Data Distillation: A Survey

Anonymous authors
Paper under double-blind review

Abstract

The popularity of deep learning has led to the curation of a vast number of massive and multifarious datasets. Despite having close-to-human performance on individual tasks, training parameter-hungry models on large datasets poses multi-faceted problems such as (a) high model-training time; (b) slow research iteration; and (c) poor eco-sustainability. As an alternative, *data distillation* approaches aim to synthesize terse data summaries, which can serve as effective drop-in replacements of the original dataset for scenarios like model training, inference, architecture search, *etc.* In this survey, we present a formal framework for data distillation, along with providing a detailed taxonomy of existing approaches. Additionally, we cover data distillation approaches for different data modalities, namely images, graphs, and user-item interactions (recommender systems), while also identifying current challenges and future research directions.

1 Introduction

(Loose) Definition 1. (Data distillation) *Approaches that aim to synthesize tiny and high-fidelity data summaries which distill the most important knowledge from a given target dataset. Such distilled summaries are optimized to serve as effective drop-in replacements of the original dataset for efficient and accurate data-usage applications like model training, inference, architecture search, etc.*

The recent "scale-is-everything" viewpoint (Ghorbani et al., 2021; Hoffmann et al., 2022; Kaplan et al., 2020), argues that training bigger models (*i.e.*, consisting of a higher number of parameters) on bigger datasets, and using larger computational resources is the sole key for advancing the frontier of artificial intelligence. On the other hand, a well-reasoned, principled solution will arguably be more amenable to scaling, thereby leading to faster progress (Sorscher et al., 2022). Data distillation (Definition 1), is a task rooted in the latter school of thought. Clearly, the scale viewpoint still holds, in that if we keep increasing the amount of data (albeit now compressed and of higher quality), we will observe an improvement in both upstream and downstream generalization, but at a faster rate.

Motivation. A terse, high-quality data summary has use cases from a variety of standpoints. First and foremost, it leads to a faster model-training procedure. In turn, faster model training equates to (1) compute-cost saving and expedited research iterations, *i.e.*, the investigative procedure of manually experimenting different ideas; and (2) improved eco-sustainability, *i.e.*, lowering the amount of compute time directly leads to a lower carbon footprint from running power-hungry accelerated hardware (Gupta et al., 2022). Additionally, a small data summary democratizes the entire pipeline, as more people can train state-of-the-art algorithms on reasonably accessible hardware using the data summary. Finally, a high-quality data summary indirectly also accelerates orthogonal procedures like neural architecture search (Liu et al., 2019), approximate nearest neighbour search (Arya et al., 1998), knowledge distillation (Hinton et al., 2015), *etc.*, where the procedure needs to iterate over the entire dataset multiple times.

Comparison with knowledge distillation & transfer learning. Despite inherently distilling some kind of knowledge, we would like to highlight both *knowledge distillation* and *transfer learning* are orthogonal procedures to data distillation, which can potentially work together to perform both tasks more efficiently. Knowledge distillation (Hinton et al., 2015) entails distilling the knowledge from a trained teacher network into

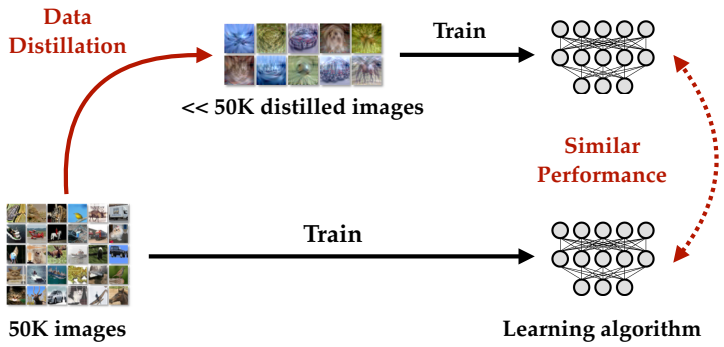

Figure 1: The premise of data distillation demonstrated using an image dataset.

a smaller student network efficiently. On the other hand, transfer learning (Pratt, 1992) is the procedure of transferring knowledge across similar tasks, *e.g.*, from image classification to image segmentation. Orthogonally, data distillation aims to distill the knowledge from a given dataset into a terse data summary. Such data summaries can be used *in conjunction* with knowledge distillation or transfer learning procedures for both (1) faster learning of the teacher models; and (2) faster knowledge transfer to the student models. The same comparison holds true for model compression techniques (LeCun et al., 1989) as well, where similar to knowledge distillation, the goal is to reduce model storage size, rather than reducing the training time or sample complexity.

In this survey, we intend to provide a succinct overview of various data distillation frameworks across different data modalities. We start by presenting a formal data distillation framework in Section 2, along with a detailed empirical comparison of existing image distillation techniques. Subsequently, in Section 3, we discuss existing data distillation frameworks for synthesizing data of different modalities, as well as outlining the associated challenges. In Section 4, we discuss alternative applications of synthesizing a high-fidelity data summary rather than simply accelerating model training along with pointers to existing work. Finally, in Section 5, we conclude by presenting common pitfalls in existing data distillation techniques, along with proposing interesting directions for future work.

2 The Data Distillation Framework

Before going into the specifics of data distillation, we start by outlining useful notation. Let $\mathcal{D} \triangleq \{(x_i, y_i)\}_{i=1}^{|\mathcal{D}|}$ be a given dataset which needs to be distilled, where $x_i \in \mathcal{X}$ are the set of input features, and $y_i \in \mathcal{Y}$ is the desired label for x_i. For classification tasks, let $\mathcal{C}$ be the set of unique classes in $\mathcal{Y}$, and $\mathcal{D}^c \triangleq \{(x_i, y_i) \mid y_i = c\}_{i=1}^{|\mathcal{D}|}$ be the subset of $\mathcal{D}$ with class c. We also define the matrices $\mathbf{X} \triangleq [x_i]_{i=1}^{|\mathcal{D}|}$ and $\mathbf{Y} \triangleq [y_i]_{i=1}^{|\mathcal{D}|}$ for convenience. Given a data budget $n \in \mathbb{Z}^+$, data distillation techniques aim to synthesize a high-fidelity data summary $\mathcal{D}_{\text{syn}} \triangleq \{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^{n}$ such that $n \ll |\mathcal{D}|$. We define $\mathcal{D}_{\text{syn}}^c$, $\mathbf{X}_{\text{syn}}$, and $\mathbf{Y}_{\text{syn}}$ similarly as defined for $\mathcal{D}$. Let $\Phi_\theta : \mathcal{X} \mapsto \mathcal{Y}$ represent a learning algorithm parameterized by θ. We also assume access to a twice-differentiable cost function $l : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$, and define $\mathcal{L}_{\mathcal{D}}(\theta) \triangleq \mathbb{E}_{(x,y)\sim\mathcal{D}}[l(\Phi_\theta(x), y)]$ for convenience. Notation is also summarized in Appendix A.

For the sake of uniformity, we refer to the data synthesized by data distillation techniques as a *data summary* henceforth. Inspired by the definition of coresets (Bachem et al., 2017), we formally define an $\epsilon-$approximate data summary, and the data distillation task as follows:

Definition 2. ($\epsilon-$approximate data summary) *Given a learning algorithm Φ, let $\theta^{\mathcal{D}}$, $\theta^{\mathcal{D}_{\text{syn}}}$ represent the optimal set of parameters for Φ estimated on $\mathcal{D}$ and $\mathcal{D}_{\text{syn}}$, and $\epsilon \in \mathbb{R}^+$; we define an $\epsilon-$approximate data summary as one which satisfies:*

$$\sup \left\{ \mid l\left(\Phi_{\theta^{\mathcal{D}}}(x), y\right) - l\left(\Phi_{\theta^{\mathcal{D}_{\text{syn}}}}(x), y\right) \mid \right\}_{\substack{x\sim\mathcal{X} \\ y\sim\mathcal{Y}}} \leq \epsilon \tag{1}$$

Definition 3. (Data distillation) *Given a learning algorithm Φ, let $\theta^{\mathcal{D}}$, $\theta^{\mathcal{D}_{\text{syn}}}$ represent the optimal set of parameters for Φ estimated on $\mathcal{D}$ and $\mathcal{D}_{\text{syn}}$; we define data distillation as optimizing the following:*

$$\arg\min_{\mathcal{D}_{\text{syn}},n} \left(\sup \left\{ \; \middle| \; l\left(\Phi_{\theta^{\mathcal{D}}}(x), y\right) - l\left(\Phi_{\theta^{\mathcal{D}_{\text{syn}}}}(x), y\right) \; \middle| \; \right\}_{\substack{x\sim\mathcal{X} \\ y\sim\mathcal{Y}}} \right) \tag{2}$$

From Definition 3, we highlight three cornerstones of evaluating data distillation methods: (1) Performance: downstream evaluation of models trained on the synthesized data summary *vs.* the full dataset (*e.g.*, accuracy, FID, nDCG, *etc.*); (2) Efficiency: how quickly can models reach full-data performance (or even exceed it), *i.e.*, the scaling of n *vs.* downstream task-performance; and (3) Transferability: how well can data summaries generalize to a diverse pool of learning algorithms, in terms of downstream evaluation.

No free lunch. The universal "No Free Lunch" theorem (Wolpert & Macready, 1997) applies to data distillation as well. For example, looking at the transferability of a data summary, it is strongly dependent on the set of encoded inductive biases, *i.e.*, through the choice of the learning algorithm Φ used while distilling, as well as the objective function $l(\cdot, \cdot)$. Such biases are unavoidable for any data distillation technique, in a sense that learning algorithms closely following the set of encoded inductive biases, will be able to generalize better on the data summary than others.

Keeping these preliminaries in mind, we now present a formal framework for data distillation, encapsulating existing data distillation approaches. Notably, the majority of existing techniques intrinsically solve a bilevel optimization problem, which are tractable surrogates of Equation (2). The inner-loop typically optimizes a representative learning algorithm on the data summary, and using the optimized learning algorithm, the outer-loop optimizes a tractable proxy of Equation (2).

