Project A – Dataset Distillation

Jangwon Suh
Edward S. Rogers St. Department of Electrical & Computer Engineering
University of Toronto
Toronto, Canada
j.suh@mail.utoronto.ca

I. Task 1: Dataset Distillation with Attention Matching

A. Basic Concepts

In this task, I used the dataset distillation with attention matching [1] to learn a synthetically small dataset for the MNIST and MHIST datasets, trained networks from scratch on the condensed images, and then evaluated them on the real testing data. This is one of the fundamental frameworks for dealing with dataset distillation in computer vision classification tasks while decreasing the computational costs.

(a) The purpose of using dataset distillation in [1] is to minimize training costs while maintaining strong generalization across datasets. Specifically, the goal is to create a small synthetic dataset that encapsulates the critical information of a larger real dataset, allowing models trained on the synthetic set to achieve performance comparable to those trained on the full dataset. This helps reduce the computational resources required for tasks like neural architecture search and continual learning.

(b) The proposed method, Dataset Distillation with Attention Matching (DataDAM), offers several advantages. First of all, it has higher accuracy. The method outperforms prior state-of-the-art methods, achieving up to a 6.5% improvement on CIFAR100 and a 4.1% improvement on ImageNet-1K. Furthermore, it shows reduced computational cost. DataDAM significantly lowers the computational expense by matching the spatial attention maps of real and synthetic datasets using randomly initialized neural networks. This reduces the need for expensive bi-level optimization techniques used in previous methods. Finally, it shows better scalability. Unlike some methods that struggle to scale with larger datasets (like ImageNet), DataDAM efficiently scales across datasets of varying resolutions, outperforming methods such as DM and CAFÉ in large-scale settings.

(c) The key novelty of DataDAM is its use of Spatial Attention Matching (SAM) to align attention maps of synthetic and real datasets across multiple neural network layers. This approach contrasts with prior methods that either focus on gradient matching or feature alignment. Additionally, DataDAM does not rely on pre-trained networks, making it more computationally efficient and reducing bias in the distilled synthetic dataset.

(d) DataDAM involves following steps. First, the synthetic dataset is either initialized randomly or through a selection process like K-Center clustering. Next, attention maps are generated for both the real and synthetic datasets at various layers of randomly initialized neural networks. These attention maps highlight the most informative regions of the input images. This is called Spatial Attention Matching (SAM). Then, this module minimizes the distance between the attention maps of real and synthetic datasets using an MSE loss function. Additionally, a complementary Maximum Mean Discrepancy (MMD) loss is used at the last layer to reduce feature distribution disparities. Finally, the synthetic dataset is optimized by minimizing both the SAM loss and the MMD loss using stochastic gradient descent (SGD).

(e) DataDAM helps store condensed representations of past tasks, improving memory efficiency and reducing catastrophic forgetting when models are trained incrementally on new tasks. Also, by using distilled synthetic data as a proxy, DataDAM accelerates the process of model evaluation during neural architecture search, reducing the time and resources needed for full dataset evaluations.

B. Dataset Distillation Learning

(a) First, I trained ConvNet-3 with MNIST dataset and ConvNet-7 with MHIST dataset, respectively. I used Stochastic Gradient Descent (SGD) as and optimizer with a cosine annealing scheduler with an initial learning rate of 0.01 for 50 epochs. Table I compares the classification accuracy and floating-point operations per second (FLOPs) trained with MNIST and MHIST dataset, respectively.

Table I. Comparison of accuracy and FLOPs between MNIST and MHIST dataset

Dataset	MNIST	MHIST
Accuracy (%)	99.52	81.27
FLOPs	61.3M	198M

(b) Next, I made my program to learn the synthetic dataset S using the selected model and Attention Matching algorithm. For initialization of condensed images, I randomly selected from real training images. The experimental setup can be found in Table II. The hyper-parameters K, T, η_S, ζ_S, η_θ, ζ_θ, and λ denote the number of random weight initializations, number of iterations, the learning rate for the condensed samples, the number of optimization steps for the condensed samples, the learning rate for the model, the number of optimization steps for the model, and the balance parameter, respectively.

