COMP6248 REPRODUCIBILITY CHALLENGE / DATASET CONDENSATION WITH GRADIENT MATCHING

Gongwei Shi, Victoria Draganova & Vinuja Sathiyakumar
School of Electronics and Computer Science
University of Southampton

ABSTRACT

In this ICLR reproducibility challenge, we have attempted to reproduce several experiments conducted in *Dataset Condensation with Gradient Matching*, where the authors tackle the problem of large datasets by learning a small set of condensed synthetic samples for training deep neural networks from scratch. Our findings through the reproduced experiments illustrate that most results are reproducible using the available information in the published paper. However, the reproduced results from some of the experiments were not quite satisfying.

1 INTRODUCTION

Reproducing scientific experiments is one of the core principle that promotes the development of science. The reproducibility is the ability for an independent research team to produce same results referring to the same methodologies and environments. This not only allows the scientist to verify whether the results are correct, but also ensures transparency and helps the researchers to have confidence in understanding exactly what was done.

Over the last three years, the International Conference on Representation Learning (ICLR) has run the reproducibility challenge to encourage more researchers to investigate reproducibility of papers submitted for publication. The challenge involves verifying the empirical results and claims made in the papers by reproducing the computational experiments via the information provided by the authors in their papers. To respond to this challenge, our group will aim to reproduce the Dataset Condensation with Gradient Matching (Zhao et al., 2020). In the following of this report, we will provide a general overview of the said paper and explain our implementation and experiments based on information provided in it. Eventually we will comment on the reproducibility of the paper and verify whether the claims made in the paper are supported by our findings.

2 OVERVIEW OF DATASET CONDENSATION WITH GRADIENT MATCHING

Large datasets are fundamental to the success of many Machine Learning applications, especially in the field of Deep Learning. To achieve high performance, deep networks require extremely large datasets. With datasets at such scale, it requires large amount of time to process and build models and storing them can be problematic. In the paper, the authors proposed to address this problem using a technique called Dataset Condensation with Gradient Matching. The authors took inspiration from Data Distillation (Wang et al., 2018). They have set up experiments to learn a small set of condensed synthetic samples that can be used by the neural networks for downstream tasks. The objective is to generate a small synthetic dataset that has similar performance to the original dataset. The authors formulate this problem as a minimization problem between two sets of gradients.

3 IMPLEMENTATION AND EXPERIMENTS

To obtain a condensed dataset, the pipeline of dataset condensation has been split into two stages. The first one, called stage C, consists of learning the condensed set of images, while the second one, stage T, includes training classifiers from scratch on the newly created synthetic data. Stage C starts with initialising a synthetic set of images from Gaussian noise. Then, a network is trained on

both the synthetic dataset and the real dataset with the goal to minimise the distance between their gradients w.r.t. its weights. This process is depicted in Figure 1.

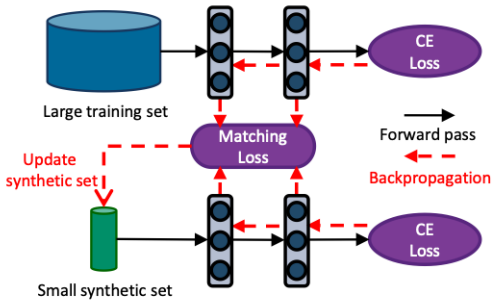

Figure 1: Dataset Condensation (from (Zhao et al., 2020))

In stage T the synthetic data is used to train a model using significantly less computational power, with the intent of being as powerful as if it was trained on the original large dataset.

An implementation problem we had was with the distance function. We found it difficult to understand its exact implementation details from the paper and thus decided to inspect the distance function from their published code to get an understanding. In the end, we used their implementation of the distance function. Another implementation detail we observed was that, in the published code, the images were grouped into different classes before the synthetic data was learned. We realised this halved the run time of the algorithm used to train the synthetic data compared to when they were ordered randomly. Therefore, we also decided to use their code for grouping the images in our implementation. Everything else in the code is implemented by us via following the original paper. Our implementation can be found here: `https://github.com/COMP6248-Reproducability-Challenge/Dataset-Condensation-with-Gradient-Matching`. We have concentrated our reproducibility efforts on the two main experiments that have been conducted by the authors in the paper. The implementation environment we have used is Google Colaboratory.

