Computational Statistics Final Project

Ben Cartwright
NYU Tandon School of Engineering
New York, NY
bcc9958@nyu.edu

1 Introduction

Data distillation is the process of compressing a large dataset into a smaller, synthetic set that preserves essential information for training machine learning models with comparable performance. It enables faster training, lower storage requirements, and efficient model development. In this project, we apply gradient matching for data distillation on the MNIST image classification dataset, implementing a class-wise optimization method to generate synthetic datasets of one image per class. Our goal is investigate the accuracy retention of these synthetic datasets across various image classification models and synthetic data initialization methods. Additionally, we explore how the addition of higher order gradient terms impact the accuracy and convergence of gradient matching.

2 Problem Set Up

Gradient matching is a data distillation method where the synthetic data is generated to mimic the loss dynamics of a model on the original data by treating the synthetic set as a learnable parameter and minimizing model loss gradient discrepancies between the original and synthetic sets. In general, gradient matching can be defined as:

Gradient Matching Given a model f and an initial dataset $\mathcal{D}_{org} = \{(x_i, y_i)\}_{i=1}^{n}$, $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. Gradient matching creates a dataset $\mathcal{D}_{syn} = \{(x_i, y_i)\}_{i=1}^{m}$, m « n, such that the match loss, $\mathcal{L}_{\text{match}} = \|\nabla_\theta \mathcal{L}_{\mathcal{D}_{syn}}(\theta) - \nabla_\theta \mathcal{L}_{\mathcal{D}_{org}}(\theta)\|_2^2$, is minimized, where $\mathcal{L}$ is the loss function of our model f.

The goal of this project is to investigate how the following three factors influence the gradient matching process:

1. **Model Selection** Three commonly implemented models on the MNIST classification problem are Convolutional Neural Networks (CNN), Multi-Layer Perceptron (MLP), and LeNet. We evaluate the gradient matching process conducted with respect to each model.
2. **Synthetic Data Initialization** We explore the impact on the gradient matching process for our CNN model from various synthetic data initialization strategies including: randomly generated, sampled, black, white and gray images.
3. **Higher Order Gradient Terms** We add the Hessian of $\mathcal{L}_{\text{match}}$ into our synthetic dataset update step attempting to improve performance with higher order loss curvature.

We explore accuracy retention across these various factors and examine the recognizably of the resulting images for each respective class. We found that CNN based gradient descent preserves the most essential information across all three models, and we were able to use the CNN model to produce recognizable synthetic images under all initialization methods besides black image initialization. Furthermore, adding the Hessian improved the accuracy retention of the CNN and MLP models, and drastically imrpoved the recognizability of images produced by the MLP model.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

3 Methodology

3.1 Gradient Matching Implementation

Gradient Matching

In general, the gradient matching algorithm minimizes $\|\nabla_\theta \mathcal{L}_{\mathcal{D}_{syn}}(\theta) - \nabla_\theta \mathcal{L}_{\mathcal{D}_{org}}(\theta)\|_2^2$ through the following iteration process (further outlined in the appendix:

For each iteration:

1. Compute $\nabla_\theta \mathcal{L}_{\text{orig}}(\theta)$ on the original dataset.
2. Compute $\nabla_\theta \mathcal{L}_{\text{syn}}(\theta)$ on the synthetic dataset.
3. Calculate the matching loss $\mathcal{L}_{\text{match}}$.
4. Update the synthetic dataset $\mathbf{X}_{\text{syn}}$ and $\mathbf{y}_{\text{syn}}$ using gradient descent.

Class-wise Implementation

The general approach to gradient matching treats $\mathcal{D}_{syn}$ as a learnable parameter and optimizes it iteratively so that the gradients of $\mathcal{L}$ with respect to θ on $\mathcal{D}_{syn}$ are as close as possible to the gradients of $\mathcal{L}$ with respect to θ on $\mathcal{D}_{org}$.We construct $\mathcal{D}_{syn}$ through gradient descent on the gradient of $\mathcal{L}_{\text{match}}$ with respect to $\mathcal{D}_{syn}$ usually starting from random noise.