Some common assumptions that existing data distillation techniques follow are: (1) static-length data summary, *i.e.*, n is fixed and is treated as a tunable hyper-parameter; and (2) we have on-demand access to the target dataset $\mathcal{D}$ which is also assumed to be `iid`. Notably, the outer-loop optimization of $\mathcal{D}_{\text{syn}}$ happens simply through gradient descent (GD) on the analogously defined $\mathbf{X}_{\text{syn}} \in \mathbb{R}^{n \times \dim(\mathcal{X})}$, which is instantiated as free parameters. Note that the labels, $\mathbf{Y}_{\text{syn}} \in \mathbb{R}^{n \times \dim(\mathcal{Y})}$, can be similarly optimized through GD as well (Bohdal et al., 2020). For the sake of notational clarity, we will interchangeably use optimization of $\mathcal{D}_{\text{syn}}$ *or* $(\mathbf{X}_{\text{syn}}, \mathbf{Y}_{\text{syn}})$ henceforth.

2.1 Data Distillation by Meta-model Matching

Meta-model matching-based data distillation approaches fundamentally optimize for the transferability of models trained on the data summary when generalized to the original dataset:

$$\arg\min_{\mathcal{D}_{\text{syn}}} \quad \mathcal{L}_{\mathcal{D}}\left(\theta^{\mathcal{D}_{\text{syn}}}\right) \quad \text{s.t.} \quad \theta^{\mathcal{D}_{\text{syn}}} \triangleq \arg\min_{\theta} \mathcal{L}_{\mathcal{D}_{\text{syn}}}(\theta), \tag{3}$$

where intuitively, the inner-loop trains a representative learning algorithm on the data summary *until convergence*, and the outer-loop subsequently optimizes the data summary for the transferability of the optimized learning algorithm to the original dataset. Besides common assumptions mentioned earlier, the key simplifying assumption for this family of methods is that a perfect classifier exists and can be estimated on $\mathcal{D}$, *i.e.*, $\exists\ \theta^{\mathcal{D}}$ s.t. $l(\Phi_{\theta^{\mathcal{D}}}(x), y) = 0,\ \forall x \sim \mathcal{X}, y \sim \mathcal{Y}$. Plugging the second assumption along with the `iid` assumption of $\mathcal{D}$ in Equation (2) directly translates to Equation (3). Despite the assumption, Equation (3) is highly expensive both in terms of computation time and memory, due to which, methods from this family typically resort to making further assumptions.

Wang et al. (2018) (DD) originally proposed the task of data distillation, and used the meta-model matching framework for optimization. DD makes the expensive optimization in Equation (3) more efficient by performing (1) local optimization *à la* stochastic gradient descent (SGD) in the inner-loop, and (2) outer-loop optimization using Truncated Back-Propagation Through Time (TBPTT), *i.e.*, unroll a limited number of inner-loop optimization steps while optimizing the outer-loop. Formally, the modified optimization objective for DD is as follows:

$$\arg\min_{\mathcal{D}_{\text{syn}}} \quad \mathop{\mathbb{E}}_{\theta_0 \sim \mathbf{P}_\theta} \left[\mathcal{L}_{\mathcal{D}}\left(\theta_T\right)\right] \quad \text{s.t.} \quad \theta_{t+1} \leftarrow \theta_t - \eta \cdot \nabla_\theta \mathcal{L}_{\mathcal{D}_{\text{syn}}}(\theta_t), \tag{4}$$

where $\mathbf{P}_\theta$ is a parameter initialization distribution of choice, T accounts for the truncation in TBPTT, and η is a tunable learning rate. Notably, TBPTT has been associated with drawbacks such as (1) computationally expensive to unroll the inner-loop at each outer-loop update (Vicol et al., 2021); (2) bias involved with truncated unrolling (Wu et al., 2018); and (3) poorly conditioned loss landscapes, particularly with long unrolls (Metz et al., 2019). Consequently, the TBPTT framework was empirically shown to be ineffective for data distillation in subsequent works (Zhao et al., 2021). However, recent work (Deng & Russakovsky, 2022) claims that using momentum-based optimizers and longer unrolling of the inner-loop can greatly improve performance. We delay a deeper discussion of this work to Section 2.5 for clarity.

Analogously, a separate line of work focuses on using Neural Tangent Kernel (NTK) (Jacot et al., 2018) based algorithms to solve the inner-loop in closed form. As a brief side note, the infinite-width correspondence states that performing Kernelized Ridge Regression (KRR) using the NTK of a given neural network, is equivalent to training the same ∞-width neural network with L2 reconstruction loss for ∞ SGD-steps. These "∞-width" neural networks have been shown to perform reasonably compared to their finite-width counterparts, while also being solved in closed-form (see Lee et al. (2020) for a detailed analysis on finite *vs.* infinite neural networks for image classification). KIP uses the NTK of a fully-connected neural network (Nguyen et al., 2021a), or a convolutional network (Nguyen et al., 2021b) in the inner-loop of Equation (3) for efficient data distillation. More formally, given the NTK $\mathcal{K} : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ of a neural network architecture, KIP optimizes the following objective:

$$\underset{\mathbf{X}_{\text{syn}}, \mathbf{Y}_{\text{syn}}}{\arg\min} \quad \left\| \mathbf{Y} - \mathbf{K}_{\mathbf{X}\mathbf{X}_{\text{syn}}} \cdot (\mathbf{K}_{\mathbf{X}_{\text{syn}}\mathbf{X}_{\text{syn}}} + \lambda I)^{-1} \cdot \mathbf{Y}_{\text{syn}} \right\|^2, \tag{5}$$

where $\mathbf{K}_{AB} \in \mathbb{R}^{|A| \times |B|}$ represents the gramian matrix of two sets A and B, and whose $(i,j)^{\text{th}}$ element is defined by $\mathcal{K}(A_i, B_j)$. Although KIP doesn't impose any additional simplifications to the meta-model matching framework, it has an $\mathcal{O}(|\mathcal{D}| \cdot n \cdot \dim(\mathcal{X}))$ time and memory complexity, limiting its scalability. Subsequently, RFAD (Loo et al., 2022) proposes using (1) the light-weight Empirical Neural Network Gaussian Process (NNGP) kernel (Neal, 2012) instead of the NTK; and (2) a classification loss (*e.g.*, NLL) instead of the L2-reconstruction loss for the outer-loop to get $\mathcal{O}(n)$ time complexity while also having better performance. On a similar note, FRePO (Zhou et al., 2022b) decouples the feature extractor and a linear classifier in Φ, and alternatively optimizes (1) the data summary along with the classifier, and (2) the feature extractor. To be precise, let $f_\theta : \mathcal{X} \mapsto \mathcal{X}'$ be the feature extractor, $g_\psi : \mathcal{X}' \mapsto \mathcal{Y}$ be the linear classifier, s.t. $\Phi(x) \equiv g_\psi(f_\theta(x)) \; \forall x \in \mathcal{X}$; the optimization objective for FRePO can be written as:

$$\begin{aligned} &\underset{\mathbf{X}_{\text{syn}}, \mathbf{Y}_{\text{syn}}}{\arg\min} \quad \mathop{\mathbb{E}}_{\theta_0 \sim \mathbf{P}_\theta} \left[\sum_{t=0}^{T} \left\| \mathbf{Y} - \mathbf{K}^{\theta_t}_{\mathbf{X}\mathbf{X}_{\text{syn}}} \cdot (\mathbf{K}^{\theta_t}_{\mathbf{X}_{\text{syn}}\mathbf{X}_{\text{syn}}} + \lambda I)^{-1} \cdot \mathbf{Y}_{\text{syn}} \right\|^2 \right] \\ &\text{s.t.} \quad \theta_{t+1} \leftarrow \theta_t - \eta \cdot \mathop{\mathbb{E}}_{(x,y) \sim \mathcal{D}_{\text{syn}}} \left[\nabla_\theta l(g_\psi(f_\theta(x)), y) \right] \; ; \; \mathbf{K}^{\theta}_{\mathbf{X}_{\text{syn}}\mathbf{X}_{\text{syn}}} \triangleq f_{\theta_t}(\mathbf{X}_{\text{syn}}) f_{\theta_t}(\mathbf{X}_{\text{syn}})^T, \end{aligned} \tag{6}$$

where T represents the number of inner-loop update steps for the feature extractor f_θ. Notably, (1) a wide architecture for f_θ is crucial for distillation quality in FRePO; and (2) despite the bilevel optimization, FRePO is shown to be more scalable compared to KIP (Equation (5)), while also being more generalizable.

2.2 Data Distillation by Gradient Matching

Gradient matching based data distillation, at a high level, performs one-step distance matching on (1) the network trained on the target dataset ($\mathcal{D}$) *vs.* (2) the same network trained on the data summary ($\mathcal{D}_{\text{syn}}$). In contrast to the meta-model matching framework, such an approach circumvents the unrolling of the inner-loop, thereby making the overall optimization much more efficient. First proposed by Zhao et al. (2021) (DC), data summaries optimized by gradient-matching significantly outperformed heuristic data samplers, principled coreset construction techniques, as well as TBPTT-based data distillation proposed by Wang et al. (2018). Formally, given a learning algorithm Φ, DC solves the following optimization objective:

$$\underset{\mathcal{D}_{\text{syn}}}{\arg\min} \quad \mathop{\mathbb{E}}_{\substack{\theta_0 \sim \mathbf{P}_\theta \\ c \sim \mathcal{C}}} \left[\sum_{t=0}^{T} \mathbf{D} \left(\nabla_\theta \mathcal{L}_{\mathcal{D}^c}(\theta_t), \nabla_\theta \mathcal{L}_{\mathcal{D}^c_{\text{syn}}}(\theta_t) \right) \right] \quad \text{s.t.} \quad \theta_{t+1} \leftarrow \theta_t - \eta \cdot \nabla_\theta \mathcal{L}_{\mathcal{D}_{\text{syn}}}(\theta_t), \tag{7}$$

where T accounts for model similarity T-steps in the future, and $\mathbf{D} : \mathbb{R}^{|\theta|} \times \mathbb{R}^{|\theta|} \mapsto \mathbb{R}$ is a distance metric of choice (typically cosine distance). In addition to assumptions imposed by the meta-model matching framework (Section 2.1), gradient-matching assumes (1) inner-loop optimization of only T steps; (2) local smoothness: two sets of model parameters close to each other (given a distance metric) imply model similarity; and (3) first-order approximation of $\theta_t^{\mathcal{D}}$: instead of exactly computing the training trajectory of optimizing θ_0 on $\mathcal{D}$ (say $\theta_t^{\mathcal{D}}$); perform first-order approximation on the optimization trajectory of θ_0 on the much smaller $\mathcal{D}_{\textsf{syn}}$ (say $\theta_t^{\mathcal{D}_{\textsf{syn}}}$), *i.e.*, approximate $\theta_t^{\mathcal{D}}$ as a single gradient-descent update on $\theta_{t-1}^{\mathcal{D}_{\textsf{syn}}}$ using $\mathcal{D}$ rather than $\theta_{t-1}^{\mathcal{D}}$ (Figure 2).

