Dataset Distillation: Approaches for Image and Text Data

Rishab Sonthalia, Mayukh Maity, Karan Kumawat, Mulla Meharaj

220150035, 220150033, 220150005, 220150009

***Abstract*—Dataset distillation aims to create compact synthetic datasets that approximate the performance of large-scale original datasets. This report explores two key approaches: gradient matching and distribution matching for image data (CIFAR10, MNIST, etc.) and extends these techniques to text data distillation. We demonstrate the effectiveness of these methods through detailed experiments, highlighting the performance improvements and limitations of both.**

I. Introduction

Dataset distillation generates smaller, synthetic datasets that approximate the performance of larger, real-world datasets, reducing the computational costs of training machine learning models. This is particularly valuable in scenarios where data collection is expensive or time-consuming. The primary techniques for distillation are Gradient Matching (GM) and Distribution Matching (DM).

Gradient matching aligns the gradients during training with synthetic data, ensuring the model's learning trajectory mirrors that of a model trained on the full dataset. Distribution matching, in contrast, aligns the statistical distributions of synthetic and real datasets, preserving diversity and representativeness.

In this report, we apply GM and DM to image datasets such as CIFAR10, MNIST, FashionMNIST, and CIFAR100, and extend these methods to text data distillation by using embedding techniques to transform textual data into continuous vector spaces, enabling synthetic text generation for NLP tasks.

The following sections detail the methodologies used in both image and text distillation, followed by a discussion of their effectiveness.

II. Methods

A. Gradient Matching

Gradient matching is a single-step parameter matching method that ensures that the gradients for the synthetic dataset are consistent with those of the real dataset. First proposed by Zhao et al., this method involves alternating between training on the synthetic and real datasets while minimizing the gradient discrepancy.

The key idea is to ensure that the gradients calculated for synthetic data S during training are similar to the gradients calculated for the real dataset T. The process alternates between updates to the model parameters from both datasets, minimizing the difference between their gradients through a loss function.

The gradient matching approach can be formalized mathematically as follows:

$$L(S,T) = \mathbb{E}_{\theta^{(0)}}\left[\sum_{t=0}^{T} D(S,T;\theta^{(t)})\right] \tag{1}$$

Where:

- $L(S,T)$ is the loss function that quantifies the gradient discrepancy between the synthetic dataset S and the real dataset T.
- $D(S,T;\theta^{(t)})$ represents the distance between the gradients obtained by training on synthetic and real datasets at iteration t.

The distance is calculated using cosine similarity, which ensures that the directions of gradient updates are consistent across datasets.

Algorithm Steps:

To better understand the gradient matching procedure, we outline the following steps:

1) **Initialize** the synthetic dataset S (can be random noise or real data).
2) **Initialize** the real dataset T.
3) **Initialize** model parameters $\theta^{(0)}$ (random initialization).
4) **Set** the learning rate η and gradient threshold ϵ.
5) **Start training loop**:
   a) **Train the model on synthetic data** S.
   b) Compute the gradient of the loss function w.r.t. model parameters $\nabla_\theta L(S)$.
   c) **Train the model on real data** T.
   d) Compute the gradient of the loss function w.r.t. model parameters $\nabla_\theta L(T)$.
   e) **Compute the gradient discrepancy** $D(S,T;\theta)$ using cosine similarity:
$$D(S,T;\theta) = \frac{\nabla_\theta L(S) \cdot \nabla_\theta L(T)}{|\nabla_\theta L(S)||\nabla_\theta L(T)|}$$
6) **Minimize the loss function** $L(S,T)$ by updating the model parameters θ.
7) **Update the synthetic dataset** S iteratively to match the real dataset's gradients.
8) **Check for convergence:**
   a) If $D(S,T;\theta) < \epsilon$ (gradient discrepancy is below threshold), **stop the algorithm**.

Explanation of Steps:

In the initialization phase, the synthetic dataset S is initialized either randomly or with real data. The real dataset T is also initialized.