In the context image classification, we need to ensure our label equally represents each class and we fix $\mathbf{y}_{\text{syn}}$ to contain n instances of each class. Then, $\mathbf{X}_{\text{syn}}$ is then iteratively updated by $\frac{\partial \mathcal{L}_{\text{match}}}{\partial \mathbf{X}_{\text{syn}}}$

The following is the gradient matching approach used throughout all experiments in this paper. The code for my implementation can be found here:

`https://github.com/BenjaminCartwright/GradientMatching`

It is adapted from the code in the this repository:

`https://github.com/VICO-UoE/DatasetCondensation`

1. For i in iterations:
  2. If i in evaluation rounds:
    Evaluate f on $\mathcal{D}_{syn}$
  3. Create a new instance of our model f
  4. For j in loop rounds:
    loss $\leftarrow 0$
    5. For c in classes:
      Get $X_{\text{org}}, y_{\text{org}}$ that are in class c.
      Get $X_{\text{syn}}, y_{\text{syn}}$ that are in class c.
      Compute the real gradient := $\nabla_\theta \mathcal{L}_{X_{\text{org}}}(\theta)$.
      Compute the synthetic gradient := $\nabla_\theta \mathcal{L}_{X_{\text{syn}}}(\theta)$.
      Compute the match loss := $\mathcal{L}_{\text{match}} = \|\nabla_\theta \mathcal{L}_{X_{\text{org}}}(\theta) - \nabla_\theta\, \mathcal{L}_{X_{\text{org}}}(\theta)\|_2^2$.
      class loss $\leftarrow$ match loss.
      loss $\leftarrow$ loss + class loss.
    6. Update X_{syn} via gradient descent.
      $\mathbf{X}_{\text{syn}} \leftarrow \mathbf{X}_{\text{syn}} - \eta \frac{\partial \mathcal{L}_{\text{match}}}{\partial \mathbf{X}_{\text{syn}}}$

The key difference is that when we define the class loss as the match loss restricted to data in that specific class, and then sum the class losses as our match loss for gradient descent on the synthetic images.

Note: Step 2 is not part of the class wise gradient matching implementation, but is an evaluation step to monitor performance by iteration.

3.2 Data Initialization

To start the gradient matching process, we need to generate an initial synthetic dataset $\mathcal{D}_{init}$. In our case, we create 1 initial synthetic image per class, and modify the structure of $\mathcal{D}_{init}$ to fit the specific model. Typically, the synthetic images are initialized as random noise, but our data initialization method lets us initiate from random, class-wise samples from the real data or as black, white or gray images. Starting with a random or sampled $\mathcal{D}_{init}$ is common practice, but I included black, white and gray image initialization to see if gradient matching can converge under these conditions.

3.3 Hessian Addition

Gradient matching typically involves updating $\mathbf{X}_{\text{syn}}$ by the gradient of the match loss $\mathcal{L}_{\text{match}}$ multiplied by the learning rate η. To compute the match loss, we usually use the squared L2-norm of the gradients of $\mathcal{L}$ with respect to θ, as shown below:

$$\mathcal{L}_{\text{match}} = \sum_i \|\nabla_\theta \mathcal{L}_{\text{org},i} - \nabla_\theta \mathcal{L}_{\text{syn},i}\|_2^2$$

Where i indexes the parameters of our model f. However, we can potentially improve the convergence of gradient matching by adding higher order gradient terms that captures second order information about the curvature of the loss landscape. We do so by adjusting our match loss definition to:

$$\mathcal{L}_{\text{match}} = \sum_i \|\nabla_\theta \mathcal{L}_{\text{org},i} - \nabla_\theta \mathcal{L}_{\text{syn},i}\|_2^2 + \mathbf{v}^\top \mathbf{H}\mathbf{v}.$$

Where:

1. $\mathbf{H} = \frac{\partial^2 \mathcal{L}}{\partial \theta \, \partial \theta^\top}$ is the Hessian matrix of the loss function $\mathcal{L}$
2. $\mathbf{H}_{ij} = \frac{\partial^2 \mathcal{L}}{\partial \theta_i \, \partial \theta_j}$ is the (i, j)-th element of the Hessian matrix
3. $\mathbf{v} = \nabla_\theta \mathcal{L}_{\text{org}} - \nabla_\theta \mathcal{L}_{\text{syn}}$ the difference between the gradients of the real and synthetic loss

The Hessian acts like a regularization term that captures the curvature of the loss function with respect to the parameters θ and allows for better gradient alignment for complex models. Overall, the idea is incorporate this second order information will improve convergence.

Note: For match loss, the L2-norm, euclidean distance, is used but we can also use other metrics like MSE or cosine similarity.

3.4 Overview

Thus far, we have outlined a gradient matching implementation suitable for image classification problems that can incorporate higher order gradient terms and initialize the synthetic data in multiple ways.

To conduct gradient matching for a given model, initialization method and decision to use the Hessian addition or not, we follow the steps below:

1. Get the original or real dataset and split into a train and testing set
2. Initialize the synthetic dataset according to our initialization method
3. Preform class-wise gradient matching to optimize the synthetic dataset based on the given model and choice to use the Hessian or not
4. Create a new instance of our model and train it on the original training data
5. Create a new instance of our model and train it on the synthetic data
6. Use both models to make predictions on the test set and compare accuracy

Additionally, the synthetic images are saved every iteration where the model is evaluated on the synthetic data and the match loss is recorded in total and by class every iteration.