Subsequently, numerous other approaches have been built atop this framework with subtle variations. DSA (Zhao & Bilen, 2021) improves over DC by performing the same image-augmentations (*e.g.*, crop, rotate, jitter, *etc.*) on both $\mathcal{D}$ and $\mathcal{D}_{\textsf{syn}}$ while optimizing Equation (7). Since these augmentations are universal and are applicable across data distillation frameworks, augmentations performed by DSA have become a common part of all methods proposed henceforth, but we omit them for notational clarity. DCC (Lee et al., 2022b) further modifies the gradient-matching objective to incorporate class contrastive signals inside each gradient-matching step and is shown to improve stability as well as performance. With θ_t evolving similarly as in Equation (7), the modified optimization objective for DCC can be written as:

$$\arg\min_{\mathcal{D}_{\textsf{syn}}} \quad \mathop{\mathbb{E}}_{\theta_0 \sim \mathbf{P}_\theta} \left[\sum_{t=0}^{T} \mathbf{D} \left(\mathop{\mathbb{E}}_{c \in \mathcal{C}} \left[\nabla_\theta \mathcal{L}_{\mathcal{D}^c}(\theta_t) \right], \mathop{\mathbb{E}}_{c \in \mathcal{C}} \left[\nabla_\theta \mathcal{L}_{\mathcal{D}_{\textsf{syn}}^c}(\theta_t) \right] \right) \right] \tag{8}$$

Most recently, Kim et al. (2022) (IDC) extend the gradient matching framework by: (1) multi-formation: to synthesize a higher amount of data within the same memory budget, store the data summary (*e.g.*, images) in a lower resolution to remove spatial redundancies, and upsample (using *e.g.*, bilinear, FSRCNN (Dong et al., 2016)) to the original scale while usage; and (2) matching gradients of the network's training trajectory over the full dataset $\mathcal{D}$ rather than the data summary $\mathcal{D}_{\textsf{syn}}$. To be specific, given a $k\times$ upscaling function $f : \mathbb{R}^{d \times d} \mapsto \mathbb{R}^{kd \times kd}$, the modified optimization objective for IDC can be formalized as:

$$\arg\min_{\mathcal{D}_{\textsf{syn}}} \quad \mathop{\mathbb{E}}_{\substack{\theta_0 \sim \mathbf{P}_\theta \\ c \sim \mathcal{C}}} \left[\sum_{t=0}^{T} \mathbf{D} \left(\nabla_\theta \mathcal{L}_{\mathcal{D}^c}(\theta_t), \nabla_\theta \mathcal{L}_{f(\mathcal{D}_{\textsf{syn}}^c)}(\theta_t) \right) \right] \quad \text{s.t.} \quad \theta_{t+1} \leftarrow \theta_t - \eta \cdot \nabla_\theta \mathcal{L}_{\mathcal{D}}(\theta_t) \tag{9}$$

Kim et al. (2022) further hypothesize that training models on $\mathcal{D}_{\textsf{syn}}$ instead of $\mathcal{D}$ in the inner-loop has two major drawbacks: (1) strong coupling of the inner- and outer-loop resulting in a chicken-egg problem (McLachlan & Krishnan, 2007); and (2) vanishing network gradients due to the small size of $\mathcal{D}_{\textsf{syn}}$, leading to an improper outer-loop optimization for gradient-matching based techniques.

2.3 Data Distillation by Trajectory Matching

Cazenavette et al. (2022) proposed MTT which aims to match the training trajectories of models trained on $\mathcal{D}$ *vs.* $\mathcal{D}_{\textsf{syn}}$. More specifically, let $\{\theta_t^{\mathcal{D}}\}_{t=0}^{T}$ represent the training trajectory of training Φ_θ on $\mathcal{D}$; trajectory matching algorithms aim to solve the following optimization:

$$\begin{aligned} \arg\min_{\mathcal{D}_{\textsf{syn}}, \eta} \quad & \mathop{\mathbb{E}}_{\theta_0 \sim \mathbf{P}_\theta} \left[\sum_{t=0}^{T-M} \frac{\mathbf{D}\left(\theta_{t+M}^{\mathcal{D}}, \theta_{t+N}^{\mathcal{D}_{\textsf{syn}}}\right)}{\mathbf{D}\left(\theta_{t+M}^{\mathcal{D}}, \theta_t^{\mathcal{D}}\right)} \right] \\ \text{s.t.} \quad & \theta_{t+i+1}^{\mathcal{D}_{\textsf{syn}}} \leftarrow \theta_{t+i}^{\mathcal{D}_{\textsf{syn}}} - \eta \cdot \nabla_\theta \mathcal{L}_{\mathcal{D}_{\textsf{syn}}}(\theta_{t+i}^{\mathcal{D}_{\textsf{syn}}}) \quad ; \quad \theta_{t+1}^{\mathcal{D}_{\textsf{syn}}} \leftarrow \theta_t^{\mathcal{D}} - \eta \cdot \nabla_\theta \mathcal{L}_{\mathcal{D}_{\textsf{syn}}}(\theta_t^{\mathcal{D}}), \end{aligned} \tag{10}$$

where $\mathbf{D} : \mathbb{R}^{|\theta|} \times \mathbb{R}^{|\theta|} \mapsto \mathbb{R}$ is a distance metric of choice (typically L2 distance). Such an optimization can intuitively be seen as optimizing for similar quality models trained with N SGD steps on $\mathcal{D}_{\textsf{syn}}$, compared to $M \gg N$ steps on $\mathcal{D}$, thereby invoking long-horizon trajectory matching. Notably, calculating the gradient of Equation (10) *w.r.t.* $\mathcal{D}_{\textsf{syn}}$ encompasses gradient unrolling through N-timesteps, thereby limiting the scalability of MTT. On the other hand, since the trajectory of training Φ_θ on $\mathcal{D}$, *i.e.*, $\{\theta_t^{\mathcal{D}}\}_{t=0}^{T}$ is independent of the optimization of $\mathcal{D}_{\textsf{syn}}$, it can be pre-computed for various $\theta_0 \sim \mathbf{P}_\theta$ initializations and directly substituted. Similar to gradient matching methods (Section 2.2), the trajectory matching framework also optimizes the first-order distance between parameters, thereby inheriting the local smoothness assumption. As a scalable

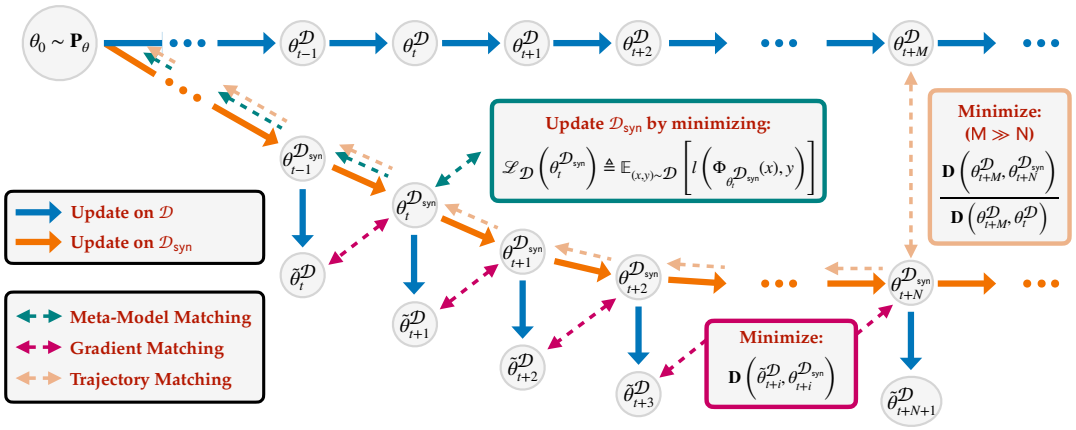

Figure 2: The underlying optimization in various data distillation frameworks.

alternative, Cui et al. (2022b) proposed TESLA, which re-parameterizes the parameter-matching loss of MTT in Equation (10) (specifically when $\mathbf{D}$ is set as the L2 distance), using linear algebraic manipulations to make the bilevel optimization's memory complexity independent of N. Furthermore, TESLA uses learnable soft-labels ($\mathbf{Y}_{\text{syn}}$) during the optimization for an increased compression efficiency.

2.4 Data Distillation by Distribution Matching

Even though the aforementioned gradient-matching or trajectory-matching based data distillation techniques have been empirically shown to synthesize high-quality data summaries, the underlying bilevel optimization, however, is oftentimes an expensive procedure both in terms of computation time and memory. To this end, distribution-matching techniques solve a correlated proxy task which restricts the optimization to a single-level, leading to a much improved scalability. More specifically, instead of matching the quality of models on $\mathcal{D}$ *vs.* $\mathcal{D}_{\text{syn}}$, distribution-matching techniques directly match the distribution of data in $\mathcal{D}$ *vs.* $\mathcal{D}_{\text{syn}}$. The key assumption for this family of methods is that two datasets which are similar according to a particular distribution divergence metric, also lead to similarly trained models.

