A Discrepancy-Based Perspective on Dataset Condensation

Anonymous authors
Paper under double-blind review

Abstract

Given a dataset of finitely many elements $\mathcal{T} = \{\mathbf{x}_i\}_{i=1}^N$, the goal of dataset condensation (DC) is to construct a synthetic dataset $\mathcal{S} = \{\tilde{\mathbf{x}}_j\}_{j=1}^M$ which is significantly smaller ($M \ll N$) such that a model trained from scratch on $\mathcal{S}$ achieves comparable or even superior generalization performance to a model trained on $\mathcal{T}$. Recent advances in DC reveal a close connection to the problem of approximating the data distribution represented by $\mathcal{T}$ with a reduced set of points. In this work, we present a unified framework that encompasses existing DC methods and extend the task-specific notion of DC to a more general and formal definition using notions of discrepancy, which quantify the distance between probability distribution in different regimes. Our framework broadens the objective of DC beyond generalization, accommodating additional objectives such as robustness, privacy, and other desirable properties.

1 Introduction

Deep learning has achieved remarkable success across a wide range of real-world applications, largely driven by the development of powerful neural network architectures (LeCun et al., 2015; Schmidhuber, 2015). In contrast to classical machine learning models, modern deep learning systems rely heavily on large and diverse datasets. However, they often suffer from limited interpretability and tend to underperform when trained on smaller datasets. Both theoretical and empirical studies suggest that increasing the size of both the dataset and the model generally leads to improved performance. This insight has motivated practitioners to scale up training datasets and neural network architectures to unprecedented levels in pursuit of higher accuracy (Kaplan et al., 2020; Hestness et al., 2017). However, such scaling comes at a cost: larger models require significantly more energy and contribute to the growing carbon footprint during both training and deployment (Strubell et al., 2020; Anthony et al., 2020; Patterson et al., 2021).

Rather than solely pursuing ever-larger models, many studies advocate for the opposite direction: regularization and compression. By incorporating suitable regularization terms into the loss function to effectively solve the dual of an optimization problem with appropriate constraints, one can reduce network size while preserving, or even enhancing, downstream performance (Srivastava et al., 2014; Zhang et al., 2016; Hinton et al., 2015; Hubara et al., 2016; Carreira-Perpinán, 2017; Carreira-Perpinán & Idelbayev, 2017; Molchanov et al., 2016; Jaderberg et al., 2014). Most existing work on compression focuses on the model itself, typically keeping the dataset and learning algorithm fixed while modifying model architecture or parameter magnitudes. Notable exceptions include few-shot learning (Chen et al., 2019; Brown et al., 2020), which uses a limited number of training samples combined with more sophisticated learning algorithms, and dataset condensation (DC) (Zhao & Bilen, 2023; Zhao et al., 2021; Cazenavette et al., 2022), which aims to reduce dataset size while keeping both the model architecture and learning algorithm unchanged. Despite growing interest in DC, most existing methods are heuristic in nature and heavily reliant on empirical tuning. The landscape of DC techniques remains fragmented, with approaches differing widely in their motivations, mechanisms, and evaluation metrics. This lack of a coherent, principled foundation makes it difficult to systematically compare methods or extend them to new objectives. **In this work, we aim to unify and formalize the field by proposing a general framework that encompasses existing DC strategies while highlighting the underlying assumptions they share.**

To formally categorize deep learning tasks, we consider three fundamental components: the dataset (associated with a probability distribution or measure), the model (or hypothesis), and the algorithm. Let $(\Omega, \mathcal{F}, \mu)$ be a probability space, where Ω is the sample space, $\mathcal{F}$ is a σ-algebra, and μ is a probability measure over Ω. When Ω corresponds to a finite dataset $\Omega = \{\mathbf{x}_i\}_{i=1}^N$, the empirical measure is defined as $\mu = \frac{1}{N}\sum_{i=1}^N \delta_{\mathbf{x}_i}$, where $\delta_{\mathbf{x}_i}$ denotes the Dirac measure at $\mathbf{x}_i$. Let $\mathcal{H}$ denote the hypothesis space. A learning algorithm $\mathcal{A}$ maps a probability measure to a hypothesis, i.e., $\mathcal{A} : \mu \mapsto h \in \mathcal{H}$. A criterion φ evaluates the quality of a hypothesis under a given measure, defined as $\varphi : (h, \mu) \mapsto v \in \mathbb{R}$. A hypothesis h is said to be optimal with respect to μ if it minimizes (or maximizes, depending on context) the criterion $\varphi(h, \mu)$. Typically, learning algorithms are designed to find such an optimal hypothesis.

Many well-known concepts have emerged from modifying one of the three components (dataset, model, or algorithm) while keeping the others fixed. In contrast, jointly altering two or more components poses greater conceptual and technical challenges. Many existing research directions can be interpreted through the lens of holding some components fixed while varying others. Examples include:

1. **Model training**: Given a fixed probability measure μ and hypothesis space $\mathcal{H}$, design an algorithm $\mathcal{A}$ that returns a model h with optimal performance under criterion φ, i.e, $\mathcal{A}(\mu) = \arg\min_{h\in\mathcal{H}} \varphi(h, \mu)$;

2. **Model compression**: Again fixing μ and $\mathcal{H}$, design an algorithm that not only achieves optimal performance but also satisfies a sparsity constraint, i.e., $\mathcal{A}(\mu) = \arg\min_{h\in\mathcal{H}}\{\varphi(h, \mu) : \|h\|_0 \leq s\}$;

3. **Dataset compression**: Fixing the hypothesis space $\mathcal{H}$ and algorithm $\mathcal{A}$, find a smaller dataset $\tilde{\mathcal{D}}$ with associated empirical measure $\tilde{\mu}$ such that the model $h^*_{\tilde{\mu}} = \mathcal{A}(\tilde{\mu})$ remains optimal when evaluated on the original distribution μ, that is, $\tilde{\mu} = \arg\min_\nu \varphi(\mathcal{A}(\nu), \mu)$, where $\mathcal{A}(\nu) = \arg\min_{h\in\mathcal{H}} \varphi(h, \nu)$.

In the following sections, we first introduce a unified theoretical framework for dataset compression. We then use this framework to categorize a wide range of existing approaches according to four key dimensions: (1) The metric used to quantify the distance between probability distributions; (2) The feature space in which the dataset is condensed; (3) The parameter space over which models are optimized; and (4) The loss function and regularization terms employed during training.

Table 1: Summary of notation used throughout the paper.

Notation	Description
$\|\cdot\|$	Norm (context-dependent: may refer to ℓ_2, sup-norm, etc.)
Ω	Input domain or sample space, $\Omega \subseteq \mathbb{R}^n$
μ	Probability distribution (measure) over domain Ω
$\mathcal{P}(\Omega)$	Space of all probability distributions over Ω
$\mathcal{D}$	Finite dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ sampled from distribution μ
$\hat{\mu}_{\mathcal{D}}$	Empirical distribution induced by dataset $\mathcal{D}$
$\mathcal{H}(\Omega)$	Hypothesis space over domain Ω
$\overline{h(\mathcal{D})}$	Empirical average of hypothesis h over dataset $\mathcal{D}$
h^*_μ	Optimal hypothesis in $\mathcal{H}$ trained under distribution μ
$\varphi(h, \mu)$	Criterion of hypothesis h under distribution μ
$\mathcal{A}$	Learning algorithm mapping datasets to hypotheses
$\mathcal{C}^n(\Omega)$	Set of n-times continuously differentiable functions over Ω
$\nabla\mathcal{C}^n(\Omega)$	Set of gradients of functions in $\mathcal{C}^n(\Omega)$
$g^\#\mu$	Push-forward measure of μ via measurable map g
$D(\mu_1, \mu_2; \mathcal{H})$	Generalization discrepancy between μ_1 and μ_2 w.r.t. $\mathcal{H}$

2 Preliminaries and Notations

Let $\mathbb{R}^n$ denote the n-dimensional real vector space. A finite subset $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N \subseteq \mathbb{R}^n$ is referred to as a *dataset*, and naturally induces an empirical distribution $\hat{\mu} = \frac{1}{N}\sum_{i=1}^N \mathbf{1}_{\mathbf{x}_i}$, where $\mathbf{1}_{\mathbf{x}_i}$ denotes the Dirac measure at $\mathbf{x}_i$, or consistency. We will use distributional notation to represent datasets throughout. Let $\mathcal{P}(\Omega)$ denote the space of probability distributions over Ω. We assume that each sample in $\mathcal{D}$ is drawn i.i.d. from an unknown ground-truth distribution $\mu \in \mathcal{P}(\Omega)$, i.e., $\hat{\mu}$ serves as an empirical approximation of μ.

To develop learning algorithms, we must first specify the performance criteria under consideration. For standard tasks in deep learning such as classification and regression, let $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ be a loss function, and let $h \in \mathcal{H}(\Omega)$ denote a model from a given hypothesis space. Two common evaluation criteria are:

1. **Test performance** is defined by the expected loss under the true distribution: $\varphi(h, \mu) = \mathbb{E}_{(\mathbf{x},y)\sim\mu}[\ell(h(\mathbf{x}), y)]$;
2. **Adversarial robustness**, with respect to a perturbation radius $\varepsilon > 0$ and norm $\|\cdot\|$, is measured by the worst-case loss under bounded input perturbations:

$$\varphi_\varepsilon(h, \mu) = \mathbb{E}_{(\mathbf{x},y)\sim\mu}\left[\max_{\|\delta\|\leq\varepsilon} \ell(h(\mathbf{x}+\delta), y)\right].$$

For a fixed distribution μ, the evaluation criterion $\varphi(h, \mu)$ quantifies how well a model h performs on data drawn from μ. In this sense, φ serves as a model-dependent statistical measurement of a distribution. When comparing two distributions, a natural approach is to evaluate both using the same criterion and model. However, such a comparison is necessary but not sufficient for distinguishing distributions: two distributions may yield identical performance under one model or criterion yet differ under another.

To address this limitation, we can define model-independent metrics by either maximizing or averaging the criterion over a hypothesis space $\mathcal{H}$, such as $\varphi(\mu) = \sup_{h\in\mathcal{H}} \varphi(h, \mu)$ or $\mathbb{E}_{h\in\mathcal{H}}[\varphi(h, \mu)]$. We refer to such quantities as *distribution discrepancy (DD)*, which is a criterion that abstracts away the specific model and captures broader statistical properties of the distribution.

In summary, we categorize statistical metrics for comparing two distributions μ_1 and μ_2 as follows:

1. **Model-specific discrepancy**: $D(\mu_1, \mu_2; h) = |\varphi(h, \mu_1) - \varphi(h, \mu_2)|$, which compares distributions with respect to a fixed model h;
2. **Model-agnostic discrepancy**: $D(\mu_1, \mu_2; \mathcal{H}) = \sup_{h\in\mathcal{H}} |\varphi(h, \mu_1) - \varphi(h, \mu_2)|$, which captures the maximal difference in performance across all models in the hypothesis space.

It is straightforward that $D(\mu_1, \mu_2; h) = 0$ whenever $\mu_1 = \mu_2$. However, to qualify as a valid statistical distance, a metric must also satisfy the converse: if $D(\mu_1, \mu_2; h) = 0$, then $\mu_1 = \mu_2$. In other words, the metric must be capable of distinguishing any two distinct distributions. Such properties are satisfied by several well-established metrics in statistics, particularly in the context of two-sample testing. Below, we summarize a few canonical examples:

1. **Integral Probability Metric (IPM)**: Let $\varphi(h, \mu) = \mathbb{E}_{(\mathbf{x},y)\sim\mu}[h(\mathbf{x}, y)]$, and define:

$$IPM(\mu_1, \mu_2; \mathcal{H}) = \sup_{h\in\mathcal{H}} \big|\varphi(h, \mu_1) - \varphi(h, \mu_2)\big|.$$

   The IPM forms a valid metric if the function class $\mathcal{H}$ is rich enough to separate distributions.

2. **Maximum Mean Discrepancy (MMD)**: Assume $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS), and again define $\varphi(h, \mu) = \mathbb{E}_{(\mathbf{x},y)\sim\mu}[h(\mathbf{x}, y)]$. Then,

$$MMD(\mu_1, \mu_2; \mathcal{H}) = \sup_{\|h\|_{\mathcal{H}}\leq 1} \big|\varphi(h, \mu_1) - \varphi(h, \mu_2)\big|. \tag{1}$$

   MMD is a special case of IPM where the function class is the unit ball in an RKHS. With a characteristic kernel, MMD satisfies $MMD(\mu_1, \mu_2; \mathcal{H}) = 0 \Leftrightarrow \mu_1 = \mu_2$.

3. **Wasserstein Metric (WM)**: Let d be a ground distance on Ω, and fix $p \in [1, +\infty]$. The Wasserstein distance of order p is defined as:

$$WM_p(\mu_1, \mu_2; \mathcal{H}) = \inf_{\gamma \in \Gamma(\mu_1, \mu_2)} \left(\mathbb{E}_{((\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2)) \sim \gamma} [d((\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2))^p] \right)^{1/p},$$

where $\Gamma(\mu_1, \mu_2)$ denotes the set of all couplings (joint distributions) with marginals μ_1 and μ_2. In the special case $p = 1$, the Wasserstein metric admits a dual formulation:

$$WM_1(\mu_1, \mu_2; \mathcal{H}) = \sup_{Lip(h) \leq 1} \left| \varphi(h, \mu_1) - \varphi(h, \mu_2) \right|,$$

where the supremum is taken over all 1-Lipschitz functions h.

We observe that both MMD and the WM are special cases of IPM, distinguished by their choice of hypothesis space $\mathcal{H}$. To briefly recap, suppose $\mathcal{H}$ is an RKHS over Ω, equipped with a kernel $k : \Omega \times \Omega \to \mathbb{R}$, inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$, and associated feature maps $\varphi_{\mathbf{x}} : \Omega \to \mathcal{H}$, then, by the Riesz representation theorem, $\mathcal{H}$ satisfies the reproducing property: for all $h \in \mathcal{H}$ and $\mathbf{x} \in \Omega$, there exists $\varphi_{\mathbf{x}} \in \mathcal{H}$ such that $h(\mathbf{x}) = \langle h, \varphi_{\mathbf{x}} \rangle_{\mathcal{H}}$ and $k(\mathbf{x}_1, \mathbf{x}_2) = \langle \varphi_{\mathbf{x}_1}, \varphi_{\mathbf{x}_2} \rangle_{\mathcal{H}}$. Popular choices of kernels include:

1. **γ-exponential kernel**: $k^{\gamma}(\mathbf{x}_1, \mathbf{x}_2) := \exp(-c\|\mathbf{x}_1 - \mathbf{x}_2\|^{\gamma})$ with $0 < \gamma \leq 2$. Special cases include the Laplacian kernel ($\gamma = 1$) and Gaussian kernel ($\gamma = 2$);
2. **Neural tangent kernel (NTK)**: $k^{\theta}(\mathbf{x}_1, \mathbf{x}_2) := \mathbb{E}_{\theta}\left[\nabla f_{\theta}(\mathbf{x}_1)^T \nabla f_{\theta}(\mathbf{x}_2)\right]$, where f_{θ} is a neural network with parameters θ.

The *kernel embedding* of a probability distribution μ into an RKHS $\mathcal{H}$ is defined as $\tilde{\mu} := \mathbb{E}_{\mathbf{x} \sim \mu}[k(\mathbf{x}, \cdot)] = \mathbb{E}_{\mathbf{x} \sim \mu}[\varphi_{\mathbf{x}}]$, where $\varphi_{\mathbf{x}}$ is the feature map associated with kernel k. This embedding enables expressing the expectation of any function $f \in \mathcal{H}$ as an inner product between f and the kernel embedding, i.e., $\mathbb{E}_{\mathbf{x} \sim \mu}[f(\mathbf{x})] = \langle f, \tilde{\mu} \rangle_{\mathcal{H}}$. In this way, the MMD between two distributions μ_1 and μ_2 defined in Equation (1) can be rewritten as the norm of the difference between their kernel embeddings:

$$MMD(\mu_1, \mu_2; \mathcal{H}) = \sup_{\|h\|_{\mathcal{H}} \leq 1} \left| \langle h, \tilde{\mu}_1 - \tilde{\mu}_2 \rangle \right| = \|\tilde{\mu}_1 - \tilde{\mu}_2\|_{\mathcal{H}}.$$

Expanding the squared norm yields a kernel-based expression that avoids explicit use of feature maps:

$$MMD(\mu_1, \mu_2; \mathcal{H})^2 = \mathbb{E}_{\mathbf{x}_1, \mathbf{x}_1' \sim \mu_1}[k(\mathbf{x}_1, \mathbf{x}_1')] - 2 \cdot \mathbb{E}_{\mathbf{x}_1 \sim \mu_1, \mathbf{x}_2 \sim \mu_2}[k(\mathbf{x}_1, \mathbf{x}_2)] + \mathbb{E}_{\mathbf{x}_2, \mathbf{x}_2' \sim \mu_2}[k(\mathbf{x}_2, \mathbf{x}_2')].$$

This kernel-only formulation is particularly useful for empirical estimation of MMD from samples.

