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, chenzh08@pcl.ac.cn
{liangys,limu2022}@hit.edu.cn, kede.ma@cityu.edu.hk
https://nouise.github.io/DD-RUO

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 an original dataset into a small set of synthetic samples, while preserving its full utility. Yet, existing methods either maximize performance under fixed storage budgets or pursue suitable synthetic data representations for redundancy removal, without jointly optimizing both objectives. In this work, we propose a joint rate-utility optimization method for dataset distillation. We parameterize synthetic samples as optimizable latent codes decoded by extremely lightweight networks. We estimate the Shannon 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 parameter 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 better rate-utility trade-offs.*

1. Introduction

The rapid advances in machine learning have been driven by the "scale-is-everything" paradigm [23, 25], in which ever-larger models trained on ever-more data yield progressively better performance across various computational prediction tasks [8, 16]. However, this scaling trend incurs steep computational, storage, and environmental costs, posing a barrier to sustainable and accessible research and deployment.

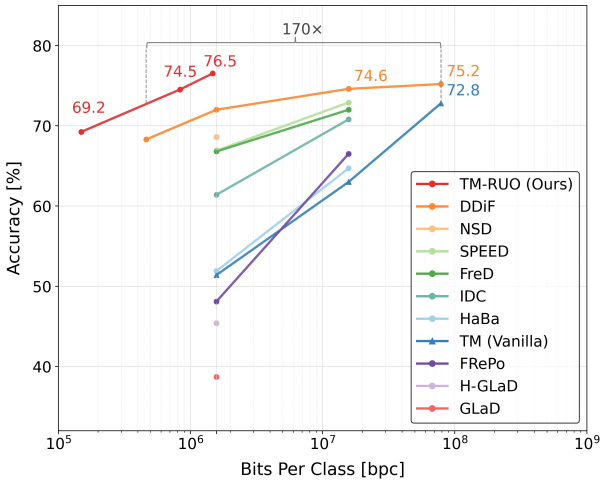


Figure 1. Comparison of the rate-utility curves on the Nette subset of ImageNet, with the rate axis displayed on a logarithmic scale. The proposed TM-RUO that integrates the trajectory matching loss [9] consistently achieves superior accuracy across a wide range of storage budgets, as measured in bits per class (bpc).

Dataset distillation (DD) [49] addresses these challenges by learning a small, synthetic dataset that enables rapid model prototyping, accelerated model training, and efficient hyperparameter tuning, while lowering compute, storage, and energy requirements [31, 33].

Initially, DD is formulated as a bilevel optimization problem [49], necessitating computationally intensive, memory-hungry backpropagation through time (BPTT), which is also prone to gradient instability [51]. To alleviate this, meta-gradient computation has been pursued or bypassed through a variety of techniques, including truncated BPTT [18], closed-form kernel ridge regression [38], gradient and distribution matching [60, 61], and implicit differentiation [35]. Meanwhile, representation-based methods [10, 27, 43, 50, 54, 58] mitigate redundancy in synthetic data by parameterizing it as low-dimensional latent codes, which can be reconstructed into full-sized images via struc-

*Equal contribution.
†Corresponding authors.

tured or generative decoders.

At the core of DD lies a trade-off between the *rate* (*i.e.*, the storage footprint of synthetic datasets) and *utility* (*i.e.*, the performance of models trained on distilled data). Despite impressive progress, existing DD methods optimize rate and utility separately, precluding Pareto-optimal trade-offs. Inspired by the rate-distortion theory that governs data compression [5, 12], we propose a joint rate-utility optimization method for DD. We parameterize synthetic samples as a set of multiscale optimizable latent codes that can be decoded by extremely lightweight networks. To compute the distillation *rate*, we model each *quantized* latent code with a context-aware Laplace distribution whose mean and scale parameters are inferred from its causal neighbors [26]. We then approximate the rate using the Shannon entropy under this Laplace prior. To assess the distillation *utility*, we leverage the plug-and-play property of our method by incorporating any off-the-shelf distillation loss [9, 60, 61]. Finally, we optimize all parameters end-to-end against the joint rate-utility objective, balancing the two terms via a tunable Lagrange multiplier [5].