First proposed by Zhao & Bilen (2023), DM uses (1) numerous parametric encoders to cast high-dimensional data into respective low-dimensional latent spaces; and (2) an approximation of the Maximum Mean Discrepancy to compute the distribution mismatch between $\mathcal{D}$ and $\mathcal{D}_{\text{syn}}$ in each of the latent spaces. More precisely, given a set of k encoders $\mathcal{E} \triangleq \{\psi_i : \mathcal{X} \mapsto \mathcal{X}_i\}_{i=1}^{k}$, the optimization objective can be written as:

$$\arg\min_{\mathcal{D}_{\text{syn}}} \quad \mathop{\mathbb{E}}_{\substack{\psi \sim \mathcal{E} \\ c \sim \mathcal{C}}} \left[\left\| \mathop{\mathbb{E}}_{x \sim \mathcal{D}^c} [\psi(x)] - \mathop{\mathbb{E}}_{x \sim \mathcal{D}^c_{\text{syn}}} [\psi(x)] \right\|^2 \right] \tag{11}$$

DM uses a set of randomly initialized neural networks (with the same architecture) to instantiate $\mathcal{E}$. They observe similar performance when instantiated with more meaningful, task-optimized neural networks, despite it being much less efficient. CAFE (Wang et al., 2022) further refines the distribution-matching idea by: (1) solving a bilevel optimization problem for jointly optimizing a *single* encoder (Φ) and the data summary, rather than using a pre-determined *set* of encoders ($\mathcal{E}$); and (2) assuming a neural network encoder (Φ), match the latent representations obtained at all intermediate layers of the encoder instead of only the last layer. Formally, given a $(L+1)$-layer neural network $\Phi_\theta : \mathcal{X} \mapsto \mathcal{Y}$ where Φ^l_θ represents Φ's output at the l^{th} layer, the optimization problem for CAFE can be specified as:

$$\begin{aligned} &\arg\min_{\mathcal{D}_{\text{syn}}} \quad \mathop{\mathbb{E}}_{c \sim \mathcal{C}} \left[\sum_{l=1}^{L} \left\| \mathop{\mathbb{E}}_{x \sim \mathcal{D}^c} \left[\Phi^l_{\theta_t}(x)\right] - \mathop{\mathbb{E}}_{x \sim \mathcal{D}^c_{\text{syn}}} \left[\Phi^l_{\theta_t}(x)\right] \right\|^2 - \beta \cdot \mathop{\mathbb{E}}_{(x,y) \sim \mathcal{D}^c} \left[\log \hat{p}(y|x, \theta_t)\right] \right] \\ &\text{s.t.} \quad \theta_{t+1} \leftarrow \theta_t - \eta \cdot \nabla_\theta \mathcal{L}_{\mathcal{D}_{\text{syn}}}(\theta_t) \;\; ; \;\; \hat{p}(y|x,\theta) \triangleq \mathop{\text{softmax}}_{y} \left(\left\langle \Phi^L_\theta(x), \mathop{\mathbb{E}}_{x' \sim \mathcal{D}^y_{\text{syn}}} \left[\Phi^L_\theta(x')\right] \right\rangle \right), \end{aligned} \tag{12}$$

where $\hat{p}(\cdot|\cdot, \theta)$ intuitively represents the nearest centroid classifier on $\mathcal{D}_{\text{syn}}$ using the latent representations obtained by last layer of Φ_θ. Analogously, IT-GAN (Zhao & Bilen, 2022) also uses the distribution-matching framework in Equation (11) to generate data that is informative for model training, in contrast to the traditional GAN (Goodfellow et al., 2014) which focuses on generating realistic data.

2.5 Data Distillation by Factorization

All of the aforementioned data distillation frameworks intrinsically maintain the synthesized data summary as a large set of free parameters, which are in turn optimized. Arguably, such a setup prohibits knowledge sharing between synthesized data points (parameters), which might introduce data redundancy. On the other hand, factorization-based data distillation techniques parameterize the data summary using two separate components: (1) bases: a set of mutually independent base vectors; and (2) hallucinators: a mapping from the bases' vector space to the joint data- and label-space. In turn, both the bases and hallucinators are optimized for the task of data distillation.

Formally, let $\mathcal{B} \triangleq \{b_i \in \mathbb{B}\}_{i=1}^{|\mathcal{B}|}$ be the set of bases, and $\mathcal{H} \triangleq \{h_i : \mathbb{B} \mapsto \mathcal{X} \times \mathcal{Y}\}_{i=1}^{|\mathcal{H}|}$ be the set of hallucinators, then the data summary is parameterized as $\mathcal{D}_{\text{syn}} \triangleq \{h(b)\}_{b\sim\mathcal{B},\ h\sim\mathcal{H}}$. Even though such a two-pronged approach seems similar to generative modeling of data, note that unlike classic generative models, (1) the input space consists *only of* a fixed and optimized set of latent codes and isn't meant to take any other inputs; and (2) given a specific $\mathcal{B}$ and $\mathcal{H}$, we can generate at most $|\mathcal{B}| \cdot |\mathcal{H}|$ sized data summaries. Notably, such a hallucinator-bases data parameterization can be optimized using any of the aforementioned data optimization frameworks (Sections 2.1 to 2.4)

This framework was concurrently proposed by Deng & Russakovsky (2022) (we take the liberty to term their unnamed model as "*Lin*-ear *Ba*-ses") and Liu et al. (2022c) (HaBa). LinBa modifies the general hallucinator-bases framework by assuming (1) the bases' vector space ($\mathbb{B}$) to be the same as the task input space ($\mathcal{X}$); and (2) the hallucinator to be linear and additionally conditioned on a given predictand. More specifically, the data parameterization can be formalized as follows:

$$\begin{aligned} \mathcal{D}_{\text{syn}} &\triangleq \left\{ \ (y\ \mathbf{H}^T\mathbf{B},\ y)\ \right\}_{\substack{y\sim\mathcal{C} \\ \mathbf{H}\sim\mathcal{H}}} \\ \text{s.t.} \quad \mathbf{B} \in \mathbb{R}^{|\mathbf{B}|\times\dim(\mathcal{X})} &\triangleq [b_i \in \mathcal{X}]_{i=1}^{|\mathbf{B}|} \quad ; \quad \mathcal{H} \triangleq \left\{\mathbf{H}_i \in \mathbb{R}^{|\mathbf{B}|\times|\mathcal{C}|}\right\}_{i=1}^{|\mathcal{H}|}, \end{aligned} \tag{13}$$

where for the sake of notational simplicity, we assume $y \in \mathbb{R}^{|\mathcal{C}|}$ represents the one-hot vector of the label for which we want to generate data, and the maximum amount of data that can be synthesized $n \leq |\mathcal{C}| \cdot |\mathcal{H}|$. Since the data generation (Equation (13)) is an end-to-end differentiable procedure, both $\mathbf{B}$ and $\mathcal{H}$ are jointly optimized using the TBPTT framework discussed in Section 2.1, albeit with some crucial modifications for vastly improved performance: (1) using momentum-based optimizers instead of vanilla SGD in the inner-loop; and (2) longer unrolling (≥ 100 steps) of the inner-loop during TBPTT. Liu et al. (2022c) (HaBa) relax the linear and predictand-conditional hallucinator assumption of LinBa, equating to the following data parameterization:

$$\mathcal{D}_{\text{syn}} \triangleq \left\{ \ (h(b),\ y)\ \right\}_{\substack{b,y\sim\mathcal{B} \\ h\sim\mathcal{H}}} \quad \text{s.t.} \quad \mathcal{B} \triangleq \left\{ \ (b_i \in \mathcal{X}, y_i \in \mathcal{Y})\ \right\}_{i=1}^{|\mathcal{B}|} \quad ; \quad \mathcal{H} \triangleq \{h_{\theta_i} : \mathcal{X} \mapsto \mathcal{X}\}_{i=1}^{|\mathcal{H}|}, \tag{14}$$

where $\mathcal{B}$ and $\mathcal{H}$ are optimized using the trajectory matching framework (Section 2.3) with an additional contrastive constraint to promote diversity in $\mathcal{D}_{\text{syn}}$ (cf. Liu et al. (2022c), Equation (6)). Following this setup, HaBa can generate at most $|\mathcal{B}| \cdot |\mathcal{H}|$ sized data summaries. Furthermore, one striking difference between HaBa (Equation (14)) and LinBa (Equation (13)) is that to generate each data point, LinBa uses a linear combination of *all* the bases, whereas HaBa generates a data point using a *single* base vector.

Lee et al. (2022a) (KFS) further build atop this framework by maintaining a different bases' vector space $\mathbb{B}$ from the data domain $\mathcal{X}$, such that $\dim(\mathbb{B}) < \dim(\mathcal{X})$. This parameterization allows KFS to store an even larger number of images, with a comparable storage budget to other methods. Formally, the data

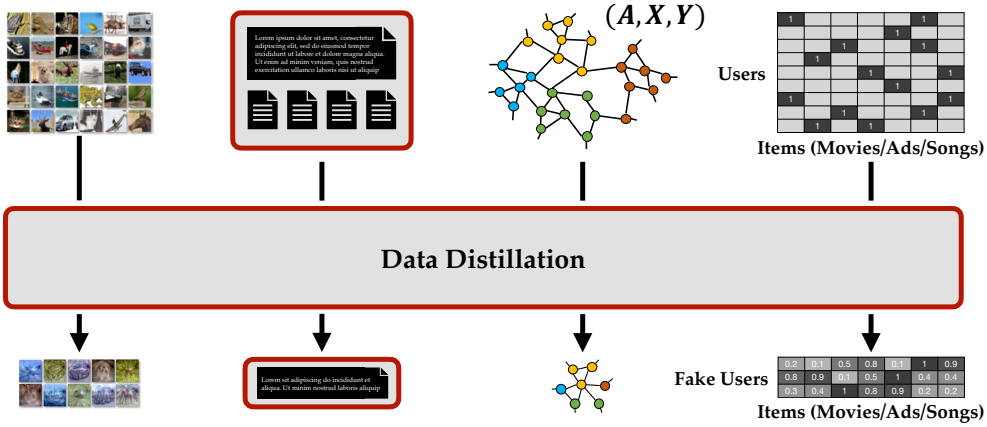

Figure 3: Overview of distilling data for a few commonly observed data modalities.

parameterization for KFS can be specified as:

$$\mathcal{D}_{\text{syn}} \triangleq \bigcup_{c \in \mathcal{C}} \{\ (h(b),\ c)\ \}_{\substack{b \sim \mathcal{B}_c \\ h \sim \mathcal{H}}}$$
$$\text{s.t.} \quad \mathcal{B} \triangleq \bigcup_{c \in \mathcal{C}} \mathcal{B}_c \quad ; \quad \mathcal{B}_c \triangleq \{b_i^c \in \mathbb{B}\}_{i=1}^{B} \quad ; \quad \mathcal{H} \triangleq \{h_{\theta_i} : \mathbb{B} \mapsto \mathcal{X}\}_{i=1}^{|\mathcal{H}|}, \tag{15}$$

where KFS stores B bases per class, equivalent to a total of $n = |\mathcal{C}| \cdot B \cdot |\mathcal{H}|$ sized data summaries. Following this data parameterization, $\mathcal{B}$ and $\mathcal{H}$ are optimized using the distribution matching framework for data distillation (Equation (11)) to ensure fast, single-level optimization.

