ECE1512 Project A Report

Rémi Grzeczkowicz
MScAC Student
University of Toronto - Department of Computer Science
Student Number: 1010905399
remigrz@cs.toronto.edu

CONTENTS

I. Introduction

II. Task 1 - DataDAM

This part relies on the paper [4].

A. Part 1 - Questions anwsering

(a) In this paper, the purpose of using Dataset distillation is to reduce the training cost while preserving the performance of the model.

(b) The advantages of their methodology over state-of-the-art are:

- It achieved unbiased representation of the real data distribution.
- It does not rely on rely on pre-trained network parameters or employ bi-level optimization.
- It has a reduced memory cost with a lower run time thanks to the fact that DataDAM does not use an inner-loop bi-level optimization.
- It outperformed other distillation methods except for one case where Matching Training Trajectory (MTT) performed better on CIFAR-10 with 10 Impage Per Class (IPC). MIT got an accuracy of $56.5\% \pm 0.7$ while DataDAM got $54.2\% \pm 0.8$.

(c) The novelty provided by this paper is the use of attention in data distillation. Indeed it has been used in knowledge distillation but never in dataset distillation.

(d) The methodology is as follows:

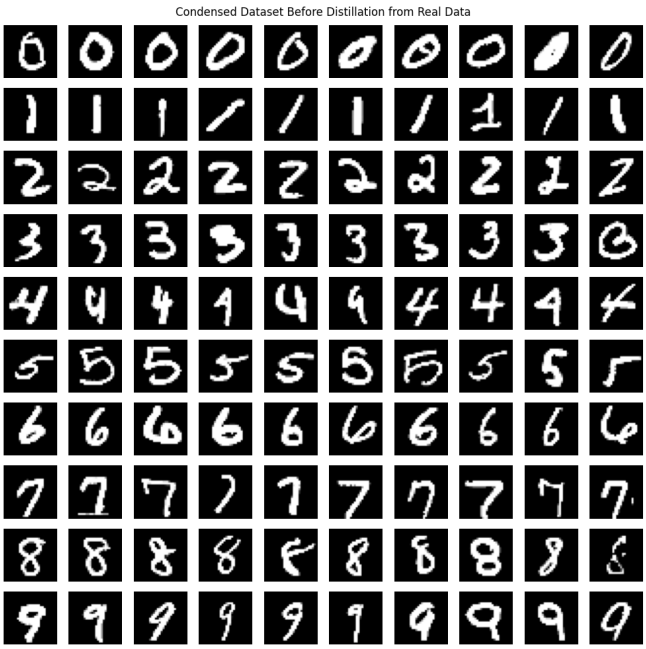
(1) Initialize a synthetic dataset $\mathcal{S}$ either using random noise or sampling from the original training dataset $\mathcal{T}$.
(2) For each class k a batch B_T^k of real images and a batch B_S^k of synthetic images are sampled from $\mathcal{T}$ and $\mathcal{S}$ respectively.
(3) Then a neural network ϕ_θ is employed to extract features from the images. The network have different layers, each creating a feature map. This multiple feature maps allow to capture low-level, mid-level and high-level representations of the data.
(4) Using the feature maps of each layer, the Spatial Attention Matching (SAM) module generates an attention map for real and synthetic images. The attention map is formulated as $A(f_{\theta,l}^{T_k}) = \sum_{i=1}^{C_l} |(f_{\theta,l}^{T_k})_i|^p$ where $(f_{\theta,l}^{T_k})_i$ is the i-th feature map in the lth layer, C_l is the number of channels and p is a parameter to adjust the weights of the feature maps.
(5) The attention maps for both datasets are then compared using the loss function $\mathcal{L}_{SAM}$.
(6) The output of the network for each dataset is also compared using the loss function $\mathcal{L}_{MMD}$ based on the Maximum Mean Discrepancy (MMD).
(7) The total loss is then given by $\mathcal{L} = \mathcal{L}_{SAM} + \mathcal{L}_{MMD}$.
(8) Then $\mathcal{S}$ is updated such as $\mathcal{S} = arg\min_{\mathcal{S}} \mathcal{L}$.

