Dataset Distillation as Data Compression: A Rate-Utility Perspective

Youneng Bao[1,3,*], Yiping Liu[1,*], Zhuo Chen[2], Yongsheng Liang[1], Mu Li[1,†], Kede Ma[3,†]
[1]Harbin Institute of Technology, Shenzhen [2]Peng Cheng Laboratory [3]City University of Hong Kong
younebao@cityu.edu.hk, yipingliu@stu.hit.edu.cn, chenzhuo.zoom@gmail.com
{liangys,limu2022}@hit.edu.cn, kede.ma@cityu.edu.hk
https://github.io/xxxx

Abstract

Driven by the 'scale-is-everything' paradigm, modern machine learning increasingly demands ever-larger datasets and models, yielding prohibitive computational and storage requirements. Dataset distillation mitigates this by compressing a full dataset into a small set of synthetic samples, preserving its original utility. Yet, existing methods either optimize performance under a fixed storage budget or remove sample redundancy, without jointly balancing both objectives, thus precluding Pareto-optimal trade-offs. In this work, we propose a joint Rate-Utility Optimization (RUO) method for dataset distillation. Specifically, we parameterize synthetic samples as optimizable latent codes decoded by lightweight networks. Such a hybrid parameterization subsumes all prior explicit and implicit representations for synthetic datasets. We estimate the entropy of quantized latents as the rate measure and plug any existing distillation loss as the utility measure, trading them off via a Lagrange multiplier. To enable fair, cross-method comparisons, we introduce bits per class (bpc), a precise storage metric that accounts for sample, label, and decoder costs. On CIFAR-10, CIFAR-100, and ImageNet-128, our method achieves up to $170\times$ *greater compression than standard distillation at comparable accuracy. Across diverse bpc budgets, distillation losses, and backbone architectures, our approach consistently establishes new state-of-the-art rate-utility trade-offs.*

1. Introduction

The rapid advances in machine learning have been driven by the 'scale-is-everything' paradigm [4, 21], in which ever-larger models trained on ever-more extensive datasets yield progressively better performance across various computational prediction tasks [5, 41]. However, this scaling trend incurs steep computational, storage, and environmental costs, posing a significant barrier to sustainable and accessible research and deployment.

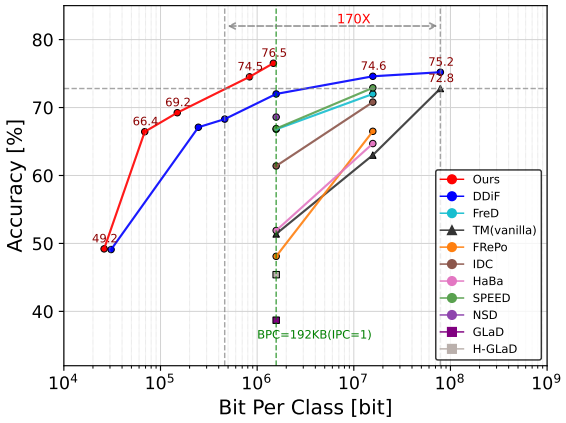


Figure 1. The Rate-utility curves of different dataset distillation methods on Imagenet-subset Nette. Our method achieves a better trade-off between rate and utility compared to existing methods. **BPC** is defined as the bits per class, directly measuring the storage cost of the distilled dataset. BPC axis uses logarithmic scale.

Dataset distillation (DD) [40] addresses these challenges by learning a compact, synthetic dataset that enables rapid model prototyping, accelerated model training, and efficient hyperparameter tuning, while lowering compute, storage, and energy requirements [26]. At the core of DD lies a trade-off between *rate* (*i.e.*, the storage footprint of the synthetic dataset) and *utility* (*i.e.*, the performance of models trained on distilled data).

Initially, DD is formulated as a bilevel meta-optimization problem [40]. Solving this exactly requires backpropagation through time (BPTT) [43] across the entire training trajectory, which suffers from computational inefficiency for long sequences, high memory consumption, and gradient instability. From the *utility* perspective, meta-gradient estimation has been pursued through a variety of techniques, including truncated BPTT [14], trajectory and distribution matching [6, 49], closed-form kernel surrogates [35],

*Equal contribution
†Corresponding author

and implicit differentiation [30]. From the *rate* perspective, recent efforts [7, 23, 36, 42, 45, 47] remove synthetic sample redundancy by embedding the dataset into a low-dimensional latent space, which is then decoded via fixed structured or generative models.

Despite impressive progress, existing DD methods optimize rate and utility separately, precluding Pareto-optimal trade-offs. Inspired by end-to-end rate-distortion optimization of learned image compression [2], we describe a joint rate-utility optimization method for DD. Our approach parametrizes synthetic samples as multiscale grids of learnable latent codes, decoded by small learnable networks, which generalizes all existing explicit and implicit distilled-dataset representations. To quantify the *rate*, we assume context-aware zero-mean Laplace distributions over *quantized* latents, where the scale parameters are predicted using causal neighbouring information, and compute the entropy under this Laplace prior. To quantify the *utility*, we exploit the plug-and-play property of our approach and directly employ any off-the-shelf distillation loss (*e.g.*, those from trajectory or distribution matching). We end-to-end optimize all parameters of our method for our joint rate-utility objective with a tuning Lagrange multiplier.

To enable fair, cross-method comparisons, we introduce bits per class (bpc), a precise storage metric that measures the average number of bits needed to store distilled data per class—contrastive to the extensively used images per class (ipc) metric, which grows linearly with raw image dimensions and fails to capture the additional bits required to store decoder parameters and to encode target labels [].

In summary, our key contributions include

- A joint rate-utility optimization method for DD that retains full compatibility with existing distillation losses,
- A hybrid parameterization of synthetic samples that subsumes prior synthetic dataset representations,
- A bpc metric for enabling standardized and fair rate-utility comparisons across various DD methods, and
- An experimental demonstration on CIFAR-10, CIFAR-100, and ImageNet-128 that our method consistently establishes new state-of-the-art rate-utility trade-offs across various datasets, storage budgets, distillation losses, and network architectures (see Fig. 1).

2. Related work

In this section, we review prior efforts in DD and data compression. Our method bridge the conceptual and computational gap between these two fields, allowing principled exploration of Pareto-optimal trade-offs between rate and utility in DD.