Table II. Experimental setup for learning the synthetic dataset S with Attention Matching algorithm.

Dataset	MNIST	MHIST
K	100	200
T	10	10
η_S	0.1	0.1
ζ_S	1	1
η_θ	0.01	0.01
ζ_θ	50	50
Optimizer	SGD	SGD
# images / class	10	50
Minibatch size	256	128
Network	ConvNet-3	ConvNet-7
λ	0.01	0.01
Network width	128	32

(c) The left and top images of Figure 1 and 2 are the visualization of the generated synthetic MNIST and MHIST dataset, respectively. As you can see from those visualizations, the condensed images are clearly recognizable, both for MNIST and MHIST dataset.

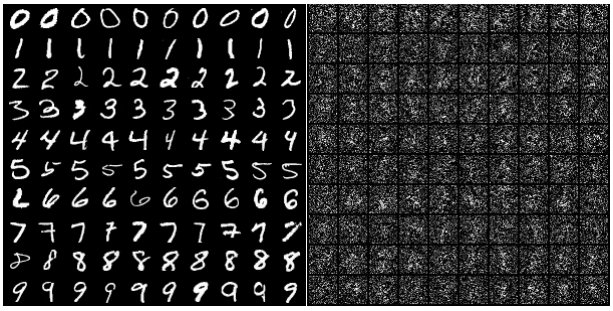

Figure 1. Generated synthetic MNIST dataset, initialized with randomly selected train images (left) and Gaussian noise (right)

(d) This time, I have initialized the condensed images with Gaussian noise, rather than randomly selected real training images. Every other hyperparameters are set identically with the previous settings. The right and bottom images of Figure 3 and 4 are the visualization of the generated synthetic MNIST and MHIST dataset, with these initializations, respectively. As you can see from those visualizations, the condensed images are clearly NOT recognizable, both for MNIST and MHIST dataset. Quantitative comparison is in part (e).

Table III. Comparison of test accuracy and training time between various settings, using MNIST dataset

Test setting	Part B-(a)	Part B-(b)	Part B-(d)	Part C
Dataset	**Original**	**Synthetic**	**Synthetic**	**Synthetic**
Initialization	**Real image**	**Real image**	**Gaussian noise**	**Real image**
Evaluated network	**ConvNet**	**ConvNet**	**ConvNet**	**MLP**
Test accuracy (%)	99.52	88.36	37.87	62.84
Training time (s)	405.3	1.594	1.688	0.9502

Table IV. Comparison of test accuracy and training time between various settings, using MHIST dataset

Test setting	Part B-(a)	Part B-(b)	Part B-(d)	Part C
Dataset	**Original**	**Synthetic**	**Synthetic**	**Synthetic**
Initialization	**Real image**	**Real image**	**Gaussian noise**	**Real image**
Evaluated network	**ConvNet**	**ConvNet**	**ConvNet**	**MLP**
Test accuracy (%)	81.27	60.69	47.19	51.53
Training time (s)	243.5	14.31	14.74	3.890

(e) Finally, I trained the selected network on a learned synthetic dataset (with 100 training images), then evaluated it on the real testing data. Table III compares the test accuracy performance and the training time with various settings, using MNIST dataset. Table IV does the same thing using MHIST dataset. (For a fair comparison, I have used the exact same experimental setting as part (a)).

First, the comparison between the "Part B-(b)" and "Part B-(d)" columns of Table III and IV reveals that using real-image-initialized synthetic datasets leads to significantly higher test accuracy compared to using Gaussian-noise-initialized datasets. Specifically, the test accuracy of the evaluated network trained with a real-image-initialized MNIST synthetic dataset is 2.333 times higher than that of the network trained with a Gaussian-noise-initialized MNIST dataset. Similarly, for the MHIST synthetic dataset, the test accuracy is 1.286 times higher when initialized with real images compared to Gaussian noise. These findings implies that initializing with real image is more effective for dataset distillation than using Gaussian noise.