(e) DataDAM could be used in machine learning for continual learning by providing an efficient memory management method by storing the synthetic data in the memory instead of the real data. This allows for a better memory usage and a lower computational cost. DataDAM could also be used for neural architecture search. Indeed, instead of training many architectures on the full dataset, those architectures could trained on the distilled dataset, leading to a faster search.

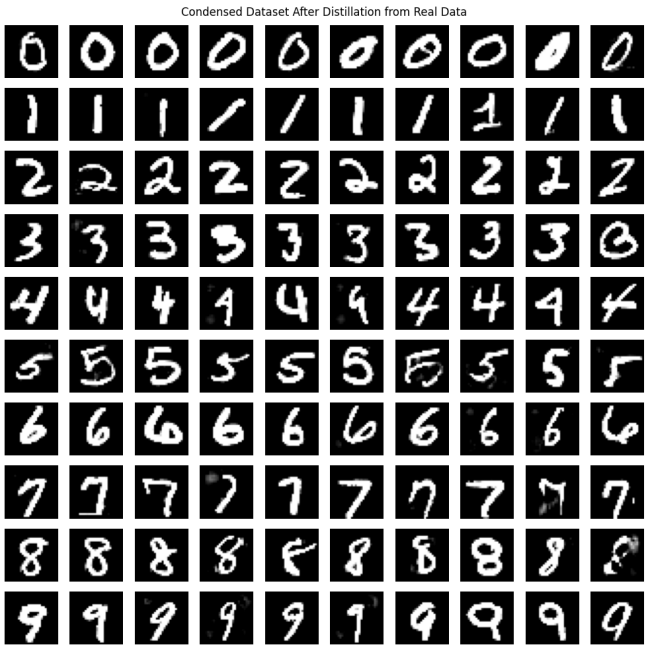
B. Part 2 - Data Distillation Learning using DataDAM

1) Build the distillation model: To build the distillation model, we used the code provided by the author of [4] in the repo [1]. This allowed us to create a DataDAM class that we could use. Some change were required to make the understanding easier and to fit our framework as well as remove the unneeded parts.
In [1], we can notice that the authors introduced a factor 100 in their loss. After various attempt, and because they use more iterations than we do, we decided to set the factor to 10000 so that the loss is not too small and the dataset is updated correctly.

2) Distillation of MNIST dataset from real data: To generate the synthetic dataset associated with the MNIST dataset, we used the DataDAM class. The parameters used were : $model = ConvNet3D$, $IPC = 10$, $K = 100$, $T = 10$, $\eta_S = 0.1$, $\zeta_S = 1$, $\eta_\theta = 0.01$, $\zeta_\theta = 50$, $\lambda_{mmd} = 0.01$ and $minibatches_{size} = 256$.



(a) The synthetic dataset before distillation from real data



(b) The synthetic dataset after distillation from real data

Fig. 1: The synthetic dataset before and after distillation from real data

Figure 1a shows the synthetic dataset before distillation from real data and Figure 1b shows the synthetic dataset after distillation from real data. At first glance, we can see that the synthetic dataset is almost identical to the real dataset. To have a better understanding of the difference between the two datasets, we can look at Figure 2. This figure shows the difference between the synthetic dataset before and after distillation from real data. We can see that the difference then resides in the detail of the shape of the digits. This is a good sign as it means that the synthetic dataset condenses the information of the real dataset.