3.1 Initial Experiment

The first experiment is to evaluate the classification performance of a Convolutional Neural Network (ConvNet) on condensed images generated out of four benchmark datasets: MNIST, FashionMNIST, SVHN, and CIFAR10. Condensed sets of size 1, 10 and 50 images per class are created out of each of the four datasets. After it is generated, each set of images is used to train 20 randomly initialized models and the obtained accuracies' mean and standard deviation are recorded. For the purpose of this experiment we have implemented a ConvNet following the paper's guidance. Next we have generated two types of condensed sets per benchmark dataset, with 1 and 10 images per class. We have decided to skip the 50 images per class setting due to it being very computationally intensive. Finally, we have trained 20 ConvNets on each set of synthetic data and have evaluated them. The computational time for conducting the whole experiment was around 16h 40min. Generating a condensed dataset with 1 image per class and evaluating it took on average 10 minutes, and one with 10 images per class - 4 hours. However, the computational time could increase depending on the GPU allocated by Google Colab. An example synthetic dataset, created from MNIST (1 image per class) with the ConvNet, can be seen in Figure 2.

Figure 2: Synthetic Dataset generated from MNIST 1 image/class

3.2 Cross-architecture generalization Experiment

The second experiment is to evaluate the performance of different neural network architectures on both learning and classifying the synthetic data. The networks used by the authors are MLP, ConvNet, LeNet, AlexNet, VGG-11, and ResNet-18. Each of them is used to learn a condensed repre-

sentation of the MNIST dataset with 1 image per class. Six condensed sets are therefore generated. Next, 20 models of each architecture are trained on each of the six condensed datasets to assess the average classification performance of the deep-network architectures. In total, 36 accuracies are reported. To reproduce this experiment, we had to first implement the networks. The hyperparameters of MLP and ConvNet were clearly mentioned in the paper, therefore we were able to build them. As the paper did not mention any modifications to AlexNet and LeNet, we built them following the original network architectures. We did not implement VGG-11 and ResNet-18. Next, we used each model to learn a condensed dataset and classified the four datasets by using 20 models of each type. In the end we obtained 16 accuracies. The time taken to obtain the results per network was around 10 minutes, with the exception of AlexNet, for which 45 minutes were necessary. These times also increased depending on the GPU allocated by Google Colab.

4 OUR FINDINGS AND DISCUSSION

Dataset	Img/class	Accuracy (Paper)	Accuracy (Our Code)
MNIST	1	91.7±0.5	87.1±0.5
	10	97.4±0.2	95.7±0.1
FashionMNIST	1	70.5±0.6	68.1±0.2
	10	82.3±0.4	80.9±0.2
SVHN	1	31.2±1.4	31.4±0.7
	10	76.1±0.6	74.3±0.4
CIFAR10	1	28.3±0.5	29.3±0.6
	10	44.9±0.5	43.1±0.3

Table 1: Testing accuracies (%) of ConvNet on four condensed datasets

Table 1 shows the results for the experiment described in Section 3.1. From Table 1 it can be seen that we were able to reproduce all the reported accuracies to a certain level. The difference between our accuracies and the ones reported in the paper are around ±2% except for MNIST with 1 image per class. Running the paper's code on MNIST with 1 image per class showed an accuracy of 91.95±0.4, which is around 4% more than what our code produced and close to what they report. We believe that this is because, in the paper it is mentioned that data augmentation is performed specifically on MNIST. However, no details on what parameters for augmentation are used are mentioned in the paper, which we identify as not enough information to reproduce the same augmentation and thus the reported accuracy.

The other reason for the small difference between the accuracies we produced and the ones reported in the paper could be due to the way they trained the networks to evaluate the synthetic data. We evaluated the synthetic data by training the networks on it for 300 epochs by using SGD as the optimizer. However, they have used SGD with momentum and a weight decay. They also reduced the learning rate after some epochs. These details were not mentioned in the paper, but when we tested using their method of evaluation (from the published code) on our synthetic data, the accuracy increased. Therefore, this shows that their method of training the networks on the synthetic data for evaluation has some effect on the accuracy. Even though we did not have this information in the paper, we were able to produce results very close to their results just by following their paper.

