning and enough training to obtain a sort of Conditional model without the need to train a classifier and update the loss function. But the updated loss would help the model to generate images that actually belong to the desired class faster, and more accurately.

6.2 Limitations

Throughout this research, we faced various limitations in terms of factors that might have affected the quality and efficiency of our model and results.

For both datasets, we were dealing with a relatively small amount of classes among with small images themselves. Therefore augmenting the channels did not significantly affect the training time as it increases the parameters of the underlying U-net. So it is possible that the NCSN++ model might get very slow if another dataset with larger images and more classes are used, for example, CIFAR-100, which consists of 100 classes instead of the 10 that MNIST currently does.

As mentioned in section 6.1 even though the CoPFGM seems to sample decently conditional images, there are still imperfect results, or in other words, images generated that

do not fit in the required class. Reasons behind such behavior might vary. First, we have the relatively small training time per dataset per model, secondly it might be due to the fact that we are using mainly the configuration file provided for the CIFAR-10 dataset for the sampling and training the underlying U-Net. In other words, fine-tuning these models might not have been the best.

7 Conclusions and Future Work

In this section we will mention some conclusions by going over both the Research question and sub-questions, and then we will cover the future work and possible improvements that we are proposing for anyone who might want to work furthermore on the CoPFGM.

7.1 Conclusions

This paper aimed to present a new model build on top of the paper and repository of the Poisson Flow Generative Model[6 17], which is able to conditionally sample images from. The proposed method consists of several parts and design ideas, which evidently are the answer to the proposed sub-questions. Starting with the first one, we introduced the information of the class itself (or the so-called label) by assigning each class a number. By having a number associated with each class, we were able to augment the color channels of the images with one-hot-like channels, with $N-1$ of them, to be filled with 0s and one of them with ones (indicating the class). Furthermore, the second sub-question is related to the updated loss function (Eq. 16) which includes the Cross-Entropy Loss of the denoised images, a metric that relates the probabilities are given by the classifier and their ground truth class. Lastly, the third question, which is about significant changes in the code structure in order to obtain such a conditional model from the PFGM, is answered by the subsection 3.1 and Figure 4. In a few words, we simply add to the start of the training 'Pipeline' a step to train a classifier with the same dataset and labels, which is then accessible within the existing loss function during each training step, and used to calculate the updated loss. The information on the labels is passed together with images to the NCSN++, so no change needs to be made.

7.2 Future Work

As it can be seen from the Limitation subsection, CoPFGM has a lot of room for improvement. Firstly, the Channel Augmentation even though proved to be effective it still might delay training and sampling for datasets with way more classes, so we are suggesting of finding an improved way to introduce the information of the label both during and sampling. Something that allows, a further extension to a generative model with full comic strips, conditioning not only on the characters in, but more context about background color, space, and even dialogues. Furthermore, both the hyper-parameter β we have introduced for the updated loss, as well as other parameters which were reused, have definite room for improvement, in a few words, improve the CoPFGM by fine-tuning it to the same datasets or other datasets. Lastly, a little bit more difficult, to switch from class labels, and move on to class embeddings, including text, which might lead to the creation of the next Stable Diffusion!

References

[1] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.

[2] Amjad Almahairi, Sai Rajeshwar, Alessandro Sordoni, Philip Bachman, and Aaron Courville. Augmented CycleGAN: Learning many-to-many mappings from unpaired data. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 195–204. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/almahairi18a.html.

[3] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier GANs. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2642–2651. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/odena17a.html.

[4] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation, 2018.

[5] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks, 2018.

[6] Yilun Xu, Ziming Liu, Max Tegmark, and Tommi Jaakkola. Poisson flow generative models, 2022.

[7] Kripasindhu Sarkar, Lingjie Liu, Vladislav Golyanik, and Christian Theobalt. Humangan: A generative model of human images. In *2021 International Conference on 3D Vision (3DV)*, pages 258–267, 2021. doi: 10.1109/3DV53792.2021.00036.

