Medical Image Dataset Distillation via Deep Generative Prior and Matching Training Trajectories

Tong Chen **JunYu Deng**

Abstract

Dataset Distillation aims to synthesize a small dataset such that a model trained on the synthetic dataset will match the test accuracy of the model trained on the full dataset. Unlike other works on dataset distillation, we mainly focus on distilling medical image datasets, which are better suit for this task. Moreover, despite the recent progress in this field, existing dataset distillation methods fail to generalize to new architectures. To overcome this issue, we propose to use a deep generative prior from pre-trained deep generative models to obtain synthetic dataset and combine it with the method of dataset distillation by matching training trajectories. Our method escalates the overall performance and improves cross-architecture generalization in all settings.

1 Introduction

Dataset Distillation, proposed by Wang et al. (5), is a technique that aims to extract a more compact and representative subset from a larger dataset. And nowadays, medical datasets have a profound significance and meaning as they directly impact clinical decision-making, early disease detection, and medical research. Enhancing the performance of models on these datasets plays a vital role in improving healthcare outcomes, enabling accurate diagnoses and supporting treatment decisions. We think it's of great significance to achieve better model performance on medical datasets.

In the context of medical datasets, *Dataset Distillation* holds great promise. Medical datasets often suffer from imbalances in the distribution of samples across different classes or medical conditions. This issue arises due to various factors such as the prevalence of certain diseases, the rarity of specific conditions. These imbalances of datasets can significantly impact the performance and reliability of machine learning models trained on such datasets. It's because when the number of images in different classes is highly skewed, models trained on these datasets tend to be biased towards the majority class. As a result, the performance of these models in accurately diagnosing or classifying rare medical conditions is severely compromised.

However, *Dataset Distillation* plays a crucial role in mitigating the imbalances within medical datasets. By selectively retaining representative samples from all classes during the distillation process, the resulting distilled dataset can effectively address the imbalance issue. This ensures that the number of images in different classes becomes more balanced, enabling the machine learning models to learn and generalize better across all medical conditions. Besides that, *Dataset Distillation* reduces the noise and redundancy present in the original dataset. It filters out noisy or irrelevant samples, enabling the models to focus on the most informative instances. This helps to enhance the model's learning capability and generalization performance, particularly when working with limited data.

Above are **the reasons why we choose to use *Dataset Distillation* on medical datasets**. *MTT Distillation (Dataset Distillation by Matching Training Trajectories)*, proposed by George Cazenavette et al. (1), is a state-of-art method of Dataset Distillation that focuses on aligning the training trajectories of teacher and student models. It introduces an additional matching objective

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

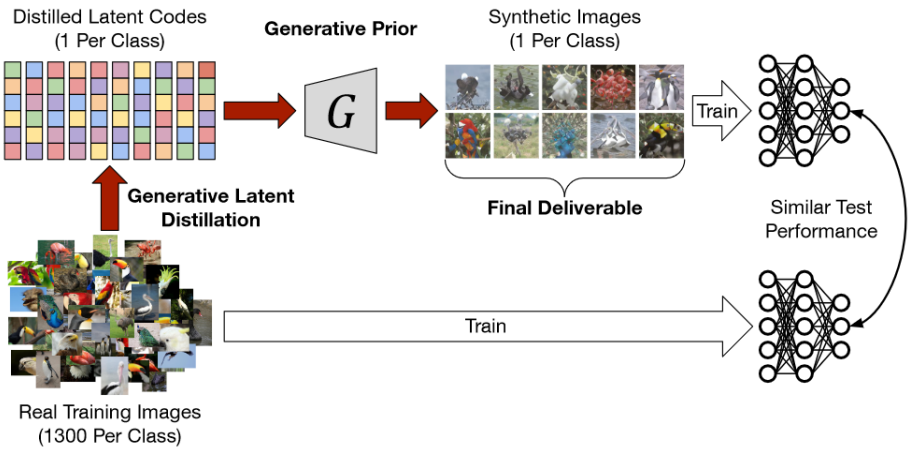

Figure 1: Rather than directly distilling a dataset into synthetic pixels, our method instead distills into the latent space of a *deep generative prior*.

during training to minimize the discrepancy between intermediate predictions of the teacher and student models. *MTT Distillation* is able to capture and transfer the dynamic evolution of model predictions during the training process, ultimately leading to a better distillation performance. More information on Dataset Distillation and MTT will be explained in the appendix.