The model is trained iteratively on both S and T, with alternating gradient updates. The gradient discrepancy between the synthetic and real datasets is computed using cosine similarity, which measures the similarity between their gradients.

The model parameters are updated to minimize the gradient discrepancy. The synthetic dataset S is updated iteratively to align its gradients with those of the real dataset.

The process continues until convergence is reached, either by a predefined number of iterations or when the gradient discrepancy becomes sufficiently small (i.e., below threshold ϵ).

The goal of this method is to generate synthetic data whose gradients are consistent with those of the real dataset, thus ensuring that the synthetic data retains the important characteristics of the real data.

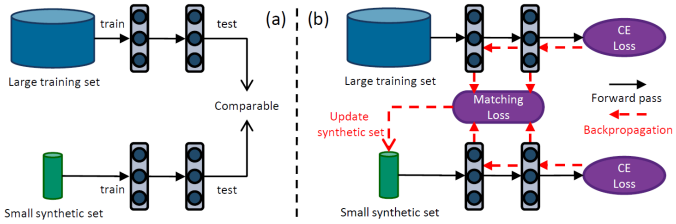

Fig. 1. Architecture of the Gradient Matching Method

B. Distribution Matching

Distribution matching aims to align the statistical properties of synthetic and real data distributions. The core idea is to ensure that the synthetic data mimics the underlying distribution of the original dataset, rather than focusing on gradient trajectories. Zhao et al. proposed using Maximum Mean Discrepancy (MMD) as a metric to measure the distance between the distributions of synthetic and real data.

The objective of distribution matching is to minimize the distance between the distributions of the real and synthetic datasets. This is achieved by using neural networks to extract features from both datasets and then computing the distance between the statistical summaries (mean and covariance) of these features. The loss function can be formulated as:

$$L(S,T) = \mathbb{E}_{\theta\in\Theta}[D(S,T;\theta)] \tag{2}$$

Where:

- $D(S,T;\theta)$ measures the distance between the distributions of synthetic and real data, using MMD or other metrics.
- Θ represents the parameter space of the neural network used to extract the features from both datasets.

The intuition behind distribution matching is that instead of directly estimating the real data distribution, which can be expensive and inaccurate (especially for high-dimensional data like images), distribution matching approximates it using neural networks. These networks serve as embedding functions, each offering a partial interpretation of the data, and their combination offers a comprehensive interpretation of the data distributions.

In the context of Zhao et al.'s work, distribution matching is defined as:

$$L(S,T) = \mathbb{E}_{\theta\in\Theta}[D(S,T;\theta)] \tag{3}$$

Where:

- S and T represent synthetic and real datasets, respectively.
- $D(S,T;\theta)$ is a distance function that measures how close the distributions of S and T are.

Maximum Mean Discrepancy (MMD): The MMD metric is used to calculate the distance between two distributions. For a given dataset S (synthetic) and T (real), the MMD is computed as the difference between the means of the feature mappings of both datasets. The function f_θ is used to map the data points to a high-dimensional feature space where the distance between distributions can be effectively measured. The formal MMD formulation is given by:

$$D(S,T;\theta) = \sum_{c=0}^{C-1} \|\mu_{\theta,s,c} - \mu_{\theta,t,c}\|_2^2 \tag{4}$$

Where:

- $\mu_{\theta,s,c}$ and $\mu_{\theta,t,c}$ are the means of the feature mappings for the synthetic and real samples of class c.
- C is the number of classes in the dataset.
- $\|\cdot\|_2^2$ represents the squared L_2-norm between the mean vectors.