2.1. Dataset Distillation

Wang *et al*. [40] first formulated DD as a bilevel optimization problem, which generalizes coreset selection [17]. The reliance of BPTT over the entire inner-loop trajectory, incurring prohibitive computation, high memory usage, and unstable meta-gradients. To reduce this burden, several meta-gradient estimation strategies have been proposed. Feng *et al*. [14] introduced random truncated BPTT to cap the number of unrolled steps. Nguyen *et al*. [] replaced the iterative inner-loop solver with a kernel ridge regression surrogate. To directly optimize the synthetic samples, gradient matching [50] chose to align instantaneous updates, while holding network parameters fixed. Instead of matching gradients at a single point, trajectory matching [6] aligns the entire sequence of parameter updates obtained by training on synthetic data with the sequence obtained on original data. Both methods require computationally expensive second-order gradient computation. Given the fact that measuring similarity in parameter space is notoriously difficult because it contains large "gauge" symmetries (*e.g.*, channel permutation invariance), Zhao *et al*. [49] proposed to match the synthetic and target data from the distribution perspective, leveraging a pretrained feature extractor. Another advantage of distribution matching is that it bypasses second-order gradient computation and thus is more scalable. Recent efforts have been put to address the scalability of DD methods to large, high-resolution datasets, typically via the decoupling trick in bilevel optimization. Teddy [] decouples the bilevel loop by Taylor-expanding the meta-objective to turn nested gradient alignment into a simple sum of inner-products between first-order gradients, and then showing that under mild assumptions, this is equivalent to matching feature-space means and variances. SRe2L [] decouples the bilevel loop by splitting dataset distillation into three independent stages—(1) train a teacher on real data, (2) recover synthetic images by matching their BatchNorm statistics and outputs against this frozen teacher in a single-level optimization, and (3) relabel and train the student from scratch—so that synthetic-data generation never requires nested model updates or higher-order gradients. Other emerging topics in DD involving addressing robustness challenges to out-of-distribution data, cross architectures, varying initialization, adversarial attacks and label distillation.

Close to our work is the so-called factorized DD [], which factorizes the synthetic dataset with a combination of codes and decoders, in an attempt to improve compression efficiency. Code-centric (*i.e.*, explicit) DD chooses to only learn a set of low-dimensional codes, while holding decoders fixed, which can be implemented by classic feature transforms like discrete cosine transform [36] and Tucker decomposition [47] and generative models like generative adversarial networks [7] and diffusion models [8, 39, 52]. It is worth noting that the original DD can be viewed as a code-centric one, where the decoder is simply an identity mapping. Decoder-centric (*i.e.*, implicit) DD solely op-

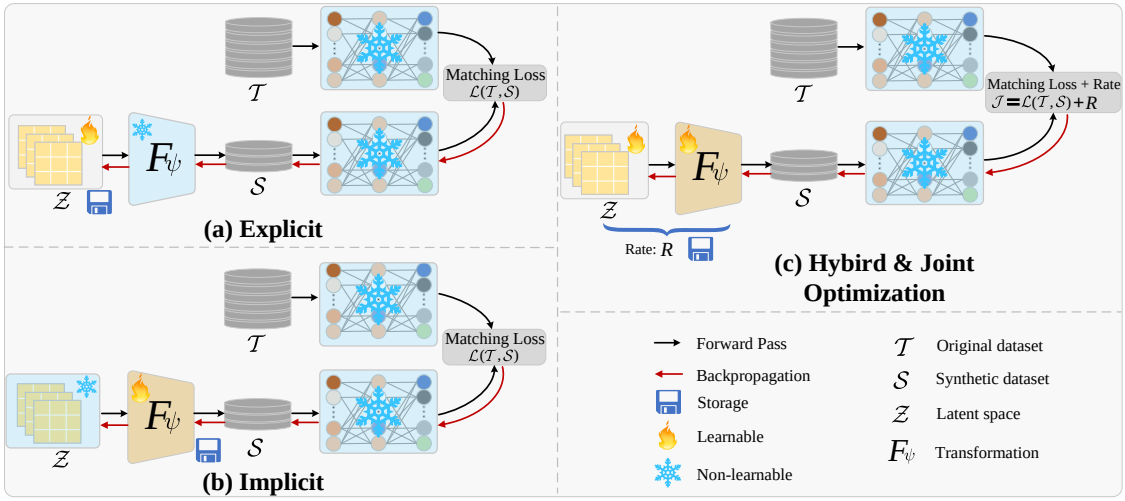


Figure 2. Comparison of dataset distillation frameworks. (a) Explicit parameterization. (b) Implicit parameterization. (c) Our framework combines explicit and implicit parametrization and generates compact representations by directly optimizing bitrate of synthetic datasets.

timizes decoders while fixing codes (typically to random noise) []. Code-decoder (*i.e.*, hybrid) DD allows training of both codes and decoders [9], to which our method belongs.

Existing DD methods do not have an explict and formal treatment of rate to quantify the total size of the synthetic data and optimizes it jointly with the utility term.

2.2. Data Compression

Data compression is broadly categorized into lossless and lossy compression paradigms. Lossless compression research has evolved from foundational statistical [33] and dictionary-based methods [10]. Modern approaches increasingly incorporate machine learning to model complex data distributions by learning long-range dependencies [15, 20, 27, 48], though traditional algorithms remain dominant in practical applications due to their simplicity and reliability. Challenges persist in accurately modeling probability distributions of high-dimensional data, addressing computational complexity[46]. Lossy data compression optimizes the trade-off between compression efficiency and reconstruction quality by controlled information loss. Neural compression, pioneered by [2, 3], employs neural networks to directly learn optimal transform strategies. Techniques such as variational autoencoders (VAEs) [18, 32], GANs [31], and diffusion models [1, 44] now achieve state-of-the-art compression-quality trade-offs. Another paradigm overfits a neural network to one image [13, 16, 22], then transmits its quantized weights as the compressed bitstream. Because the decoder's task is limited to reconstructing a single image, decoding complexity is lowered by several orders of magnitude.

Both dataset distillation and data compression fundamentally pursue minimum-entropy representations to eliminate information redundancy. Capitalizing on this essential commonality, our research introduces data compression's end-to-end optimization framework into dataset distillation, establishing a rate-utility framework.