The last important property of distributional metrics is injectivity: under what conditions does a vanishing discrepancy imply that two distributions are identical? Formally, we ask when $D(\mu_1, \mu_2; \mathcal{H}) = 0$ implies $\mu_1 = \mu_2$. For $\Omega \subseteq \mathbb{R}^n$, let $\mathcal{C}(\Omega)$ (resp. $\mathcal{C}^{\infty}(\Omega)$) denote the set of all bounded continuous (resp. smooth) functions over Ω. We summarize the sufficient conditions for injectivity in the following proposition:

Proposition 1. *Let $DD(\mu_1, \mu_2; \mathcal{H})$ be one of the discrepancy measures previously defined. Then $DD(\mu_1, \mu_2; \mathcal{H}) = 0$ implies $\mu_1 = \mu_2$ if any of the following holds:*

(1) $DD(\mu_1, \mu_2; \mathcal{H}) = IPM(\mu_1, \mu_2; \mathcal{H})$, for $\mathcal{H} = \mathcal{C}(\Omega), \mathcal{C}^{\infty}(\Omega), \nabla\mathcal{C}^{\infty}(\Omega)$;

(2) $DD(\mu_1, \mu_2; \mathcal{H}) = MMD(\mu_1, \mu_2; \mathcal{H})$, for $\mathcal{H}$ an RKHS with a universal kernel;

(3) $DD(\mu_1, \mu_2; \mathcal{H}) = WM_p(\mu_1, \mu_2; \mathcal{H})$, for any $p \in [1, +\infty]$.

Proof. (Adapted from (Dudley, 2002, Lemma 9.3.2)) Let $U \subseteq \Omega$ be an arbitrary open set. Then the indicator function $\mathbf{1}_U$ is a characteristic functional of the distribution μ, and testing against it suffices to distinguish measures. Since $\mathbf{1}_U$ can be uniformly approximated by bounded continuous functions, it follows that if $\mathcal{H}(\Omega) \subseteq \mathcal{C}(\Omega)$ is dense, then the discrepancy $D(\mu_1, \mu_2; \mathcal{H})$ detects all differences between μ_1 and μ_2. The rest follows from known results on density of smooth functions in $\mathcal{C}(\Omega)$, and the universal approximation property of RKHSs with universal kernels. □

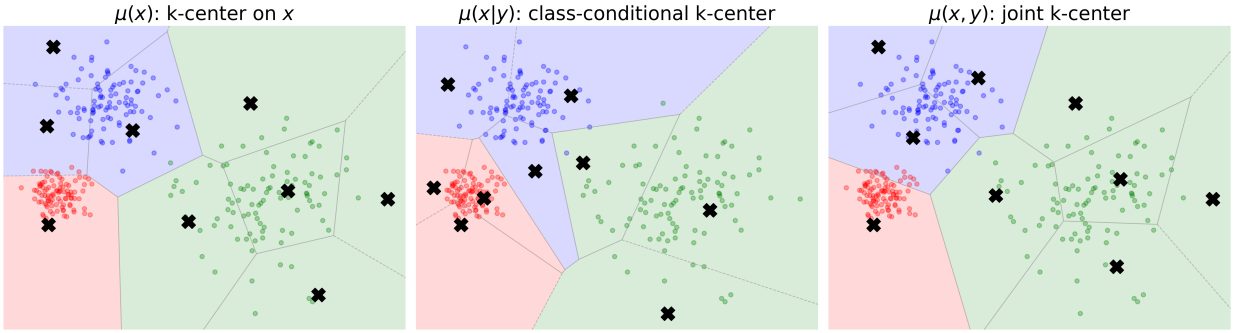


Figure 1: Comparison of k-center selection for $\mu(x)$, $\mu(x|y)$ and $\mu(x, y)$. Note how the selected subset points differ depending on the distribution being considered.

This observation shows that the density of $\mathcal{H}(\Omega)$ in $\mathcal{C}(\Omega)$ is sufficient (though stronger than necessary) for injectivity. In fact, we only require the following weaker condition: for any open set $U \subseteq \Omega$ and any $\varepsilon > 0$, there exists $h \in \mathcal{H}(\Omega)$ such that $\|h - \mathbf{1}_U\| \leq \varepsilon$. Notably, $\mathcal{H}(\Omega)$ need not even be a vector space, only that it is rich enough to approximate indicator functions.

3 Dataset Condensation

Most of the DC literature focuses on classification tasks, where the condensation process is typically performed on a per-class basis. That is, for each class label y, condensation aims to approximate the conditional distribution of the data given the class. Consequently, the key objective becomes measuring the discrepancy between the class-conditional distributions $\mu_1(\cdot|y)$ and $\mu_2(\cdot|y)$. Note that we can also consider soft labels with stochasticity, allowing data and labels to be synthesized jointly. From this perspective, dataset condensation can be naturally generalized to regression tasks. In the following discussion, we adopt this classification convention and assume that μ_1 and μ_2 denote class-conditional distributions. All functions f are understood to act on the input variable $\mathbf{x}$, i.e., $f : \mathbb{R}^n \to \mathbb{R}$. In the current literature, DC is typically studied in the context of supervised learning. Given a target data distribution μ_1, the goal is to construct a synthetic distribution μ_2 such that models trained on μ_2 generalize similarly to those trained on μ_1. Formally, this objective can be formulated as the following bi-level optimization problem:

$$\inf_{\mu_2 \in \mathcal{H}} \{\varphi(h^*_{\mu_2}, \mu_1) : h^*_{\mu_2} \in \arg\min_{h \in \mathcal{H}} \varphi(h, \mu_2)\}, \tag{DC}$$

where $\mathcal{H}$ is a hypothesis space and $\varphi : \mathcal{H} \times \mathcal{P} \to \mathbb{R}$ denotes the expected generalization loss, defined as $\varphi(h, \mu) := \mathbb{E}_{(\mathbf{x},y)\sim\mu}[\ell(h(\mathbf{x}), y)]$, for some loss function ℓ. Intuitively, we aim to find a synthetic distribution μ_2 such that the model trained on μ_2 performs well on the original distribution μ_1. Naturally, the optimal solution occurs when $\mu_1 = \mu_2$, i.e., when there is no distributional shift.

Remark 2 (Extension to DC with soft label (Qin et al., 2024))**.** *We extend the standard dataset condensation framework by allowing L labels to take values in the probability simplex Δ^{L-1}, i.e., each sample is of the form $(\mathbf{x}, \tilde{y})$ with $\tilde{y} \in \Delta^{L-1}$. In this setting, the synthetic distribution μ_2 is defined over $\mathcal{X} \times \Delta^{L-1}$, and the loss is generalized to $\ell(h(\mathbf{x}), \tilde{y})$, e.g., using cross-entropy or squared loss. This formulation enables the condensation process to match not only the input distribution but also the uncertainty structure of labels, effectively capturing the joint distribution $\mu(\mathbf{x}, \tilde{y})$ rather than class-conditional distributions alone. In particular, soft labels provide richer supervision by encoding inter-class similarity and ambiguity, which can be especially beneficial in low-data regimes.*

Figure 1 illustrates this effect: compared to hard-label condensation, which selects representative points based solely on geometric coverage in $\mathcal{X}$, soft-label condensation additionally accounts for variations in label distributions, leading to the selection of samples that better capture decision boundaries and regions of high uncertainty.

In practice, neither μ_1 or μ_2 is known explicitly. Instead, they are approximated by empirical distributions: $\hat{\mu}_1$ which is induced by the full dataset $\mathcal{T} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, and $\hat{\mu}_2$ which is induced by a small synthetic dataset $\mathcal{S} = \{(\tilde{\mathbf{x}}_j, \tilde{y}_j)\}_{j=1}^M$ with $M \ll N$. In the context of dataset condensation, the size of condensed dataset $\mathcal{S}$ is expected to be much smaller than the original dataset $\mathcal{T}$, i.e., $M \ll N$. To better understand the optimization goal, define the *generalization discrepancy (GD)* between two distributions μ_1 and μ_2 with respect to a hypothesis space $\mathcal{H}$ as $GD(\mu_1, \mu_2; \mathcal{H}) := |\varphi(h^*_{\mu_2}, \mu_1) - \varphi(h^*_{\mu_1}, \mu_1)|$. This measures how much generalization performance deteriorates when we train on μ_2 rather than μ_1. Note that since $h^*_{\mu_1}$ minimizes $\varphi(\cdot, \mu_1)$, minimizing $\varphi(h^*_{\mu_2}, \mu_1)$ over μ_2 is equivalent to minimizing $GD(\mu_1, \mu_2; \mathcal{H})$. We can upper bound this generalization discrepancy using the distribution discrepancy $DD(\mu_1, \mu_2; \mathcal{H})$ as follows:

$$\begin{aligned} GD(\mu_1, \mu_2; \mathcal{H}) &= |\varphi(h^*_{\mu_2}, \mu_1) - \varphi(h^*_{\mu_2}, \mu_2) + \varphi(h^*_{\mu_2}, \mu_2) - \varphi(h^*_{\mu_1}, \mu_1)| \\ &\leq |\varphi(h^*_{\mu_2}, \mu_1) - \varphi(h^*_{\mu_2}, \mu_2) + \varphi(h^*_{\mu_1}, \mu_2) - \varphi(h^*_{\mu_1}, \mu_1)| \\ &\leq |\varphi(h^*_{\mu_2}, \mu_1) - \varphi(h^*_{\mu_2}, \mu_2)| + |\varphi(h^*_{\mu_1}, \mu_2) - \varphi(h^*_{\mu_1}, \mu_1)| \leq 2 \cdot DD(\mu_1, \mu_2; \mathcal{H}), \end{aligned}$$

assuming $\mathcal{H}$ contains both $h^*_{\mu_1}$ and $h^*_{\mu_2}$. This bound reveals two key insights:

- Minimizing the generalization discrepancy $GD(\mu_1, \mu_2; \mathcal{H})$ does not ensure that $\mu_1 = \mu_2$. In fact, small generalization error can arise from significantly different distributions.
- Conversely, minimizing the distribution discrepancy $DD(\mu_1, \mu_2; \mathcal{H})$ serves as a principled surrogate objective: it bounds the generalization loss difference and is often much easier to optimize.

Indeed, generalization discrepancy requires solving a bi-level optimization problem, i.e., training a model at each iteration to evaluate the outer objective, whereas minimizing the distribution discrepancy typically involves a single-level optimization over distributional parameters or dataset embeddings.

Example 3 (Zero generalization discrepancy under distribution mismatch)**.** *Let $\mathcal{X} = \mathbb{R}$ and $\mathcal{Y} = \{-1, +1\}$. Consider the hypothesis class $\mathcal{H} := \{h_t(x) = \text{sign}(x - t) : t \in \mathbb{R}\}$ and the 0–1 loss $\ell(h(x), y) = \mathbf{1}\{h(x) \neq y\}$. Define two distributions μ_1 and μ_2 on $\mathcal{X} \times \mathcal{Y}$ as follows:*

- *μ_1: $x \sim \text{Uniform}([-1, 1])$, and $y = \text{sign}(x)$;*
- *μ_2: $\mu_2 = \frac{1}{2}\delta_{(-1,-1)} + \frac{1}{2}\delta_{(1,1)}$.*

For μ_1, the Bayes-optimal classifier is $h^*_{\mu_1}(x) = \text{sign}(x)$, which achieves zero risk $\varphi(h^*_{\mu_1}, \mu_1) = 0$. For μ_2, any threshold $t \in (-1, 1)$ yields zero classification error on the support $\{-1, 1\}$. Hence, $h^*_{\mu_2} \in \{\text{sign}(x - t) : t \in (-1, 1)\}$, and in particular, $h(x) = \text{sign}(x)$ is one such minimizer. Therefore, $\varphi(h^*_{\mu_2}, \mu_2) = 0$. Selecting $h^*_{\mu_2}(x) = \text{sign}(x)$, we evaluate its performance on μ_1: $\varphi(h^*_{\mu_2}, \mu_1) = 0 = \varphi(h^*_{\mu_1}, \mu_1)$. Thus, the generalization discrepancy satisfies $GD(\mu_1, \mu_2; \mathcal{H}) = 0$. Although $\mu_1 \neq \mu_2$ (they have different supports and densities), they even induce a common optimal classifier within $\mathcal{H}$ for this specific setting. Consequently, training on μ_2 yields the same generalization performance on μ_1 as training on μ_1 itself.

Let us now take a closer look at the generalization discrepancy $GD(\mu_1, \mu_2; \mathcal{H})$. Recall that this quantity measures how much the test performance (on distribution μ_1) degrades when we train a model on a different distribution μ_2. In general, $\varphi(h, \mu) = \mathbb{E}_{(\mathbf{x},y)\sim\mu}[\ell(h(\mathbf{x}), y)]$ is unbounded due to the potential unboundedness of the loss function ℓ and the hypothesis h. However, this behavior can be controlled by restricting the hypotheses to a uniformly bounded class over a compact input domain. For instance, if we assume normalized inputs $\mathbf{x} \in [0, 1]^n$ and continuous hypotheses $h_\theta : [0, 1]^n \to \mathbb{R}$ with uniformly bounded parameters $\|\theta\| \leq B$, then the image of any model $h \in \mathcal{H}$, that is, $h([0, 1]^n])$, is contained in a compact set $\mathbf{K} \subseteq \mathbb{R}$. Assuming further that the loss function $\ell(\cdot, \cdot)$ is L-Lipschitz continuous in its first argument over $\mathbf{K}$, we obtain

$$GD(\mu_1, \mu_2; \mathcal{H}) \leq L \cdot \|h^*_{\mu_1} - h^*_{\mu_2}\|,$$

where $\|h^*_{\mu_1} - h^*_{\mu_2}\| := \sup_{\mathbf{x}\in[0,1]^n} |h^*_{\mu_1}(\mathbf{x}) - h^*_{\mu_2}(\mathbf{x})|$ denotes the uniform (sup) norm over inputs. Now suppose both models $h^*_{\mu_1}$ and $h^*_{\mu_2}$ belong to a neural network class with fixed architecture, and let $\theta^*_{\mu_1}$ and $\theta^*_{\mu_2}$ be

the parameter vectors of these trained models. If the neural network is Lipschitz continuous with respect to its parameters, i.e., $\sup_{\mathbf{x}\in[0,1]^n} |h^*(\theta^*_{\mu_1}, \mathbf{x}) - h^*(\theta^*_{\mu_2}, \mathbf{x})| \leq C \cdot \|\theta^*_{\mu_1} - \theta^*_{\mu_2}\|$, for some global constant C. Then we obtain:

$$\|h^*_{\mu_1} - h^*_{\mu_2}\| \leq C \cdot \|\theta^*_{\mu_1} - \theta^*_{\mu_2}\|.$$

This motivates two additional discrepancy measures:

1. **Value discrepancy (VD)**, defined by $VD(\mu_1, \mu_2; \mathcal{H}) := \|h^*_{\mu_1} - h^*_{\mu_2}\|$;
2. **Parameter discrepancy (PD)**, defined by $PD(\mu_1, \mu_2; \mathcal{H}) := \|\theta^*_{\mu_1} - \theta^*_{\mu_2}\|$.

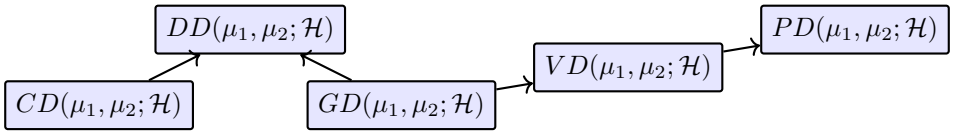
These relationships imply a hierarchy of upper bounds on generalization discrepancy:

$$GD(\mu_1, \mu_2; \mathcal{H}) \leq L \cdot VD(\mu_1, \mu_2; \mathcal{H}) \leq LC \cdot PD(\mu_1, \mu_2; \mathcal{H}).$$

Thus, value discrepancy and parameter discrepancy provide more tractable surrogates for bounding or approximating generalization performance loss in practice. They also highlight that preserving either model predictions or parameter configurations under distribution shifts can serve as effective criteria for dataset condensation.