CAFE Method: In some cases, instead of matching features before the last layer, Wang et al. propose the CAFE (Class-Aware Feature Embedding) method, which forces the statistics of features extracted from each layer of a neural network (except the final one) to be consistent across both synthetic and real datasets. This is a more flexible approach as it aligns features at each layer, not just at the final output. The corresponding loss function for this approach is:

$$D(S,T;\theta) = \sum_{c=0}^{C-1}\sum_{i=0}^{L-1} \|\mu_{\theta,s,c}^{(i)} - \mu_{\theta,t,c}^{(i)}\|_2^2 \tag{5}$$

Where:

- L is the number of layers in the neural network.
- i represents the layer index.
- The feature statistics are computed for each layer, and the distance is minimized across all layers.

Discriminative Loss : Additionally, the CAFE method incorporates a discriminative loss D_r to explicitly learn discriminative synthetic images. The discriminative loss encourages the network to classify real samples using the synthetic samples as classifiers. The loss is defined as:

$$D_r(S,T;\theta) = -\sum_{c=0}^{C-1}\sum_{j=1}^{N_c} \log p(c|X_{t,c}^{(j)}, S, \theta) \tag{6}$$

Where:

- $p(c|X_{t,c}^{(j)}, S, \theta)$ is the probability of class c given the input $X_{t,c}^{(j)}$ from the real dataset T and synthetic dataset S, computed by the classifier.
- The term $\mu_{\theta,s,c}^{(L-1)}$ corresponds to the embedding output of the penultimate layer.

Algorithm Steps for Distribution Matching: The following steps describe the algorithm for distribution matching:

1) **Initialize** the synthetic dataset S and real dataset T.
2) **Initialize** the neural network model parameters $\theta^{(0)}$.
3) **Set** hyperparameters: learning rate η, balance factor λ, and convergence threshold.
4) **Train the network on synthetic data** S and compute the feature embeddings for each class.
5) **Train the network on real data** T and compute the feature embeddings for each class.
6) **Compute the distance** between the distributions using MMD (Equation 4).
7) **Minimize the MMD loss** by updating the model parameters θ.
8) **If CAFE is used, incorporate discriminative loss** D_r and update the model accordingly.
9) **Repeat** until convergence (i.e., when the distance between the distributions becomes small or the maximum number of iterations is reached).

The distribution matching approach allows the synthetic data to more closely approximate the distribution of the real data, ensuring that the synthetic data captures the key statistical features necessary for training machine learning models.

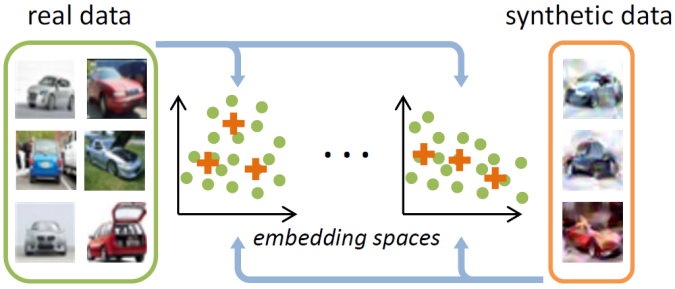

Fig. 2. Architecture of the Distribution Matching Method

C. Text Data Distillation

Text data distillation follows a similar approach to image data distillation but involves adapting the methods to the unique structure of text data. However, it is difficult to use gradient methods directly on text data as it is discrete. In order to be able to use Soft Label Dataset Distillation with text data, we need to first embed the text data into a continuous space. This is a common practice when working with many modern natural language processing models, though the embedding method itself can vary greatly (Ma and Hovy, 2016; Devlin et al., 2018; Peters et al., 2018). Any popular embedding method can be used, we used pre-trained GloVe embeddings (Pennington et al., 2014). Once the text is embedded into a continuous space, the problem of distilling it becomes analogous to soft-label image distillation. If all sentences are padded/truncated to some pre-determined length, then each sentence is essentially just a one-channel image of size [length] x [embedding dimension].

*1) **Soft Label Dataset Distillation(SLDD):*** Soft label dataset distillation is an advanced technique in which the distilled dataset is created with "soft" labels, instead of hard, discrete class labels. These soft labels represent probabilistic distributions over classes rather than definitive class assignments, providing richer information about the data's underlying structure. By utilizing soft labels, the distilled dataset captures more nuanced information, which can be particularly beneficial when the original dataset is complex or contains ambiguous instances. This approach can improve the training of machine learning models by making the synthetic data more informative and representative of the true data distribution.