Next, the comparison between the "Part B-(a)" and "Part B-(b)" columns of Table III and IV reveals that using synthetic datasets leads to not only lower test accuracy but also less training time compared to using original datasets. Specifically, the test accuracy of the evaluated network trained with a MNIST synthetic dataset is 1.126 times lower than that of the network trained with an original MNIST dataset. Similarly, for the MHIST dataset, the test accuracy is 1.339 times lower when trained with synthetic dataset compared to original dataset.

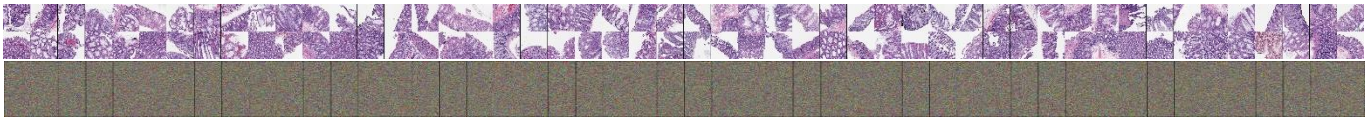

Figure 2. Generated synthetic MHIST dataset, initialized with randomly selected train images (top) and Gaussian noise (bottom)

However, the benefit arises from training time. Specifically, the training time of the evaluated network trained with a MNIST synthetic dataset is 254.3 times less than that of the network trained with an original MNIST dataset. Similarly, for the MHIST dataset, the training time is 17.02 times less when trained with synthetic dataset compared to original dataset. These findings implies that using synthetic dataset gains a huge benefit in training time, while sacrificing acceptable degree of training accuracy, which is easily solvable by tuning other hyperparameters, such as epochs.

C. Cross-architecture Generalization

Another key advantage of the dataset distillation approaches is that the condensed images learned using one architecture can be used to train another, unseen one. To verify this, I made a program to learn the synthetic datasets for the MNIST and MHIST over ConvNet. Once the condensed sets are synthesized, it trained MLP and evaluated its cross-architecture performance in terms of classification accuracy on the test sets. Except using MLP for evaluation, I used exactly same hyperparameter, list in Table II, for the comparison. The comparison between the "Part B-(b)" and "Part C" columns of Table III and IV reveals that cross-architecture generalization of distilled dataset is also possible, but not truly successful. Specifically, the test accuracy of the MLP trained with a MNIST dataset was 1.406 times lower than that of the ConvNet. Similarly, for the MHIST dataset, the test accuracy is 1.178 times lower for MLP compared to ConvNet. These findings implies that cross-architecture generalization is possible, but it needs further hyperparameter tuning, such as epochs, network width and depth, to show similar performance with identical-network-evaluated ones.

Table V. Comparison of test accuracy with method [1] and [2]

Method	**[1]**	**[2]**
Test accuracy (%)	88.36	87.11

D. Application

I have also applied my synthetic datasets to DC-Bench [2]. It provides the first large-scale standardized benchmark on dataset condensation. It consists of a suite of evaluations to comprehensively reflect the generability and effectiveness of condensation methods through the lens of their generated dataset. By testing our distilled dataset through this benchmark, we can literally verify our dataset's generability and effectiveness. The comparison of test accuracy of trained models using synthetic dataset using method [1] and [2] is summarized in Table V. I used exactly same hyperparameters, listed in Table II, for the comparison. We can clearly see that the test accuracy measured using [1] and [2] matches well (only 1.25% difference), implying the reliability of synthetic dataset testing method suggested in [1].

II. TASK 2: COMPARISON WITH STATE-OF-THE-ARTS METHODS

A. Basic Concepts

In this task, I used one of the state-of-the-art methods and compared them with the Attention Matching algorithm that I have used in Task 1 to further explore the effect of dataset distillation methods in visual classification tasks. I used the paper "Prioritize Alignment in Dataset Distillation" [3].