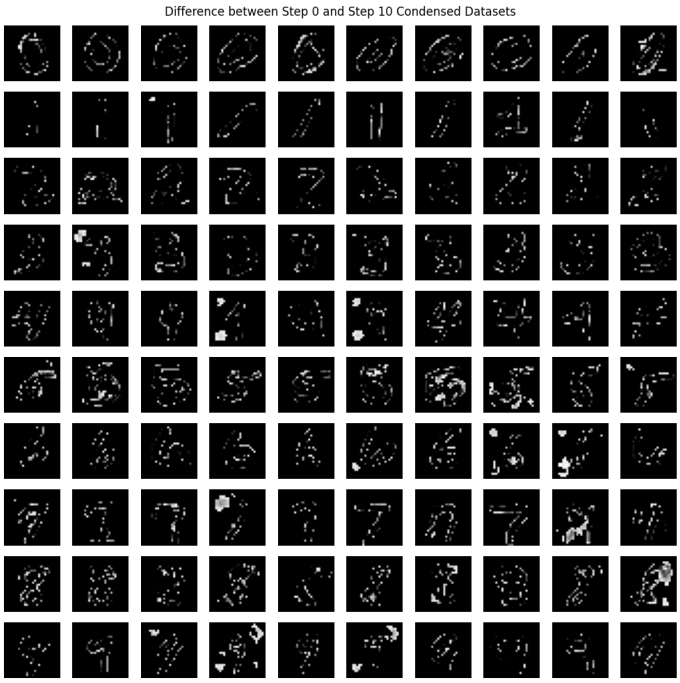
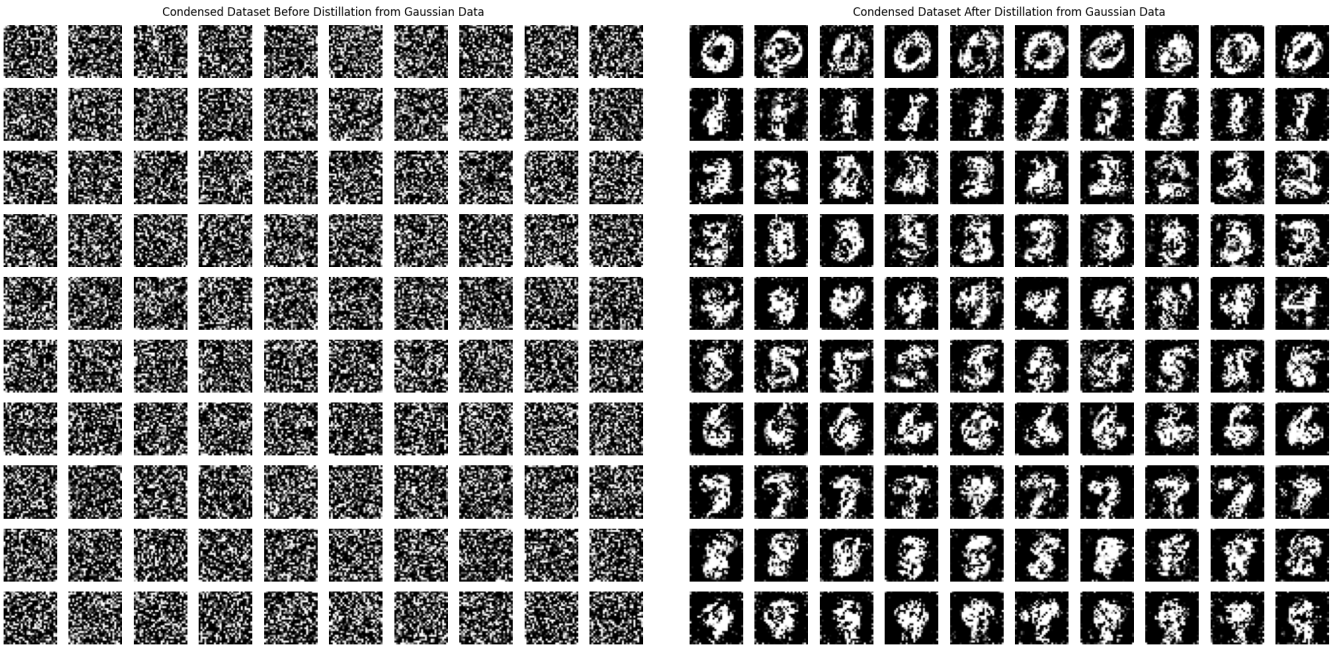


Fig. 2: The difference between the synthetic dataset before and after distillation from real data

3) Distillation of MNIST dataset from gaussian noise: To generate the gaussin noise, we used a standard normal distribution. The parameters used were the same as for the distillation from real data.