[8] Christoph Lassner, Gerard Pons-Moll, and Peter V. Gehler. A generative model of people in clothing. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 853–862, Oct 2017. doi: 10.1109/ICCV.2017.98.

[9] Zheng Wang and Qingbiao Wu. An integrated deep generative model for text classification and generation. *Journal of Artificial Intelligence Research*, 2018: 7529286, 2018. doi: 10.1155/2018/7529286.

[10] Yang Li, Quan Pan, Suhang Wang, Tao Yang, and Erik Cambria. A generative model for category text generation. *Information Sciences*, 450:301–315, 2018. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2018.03.050. URL https://www.sciencedirect.com/science/article/pii/S0020025518302366.

[11] A.T. Cemgil, H.J. Kappen, and D. Barber. A generative model for music transcription. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(2):679–694, 2006. doi: 10.1109/TSA.2005.852985.

[12] Simon Rouard and Gaëtan Hadjeres. Crash: Raw audio score-based generative modeling for controllable high-resolution drum sound synthesis, 2021.

[13] Hyunsuk Ko, Dae Yeol Lee, Seunghyun Cho, and Alan C. Bovik. Quality prediction on deep generative images. *IEEE Transactions on Image Processing*, pages 13–14, 2023.

[14] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022.

[15] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

[16] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 8780–8794. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/49ad23d1ec9fa4bd8d77d02681df5cfa-Paper.pdf.

[17] Newbeeer. Poisson flow, 2022. URL https://github.com/Newbeeer/Poisson_flow.

[18] Poisson. Mémoire sur la théorie du magnétisme en mouvement [memoir on the theory of magnetism in motion]. *Mémoires de l'Académie Royale des Sciences de l'Institut de France*, 6:441–570, 1823. From p. 463: "Donc, d'après ce qui précède, nous aurons enfin:".

[19] David J. Griffiths. *Introduction to Electrodynamics*. Prentice Hall, 2005.

[20] Ryan O'Connor. An introduction to poisson flow generative models. Assembly AI, October 2022. URL https://www.assemblyai.com/blog/an-introduction-to-poisson-flow-generative-models/.

[21] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

[22] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations, 2021.

A Graphical representation of the classifier

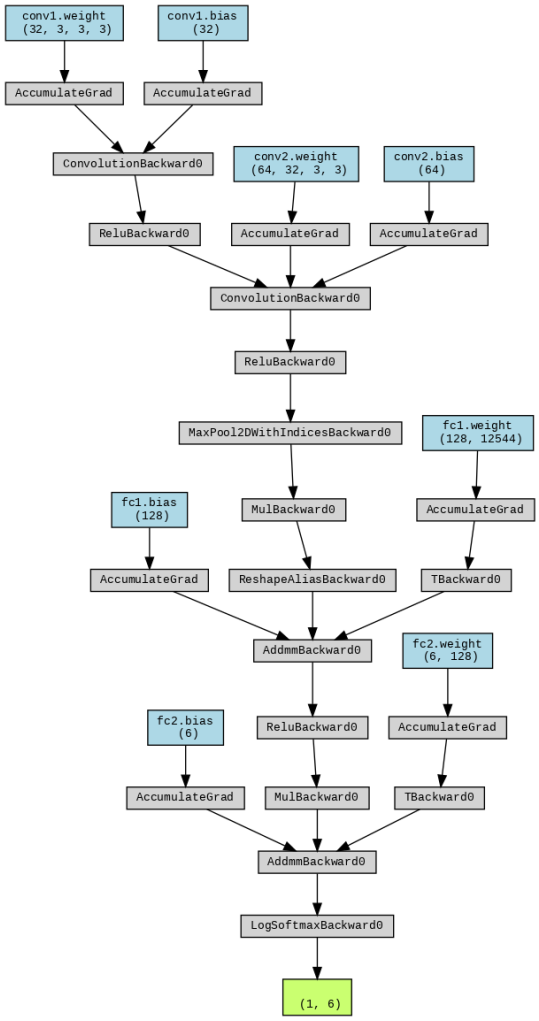

Figure 8: Diagram generated with PyTorch Visualisation library, of the model, which takes as input 32x32 images and outputs 6 value, probabilities for each class.