Another classical method to verify whether two probability distributions are identical is through their characteristic functions. For a distribution μ, the characteristic function is defined as: $F_\mu(\mathbf{t}) := \mathbb{E}_{\mathbf{x}\sim\mu}[e^{i\langle \mathbf{x},\mathbf{t}\rangle}]$ for $\mathbf{t} \in \mathbb{R}^n$. Two distributions μ_1 and μ_2 are equal if and only if $F_{\mu_1}(\mathbf{t}) = F_{\mu_2}(\mathbf{t})$ for all $\mathbf{t}$. This motivates the definition of *characteristic discrepancy (CD)* defined as $CD(\mu_1, \mu_2; \mathcal{H}) := \|F_{\mu_1} - F_{\mu_2}\|$, where the norm is typically taken to be the L_∞ or L_2 norm over $\mathbf{t} \in \mathbb{R}^n$. Since the set of test functions $\{e^{i\langle \mathbf{x},\mathbf{t}\rangle} : \mathbf{t} \in \mathbb{R}^n\}$ spans only a subspace of $\mathcal{C}(\Omega)$, it follows that $CD(\mu_1, \mu_2; \mathcal{H}) \leq D(\mu_1, \mu_2; \mathcal{H})$, where $D(\mu_1, \mu_2; \mathcal{H})$ denotes a more general IPM over a larger function class $\mathcal{H}$. Importantly, the characteristic discrepancy is purely distributional: it measures the closeness of two distributions in a general sense, without reference to a learning task. Therefore, it provides no direct guarantee on the generalization performance of models trained on one distribution and tested on another.

To summarize the relationships among the discussed discrepancy measures, we provide the following schematic diagram. Arrows indicate an upper bound (up to a constant), i.e., $A \to B$ implies $A \leq cB$ for some constant $c < \infty$.



In the context of DC methods, these discrepancies serve as tools to quantify the mismatch between the source and target distributions. By choosing a suitable discrepancy with balance between computational tractability and task relevance, one can guide adaptation strategies that reduce distributional shifts, thereby improving cross-domain generalization.

4 Change of Space

In the previous section, we discussed various types of discrepancies that measure the similarity between two probability distributions. Some of these are proper distance metrics, while others not only quantify distributional similarity but also provide guarantees on generalization performance. At a high level, dataset condensation aims to minimize the discrepancy between a target distribution μ_1 (e.g., the original data distribution) and a synthesized distribution μ_2, by solving: $\min_{\mu_2} D(\mu_1, \mu_2; \mathcal{H})$. This formulation implicitly assumes that both μ_1 and μ_2 are defined over a common input space $\Omega \subseteq \mathbb{R}^n$. Now consider a more general case where there is a measurable mapping $g : (\mathbb{R}^n, \mathcal{F}_1) \to (\mathbb{R}^m, \mathcal{F}_2)$ between two measurable spaces, and where the original space $\mathbb{R}^n$ is endowed with a measure $\mu : \mathcal{F}_1 \to [0, +\infty]$. The *push-forward measure* of μ under g, denoted by $g^\# \mu$, is a measure on $\mathbb{R}^m$ defined as:

$$g^\# \mu(U) = \mu \circ g^{-1}(U), \ \forall\, U \in \mathcal{F}_2.$$

In deep learning applications, high-dimensional input spaces often lead to optimization challenges, both computationally and statistically, due to the curse of dimensionality. One common strategy to mitigate this issue is to employ latent variable generative models that transform high-dimensional data into a lower-dimensional feature space. A prominent example is the autoencoder, which consists of two components: an *encoder* $g_e : \mathbb{R}^n \to \mathbb{R}^m$ which maps high-dimensional inputs to a low-dimensional latent space, and a *decoder* $g_d : \mathbb{R}^m \to \mathbb{R}^n$ which reconstructs the input from its latent representation. The decoder g_d is intended to approximate the inverse of the encoder g_e, so that their composition $g_d \circ g_e$ approximates the identity function $\mathrm{id}_{\mathbb{R}^n}$, or at least locally approximates id_Ω for a compact domain $\Omega \subseteq \mathbb{R}^n$.

This structure enables us to push forward probability measures from the original space to the latent space via the encoder. That is, given a measure μ over $\mathbb{R}^n$, we define the push-forward measure $g_e^\# \mu := \mu \circ g_e^{-1}$, which is a measure on $\mathbb{R}^m$. With this formulation, we gain additional flexibility in choosing the space where we perform distribution matching and optimization, either in the original input space or in the latent feature space. See Table 2 for a summary of possible choices. Importantly, if optimization is conducted in the latent space, we must use the push-forward (via the decoder) to recover the corresponding distribution in the input space.

Table 2: Regimes of optimizing and matching distributions in input and latent spaces.

Match \ Optimize	Input space $\mathbb{R}^n$	Latent space $\mathbb{R}^m$
Input space $\mathbb{R}^n$	$\min_{\mu_2 \in \mathcal{P}(\mathbb{R}^n)} D(\mu_1, \mu_2; \mathcal{H})$	$\min_{\nu \in \mathcal{P}(\mathbb{R}^m)} D(\mu, g_d^\# \nu; \mathcal{H})$
Latent space $\mathbb{R}^m$	$\min_{\mu_2 \in \mathcal{P}(\mathbb{R}^n)} D(g_e^\# \mu_1, g_e^\# \mu_2; \mathcal{H})$	$\min_{\nu \in \mathcal{P}(\mathbb{R}^m)} D(g_e^\# \mu, \nu; \mathcal{H})$

According to Proposition 1, for suitable choices of function class $\mathcal{H}$, the discrepancy measure $D(\mu_1, \mu_2; \mathcal{H})$ becomes a valid distance metric over distributions on the input space $\mathbb{R}^n$. A natural question arises: *Does this metric property still hold when we change the space in which distribution matching and optimization take place?* We consider three distinct cases involving push-forward measures:

1. **Push-forward both distributions via the encoder**: Define

$$d(\mu_1, \mu_2) := D(g_e^\# \mu_1, g_e^\# \mu_2; \mathcal{H}).$$

   This defines a metric on distributions over $\mathbb{R}^n$ if and only if $d(\mu_1, \mu_2) = 0$ implies $\mu_1 = \mu_2$. By definition, this holds precisely when $g_e^\# \mu_1 = g_e^\# \mu_2$ implies $\mu_1 = \mu_2$, which is true if and only if the encoder g_e is injective.

2. **Push-forward only the synthetic distribution via the decoder**: Define

$$d(\mu, \nu) = D(\mu, g_d^\# \nu; \mathcal{H}),$$

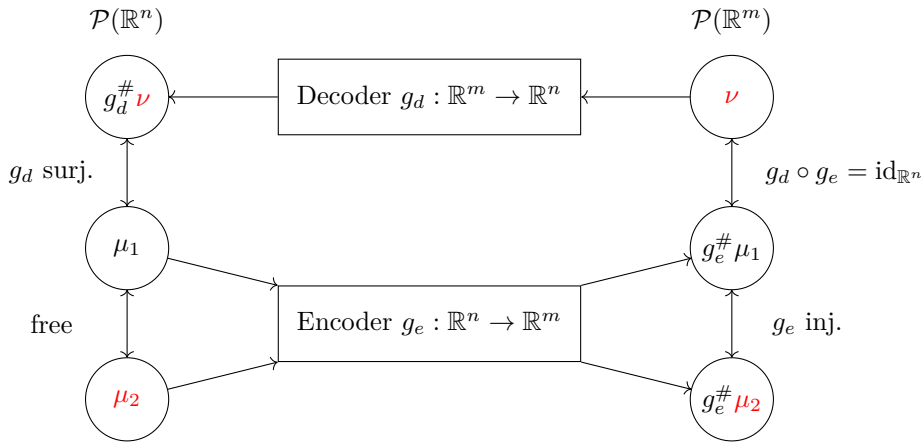
   where μ is defnied over $\mathbb{R}^n$ and ν over $\mathbb{R}^m$. Here $d(\mu, \nu) = 0$ implies that $\mu = g_d^\# \nu$. Therefore, $d(\mu, \nu)$ can vanish for some ν if and only if the decoder g_d is surjective, i.e., every μ over $\mathbb{R}^n$ can be obtained as the push-forward of some ν.

3. **Push-forward only the target distribution via the encoder**: Define

$$d(\mu, \nu) = D(g_e^\# \mu, \nu; \mathcal{H}),$$

   where again μ is over $\mathbb{R}^n$ and ν over $\mathbb{R}^m$. In this case, $d(\mu, \nu) = 0$ implies $\nu = g_e^\# \mu$, which requires the encoder g_e to be surjective. Furthermore, if we define the synthetic distribution over $\mathbb{R}^n$ as $g_d^\# \nu$, then $g_d^\# \nu = g_d^\# g_e^\# \mu = (g_d \circ g_e)^\# \mu$. Thus $d(\mu, \nu) = 0$ implies $\mu = g_d^\# \nu$ if and only if $g_d \circ g_e = \mathrm{id}_{\mathbb{R}^n}$, i.e., the autoencoder perfectly reconstructs the input.

From the above discussion, we see that there is no free lunch in dataset condensation: while performing optimization or distribution matching directly in the original input space ensures full fidelity, it is computationally expensive due to high dimensionality. On the other hand, dimensionality reduction via an autoencoder offers computational benefits, but necessarily incurs approximation error arising from the non-injectivity of the encoder and the non-surjectivity of the decoder, which can compromise the accuracy of distribution matching. We summarize the relationships among the different matching strategies and their required properties in the following diagram:



In practice, when given a distribution μ over $\mathbb{R}^n$, the expectation under its push-forward $g^{\#}\mu$ over $\mathbb{R}^m$ can be computed using the change of variable formula $\mathbb{E}_{\mathbf{z}\sim g^{\#}\mu}[\mathbf{z}] = \mathbb{E}_{\mathbf{x}\sim\mu}[g(\mathbf{x})]$. Given a finite dataset $\mathcal{T} = \{\mathbf{x}_i\}_{i=1}^N$ with samples i.i.d. from μ, we can approximate μ using the empirical distribution $\hat{\mu}_{\mathcal{T}} = \frac{1}{N}\sum_{i=1}^N \delta_{\mathbf{x}_i}$. The expectation under the push-forward of this empirical distribution is then $\mathbb{E}_{\mathbf{z}\sim g^{\#}\hat{\mu}_{\mathcal{T}}}[\mathbf{z}] = \mathbb{E}_{\mathbf{x}\sim\hat{\mu}_{\mathcal{T}}}[g(\mathbf{x})] = \frac{1}{N}\sum_{i=1}^N g(\mathbf{x}_i)$. The formulations given in Table 2 can thus be replaced with their empirical counterparts, leading to practical optimization objectives commonly used in the dataset condensation (DC) literature. Table 3 summarizes these empirical formulations using integral probability metrics.

Match \ Optimize	Input space $\mathbb{R}^n$	Latent space $\mathbb{R}^m$
Input space $\mathbb{R}^n$	$\min_{\mathcal{S}\subseteq\mathbb{R}^n} \sup_{h\in\mathcal{H}(\mathbb{R}^n)} \lvert\overline{h(\mathcal{T})} - \overline{h(\mathcal{S})}\rvert$	$\min_{\mathcal{Z}\subseteq\mathbb{R}^m} \sup_{h\in\mathcal{H}(\mathbb{R}^n)} \lvert\overline{h(\mathcal{T})} - \overline{h\circ g_d(\mathcal{Z})}\rvert$
Latent space $\mathbb{R}^m$	$\min_{\mathcal{S}\subseteq\mathbb{R}^n} \sup_{h\in\mathcal{H}(\mathbb{R}^m)} \lvert\overline{h\circ g_e(\mathcal{T})} - \overline{h\circ g_e(\mathcal{S})}\rvert$	$\min_{\mathcal{Z}\subseteq\mathbb{R}^n} \sup_{h\in\mathcal{H}(\mathbb{R}^m)} \lvert\overline{h\circ g_e(\mathcal{T})} - \overline{h(\mathcal{Z})}\rvert$

Table 3: Regimes of optimizing and matching datasets in input and latent spaces.

In particular, let $\mathcal{H}(\mathbb{R}^m)$ be an RKHS with a universal kernel $k : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$. If the encoder $g_e : \mathbb{R}^n \to \mathbb{R}^m$ is injective, then the pullback kernel $k_e := k \circ (g_e \otimes g_e)$ defines a universal kernel on $\mathbb{R}^n$ (Li et al., 2017). Consequently, matching two distributions μ_1 and μ_2 in a universal RKHS over the input space $\mathbb{R}^n$ is equivalent to matching their push-forward distributions $g_e^{\#}\mu_1$ and $g_e^{\#}\mu_2$ in the corresponding universal RKHS over the latent space $\mathbb{R}^m$. This equivalence relies on the injectivity of the encoder, which ensures that no information is lost during the transformation.

5 Taxonomy for Dataset Condensation Methods

Based on the discussions in Section 3, we categorize existing DC methods according to the discrepancy metric used and the distribution matching and optimization spaces considered. Further improvements and variants can be derived by either augmenting the dataset or incorporating regularization terms into the main loss function as discussed in Appendix A. In this section, we organize the most popular DC approaches within the theoretical framework we have established. For each category or sub-category, we first introduce the

key concept (e.g., the metric or the space), followed by a concise description of the algorithms proposed in the related works. Note that we use a consistent set of notations and terminologies throughout, which may differ from those in the original papers, to maintain alignment with the definitions introduced in the previous sections.

5.1 Discrepancy-based Classification

Given a probability distribution μ_1, our goal is to find another distribution μ_2 that is closest to μ_1 under a specified discrepancy metric $D(\mu_1, \mu_2; \mathcal{H})$, where $\mathcal{H}$ denotes a hypothesis class. The optimization problem is formulated as:

$$\min_{\mu_2} \; D(\mu_1, \mu_2; \mathcal{H}).$$

In practice, we do not have access to the true distributions μ_1 and μ_2, but instead observe finite i.i.d. samples from them. Thus, we approximate them using empirical distributions. Let $\mathcal{T}$ be a dataset sampled from μ_1 and $\mathcal{S}$ be a synthetic dataset representing μ_2. We then approximate the discrepancy using $D(\hat{\mu}_{\mathcal{T}}, \hat{\mu}_{\mathcal{S}}; \mathcal{H})$, where $\hat{\mu}_{\mathcal{T}}$ and $\hat{\mu}_{\mathcal{S}}$ denote the empirical distributions associated with $\mathcal{T}$ and $\mathcal{S}$, respectively. For convenience, we define the empirical loss of a model f on a dataset $\mathcal{T}$ as $\mathcal{L}(f, \mathcal{T}) := \mathbb{E}_{(\mathbf{x},y)\sim\hat{\mu}_{\mathcal{T}}}[l(f(\mathbf{x}), y)] = \frac{1}{N}\sum_{i=1}^{N} l(f(\mathbf{x}_i), y_i)$, where l is a task-specific loss function.