(a) The synthetic dataset before distillation from gaussian noise (b) The synthetic dataset after distillation from gaussian noise

Fig. 3: The synthetic dataset before and after distillation from gaussian noise

Figure 3a shows the synthetic dataset before distillation from gaussian noise and Figure 3b shows the synthetic dataset after distillation from gaussian noise. We can see that the distillation method succeeded in creating a synthetic dataset that shows digits we can recognize. However, those digits are not high quality.

4) Resutls and discussion for the MNIST dataset: We then trained a ConvNet3D model for 50 epochs on the full MNIST dataset and on the two synthetic datasets. The results are shown in Table I.

Dataset	Full	From gaussian	From real
Accuracy	99.49%	74.44%	91.81%
Training Time	7 min 7 s	$<$ 1 s	$<$ 1 s

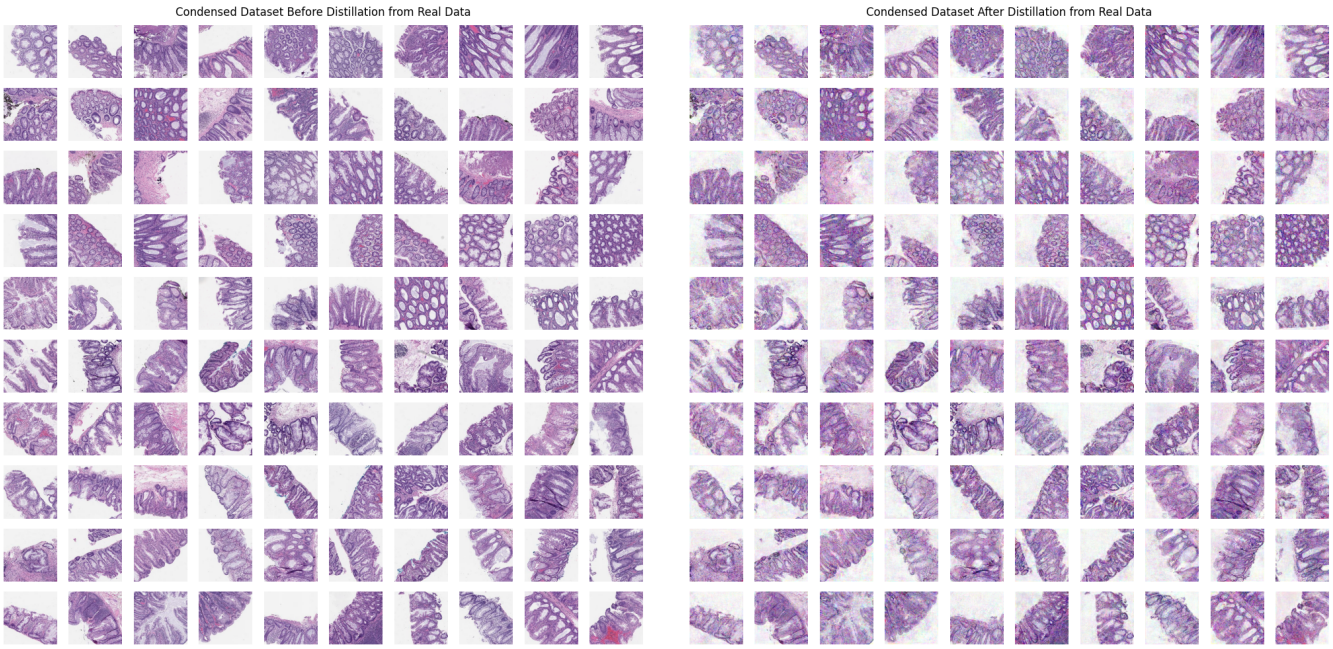
TABLE I: Results for the MNIST dataset

We can see that the model trained on the real dataset as the best accuracy. However, we also see that training was much longer than for the synthetic datasets. The model trained on the synthetic dataset from gaussian noise has the worst accuracy. This is expected as the synthetic dataset is of low quality. The model trained on the synthetic dataset from real data has a good accuracy and a training time that is much lower than the model trained on the full dataset. This shows that the distillation method is efficient in reducing the training time while preserving the performance of the model.