B Full sampled images of MNIST digits

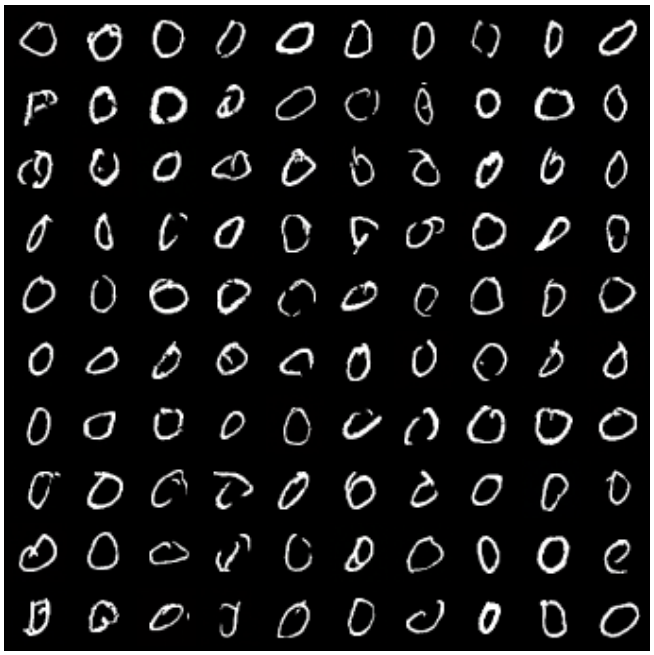

Figure 9: 100 sampled images of the number 0 with the CoPFGM after 100k iterations.

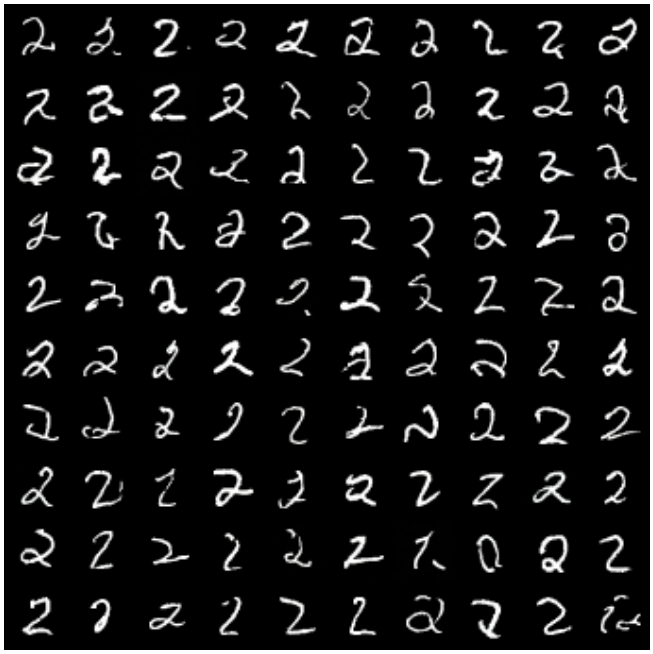

Figure 11: 100 sampled images of the number 2 with the CoPFGM after 100k iterations.