(a) The Prioritize Alignment in Dataset Distillation (PAD) method addresses the problem of misaligned information during the information extraction and embedding stages in dataset distillation. Previous methods often introduced redundant and detrimental information due to this misalignment, impacting the quality of the distilled dataset. PAD fills the gap by aligning information in these stages to enhance the quality of the synthetic dataset.

(b) PAD introduces two novel strategies. First one is Filtering Information Extraction (FIEX), which prunes the target dataset based on sample difficulty, ensuring that only relevant information aligned with the compression ratio is extracted. The second one is Filtering Information Embedding (FIEM), which discards shallow-layer parameters during distillation to prevent embedding low-level, redundant signals. These strategies lead to significant improvements in distillation performance.

(c) The PAD method operates in two stages. First stage is information extraction. This method uses an agent model to extract information from a target dataset. Unlike previous methods, PAD incorporates a data scheduler to control the use of easy and hard samples according to the Iterative Per-Class (IPC) setting. A difficulty scoring function, such as EL2N, evaluates sample difficulty for optimized extraction. Secon stage is information embedding. Instead of using all model parameters, PAD only utilizes deep-layer parameters for metric matching. This strategy avoids injecting low-level, basic information and focuses on embedding high-quality, semantic features, improving the final synthetic dataset.

(d) The advantages are improved performance and versatility. First of all, PAD achieves state-of-the-art results by aligning the information used in distillation, ensuring better quality synthetic datasets. Also, this method can be applied to gradient, distribution, and trajectory matching-based approaches, showcasing its adaptability. The disadvantages mainly comes from its complexity. This method requires the use of a data scheduler and selective parameter masking, which may increase the implementation and computational overhead. I think PAD shows potential for scaling to large datasets like ImageNet. Its strategy of selectively using deep-layer parameters and difficulty-aligned data scheduling enhances its robustness and adaptability to complex datasets. However, computational limitations could pose challenges, as indicated by the authors' note on requiring significant resources for validation.

B. Applying to MNIST Dataset

(a) Like Task 1, the learning pipeline has two stages: (1) learn the condensed images using the selected method; (2) train

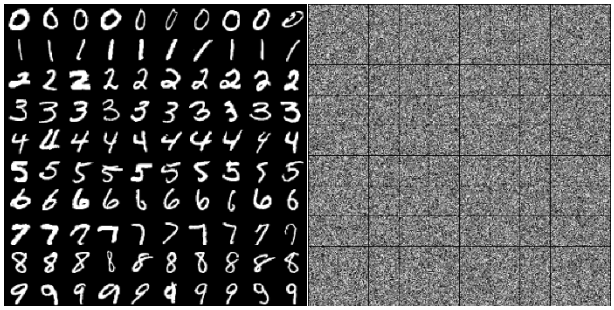

Figure 3. Generated synthetic MNIST dataset, initialized with randomly selected train images (left) and Gaussian noise (right), using method [3]

the network from scratch on the condensed images, then evaluate them on the real testing data. To make the comparison fair, I have used exactly same hyperparameter of Task 1, listed in Table II. I have only used MNIST dataset for comparison.

(b) Figure 3 is the visualization of the generated synthetic MNIST dataset, initialized with randomly selected train images (left) and Gaussian noise (right). As you can see from those visualizations, the condensed images initialized with real images are clearly recognizable, while that with Gaussian noises are NOT clearly recognizable, which is same as the result of method [1] in Task 1.