To facilitate equitable comparisons across DD methods of different design philosophies, we introduce the bits per class (bpc) metric, which quantifies the average number of bits needed to store a distilled dataset—including sample representations, class labels, and decoder parameters (if any)—on a per-class basis (see Fig. 1). In contrast, the widely adopted images per class (ipc) metric scales linearly with raw image dimensions and overlooks both the bit cost of encoding labels and decoder parameters.

In summary, our key contributions include

- A joint rate-utility optimization method for DD that retains full compatibility with existing distillation losses;
- A bpc metric that enables standardized and fair rate-utility comparisons across various DD methods;
- An experimental demonstration on CIFAR-10, CIFAR-100, and ImageNet-128 that our method achieves better rate-utility trade-offs across various storage budgets, distillation losses, and network architectures.

2. Related Work

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

2.1. Dataset Distillation (DD)

Wang *et al*. [49] first formulated DD as a bilevel optimization problem, which requires BPTT, incurring prohibitive computation due to long unrolls and nested differentiation. To reduce this burden, several meta-gradient estimation strategies have been proposed. Feng *et al*. [18] introduced random truncated BPTT to cap the number of unrolled steps, while Nguyen *et al*. [39] replaced the iterative inner-loop solver with closed-form kernel ridge regression.

Intuitively, aligning the training dynamics of original and synthetic datasets provides more granular supervisory signals than merely measuring final performance. As specific instances, gradient matching [61] aligns instantaneous parameter updates, whereas trajectory matching [9] extends this alignment across the entire sequence. By aligning distributions in a predefined feature space, distribution matching [60] bypasses the challenge of measuring functional similarity in parameter space and avoids costly second-order gradient computations.

More recent work focuses on decoupling or reframing the bilevel structure to handle large-scale, high-resolution datasets. The Teddy method [57] applies first-order Taylor expansion to convert nested gradients into sums of inner products, equivalent to matching feature-space means and variances under mild assumptions. SRe2L [56] decomposes DD into three sequential stages: 1) train a teacher network on original data, 2) optimize synthetic samples by matching the BatchNorm statistics and outputs against the frozen teacher, and 3) relabel the synthetic samples.

In parallel, representation-based DD methods have been devised to reduce the storage footprint of synthetic data. Explicit (*i.e.*, code-centric) approaches learn low-dimensional (latent) codes with fixed decoders—implemented via discrete cosine transform (DCT) [43], Tucker decomposition [58], or deep generative priors such as generative adversarial networks (GANs) [10] and diffusion models [11, 46, 63][1]. Implicit (*i.e.*, decoder-centric) methods encode synthetic samples solely in decoder parameters given random latent inputs [44]. Hybrid schemes optimize both latents and decoders for improved performance [14, 34].

However, to the best of our knowledge, no existing DD method formalizes storage cost using information-theoretic rate measures (*e.g.*, Shannon entropy), nor incorporates such rate considerations into a joint end-to-end optimization of storage and utility in DD.

2.2. Data Compression

Data compression is generally divided into two complementary paradigms: lossless and lossy compression. Lossless methods aim to encode data exactly, exploiting statistical redundancies via entropy coding (*e.g.*, Huffman coding [24] and arithmetic coding [52]) in concert with probability models. Classic schemes such as dictionary-based algorithms (*e.g.*, LZW [65]) and transform coding remain popular in applications for their reliability and low complexity. Modern approaches increasingly integrate learnable generative models to capture high-dimensional, long-

[1]The vanilla DD [49] can also be regarded as an explicit scheme, where the decoder reduces to the identity mapping.

range dependencies: Auto-regressive neural networks [47], latent-variable models [28], and invertible flow architectures [15] can be paired with entropy coders to approach the information-theoretic limits [55].

By contrast, lossy compression introduces a controlled distortion to achieve greater bitrate savings. Rate-distortion theory formalizes this trade-off by minimizing a weighted sum of expected bitrate and distortion [42]. Traditional pipelines apply fixed linear transforms (such as DCT or wavelets [3]), followed by quantization and entropy coding. Neural lossy compression [5] replaces these handcrafted transforms with trainable encoders and decoders[2], augmented by learned quantization schemes and by rich entropy models combining hyperpriors [6] with autoregressive [37] and discretized mixture-of-likelihood [32] contexts. Moreover, advanced transform modules (*e.g.*, generalized divisive normalization [4], residual connection [22], and non-local attention [48]) for improved latent representation, GAN-based distortion measures for perceptual realism [36], and diffusion-based decoders for fine-grained reconstruction [2, 53] have each pushed the frontier of rate-distortion performance.