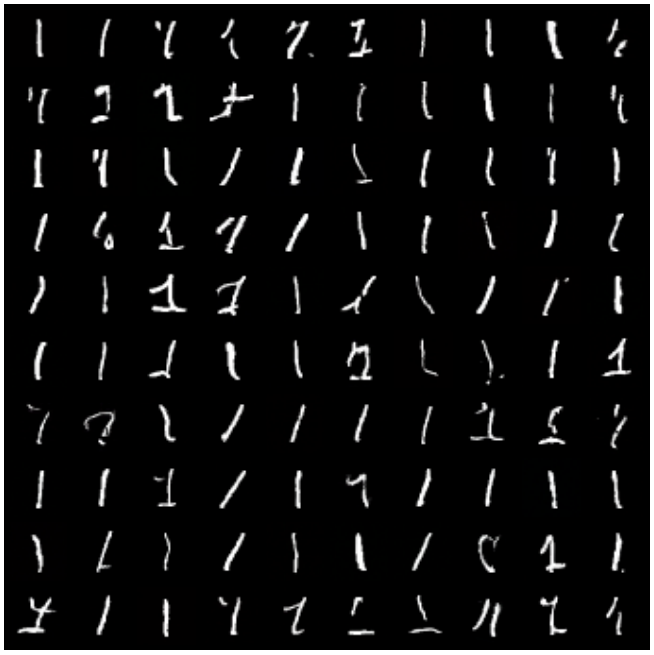

Figure 10: 100 sampled images of the number 1 with the CoPFGM after 100k iterations.

Figure 12: 100 sampled images of the number 3 with the CoPFGM after 100k iterations.

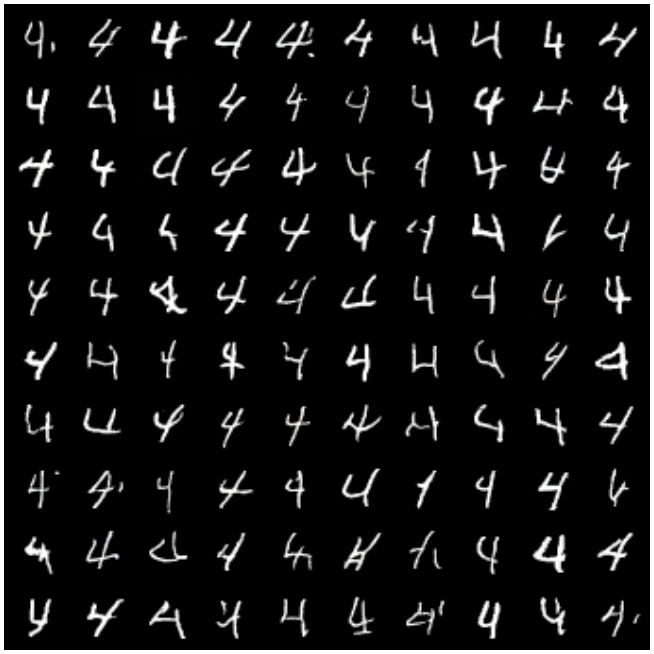

Figure 13: 100 sampled images of the number 4 with the CoPFGM after 100k iterations.

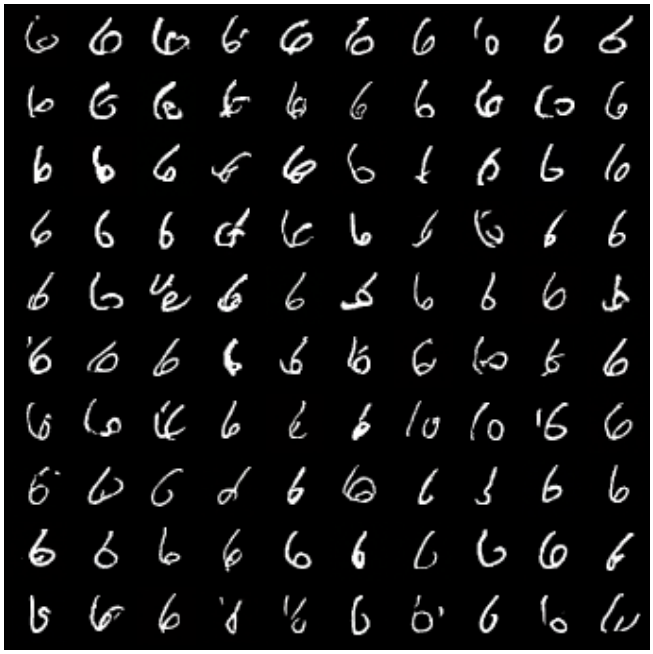

Figure 15: 100 sampled images of the number 6 with the CoPFGM after 100k iterations.