Unfortunately, MTT faces a major challenge, limiting both its scientific value and empirical applications—The distilled synthetic dataset is often optimized with respect to a specific network architecture, but struggles to generalize to other architectures. This issue may be partially caused by parameterizing the synthetic dataset in pixel space. Directly optimizing pixels can be susceptible to learning high-frequency patterns that overfit the specific architecture used in training. To address this, we consider regularizing the distillation process to some prior that may help cross-architecture generalization. We propose to employ a *deep generative prior* by parameterizing the synthetic dataset in the intermediate feature space of generative models, such as Generative Adversarial Networks (GANs).(2) Our prior encourages the learned datasets to be more generalizable to new architectures but is also lax enough to not prohibitively restrict the expressiveness of the distilled dataset. This is added to the beginning of the state-of-art method, *MTT*, as shown in Figure 1. It will be detailed in Section 3.3. Our method not only enhances the quality of distilled medical dataset but also improves its cross-architecture generalization.

2 Dataset

2.1 PG (PatchGastricADC22) Dataset

The PG (PatchGastricADC22) dataset consists of histopathological captions of stomach adenocarcinoma endoscopic biopsy specimens, which are extracted from diagnostic reports and paired with patches extracted from the associated whole slide images.

The dataset is divided in three subtypes: Well differentiated tubular adenocarcinoma, Moderately differentiated tubular adenocarcinoma and Poorly differentiated tubular adenocarcinoma. The first two categories account for the most part of it, while at the same time, the poorly differentiated accounts for only a small part of the dataset.

The site for download is here: `https://zenodo.org/record/6550925#.ZBpUT3ZByPr`

2.2 CRC (Colorectal Cancer) Dataset

This CRC Dataset contains 411,890 unique image patches derived from histological images of colorectal cancer and gastric cancer patients in the TCGA cohort. The TCGA (Cancer Genome Atlas) was a landmark research project that aimed to characterize and understand the molecular basis of

cancer. The data generated by TCGA allowed researchers to identify and characterize cancer-related genes, pathways, and molecular subtypes across multiple cancer types, serving as a great dataset resource for artifitial intelligence research concerning medical datasets learning.

The subtypes for tumor actually are composed of three parts: MSI(Microsatellite Instable), MSS(Microsatellite Stable) and MUT(Highly Mutated). The MSS were collected into a single class and accounts for about 70% of the whole dataset, while for MSI and MSS, they are collected altogether to for a new class called MSIMUT, accounting for 30% in total. And this makes the whole baseline of the CRC dataset a binary classification problem.

The site for download is here: `https://zenodo.org/record/2530835#.Y9-dmS9BxPb`

2.3 Exploratory Data Analysis

Performing EDA on CRC Dataset, we found that the mean of the three channels(R G B) would be:[0.4914, 0.4822, 0.4465]; the standard deviation of the three channels would be:[0.2023, 0.1994, 0.2010]; size of each image would be 224*224; 2 classes: [MSS] [MSIMUT]

Performing EDA on PG Dataset, we found that the mean of the three channels(R G B) would be:[0.485, 0.456, 0.406]; the standard deviation of the three channels would be:[0.229, 0.224, 0.225]; size of each image would be 224*224; 3 classes:[Well Differentiated] [Moderately differentiated] [Poorly Differentiatd]

All the indexes above are normalized.

2.4 Implementation

We perform some changes on the above two datasets, including the following moves:

1: Resize the input image to a fixed size of 224x224 pixels.

2: Apply random color jittering to the input image. We modify the brightness (0.2), saturation (0,0.2) and hue (0.1) within specified ranges.

3: Randomly flip the input image horizontally and vertically with a probability of 0.5.