We then evaluate the dataset on a cross-architecture setup. For this, we train AlexNet on the full MNIST dataset and on the two synthetic datasets. The results are shown in Table II. Results show that the cross architecture permforms poorly on the synthetic datasets. This could be due by the fact that the synthetic dataset is created using the specificity of a ConvNet3D model and that AlexNet is not able to capture the same information.

Dataset	Full	From gaussian	From real
Accuracy	99.19%	17.9%	23.48%
Training Time	9 min 22 s	$<$ 1 s	$<$ 1 s

TABLE II: Results for the MNIST dataset on cross-architecture setup (AlexNet)

5) Distillation of the MHIST dataset from real data: To generate the synthetic dataset associated with the MHIST dataset, we used the DataDAM class. The parameters used were : $model = ConvNet7D$, $IPC = 50$, $K = 200$, $T = 10$, $\eta_S = 100$, $\zeta_S = 1$, $\eta_\theta = 0.01$, $\zeta_\theta = 50$, $\lambda_{mmd} = 0.01$ and $minibatches_{size} = 128$.



(a) The synthetic dataset before distillation from real data

(b) The synthetic dataset after distillation from real data

Fig. 4: The synthetic dataset before and after distillation from real data

Figure 4a shows the synthetic dataset before distillation from real data and Figure 4b shows the synthetic dataset after distillation from real data. At first glance, we can see that the synthetic dataset is almost identical to the real dataset. The main difference could be observed in the background of the images. To have a better understanding of the difference between the two datasets, we can look at Figure 5. This figure shows the difference between the synthetic dataset before and after distillation from real data. We can then see that the entire image is updated, meaning that the synthetic dataset is not a simple copy of the real dataset.