5.1.1 Generalization discrepancy

$$\boxed{\min_{\mathcal{S}} \; GD(\hat{\mu}_{\mathcal{T}}, \hat{\mu}_{\mathcal{S}}; \mathcal{H}) \Longleftrightarrow \min_{\mathcal{S}} \; \mathcal{L}(h^*_{\mathcal{S}}, \mathcal{T}), \; h^*_{\mathcal{S}} \in \min_{h} \; \mathcal{L}(h, \mathcal{S})}$$

Backpropagation-Through-Time (BPTT) (Wang et al., 2018) Given a fixed original dataset $\mathcal{T}$, this method defines an updated loss function that no longer explicitly depends on $\mathcal{T}$: $\tilde{\mathcal{L}}(h, \eta, \mathcal{S}) := \mathcal{L}(h - \eta \cdot \frac{\partial \mathcal{L}}{\partial h}\big|_{(h,\mathcal{S})}, \mathcal{T})$. This formulation allows us to differentiate through one step of model training on synthetic data $\mathcal{S}$ and evaluate performance on the original data $\mathcal{T}$. The optimization is performed iteratively as follows: At iteration k, given current parameters h^k, learning rate η^k, and synthetic dataset $\mathcal{S}^k$, update:

$$\eta^{k+1} \leftarrow \eta^k - \alpha \cdot \frac{\partial \tilde{\mathcal{L}}}{\partial \eta}\bigg|_{(h^k,\eta^k,\mathcal{S}^k)}, \; \mathcal{S}^{k+1} \leftarrow \mathcal{S}^k - \alpha \cdot \frac{\partial \tilde{\mathcal{L}}}{\partial \mathcal{S}}\bigg|_{(h^k,\eta^k,\mathcal{S}^k)}$$
$$h^{k+1} \leftarrow h^k - \eta^{k+1} \cdot \frac{\partial \mathcal{L}}{\partial h}\bigg|_{(h^k,\eta^{k+1},\mathcal{S}^{k+1})}$$

This iterative procedure decomposes the original bi-level optimization problem into two sub-problems: (1) updating the synthetic data and learning rate by differentiating through the inner optimization, and (2) updating the model using standard gradient descent on synthetic data.

Convexified Implicit Gradients (CIG) (Loo et al., 2023) This method formulates the DC objective using a bi-level optimization framework. The outer loss is defined as $\tilde{\mathcal{L}}(h^*_{\mathcal{S}}, \mathcal{S}) := \mathcal{L}(h^*_{\mathcal{S}}, \mathcal{T})$, where $h^*_{\mathcal{S}}$ is the model trained to optimality on the synthetic dataset $\mathcal{S}$, and $\mathcal{L}(\cdot, \mathcal{T})$ is the empirical loss on the true dataset $\mathcal{T}$. Rather than relying on backpropagation through the inner optimization trajectory (as in BPTT), CIG employs implicit differentiation via the chain rule on the optimality conditions. At iteration k, compute surrogate optimal model h^k (e.g., using kernel ridge regression, discussed in the next section). Then compute implicit gradient via the formula:

$$v^k = \left(\frac{\partial^2 \mathcal{L}}{\partial h \partial h^T}\right)^{-1}\bigg|_{(h^k,\mathcal{S}^k)} \cdot \left(\frac{\partial \tilde{\mathcal{L}}}{\partial h}\right)\bigg|_{(h^k,\mathcal{S}^k)}$$
$$\nabla g(\mathcal{S}^k) = \frac{\partial \tilde{\mathcal{L}}}{\partial \mathcal{S}}\bigg|_{(h^k,\mathcal{S}^k)} + \frac{\partial}{\partial \mathcal{S}}\left(\left(\frac{\partial \mathcal{L}}{\partial h}\right)^T\bigg|_{(h^k,\mathcal{S}^k)} \cdot v^k\right)$$

Finally update synthetic dataset using the gradient: $\mathcal{S}^{k+1} \leftarrow \mathcal{S}^k - \alpha \cdot \nabla g(\mathcal{S}^k)$. The key distinction from BPTT is that CIG avoids differentiating through the full training trajectory by exploiting the optimality

condition $\partial\mathcal{L}(h^*_{\mathcal{S}}, \mathcal{S})/\partial h = 0$ and applying the implicit function theorem. This leads to a computationally efficient and stable surrogate for the gradient of the outer objective with respect to $\mathcal{S}$.

5.1.2 Value discrepancy

$$\boxed{\min_{\mathcal{S}} \; VD(\hat{\mu}_{\mathcal{T}}, \hat{\mu}_{\mathcal{S}}; \mathcal{H}) = \|h^*_{\mathcal{S}} - h^*_{\mathcal{T}}\|, \; h^*_{\mathcal{S}} \in \min_{h} \; \mathcal{L}(h, \mathcal{S}), \; h^*_{\mathcal{T}} \in \min_{h} \; \mathcal{L}(h, \mathcal{T})}$$

Feature matching (Wang et al., 2022) Let $h^*_{\mathcal{S}}$ denote the model that minimizes the empirical loss over the synthetic dataset $\mathcal{S}$, i.e., $h^*_{\mathcal{S}} \in \min_h \; \mathcal{L}(h, \mathcal{S})$. The goal is to minimize the discrepancy between the feature representations of the real and synthetic datasets, as evaluated by the optimal model $h^*_{\mathcal{S}}$. Specifically, we compare the layer-wise mean features of the original dataset $\mathcal{T}$ and the synthetic dataset $\mathcal{S}$, and define the objective:

$$\mathcal{S}^* \in \arg\min_{\mathcal{S}} \sum_{l=1}^{L} \left\| \overline{(h^*_{\mathcal{S}})_l(\mathcal{T})} - \overline{(h^*_{\mathcal{S}})_l(\mathcal{S})} \right\|_2^2 + L_{dis}, \; h^*_{\mathcal{S}} \in \min_{h} \; \mathcal{L}(h, \mathcal{S}),$$

where L is the number of layers in the model h, $(h)_l(\mathcal{D})$ denotes the activation (feature) output of layer l on dataset $\mathcal{D}$, $\overline{(\cdot)}$ indicates averaging over the dataset, and $\mathcal{L}_{\text{dis}}$ is a discrimination loss that encourages class separability (as described in Appendix A.2). The above is a bi-level optimization problem, where the inner optimization finds $h^*_{\mathcal{S}}$ and the outer optimization updates $\mathcal{S}$. It is solved using backpropagation-through-time (BPTT), as discussed in Section 5.1.1.

5.1.3 Parameter discrepancy

$$\boxed{\min_{\mathcal{S}} \; PD(\hat{\mu}_{\mathcal{T}}, \hat{\mu}_{\mathcal{S}}; \mathcal{H}) = \|\theta^*_{\mathcal{S}} - \theta^*_{\mathcal{T}}\|, \; \theta^*_{\mathcal{S}} \in \min_{\theta} \; \mathcal{L}(h_\theta, \mathcal{S})}$$

Trajectory matching (Cazenavette et al., 2022) This approach optimizes a synthetic dataset so that training on it reproduces segments of the training dynamics observed on the real dataset. Let $\{\theta_{t,\mathcal{T}}\}_{t=0}^{T}$ denote a precomputed training trajectory obtained on the real dataset $\mathcal{T}$. Instead of training from random initialization, the model trained on the synthetic dataset $\mathcal{S}$ is initialized from selected checkpoints along this real trajectory. Specifically, for a chosen starting point $\theta_{t,\mathcal{T}}$, the parameters are updated for several steps using $\mathcal{S}$, yielding $\theta_{t+\tau,\mathcal{S}}$. The objective is then to match the resulting parameters to the corresponding point on the real trajectory:

$$\mathcal{S}^* \in \arg\min_{\mathcal{S}} \sum_{t=0}^{T} \sum_{\tau=1}^{K} \|\theta^*_{t+\tau,\mathcal{S}} - \theta^*_{t+\tau,\mathcal{T}}\|,$$

where $\theta_{t+\tau,\mathcal{S}}$ is obtained by training on $\mathcal{S}$ starting from $\theta_{t,\mathcal{T}}$. This formulation mitigates the ill-posedness of directly matching independently trained parameter trajectories in over-parameterized networks, where symmetries (e.g., neuron permutations) and flat regions in the loss landscape can lead to large parameter discrepancies despite similar functional behavior. The resulting bi-level optimization problem is solved using backpropagation-through-time (BPTT), as described in Section 5.1.1.

5.1.4 Characteristic Discrepancy

$$\boxed{\min_{\mathcal{S}} \; CD(\hat{\mu}_{\mathcal{T}}, \hat{\mu}_{\mathcal{S}}; \mathcal{H})}$$

Dataset distillation with neural characteristic function (Wang et al., 2025b) This approach formulates dataset condensation by minimizing a discrepancy between the empirical distributions of real and synthetic data using *characteristic functions*. Specifically, let CD denote a characteristic discrepancy defined over a function class $\mathcal{H}$ that approximates complex exponentials via neural networks. The objective becomes:

$$\mathcal{S}^* \in \arg\min_{\mathcal{S}} \; \mathbb{E}_{\omega \sim p(\omega)} \left\| \overline{h_\omega(\mathcal{T})} - \overline{h_\omega(\mathcal{S})} \right\|_2^2,$$

where $h_\omega \in \mathcal{H}$ is a neural network parameterization of complex-valued features. Unlike standard IPMs, characteristic discrepancies leverage the fact that characteristic functions uniquely determine probability distributions. By learning a flexible parameterization of these functions, the method captures higher-order statistics and global structure of the data distribution. In practice, the expectation over ω is approximated using a finite set of sampled frequencies, and the neural parameterization enables scalable and expressive matching beyond fixed kernels.

5.1.5 Distribution discrepancy

$$\boxed{\min_{\mathcal{S}} \; DD(\hat{\mu}_{\mathcal{T}}, \hat{\mu}_{\mathcal{S}}; \mathcal{H})}$$

Distribution matching (Zhao & Bilen, 2023) This approach formulates dataset condensation as minimizing a distributional discrepancy between real and synthetic datasets, measured via an IPM. Specifically, let $DD = \text{IPM}$ and consider the function space $\mathcal{H} = \mathcal{F}$ (e.g., all measurable or admissible functions). The objective becomes:

$$\mathcal{S}^* \in \arg\min_{\mathcal{S}} \sup_{h \in \mathcal{H}} \left\| \overline{h(\mathcal{T})} - \overline{h(\mathcal{S})} \right\|_2^2.$$

In practice, the supremum over $\mathcal{H}$ is approximated by evaluating the discrepancy over a batch of randomly initialized or pretrained models h sampled from the function class.

Gradient matching (Zhao et al., 2021) This method treats dataset condensation as a distribution alignment problem, using an IPM defined over gradients of the loss function. Specifically, let $DD = \text{IPM}$ and let $\mathcal{H} = \mathcal{C}^1$ be the space of continuously differentiable functions. The objective becomes:

$$\mathcal{S}^* \in \arg\min_{\mathcal{S}} \sup_{h \in \mathcal{H}} \left\| \overline{\nabla h(\mathcal{T})} - \overline{\nabla h(\mathcal{S})} \right\|_2^2.$$

Here, $\nabla h(\mathcal{T})$ and $\nabla h(\mathcal{S})$ denote the gradients of the loss with respect to model parameters, evaluated on real and synthetic datasets, respectively. The goal is to match the average gradient signal induced by the two datasets. In practice, the supremum over $\mathcal{H}$ is approximated using the gradients computed from a batch of randomly initialized or pretrained models h.

Higher-order moment matching (Yu et al., 2024) While standard distribution matching aligns only the first-order moments (means) of features extracted from the real and synthetic datasets, higher-order moment matching extends this idea by also aligning second-order moments (covariances or variances). Specifically, for a function class $\mathcal{H}$ and using an integral probability metric (IPM), the objective becomes:

$$\mathcal{S}^* \in \arg\min_{\mathcal{S}} \sup_{h \in \mathcal{H}} \left\| \overline{h(\mathcal{T})} - \overline{h(\mathcal{S})} \right\|_2^2 + \left\| Var(h(\mathcal{T})) - Var(h(\mathcal{S})) \right\|_2^2.$$

Here, $\overline{h(\cdot)}$ denotes the empirical mean of features and $\text{Var}(h(\cdot))$ denotes the feature-wise variance under function h. Interestingly, (Yu et al., 2024) show that gradient matching is a special case of this approach, which implicitly aligns both first- and second-order moments. In practice, the supremum over the function class $\mathcal{H}$ is approximated using the average over a batch of randomly initialized or pretrained models h.

M3D (Zhang et al., 2024) This method formulates dataset condensation as a distribution matching problem using the maximum mean discrepancy (MMD) as the discrepancy metric. Specifically, let $\mathcal{H}$ be an RKHS associated with a kernel k. Then the objective is:

$$\mathcal{S}^* \in \arg\min_{\mathcal{S}} \sup_{\|h\|_{\mathcal{H}} \leq 1} \left\| \overline{h(\mathcal{T})} - \overline{h(\mathcal{S})} \right\|_2^2.$$

This corresponds exactly to minimizing the MMD between the empirical distributions of $\mathcal{T}$ and $\mathcal{S}$. In practice, common choices for the kernel k include the Gaussian kernel, Laplacian kernel, and the neural tangent kernel (NTK). The MMD is then computed using the kernel embedding formulation described in Section 2, enabling efficient and differentiable optimization of the synthetic dataset.

Wasserstein distance (Liu et al., 2023a) This approach uses the Wasserstein distance as the discrepancy measure to align the empirical distributions of the original and synthetic datasets. Specifically, let $DD = WM$ be the Wasserstein metric and $\mathcal{H}$ be the set of all 1-Lipschitz functions. The optimization problem becomes:

$$\mathcal{S}^* \in \arg\min_{\mathcal{S}} \sup_{h \in \mathcal{H}} \left\| \overline{h(\mathcal{T})} - \overline{h(\mathcal{S})} \right\|_2^2.$$

By Kantorovich-Rubinstein duality, this corresponds to computing the 1-Wasserstein distance between the empirical distributions $\hat{\mu}\mathcal{T}$ and $\hat{\mu}\mathcal{S}$. In practice, the dual formulation reduces to a tractable linear program (LP) that can be efficiently solved, enabling gradient-based optimization of the synthetic dataset $\mathcal{S}$.

Minimum finite covering (Chen & Selvan, 2025) This approach uses the Hausdorff distance as the discrepancy metric between the original dataset $\mathcal{T} = (\mathbf{x}_i, y_i)_{i=1}^N$ and the synthetic dataset $\mathcal{S} = (\tilde{\mathbf{x}}_j, \tilde{y}_j)_{j=1}^M$, which requires no explicit function space $\mathcal{H}$. The condensation problem is formulated as:

$$\mathcal{S}^* \in \arg\min_{\mathcal{S}} d_H(\mathcal{T}, \mathcal{S}),$$

where the Hausdorff distance is defined by

$$d_H(\mathcal{T}, \mathcal{S}) = \max\left\{ \max_i \min_j d(\mathbf{x}_i, \tilde{\mathbf{x}}_j), \max_j \min_i d(\mathbf{x}_i, \tilde{\mathbf{x}}_j) \right\}.$$

For tasks such as coreset selection, minimizing the Hausdorff distance can be formulated as a mixed integer linear programming (MILP) problem, allowing for exact or approximate combinatorial optimization of the synthetic subset.

5.2 Space-based Classification

In Section 5.1, we discussed how to practically adapt various discrepancy metrics within the theoretical framework of dataset condensation. However, we have not yet considered variations arising from changes in the matching or optimization spaces. Recall that, in the general formulation equation DC, the outer optimization is performed over the space of distributions (i.e., the data space), while the inner optimization is over the hypothesis space (i.e., the model space). Consequently, variants of the DC methods introduced in Section 5.1 can be derived by altering either the data space or the model space.

5.2.1 Data space

For the data space, we can move from the original high-dimensional input space to a lower-dimensional latent space using generative models, as discussed in Appendix A.1. It is important to note that, in practice, an encoder mapping from the high-dimensional space to the latent space is typically not injective, and likewise, a decoder from the latent space back to the high-dimensional space is generally not surjective. As a result, the push-forward metric induced by the generative model no longer satisfies all the properties of a true metric. Nevertheless, this approach can still be effectively used to approximate the data distribution within the framework derived in Section 4.

Distribution/Gradient matching with pre-trained GANs (Zhao & Bilen, 2022) This approach synthesizes a condensed dataset by optimizing a set of latent vectors $\mathcal{Z}$, which are then passed through a pre-trained Generative Adversarial Network (GAN) generator g_d. Let $\mathcal{T}$ represent the original dataset. The optimal latent set $\mathcal{Z}^*$ found by minimizing the difference between the feature representations of the original and synthetic data, a process known as distribution or gradient matching:

$$\mathcal{S}^* = g_d(\mathcal{Z}^*),\ \mathcal{Z}^* \in \arg\min_{\mathcal{Z}} \sup_{h \in \mathcal{H}} \left\| \overline{h(\mathcal{T})} - \overline{h \circ g_d(\mathcal{Z})} \right\|_2^2.$$

Here, $\mathcal{H}$ is a function space of feature extractors. In practice, the supremum over $\mathcal{H}$ approximated by averaging the outputs from a batch of pre-trained models h.