4: Set the pattern for *Differentiable Siamese Augmentation* (3): to be specific, we set the default operations to be: [probability for flip] = 0.5, [ratio for scaling] = 1.2, [ratio for rotating] = 15.0, [ratio for cropping] = 0.125, [ratio for cutout] = 0.5, [ratio for noise] = 0.05; Besides that, we let the user of the whole program to freely choose their configurations for Data Augmentation, the parameters here includes: crop, scale, rotate and noise.

5: Read the labels, bagnames, data and annotations separately. The images in both datasets are obtained by dividing the original images into smaller patches. So each image belonging to the same original image holds the same bagname.

3 Methodology

Dataset Distillation refers to the curation of a small, synthetic training set $\mathcal{D}_{syn}$ such that a model trained on this synthetic data will have similar performance on the real test set as a model trained on the large, real training set D_{real}.

In this section, we first describe our method of dataset distillation that mimics the long-range behavior of real-data training, matching multiple training steps on distilled data to many more steps on the real data. After that, we will introduce a *deep generative prior* to the distillation process as a form of regularization by optimizing latent codes of a pre-trained generative model(Figure 1).

3.1 Expert Trajectories

The method of matching training trajectories involves using expert trajectories τ^* to guide the distillation of our synthetic dataset. By expert trajectories, we mean the time sequence of parameters $\{\theta_t^*\}_0^T$ obtained during the training of a neural network on the full, real dataset. To generate these expert trajectories, we simply train a network on the real dataset and save its snapshot parameters

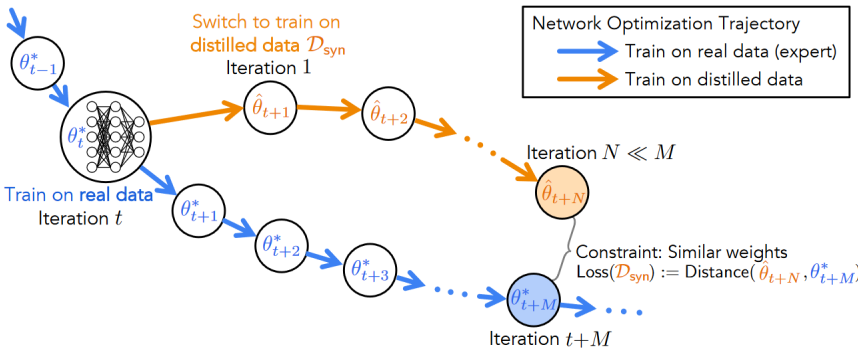

Figure 2: We perform long-range parameter matching between training on distilled synthetic data and training on real data.(1) Starting from the same initial parameters, we train distilled data $\mathcal{D}_{syn}$ such that N training steps on them match the same result (in parameter space) from much more M steps on real data.

at every epoch. These sequences of parameters are called "expert trajectories", which represent the theoretical upper bound for the dataset distillation task: the performance of a network trained on the full, real dataset. Similarly, we define student parameters $\hat{\theta}_t$ as the network parameters trained on synthetic images at the training step t. Our goal is to distill a dataset that will induce a similar trajectory (given the same starting point) as that induced by the real training set such that we end up with a similar model.

Since these expert trajectories are computed using only real data, we can pre-compute them before distillation. All of our experiments for a given dataset were performed using the same pre-computed set of expert trajectories, allowing for rapid distillation and experimentation.

3.2 Matching Expert Trajectories

Our distillation process learns from the generated sequences of parameters making up our expert trajectories $\{\theta^*_t\}_0^T$. At each distillation step, we first sample parameters from one of our expert trajectories at a random timestep θ^*_t and use these to initialize our student parameters $\hat{\theta}^*_t := \theta^*_t$. We place an upper bound T^+ on t to ignore the less informative later parts of the expert trajectories where the parameters do not change much. With our student network initialized, we then perform N gradient descent updates on the student parameters with respect to the classification loss of the synthetic data:

$$\hat{\theta}_{t+n+1} = \hat{\theta}_{t+n} - \alpha \nabla l(\mathcal{A}(\mathcal{D}_{syn}); \hat{\theta}_{t+n}) \tag{1}$$

where $\mathcal{A}$ is the differentiable augmentation technique, and α is the trainable learning rate used to update the student network. The data augmentation technique used must be differentiable so that we can back-propagate through the augmentation layer to our synthetic data.