Instead of relying on a single, shared decoder, recent "overfitted" compression methods [17, 21, 26] tailor both the synthesis and entropy networks to each image, storing their quantized weights alongside the latent codes. Although it incurs per-sample training overhead, this approach produces neural codecs that are both highly efficient and straightforward to deploy, striking a compelling balance between compression efficacy and run-time simplicity. Consequently, it is especially well suited for representing synthetic data and for obtaining precise rate measurements in DD, as demonstrated in our study.

Although DD and data compression aim for distinct goals, they both seek minimum-entropy representations that preserve only the information essential for downstream tasks. We exploit this commonality by embedding differentiable quantization, entropy modeling, and rate-distortion optimization from neural lossy compression into our method, enabling joint end-to-end optimization of rate and utility in DD.

3. Proposed Method

In this section, we present our DD method through the lens of joint rate-utility optimization (see Fig. 2). We first introduce preliminaries in Sec. 3.1 and give our general problem formulation in Sec. 3.2. We then derive a differentiable rate term tailored to synthetic datasets and embed it in our joint objective in Secs. 3.3 and 3.4. Finally, Sec. 3.5 defines a precise bpc metric for standardized evaluation.

[2]In signal processing, the encoder and decoder are referred to as analysis and synthesis transforms, respectively.

3.1. Preliminaries

Let $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{M}$ denote the original dataset of M labeled examples and let $\mathcal{S} = \{(\bar{x}^{(i)}, \bar{y}^{(i)})\}_{i=1}^{N}$ be a much smaller synthetic dataset with $N \leq M$. The goal of DD is to learn $\mathcal{S}$ so that a model trained on $\mathcal{S}$ achieves performance close to one trained on $\mathcal{D}$. Traditionally, this is cast as a bilevel optimization problem:

$$\begin{aligned} &\min_{\mathcal{S}} \ \ell_{\text{outer}}\left(\mathcal{D}; \theta^{\star}(\mathcal{S})\right) \\ &\text{subject to } \theta^{\star}(\mathcal{S}) = \arg\min_{\theta} \ \ell_{\text{inner}}(\mathcal{S}; \theta), \end{aligned} \tag{1}$$

where

$$\ell_{\text{inner}}(\mathcal{S}; \theta) = \mathbb{E}_{(\bar{x}, \bar{y}) \sim \mathcal{S}}\left[\ell_{\text{inner}}\left(f\left(\bar{x}; \theta\right), \bar{y}\right)\right] \tag{2}$$

is the training loss (*e.g.*, cross-entropy) on the synthetic dataset, and $f(\cdot; \theta)$ is a differentiable classifier (*e.g.*, a neural network), parameterized by θ. $\theta^{\star}(\mathcal{S})$ is typically found by unrolling T gradient steps, with an initial guess $\theta^{(0)}$:

$$\theta^{(t)} = \theta^{(t-1)} - \eta \nabla_{\theta} \ell_{\text{inner}}(\mathcal{S}; \theta^{(t-1)}), \ t = 1, \ldots, T, \tag{3}$$

where η is the learning rate and $\theta^{(T)}$ serves as an approximation of the true optimum $\theta^{\star}(\mathcal{S})$. Finally, ℓ_{outer} measures how well $f(\cdot; \theta^{\star}(\mathcal{S}))$ generalizes to the original data $\mathcal{D}$.