3. Methodology

3.1. Problem Formulation

Dataset distillation can be modeled as a bi-level optimization problem. Given an original dataset $\mathcal{T} := (\mathcal{X}_T, \mathcal{Y}_T) = \{(x_i, y_i)\}_{i=1}^N$ consisting of N data points x_i and corresponding labels y_i, the goal of dataset distillation is to find a smaller, synthetic dataset $\mathcal{S} := (\mathcal{X}_S, \mathcal{Y}_S) = \{(\tilde{x}_j, \tilde{y}_j)\}_{j=1}^M$, and $M \ll N$, such that a model trained on $\mathcal{S}$ performs comparably to a model trained on $\mathcal{T}$. More formally, let $f(x;\theta)$ represent a model parameterized by θ. The objective of dataset distillation is formulated as follows:

$$\begin{aligned} &\min_{\mathcal{S}} \mathbb{E}_{(x,y)\sim p(x,y)}[\ell(f_{\theta_{\mathcal{S}}}(x), y)] \\ &\text{where } \theta_{\mathcal{S}} = \arg\min_{\theta} \frac{1}{M} \sum_{(\tilde{x}_j,\tilde{y}_j)\in\mathcal{S}} \ell(f_\theta(\tilde{x}_j), \tilde{y}_j) \end{aligned} \tag{1}$$

where ℓ is a loss function (e.g., cross-entropy loss), and $\theta_{\mathcal{S}}$ are model parameters trained by synthetic dataset. Eq (1) is a bi-level optimization, separately optimizing the synthetic dataset $\mathcal{S}$ and the model parameters θ. This formulation requires solving the inner optimization repeatedly for different candidate synthetic datasets during the search for $\mathcal{S}$, increasing computational cost. A research direction is **Efficient Learning**, summarized in Section (2.1). For clarity, the aforementioned process is denoted as $\mathcal{L}(\mathcal{T}, \mathcal{S})$. Regarding the Representation of distilled dataset $\mathcal{S}$, expressed by the following formula:

Table 1. DD methods categorized by their representation approaches. The synthetic samples are parameterized as $\tilde{x}_j = F_{\psi_j}(\mathcal{Z})$.

Category	Method	Transformation F_{ψ_j}	Input Space $\mathcal{Z}$	Learned	Stored
Explicit	DD [40], DM [49], TM [6], FRePo [53]	Identity Mapping, ψ fixed	Learnable Latent	$\mathcal{Z}$	$\mathcal{Z}$
	IDC [23], FreD [36]	Kernel Function, ψ fixed		$\mathcal{Z}$	$\mathcal{Z}$
	TuckD [47], NSD [45]	Tensor Decomposition, ψ fixed		$\mathcal{Z}$	$\mathcal{Z}$
	GLaD [7],H-GLaD [52]	GAN,ψ fixed		$\mathcal{Z}$	$\mathcal{Z}$
	IGD [8], D4M [39], LD3M [34]	Diffusion Model		$\psi, \mathcal{Z}$	$\mathcal{Z}$
	HaBa [28], LatentDD [12]	AE		$\psi, \mathcal{Z}$	$\mathcal{Z}$
Implicit	DDiF [37]	NeRF, $\mathcal{Z}$ fixed	Coordinate Space	ψ_j	ψ_j
Hybrid	SPEDD [42],Proposed	Neural Network	Learnable Latent	$\psi_j, \mathcal{Z}$	$\psi_j, \mathcal{Z}$

$$\begin{aligned} &\min_{\Psi} \mathcal{L}(\mathcal{T}, \mathcal{S}), \\ &\text{where} \quad \mathcal{S} \triangleq \{(F_\psi(z_j), \tilde{y}_j) \mid z_j \sim \mathcal{Z}\}_{j=1}^{|\Psi|}, \end{aligned} \tag{2}$$

where $\mathcal{Z}$ represents latent space and $\Psi = \{\psi_j\}_{j=1}^{|\Psi|}$ denotes a set of parameters defining F_ψ. F_ψ can be implemented as a neural network or other differentiable mapping. Each function maps z_j to create a synthetic sample $\tilde{x}_j$.

Equation (2) provides a unified framework for parameterized dataset distillation. Existing methods fall into two categories: explicit methods, which directly model the synthetic dataset with parameters, and implicit methods, which use parameterized transformations in a latent space. Table 1 compares their characteristics. Existing parameterization methods improve feature diversity and task performance under storage limits. However, because they **fail to model the rate as a differentiable variable for end-to-end joint optimization,** storage cost and performance are treated separately, hindering Pareto optimality.

3.2. Hybrid Parameterization Framework

As shown in Figure 3, we propose a hybrid framework that combines explicit and implicit parameterization by jointly learning both the latent feature $\mathcal{Z}$ and transformation function F_ψ, while storing parameters of both components. The total rate consists of explicit and implicit components: the explicit component corresponds to the encoding of latent features $\mathcal{Z}$, while the implicit component accounts for the parameters of the transformation function F_ψ. The total rate is formulated as:

$$R_{\text{total}} = R_{\text{ex}} + R_{\text{im}} \tag{3}$$

where R_{ex} and R_{im} represent the explicit part and implicit part of the rate, respectively. The explicit part preserves the core information of data through direct encoding of latent features, while the implicit part further compresses storage requirements by parameterizing the transformation function, thereby enhancing the framework's capability to generate diverse synthetic data.

3.2.1. Explicit Parametrization

To obtain more compact latent features, z first undergoes numerical quantization Q, followed by entropy coding, and the bit rate after quantization can be expressed as:

$$R(\hat{z}) = \mathbb{E}_{\hat{z}\sim q}\left[-\log_2 p(\hat{z})\right], \text{ where } \hat{z} = Q(z). \tag{4}$$

Since quantization is non-differentiable, during backpropagation, the straight-through estimator is used to make it differentiable. $p(\hat{z})$ is the probability distribution estimate of $\hat{z}$. Inspired by [3] and the COOL-CHIC series[22, 25], it is assumed that $\hat{z}$ follows an independent Laplace distribution $\mathscr{L} \sim (\mu, \sigma)$, and then a simple autoregressive model is used to estimate the distribution parameters μ, σ, expressed by the following formula:

$$\begin{aligned} P_{\phi,\Omega}(\hat{z}^n) &= \prod_{i,j} \mathscr{L}\left(\hat{z}_{ij}^n \mid \mu_{ij}^n, \sigma_{ij}^n\right), \\ \mu_{ij}^n, \sigma_{ij}^n &= g_\phi\left(h_\Omega\left(\hat{z}_{ij}^n\right)\right). \end{aligned} \tag{5}$$

Here, h_Ω is a causal context, implemented by masked convolution with parameters Ω. g_ϕ is a simple MLP with parameters ϕ . Both estimate the entropy parameters at position (n, i, j). Thus, the explicit part not only needs to store the quantized feature $\hat{\mathbf{z}}$ but also needs to store the NN parameters Ω and g_ϕ. Finally, the rate present sums up to:

$$\begin{aligned} R_{\text{ex}} &= R(\hat{z}) + R(\phi, \Omega) \\ &= -\log_2 \prod_{i,j} \mathscr{L}\left(\hat{z}_{ij}^n \mid \mu_{ij}^n, \sigma_{ij}^n\right) + R(\phi, \Omega) \\ &= \sum_{i,j,n} \mathscr{L}\left(\hat{z}_{ij}^n \mid \mu_{ij}^n, \sigma_{ij}^n\right) + R(\phi, \Omega) \end{aligned} \tag{6}$$

The network parameters $\{\Omega, \psi\}$ are trained using 32-bit floating-point, and post-quantization techniques are applied after training, followed by [22].