Algorithm 1a Soft-Label Dataset Distillation (SLDD)

Input: $p(\theta_0)$: distribution of initial weights; M: the number of distilled data; α: step size; n: batch size; T: number of optimization iterations; $\tilde{y}_0$: initial value for $\tilde{y}$; $\tilde{\eta}_0$: initial value for $\tilde{\eta}$

Initialize distilled data $\tilde{\mathbf{x}} = \{\tilde{x}_i\}_{i=1}^{M}$ randomly, $\tilde{\mathbf{y}} = \{\tilde{y}_i\}_{i=1}^{M} \leftarrow \tilde{y}_0$, $\tilde{\eta} \leftarrow \tilde{\eta}_0$
for each training step t = 1 to T **do**
  Get a mini-batch of real training data $(\mathbf{x}_t, \mathbf{y}_t) = \{x_{t,j}, y_{t,j}\}_{j=1}^{n}$
  One-hot encode the labels $(\mathbf{x}_t, \mathbf{y^*}_t) = \{x_{t,j}, \text{Encode}(y_{t,j})\}_{j=1}^{n}$
  Sample a batch of initial weights $\theta_0^{(j)} \sim p(\theta_0)$
  for each sampled $\theta_0^{(j)}$ **do**
    Compute updated model parameter with GD $\theta_1^{(j)} = \theta_0^{(j)} - \tilde{\eta}\nabla_{\theta_0^{(j)}}\ell\left(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}, \theta_0^{(j)}\right)$
    Evaluate the objective function on real training data: $\mathcal{L}^{(j)} = \ell\left(\mathbf{x}_t, \mathbf{y^*}_t, \theta_1^{(j)}\right)$
  end for
  Update distilled data $\tilde{\mathbf{x}} \leftarrow \tilde{\mathbf{x}} - \alpha\nabla_{\tilde{\mathbf{x}}}\sum_j \mathcal{L}^{(j)}$, $\tilde{\mathbf{y}} \leftarrow \tilde{\mathbf{y}} - \alpha\nabla_{\tilde{\mathbf{y}}}\sum_j \mathcal{L}^{(j)}$, and $\tilde{\eta} \leftarrow \tilde{\eta} - \alpha\nabla_{\tilde{\eta}}\sum_j \mathcal{L}^{(j)}$
end for

Output: distilled data $\tilde{\mathbf{x}}$; distilled labels $\tilde{\mathbf{y}}$; optimized learning rate $\tilde{\eta}$

Fig. 3. Soft Label Dataset Distillation Algorithm

*2) **Text Data Distillation Methodology:*** Text data distillation represents an innovative approach to condensing and representing large textual datasets through advanced embedding techniques. Unlike traditional data compression methods, this technique focuses on capturing the essential semantic characteristics of textual information within a compact, mathematically representable space.

a) Embedding Transformation: The fundamental challenge in text data distillation lies in transforming discrete textual data into a continuous embedding space. By utilizing pre-trained word embedding models like GloVe, researchers can convert individual words and sentences into high-dimensional vector representations. These embeddings capture nuanced semantic relationships, allowing complex linguistic information to be represented as mathematical coordinates in a continuous vector space.

b) Synthetic Data Generation: The core of the distillation process involves generating synthetic sentences that encapsulate the critical informational and distributional characteristics of the original dataset. These synthetic representations are not mere approximations but carefully optimized constructs that maintain the essential statistical properties of the source text. The generation process employs sophisticated optimization techniques to ensure that the synthetic data preserves the key linguistic patterns and semantic structures of the original corpus.

c) Optimization Dynamics: Optimization occurs through an iterative process where synthetic sentences and their associated soft labels are progressively refined. This involves computing gradient-based updates that minimize the discrepancy between gradients derived from real and synthetic data. The soft labels, represented as continuous values rather than discrete class assignments, provide additional flexibility in capturing the nuanced probabilistic nature of textual information.