Distribution matching with GAN training (Wang et al., 2025a) Let $\mathcal{T}$ denote the original dataset, and g_d be the generator of a GAN. The condensed dataset is generated by

$$\mathcal{S}^* = g_d^*(\mathcal{Z}),\ g_d^* \in \arg\min_{g_d} \Bigg(\underbrace{\sup_{h\in\mathcal{H}} \left\| \overline{h(\mathcal{T})} - \overline{h \circ g_d(\mathcal{Z})} \right\|_2^2}_{L_{\mathcal{Z}}} + L_{GAN} \Bigg),$$

where $\mathcal{Z}$ is a fixed set of random latent codes, and L_{GAN} is the standard min-max loss of training a conditional GAN with generator g_d and a discriminator g_c:

$$L_{GAN} = \mathbb{E}_{\mathbf{x}\sim\mu}[\log g_c(\mathbf{x}|y)] + \mathbb{E}_{\mathbf{z}\sim\nu}[\log(1 - g_c(g_d(\mathbf{z}|y)))].$$

The training proceeds by first pretraining g_d through minimizing L_{GAN} for several epochs to obtain a vanilla GAN. Then, the distribution matching loss $L_{\mathcal{Z}}$ is incorporated to further update g_d, encouraging the generator not only to produce realistic images but also to generate synthetic data that improves downstream model training performance. In practice, the supremum over the function space $\mathcal{H}$ is approximated by taking averaging over a batch of random initialized models h.

Feature matching with GAN training (Zhang et al., 2023) Let $\mathcal{T}$ be the original dataset, $\mathcal{Z}$ be the synthetic latent set, and g_d be the GAN generator. The condensed dataset is generated as

$$\mathcal{S}^* = g_d^*(\mathcal{Z}^*),\ (\mathcal{Z}^*, g_d^*) \in \arg\min_{g_d} \Bigg(\underbrace{\sum_{l=1}^{L} \left\| \overline{h_l(\mathcal{T})} - \overline{h_l \circ g_d(\mathcal{Z})} \right\|_2^2}_{L_{\mathcal{Z}}} + L_{GAN} + L_* \Bigg),$$

where h_l denotes the intermediate feature maps extracted from layer l of a neural network h, and L is the total number of such layers considered. The network h itself is updated every few epochs during the alternating updates of $\mathcal{Z}$ and g_d. The additional regularization terms collected in L_* include intra-class diversity loss L_{intra} and inter-class discrimination loss L_{inter}, as described in Appendix A.2.

Feature matching with diffusion model training (Gu et al., 2024) First, a latent diffusion model (LDM) is trained by optimizing its decoder g_d^* and denoiser g_{dn}^* to minimize the total loss:

$$L_{total} = L_{diff} + L_*,$$

where L_{diff} is the primary diffusion training loss, and L_* includes regularization terms such as the representative loss L_{rep} and diversity loss L_{div}, as discussed in Appendix A.2. After training, the condensed dataset is generated by

$$\mathcal{S}^* = (g_d^* \circ g_{dn}^*)(\mathcal{Z}),$$

where $\mathcal{Z}$ is a fixed set of random latent noise vectors.

General DC method with random or pre-trained GAN (Cazenavette et al., 2023) Let μ_1 denote the original data distribution, ν_2 be the synthetic distribution in the latent space, and g_d be the GAN generator. The synthetic distribution in data space is defined as a push-forward $\mu_2^* = g_d^{\#}\nu_2^*$, where the optimal latent distribution ν_2^* is obtained by minimizing a discrepancy metric D between μ_1 and $\mu_2 = g_d^{\#}\nu_2$, i.e.,

$$\mu_2^* = g_d^{\#}\nu_2^*,\ \nu_2^* = \arg\min_{\nu_2} D(\mu_1, g_d^{\#}\nu_2; \mathcal{H}).$$

In practice, g_d can be either randomly initialized or pretrained, and the choice of discrepancy metric D is flexible and can correspond to any dataset condensation metric discussed in previous sections.

General DC methods with pre-trained diffusion model (Moser et al., 2024) Let μ_1 be the original data distribution, ν_2 be the synthetic distribution in the latent space, and g_d, g_{dn} be the decoder and denoiser components of an LDM. The synthetic data distribution is given by the push-forward $\mu_2^* = (g_d \circ g_{dn})^{\#}\nu_2^*$, where the optimal latent distribution ν_2^* is obtained by minimizing a discrepancy metric D between μ_1 and $\mu_2 = (g_d \circ g_{dn})^{\#}\nu_2$, that is,

$$\mu_2^* = (g_d \circ g_{dn})^{\#}\nu_2^*,\ \nu_2^* = \arg\min_{\nu_2} D(\mu_1, (g_d \circ g_{dn})^{\#}\nu_2; \mathcal{H}).$$

In practice, the LDM is pre-trained, with the denoiser g_{dn} typically implemented by a U-Net architecture. The discrepancy metric D can be chosen from any dataset condensation metric introduced earlier.

General DC method with pretrained hallucinator-extractor (Liu et al., 2022) Let μ_1 denote the original data distribution, ν_2 be the synthetic distribution in latent space, and $\mathcal{H}$ a hypothesis space of hallucinators (learned data generators). The synthetic distribution is obtained by pushing forward ν_2 through a hallucinator $h \in \mathcal{H}$, i.e.,

$$\mu_2^* = (h^*)^{\#}\nu_2^*,\ (h^*, \nu_2^*) = \arg\min_{h\in\mathcal{H},\ \nu_2\in\mathcal{P}} D(\mu_1, h^{\#}\nu_2; \mathcal{H}) + L_*,$$

where $\mathcal{P}$ denotes the space of probability distributions over the latent space. The regularization term L_* includes task-specific losses (e.g., cross-entropy for classification), a contrastive loss L_{con}, and a cosine similarity loss L_{cos}, as detailed in Appendix A.2.

K-means clustering with pretrained diffusion model (Su et al., 2024) Let $\mathcal{T}$ denote the original dataset, and cl represent the operator that computes the centers of K-means clustering. We utilize a pretrained LDM with an encoder g_e^*, decoder g_d^*, and denoiser g_{dn}^*. The condensed dataset is generated as follows:

$$\mathcal{S}^* = (g_d^* \circ g_{dn}^*)(cl(g_e^*(\mathcal{T}))).$$

The process involves encoding the original dataset $\mathcal{T}$ into a low-dimensional latent space using the pretrained encoder g_e^*, applying K-means clustering to select K representative centroids in this latent space, and then reconstructing the synthetic dataset $\mathcal{S}^*$ through the pretrained decoder g_d^* and denoiser g_{dn}^*. Notably, the model is not trained on the synthetic dataset by minimizing a loss between predictions and fixed labels. Instead, it leverages soft labels from a distilled student network, optimizing the parameters θ_S^* as:

$$\theta_S^* \in \arg\min_{\theta_S} L(f_{\theta_T}(\mathbf{x}), f_{\theta_S}(\mathbf{x})),$$

where f_{θ_T} is the teacher network and f_{θ_S} is the student model trained on the synthetic dataset.

5.2.2 Model space

For model space, our focus is on the training trajectory and kernel embedding for dataset compression, driven by two key motivations. First, although model parameters reside in a high-dimensional space, the updates during training typically span a low-dimensional subspace, enabling efficient representation of the training dynamics. Second, by leveraging kernel embedding, we can transform feature representations to effectively condense the dataset, capturing essential information in a compact form.

Distribution matching with expert subspace projection (Ma et al., 2023) Let $\mathcal{T}$ represent the original dataset, $\mathcal{S}$ the synthetic dataset, and $h_{\theta^*} \in \mathcal{H}$ a teacher model with training trajectory $\Theta = \{\theta_t^*\}_{t=1}^T$, where $\mathbb{S}$ denotes the linear subspace spanned by Θ. The condensed dataset $\mathcal{S}^*$ is obtained by solving:

$$\mathcal{S}^* \in \arg\min_{\mathcal{S}} \sup_{\theta} \left\|\overline{h_\theta(\mathcal{T})} - \overline{h_\theta(\mathcal{S})}\right\|_2^2 + L_{proj}(\theta).$$

where $L_{proj}(\theta) = \|\theta - \mathrm{Proj}_{\mathbb{S}}(\theta)\|_1$ measures the ℓ_1-distance between the model parameter θ and its projection onto $\mathbb{S}$. In practice, the supremum over the function space $\mathcal{H}$ is approximated by averaging over a batch of randomly initialized or pretrained models h.

Kernel Ridge regression with NTK (Nguyen et al., 2021b;a) Given a real dataset $\mathcal{T} = (\mathbf{x}_i, y_i)_{i=1}^N$ and a synthetic dataset $\mathcal{S} = (\tilde{\mathbf{x}}_j, \tilde{y}_j)_{j=1}^M$, the solution to linear ridge regression over $\mathcal{S}$ is obtained by solving:

$$\mathbf{w}_{\mathcal{S}} \in \arg\min_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{2}\sum_{j=1}^{M} \|\mathbf{w}^T\tilde{\mathbf{x}}_j - \tilde{y}_j\|_2^2 + \frac{1}{2}\lambda\|\mathbf{w}\|_2^2. \tag{Ridge}$$

Let $\tilde{\mathbf{X}}_{\mathcal{S}} = \begin{bmatrix}\tilde{\mathbf{x}}_1 & \cdots & \tilde{\mathbf{x}}_M\end{bmatrix} \in \mathbb{R}^{n\times M}$ denote the matrix whose columns are the synthetic feature vectors $\tilde{\mathbf{x}}_j$, and let $\tilde{\mathbf{y}}_{\mathcal{S}} \in \mathbb{R}^M$ be the corresponding label vector. Then the closed-form solution of problem equation Ridge is:

$$\mathbf{w}_{\mathcal{S}} = \tilde{\mathbf{X}}_{\mathcal{S}}\big(\tilde{\mathbf{X}}_{\mathcal{S}}^T\tilde{\mathbf{X}}_{\mathcal{S}} + \lambda\mathbf{I}_M\big)^{-1}\tilde{\mathbf{y}}_{\mathcal{S}}.$$

To evaluate the quality of the synthetic dataset $\mathcal{S}$, one minimizes the predictive error of $\mathbf{w}_{\mathcal{S}}$ on the original dataset $\mathcal{T}$:

$$\mathcal{S}^* \in \arg\min_{\mathcal{S}} \sum_{i=1}^{N} \|y_i - \mathbf{w}_{\mathcal{S}}^T\mathbf{x}_i\|_2^2 = \sum_{i=1}^{N} \|y_i - \mathbf{x}_i^T\tilde{\mathbf{X}}_{\mathcal{S}}\big(\tilde{\mathbf{X}}_{\mathcal{S}}^T\tilde{\mathbf{X}}_{\mathcal{S}} + \lambda\mathbf{I}_M\big)^{-1}\tilde{\mathbf{y}}_{\mathcal{S}}\|_2^2.$$

This framework can be extended to nonlinear models via kernel methods. Let $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be a positive-definite kernel function, and define the kernel matrices: $K_{\mathcal{S},\mathcal{S}} \in \mathbb{R}^{M\times M}$ with entries $k(\tilde{\mathbf{x}}_j, \tilde{\mathbf{x}}_{j'})$, $K_{\mathcal{T},\mathcal{S}} \in \mathbb{R}^{N\times M}$ with entries $k(\mathbf{x}_i, \tilde{\mathbf{x}}_j)$. Then the kernel ridge regression predictor trained on $\mathcal{S}$ is:

$$h_{\mathcal{S}}^*(\mathbf{x}) = \sum_{j=1}^{M} \alpha_j k(\mathbf{x}, \tilde{\mathbf{x}}_j), \text{ with } \alpha = \big(K_{\mathcal{S},\mathcal{S}} + \lambda\mathbf{I}_M\big)^{-1}\tilde{\mathbf{y}}_{\mathcal{S}}$$

The predicted labels on $\mathcal{T}$ are then given by $K_{\mathcal{T},\mathcal{S}}\big(K_{\mathcal{S},\mathcal{S}} + \lambda\mathbf{I}_M\big)^{-1}\tilde{\mathbf{y}}_{\mathcal{S}}$. Hence, the optimal synthetic dataset minimizes the empirical loss on $\mathcal{T}$ under the kernel predictor:

$$\mathcal{S}^* \in \arg\min_{\mathcal{S}} \mathcal{L}(h_{\mathcal{S}}^*, \mathcal{T}),$$

where the loss is typically mean squared error.

Kernel Ridge regression with neural network Gaussian process (NNGP) kernel (Loo et al., 2022) Let $\mathcal{T}$ denote the original dataset and $\mathcal{S}$ represent the synthetic dataset. We define a kernel function k with feature maps $\varphi(\mathbf{x}) = \frac{1}{\sqrt{p}}[h_1(\mathbf{x}), \ldots, h_p(\mathbf{x})]^T$, where each h_i is drawn from a Gaussian process. The condensed dataset $\mathcal{S}^*$ is obtained by minimizing the loss function $\mathcal{L}$ on the original dataset $\mathcal{T}$ evaluated using the kernel Ridge regression model trained on $\mathcal{S}$:

$$\mathcal{S}^* \in \arg\min_{\mathcal{S}} \mathcal{L}(h_{\mathcal{S}}^*, \mathcal{T}),$$

where $h_{\mathcal{S}}^* = K_{\cdot,\mathcal{S}}\big(K_{\mathcal{S},\mathcal{S}} + \lambda\mathbf{I}_M\big)^{-1}\tilde{\mathbf{y}}_{\mathcal{S}}$ is the closed-form solution to Kernel Ridge regression with the NNGP kernel. This solution provides a compact representation of the original dataset, facilitating efficient compression and processing.

Kernel Ridge regression with Neural Feature Kernel (NFK) (Zhou et al., 2022) Let $\mathcal{T}$ denote the original dataset and $\mathcal{S}$ represent the synthetic dataset. We define a kernel function k based on a neural feature mapping, defined by $k(\mathbf{x}_1, \mathbf{x}_2) = h(\mathbf{x}_1)^T h(\mathbf{x}_2)$, where h is a randomly initialized model that maps input data to a higher-dimensional feature space. Then the condensed dataset $\mathcal{S}^*$ is obtained by minimizing the loss function $\mathcal{L}$ with respect to the synthetic dataset, which can be expressed as:

$$\mathcal{S}^* \in \arg\min_{\mathcal{S}} \mathcal{L}(h_{\mathcal{S}}^*, \mathcal{T}),$$

where $h_{\mathcal{S}}^* = K_{\cdot,\mathcal{S}}\big(K_{\mathcal{S},\mathcal{S}} + \lambda\mathbf{I}_M\big)^{-1}\tilde{\mathbf{y}}_{\mathcal{S}}$ is the closed-form solution to Kernel Ridge regression with the NFK kernel. This approach leverages the neural feature mapping to capture complex relationships between data points, enabling effective dataset compression.

5.3 Further Improvements

To further enhance the quality and effectiveness of the synthetic dataset, we explore the addition of data augmentation techniques and regularization terms to the loss function. The primary goal of these modifications is to improve the downstream performance of models trained on the synthetic dataset, as well as increase its diversity and representativeness compared to the original dataset. By incorporating these additional components into the loss function, we aim to create a more robust and generalizable synthetic dataset that can better support a wide range of downstream applications.

5.3.1 Data augmentation

The data augmentation techniques discussed in this section are detailed in Appendix A.3. By augmenting both the original dataset $\mathcal{T}$ and the synthetic dataset $\mathcal{S}$, we can effectively expand the data distribution, thereby enhancing the expressiveness and downstream performance of models trained on the synthetic data. This expansion enables the model to capture a wider range of patterns and relationships, ultimately leading to improved generalization and robustness in downstream applications.

Distribution matching with Spatial Attention Matching (SAM) (Sajedi et al., 2023) We define an attention map $\mathcal{A}: \mathbb{R}^{c\times n} \rightarrow \mathbb{R}^{n}$ as $\mathcal{A}(\mathbf{x}) = \sum_{i=1}^{c} \|\mathbf{x}_{i,:}\|_p^p$, which operates on input tensors $\mathbf{x}$ of shape $c \times n$. The synthetic dataset $\mathcal{S}^*$ is obtained by solving the following minimax optimization problem:

$$\mathcal{S}^* \in \arg\min_{\mathcal{S}} \sup_{h\in\mathcal{H}} \sum_{l=1}^{L} \left\|\overline{\mathcal{A}(h_l(\mathcal{T}))} - \overline{\mathcal{A}(h_l(\mathcal{S}))}\right\|_2^2,$$

where h_l denotes the intermediate feature maps of a neural network h. In practice, we approximate the supremum over the function space $\mathcal{H}$ by averaging the results from a batch of randomly initialized or pre-trained models h.