3.2.2. Implicit Parametrization

To generate synthetic data from $\mathcal{Z}$, we employ a compact synthetic network F_ψ with parameters ψ, defined as:

$$\tilde{x}_j = F_\psi(z). \tag{7}$$

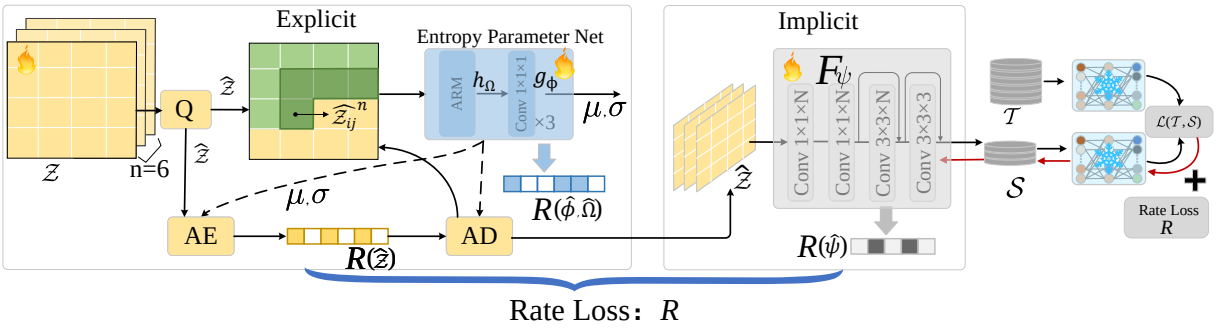


Figure 3. Our proposed end-to-end joint rate-utility optimization framework for dataset distillation.

The trained parameters ψ of F_ψ are compressed via quantization and entropy encoding, following the method in [22]. Initially, F_ψ is trained using full-precision floating-point weights. After full-precision training, the parameters ψ are quantized as:

$$\hat{\psi} = Q(\psi, \Delta_\psi), \tag{8}$$

where Δ_ψ denotes the quantization step size determined by a greedy algorithm to balance precision and bitrate, consistent with the compression strategy for parameters in $\{\Omega, \psi\}$. The quantized weights $\hat{\psi}$ are further compressed using entropy encoding, assuming their distribution follows a zero-mean Laplace distribution. The probability distribution is modeled as:

$$P(\hat{\psi}) = \prod_i P\left(\hat{\psi}_i \mid \mu = 0, \sigma = \frac{1}{\sqrt{2}}\text{std}(\hat{\psi})\right), \tag{9}$$

enabling efficient entropy coding. Therefore, the rate of the implicit part is:

$$R_{\text{im}} = -\log_2(P(\hat{\psi})). \tag{10}$$

3.3. Joint Rate-Utility Optimization

Under the hybrid framework, we achieve joint rate-utility optimization by defining a Rate-Utility Loss $\mathcal{J}$, formally defined as follows:

$$\min_{\hat{z},\boldsymbol{\phi},\boldsymbol{\psi},\boldsymbol{\Omega}} \{\mathcal{L}(\mathcal{T}, F_\psi(\hat{z})) - \lambda \sum \log_2 p_{\boldsymbol{\phi},\boldsymbol{\Omega}}(\hat{z})\}. \tag{11}$$

The first term represents utility loss, quantifying discrepancy between models trained on original and synthetic data (e.g., DM, TM). This design renders our framework plug-and-play and compatible with arbitrary matching loss methodologies. The second term represents the rate loss, which can be further decomposed via equations (5) and (6). λ is a hyperparameter that controls the trade-off between utility and rate. A larger λ emphasizes compression more, while a smaller λ emphasizes utility. The optimization process yields the learned, optimized parameters that minimize

Algorithm 1 Joint Rate-Utility Optimization Dataset Distillation

Input: Original dataset $\mathcal{T}$; Number of classes C; Distillation loss $\mathcal{L}$; Decoded instance per class k; Trade-off coefficient λ and β; Learning rate η and η_1; Batch size B;
Initialize: latent $\mathbf{z}$, synthetic network parameters $\boldsymbol{\psi}$, distribution net parameters $\boldsymbol{\phi}, \boldsymbol{\Omega}$.
for $c = 1$ to C **do**
  Sample a mini-batch $B_c = \{(x_i, y_i)\}_{i=1}^{B} \sim \mathcal{T}^{(c)}$.
  Initialize parameters by overfitting the neural network, fitting target Eq.(12).
end for
repeat
  Compute the total Rate-Utility Loss $\mathcal{J}$
  Update parameters using gradient descent:
  $\{\mathbf{z}, \boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\Omega}\} \leftarrow \{\mathbf{z}, \boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\Omega}\} - \eta \nabla \mathcal{J}(\mathbf{z}, \boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\Omega})$
until $\mathcal{J}$ convergence
Post-Quantization
Output: $\{\hat{\mathbf{z}}, \hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\psi}}, \hat{\boldsymbol{\Omega}}\}$