From this point, we return to our expert trajectory and retrieve the expert parameters from M training updates after those used to initialize the student network θ^*_{t+M}. Finally, we update our distilled images according to the weight matching loss: the normalized squared L_2 error between the updated student parameters $\hat{\theta}_{t+N}$ and the known future expert parameters θ^*_{t+M}:

$$\mathcal{L} = \frac{\|\hat{\theta}_{t+N} - \theta^*_{t+M}\|_2^2}{\|\theta^*_t - \theta^*_{t+M}\|_2^2} \tag{2}$$

We normalize the L_2 error by the expert distance traveled so that we still get a strong signal from later training epochs where the expert does not move as much. This loss function directly encourages the distilled dataset to guide the network optimization along a similar trajectory (Figure 2). We have also experimented with other choices of loss functions such as a cosine distance, but find that the L2 loss works better. We then minimize this objective to update $\mathcal{D}_{syn}$ and our trainable learning rate α by back-propagating through all N updates to the student network using SGD with momentum.

3.3 Adding a Deep Generative Prior

The two sections above have illustrated the process of dataset distillation by matching long-range parameter trajectories to get optimized distilled images $\mathcal{D}_{syn}$. To date, most existing methods of dataset distillation optimize $\mathcal{D}_{syn}$ in pixel space. This will allow too much freedom to overfit to the backbone architecture. So we propose introducing a deep generative prior by distilling the dataset into the latent space of pre-trained generative models, such as Generative Adversarial Networks(GANs).

Concretely, we consider a deep generative model G that outputs samples $G(z)$ given latent vector z. At distillation time, we parameterize the small synthetic dataset $\mathcal{D}_{syn}$ as

$$\mathcal{D}_{syn} := \{G(z) : z \in \mathcal{Z}\} \tag{3}$$

where $\mathcal{Z}$ is a set of latent vectors. Since G is fully differentiable, we can optimize $\mathcal{Z}$ w.r.t. $\mathcal{L}$. Please see Algorithm 1 for a complete write-up of our method.

Algorithm 1 Dataset Distillation via Deep Generative Prior and Matching Training Trajectories

Input: $\{\tau_i^*\}$: set of expert parameter trajectories trained on D_{real}.
Input: M: # of updates between starting and target expert params.
Input: N: # of updates to student network per distillation step.
Input: $\mathcal{A}$: Differentiable augmentation function.
Input: $T^+ < T$: Maximum start epoch.
Input: G: Pre-trained generator.
Input: P_z: Distribution of latent initializations.
▷ Initialize distilled latents: $\mathcal{Z} \sim P_z$
▷ Initialize trainable learning rate: $\alpha := \alpha_0$
for each distillation step... **do**
  ▷ Get distilled images from latents: $\mathcal{D}_{syn} = G(\mathcal{Z})$
  ▷ Sample expert trajectory: $\tau^* = \{\theta_t^*\}_0^T$
  ▷ Choose random start epoch: $t \leq T^+$
  ▷ Initialize student network with expert params: $\hat{\theta}_t := \theta_t^*$
  for $n = 0 \rightarrow N - 1$ **do**
    ▷ Sample a mini-batch of distilled images:
      $b_{t+n} \sim \mathcal{D}_{syn}$
    ▷ Update student network w.r.t. classification loss:
      $\hat{\theta}_{t+n+1} = \hat{\theta}_{t+n} - \alpha \nabla l(\mathcal{A}(\mathcal{D}_{syn}); \hat{\theta}_{t+n})$
  end for
  ▷ Compute loss between ending student and expert params:
    $\mathcal{L} = \|\hat{\theta}_{t+N} - \theta^*_{t+M}\|_2^2 \;/\; \|\theta_t^* - \theta^*_{t+M}\|_2^2$
  ▷ Update Z with respect to $\mathcal{L}$:
    $\mathcal{Z} \leftarrow SGD(\mathcal{Z}; \mathcal{L})$
end for
Output: Distilled images $\mathcal{D}_{syn} = G(\mathcal{Z})$

4 Experiments

We evaluate our method on various medical datasets, including 3-class PG dataset and 2-class CRC dataset. For results with and without the generative prior, we use the same set of hyper-parameters (N, M, T^+, #iterations, etc.) to ensure a fair comparison. We provide additional visualizations in the appendix.