4 Experiments

The code for all experiments can be found here:

`https://github.com/VICO-UoE/DatasetCondensation`

In each instance of gradient matching, we create one synthetic image per class, use 0.1 as our update learning rate parameter η, 10 for the number of outer loops and 15 - 25 for the number if iterations. For evaluation rounds, we train the model with a learning rate of 0.01 for 10 epochs on the synthetic data before evaluation on the test set.

Note: Higher iterations failed to improve performance for single image per class distillation.

4.1 Initial Model Evaluations

First, we evaluate each model on the original dataset to establish a baseline performance for comparison to the synthetic dataset. Each model was trained for 10 epochs at a learning rate of 0.01 and batch size of 1024.

Table 1: Model accuracies on the original dataset (MNIST)

Model	CNN	MLP	LeNet
Accuracy	0.9930	0.9730	0.9855

4.2 Model Selection

Now, we use each of the CNN, MLP and LeNet models to preform gradient matching to create a synthetic image dataset of one image per class. The following shows the accuracy of each model on their respective synthetic dataset per iteration. It also shows the final accuracy from a model instance trained on the synthetic dataset outside of the gradient matching loop.

We observed that gradient matching on the CNN model achieved the highest accuracy retention by a significant margin. The synthetic images for the the CNN and LeNet models are also highly recognizable as the digits 0 - 9, whereas several of the MLP synthetic images become highly distorted after a certain point.

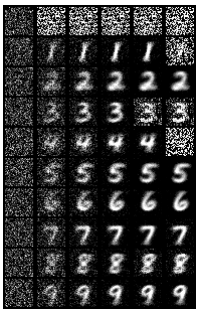
Table 2: Model Accuracy at Different Iterations with Final Accuracy

Model	**Iter 0**	**Iter 5**	**Iter 10**	**Iter 15**	**Iter 20**	**Iter 25**	**Final**
MLP	0.0899	0.2122	0.4867	0.5675	0.5245	0.5489	0.5706
CNN	0.0975	0.6862	0.6639	0.7303	0.6851	0.7218	0.8365
LeNet	0.0937	0.3105	0.4568	0.5869	0.6057	0.6225	0.4621

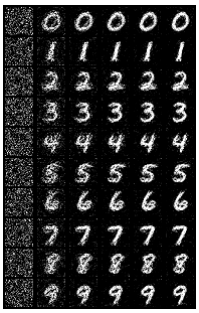
4.3 Synthetic Data Initialization

We now compare the evaluation accuracy on the synthetic set by iteration across various synthetic initialization methods using the CNN model for gradient matching. The process converges to synthetic datasets with comparable accuracy retention for all the initialization methods besides starting with all black images. Also, we see random initialization out preform sampling especially considering initial accuracy.

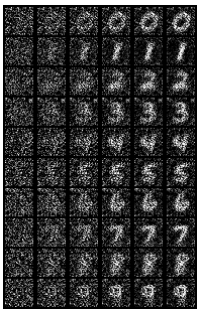
Figure 1: Comparison of Synthetic Image Progression



(a) MLP



(b) CNN



(c) LeNet

Table 3: Initialization Types and Accuracy over Iterations

Init Type	**iter0**	**iter10**	**iter15**	**Final**
Random	0.1195	0.6479	0.6758	0.8256
Sample	0.3612	0.6384	0.6551	0.8082
Black	0.0982	0.0539	0.1021	0.0952
White	0.0996	0.6586	0.6541	0.8027
Grey	0.0662	0.6028	0.6414	0.7608

4.4 Higher Order Gradient Terms

Table 4 compares the final accuracies of each model, on the original data (Full), the synthetic data (Syn), and the synthetic data generated using the hessian addition match loss (Syn Hess). The Hessian addition dramatically reduces the accuracy retention for the LeNet model, but for CNN and MLP the the Hessian addition improves performance noticeably. The recognizably of the synthetic images generated with the MLP model gradients is improved as well. We no longer observe a spontaneous decay in quality or an inability to recognize specific numbers.

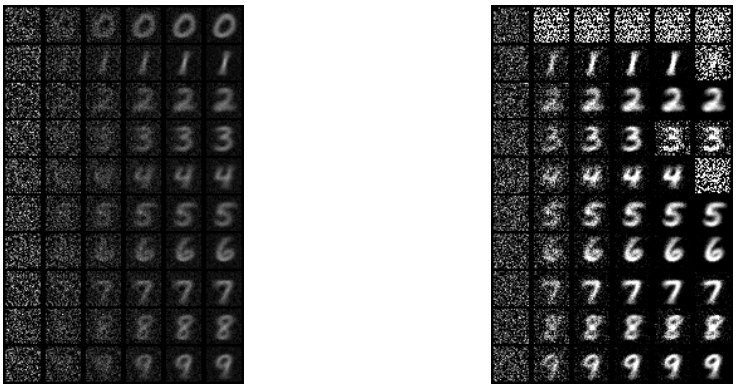
Table 4: Model Accuracy Comparison

Model	**CNN**	**MLP**	**LeNet**
Full	0.993	0.973	0.9855
Syn	0.8365	0.5706	0.4621
Syn Hess.	0.8563	0.6359	0.1189