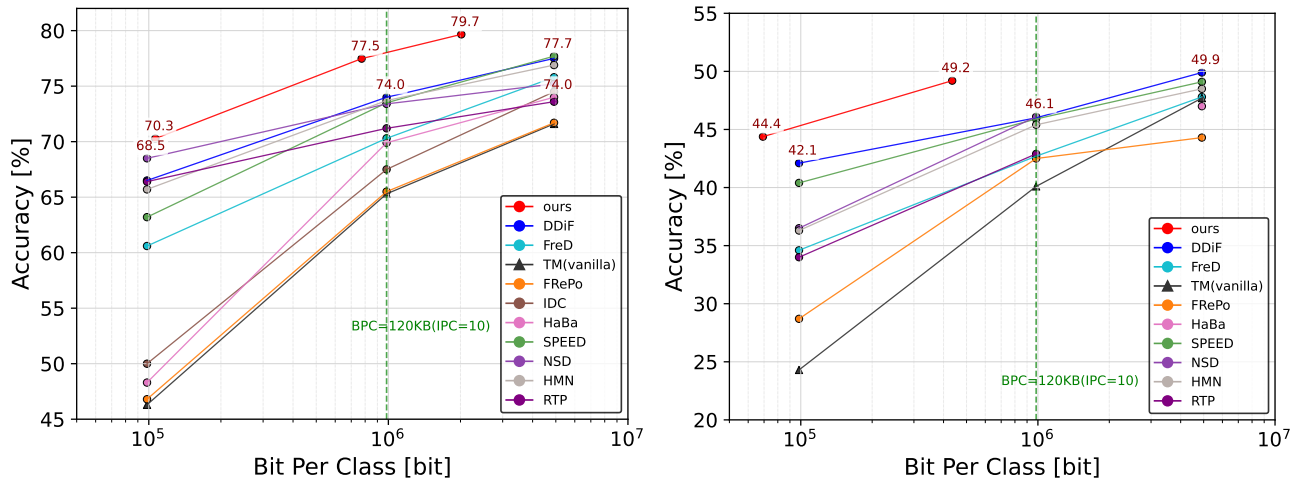
the Rate-Utility loss, including the latent variables $\hat{\mathbf{z}}$, the parameters $\boldsymbol{\phi}, \boldsymbol{\Omega}$ of the probability estimation network, and the parameters $\boldsymbol{\psi}$ of the synthetic data generation network. To accelerate convergence, before optimizing $\mathcal{J}$, the parameters of the neural network are obtained by overfitting the original samples and used as initialization. The fitting target is:

$$\mathcal{L}_{init} = \|F_{\boldsymbol{\psi}}(z) - x_i\|_2^2 - \beta \sum \log_2 p_{\boldsymbol{\phi},\boldsymbol{\Omega}}(\hat{z})\}, \tag{12}$$

Here, β is used to trade off between the two terms.

3.4. Bits Per Class Metric

IPC solely measures the total parameter volume of synthetic images, neglecting their storage cost. Even with IPC fixed at 1, both the storage size and performance of the distilled dataset can vary significantly depending on the representation strategy, as shown in Figure **??**. Therefore, IPC unreliable for assessing dataset distillation efficacy. We introduce



(a) CIFAR-10 (b) CIFAR-100

Figure 4. **Comparisons of Rate-Utility curves with other methods on the CIFAR-10/100 datasets. TM is used as the utility loss.** The horizontal axis uses a logarithmic scale. For detailed quantitative results of each method, please refer to Supplementary materials.

Bits Per Class (BPC) as follows:

$$\text{BPC} = \frac{R_S}{N_C}, \tag{13}$$

where R_S is the total storage size (in bits) of synthetic data, N_C is the number of classes. BPC is a superior metric to IPC for evaluating dataset distillation because it directly measures storage efficiency. BPC accounts for both compression and storage costs, penalizing methods that achieve high IPC but require excessive storage per sample. This allows for fairer comparisons and analysis of the accuracy-storage trade-off.

4. Experiments

4.1. Experimental Setup

Datasets We validated the results on CIFAR-10 and CIFAR-100 with a resolution of 32 × 32, as well as on a subset of ImageNet with a resolution of 128×128. The ImageNet-Subset includes Nette, Woof, Fruit, Meow, Squawk, and Yellow.

Baselines and Networks. We compared the current parameterized distillation methods, including TM [6], FRePo [53], IDC [23], FreD [36], HaBa[28], SPEED [42], NSD[45], GLaD [7], H-GLaD [7], HMN [51], RTP[9], and DDiF [37]. To verify the adaptation to different matching loss, we used three widely-used methods: DC [50], DM [49], and TM [6]. To evaluate cross-architecture generalization, we adopted ConvNet [29] as the foundational architecture and conducted training from scratch on multiple network models including AlexNet [24], ResNet-18 [19], VGG-11 [38] , and ViT-16 [11], with validation accuracies systematically recorded throughout the process. All experiments were conducted 5 times for reducing randomness.

Table 2. Test accuracies (%) on ImageNet-Subset with regard to various dataset parameterizations under a BPC budget $<$ 192KB. We utilize trajectory matching (TM) as Vanilla. "-" indicates no reported results in original paper.

Method	ImageNet-Subset(128×128)					
	Nette	Woof	Fruit	Yellow	Meow	Squawk
Whole Dataset	87.4	67.0	63.9	84.4	66.7	87.5
Vanilla[6]	51.4	29.7	28.8	47.5	33.3	41.0
FRePo[53]	48.1	29.7	-	-	-	-
IDC[23]	61.4	34.5	38.0	56.5	39.5	50.2
FreD[36]	66.8	38.3	43.7	63.2	43.2	57.0
HaBa[28]	51.9	32.4	34.7	50.4	36.9	41.9
SPEED[42]	66.9	38.0	43.4	62.6	43.6	60.9
NSD[45]	68.6	35.2	39.8	61.0	45.2	52.9
GLaD[7]	38.7	23.4	23.1	-	26.0	35.8
H-GLaD[52]	45.4	28.3	25.6	-	29.6	39.7
DDiF[37]	72.0	42.9	48.2	69.0	47.4	67.0
Ours	**76.5**	**46.0**	**50.8**	**74.3**	**53.5**	**74.8**
Ours BPC(KB)	179.7	146.0	145.4	171.5	170.6	161.4

Parameter settings In line with previous work, we used different ConvNet architectures for each dataset: a 3-layer ConvNet for CIFAR-10/100 and a 5-layer ConvNet for ImageNet-subsets. For detailed settings of other parameters, please refer to the supplementary materials.

4.2. Main Results

We report the performance of the synthetic dataset under different BPC settings, where the distillation loss is TM by default.

Rate-Utility Trade-off Performance Figure 4 shows rate-

Table 3. Average test accuracies (%) on ImageNet-Subset (128×128) across AlexNet, VGG11, ResNet18, and ViT, under a BPC budget $<$ 192KB. MTT is used as the utility loss.

Method	Nette	Woof	Fruit	Yellow	Meow	Squawk	Avg
TM[6]	22.0	14.8	17.1	22.3	16.2	25.5	19.7
IDC[23]	27.9	19.5	23.9	28.0	19.8	29.9	24.8
FreD[36]	36.2	23.7	23.6	31.2	19.1	37.4	28.5
GLaD[7]	30.4	17.1	21.1	-	19.6	28.2	23.3
H-GLaD[52]	30.8	17.4	21.5	-	20.1	28.8	23.7
LD3M[34]	32.0	19.9	21.4	-	22.1	30.4	25.2
DDiF[37]	59.3	34.1	39.3	51.1	33.8	54.0	45.3
Ours	**65.6**	**36.0**	**42.8**	**58.0**	**40.6**	**61.2**	**50.7**

Table 4. Test accuracies (%) on ImageNet-Subset (128×128) with a BPC budget $<$ 192KB: A comparison of different utility losses and methods (DC[50], DM[49]).