Evaluation Protocol and Baselines. After distilling our synthetic datasets with their respective algorithm, we then evaluate them on the same backbone architecture and a set of unseen architectures. To evaluate a synthetic dataset on a given architecture, we train a network from scratch on the distilled dataset and then evaluate it on the validation set.

To generate the distilled images for our method, we employ the distillation process detailed in the previous section and Algorithm 1. The training regiment is the same for all networks and datasets: SGD with momentum, l_2 weight decay, and 500 epochs of linear warm-up followed by another 500

of cosine decay. An appropriate (fixed) starting learning rate is used for each architecture, and the final validation set evaluation is done using the exponential moving average of the model's weights. This process is repeated 5 times, and the mean validation accuracy ±1 standard deviation is reported.

We compare to several recent methods including Dataset Distillation (DD) and Dataset Distillation by matching training trajectory *without* deep generative prior (MTT).

Network Architectures. We use the ConvNet (4) architecture as our backbone network. A Depth-n ConvNet consists of n blocks followed by a fully-connected layer where each block consists of a 3 × 3 convolutional layer with 128 filters, instance, ReLU non-linearity, and 2 × 2 average pooling with stride 2. We use the AlexNet, VGG-11 and ResNet-18 for for our cross-architecture generalization experiments.

4.1 PG and CRC dataset

Unlike the prior method of dataset distillation(DD), MTT and our method perform long-range parameter matching, where N training steps on distilled data match a much larger M steps on real data. Our method distinguishes from MTT in the way we parameterize the synthetic dataset. As depicted in Table 1, by introducing the deep generative prior, the performances of DD and MTT fall short of our method.

In Table 1, we can discover that there is a significant improvement of performance when applying the method of matching training trajectories, but introducing the generative prior only brings a relatively small improvement on the performance. The generative prior is expert in preserving variances into the distilled images, which will contribute to the improvement of the performance. But the medical dataset we use may be lack in variance, so the generative prior doesn't do much help.

Dataset	Img/Cls	Ratio %	DD	MTT	Ours
PG	1	0.0028	35.8 ± 3.2	40.2 ± 1.6	41.3 ± 1.4
	10	0.0283	36.1 ± 1.5	46.7 ± 1.3	47.6 ± 2.2
	20	0.0566	39.9 ± 0.8	45.8 ± 1.2	**50.7** ± 1.1
CRC	1	0.0010	52.5 ± 2.6	60.3 ± 2.9	60.8 ± 2.1
	10	0.0107	55.1 ± 0.7	65.8 ± 1.8	66.5 ± 0.6
	20	0.0214	56.3 ± 0.7	71.2 ± 0.9	**73.4** ± 1.8

Table 1: Comparing distillation methods on PG and CRC dataset. The metrics for PG dataset is accuracy and for CRC dataset is AUC for binary classification. We distill the given number of images per class on the training set using a 128-width ConvNet as the backbone architecture. The results come from training 3 ConvNets from scratch using synthetic dataset and averaging their performances on the real test data. Combining the method of applying the deep generative prior and matching training trajectories significantly improves the quality of the distilled dataset.

Distillation Method	AlexNet	VGG11	ResNet18	Average
DD	25.7 ± 0.9	27.1 ± 1.2	27.6 ± 1.5	26.8 ± 1.2
MTT	32.3 ± 1.8	34.6 ± 0.7	35.2 ± 1.1	34.0 ± 1.2
Ours	36.2 ± 0.7	36.8 ± 1.4	38.5 ± 0.9	**37.2** ± 1.0

Table 2: PG dataset performance on unseen architectures.