Table 2 shows our recorded accuracies when the synthetic data were learned and evaluated using different neural networks (experiment described in Section 3.2). The synthetic datasets were created using MNIST. The results of the experiments where MLP and ConvNet were used for learning and/or evaluating the synthetic data, were quite good. They are close to the values reported in the paper. However, the experiments where LeNet and AlexNet were involved in did not produce average accuracies close to those reported in the paper. After investigating their published implementation, we realised their implementation of AlexNet and LeNet was different compared to the original architectures, but this was not mentioned in the paper. The hyperparameters of the AlexNet had been

changed a lot. For example, the original network produces up to 384 output channels, however, in the network implemented by the authors the maximum output channels are 256. Also, compared to the original LeNet network, they used ReLU as the activation function whereas the original network uses the sigmoid function. They also changed the average pooling layers to max pooling layers. Since the changes to the networks were not specified in the paper, we were not able to obtain results that are close to what has been reported, just by following the paper.

C/T	MLP	ConvNet	LeNet	AlexNet
MLP	66.8±0.7 **(70.5±1.2)**	68.5±2.2 **(63.9±6.5)**	10.9±3.0 **(77.3±5.8)**	10.3±3.6 **(70.9±11.6)**
ConvNet	72.8±1.0 **(69.6±1.6)**	87.7±0.7 **(91.7±0.5)**	9.6±2.8 **(85.3±1.8)**	11.7±2.9 **(85.1±3.0)**
LeNet	47.6±3.6 **(71.0±1.6)**	53.8±5.4 **(90.3±1.2)**	11.7±3.4 **(85.0±1.7)**	10.4±2.0 **(84.7±2.4)**
AlexNet	21.2±4.2 **(72.1±1.7)**	62.3±3.4 **(87.5±1.6)**	25.0±10.1 **(84.0±2.8)**	19.5±5.9 **(82.7±2.9)**

Table 2: Cross-architecture performance in testing accuracy (%) for condensed 1 image/class in MNIST. Accuracies recorded in the paper are displayed in the brackets.

To see the impact their implementation of AlexNet and LeNet had on the accuracies, we ran their implementations of them, in our code. The results are shown in Table 3. As can be seen, the average accuracies of all the experiments increased significantly when their implementations of AlexNet and LeNet were used. However, the accuracies produced by our code with their implementation of the networks are slightly lower to the ones reported in the paper. We believe this difference could be due to a combination of both, as mentioned before, data augmentation carried out on the MNIST data when training the network, and the way the synthetic data is evaluated. We also believe that the random initialisations of both the networks used to learn the synthetic data and the initial synthetic data cause the accuracies to vary.

C/T	MLP	ConvNet	LeNet	AlexNet
LeNet	74.3±0.9 **(71.0±1.6)**	82.9±1.2 **(90.3±1.2)**	70.8±8.5 **(85.0±1.7)**	75.5±1.1 **(84.7±2.4)**
AlexNet	75.4±0.6 **(72.1±1.7)**	82.9±1.5 **(87.5±1.6)**	73.7±3.3 **(84.0±2.8)**	75.6±1.5 **(82.7±2.9)**

Table 3: Cross-architecture performance in testing accuracy (%) for condensed 1 image/class in MNIST by using the network architectures implemented in author's published code, in our code. Accuracies recorded in the paper are displayed in the brackets.

5 Conclusion

In this report, we have tried to reproduce the results in *Dataset Condensation with Matching Gradients* (Zhao et al., 2020). Overall, we found that most of the results were reproducible by following the paper, except for a few. This is because in the case when MNIST dataset was used the data augmentation they made to the dataset was not clearly explained in the paper. Also, changes made to the original architectures of the neural networks were not specified. In addition, their method of training the networks on the synthetic data for evaluation was not mentioned, which could have also contributed to the difference in the results. However, with the access to their code, the results reported in this paper are fully reproducible.

References

Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. *arXiv preprint arXiv:2006.05929*, 2020.