Distribution matching with multi-formation and model enrichment (Zhao et al., 2023) Let $\mathcal{A}_r$ denote a multi-formation operator that transforms the synthetic dataset. The optimal synthetic dataset $\mathcal{S}^*$ is obtained by minimizing the maximum discrepancy between the expected values of a set of functions applied to the original dataset $\mathcal{T}$ and the transformed synthetic dataset $\mathcal{A}(\mathcal{S})$. This can be formulated as:

$$\mathcal{S}^* \in \arg\min_{\mathcal{S}} \sup_{h\in\mathcal{H}} \left\|\overline{h(\mathcal{T})} - \overline{h(\mathcal{A}(\mathcal{S}))}\right\|_2^2.$$

In practice, we approximate the supremum over the function space $\mathcal{H}$ by averaging the outputs of a batch of models, including both randomly initialized and pre-trained models. To further enrich the function space, we also incorporate models trained at intermediate epochs, providing a more comprehensive and diverse set of functions to match the distributions.

Gradient matching with data augmentation (Zhao & Bilen, 2021) Let $\mathcal{A}$ be a differentiable siamese augmentation operator, which applies the same differentiable augmentation to both the original dataset $\mathcal{T}$ and the synthetic dataset $\mathcal{S}$. The optimal synthetic dataset $\mathcal{S}^*$ is obtained by minimizing the maximum discrepancy between the expected gradients of a set of functions applied to the augmented original and synthetic datasets. This can be formulated as:

$$\mathcal{S}^* \in \arg\min_{\mathcal{S}} \sup_{h\in\mathcal{H}} \left\|\overline{\nabla h(\mathcal{A}(\mathcal{T}))} - \overline{\nabla h(\mathcal{A}(\mathcal{S}))}\right\|_2^2.$$

In practice, we approximate the supremum over the function space $\mathcal{H}$ by averaging the gradients of a batch of models, including both randomly initialized and pre-trained models h. By matching the gradients of the models on the augmented datasets, we aim to capture the underlying patterns and relationships in the data, thereby improving the quality of the synthetic dataset.

Gradient matching with multi-formation (Kim et al., 2022) Let $\mathcal{A}_r$ denote a multi-formation operator as described in Appendix A.3, which applies a set of transformations to the original and synthetic datasets. The optimal synthetic dataset $\mathcal{S}^*$ is obtained by minimizing the maximum discrepancy between the expected gradients of a set of functions applied to the transformed original and synthetic datasets. This can be formulated as:

$$\mathcal{S}^* \in \arg\min_{\mathcal{S}} \sup_{h \in \mathcal{H}} \left\| \overline{\nabla h(\mathcal{A}(\mathcal{T}))} - \overline{\nabla h(\mathcal{A}(\mathcal{S}))} \right\|_2^2.$$

In practice, we approximate the supremum over the function space $\mathcal{H}$ by averaging the gradients of a batch of models, including both randomly initialized and pre-trained models h. By leveraging the multi-formation operator to generate diverse transformations of the data, we aim to improve the robustness and generalizability of the synthetic dataset.

Gradient matching with channel-wise multi-formation (Zhou et al., 2024) Let $\mathcal{A}_c$ denote a channel-wise multi-formation operator as described in Appendix A.3, which applies a set of transformations to the original and synthetic datasets in a channel-wise manner. The optimal synthetic dataset $\mathcal{S}^*$ is obtained by minimizing the maximum discrepancy between the expected gradients of a set of functions applied to the transformed original and synthetic datasets. This can be formulated as:

$$\mathcal{S}^* \in \arg\min_{\mathcal{S}} \sup_{h \in \mathcal{H}} \left\| \overline{\nabla h(\mathcal{A}(\mathcal{T}))} - \overline{\nabla h(\mathcal{A}(\mathcal{S}))} \right\|_2^2.$$

In practice, we approximate the supremum over the function space $\mathcal{H}$ by averaging the gradients of a batch of models, including both randomly initialized and pre-trained models h. By leveraging the channel-wise multi-formation operator to generate diverse transformations of the data, we aim to capture the complex patterns and relationships in the data, thereby improving the quality and robustness of the synthetic dataset.

Trajectory matching with neural spectral decomposition (Yang et al., 2024) Given the collection of spectral tensors $\{\mathbf{T}_i\}$ and kernel tensors $\{\mathbf{K}_j\}$ as described in Appendix A.1.5, we construct a synthetic dataset $\mathcal{S} = \{\mathbf{T}_i \otimes \mathbf{K}_j\}$. To obtain the optimal synthetic dataset $\mathcal{S}^*$, we solve a bi-level optimization problem, which can be formulated as:

$$\mathcal{S}^* \in \min_{\mathcal{S}} \ \|\theta^*_{\mathcal{S}} - \theta^*_{\mathcal{T}}\|, \ \theta^*_{\mathcal{S}} \in \min_{\theta} \ \mathcal{L}(h_\theta, \mathcal{S}).$$

This bi-level optimization problem is efficiently solved using BPTT, as detailed in Section 5.1.1.

5.3.2 Loss improvement and regularization

To further enhance the quality of the synthetic dataset, we can modify the loss objective to incorporate intermediate features of the network, in addition to the output feature. By doing so, we can capture a more comprehensive range of underlying features present in the original dataset, leading to improved loss performance and regularization. This multi-feature loss objective enables the synthetic dataset to better preserve the underlying structure and relationships present in the original data, resulting in a more accurate and informative representation.

Gradient matching with contrastive signals (Lee et al., 2022) The original gradient matching formulation involves computing the distance between the mean gradients of samples from each distinct class and then summing up these distances across classes. This can be expressed as:

$$\mathcal{S}^* \in \arg\min_{\mathcal{S}} \sup_{h \in \mathcal{H}} \sum_{y=1}^{C} \left\| \overline{\nabla h(\mathcal{T}^y)} - \overline{\nabla h(\mathcal{S}^y)} \right\|_2^2,$$

where $\mathcal{T}^y$ and $\mathcal{S}^y$ denote the subsets of the original and synthetic datasets, respectively, belonging to class y. In practice, the supremum over the function space $\mathcal{H}$ is approximated by iteratively updating the model h after certain iterations of updating the synthetic dataset $\mathcal{S}$.

To improve upon this formulation, (Lee et al., 2022) propose a modified version of gradient matching that leverages contrastive signals by summing up the mean gradients across classes before computing the difference

between the original and synthetic datasets. This can be expressed as:

$$\mathcal{S}^* \in \arg\min_{\mathcal{S}} \left\| \sum_{y=1}^{C} \overline{\nabla h^*_{\mathcal{S}}(\mathcal{T}^y)} - \sum_{y=1}^{C} \overline{\nabla h^*_{\mathcal{S}}(\mathcal{S}^y)} \right\|_2^2, \; h^*_{\mathcal{S}} \in \min_{h \in \mathcal{H}} \mathcal{L}(h, \mathcal{S}).$$

The key difference between these two formulations lies in the order of operations: the original formulation computes the difference between mean gradients for each class separately, whereas the improved formulation sums up the mean gradients across classes before computing the difference.

Gradient matching with loss curvature regularization (Shin et al., 2023) To incorporate loss-curvature information, we define the Hessian matrix $\mathbf{H}_{\mathcal{T}}(h)$ of the loss function $\mathcal{L}(h, \mathcal{T})$ with respect to the model $h \in \mathcal{H}$. Let $\lambda^{+}_{\mathcal{T},\mathcal{S}}(h)$ denote the maximum eigenvalue of the matrix $\mathbf{H}_{\mathcal{T}}(h) - \mathbf{H}_{\mathcal{S}}(h)$, which represents the difference in loss curvature between the original dataset $\mathcal{T}$ and the synthetic dataset $\mathcal{S}$. The synthetic dataset $\mathcal{S}^*$ is then obtained by solving the following optimization problem:

$$\mathcal{S}^* \in \arg\min_{\mathcal{S}} \left(\sup_{h \in \mathcal{H}} \left\| \overline{\nabla h(\mathcal{T})} - \overline{\nabla h(\mathcal{S})} \right\|_2^2 + \frac{1}{2} \rho \lambda^{+}_{\mathcal{T},\mathcal{S}}(h) \right),$$

where ρ is a hyperparameter that controls the strength of the loss-curvature regularization. In practice, the supremum over the function space $\mathcal{H}$ is approximated by iteratively updating the model h after certain iterations of updating the synthetic dataset $\mathcal{S}$.

5.3.3 Efficiency improvement

When dealing with large and computationally intractable datasets, a two-stage approach can be employed: first, a rough compression can be achieved using naive and greedy clustering methods, such as K-means, to reduce the dataset size; then, advanced DC methods can be applied to further refine and condense the dataset, resulting in a more compact and representative subset of the original data.

BPTT with randomized truncation (Feng et al., 2024) The standard BPTT algorithm equation DC requires iterative updates of the model (inner-loop) and the synthetic dataset (outer-loop), resulting in a substantial computational burden. To mitigate this issue, the Truncated BPTT (T-BPTT) method (Puskorius & Feldkamp, 1994; Williams & Peng, 1990) truncates the outer-loop iteration steps, significantly improving the efficiency of BPTT. Furthermore, the Randomized Truncated BPTT (RaT-BPTT) method (Feng et al., 2024) introduces an additional layer of randomness to the truncation process, allowing for more flexible and efficient optimization. By incorporating randomness into the truncation, RaT-BPTT can better navigate the complex optimization landscape and converge to a more accurate solution.

Gradient matching with K-means clustering (Liu et al., 2023b;c) To reduce the computational complexity of DC methods, we apply a clustering operator cl to the original dataset $\mathcal{T}$ every T steps, resulting in a subset of $\mathcal{T}$ with a fixed size. This subset serves as a proxy for $\mathcal{T}$, allowing us to efficiently conduct DC methods. The synthetic dataset $\mathcal{S}^*$ is then obtained by solving the following optimization problem:

$$\mathcal{S}^* \in \arg\min_{\mathcal{S}} \sup_{h \in \mathcal{H}} \left\| \overline{\nabla h(cl(\mathcal{T}))} - \overline{\nabla h(\mathcal{S})} \right\|_2^2.$$

In practice, we approximate the supremum over the function space $\mathcal{H}$ by iteratively updating the model h after a certain number of iterations of updating the synthetic dataset $\mathcal{S}$. This approach enables efficient gradient matching with a reduced computational burden, while still maintaining the accuracy of the DC methods.

Using the notions of discrepancy, spaces and additional tricks presented, we present a high-level taxonomy of eminent DC methods in Table 4.

[1]While NTK/NNGP-based methods can be interpreted as shaping the outputs of the kernel-induced predictor, their primary contribution is not to match features or outputs explicitly. Instead, they reformulate the dataset condensation objective in a linearized regime where training admits a closed-form solution, thereby converting the original bi-level optimization into a single-level problem.

Table 4: Overview of the taxonomy for dataset condensation methods. Legend for the symbols in Tricks column: (data†, model‡, augmentation*, regularization¶, efficiency§, multi-objective$^{\intercal}$).

Discrepancy	Description	Tricks	Method
GD	BPTT for gradients	N/A	Wang et al. (2018)
	BPTT for gradients	randomized truncation§	Feng et al. (2024)
	CIG for gradients	N/A	Loo et al. (2023)
	N/A	robustness$^{\intercal}$	Wu et al. (2022)
	N/A	curvature regularization¶, robustness$^{\intercal}$	Xue et al. (2025)
VD	intermediate feature matching	with discrimination loss¶	Wang et al. (2022)
		GAN†, intra-class diversity¶, inter-class discrimination¶	Zhang et al. (2023)
		diffusion model training†, representative loss¶, diversity loss¶	Gu et al. (2024)
	linearized training objective [1]	Kernel Ridge regression with NTK‡	Nguyen et al. (2021a;b)
		Kernel Ridge regression with NNGP‡	Loo et al. (2022)
		Kernel Ridge regression with NFK‡	Zhou et al. (2022)
		Kernel Ridge regression with NTK‡, robustness$^{\intercal}$	Tsilivis et al. (2022)
PD	trajectory matching	N/A	Cazenavette et al. (2022)
		neural spectral decomposition*	Yang et al. (2024)
CD	neural characteristic function	N/A	(Wang et al., 2025b)
DD	distribution matching (IPM)	N/A	Zhao & Bilen (2023)
		pre-trained GANs†	Zhao & Bilen (2022)
		GAN training†	Wang et al. (2025a)
		expert subspace projection‡	Ma et al. (2023)
		spatial attention matching*	Sajedi et al. (2023)
		multi-formation*	Zhao et al. (2023)
	gradient matching (IPM)	N/A	Zhao et al. (2021)
		siamese augmentation*	Zhao & Bilen (2021)
		multi-formation*	Kim et al. (2022)
		channel-wise multi-formation*	Zhou et al. (2024)
		contrastive signal¶	Lee et al. (2022)
		loss curvature regularization¶	Shin et al. (2023)
		private set generation$^{\intercal}$	Chen et al. (2022)
		K-means clustering§	Liu et al. (2023b;c)
	higher-order moment matching (IPM)	N/A	Yu et al. (2024)
	mean feature matching (IPM)	K-means clustering§	Su et al. (2024)
	MMD with NTK	N/A	Zhang et al. (2024)
		differential privacy$^{\intercal}$	Harder et al. (2021)
	Wasserstein metric	N/A	Liu et al. (2023a)
	Hausdorff distance	N/A	Chen & Selvan (2025)
General	N/A	pre-trained GANs†	Cazenavette et al. (2023)
		pre-trained diffusion models†	Moser et al. (2024)
		pre-trained hallucinators†	Liu et al. (2022)

6 Multi-Objective-Aware Dataset Compression

The primary objective of DC is to distill large datasets into smaller, more efficient representations while preserving downstream performance, i.e., accuracy. By doing so, DC aims to address the long-standing challenge of jointly optimizing efficiency and accuracy in model training. However, in recent years, it has become increasingly evident that accuracy and efficiency are no longer sufficient metrics for evaluating the quality of a dataset. Concerns have been raised regarding the social aspects of models, such as privacy and robustness, and how they are affected when training models on condensed synthetic datasets. Specifically, it remains unclear how DC impacts the privacy and robustness of models, and whether it is possible to jointly improve accuracy, privacy, and robustness through DC. This poses a significant challenge and open question in the field, highlighting the need for further research into the broader implications of DC on model training and deployment.

6.1 Privacy-aware compression

In this context, we focus on differential privacy (DP) as the primary notion of privacy. Specifically, a model $h \in \mathcal{H}$ is considered (ε, δ)-differentially private if it satisfies the following condition: for any possible set of model outputs Λ and any pair of datasets $\mathcal{T}$ and $\mathcal{T}'$ that differ by only one element, the probability of $h(\mathcal{T})$ being in Λ is bounded by $e^{\varepsilon} \cdot \mathbf{Pr}[h(\mathcal{T}') \in \Lambda] + \delta$. This can be formally expressed as:

$$\mathbf{Pr}[h(\mathcal{T}) \in \Lambda] \leq e^{\varepsilon} \cdot \mathbf{Pr}[h(\mathcal{T}') \in \Lambda] + \delta.$$

By imposing DP constraints on the model, we can establish an upper bound on the amount of information that can be leaked by any individual sample in the dataset. The primary objective of DP-aware dataset condensation is to embed DP into the synthetic datasets, resulting in models that possess enhanced DP properties. It is worth noting that other notions of privacy, such as membership inference privacy (MIP) (Shokri et al., 2017), have also been explored. For instance, (Dong et al., 2022) have demonstrated that synthetic datasets obtained via distribution matching can provide a certain level of MIP. However, the development of DP-aware dataset condensation methods remains a crucial area of research, as it has the potential to provide strong privacy guarantees for models trained on synthetic datasets.

Distribution matching using MMD for DP (Harder et al., 2021) To achieve differential privacy (DP) in distribution matching, we add random Gaussian noise to the original dataset to induce DP guarantees. The privacy-aware synthetic dataset $\mathcal{S}^*$ is obtained by solving the following optimization problem:

$$\mathcal{S}^* \in \arg\min_{\mathcal{S}} \sup_{\|h\|_{\mathcal{H}} \leq 1} \left\| \overline{h(\mathcal{T} + \varepsilon)} - \overline{h(\mathcal{S})} \right\|_2^2,$$

where ε is a random Gaussian noise vector drawn from a normal distribution $\mathcal{N}(0, \Sigma)$, and h is a feature map chosen to be random Fourier features. By adding noise to the original dataset, we can ensure that the synthetic dataset $\mathcal{S}^*$ satisfies the DP guarantees, while the use of random Fourier features as the feature map enables an efficient and effective distribution matching process. This approach provides a promising way to generate synthetic datasets that balance data utility and privacy.