4.2 Improving Cross-Architecture Generalization

Arguably, the most lacking point of all previous dataset distillation methods, cross-architecture generalization gives a good understanding of how well the distillation method "understands" the classification task rather than simply over-fitting to a given architecture. In Table 2 and Table 3, we show cross-architecture results for DD, MTT, and our method. For each method and dataset, a 1

Distillation Method	AlexNet	VGG11	ResNet18	Average
DD	43.7 ± 0.9	45.2 ± 1.2	45.8 ± 0.7	44.9 ± 0.9
MTT	52.3 ± 1.4	53.1 ± 0.8	52.9 ± 2.4	52.8 ± 1.5
Ours	55.2 ± 0.7	55.8 ± 1.4	56.5 ± 0.9	**55.8** ± 1.0

Table 3: CRC dataset performance on unseen architectures.

image-per-class synthetic dataset is distilled using a ConvNet (4) as the backbone architecture. To evaluate cross-architecture generalization, we use the distilled set to train AlexNet (6), VGG11 (7) and ResNet18 (8) from scratch and record the validation performance. For both tested dataset, our method's addition of the generation prior slightly or significantly improved the cross-architecture generalization of all 3 methods.

4.3 Hyper-Parameters and Experimental Details

In Table 4 and Table 5, we show the performance of the distilled images on the backbone architecture(the architecture used for distillation). The performance is tested on different settings of hyper-parameters. We fix the backbone architecture as ConvNet and we evaluate the synthetic images on the model every 100 iteration and evaluate 3 times each time. We fix the batch size as 256 and we run the whole distillation process for 3000 iterations.

The results show that more images per class(ipc) and more number of synthetic steps per iteration (syn_steps) are likely to improve the overall performance. But the synthetic steps should not be set to a very large number, for this will cause overfitting on the training data. It's often the best choice to set the T^+ (max_start_epoch) we choose in the algorithm 1 as input to 1 and the number of epoch that the expert trajectory needs to distill (expert_epochs) to 2. This is because large max start epoch may make the distillation process less stable, triggering poor performance. There are many different learning rates need to be tuned: the learning rate for updating $\mathcal{D}_{syn}$ (lr_img), for updating the synthetic latent w (lr_w) and for updating the GAN weights (lr_g). The best choice for these learning rates values is subtle and they are subject to the specific dataset. The AUC or ACC for recorded in the tables are the best result we can get during the distillation process (too much distillation may distill noise into the synthetic dataset or make distilled dataset overfit to the training data, leading to worse performance).

5 Conclusion

In this paper, we first introduce the basic logic of Dataset Distillation, and then elaborate on our method of employing deep generative prior and matching training trajectories. Experiments on the PG and CRC datasets show that our method results in a prominent increase in ACC / AUC compared to state-of-the-art methods DD and MTT. Besides that, we have done experiments to sort out the best configurations for hyper-parameters and the performance skyrockets to 74% for the AUC metric and 49.6% for the ACC metric. We also show promising improvements in cross-model generalization. All in all, with the help of our methodology, The impact of enhancements on ACC and AUC(Area under the ROC Curve) of the model is profound, showcasing the potential of our method to push the boundaries of performance in the task of dataset distillation and tackle the imbalance problem of medical datasets to a great extent.

References

[1] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A. Efros, Jun-Yan Zhu. Dataset Distillation by Matching Training Trajectories. In CVPR, 2022.

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. NeurIPS, 2014.

[3] Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation. In ICML, 2021.

ipc	max_start_epoch	syn_steps	expert_epochs	lr_img	lr_w	lr_g	AUC
1	5	30	3	1E+04	1E+01	1E-01	59.8
2	5	30	3	1E+04	1E+01	1E-01	59.7
5	5	30	3	1E+04	1E+01	1E-01	60.4
10	5	30	3	1E+04	1E+01	1E-01	65.3
20	5	30	3	1E+04	1E+01	1E-01	72.1
5	5	30	3	1E+04	1E+01	1E+00	58.6
5	5	30	3	1E+05	1E+01	1E-01	58.4
5	20	30	3	1E+04	1E+01	1E-01	52.2
5	5	30	3	1E+04	1E+02	1E-01	57.9
5	1	30	3	1E+04	1E+01	1E-01	61.3
5	1	50	3	1E+04	1E+01	1E-01	63.5
5	1	50	3	1E+04	1E+00	1E-01	63.7
5	1	50	4	1E+04	1E+00	1E-01	61.4
5	1	50	2	1E+04	1E+00	1E-01	**64.2**
5	1	40	2	1E+04	1E+00	1E-01	63.4
7	1	50	2	1E+04	1E+00	1E-01	65.8
10	1	50	2	1E+04	1E+00	1E-01	**66.9**
20	1	50	2	1E+04	1E+00	1E-01	**74.0**
5	1	100	2	1E+04	1E-01	1E-01	57.6