Data Distillation *vs.* Data Compression. We highlight that it is non-trivial to ensure a fair comparison between data distillation techniques that (1) are "non-factorized", *i.e.*, maintain each synthesized data point as a set of free-parameters (Sections 2.1 to 2.4); and (2) use factorized approaches discussed in this section to efficiently organize the data summary. If we use the size of the data summary (n) as the efficiency metric, factorized approaches are adversely affected as they need a much smaller storage budget to synthesize the same-sized data summaries. On the other hand, if we use "end-to-end bytes of storage" as the efficiency metric, non-factorized approaches are adversely affected as they perform no kind of data compression, but focus solely on better understanding the model-to-data relationship through the lens of optimization. For a better intuition, one can apply posthoc lossless compression (*e.g.*, Huffman coding) on data synthesized by non-factorized data distillation approaches to fit more images in the same storage budget (Schirrmeister et al., 2022). Such techniques unintentionally deviate from the original intent of data distillation, and progress more toward better data compression techniques. As a potential solution, we encourage the community to consider reporting results for both scenarios: a fixed data summary size n, as well as fixed bytes-of-storage. Nonetheless, for the ease of empirical comparison amongst the discussed data distillation techniques, we provide a collated set of results over four image-classification datasets in Table 1.

3 Data Modalities

Having learned about different kinds of optimization frameworks for data distillation, we now discuss an orthogonal (and important) aspect of data distillation – *what kinds of data can data distillation techniques summarize?* From continuous-valued images to heterogeneous, discrete, and semi-structured graphs, the underlying data for each unique application of machine learning has its own modality, structure, and set of assumptions. While the earliest data distillation techniques were designed to summarize images for classification, recent steps have been taken to expand the horizon of data distillation into numerous other scenarios. In what follows, we categorize existing data distillation techniques as per their intended data modality, while also discussing their unique challenges.

Table 1: Comparison of data distillation methods. Each method (1) synthesizes the data summary on the train-set; (2) unless mentioned, trains a 128-width ConvNet (Gidaris & Komodakis, 2018) on the data summary; and (3) evaluates it on the test-set. Confidence intervals are obtained by training at least 5 networks on the data summary. LinBa (No Fact.) represents LinBa with the no factorization. Methods evaluated using KRR are marked as (∞-Conv) or (∞-FC). The equivalent storage-in-bytes is used for factorization-based techniques instead of IPC. The best method in their category is **emboldened**, the best-overall non-factorized method evaluated on ConvNet is **colored orange**, and the best-overall factorized method is **colored blue**.

	Dataset	**MNIST**			**CIFAR-10**			**CIFAR-100**			**Tiny ImageNet**		
	Imgs/Class (IPC)	1	10	50	1	10	50	1	10	50	1	10	50
Baselines	Random	64.9 ±3.5	**95.1 ±0.9**	**97.9 ±0.2**	14.4 ±2.0	26.0 ±1.2	**43.4 ±1.0**	4.2 ±0.3	14.6 ±0.5	30.0 ±0.4	**1.5 ±0.1**	**6.0 ±0.8**	**16.8 ±1.8**
	Herding[1]	**89.2 ±1.6**	93.7 ±0.3	94.9 ±0.2	**21.5 ±1.2**	**31.6 ±0.7**	40.4 ±0.6	8.4 ±0.3	17.3 ±0.5	33.7 ±0.5	-	-	-
	Forgetting[2]	35.5 ±5.6	68.1 ±3.3	88.2 ±1.2	13.5 ±1.2	23.3 ±1.0	23.3 ±1.1	4.5 ±0.2	15.1 ±0.3	30.5 ±0.3	-	-	-
Meta-model Matching	DD[3]	-	79.5 ±8.1	-	-	36.8 ±1.2	-	-	-	-	-	-	-
	LinBa (No Fact.)[15]	**95.2 ±0.3**	**98.8 ±0.1**	**99.2 ±0.1**	49.1 ±0.6	62.4 ±0.4	70.5 ±0.4	21.3 ±0.6	34.7 ±0.5	-	-	-	-
	KIP (ConvNet)[4]	90.1 ±0.1	97.5 ±0.0	98.3 ±0.1	49.9 ±0.2	62.7 ±0.3	68.6 ±0.2	15.7 ±0.2	28.3 ±0.1	-	-	-	-
	RFAD (ConvNet)[5]	94.4 ±1.5	98.5 ±0.1	98.8 ±0.1	**53.6 ±1.2**	**66.3 ±0.5**	71.1 ±0.4	26.3 ±1.1	33.0 ±0.3	-	-	-	-
	FRePO (ConvNet)[6]	93.0 ±0.4	98.6 ±0.1	**99.2 ±0.1**	46.8 ±0.7	65.5 ±0.6	**71.7 ±0.2**	**28.7 ±0.1**	**42.5 ±0.2**	**44.3 ±0.2**	**15.4 ±0.3**	**25.4 ±0.2**	-
	KIP (∞-FC)[7]	85.5 ±0.1	97.2 ±0.2	98.4 ±0.1	40.5 ±0.4	53.1 ±0.5	58.6 ±0.4	-	-	-	-	-	-
	KIP (∞-Conv)[4]	**97.3 ±0.1**	**99.1 ±0.1**	**99.5 ±0.1**	**64.7 ±0.2**	**75.6 ±0.2**	**80.6 ±0.1**	34.9 ±0.1	**49.5 ±0.3**	-	-	-	-
	RFAD (∞-Conv)[5]	97.2 ±0.2	99.1 ±0.0	99.1 ±0.0	61.4 ±0.8	73.7 ±0.2	76.6 ±0.3	**44.1 ±0.1**	46.8 ±0.2	-	-	-	-
	FRePO (∞-Conv)[6]	92.6 ±0.4	98.6 ±0.1	99.2 ±0.1	47.9 ±0.6	68.0 ±0.2	74.4 ±0.1	32.3 ±0.1	44.9 ±0.2	43.0 ±0.3	**19.1 ±0.3**	**26.5 ±0.1**	-
Gradient Matching	DC[8]	**91.7 ±0.5**	97.4 ±0.2	98.2 ±0.2	28.3 ±0.5	44.9 ±0.5	53.9 ±0.5	12.8 ±0.3	25.2 ±0.3	30.5 ±0.3	4.6 ±0.6	11.2 ±1.6	10.9 ±0.7
	DSA[9]	88.7 ±0.6	**97.8 ±0.1**	**99.2 ±0.1**	28.8 ±0.7	52.1 ±0.5	60.6 ±0.5	13.9 ±0.3	32.3 ±0.3	**42.8 ±0.4**	**6.6 ±0.2**	**14.4 ±2.0**	**22.6 ±2.6**
	DCC[10]	-	-	-	**34.0 ±0.7**	**54.5 ±0.5**	**64.2 ±0.4**	**14.6 ±0.3**	**33.5 ±0.3**	39.3 ±0.4	-	-	-
Distr. Matching	DM[11]	89.7 ±0.6	**97.5 ±0.1**	98.6 ±0.1	26.0 ±0.8	48.9 ±0.6	**63.0 ±0.4**	11.4 ±0.3	29.7 ±0.3	**43.6 ±0.4**	**3.9 ±0.2**	**12.9 ±0.4**	**24.1 ±0.3**
	CAFE[12]	**90.8 ±0.5**	**97.5 ±0.1**	**98.9 ±0.2**	**31.6 ±0.8**	**50.9 ±0.5**	62.3 ±0.4	**14.0 ±0.3**	**31.5 ±0.2**	42.9 ±0.2	-	-	-
Traj. Matching	MTT[13]	-	-	-	46.3 ±0.8	65.3 ±0.7	71.6 ±0.2	24.3 ±0.3	40.1 ±0.4	47.7 ±0.2	**8.8 ±0.3**	**23.2 ±0.2**	**28.0 ±0.3**
	TESLA[14]	-	-	-	**48.5 ±0.8**	**66.4 ±0.8**	**72.6 ±0.7**	**24.8 ±0.4**	**41.7 ±0.3**	**47.9 ±0.3**	-	-	-
Factorization	IDC[15]	-	-	-	50.0 ±0.4	67.5 ±0.5	74.5 ±0.1	-	44.8 ±0.2	-	-	-	-
	LinBa[16]	**98.7 ±0.7**	**99.3 ±0.5**	**99.4 ±0.4**	**66.4 ±0.4**	71.2 ±0.4	73.6 ±0.5	34.0 ±0.4	42.9 ±0.7	-	16.0 ±0.7	-	-
	HaBa[17]	-	-	-	48.3 ±0.8	69.9 ±0.4	74.0 ±0.2	33.4 ±0.4	40.2 ±0.2	**47.0 ±0.2**	-	-	-
	KFS[18]	-	-	-	59.8 ±0.5	**72.0 ±0.3**	**75.0 ±0.2**	**40.0 ±0.5**	**50.6 ±0.2**	-	**22.7 ±0.2**	**27.8 ±0.2**	-
Full Dataset			99.6 ±0.1			84.8 ±0.1			56.2 ±0.3			37.6 ±0.4	

[1] (Welling, 2009), [2] (Toneva et al., 2019), [3] (Wang et al., 2018), [4] (Nguyen et al., 2021b), [5] (Loo et al., 2022)
[6] (Zhou et al., 2022b), [7] (Nguyen et al., 2021a), [8] (Zhao et al., 2021), [9] (Zhao & Bilen, 2021), [10] (Lee et al., 2022b)
[11] (Zhao & Bilen, 2023), [12] (Wang et al., 2022), [13] (Cazenavette et al., 2022), [14] (Cui et al., 2022b)
[15] (Kim et al., 2022), [16] (Deng & Russakovsky, 2022), [17] (Liu et al., 2022c), [18] (Lee et al., 2022a)

Images. A large-portion of existing data distillation techniques are designed for image classification data (Cazenavette et al., 2022; Deng & Russakovsky, 2022; Kim et al., 2022; Lee et al., 2022a;b; Liu et al., 2022c; Loo et al., 2022; Nguyen et al., 2021a;b; Wang et al., 2022; 2018; Zhao & Bilen, 2021; 2022; 2023; Zhao et al., 2021; Zhou et al., 2022b) simply because images have a real-valued, continuous data-domain ($\mathcal{X} \equiv \mathbb{R}^{d \times d}$). This allows SGD-based optimization directly on the data, which is treated as a set of free parameters. Intuitively, incrementally changing each pixel value can be treated as slight perturbations in the color space, and hence given a suitable data distillation loss, can be naïvely optimized using SGD.