Gradient matching for private set generation (Chen et al., 2022) To generate private synthetic datasets, we add random random Gaussian noise to the average gradients computed over the original dataset, which induces DP guarantees. The privacy-aware synthetic dataset $\mathcal{S}^*$ is obtained by solving the following optimization problem:

$$\mathcal{S}^* \in \arg\min_{\mathcal{S}} \sup_{h \in \mathcal{H}} \left\| \left(\overline{\nabla h(\mathcal{T})} + \epsilon \right) - \overline{\nabla h(\mathcal{S})} \right\|_2^2,$$

where ε is a random Gaussian noise vector drawn from a normal distribution $\mathcal{N}(0, \Sigma)$. In practice, the supremum over the function space $\mathcal{H}$ is approximated by iteratively updating the model h after a certain number of iterations of updating the synthetic dataset $\mathcal{S}$. By adding noise to the average gradients, this approach ensures that the synthetic dataset $\mathcal{S}^*$ satisfies the DP guarantees, while the gradient matching process enables the generation of high-quality synthetic datasets that preserve the underlying patterns and relationships of the original data.

6.2 Robustness-aware compression

In this context, we focus on adversarial robustness as the primary notion of robustness. Given a perturbation radius $\varepsilon \geq 0$, a norm $\|\cdot\|$, and labeled data $(\mathbf{x}, y)$ sampled from a probability distribution μ, the ε-adversarial robustness of a model $h \in \mathcal{H}$ is quantified by the probability that the model correctly classifies the input data even when it is perturbed by an adversarial attack within the specified radius: $\mathbf{Pr}[h(\mathbf{x}+\delta) = y,\ \forall \|\delta\| \leq \varepsilon]$. A common approach to improving the adversarial robustness of a model is to minimize the adversarial loss function:

$$\mathcal{L}^{adv}(h, \mu; \varepsilon) := \mathbb{E}_{(\mathbf{x},y)\sim\mu}\left[\max_{\|\delta\|\leq\varepsilon} l(h(\mathbf{x}+\delta), y)\right].$$

Recent work by (Ilyas et al., 2019) has shown that it is possible to extract robust and non-robust features from a dataset, such that standard model training over robust (or non-robust) features leads to robust (or non-robust) models. This raises an intriguing question: *Can we further enhance the condensed dataset by embedding adversarial robustness into it*? In other words, can we design a dataset condensation method that not only preserves the key features of the original dataset but also improves the robustness of models trained on the condensed dataset?

Adversarially robust condensation by bi-level optimization (Wu et al., 2022) To obtain a synthetic dataset with robustness guarantees, similar to formulation equation DC for dataset condensation, we can employ the following bi-level optimization objective:

$$\mu_2^* \in \inf_{\mu_2}\ \mathcal{L}^{adv}(h^*_{\mu_2}, \mu_1; \varepsilon),\ \text{s.t.}\ h^*_{\mu_2} \in \arg\min_{h}\ \mathcal{L}(h, \mu_2), \tag{RobDC}$$

where μ_1 represents the original distribution and μ_2 denotes the synthetic distribution. The formulation in equation RobDC reveals that the model trained on the optimized synthetic distribution μ_2^* achieves optimal robustness performance on the original dataset. By solving this bi-level optimization problem, we can generate a synthetic dataset that not only preserves the key characteristics of the original data but also enhances the robustness of models trained on it. This approach provides a promising way to improve the adversarial robustness of machine learning models while maintaining their performance on the original task.

Adversarially robust condensation by kernel Ridge regression (Tsilivis et al., 2022) Let $h^*_{\mathcal{S}} = K_{\cdot,\mathcal{S}}\big(K_{\mathcal{S},\mathcal{S}} + \lambda \mathbf{I}_M\big)^{-1}\tilde{\mathbf{y}}_{\mathcal{S}}$ denote the closed-form solution of kernel Ridge regression trained on the synthetic dataset $\mathcal{S}$. The method jointly optimizes the synthetic dataset $\mathcal{S}$ and an adversarial perturbation δ applied to the original dataset $\mathcal{T}$:

$$(\mathcal{S}^*, \delta^*) \in \arg\min_{\mathcal{S}} \max_{\delta} \mathcal{L}(h^*_{\mathcal{S}}, \mathcal{T} + \delta). \tag{RidgeDC}$$

This formulation defines a two-player game: the adversarial perturbation δ aims to degrade the performance of the model $h^*_{\mathcal{S}}$, while the synthetic dataset $\mathcal{S}$ is optimized to ensure robustness and strong generalization performance of $h^*_{\mathcal{S}}$ even on the perturbed dataset. Compared to the adversarial objective in (Wu et al., 2022), which applies perturbations at the level of individual data points and maximizes each per-instance loss before averaging, this method performs the maximization over the regularized loss on the entire perturbed dataset. This shift from pointwise to dataset-level robustness yields a more global form of adversarial resistance during condensation.

Adversarially robust condensation by curvature regularization (Xue et al., 2025) Motivated by the established link between adversarial robustness and model smoothness, which is quantified through curvature and the Lipschitz constant, this method augments the standard dataset condensation objective with a curvature regularization term that encourages locally flat loss landscapes:

$$\mu_2^* \in \inf_{\mu_2}\ \mathcal{L}(h^*_{\mu_2}, \mu_1) + \lambda L_*,\ \text{s.t.}\ h^*_{\mu_2} \in \arg\min_{h}\ \mathcal{L}(h, \mu_2), \tag{CurvDC}$$

Here, μ_1 is the original data distribution, μ_2 is the learned synthetic data distribution, and L_* is a regularization term estimating the largest eigenvalue of the Hessian of the loss function l, serving as a proxy for

curvature. By training models on the synthetic distribution μ_2 while explicitly penalizing high curvature, this approach biases learning toward flatter minima, i.e., regions of the loss surface with low sensitivity to input perturbations. Consequently, the resulting models exhibit reduced Lipschitz constants and enhanced adversarial robustness, even though they are trained on a compressed synthetic dataset.

7 Conclusion and Discussion

In this work, we introduced a unified theoretical framework for DC, aiming to provide a principled foundation for a field that has so far developed in a largely heuristic and fragmented manner. By formalizing DC through the lens of distribution discrepancies and data/hypothesis spaces, we established a general definition that situates existing methods within a common structure. This allowed us to categorize the landscape of DC approaches along several key dimensions: the discrepancy metric employed, the space in which distributions are matched and optimized, the parameterization of models, and the choice of loss functions and regularization.

Our framework not only clarifies the assumptions and limitations of current approaches but also highlights how DC can be extended beyond its conventional goal of preserving generalization accuracy. In particular, we discussed how DC can be viewed as a multi-objective problem, where efficiency, robustness, and privacy become central considerations alongside accuracy. This perspective opens the door to more holistic objectives in the design of synthetic datasets and learning algorithms.

Looking forward, the theoretical contributions presented here call for further empirical investigation. A systematic benchmark grounded in our framework would provide a much-needed basis for comparing methods on equal footing, assessing their strengths and weaknesses across multiple objectives, and guiding the development of new algorithms. Establishing such a benchmark represents an important next step toward bridging theory and practice in dataset condensation, ultimately advancing the reliability and applicability of DC in real-world settings.

References

Lasse F Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. *arXiv preprint arXiv:2007.03051*, 2020.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pp. 214–223. PMLR, 2017.

James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2(3):8, 2023.

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.

Miguel A Carreira-Perpinán. Model compression as constrained optimization, with application to neural nets. part i: General framework. *arXiv preprint arXiv:1707.01209*, 2017.

Miguel A Carreira-Perpinán and Yerlan Idelbayev. Model compression as constrained optimization, with application to neural nets. part ii: Quantization. *arXiv preprint arXiv:1707.04319*, 2017.

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.

George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. Generalizing dataset distillation via deep generative prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3739–3748, 2023.

Dingfan Chen, Raouf Kerkouche, and Mario Fritz. Private set generation with discriminative information. *Advances in Neural Information Processing Systems*, 35:14678–14690, 2022.

Tong Chen and Raghavendra Selvan. Is adversarial training with compressed datasets effective? In *Image Analysis*, pp. 88–103. Springer Nature Switzerland, 2025. ISBN 978-3-031-95911-0.

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

Tian Dong, Bo Zhao, and Lingjuan Lyu. Privacy for free: How does dataset condensation help privacy? In *International Conference on Machine Learning*, pp. 5378–5396. PMLR, 2022.

R. M. Dudley. *Real Analysis and Probability*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2 edition, 2002.

Yunzhen Feng, Shanmukha Ramakrishna Vedantam, and Julia Kempe. Embarrassingly simple dataset distillation. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=PLoWVP7Mjc`.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Jianyang Gu, Saeed Vahidian, Vyacheslav Kungurtsev, Haonan Wang, Wei Jiang, Yang You, and Yiran Chen. Efficient dataset distillation via minimax diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15793–15803, 2024.

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.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*, 2017.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

I Hubara, M Courbariaux, D Soudry, R El-Yaniv, and Y Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. arxiv. *arXiv preprint arXiv:1609.07061*, 2016.

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *Advances in neural information processing systems*, 32, 2019.

Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014.

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.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8110–8119, 2020.

Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *Advances in neural information processing systems*, 34:852–863, 2021.

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 *International Conference on Machine Learning*, pp. 11102–11118. PMLR, 2022.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

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

Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. Mmd gan: Towards deeper understanding of moment matching network. *Advances in neural information processing systems*, 30, 2017.

Haoyang Liu, Tiancheng Xing, Luwei Li, Vibhu Dalal, Jingrui He, and Haohan Wang. Dataset distillation via the wasserstein metric. *arXiv preprint arXiv:2311.18531*, 2023a.

Songhua Liu, Kai Wang, Xingyi Yang, Jingwen Ye, and Xinchao Wang. Dataset distillation via factorization. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=luGXvawYWJ`.

Yanqing Liu, Jianyang Gu, Kai Wang, Zheng Zhu, Wei Jiang, and Yang You. Dream: Efficient dataset distillation by representative matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17314–17324, 2023b.

Yanqing Liu, Jianyang Gu, Kai Wang, Zheng Zhu, Kaipeng Zhang, Wei Jiang, and Yang You. Dream+: Efficient dataset distillation by bidirectional representative matching. *arXiv preprint arXiv:2310.15052*, 2023c.

Noel Loo, Ramin Hasani, Alexander Amini, and Daniela Rus. Efficient dataset distillation using random feature approximation. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=h8Bd7Gm3muB`.

Noel Loo, Ramin Hasani, Mathias Lechner, and Daniela Rus. Dataset distillation with convexified implicit gradients. In *International Conference on Machine Learning*, pp. 22649–22674. PMLR, 2023.

Zhiheng Ma, Dezheng Gao, Shaolei Yang, Xing Wei, and Yihong Gong. Dataset condensation via expert subspace projection. *Sensors*, 23(19):8148, 2023.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016.

Brian B Moser, Federico Raue, Sebastian Palacio, Stanislav Frolov, and Andreas Dengel. Latent dataset distillation with diffusion models. *arXiv preprint arXiv:2403.03881*, 2024.

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*, 34:5186–5198, 2021b.

David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

GV Puskorius and LA Feldkamp. Truncated backpropagation through time and kalman filter training for neurocontrol. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 4, pp. 2488–2493. IEEE, 1994.

Tian Qin, Zhiwei Deng, and David Alvarez-Melis. A label is worth a thousand images in dataset distillation. *Advances in Neural Information Processing Systems*, 37:131946–131971, 2024.

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.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.

Ahmad Sajedi, Samir Khaki, Ehsan Amjadian, Lucy Z Liu, Yuri A Lawryshyn, and Konstantinos N Plataniotis. Datadam: Efficient dataset distillation with attention matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17097–17107, 2023.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4570–4580, 2019.

Seungjae Shin, Heesun Bae, Donghyeok Shin, Weonyoung Joo, and Il-Chul Moon. Loss-curvature matching for dataset selection and condensation. In *International Conference on Artificial Intelligence and Statistics*, pp. 8606–8628. PMLR, 2023.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pp. 3–18. IEEE, 2017.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1): 1929–1958, 2014.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for modern deep learning research. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(09):13693–13696, Apr. 2020. doi: 10.1609/aaai.v34i09.7123. URL `https://ojs.aaai.org/index.php/AAAI/article/view/7123`.

Duo Su, Junjie Hou, Weizhi Gao, Yingjie Tian, and Bowen Tang. D^4: Dataset distillation via disentangled diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5809–5818, 2024.

Nikolaos Tsilivis, Jingtong Su, and Julia Kempe. Can we achieve robustness from data alone? *arXiv preprint arXiv:2207.11727*, 2022.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

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.

Kai Wang, Jianyang Gu, Hansong Zhang, Daquan Zhou, Zheng Zhu, Wei Jiang, and Yang You. Dim: Distilling dataset into generative model. In *European Conference on Computer Vision*, pp. 42–59. Springer, 2025a.

Shaobo Wang, Yicun Yang, Zhiyuan Liu, Chenghao Sun, Xuming Hu, Conghui He, and Linfeng Zhang. Dataset distillation with neural characteristic function: A minmax perspective. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 25570–25580, 2025b.

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

Ronald J Williams and Jing Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural computation*, 2(4):490–501, 1990.

Yihan Wu, Xinda Li, Florian Kerschbaum, Heng Huang, and Hongyang Zhang. Towards robust dataset learning. *arXiv preprint arXiv:2211.10752*, 2022.

Eric Xue, Yijiang Li, Haoyang Liu, Peiran Wang, Yifan Shen, and Haohan Wang. Towards adversarially robust dataset distillation by curvature regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 9041–9049, 2025.

Shaolei Yang, Shen Cheng, Mingbo Hong, Haoqiang Fan, Xing Wei, and Shuaicheng Liu. Neural spectral decomposition for dataset distillation. In *European Conference on Computer Vision*, pp. 275–290. Springer, 2024.

Ruonan Yu, Songhua Liu, Jingwen Ye, and Xinchao Wang. Teddy: Efficient large-scale dataset distillation via taylor-approximated matching. In *European Conference on Computer Vision*, pp. 1–17. Springer, 2024.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

David Junhao Zhang, Heng Wang, Chuhui Xue, Rui Yan, Wenqing Zhang, Song Bai, and Mike Zheng Shou. Dataset condensation via generative model. *arXiv preprint arXiv:2309.07698*, 2023.

Hansong Zhang, Shikun Li, Pengju Wang, Dan Zeng, and Shiming Ge. M3d: Dataset condensation by minimizing maximum mean discrepancy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 9314–9322, 2024.

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. In *NeurIPS 2022 Workshop on Synthetic Data for Empowering ML Research*, 2022. URL `https://openreview.net/forum?id=frAvOjtUMfS`.

Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 6514–6523, 2023.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=mSAKhLYLSsl`.

Ganlong Zhao, Guanbin Li, Yipeng Qin, and Yizhou Yu. Improved distribution matching for dataset condensation. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7856–7865, 2023.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infovae: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*, 2017.

Wenbo Zhou, Guoqing Zheng, and Xinghao Ding. Dataset distillation with channel efficient process. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5505–5509. IEEE, 2024.

Yongchao Zhou, Ehsan Nezhadarya, and Jimmy Ba. Dataset distillation using neural feature regression. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=2clwrA2tfik`.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

A Dataset Condensation in Practice

In Section 3, we introduced a theoretical framework for dataset condensation, which consists of two main steps: matching distributions and optimizing synthetic datasets. Both steps can be performed either in the original high-dimensional input space or within a lower-dimensional latent space. In this section, we explore practical strategies for constructing such latent spaces, as well as technical techniques for enhancing downstream performance and improving the quality of the synthetic data.