$\mathcal{L}$	Method	Nette	Woof	Fruit	Meow	Squawk	Avg
DC	Vanilla	34.2	22.5	21.0	22.0	32.0	26.3
	IDC	45.4	25.5	26.8	25.3	34.6	31.5
	FreD	49.1	26.1	30.0	28.7	39.7	34.7
	GLaD	35.4	22.3	20.7	22.6	33.8	27.0
	H-GLaD	36.9	24.0	22.4	24.1	35.3	28.5
	DDiF	61.2	**35.2**	**37.8**	39.1	54.3	45.5
	Ours	**67.2**	**33.1**	**37.0**	**42.0**	**58.6**	**47.6**
DM	Vanilla	30.4	20.7	20.4	20.1	26.6	23.6
	IDC	48.3	27.0	29.9	30.5	38.8	34.9
	FreD	56.2	31.0	33.4	33.3	42.7	39.3
	GLaD	32.2	21.2	21.8	22.3	27.6	25.0
	H-GLaD	34.8	23.9	24.4	24.2	29.5	27.4
	DDiF	69.2	42.0	45.3	45.8	64.6	53.4
	Ours	**71.9**	**46.4**	**49.0**	**48.8**	**69.0**	**57.0**

utility curves on CIFAR-10/100. Our method achieves the highest accuracy across all BPC ranges on CIFAR-10. With BPC=94KB ($<$120KB budge), our accuracy is 78%, BPC=246KB ($<$600KB budge), accuracy is 79.7%, significantly outperforming others. On CIFAR-100, our method also excels. We reach 44.4% , 49.2% accuracy with only 8KB, 53KB BPC respectively, also achieved SOTA. Figure 1 and Table 2 show rate-utility performance on ImageNet-sub. With average BPC=162KB ($<$192KB), we also achieve the best performance, like 76.5% on Nette and 74.3% on Yellow, an average 24% improvement over Vanilla. The experimental results strongly prove that our method achieves the best rate-utility trade-off. For more detailed results, please refer to the supplementary materials.

Cross-architecture Generalization. To evaluate the generalization capability of synthetic data across different network architectures, classification tasks were trained on four widely used architectures. As shown in Table 3, under a strict bitrate budget (BPC $<$ 192KB), our method achieved the highest average test accuracy of 50.7% across diverse architectures (AlexNet, VGG11, ResNet18, and ViT), outperforming the second-best method by 5.4 %, with optimal results on all subsets. For instance, our method achieved 65.6% accuracy on the Nette subset, surpassing DDiF (59.3%) by 6.3%, and reached 40.6% accuracy on the Meow subset, exceeding other methods by nearly 10%. These results demonstrate the robustness and adaptability of the proposed method in cross-architecture scenarios.

Compatibility of Dataset Distillation Loss. To verify the adaptability of our method to different distillation losses, Table 4 reports the performance optimized using DC and DM as utility terms. As shown in Table 4, under a strict bitrate budget (BPC $<$ 192KB), our method achieves the best performance in both frameworks: in the DC framework, the average accuracy reaches 47.6%; in the DM framework, the average accuracy increases to 57.0%, achieving SOTA results in both cases. These results prove that our end-to-end method can adapt to different distillation losses and stably enhance performance, exhibiting broad compatibility.

4.3. Ablation Studies

The Parameter Quantity Budget. We first investigates the efficiency-performance trade-off of our method under constraints of total parameter quantity and decoded image count. With the same number of decoded instances, our method reduces storage costs by 47% compared to DDiF [37] (e.g., BPC decreases from 192 KB to 101.7 KB when DIPC=51) while achieving up to 2.5% higher accuracy. Under a parameter budget constraint of approximately 2.5M parameters per class, our method improves storage efficiency by 98% (BPC reduced from 9592 KB to 180 KB). This demonstrates that joint rate-utility optimization, rather than merely increasing parameter scale, can significantly enhance information density and achieve superior model performance under resource-constrained conditions.

Table 5. Performance comparison with DDIF [37] **under the same decoded synthetic image per class(DIPC)**, on ImageNet-subset nette.

DIPC	Method	BPC(KB)	Acc(%)
1	DDiF [37]	3.7	49.1
	Ours	3.2	49.2
8	DDiF [37]	30.0	67.1
	Ours	8.4	66.4
15	DDiF [37]	56.4	68.3
	Ours	18.2	69.2
51	DDiF [37]	192.0	72.0
	Ours	101.7	74.5

Synthetic Data Visualizations. To better understand the mechanism of generating synthetic data, Figure 5 demonstrates the decoding process of features. Object contour information can be observed in latent features at two different

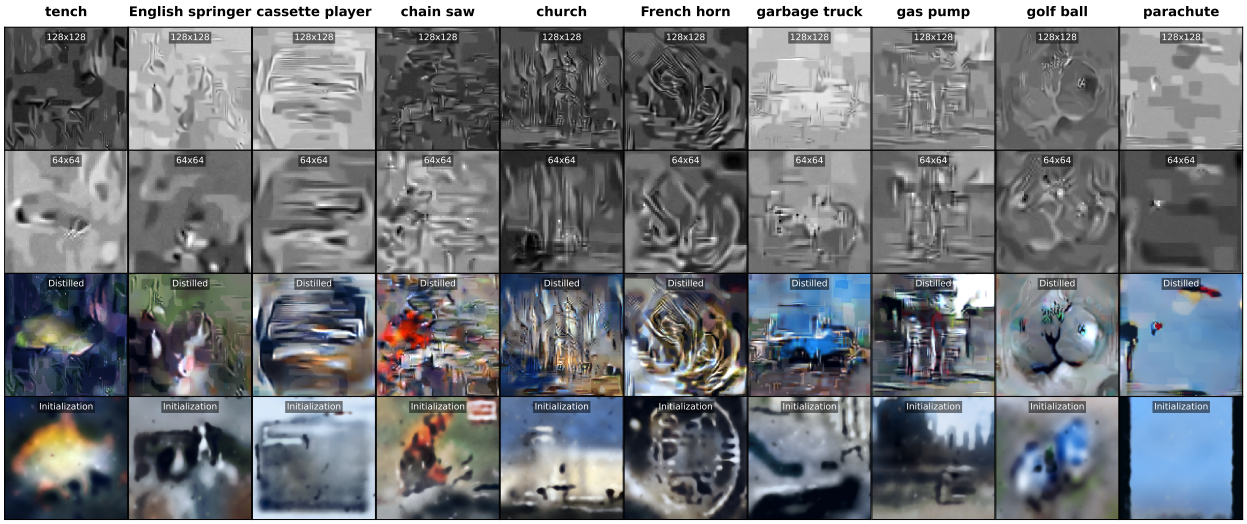


Figure 5. Visualization of synthetic data from ImageNet-Subset Nette. The first and second rows represent latent features $\mathbf{z}$ at different scales, the third row shows the synthetic data, and the fourth row displays the initialized information.