Text. Textual data is available in large amounts from sources like websites, news articles, academic manuscripts, *etc.*, and is also readily accessible with datasets like the common crawl[1] which sizes up to almost 541TB. Furthermore, with the advent of large language models (LLM) (Brown et al., 2020; Devlin et al., 2019; Thoppilan et al., 2022), training such models from scratch on large datasets has become an increasingly expensive procedure. Despite recent efforts in democratizing LLM training (Geiping & Goldstein, 2022; Scao et al., 2022; Wolf et al., 2020), effectively distilling large-scale textual data as a solution is yet to be explored. The key bottlenecks for distilling textual data are: (1) the inherently discrete nature of data, where a token should belong in a limited vocabulary of words; (2) the presence of a rich underlying structure, *i.e.*, sentences of words (text) obey fixed patterns according to a grammar; and (3) richness of context, *i.e.*, a given piece of text could have wildly different semantic interpretations under different contexts.

Sucholutsky & Schonlau (2021) take a latent-embedding approach to textual data distillation. On a high level, to circumvent the discreteness of the optimization, the authors perform distillation in a continuous embedding space. More specifically, assuming access to a latent space specified by a *fixed* text-encoder, the authors learn continuous *representations* of each word in the distilled text and optimize it using the TBPTT data-distillation framework proposed by Wang et al. (2018) (Equation (4)). Finally, the distilled text representations are decoded by following a simple nearest-neighbor protocol.

Graphs. A wide variety of data and applications can inherently be modeled as graphs, *e.g.*, user-item interactions (Mittal et al., 2021; Sachdeva & McAuley, 2020; Wu et al., 2020), social networks (Fan et al., 2019), autonomous driving (Casas et al., 2020; Sachdeva et al., 2022b), *etc.* Taking the example of social networks, these user-user graphs in the modern-era easily scale up to the billion-scale (Chen et al., 2021), calling for principled scaling solutions. Graph distillation could trivially solve a majority of the scale challenges, but synthesizing tiny, high-fidelity graphs has the following hurdles: (1) nodes in a graph can be highly abstract, *e.g.*, users, products, text articles, *etc.* some of which could be discrete, heterogeneous, or even simply numerical IDs; (2) graphs follow a variety of intrinsic patterns (*e.g.*, spatial (Kipf & Welling, 2017)) which need to be retained in the distilled graphs; and (3) quadratic size of the adjacency matrix could be computationally prohibitive even for moderate-sized graphs.

Jin et al. (2022b) propose GCOND which distills graphs in the inductive node-classification setting, specified by its node-feature matrix $\mathbf{X}$, adjacency matrix $\mathbf{A}$, and node-target matrix $\mathbf{Y}$. GCOND distills the given graph by learning a synthetic node-feature matrix $\mathbf{X}_{\mathsf{syn}}$, and using $\mathbf{X}_{\mathsf{syn}}$ to generate $\mathbf{A}_{\mathsf{syn}} \triangleq f_\theta(\mathbf{X}_{\mathsf{syn}})$ which can be realized, *e.g.*, through a parametric similarity function $\mathrm{sim}_\theta(\cdot, \cdot)$ between the features of two nodes, *i.e.*, $\mathbf{A}^{i,j}_{\mathsf{syn}} \triangleq \sigma(\mathrm{sim}_\theta(\mathbf{X}^i_{\mathsf{syn}}, \mathbf{X}^j_{\mathsf{syn}}))$, where $\sigma(\cdot)$ is the sigmoid function. Finally, both $\mathbf{X}_{\mathsf{syn}}$ and θ are optimized using the gradient-matching framework proposed by Zhao et al. (2021) (Equation (7)). Another work (Liu et al., 2022a) (GCDM) shares the same framework as GCOND but instead uses the distribution matching framework proposed by Zhao & Bilen (2023) (Equation (11)) to optimize $\mathbf{X}_{\mathsf{syn}}$ and θ. Extending to a graph-classification setting, Jin et al. (2022a) further propose DOSCOND with two major changes compared to GCOND: (1) instead of parameterizing the adjacency matrix using a similarity function on $\mathbf{X}_{\mathsf{syn}}$, they maintain a free-parameter matrix Ω with the same size as the adjacency matrix, and sample each $\mathbf{A}^{i,j}_{\mathsf{syn}}$ entry through an independent Bernoulli draw on $\Omega^{i,j}$ as the prior using the reparameterization trick (Maddison et al., 2017). Such a procedure ensures differentiability as well as discrete matrix synthesis; and (2) $\mathbf{X}_{\mathsf{syn}}$ and Ω are still optimized using the gradient-matching framework (Equation (7)), albeit with only a single-step, *i.e.*, $T = 1$ for improved scalability and without empirically observing a loss in performance.

[1] https://commoncrawl.org/the-data/

Recommender Systems. The amount of online user-feedback data available for training recommender systems is rapidly increasing (Wu et al., 2022). Furthermore, typical user-facing recommender systems need to be periodically re-trained (Naumov et al., 2019), which adds to requirements for smarter data summarization solutions (see Sachdeva et al. (2022c) for background on sampling recommender systems data). However, distilling recommender systems data has the following challenges: (1) the data is available in the form of abstract and discrete `(userID, itemID, relevance)` tuples, which departs from the typical `(features, label)` setup; (2) the distribution of both user- and item-popularity follows a strong power-law which leads to data scarcity and unstable optimization; and (3) the data inherits a variety of inherent structures, *e.g.*, sequential patterns (Kang & McAuley, 2018; Sachdeva et al., 2019), user-item graph patterns (Wu et al., 2019), item-item co-occurrence patterns (Steck, 2019), missing-not-at-randomness (Sachdeva et al., 2020; Schnabel et al., 2016), *etc.*

Sachdeva et al. (2022a) propose DISTILL-CF which distills implicit-feedback recommender systems data, *i.e.*, when the observed user-item relevance is binary (*e.g.*, click or no-click). Such data can be visualized as a binary user-item matrix $\mathbf{R}$ where each row represents a single user, and each column represents an item. On a high-level, DISTILL-CF synthesizes fake users along with their item-consumption histories, visualized as a synthetic user-item matrix $\mathbf{R}_{\text{syn}}$. Notably, to preserve semantic meaning, the item-space in $\mathbf{R}_{\text{syn}}$ is the same as in $\mathbf{R}$. To alleviate the data discreteness problem, DISTILL-CF maintains a sampling-prior matrix Ω which has the same size as $\mathbf{R}_{\text{syn}}$, and can in-turn be used to generate $\mathbf{R}_{\text{syn}}$ using multi-step Gumbel sampling with replacement (Jang et al., 2017) for each user's prior in Ω (equivalent to each row). Such a formulation automatically also circumvents the dynamic user- and item-popularity artifact in recommender systems data, which can analogously be controlled by the row- and column-wise entropy of Ω. Finally, Ω is optimized using the meta-model matching framework proposed by Nguyen et al. (2021a). Notably, Sachdeva et al. (2022a) also propose infinite-width autoencoders which suit the task of item recommendation while also leading to closed-form computation of the inner-loop in the meta-model matching framework (Equation (5)).

4 Applications

While the data distillation task was originally designed to accelerate model training, there are numerous other applications of a high-fidelity data summary. Below we briefly discuss a few such promising applications, along with providing pointers to existing works.

Differential Privacy. Data distillation was recently shown to be a promising solution for differential privacy as defined by Dwork (2008). Dong et al. (2022) show that data distillation techniques can perform better than existing state-of-the-art differentially-private data generators (Cao et al., 2021; Harder et al., 2021) on both performance and privacy grounds. Notably, the privacy benefits of data distillation techniques are virtually *free*, as none of these methods were optimized for generating differentially-private data. Chen et al. (2022) further modify the gradient matching framework (Equation (7)) by clipping and adding white noise to the gradients obtained on the original dataset while optimization. Such a routine was shown to have better sample utility, while also satisfying strict differential privacy guarantees. From a completely application perspective, data distillation has been used to effectively distill sensitive medical data as well (Li et al., 2020a; 2022).

Neural Architecture Search (NAS). Automatic searching of neural-network architectures can alleviate the manual effort, as well as lead to better models (see Elsken et al. (2019) for a detailed review). Analogous to using model extrapolation, *i.e.*, extrapolating the performance of an under-trained model architecture on the full dataset; data extrapolation, on the other hand, aims to train models on a small, high-fidelity data sample till convergence. Numerous data distillation techniques (Such et al., 2020; Zhao et al., 2021) show promise on small NAS test-beds by employing the data extrapolation framework. However, Cui et al. (2022a) show that data distillation *does not* perform well when evaluating diverse architectures on bigger NAS test-beds, calling for better rank-preserving data distillation techniques.

Continual Learning. Never-ending learning (see Parisi et al. (2019) for a detailed review) has been frequently associated with catastrophic forgetting (French, 1999), *i.e.*, patterns extracted from old data/tasks are easily forgotten when patterns from new data/tasks are learned. Data distillation has been shown as

an effective solution to alleviate catastrophic forgetting, by simply using the distilled data summary in a replay buffer that is continually updated and used in subsequent data/task training (Rosasco et al., 2021; Sangermano et al., 2022; Wiewel & Yang, 2021). Deng & Russakovsky (2022) show further evidence of a simple *compress-then-recall* strategy outperforming existing state-of-the-art continual learning approaches. Notably, *only* the data summary is stored for each task, and a new model is trained (from scratch) using all previous data summaries, for each new incoming task.