Table 4: CRC performance on different settings of hyper-parameters

ipc	max_start_epoch	syn_steps	expert_epochs	lr_img	lr_w	lr_g	ACC
1	1	50	2	1E+04	1E+01	1E-01	35.5
1	1	50	2	1E+03	1E+01	1E-01	37.1
1	1	50	2	1E+02	1E+01	1E-01	36.4
1	1	50	2	1E+03	1E+00	1E-01	38.5
1	1	50	2	1E+03	1E+00	1E-02	**39.7**
1	2	50	2	1E+03	1E+00	1E-02	37.7
1	1	40	2	1E+03	1E+00	1E-02	38.8
1	1	60	2	1E+03	1E+00	1E-02	39.2
1	1	50	3	1E+03	1E+00	1E-02	39.3
5	1	50	2	1E+03	1E+00	1E-02	**45.9**
10	1	50	2	1E+03	1E+00	1E-02	**48.1**
20	1	50	2	1E+03	1E+00	1E-02	**49.6**

Table 5: PG performance on different settings of hyper-parameters

[4] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In CVPR, 2018.

[5] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In NeurIPS, 2012.

[7] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, 2016.

A. Appendix

A.1. Introduction to Dataset Distillation

The implementation of *Dataset Distillation* involves selecting representative samples from the original dataset and training a model on this distilled dataset. The process of *Dataset Distillation* typically involves the following steps:

Algorithm 2 Dataset Distillation

Input: $p(\theta_0)$ - distribution of initial weights;
Input: M - number of distilled data
Input: $x^{\sim} = \{x_i^{\sim}\}_{i=1}^{M}$ randomly
Input: α - step size;
Input: n - batch size;
Input: T - number of optimization iterations;
Input: $\tilde{\eta}_0$ - initial value for $\tilde{\eta}$
▷ Initialize , $\tilde{\eta} \leftarrow \tilde{\eta}_0$
▷ Initialize trainable learning rate: $\alpha := \alpha_0$
for each training step$t = 1$ to T do **do**
  ▷ Get a minibatch of real training data $x_t = \{x_{t,j}\}_{j=1}^{n}$
  ▷ Sample a batch of initial weights $\theta_0^{(j)} \sim p(\theta_0)$
  for each sampled$\theta_0^{(j)}$ do **do**
    ▷ Compute updated parameter with GD: $\theta_1^{(j)} = \theta_0^{(j)} - \tilde{\eta}\nabla_{\theta_0^{(j)}}\mathcal{L}(x^{\sim}, \theta_0^{(j)})$
    ▷ Evaluate the objective function on real training data: $\mathcal{L}^{(j)} = \mathcal{L}(x_t, \theta_1^{(j)})$
  end for
  ▷ Update $x^{\sim} \leftarrow x^{\sim} - \alpha\nabla_{x^{\sim}} \sum_j \mathcal{L}^{(j)}$, and $\tilde{\eta} \leftarrow \tilde{\eta} - \alpha\nabla_{\tilde{\eta}} \sum_j \mathcal{L}^{(j)}$
end for
Output: Distilled data $x^{\sim}$ and optimized learning rate $\tilde{\eta}$

You can check more information here: `https://arxiv.org/abs/1811.10959`

A.2. Introduction to MTT

Because *Dataset Distillation* requires huge condensation and want to preserve as much as information as possible, this would generate many problems. For example, it requires huge compute and memory and suffer from inexact relaxation or training instability of unrolling many iterations. Besides that, problems also include focus on short-range behavior, enforcing a single training step on distilled data to match that on real data. However, error may accumulate in evaluation, resulting in huge difference between the original and distilled dataset.