Solving Problem (1) via BPTT incurs substantial computational and memory overhead. To alleviate this, one can truncate the iterative inner-loop solver [18], replace it with closed-form surrogates [39], or decouple the dependence of the optimal parameters $\theta^{\star}$ from the synthetic dataset $\mathcal{S}$ [56, 57]. To reduce storage requirements, the synthetic dataset itself may be parameterized by a collection of latent codes $\mathcal{Z} = \{\bar{z}^{(i)}\}_{i=1}^{N}$ and class labels $\mathcal{Y} = \{\bar{y}^{(i)}\}_{i=1}^{N}$, together with lightweight decoders $g(\cdot, \Psi)$ with parameters $\Psi = \{\bar{\psi}^{(i)}\}_{i=1}^{N}$:

$$\mathcal{S}\left(\mathcal{Z}, \mathcal{Y}, \Psi\right) = \left\{\left(g\left(\bar{z}^{(i)}; \bar{\psi}^{(i)}\right), \bar{y}^{(i)}\right)\right\}_{i=1}^{N}. \tag{4}$$

Prior methods enforce a hard constraint on pixel-based storage budget (*i.e.*, #pixels $\times$ bit-depth) but do not explicitly model the information-theoretic cost of encoding these latents, labels, and decoder parameters.

3.2. Problem Formulation

Inspired by the rate-distortion theory [12, 42], we propose to cast DD as a joint rate-utility optimization problem. Specifically, we introduce a differentiable rate term $r(\mathcal{S})$, and our objective balances this cost against any choice of distillation loss $\ell(\mathcal{D}; \mathcal{S})$ on the original data:

$$\min_{\mathcal{S}} \ r(\mathcal{S}) + \lambda \ell(\mathcal{D}; \mathcal{S}), \tag{5}$$

where $\lambda > 0$ governs the trade-off between the two terms. In this study, we instantiate $\ell(\mathcal{D}; \mathcal{S})$ in three different ways.

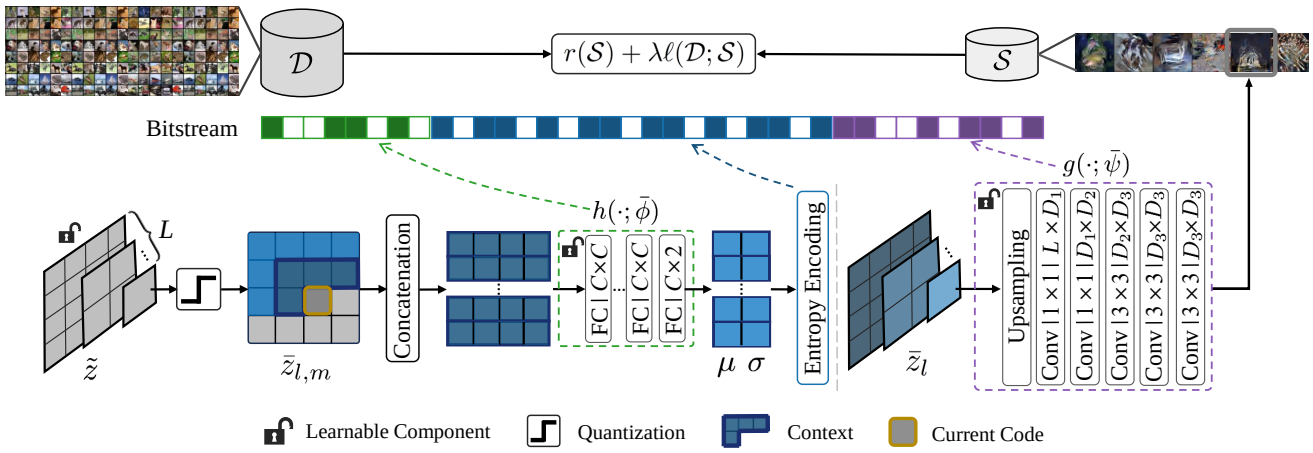


Figure 2. System diagram of our proposed joint rate-utility optimization method for DD. "FC" denotes a fully connected layer whose parameter dimensions are given by the product of input width and output width, and C represents the context length. Convolution layers are specified using the format: "kernel height $\times$ kernel width | input channels $\times$ output channels."