Federated Learning. Federated or collaborative learning (see Li et al. (2020b) for a detailed survey) involves training a learning algorithm in a decentralized fashion. A standard approach to federated learning is to synchronize local parameter updates to a central server, instead of synchronizing the raw data itself (Konečnỳ et al., 2016). Data distillation, on the other hand, alleviates the need to synchronize large parametric models across clients and servers, by synchronizing tiny synthesized data summaries to the central server instead. Subsequently, the entire training happens only on the central server. Such data distillation-based federated learning methods (Goetz & Tewari, 2020; Hu et al., 2022; Liu et al., 2022b; Song et al., 2022; Xiong et al., 2022; Zhou et al., 2020) are shown to perform better than model-synchronization based federated learning approaches, while also requiring multiple orders lesser client-server communication.

5 Challenges & Future Directions

Despite achieving remarkable progress in data-efficient learning, there are numerous framework-based, theoretical, and application-based directions yet to be explored in data distillation. In what follows, we highlight and discuss such directions for the community to further explore, based either on early evidence or our intuition.

New data modalities & settings. Extending on the discussion in Section 3, existing data distillation techniques have largely been restricted to image-classification settings, due to the easy availability of datasets, and amenable data-optimization. However, taking a step back to the broad field of computer vision (see Shapiro et al. (2001) for a thorough background), there are numerous equally important tasks that can benefit from a high-quality data summary. For example, increasing the sample efficiency of training image-generation models is both highly important due to their massive size and popularity (Ramesh et al., 2022; Rombach et al., 2022), and is also highly non-trivial to fit into the existing data distillation framework. Similarly, a variety of important machine learning applications don't enjoy a continuous data domain like images, making it hard for existing data distillation techniques to scale and work as expected. In addition to recent efforts on distilling discrete data like graphs (Jin et al., 2022a;b) and recommender systems (Sachdeva et al., 2022a), developing a unified, principled data distillation framework for inherently sparse and discrete data will be useful for a variety of research communities (*e.g.*, text, tabular-data, extreme classification, *etc.*).

Better scaling. Existing data distillation techniques validate their prowess *only* in the super low-data regime (typically $1-50$ data points per class). However, Cui et al. (2022a) show that as we keep scaling the size of the data summary (larger distilled data), most distillation methods collapse to the random-sampling baseline. While convergent behavior is expected, the distillation performance collapses much more rapidly with larger data summaries. Analogously, for data distillation to practically replace full-data training, deeper investigations of the causes and potential fixes of such scaling artifacts are highly necessary.

Improved optimization. A unifying thread across data distillation techniques is an underlying bilevel optimization, which is provably NP-hard even in the linear inner-optimization case (Vicente et al., 1994). Notably, bilevel optimization has been successfully applied in a variety of other applications like meta-learning (Finn et al., 2017; Li et al., 2017), hyper-parameter optimization (Lorraine et al., 2020; Maclaurin et al., 2015), neural architecture search (Liu et al., 2019), coreset construction (Borsos et al., 2020; Zhou et al., 2022a), *etc.* Despite its success, many theoretical underpinnings are yet to be explored, *e.g.*, the effect of commonly-used singleton solution assumption (Franceschi et al., 2018), the effect of over-parameterization on bilevel optimization (Vicol et al., 2022), connections to statistical influence functions (Bae et al., 2022), the bias-variance tradeoff (Vicol et al., 2021), *etc.* Clearly, an overall better understanding of bilevel optimization will directly enable the development of better data distillation techniques.

References

Sunil Arya, David M Mount, Nathan S Netanyahu, Ruth Silverman, and Angela Y Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM (JACM)*, 45(6):891–923, 1998.

Olivier Bachem, Mario Lucic, and Andreas Krause. Practical coreset constructions for machine learning. *arXiv preprint arXiv:1703.06476*, 2017.

Juhan Bae, Nathan Ng, Alston Lo, Marzyeh Ghassemi, and Roger Grosse. If influence functions are the answer, then what is the question? *arXiv preprint arXiv:2209.05364*, 2022.

Ondrej Bohdal, Yongxin Yang, and Timothy Hospedales. Flexible dataset distillation: Learn labels instead of images. *arXiv preprint arXiv:2006.08572*, 2020.

Zalán Borsos, Mojmir Mutny, and Andreas Krause. Coresets via bilevel optimization for continual learning and streaming. *Advances in Neural Information Processing Systems*, 33:14879–14890, 2020.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Tianshi Cao, Alex Bie, Arash Vahdat, Sanja Fidler, and Karsten Kreis. Don't generate me: Training differentially private generative models with sinkhorn divergence. *Advances in Neural Information Processing Systems*, 2021.

Sergio Casas, Cole Gulino, Renjie Liao, and Raquel Urtasun. Spagnn: Spatially-aware graph neural networks for relational behavior forecasting from sensor data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9491–9497. IEEE, 2020.

George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4750–4759, 2022.

Dingfan Chen, Raouf Kerkouche, and Mario Fritz. Private set generation with discriminative information. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Qi Chen, Bing Zhao, Haidong Wang, Mingqin Li, Chuanjie Liu, Zengzhong Li, Mao Yang, and Jingdong Wang. Spann: Highly-efficient billion-scale approximate nearest neighborhood search. *Advances in Neural Information Processing Systems*, 34:5199–5212, 2021.

Justin Cui, Ruochen Wang, Si Si, and Cho-Jui Hsieh. DC-BENCH: Dataset condensation benchmark. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022a.

Justin Cui, Ruochen Wang, Si Si, and Cho-Jui Hsieh. Scaling up dataset distillation to imagenet-1k with constant memory. *arXiv preprint arXiv:2211.10586*, 2022b.

Zhiwei Deng and Olga Russakovsky. Remember the past: Distilling datasets into addressable memories for neural networks. In *Advances in Neural Information Processing Systems*, 2022.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In *European conference on computer vision*, pp. 391–407. Springer, 2016.

Tian Dong, Bo Zhao, and Lingjuan Lyu. Privacy for free: How does dataset condensation help privacy? In *Proceedings of the 39th International Conference on Machine Learning*. PMLR, 2022.

Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pp. 1–19. Springer, 2008.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pp. 417–426, 2019.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning*, pp. 1568–1577. PMLR, 2018.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

Jonas Geiping and Tom Goldstein. Cramming: Training a language model on a single gpu in one day, 2022.

Behrooz Ghorbani, Orhan Firat, Markus Freitag, Ankur Bapna, Maxim Krikun, Xavier Garcia, Ciprian Chelba, and Colin Cherry. Scaling laws for neural machine translation. *arXiv preprint arXiv:2109.07740*, 2021.

Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4367–4375, 2018.

Jack Goetz and Ambuj Tewari. Federated learning via synthetic data. *arXiv preprint arXiv:2008.04489*, 2020.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Udit Gupta, Young Geun Kim, Sylvia Lee, Jordan Tse, Hsien-Hsin S Lee, Gu-Yeon Wei, David Brooks, and Carole-Jean Wu. Chasing carbon: The elusive environmental footprint of computing. *IEEE Micro*, 42(4):37–47, 2022.

Frederik Harder, Kamil Adamczewski, and Mijung Park. Dp-merf: Differentially private mean embeddings with randomfeatures for practical privacy-preserving data generation. In *International conference on artificial intelligence and statistics*, pp. 1819–1827. PMLR, 2021.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Shengyuan Hu, Jack Goetz, Kshitiz Malik, Hongyuan Zhan, Zhe Liu, and Yue Liu. Fedsynth: Gradient compression via synthetic data in federated learning. *arXiv preprint arXiv:2204.01273*, 2022.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=rkE3y85ee.

Wei Jin, Xianfeng Tang, Haoming Jiang, Zheng Li, Danqing Zhang, Jiliang Tang, and Bing Yin. Condensing graphs via one-step gradient matching. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 720–730, 2022a.

Wei Jin, Lingxiao Zhao, Shichang Zhang, Yozen Liu, Jiliang Tang, and Neil Shah. Graph condensation for graph neural networks. In *International Conference on Learning Representations*, 2022b.

Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pp. 197–206. IEEE, 2018.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Jang-Hyun Kim, Jinuk Kim, Seong Joon Oh, Sangdoo Yun, Hwanjun Song, Joonhyun Jeong, Jung-Woo Ha, and Hyun Oh Song. Dataset condensation via efficient synthetic-data parameterization. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.

Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17, 2017.

Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

Hae Beom Lee, Dong Bok Lee, and Sung Ju Hwang. Dataset condensation with latent space knowledge factorization and sharing. *arXiv preprint arXiv:2208.10494*, 2022a.

Jaehoon Lee, Samuel Schoenholz, Jeffrey Pennington, Ben Adlam, Lechao Xiao, Roman Novak, and Jascha Sohl-Dickstein. Finite versus infinite neural networks: an empirical study. *Advances in Neural Information Processing Systems*, 33:15156–15172, 2020.

Saehyung Lee, Sanghyuk Chun, Sangwon Jung, Sangdoo Yun, and Sungroh Yoon. Dataset condensation with contrastive signals. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 12352–12364, 2022b.

Guang Li, Ren Togo, Takahiro Ogawa, and Miki Haseyama. Soft-label anonymous gastric x-ray image distillation. In *2020 IEEE International Conference on Image Processing (ICIP)*, pp. 305–309. IEEE, 2020a.

Guang Li, Ren Togo, Takahiro Ogawa, and Miki Haseyama. Compressed gastric image generation based on soft-label dataset distillation for medical data sharing. *Computer Methods and Programs in Biomedicine*, pp. 107189, 2022.

Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020b.

Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1eYHoC5FX.

Mengyang Liu, Shanchuan Li, Xinshi Chen, and Le Song. Graph condensation via receptive field distribution matching. *arXiv preprint arXiv:2206.13697*, 2022a.

Ping Liu, Xin Yu, and Joey Tianyi Zhou. Meta knowledge condensation for federated learning. *arXiv preprint arXiv:2209.14851*, 2022b.

Songhua Liu, Kai Wang, Xingyi Yang, Jingwen Ye, and Xinchao Wang. Dataset distillation via factorization. *NeurIPS*, 2022c.

Noel Loo, Ramin Hasani, Alexander Amini, and Daniela Rus. Efficient dataset distillation using random feature approximation. In *Advances in Neural Information Processing Systems*, 2022.

Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *International Conference on Artificial Intelligence and Statistics*, pp. 1540–1552. PMLR, 2020.

Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *International conference on machine learning*, pp. 2113–2122. PMLR, 2015.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017.

Geoffrey J McLachlan and Thriyambakam Krishnan. *The EM algorithm and extensions*. John Wiley & Sons, 2007.