MTT Distillation matches segments of parameter trajectories trained on synthetic data with segments of pre-recorded trajectories from models trained on real data and thus avoid being short-sighted (i.e., focusing on single steps) or difficult to optimize (i.e., modeling the full trajectories).And it considers the induced sequence of network parameters to be an expert trajectory.

The basic logic of *MTT Distillation* is like this: firstly, train a set of models from scratch on the real dataset and record their expert training trajectories, then initialize a new model with a random time step from a randomly chosen expert trajectory and train for several iterations on the synthetic dataset, finally, penalize the distilled data based on how far this synthetically trained network deviated from the expert trajectory and back-propagate through the training iterations.

Algorithm 3 MTT Distillation

Input: $\{\tau_i^*\}$: set of expert parameter trajectories trained on D_{real}.
Input: M: # of updates between starting and target expert params
Input: N: # of updates to student network per distillation step
Input: A: Differentiable augmentation function
Input: $T < T^+$: Maximum start epoch
Input: $\tilde{\eta}_0$ - initial value for $\tilde{\eta}$

▷ Initialize distilled data $D_{syn} \sim D_{real}$
▷ Initialize trainable learning rate $\alpha := \alpha_0$
for each distillation step...do **do**
  ▷ Sample expert trajectory: $\tau^* \sim \{\tau_i^*\}$ with $\tau^* = \{\theta_t^*\}_{T_0}$
  ▷ Choose random start epoch, $t \leq T^+$
  ▷ Initialize student network with expert params: $\hat{\theta}_t := \theta_t^*$
  for $n = 0 \rightarrow N - 1$ do **do**
    ▷ Sample a mini-batch of distilled images: $b_{t+n} \sim D_{syn}$
    ▷ Update student network w.r.t. classification loss: $\hat{\theta}_{t+n+1} = \hat{\theta}_{t+n} - \alpha \nabla'(A(b_{t+n}); \hat{\theta}_{t+n})$
  end for
  ▷ Compute loss between ending student and expert params: $L = \frac{\|\hat{\theta}_{t+N} - \theta_{t+M}^*\|_2^2}{\|\theta_t^* - \theta_{t+M}^*\|_2^2}$
  ▷ Update D_{syn} and α with respect to L
end for

Output: distilled data D_{syn} and learning rate α

A.3. Visualizations

Here presents some visualizations of our distilled medical images.

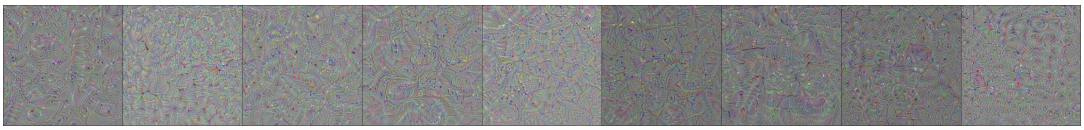

Figure 3: Distilled PG dataset. 3 images per class.

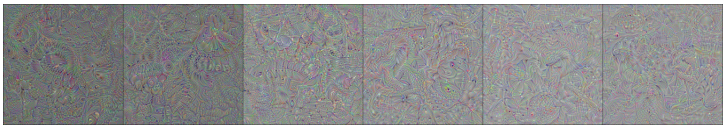

Figure 4: Distilled CRC dataset. 3 images per class.

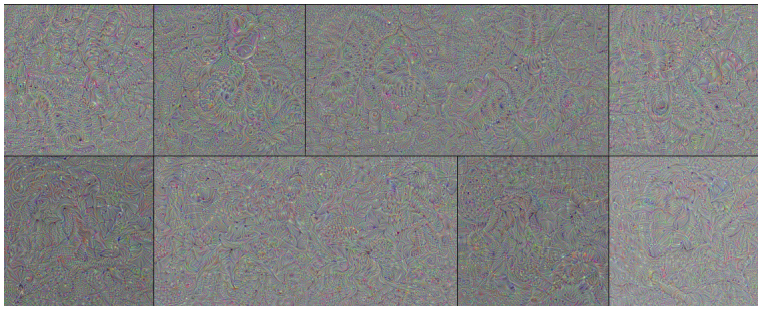

Figure 5: Distilled CRC dataset. 5 images per class.