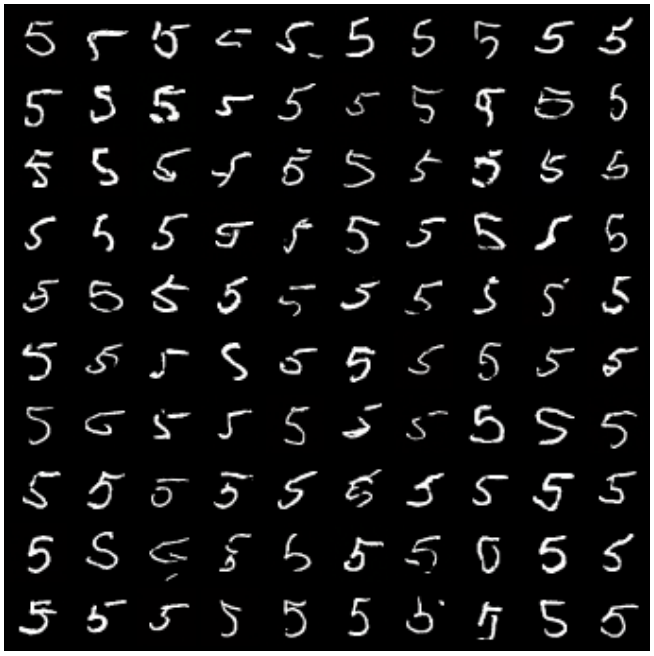

Figure 14: 100 sampled images of the number 5 with the CoPFGM after 100k iterations.

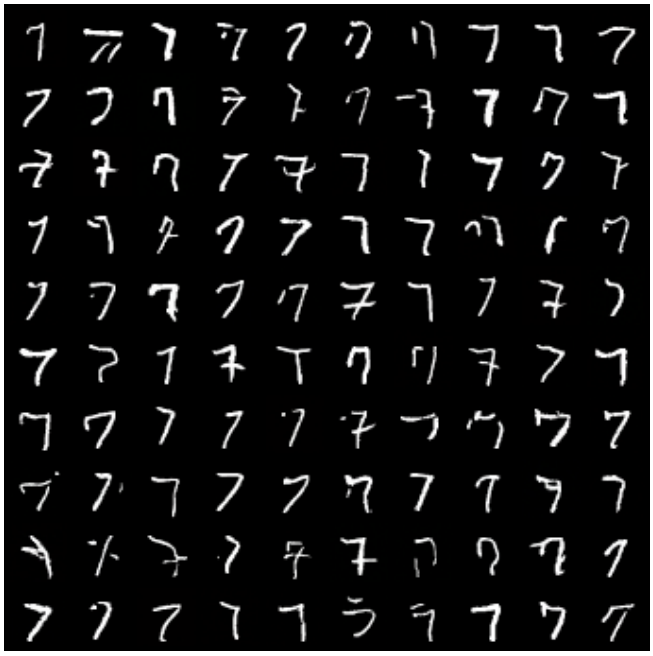

Figure 16: 100 sampled images of the number 7 with the CoPFGM after 100k iterations.