Table VI. Comparison of test accuracy and training time between various settings, using MNIST dataset

Distillation method	**[1]**	**[3]**	**[1]**	**[3]**
Initialization	**Real image**	**Real image**	**Gaussian noise**	**Gaussian noise**
Test accuracy (%)	88.36	90.55	37.87	14.96
Training time (s)	1.594	4.583	1.688	4.683

Table VI compares the test accuracy and training time of evaluation network (ConvNet) between various settings, using MNIST synthetic dataset. The test accuracy was 1.025 times lower and training time was 2.875 times less for the synthetic dataset initialized with real images generated with method [1], compare to that of method [3]. Also, the test accuracy was 2.531 times higher and training time was 2.774 times lower for the synthetic dataset initialized with Gaussian noises generated with method [1], compare to that of method [3]. This clearly shows that method [1] shows similar test accuracy when the synthetic dataset is initialized with real images, and shows considerably better test accuracy when the synthetic dataset is initialized with Gaussian noise. This does not implies that method [1] outperforms method [3], since initialization is not a big problem to solve.

Overall, The dataset distillation methods have significant effects on the generalization and recognition abilities of the models trained on the synthetic datasets.

Methods like [1] and [3] are designed to produce synthetic datasets that help models achieve comparable performance to those trained on the full original dataset. This ability to generalize is validated through cross-architecture evaluations. For instance, [3]'s distilled datasets have shown strong performance not just on the architecture used during distillation (e.g., ConvNet) but also on unseen architectures like ResNet, VGG, and AlexNet. This suggests that the distilled data captures essential features that are generalizable across various model architectures. Also, By distilling information effectively, these methods create smaller, high-quality datasets that models can use to generalize well even with fewer training samples. This is particularly useful in scenarios with limited computational resources or data storage constraints.

Methods like [1] and [3] also focus on embedding higher-level, semantically rich features into the synthetic dataset by filtering out low-level information during distillation. This approach enhances the model's ability to recognize and classify complex patterns accurately. By ensuring that the distilled data retains crucial, abstract features learned in the deep layers of the network, the models trained on these datasets achieve better recognition capabilities. [3]'s strategy of discarding shallow-layer parameters helps to prevent embedding low-level, noisy signals into the synthetic dataset. This leads to clearer, more informative training data, which enhances the model's ability to make precise classifications and improves recognition accuracy.

Table VII. Comparison of test accuracy and training time between original dataset and synthetic dataset using various IPC values

IPC	**Original dataset**	**1**	**10**	**100**
Test accuracy (%)	99.52	49.97	90.55	97.04
Training time (s)	405.3	4.275	4.396	8.986

III. Lossless Dataset Distillation

Lastly, I utilized the findings from the recent research presented in [4] and put into practice the algorithm to determine the optimal number of images per class (IPCs). The objective is to configure this synthetic dataset in a manner that ensures the performance of the model trained on it matches or surpasses that of a model trained on the original dataset. The distillation and evaluation process are conducted using ConvNet architecture. Table VII summarizes the comparison of test accuracy and training time between original dataset and synthetic dataset using various IPC values. As you can see from Table VII, test accuracy tends to get better as IPC increases, but the training time also gets longer as a tradeoff. Using IPC value of 100 gives enough test accuracy (only 2.48% of difference compared to original dataset) while earning a huge amount of training time reduction (45.10 times shorter than original dataset).

References

[1] 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

[2] Justin Cui, Ruochen Wang, Si Si, and Cho-Jui Hsieh. Dc-bench: Dataset condensation benchmark. arXiv preprint arXiv:2207.09639, 2022. https://openreview.net/pdf?id=Bs8iFQ7AM6.

[3] Zekai Li, Ziyao Guo, Wangbo Zhao, Tianle Zhang, Zhi-Qi Cheng, Samir Khaki, Kaipeng Zhang, Ahmad Sajedi, Konstantinos N Plataniotis, Kai Wang, and Yang You. Prioritize alignment in dataset distillation, 2024.

[4] Ziyao Guo, Kai Wang, George Cazenavette, Hui Li, Kaipeng Zhang, and Yang You. Towards lossless dataset distillation via difficulty-aligned trajectory matching. arXiv preprint arXiv:2310.05773, 2023

The code is provided in https://github.com/j-suh-1998/ECE1512H-Project-A.git