5 Conclusions and Next Steps

In summary, we found gradient matching with CNNs can preserve information very well under significant size reduction and across a variety of initialization methods. Most notably, we saw the Hessian addition improve accuracy retention and observed interesting dynamics between the Hessian and recognizably of the MLP based images. Additionally, Figure 2 (a) displays recognizable images that are much darker in color. This warrants further exploration into post-distillation transformations to add contrast. My next step would be to explore how the scaling parameter for the hessian and update learning rate impact the the pixel color range and general smoothness of the synthetic images. However, properly observing the effects would require reducing the image update learning rate and increase total and inner class iterations.

Figure 2: Comparison of Synthetic Image Progression



(a) Hessian

(b) No Hessian

6 References

1. G. Cazenavette, T. Park, A. Halawi, S. Arora, Y. Singer, "Dataset Distillation by Matching Training Trajectories," Available: `https://openreview.net/forum?id=mSAKhLYLSsl`, 2023.
2. J. Wang, W. Bao, Y. Zhang, Q. Yang, "Dataset Distillation: A Comprehensive Review," arXiv preprint, 2021. Available: `https://arxiv.org/abs/2110.04181`.
3. B. Zhao, H. Bilen, "Dataset Condensation with Gradient Matching," in *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021. Available: `https://icml.cc/virtual/2021/poster/8609`.
4. VICO-UoE, "Dataset Condensation GitHub Repository," Available: `https://github.com/VICO-UoE/DatasetCondensation`, 2021.

7 Appendix

7.1 Gradient Matching for Data Distillation

The goal of gradient matching in data distillation is to create a small synthetic dataset that matches the learning dynamics of the original dataset by aligning the gradients of the loss function.

Optimization Objective

The optimization objective is to minimize the gradient difference between the synthetic dataset and the original dataset:

$$\min_{\mathcal{D}_{syn}} \|\nabla_\theta \mathcal{L}_{\mathcal{D}_{syn}}(\theta) - \nabla_\theta \mathcal{L}_{\mathcal{D}_{org}}(\theta)\|_2^2$$

where:

- $\mathbf{X}_{\text{syn}}, \mathbf{y}_{\text{syn}}$ are the synthetic data points and their labels.
- $\mathcal{L}_{\text{syn}}$ is the loss function evaluated on the synthetic dataset.
- $\mathcal{L}_{\text{orig}}$ is the loss function evaluated on the original dataset.
- θ represents the model parameters.

Update Process

The synthetic dataset $\mathbf{X}_{\text{syn}}$ and labels $\mathbf{y}_{\text{syn}}$ are treated as learnable parameters and updated to minimize the objective. The update rule is given by:

$$\mathbf{X}_{\text{syn}} \leftarrow \mathbf{X}_{\text{syn}} - \eta \frac{\partial \mathcal{L}_{\text{match}}}{\partial \mathbf{X}_{\text{syn}}}$$

$$\mathbf{y}_{\text{syn}} \leftarrow \mathbf{y}_{\text{syn}} - \eta \frac{\partial \mathcal{L}_{\text{match}}}{\partial \mathbf{y}_{\text{syn}}}$$

where:

- η is the learning rate.
- $\mathcal{L}_{\text{match}} = \|\nabla_\theta \mathcal{L}_{\text{syn}}(\theta) - \nabla_\theta \mathcal{L}_{\text{orig}}(\theta)\|_2^2$ is the gradient matching loss.
- $\frac{\partial \mathcal{L}_{\text{match}}}{\partial \mathbf{X}_{\text{syn}}}$ and $\frac{\partial \mathcal{L}_{\text{match}}}{\partial \mathbf{y}_{\text{syn}}}$ are the gradients of the matching loss with respect to the synthetic data and labels.

Iterative Process

The optimization proceeds iteratively as follows:

1. Compute $\nabla_\theta \mathcal{L}_{\text{orig}}(\theta)$ on the original dataset.
2. Compute $\nabla_\theta \mathcal{L}_{\text{syn}}(\theta)$ on the synthetic dataset.
3. Calculate the matching loss $\mathcal{L}_{\text{match}}$.
4. Update the synthetic dataset $\mathbf{X}_{\text{syn}}$ and $\mathbf{y}_{\text{syn}}$ using gradient descent.

The synthetic dataset is refined iteratively until the gradients of the synthetic dataset closely approximate those of the original dataset.