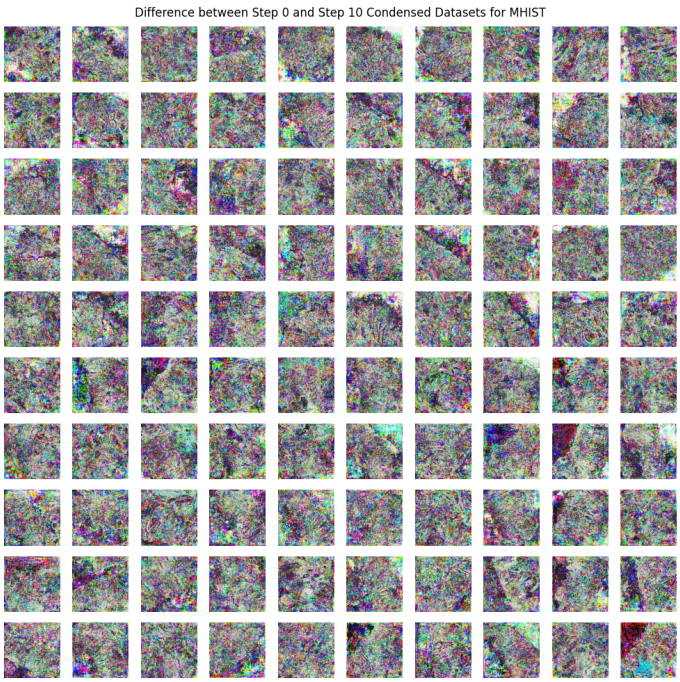
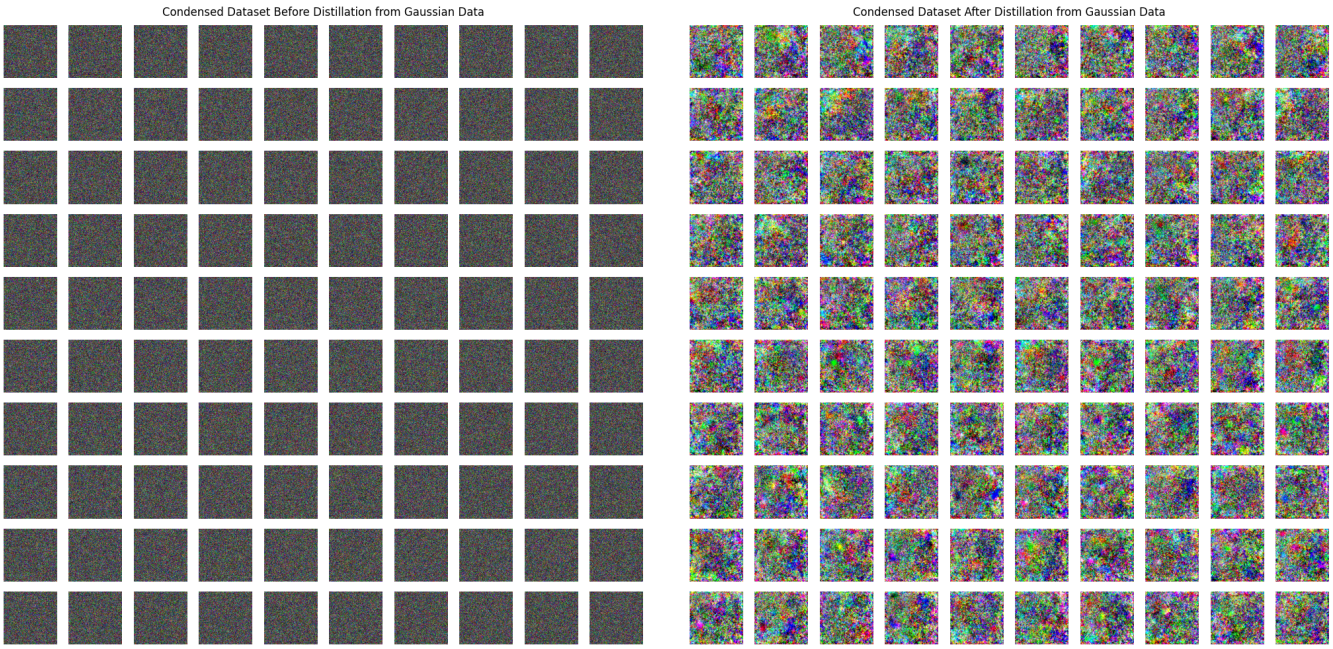


Fig. 5: The difference between the synthetic dataset before and after distillation from real data

6) Distillation of the MHIST dataset from gaussian noise: To generate the gaussin noise, we used a standard normal distribution. The parameters used were the same as for the distillation from real data.



(a) The synthetic dataset before distillation from gaussian noise

(b) The synthetic dataset after distillation from gaussian noise

Fig. 6: The synthetic dataset before and after distillation from gaussian noise

Figure 6a shows the synthetic dataset before distillation from gaussian noise and Figure 6b shows the synthetic dataset after distillation from gaussian noise. On the contrary of the MNIST dataset, the synthetic dataset from gaussian noise of the MHIST dataset is not recognizable. The results was expected as the data is much more complex with the three channels as well as in terms of the content of each image.

7) Resutls and discussion for the MHIST dataset: We then trained a ConvNet7D model for 50 epochs on the full MHIST dataset and on the two synthetic datasets. The results are shown in Table III.