- *Gradient matching* [61]. We align minibatch gradients computed on $\mathcal{S}$ with those on $\mathcal{D}$ at each training step:

$$\ell_{\text{grad}}(\mathcal{D};\mathcal{S}) = \sum_{t=0}^{T} \left\| \nabla_\theta \ell\left(\mathcal{D};\theta^{(t)}\right) - \nabla_\theta \ell\left(\mathcal{S};\theta^{(t)}\right) \right\|_2^2, \tag{6}$$

  where $\{\theta^{(t)}\}_{t=0}^{T}$ are obtained by training on $\mathcal{S}$.
- *Trajectory matching* [9]. We encourage the training trajectory on $\mathcal{S}$ to follow an "expert" trajectory on $\mathcal{D}$:

$$\ell_{\text{traj}}(\mathcal{D};\mathcal{S}) = \sum_{t=0}^{T} \frac{\left\|\theta^{(t+t_2)}(\mathcal{D}) - \theta^{(t+t_1)}(\mathcal{S})\right\|_2^2}{\left\|\theta^{(t+t_2)}(\mathcal{D}) - \theta^{(t)}(\mathcal{D})\right\|_2^2}, \tag{7}$$

  where $t_1 < t_2$ and the denominator, as a form of normalization, ensures that the loss measures relative alignment rather than absolute magnitude.
- *Distribution matching* [60]. We match the feature-space distributions over $\mathcal{D}$ and $\mathcal{S}$:

$$\ell_{\text{dist}}(\mathcal{D};\mathcal{S}) = d\left(\mathbb{E}_{x\sim\mathcal{D}}\left[\varphi(x)\right], \mathbb{E}_{\bar{x}\sim\mathcal{S}}\left[\varphi(\bar{x})\right]\right), \tag{8}$$

  where $\varphi(\cdot)$ denotes a fixed feature extractor and $d(\cdot,\cdot)$ is a statistical distance (*e.g.*, maximum mean discrepancy [20]).

By integrating rate and utility into a single end-to-end objective, our method enables systematic exploration of distilled datasets that optimally trade off storage cost against generalization performance.

3.3. Rate Modeling

In our joint rate-utility optimization method, we employ a hybrid parameterization [30, 34] of the synthetic dataset $\mathcal{S}(\mathcal{Z},\mathcal{Y},\Psi)$ in Eq. (4), where each synthetic sample $\bar{x}^{(i)} \in \mathcal{S}$ is generated from a set of latent codes at L scales $\{\bar{z}_l^{(i)}\}_{l=1}^{L} \subset \mathcal{Z}$ via a lightweight decoder $g\left(\cdot;\bar{\psi}^{(i)}\right)$ with $\bar{\psi}^{(i)} \subset \Psi$ and annotated with a class label $\bar{y}^{(i)} \in \mathcal{Y}$. We decompose the total bitrate into three additive terms:

$$r(\mathcal{S}) = r(\mathcal{Z}) + r(\mathcal{Y}) + r(\Psi). \tag{9}$$

Below, we describe how each term is modeled.

Bitrate for Latent Codes. We arrange the *continuous* latent code $\tilde{z}$ at L scales[3] so that $\tilde{z} = \{\tilde{z}_l\}_{l=1}^{L}$, where $\tilde{z}_l \in \mathbb{R}^{\lfloor \frac{H}{2^{l-1}} \rfloor \times \lfloor \frac{W}{2^{l-1}} \rfloor}$, and H and W are the height and width of the original image. To enable entropy coding, the m-th latent code at the l-th scale, $\tilde{z}_{l,m}$, is first quantized by rounding to the nearest integer:

$$\bar{z}_{l,m} = \text{round}(\tilde{z}_{l,m}) = \left\lfloor \tilde{z}_{l,m} + \frac{1}{2} \right\rfloor. \tag{10}$$

We then fit a lightweight, sample-specific, and spatially autoregressive entropy model:

$$P(\bar{z}_{l,m}|\bar{c}_{l,m}) = \int_{\bar{z}_{l,m}-\frac{1}{2}}^{\bar{z}_{l,m}+\frac{1}{2}} p(\xi|\bar{c}_{l,m})\,\mathrm{d}\xi, \tag{11}$$

where the context $\bar{c}_{l,m}$ aggregates previously decoded neighbors, and

$$p(\xi|\bar{c}_{l,m}) = \text{Laplace}(\xi;\mu_{l,m},\sigma_{l,m}), \tag{12}$$

with $\mu_{l,m},\sigma_{l,m} = h(\bar{c}_{l,m};\phi)$ produced by an auxiliary entropy network with *continuous* parameters ϕ. Under the assumption of independence across channels and scales, the bitrate of the latent codes $\mathcal{Z}$ can be computed by

$$r\left(\mathcal{Z};\Phi\right) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{l,m} \log_2 P\left(\bar{z}_{l,m}^{(i)}\middle|\bar{c}_{l,m}^{(i)}\right). \tag{13}$$

[3]To improve notational clarity, we omit the superscript image index from the mathematical expressions.