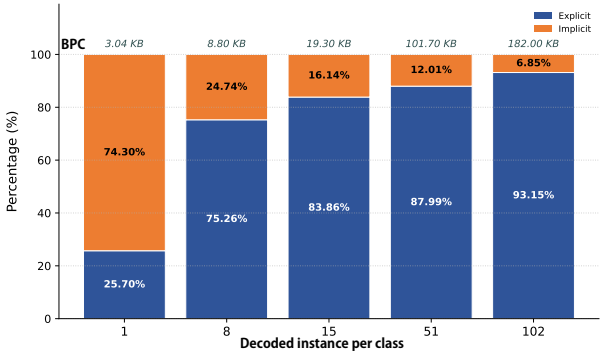


Figure 6. The distribution of explicit and implicit bit rates across varying configurations of decoded instance counts on the ImageNet-sub Nette, and post-quantization mse error $< 5\times10^{-7}$.

Table 6. Performance comparison on Nette when the **number of parameters per class is similar**.

Method	Para./class($\times10^6$)	BPC(KB)	DPIC	Acc(%)
Vanilla [6]	2.51	9792	51	73.0
DDiF [37]	2.46	9592	697	75.2
Ours	**2.24**	**180**	**102**	**76.5**

scales, with significant inter-category differences in contour characteristics. This indicates that the explicit component successfully captures key structural features between categories (e.g., fundamental contours) and establishes fundamental category representations by encoding key semantic features such as object shapes. After implicit synthesis, the feature space not only preserves basic semantics but also incorporates more discriminative detailed features (e.g., category-specific elements like church spires). This demonstrates that the implicit component injects rich details to ensure sample diversity in synthesized data.

The Distribution of Bits. The explicit component encodes the critical information of data, while the implicit component focuses on synthesizing data to maintain diversity. As shown in Figure 6, the bitrate allocation between explicit and implicit components exhibits significant differences with varying BPC. In low-BPC scenarios (e.g., BPC=3.04KB), the explicit component accounts for only 25.70%, indicating that limited data bits necessitate reliance on the implicit component to generate sample diversity. As BPC increases, the explicit component rapidly dominates (e.g., reaching 75.26% at BPC=8KB), reflecting enhanced capacity to encode task-critical features. In high-BPC scenarios (e.g., BPC=182KB), the explicit component overwhelmingly dominates (93.15%), while the implicit component diminishes to 6.85%, demonstrating that abundant data enables near-complete reliance on explicit features to convey task-critical patterns. This trend reveals optimization directions: in low-BPC regimes, balancing explicit and implicit components is crucial for generalization, whereas in high-BPC scenarios, prioritizing explicit feature compression maximizes efficiency.

5. Conclusion

This work redefined dataset distillation from a rate-utility perspective, proposing a unified framework that deeply integrates the fields of data compression and dataset distillation for the first time. We designed a unified representation comprising latent grids and a parametric decoder, which decouples sample-specific information (encoded in latent space) from domain knowledge (encoded in synthesis network). By explicitly modeling the rate of synthetic datasets as an optimizable metric, we established an end-

to-end trainable framework for joint rate-utility optimization, bridging the gap between data compression and dataset distillation. We introduced the BPC metric, providing a foundation for fair comparisons of rate-utility trade-offs. Our framework achieves SOTA dataset distillation performance on CIFAR-10/100 and ImageNet-128. For instance, on ImageNet-Subset, our method delivers a 170× compression rate improvement over the Vanilla while maintaining comparable accuracy.

The current framework handles network parameter bitrate compression through post-quantization. Future work could unify network parameter compression and explore broader data modalities and more compression techniques.

References

[1] Eirikur Agustsson, David Minnen, George Toderici, and Fabian Mentzer. Multi-realism image compression with a conditional generator. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22324–22333, 2023. 3

[2] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. End-to-end optimized image compression. In *International Conference on Learning Representations*, 2017. 2, 3

[3] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *International Conference on Learning Representations*, 2018. 3, 4

[4] Yoshua Bengio and Yann LeCun. Scaling learning algorithms towards ai. *Large-scale kernel machines*, 34(5):1–41, 2007. 1

[5] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, pages 1877–1901, 2020. 1

[6] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A. Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 4749–4758. IEEE, 2022. 1, 2, 4, 6, 7, 8

[7] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A. Efros, and Jun-Yan Zhu. Generalizing dataset distillation via deep generative prior. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3739–3748, 2023. 2, 4, 6, 7

[8] Mingyang Chen, Jiawei Du, Bo Huang, Yi Wang, Xiaobo Zhang, and Wei Wang. Influence-guided diffusion for dataset distillation. In *International Conference on Learning Representations*, 2025. 2, 4

[9] Zhiwei Deng and Olga Russakovsky. Remember the past: distilling datasets into addressable memories for neural networks. In *Advances in Neural Information Processing Systems*, 2022. 3, 6

[10] HN Dheemanth. Lzw data compression. *American journal of engineering research*, 3(2):22–26, 2014. 3

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 6

[12] Yuxuan Duan, Jianfu Zhang, and Liqing Zhang. Dataset distillation in latent space. *ArXiv preprint*, abs/2311.15547, 2023. 4

[13] Emilien Dupont, Adam Golinski, Milad Alizadeh, Yee Whye Teh, and Arnaud Doucet. COIN: COmpression with implicit neural representations. In *International Conference on Learning Representations Workshops*, 2021. 3

[14] Yunzhen Feng, Shanmukha Ramakrishna Vedantam, and Julia Kempe. Embarrassingly simple dataset distillation. In *The Twelfth International Conference on Learning Representations*, 2024. 1, 2

[15] Lingyu Gu, Yongqi Du, Yuan Zhang, Di Xie, Shiliang Pu, Robert C. Qiu, and Zhenyu Liao. "lossless" compression of deep neural networks: A high-dimensional neural tangent kernel approach. In *Advances in Neural Information Processing Systems*, 2022. 3