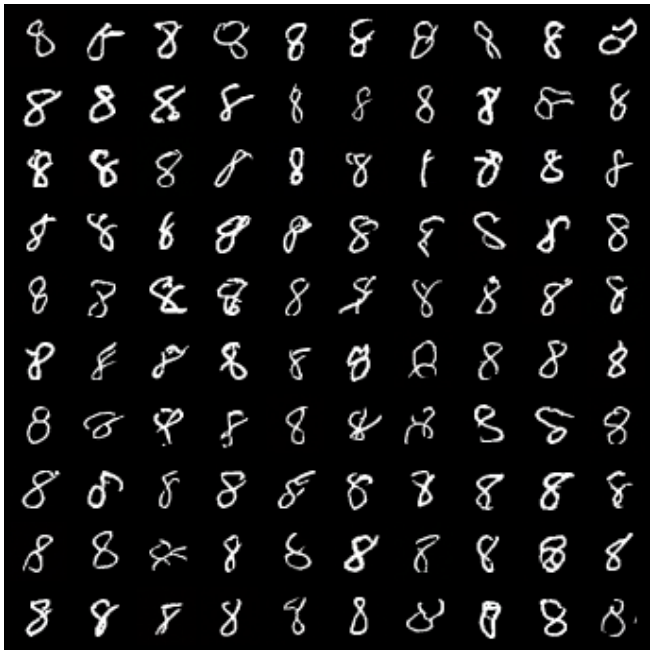

Figure 17: 100 sampled images of the number 8 with the CoPFGM after 100k iterations.

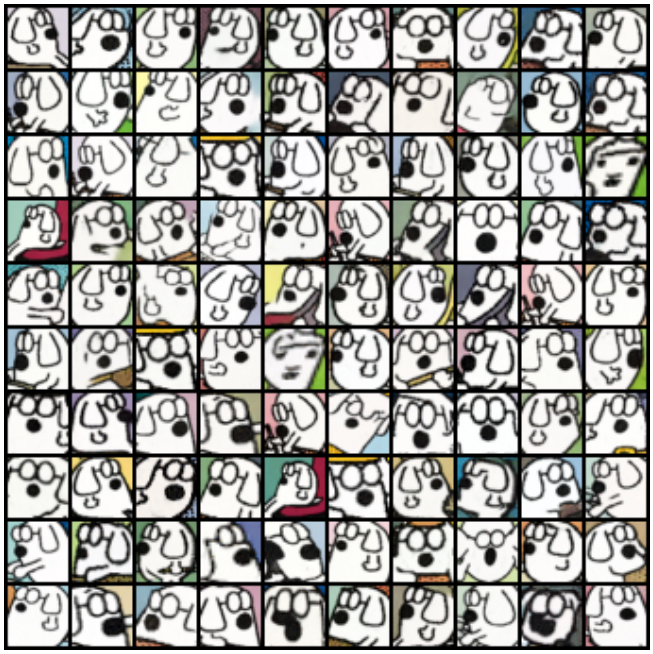

Figure 19: 100 sampled images of Dogbert with the CoPFGM after 13k iterations.

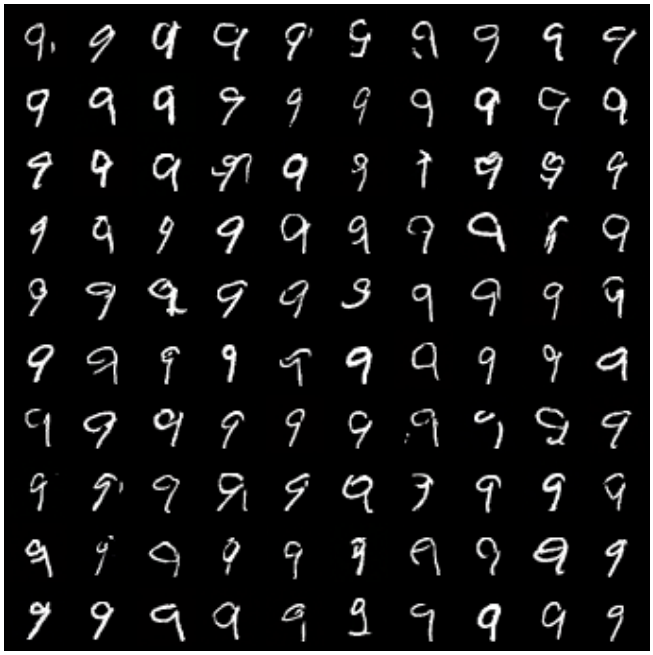

Figure 18: 100 sampled images of the number 9 with the CoPFGM after 100k iterations.