Algorithm 1 Joint Rate-Utility Optimization for DD

Phase 1: Initialization

Randomly initialize $\{\tilde{z}^{(i)}, \phi^{(i)}, \psi^{(i)}, \bar{y}^{(i)}\}_{i=1}^{N}$
for $i = 1$ to N **do** ▷ in parallel
  Randomly sample an original pair (x, y) in $\mathcal{D}$
  Optimize $\{\tilde{z}^{(i)}, \phi^{(i)}, \psi^{(i)}\}$ by solving Eq. (22)
  $\bar{y}^{(i)} \leftarrow y$
end for

Phase 2: Joint Rate-Utility Optimization

repeat
  Sample an original mini-batch from $\mathcal{D}$ and a synthetic mini-batch from $\mathcal{S}(\mathcal{Z}, \Phi, \Psi, \mathcal{Y})$
  Compute the total rate-utility loss on the sampled mini-batches using Eq. (20)
  Update parameters $\{\mathcal{Z}, \Phi, \Psi\}$ via gradient descent
until convergence

Phase 3: Post-Quantization

Apply post-quantization to the parameters of the entropy and decoder networks

To account for the additional storage overhead of the entropy network parameters ϕ, which are represented as 32-bit floats during training, we apply a uniform, post-training quantization, as suggested in [26, 30]. A step size Q_e is selected by grid-search to balance bitrate against reconstruction error (as measured by the mean squared error). Concretely, the j-th parameter ϕ_j is quantized as

$$\bar{\phi}_j = q\left(\phi_j; Q_e\right) = \text{round}\left(\frac{\phi_j}{Q_e}\right) \times Q_e. \tag{14}$$

The quantized weights $\bar{\phi} \subset \Phi$ are then encoded losslessly under a discretized, zero-mean Laplace prior, whose scale is set to $\text{std}(\bar{\phi})/\sqrt{2}$. Averaging over N entropy network instances, the resulting bitrate becomes

$$r(\Phi) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j}\log_2 P\left(\bar{\phi}_j^{(i)}\right). \tag{15}$$

Bitrate for Class Labels. Let $\mathcal{Y} = \{\bar{y}^{(i)}\}_{i=1}^{N}$ denote the class labels of our N synthetic samples, each taking one of K discrete categories with probabilities $\{\bar{y}_k^{(i)}\}_{k=1}^{K}$, where $\bar{y}_k^{(i)} \geq 0$ and $\sum_k \bar{y}_k^{(i)} = 1$. Under Shannon's source coding theorem [41], the minimal bitrate required to losslessly encode $\mathcal{Y}$ is upper-bounded by

$$r(\mathcal{Y}) < \text{Entropy}(\bar{y}) + 1 \leq \log_2 K + 1, \tag{16}$$

where $\text{Entropy}(\bar{y}) = -\sum_k \bar{y}_k \log_2 \bar{y}_k$ is the Shannon entropy, and equality in the upper bound holds when the labels are uniformly distributed across all K classes. In many DD methods [13, 56], however, one deliberately employs *soft* labels, that is, probability vectors $\tilde{y} = [\tilde{y}_1, \ldots, \tilde{y}_K]$ lying in the simplex Δ^{K-1}. Because these vectors are continuous, vector quantization [19] should first be applied before entropy coding. For example, to guarantee that each coordinate $\tilde{y}_k$ is approximated to within ϵ, the $(K-1)$-simplex can be uniformly partitioned into on the order of $\epsilon^{-(K-1)}/(K-1)!$ bins. Hence, the resulting bitrate obeys

$$r(\mathcal{Y}) \lesssim \log_2(\#\text{bins}) = -\log_2(K-1)! - (K-1)\log_2\epsilon, \tag{17}$$

which makes it clear that storing soft labels at full floating-point precision can incur a dramatically higher cost than simply recording hard labels—and in extreme cases may even exceed the cost of transmitting the latent codes [33] (see Appendix A1 for detailed derivation).