Dataset	Full	From gaussian	From real
Accuracy	80.14%	53.63%	63.87%
Training Time	6 min 12 s	10 s	10 s

TABLE III: Results for the MHIST dataset

We can see that the model trained on the real dataset as the best accuracy. However, we also see that training was much longer than for the synthetic datasets. The model trained from the synthetic dataset from gaussian noise has the worst accuracy, close to the random guess. The model trained on the synthetic dataset from real data has a better accuracy. The training time is much lower than the model trained on the full dataset. This shows that the distillation method is efficient in reducing the training time but comes with a loss of accuracy. This is expected as the synthetic dataset is not a perfect representation of the real dataset. We then evaluate the dataset on a cross-architecture setup. For this, we train AlexNet on the full MHIST dataset and on the two synthetic datasets. The results are shown in Table IV. Surprinsingly, the model trained on the synthetic dataset from real data has a lower accuracy than the model trained on the synthetic dataset from gaussian noise and the latter performs nearly as well as the model trained on the full dataset. This could be due to the fact that the synthetic dataset from real data is too good for the ConvNet7D model where as the synthetic dataset from gaussian noise is not good enough so it generalizes better on a different architecture.

Dataset	Full	From gaussian	From real
Accuracy	68.37%	63.15%	36.87%
Training Time	6 min 40 s	17 s	18 s

TABLE IV: Results for the MHIST dataset on cross-architecture setup (AlexNet)

8) Neural architecture search: We then proposed to use the synthetic dataset to perform a neural architecture search. We want to find the best ConvNet model to classify either the MNIST or the MHIST dataset. We will search the model among ConvNetD1, ConvNetD2, ConvNetD3, ConvNetD4, ConvNetD5, ConvNetW32, ConvNetW64, ConvNetW128', ConvNetW256, ConvNetAS, ConvNetAR, ConvNetAL, ConvNetASwish, ConvNetASwishBN, ConvNetNN, ConvNetBN, ConvNetLN, ConvNetIN, ConvNetGN, ConvNetNP, ConvNetMP and ConvNetAP.

Results for the MNIST dataset are shown in Table V. We can observe that searching the best architecture using the synthetic dataset is much faster than just training the best architecture on the full dataset. In both cases, the best model achieve a high accuracy on the full dataset. This shows that the synthetic dataset is efficient to perform a neural architecture search.

Search	From Real	From Gaussian
Best Model	ConvNetW256	ConvNetW64
Accuracy on best model trained with synthetic data	91.92%	75.79%
Research time	34 s	82.83 s
Accuracy on best model trained on full data	99.52%	99.5%
Training time on full data	8 min 4 s	7 min 27 s

TABLE V: Results for the MNIST dataset

For the MHIST dataset, the results are shown in Table VI. We can observe that searching the best architecture using the synthetic dataset is much faster than just training the best architecture on the full dataset. However, the model found using the neural architecture search is not as good as the model trained on the full dataset in part II-B7. This could be due to the fact that, contrary to the MNIST dataset, the synthetic dataset of the MHIST dataset is not perfroming really well and then using it for a neural architecture search is not as efficient as it propagates the error of the synthetic dataset.

Search	From Real	From Gaussian
Best Model	ConvNetGN	ConvNetAS
Accuracy on best model trained with synthetic data	64.79%	63.15%
Research time	10 min 56 s	10 min 55 s
Accuracy on best model trained on full data	67.04%	63.66%
Training time on full data	3 min 45 s	3 min 36 s

TABLE VI: Results for the MHIST dataset

III. Task 2 - PAD

This part relies on the paper [3].

A. Part 1 - Questions anwsering

(a) The PAD method wants to solve the problem of misaligned information whether extracted or embedded into the synthetic dataset. The goal is to une only high-quality informations to improve the synthetic dataset.