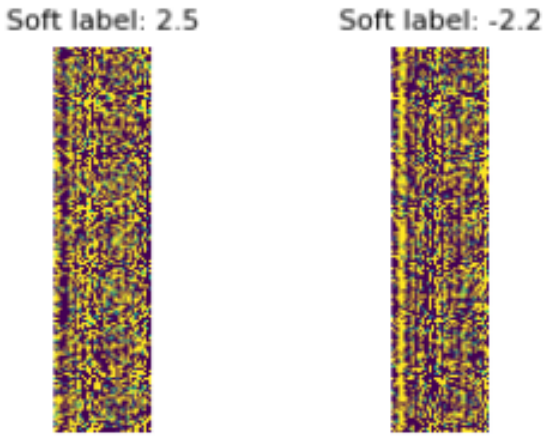

Fig. 4. TDD can learn 2 distilled sentences for binary sentiment classification task. Each image corresponds to a distilled embedding.

III. Results

In this section, we present and analyze the results of applying Gradient Matching (GM) and Distribution Matching (DM) on multiple image datasets, including MNIST, FashionMNIST, SVHN, CIFAR10, and CIFAR100. We evaluate the performance using various settings of the synthetic dataset, measured by the number of synthetic images per class. The results are shown in both tabular and graphical formats.

A. Gradient Matching (GM) Results

We first present the performance of the gradient matching approach on image datasets, summarized in Table I. The table shows the accuracy of the distillation method for various numbers of synthetic images per class (1, 10, and 50 images per class) across the five datasets.

Dataset	1 img/cls	10 img/cls	50 img/cls
MNIST	88.7	97.8	99.2
FashionMNIST	70.6	84.6	88.7
SVHN	27.5	79.2	84.4
CIFAR10	28.8	52.1	60.6
CIFAR100	13.9	32.3	-

TABLE I
METRICS FOR IMAGE DATA DISTILLATION (GRADIENT MATCHING)

Interpretation of Results for GM:

- **MNIST**: With just 1 synthetic image per class, the model achieves an accuracy of 88.7%, which increases to 99.2% when 50 images/class are used. This indicates that GM can effectively generate synthetic data that closely matches the real data distribution.
- **FashionMNIST**: The accuracy improves from 70.6% to 88.7% as the number of synthetic images increases, showing that GM can handle more complex data distributions, although its performance is still higher for simpler datasets like MNIST.
- **SVHN**: The performance improves significantly from 27.5% with 1 image/class to 84.4% with 50 images/class. This demonstrates GM's ability to enhance synthetic data quality, though it still lags behind simpler datasets like MNIST and FashionMNIST.
- **CIFAR10**: Accuracy increases from 28.8% to 60.6% as the number of synthetic images increases, suggesting that GM faces challenges in capturing the more complex diversity of CIFAR10 images, though there is moderate improvement as more synthetic data is used.
- **CIFAR100**: The performance with 1 and 10 images per class is reported, but the result for 50 images/class is not available. This indicates either an incomplete dataset or difficulty in generating synthetic data for the highly complex CIFAR100 dataset, which has more classes and intricate image features.

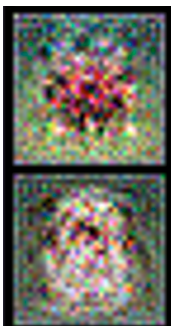

Fig. 5. Visualization of Synthetic Data Generated by Gradient Matching

B. Synthetic Data Visualizations

We also provide visualizations of the synthetic data generated by the GM method for different datasets. The images below demonstrate how the synthetic data aligns with real samples across various datasets. These visualizations help to qualitatively assess the effectiveness of GM in generating realistic images that match the distribution of real data.