Figure 20: 100 sampled images of Dilbert with the CoPFGM after 13k iterations.

Figure 21: 100 sampled images of Carol with the CoPFGM after 13k iterations.

Figure 23: 100 sampled images of Pointy-Haired Boss with the CoPFGM after 13k iterations.

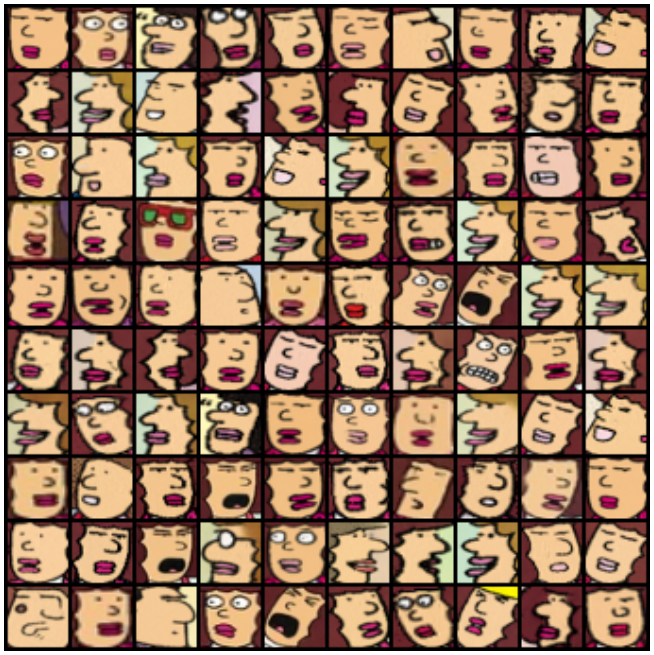

Figure 22: 100 sampled images of Alice with the CoPFGM after 13k iterations.

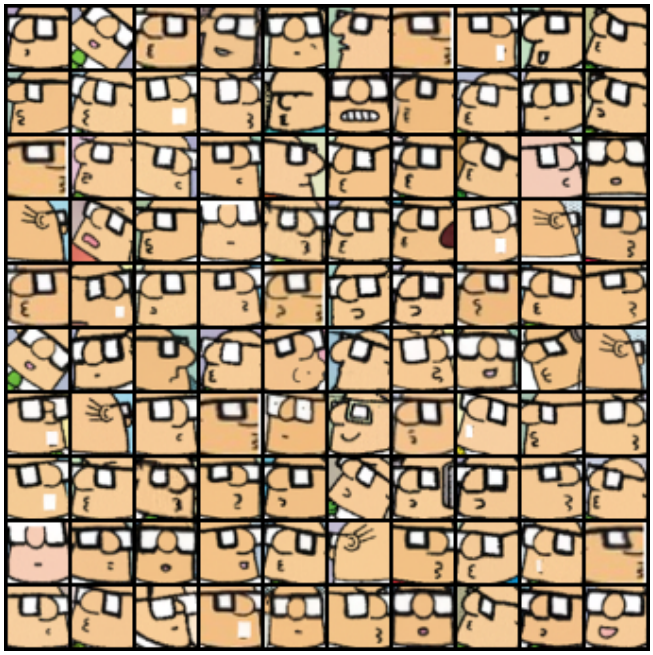

Figure 24: 100 sampled images of Wally with the CoPFGM after 13k iterations.