(b) PAD introduced two novelties. The first one is the filtering of misaligned information during the extraction step by chosing the data to use according to the difficulty of samples and the image per class ration. The second novelty is the use of a layer-wise filtering in embedding. PAD only uses the deep layers of the agent model to perform distillation as they tend to captures higher quality information.

(c) The methodology is as follows:

(1) The real dataset is scored using a difficulty scoring function $\chi_t(x, y) = \mathbb{E} \left\| p(w_t, x) - y \right\|_2$ where x is a data point, y its label and $p(w_t, x) = \sigma(f(w_t, x))$ is the output of a model f transformed into a probability distribution.

(2) Using this difficulty score, a data scheduler is define as follow. All the easy data are selected first, then the hard data is gradually added to the set following the order of difficulty. When all the data is in the set, the easy data is gradually removed.

(3) An agent model is trained on the real dataset $\mathcal{D}_R$ using a Trajectory-Matching based method and the scheduler described. Then the change of parameters are stored. Let $\{\theta_t^*\}_0^N$ be a parameter sequence of the agent model (it is also called the trajectory).

(4) At each ineration, θ_t^* and θ_{t+M}^* are selected as start and target parameters. Let $\hat{\theta}_t$ be the parameters of the student agent model trained on the synthetic dataset $\mathcal{D}_S$. Then for N steps $\hat{\theta}_{t+i+1} = \hat{\theta}_{t+i} + \alpha \nabla \mathcal{L}_{CE}(\hat{\theta}_{t+i}, \mathcal{D}_S)$ where $\mathcal{L}_{CE}$ is the cross-entropy loss.

(5) To filter information $\hat{\theta}_{t+N}$ is defined such as only the $L - k$ last layers are used for matching where $k = \alpha * L$ and α is a hyperparameter that should be low in case of small IPC and high in case of high IPC.

(6) Then the synthetic dataset is updated such as it minimizes $\mathcal{L}$ where $\mathcal{L} = \frac{\left\| \hat{\theta}_{t+N} - \theta_{t+M}^* \right\|}{\left\| \theta_{t+M}^* - \theta_t^* \right\|}$.

(d) The advantages of PAD are: an improved alignment of information, a high-level of information (by discarding shallow layers) and, according to the paper, a better performance than other methods. The desadvanteges are the introduction of hyperparameter (notably α) that have to be tuned, the dependence of the difficulty scoring function that could be hard to define/improve. I believe that this method could distill the dataset correctly accordingt to the performances authors got. I believe that PAD will have more trouble with large scale dataset as the difficulty scoring function may score the entire dataset as difficult and thus leading to a non optimal scheduler.

B. Part 2 - Data Distillation Learning using PAD

1) Build the distillation model: To build the distillation model, we used the code provided by the author of [3] in the repo [2]. This allowed us to create a PAD class that we could use. As previously, some change were required to make the understanding easier and to fit our framework as well as remove the unneeded parts.

REFERENCES

[1] GitHub - DataDistillation/DataDAM: [ICCV 2023] DataDAM: Efficient Dataset Distillation with Attention Matching — github.com. https://github.com/DataDistillation/DataDAM. [Accessed 21-10-2024].

[2] GitHub - NUS-HPC-AI-Lab/PAD: Prioritize Alignment in Dataset Distillation — github.com. https://github.com/NUS-HPC-AI-Lab/PAD. [Accessed 22-10-2024].

[3] Zekai Li, Ziyao Guo, Wangbo Zhao, Tianle Zhang, Zhi-Qi Cheng, Samir Khaki, Kaipeng Zhang, Ahmad Sajed, Konstantinos N Plataniotis, Kai Wang, et al. Prioritize alignment in dataset distillation. *arXiv preprint arXiv:2408.03360*, 2024.

[4] Ahmad Sajedi, Samir Khaki, Ehsan Amjadian, Lucy Z Liu, Yuri A Lawryshyn, and Konstantinos N Plataniotis. Datadam: Efficient dataset distillation with attention matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17097–17107, 2023.