The synthetic data images show that GM is able to generate reasonably realistic images, especially for simpler datasets like MNIST and FashionMNIST. However, as the complexity of the dataset increases (e.g., CIFAR10 and CIFAR100), the quality of the synthetic images slightly deteriorates, reflecting the challenges GM faces in capturing intricate data distributions.

C. Distribution Matching (DM) Results

We now turn to the results of the Distribution Matching (DM) approach. DM directly optimizes the distance between the distributions of synthetic and real data using a metric such as Maximum Mean Discrepancy (MMD). The results for DM are presented in the following graph, which shows the accuracy of performance matching for the CIFAR10 dataset.

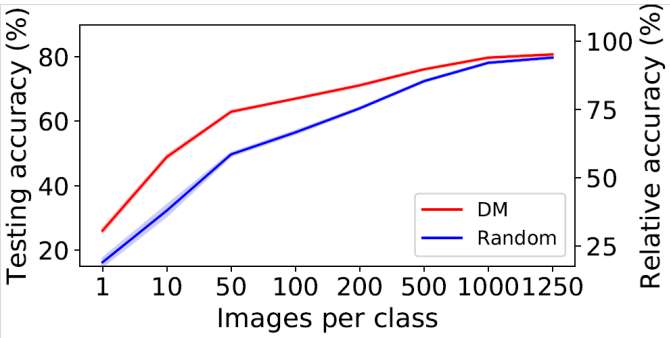

Fig. 6. Performance of Distribution Matching on CIFAR10 Dataset

Interpretation of Results for DM:

- As seen from the graph, the accuracy for performance matching improves with the increase in the number of synthetic images per class. Performance gains are more significant when the size of the synthetic dataset is increased from 1 image/class to 10 and 50 images/class. This indicates that DM benefits from larger synthetic datasets.
- DM demonstrates significantly better results compared to GM, especially for more complex datasets like CIFAR10. This suggests that DM is more effective in aligning the synthetic data distribution with real data, resulting in improved accuracy and more realistic synthetic images.

D. Comparison of GM and DM

When comparing GM and DM, it is evident that while both methods show improvement as the number of synthetic images per class increases, DM generally outperforms GM, particularly for more complex datasets like CIFAR10. The ability of DM to directly match distributions allows for better alignment with real data, resulting in higher accuracy and more realistic synthetic images.

Key Observations:

- **For Simpler Datasets (MNIST, FashionMNIST)**: Both GM and DM perform well, but DM slightly outperforms GM in terms of accuracy and synthetic data quality. While GM performs well on these datasets, DM's distribution matching approach results in better synthetic data and higher accuracy.
- **For More Complex Datasets (CIFAR10, CIFAR100)**: DM shows significantly better results than GM. For CIFAR10, DM achieves higher accuracy and generates more realistic synthetic images, highlighting its ability to capture complex data distributions effectively.
- **Synthetic Data Visualization**: The DM-generated images are qualitatively superior to GM-generated images, particularly for high-dimensional datasets like CIFAR10. The synthetic images generated by DM are closer to the real data in terms of both visual appearance and distribution.

E. Text Data Distillation Results

For text data distillation, we will provide similar performance metrics and accuracy graphs, following the same format as the image data results. The results for text datasets (e.g., IMDB, SST5, etc.) and will compare them against traditional machine learning algorithms and deep learning techniques used for sentiment analysis.

1) Text Dataset Distillation Accuracy Across Datasets: We will illustrate the performance of different models across multiple datasets. The TDD Accuracy is compared against the performance of training data after 10 gradient descent steps and K-NN performance. Error bars are included for training data and K-NN performance to show variability. The Internet Movie Database (IMDB) dataset is a widely used benchmark for natural language processing (NLP) tasks, particularly for sentiment analysis. It consists of movie reviews and includes labels indicating whether each review is positive or negative. SST5 includes five sentiment categories: very negative, negative, neutral, positive, and very positive. The TREC6 dataset is part of the Text Retrieval Conference (TREC) Question Answering track. It consists of questions categorized into six different classes. TREC50 has 50 classes.