Luke Metz, Niru Maheswaranathan, Jeremy Nixon, Daniel Freeman, and Jascha Sohl-Dickstein. Understanding and correcting pathologies in the training of learned optimizers. In *International Conference on Machine Learning*, pp. 4556–4565. PMLR, 2019.

Anshul Mittal, Noveen Sachdeva, Sheshansh Agrawal, Sumeet Agarwal, Purushottam Kar, and Manik Varma. Eclare: Extreme classification with label graph correlations. In *Proceedings of the Web Conference 2021*, WWW '21, 2021.

Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G. Azzolini, Dmytro Dzhulgakov, Andrey Mallevich, Ilia Cherniavskii, Yinghai Lu, Raghuraman Krishnamoorthi, Ansha Yu, Volodymyr Kondratenko, Stephanie Pereira, Xianjie Chen, Wenlin Chen, Vijay Rao, Bill Jia, Liang Xiong, and Misha Smelyanskiy. Deep learning recommendation model for personalization and recommendation systems. *CoRR*, abs/1906.00091, 2019.

Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. Dataset meta-learning from kernel ridge-regression. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=l-PrrQrK0QR.

Timothy Nguyen, Roman Novak, Lechao Xiao, and Jaehoon Lee. Dataset distillation with infinitely wide convolutional networks. *Advances in Neural Information Processing Systems*, 34, 2021b.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

Lorien Y Pratt. Discriminability-based transfer between neural networks. *Advances in neural information processing systems*, 5, 1992.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Andrea Rosasco, Antonio Carta, Andrea Cossu, Vincenzo Lomonaco, and Davide Bacciu. Distilled replay: Overcoming forgetting through synthetic samples. *arXiv preprint arXiv:2103.15851*, 2021.

Noveen Sachdeva and Julian McAuley. *How Useful Are Reviews for Recommendation? A Critical Review and Potential Improvements*, pp. 1845–1848. SIGIR '20. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 9781450380164. doi: 10.1145/3397271.3401281. URL https://doi.org/10.1145/3397271.3401281.

Noveen Sachdeva, Giuseppe Manco, Ettore Ritacco, and Vikram Pudi. Sequential variational autoencoders for collaborative filtering. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pp. 600–608, 2019.

Noveen Sachdeva, Yi Su, and Thorsten Joachims. Off-policy bandits with deficient support. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, pp. 965–975, New York, NY, USA, 2020. Association for Computing Machinery. doi: 10.1145/3394486.3403139.

Noveen Sachdeva, Mehak Preet Dhaliwal, Carole-Jean Wu, and Julian McAuley. Infinite recommendation networks: A data-centric approach. In *Advances in Neural Information Processing Systems*, 2022a.

Noveen Sachdeva, Ziran Wang, Kyungtae Han, Rohit Gupta, and Julian McAuley. Gapformer: Fast autoregressive transformers meet rnns for personalized adaptive cruise control. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2528–2535, 2022b. doi: 10.1109/ITSC55140.2022.9922275.

Noveen Sachdeva, Carole-Jean Wu, and Julian McAuley. On sampling collaborative filtering datasets. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, WSDM '22, 2022c.

Mattia Sangermano, Antonio Carta, Andrea Cossu, and Davide Bacciu. Sample condensation in online continual learning. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 01–08. IEEE, 2022.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

Robin Tibor Schirrmeister, Rosanne Liu, Sara Hooker, and Tonio Ball. When less is more: Simplifying inputs aids neural network understanding. *arXiv preprint arXiv:2201.05610*, 2022.

Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. Recommendations as treatments: Debiasing learning and evaluation. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1670–1679. PMLR, 2016.

Linda G Shapiro, George C Stockman, et al. *Computer vision*, volume 3. Prentice Hall New Jersey, 2001.

Rui Song, Dai Liu, Dave Zhenyu Chen, Andreas Festag, Carsten Trinitis, Martin Schulz, and Alois Knoll. Federated learning via decentralized dataset distillation in resource-constrained edge environments. *arXiv preprint arXiv:2208.11311*, 2022.

Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari S. Morcos. Beyond neural scaling laws: beating power law scaling via data pruning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022.

Harald Steck. Embarrassingly shallow autoencoders for sparse data. In *The World Wide Web Conference*, 2019.

Felipe Petroski Such, Aditya Rawal, Joel Lehman, Kenneth Stanley, and Jeffrey Clune. Generative teaching networks: Accelerating neural architecture search by learning to generate synthetic training data. In *International Conference on Machine Learning*, pp. 9206–9216. PMLR, 2020.

Ilia Sucholutsky and Matthias Schonlau. Soft-label dataset distillation and text dataset distillation. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2021.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BJlxm30cKm.

Luis Vicente, Gilles Savard, and Joaquim Júdice. Descent approaches for quadratic bilevel programming. *Journal of Optimization theory and applications*, 81(2):379–399, 1994.

Paul Vicol, Luke Metz, and Jascha Sohl-Dickstein. Unbiased gradient estimation in unrolled computation graphs with persistent evolution strategies. In *International Conference on Machine Learning*, pp. 10553–10563. PMLR, 2021.

Paul Vicol, Jonathan P Lorraine, Fabian Pedregosa, David Duvenaud, and Roger B Grosse. On implicit bias in overparameterized bilevel optimization. In *International Conference on Machine Learning*. PMLR, 2022.

Kai Wang, Bo Zhao, Xiangyu Peng, Zheng Zhu, Shuo Yang, Shuo Wang, Guan Huang, Hakan Bilen, Xinchao Wang, and Yang You. Cafe: Learning to condense dataset by aligning features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12196–12205, 2022.

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

Max Welling. Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, 2009.

Felix Wiewel and Bin Yang. Condensed composite memory continual learning. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2021.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6.

D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997. doi: 10.1109/4235.585893.

Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, Jinshi Huang, Charles Bai, et al. Sustainable ai: Environmental implications, challenges and opportunities. *Proceedings of Machine Learning and Systems*, 4:795–813, 2022.

Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys (CSUR)*, 2020.

Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, 2019.

Yuhuai Wu, Mengye Ren, Renjie Liao, and Roger Grosse. Understanding short-horizon bias in stochastic meta-optimization. In *International Conference on Learning Representations*, 2018.

Yuanhao Xiong, Ruochen Wang, Minhao Cheng, Felix Yu, and Cho-Jui Hsieh. Feddm: Iterative distribution matching for communication-efficient federated learning. *arXiv preprint arXiv:2207.09653*, 2022.

Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation. In *International Conference on Machine Learning*, pp. 12674–12685. PMLR, 2021.

Bo Zhao and Hakan Bilen. Synthesizing informative training samples with gan. *arXiv preprint arXiv:2204.07513*, 2022.

Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2023.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *International Conference on Learning Representations*, 2021.

Xiao Zhou, Renjie Pi, Weizhong Zhang, Yong Lin, Zonghao Chen, and Tong Zhang. Probabilistic bilevel coreset selection. In *International Conference on Machine Learning*, pp. 27287–27302. PMLR, 2022a.

Yanlin Zhou, George Pu, Xiyao Ma, Xiaolin Li, and Dapeng Wu. Distilled one-shot federated learning. *arXiv preprint arXiv:2009.07999*, 2020.

Yongchao Zhou, Ehsan Nezhadarya, and Jimmy Ba. Dataset distillation using neural feature regression. In *Advances in Neural Information Processing Systems*, 2022b.

A Notation

Dataset related

$\mathcal{D} \triangleq \{(x_i \in \mathcal{X}, y_i \in \mathcal{Y})\}_{i=1}^{\lvert\mathcal{D}\rvert}$	The target dataset to be distilled
$\mathcal{X}$	Data domain
$\mathcal{Y}$	Predictand domain
$\mathcal{C}$	Set of unique classes in $\mathcal{Y}$
$\mathcal{D}^c \triangleq \{(x_i, y_i) \mid y_i = c\}_{i=1}^{\lvert\mathcal{D}\rvert}$	Portion of $\mathcal{D}$ with class c
$\mathbf{X} \triangleq [x_i]_{i=1}^{\lvert\mathcal{D}\rvert}$	Matrix of all features in $\mathcal{D}$
$\mathbf{Y} \triangleq [y_i]_{i=1}^{\lvert\mathcal{D}\rvert}$	Matrix of all predictands in $\mathcal{D}$
n	Size of data summary
$\mathcal{D}_{\mathsf{syn}} \triangleq \{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^{n}$	Data summary
$\mathcal{D}_{\mathsf{syn}}^c \triangleq \{(\tilde{x}_i, \tilde{y}_i) \mid \tilde{y}_i = c\}_{i=1}^{n}$	Portion of $\mathcal{D}_{\mathsf{syn}}$ with class c
$\mathbf{X}_{\mathsf{syn}} \triangleq [\tilde{x}_i]_{i=1}^{n}$	Matrix of all features in $\mathcal{D}_{\mathsf{syn}}$
$\mathbf{Y}_{\mathsf{syn}} \triangleq [\tilde{y}_i]_{i=1}^{n}$	Matrix of all predictands in $\mathcal{D}_{\mathsf{syn}}$

Learning related

$\Phi_\theta : \mathcal{X} \mapsto \mathcal{Y}$	Learning algorithm parameterized by θ
$l : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$	Twice-differentiable cost function
$\mathcal{L}_{\mathcal{D}}(\theta) \triangleq \mathbb{E}_{(x,y)\sim\mathcal{D}}[l(\Phi_\theta(x), y)]$	Expected loss of Φ on $\mathcal{D}$
$\mathcal{L}_{\mathcal{D}_{\mathsf{syn}}}(\theta) \triangleq \mathbb{E}_{(x,y)\sim\mathcal{D}_{\mathsf{syn}}}[l(\Phi_\theta(x), y)]$	Expected loss of Φ on $\mathcal{D}_{\mathsf{syn}}$

General

$\dim(\mathcal{A})$	Size of basis of $\mathcal{A}$
$\lvert\mathcal{A}\rvert$	Number of elements in $\mathcal{A}$
$\sup$	Supremum
$\arg\min_\theta \ f(\theta)$	Optimum value of θ which minimizes $f(\theta)$
$\mathbb{E}_x[f(x)] \triangleq \sum_x p(x) \cdot f(x)$	Expected value of $f(x)$ when domain of x is discrete