[16] Zongyu Guo, Gergely Flamich, Jiajun He, Zhibo Chen, and José Miguel Hernández-Lobato. Compression with bayesian implicit neural representations. In *Advances in Neural Information Processing Systems*, 2023. 3

[17] Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *ACM Symposium on Theory of Computing*, pages 291–300, 2004. 2

[18] Dailan He, Ziming Yang, Weikun Peng, Rui Ma, Hongwei Qin, and Yan Wang. ELIC: efficient learned image compression with unevenly grouped space-channel contextual adaptive coding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5708–5717, 2022. 3

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 6

[20] Ning Kang, Shanzhao Qiu, Shifeng Zhang, Zhenguo Li, and Shutao Xia. PILC: practical image lossless compression with an end-to-end GPU oriented neural framework. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3729–3738, 2022. 3

[21] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeff Wu, and Dario Amodei. Scaling laws for neural language models. *ArXiv preprint*, abs/2001.08361, 2020. 1

[22] Hyunjik Kim, Matthias Bauer, Lucas Theis, Jonathan Richard Schwarz, and Emilien Dupont. C3: high-performance and low-complexity neural compression from a single image or video. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9347–9358, 2024. 3, 4, 5

[23] 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*, pages 11102–11118, 2022. 2, 4, 6, 7

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114, 2012. 6

[25] Théo Ladune, Pierrick Philippe, Félix Henry, Gordon Clare, and Thomas Leguay. COOL-CHIC: coordinate-based low complexity hierarchical image codec. In *IEEE/CVF International Conference on Computer Vision*, pages 13469–13476, 2023. 4

[26] Shiye Lei and Dacheng Tao. A comprehensive survey of dataset distillation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(1):17–32, 2023. 1

[27] Mu Li, Kede Ma, Jane You, David Zhang, and Wangmeng Zuo. Efficient and effective context-based convolutional entropy modeling for image compression. *IEEE Transactions on Image Processing*, 29:5900–5911, 2020. 3

[28] Songhua Liu, Kai Wang, Xingyi Yang, Jingwen Ye, and Xinchao Wang. Dataset distillation via factorization. In *Advances in Neural Information Processing Systems*, 2022. 4, 6

[29] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11966–11976, 2022. 6

[30] Noel Loo, Ramin M. Hasani, Mathias Lechner, and Daniela Rus. Dataset distillation with convexified implicit gradients. In *International Conference on Machine Learning*, pages 22649–22674, 2023. 2

[31] Fabian Mentzer, George Toderici, Michael Tschannen, and Eirikur Agustsson. High-fidelity generative image compression. In *Advances in Neural Information Processing Systems*, 2020. 3

[32] David Minnen and Saurabh Singh. Channel-wise autoregressive entropy models for learned image compression. In *International Conference on Image Processing*, pages 3339–3343, 2020. 3

[33] Alistair Moffat. Huffman coding. *ACM Computing Surveys (CSUR)*, 52(4):1–35, 2019. 3

[34] Brian B. Moser, Federico Raue, Sebastian Palacio, Stanislav Frolov, and Andreas Dengel. Latent dataset distillation with diffusion models. *ArXiv preprint*, abs/2403.03881, 2024. 4, 7

[35] Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. Dataset meta-learning from kernel ridge-regression. In *International Conference on Learning Representations*, 2021. 1

[36] DongHyeok Shin, Seungjae Shin, and Il-Chul Moon. Frequency domain-based dataset distillation. In *Advances in Neural Information Processing Systems*, 2023. 2, 4, 6, 7

[37] Donghyeok Shin, HeeSun Bae, Gyuwon Sim, Wanmo Kang, and Il chul Moon. Distilling dataset into neural field. In *International Conference on Learning Representations*, 2025. 4, 6, 7, 8

[38] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 6

[39] Duo Su, Junjie Hou, Weizhi Gao, Ying jie Tian, and Bowen Tang. D4m: Dataset distillation via disentangled diffusion model. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5809–5818, 2024. 2, 4

[40] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *ArXiv preprint*, abs/1811.10959, 2018. 1, 2, 4

[41] Xiao Wang, Ibrahim Alabdulmohsin, Daniel Salz, Zhe Li, Keran Rong, and Xiaohua Zhai. Scaling pre-training to one hundred billion data for vision language models. *ArXiv preprint*, abs/2502.07617, 2025. 1

[42] Xing Wei, Anjia Cao, Funing Yang, and Zhiheng Ma. Sparse parameterization for epitomic dataset distillation. In *Advances in Neural Information Processing Systems*, 2023. 2, 4, 6

[43] P.J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990. 1

[44] Ruihan Yang and Stephan Mandt. Lossy image compression with conditional diffusion models. In *Advances in Neural Information Processing Systems*, 2023. 3

[45] 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*, pages 275–290, 2024. 2, 4, 6

[46] Yibo Yang, Stephan Mandt, Lucas Theis, et al. An introduction to neural data compression. *Foundations and Trends® in Computer Graphics and Vision*, 15(2):113–200, 2023. 3

[47] Jiaqing Zhang, Mingjia Yin, Hao Wang, Yawen Li, Yuyang Ye, Xingyu Lou, Junping Du, and Enhong Chen. Td3: Tucker decomposition based dataset distillation method for sequential recommendation. In *Proceedings of the ACM on Web Conference*, page 3994–4003, 2025. 2, 4

[48] Zhe Zhang, Huairui Wang, Zhenzhong Chen, and Shan Liu. Learned lossless image compression based on bit plane slicing. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 27569–27578, 2024. 3

[49] Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6514–6523, 2023. 1, 2, 4, 6, 7

[50] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *International Conference on Learning Representations*, 2021. 2, 6, 7

[51] Haizhong Zheng, Jiachen Sun, Shutong Wu, Bhavya Kailkhura, Z Morley Mao, Chaowei Xiao, and Atul Prakash. Leveraging hierarchical feature sharing for efficient dataset condensation. In *European Conference on Computer Vision*, pages 166–182, 2024. 6

[52] Xinhao Zhong, Hao Fang, Bin Chen, Xulin Gu, Tao Dai, Meikang Qiu, and Shu-Tao Xia. Hierarchical features matter: A deep exploration of gan priors for improved dataset distillation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. 2, 4, 6, 7

[53] Yongchao Zhou, Ehsan Nezhadarya, and Jimmy Ba. Dataset distillation using neural feature regression. In *Advances in Neural Information Processing Systems*, 2022. 4, 6