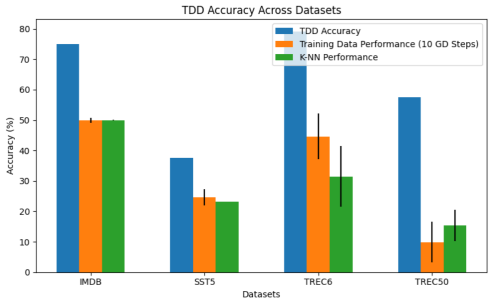

Fig. 7. TDD Accuracy Across Datasets

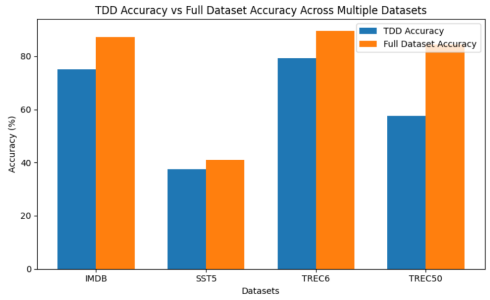

Fig. 8. TDD Accuracies against models trained on full datasets

2) Text Dataset Distillation Accuracy on Custom Twitter Comments Dataset: Additionally, we have trained the TDD model on our custom Twitter dataset for sentiment analysis with three sentiment categories: 0 (negative), 1 (neutral), and 2 (positive). This custom dataset allows for evaluating the model's performance on social media text, which often presents unique challenges due to its informal and varied language.

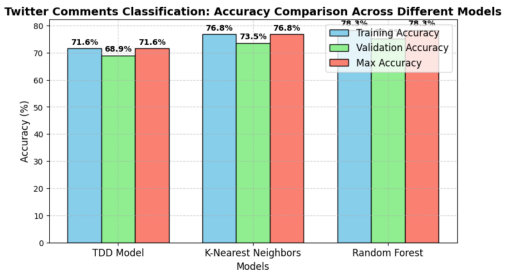

Fig. 9. Custom Twitter Comments Sentiment Analysis

IV. Discussion and Future Work

This study highlights the effectiveness of both Gradient Matching (GM) and Distribution Matching (DM) techniques in dataset distillation, particularly for image data. GM excels in scenarios where fewer images per class are available, as it directly optimizes for local gradient consistency. In contrast, DM, which focuses on aligning entire data distributions, performs better when more images per class are available, particularly for complex datasets like CIFAR10. The global alignment of data distributions provided by DM results in more realistic and coherent synthetic data.

For text data distillation, a similar framework is applied, but with adaptations to handle the unique structure of text. Due to the discrete nature of text, embedding techniques, such as GloVe, are essential for transforming text into a continuous space. This transformation allows for the use of soft-label dataset distillation (SLDD) methods. Once embedded, text distillation becomes analogous to image distillation, where each sentence is treated as a "one-channel image" in the embedding space, enabling more efficient synthetic data generation.

Looking ahead, future work will focus on hybrid methods that combine the strengths of GM and DM, improving the generalization and realism of synthetic datasets. Additionally, we plan to explore advanced models for text distillation, particularly transformer-based architectures, and evaluate the scalability of these approaches for larger, more diverse text datasets.

By integrating gradient and distribution matching techniques, along with advancing text distillation models, future research will enhance the quality, scalability, and applicability of synthetic data generation across various domains, from computer vision to natural language processing.

V. References

- Ruonan Yu, Songhua Liu, Xinchao Wang, *Dataset Distillation: A Comprehensive Review*.
- Ilia Sucholutsky, Matthias Schonlau, *Soft-Label Dataset Distillation and Text Dataset Distillation*.
- George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A. Efros, Jun-Yan Zhu, *Dataset Distillation by Matching Training Trajectories*.
- Bo Zhao, Hakan Bilen, *Dataset Condensation with Distribution Matching*.