A.1 Generative Modeling

A generative model typically consists of two components: an encoder $g_e : \mathbb{R}^n \to \mathbb{R}^m$ and a decoder $g_d : \mathbb{R}^m \to \mathbb{R}^n$, where the input dimension n is significantly larger than the latent dimension m, i.e., $n \gg m$. Given a data distribution μ over $\mathbb{R}^n$, the goal of the generative model is to learn a structured latent representation of the data such that we can sample a latent vector $\mathbf{z} \in \mathbb{R}^m$ from the latent space and reconstruct a data point $\mathbf{x} = g_d(\mathbf{z}) \in \mathbb{R}^n$ that approximately lies in the support of μ.

A.1.1 Variational Autoencoder (VAE) (Kingma & Welling, 2013)

A Variational Autoencoder (VAE) consists of an encoder g_e and a decoder g_d. The encoder maps input data to a latent distribution, typically matching it to a prior such as a multivariate Gaussian. The decoder reconstructs data samples by mapping latent variables drawn from this prior back to the original data space. This framework enables both representation learning and generative modeling under a probabilistic formulation. Several variants of the VAE have been developed to enhance performance or address specific limitations, such as β-VAE (for disentangled representations) (Higgins et al., 2017), Conditional VAE (CVAE, for supervised generation) (Sohn et al., 2015), InfoVAE (Zhao et al., 2017), and Vector Quantized VAE (VQ-VAE) (Van Den Oord et al., 2017).

A.1.2 Generative Adversarial Network (GAN) (Goodfellow et al., 2014)

A Generative Adversarial Network (GAN) consists of two components: a generator g_d, which maps latent vectors from a low-dimensional space to the data space, and a discriminator $g_c : \mathbb{R}^n \to \mathbb{R}$, which aims to distinguish between real samples and those generated by the generator. The two networks are trained in an adversarial setup: the discriminator is optimized to accurately classify real versus fake data, while the generator is trained to produce samples that are indistinguishable from real ones, thereby "fooling" the discriminator. This adversarial training framework leads to a minimax optimization problem. Numerous variants of GANs have been proposed to improve training stability and generation quality, such as Conditional GAN (Mirza & Osindero, 2014), WGAN (Arjovsky et al., 2017), StyleGAN (Karras et al., 2019; 2020; 2021), SinGAN (Shaham et al., 2019), and CycleGAN (Zhu et al., 2017).

A.1.3 Latent Diffusion Model (LDM) (Rombach et al., 2022)

A Latent Diffusion Model (LDM) is a generative model that, like a VAE, operates in a compressed latent space. However, unlike VAEs, LDMs use a diffusion process to model the data distribution. Specifically, an encoder g_e first maps high-dimensional data to a lower-dimensional latent space ($n \gg m$). A forward diffusion process $g_{df} : \mathbb{R}^m \to \mathbb{R}^m \to \cdots \to \mathbb{R}^m$ then gradually corrupts the latent variables by adding Gaussian noise over multiple time steps, producing a sequence of noisy latent representations. A learned denoising process g_{dn}, typically implemented as a U-Net, then reverses this diffusion to recover clean latent samples. Finally, a decoder g_d maps the denoised latent variables back to the data space:

$$g_d \circ g_{dn} \circ g_{df} \circ g_e : \mathbb{R}^n \to \mathbb{R}^n.$$

The encoder and decoder are typically built from convolutional neural networks (CNNs), while the denoiser is trained using a noise prediction objective similar to that in denoising diffusion probabilistic models (DDPMs) (Ho et al., 2020). Several variants of LDMs exist, including Stable Diffusion (Rombach et al., 2022), Imagen (Saharia et al., 2022), and DALL · E 3 (Betker et al., 2023).

A.1.4 Hallucinator (Liu et al., 2022)

A hallucinator is a type of image generator that transforms a basis vector $\mathbf{x} \in \mathbb{R}^{h \times w \times c}$ into an output image $\mathbf{x}' \in \mathbb{R}^{h' \times w' \times c'}$ through a composition of three components: an encoder, a linear transformation, and a decoder:

$$g_d \circ g_l \circ g_e : \mathbb{R}^{h \times w \times c} \to \mathbb{R}^{h' \times w' \times c'}.$$

Here, the encoder g_e and decoder g_d are typically implemented using convolutional neural networks (CNNs), while the linear transformer g_l performs an affine transformation of the form $g_l(\mathbf{x}) = \sigma \mathbf{x} + \mu$, where σ and μ are learnable parameters. Importantly, the shape of the input basis vector $\mathbf{x}$ need not match the shape of the generated image $\mathbf{x}'$, which allows for flexible dimensionality and structured generation.

A.1.5 Tensor decomposition (Yang et al., 2024)

A tensor decomposition operator maps a high-dimensional tensor (e.g. a 4D-tensor for images of standard size $b \times c \times h \times w$, where b, c, h, w represent batch size, channels, height, and width respectively) into a pair of tensors: a spectral tensor $\mathbf{T}$ and a kernel tensor $\mathbf{K}$,

$$\mathcal{A} : \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d} \to \mathbb{R}^{n'_1 \times n'_2 \times \cdots \times n'_d} \times \mathbb{R}^{n'_1 \times n'_2 \times \cdots \times n'_d \times n_1 \times n_2 \times \cdots \times n_d},\ \mathbf{X} \mapsto (\mathbf{T}, \mathbf{K}),$$

such that the original tensor $\mathbf{X}$ can be reconstructed by the tensor product of these components $\mathbf{X} = \mathbf{T} \otimes \mathbf{K}$. We further assume the kernel tensor $\mathbf{K}$ is separable, meaning it can be expressed as a tensor product of lower-dimensional matrices $\mathbf{K} = \hat{\mathbf{K}}_1 \otimes \cdots \otimes \hat{\mathbf{K}}_d$, where each $\hat{\mathbf{K}}_i \in \mathbb{R}^{n'_i \times n_i}$ is a 2D tensor. The original tensor $\mathbf{X}$ contains $\prod_{i=1}^{d} n_i$ elements. In contrast, storing $(\mathbf{T}, \mathbf{K})$ requires only $\prod_{i=1}^{d} n'_i + \sum_{i=1}^{d} n_i n'_i = O(\sum_{i=1}^{d} n_i)$ elements, since in practice the dimensions are chosen such that $n'_i \ll n_i$ in practice, resulting in a substantial reduction in storage.

A.2 Loss Regularization

The loss landscape of a neural network's training objective is typically non-convex and highly complex, making it challenging to find a globally optimal solution. To address this, various loss regularization terms are commonly introduced in practice. Depending on the desired property, these regularizations can serve different purposes: accelerating convergence, preventing overfitting, or encouraging additional diversity or similarity within the synthetic dataset.

A.2.1 Diversity term (Zhang et al., 2023)

Let $\mathcal{S} \subseteq \mathbb{R}^n$ denote a synthetic dataset, and let $\mathcal{S}^y \subseteq \mathcal{S}$ be the subset of samples labeled with class y. To balance intra-class diversity and inter-class discriminability, while also mitigating overfitting during training on synthetic data, we introduce the intra-class diversity loss:

$$L_{intra}(\mathbf{x}_i^y, \tau) = -\log \frac{e^{\langle h(\mathbf{x}_i^y), c(y) \rangle / \tau}}{e^{\langle h(\mathbf{x}_i^y), c(y) \rangle / \tau} + \sum_{i \neq j} e^{\langle h(\mathbf{x}_i^y), h(\mathbf{x}_j^y) \rangle / \tau}},$$

where $\mathbf{x}_i^y \in \mathcal{S}^y$ is a synthetic sample with label y, $c(y)$ is a learned class embedding for label y, $h(\cdot)$ denotes the feature extractor (e.g., a neural network encoder), and $\tau > 0$ is a temperature hyperparameter controlling the sharpness of similarity scores. Minimizing this loss promotes two effects: (1) it encourages a higher similarity (inner product) between the sample representation $h(\mathbf{x}_i^y)$ and its corresponding class embedding $c(y)$, reinforcing correct class prediction, and (2) it penalizes high similarity between different samples $\mathbf{x}_i^y$ and $\mathbf{x}_j^y$ $(i \neq j)$ of the same class, encouraging feature diversity within class y. This objective helps ensure that synthetic samples are both representative and diverse, improving generalization while maintaining class consistency.

Complementary to the intra-class diversity objective, we introduce the inter-class diversity loss to promote separation between synthetic samples from different classes. It is defined as:

$$L_{inter}(\mathcal{S}, \tau) = \sum_{y_1 \neq y_2} \mathrm{ReLU}\big(\tau - \big\|\overline{h(\mathcal{S}^{y_1})} - \overline{h(\mathcal{S}^{y_2})}\big\|_2\big),$$

where $\tau > 0$ is a margin (threshold) hyperparameter. This loss penalizes class pairs whose mean feature embeddings are too close, i.e., when their Euclidean distance falls below the threshold τ. By minimizing L_{inter}, the feature centroids of different classes are encouraged to be at least τ apart, promoting inter-class separability and reducing class confusion in the learned representation space. If the difference between the mean features of two different classes exceeds the margin, then the penalty to this class pair is zero.

A.2.2 Representative term (Gu et al., 2024)

To ensure that the small-scale synthetic dataset $\mathcal{S}$ effectively captures the characteristics of the original dataset $\mathcal{T}$, we introduce a representativeness loss that encourages each synthetic sample to align closely with at least one real data point. For a synthetic sample $\mathbf{x}_1 \in \mathcal{S}$, the loss is defined as:

$$L_{\text{rep}}(\mathbf{x}_1) = -\min_{\mathbf{x}_2 \in \mathcal{T}} \sigma(\mathbf{x}_1, \mathbf{x}_2),$$

where $\sigma(\mathbf{x}_1, \mathbf{x}_2)$ denotes the cosine similarity $\sigma(\mathbf{x}_1, \mathbf{x}_2) = \frac{\langle \mathbf{x}_1, \mathbf{x}_2 \rangle}{\|\mathbf{x}_1\|_2 \|\mathbf{x}_2\|_2}$. Minimizing L_{rep} encourages each synthetic point to closely resemble at least one sample in the original dataset, improving its representational fidelity. To further enhance coverage, we introduce a diversity loss that discourages redundancy among synthetic samples. For a given $\mathbf{x}_1 \in \mathcal{S}$, the loss is defined as:

$$L_{\text{div}}(\mathbf{x}_1) = \max_{\mathbf{x}_2 \in \mathcal{S}, \mathbf{x}_1 \neq \mathbf{x}_2} \sigma(\mathbf{x}_1, \mathbf{x}_2).$$

Minimizing L_{div} pushes synthetic samples away from each other in feature space, promoting diversity within $\mathcal{S}$. Together, minimizing both L_{rep} and L_{div} ensures that each synthetic sample is representative of the original data while remaining distinct from other synthetic points, striking a balance between coverage and diversity.

A.2.3 Contrastive term (Liu et al., 2022)

Let $\mathcal{H}$ be a finite set of hypotheses (e.g., neural network models). To evaluate how consistently synthetic data features are represented across different models within the same class, we introduce the adversarial contrastive loss:

$$L_{con}(\mathcal{H}, \mathcal{S}^y, \tau) = -\frac{1}{|\mathcal{H}|^2} \frac{1}{|\mathcal{S}^y|} \sum_{j \neq k} \sum_{i=1}^{|\mathcal{S}^y|} \log \frac{e^{\langle h_j(\mathbf{x}_i^y), h_k(\mathbf{x}_i^y) \rangle / \tau}}{\sum_{t=1}^{|\mathcal{S}^y|} e^{\langle h_j(\mathbf{x}_i^y), h_k(\mathbf{x}_t^y) \rangle / \tau}}.$$

where $h_j, h_k \in \mathcal{H}$ are hypotheses, $\mathbf{x}_i^y \in \mathcal{S}^y$ are synthetic samples from class y, and τ is a temperature scaling hyperparameter. This loss encourages consistency across models by aligning features extracted by different hypotheses for the same input, while contrasting against features from other class-y samples. In parallel, to quantify feature diversity within class y across models, we define the cosine similarity loss:

$$L_{cos}(\mathcal{H}, \mathcal{S}^y, \tau) = \frac{1}{|\mathcal{H}|^2} \frac{1}{|\mathcal{S}^y|} \sum_{j \neq k} \sum_{i=1}^{|\mathcal{S}^y|} \sigma(h_j(\mathbf{x}_i^y), h_k(\mathbf{x}_i^y)),$$

where $\sigma(\cdot, \cdot)$ denotes the cosine similarity $\sigma(\mathbf{x}_1, \mathbf{x}_2) = \frac{\langle \mathbf{x}_1, \mathbf{x}_2 \rangle}{\|\mathbf{x}_1\|_2 \|\mathbf{x}_2\|_2}$. By minimizing both L_{con} and L_{cos}, we enforce a dual objective: (1) inter-model alignment of features within each class to ensure semantic consistency, and (2) intra-class variation to maintain feature diversity, avoiding degenerate or overly uniform representations across the hypothesis ensemble.

A.2.4 Discrimination term (Wang et al., 2022)

Let $\mathcal{T}$ be a finite dataset, $\mathcal{S}$ a synthetic dataset, and h a trained model. For each class label $y \in 1, \ldots, C$, denote by $\overline{h(\mathcal{S}^y)}$ the mean feature representation of synthetic samples from class y. For each sample $\mathbf{x}_i^y \in \mathcal{T}^y$ in the real dataset, define the class-similarity score vector $\mathbf{O}_i^y = [\langle h(\mathbf{x}_i^y), \overline{h(\mathcal{S}^y)} \rangle]_{y=1}^C \in \mathbb{R}^C$. Intuitively, $\mathbf{O}_i^y$

encodes the similarity between the feature of a real sample and the average synthetic features from each class. We then define the discrimination loss as:

$$L_{dis} = -\frac{1}{C}\frac{1}{|\mathcal{T}^y|}\sum_{y=1}^{C}\sum_{i=1}^{|\mathcal{T}^y|}\log c(\mathbf{O}_i^y),$$

where $c(\mathbf{O}_i^y)_y$ denotes the softmax probability assigned to the true class y. This loss measures how well the average synthetic class features serve as discriminative class prototypes for real data. By minimizing L_{dis}, the synthetic dataset $\mathcal{S}$ is encouraged to produce class-wise features that closely align with the features extracted from real samples in $\mathcal{T}$, thereby improving the semantic alignment between synthetic and real data.

A.3 Data Augmentation

While generative modeling seeks to reduce the dimensionality of the data space, a complementary technique is data augmentation, which aims to increase the diversity or complexity of the data. The goal is to improve downstream performance by training models on augmented data that better captures the variability of the original distribution.

Given a 4D tensor representing a batch of images of shape $b \times c \times h \times w$, where b, c, h, and w denote the batch size, number of channels, height, and width respectively, the multi-formation operator (Kim et al., 2022) increases the number of channels by a factor of r, referred to as the formation factor:

$$\mathcal{A}_r : \mathbb{R}^{b\times c\times h\times w} \to \mathbb{R}^{b\times c(r^2+1)\times h\times w}.$$

The illustration of multi-formation process is shown in Figure 2.

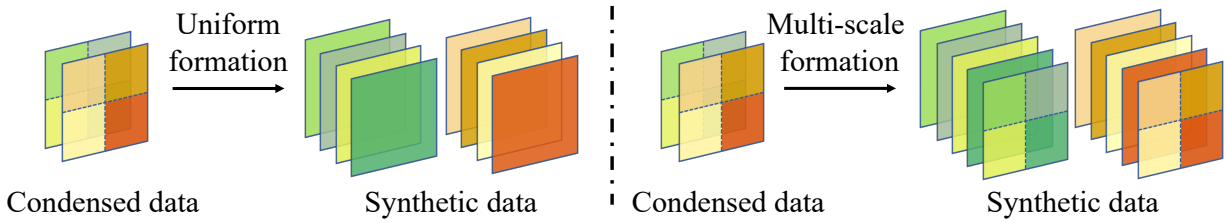


Figure 2: Illustration of multi-formation operator with a factor of 2.

In recent work, (Zhou et al., 2024) propose a channel-wise multi-formation operator for data augmentation, defined as

$$\mathcal{A}_c : \mathbb{R}^{b\times c\times h\times w} \to \mathbb{R}^{4b\times c\times h\times w},$$

which expands the dataset by a factor of four. Specifically, each input sample is augmented using three 1×1 convolutional layers, each acting as a color mapping function. As a result, a batch of b samples is transformed into $3b$ additional color-mapped samples via $3b$ distinct convolution layers. The original and augmented (color-shifted) data are then concatenated along the batch dimension, and the resulting tensor can be used for model training or as input to dataset condensation pipelines.