Bitrate for Decoder Parameters. The decoder $g(\cdot;\psi)$ reconstructs each synthetic image $\bar{x}$ from its multiscale latent codes. Concretely, the latents $\{\bar{z}_l\}$ are first upsampled to the original resolution and concatenated into a tensor $\text{Up}(\bar{z}) \in \mathbb{R}^{H\times W\times L}$. $g(\cdot;\psi)$ then maps $\text{Up}(\bar{z})$ to $\bar{x}$, which is subsequently learned by the downstream classifier $f(\cdot;\theta)$ for utility measurement.

To render the decoder parameters ψ amenable to entropy coding, we apply the same uniform quantization used for the entropy network parameters, *i.e.*, $\bar{\psi}_j = q(\psi_j; Q_d)$, where the quantization step size Q_d is chosen by grid search. Averaging over N decoder instances under a discretized, zero-mean Laplace prior, the resulting bitrate is

$$r(\Psi) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j}\log_2 P\left(\bar{\psi}_j^{(i)}\right). \tag{18}$$

3.4. Joint Rate-Utility Optimization

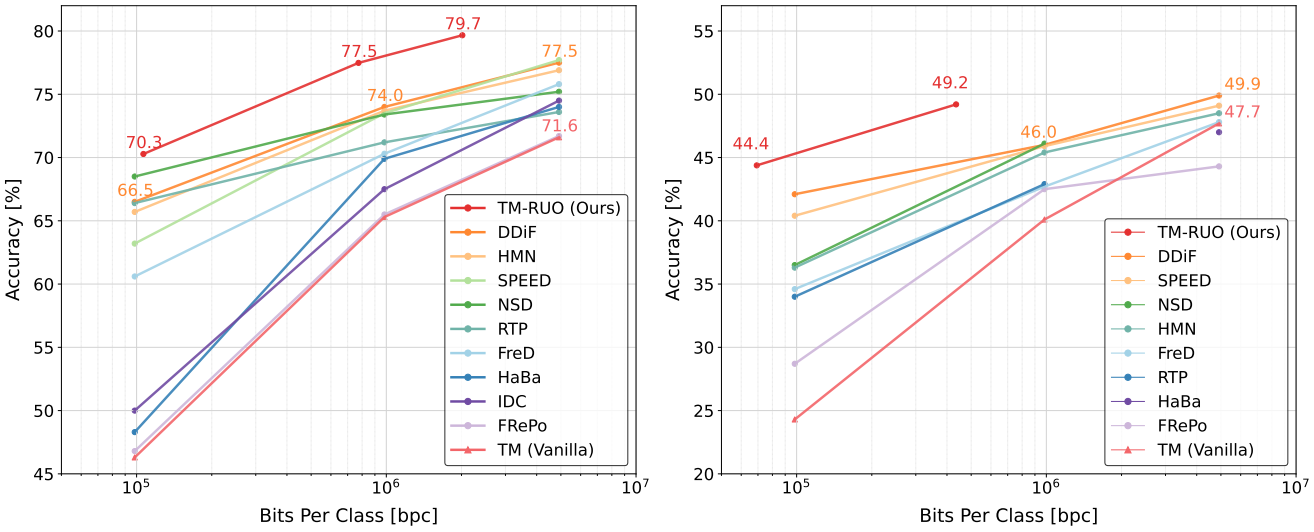
Plugging the decomposed rate terms in Eqs. (13), (15), (17), and (18) into the general formulation in Eq. (5), we can write the full joint rate-utility objective as

$$\min_{\mathcal{S}=\{\mathcal{Z},\mathcal{Y},\Phi,\Psi\}} r(\mathcal{Z};\Phi) + r(\mathcal{Y}) + r(\Phi) + r(\Psi) + \lambda\ell(\mathcal{D};\mathcal{S}). \tag{19}$$

To overcome the non-differentiability of quantization and enable end-to-end optimization, several continuous relaxations can be leveraged, including the straight-through estimator [7], additive uniform noise injection [5], and soft (annealed) quantization [1]. In our current implementation, we specialize this to a simplified objective:

$$\min_{\mathcal{Z},\Phi,\Psi} r(\mathcal{Z};\Phi) + \lambda\ell(\mathcal{D};\mathcal{Z},\Phi,\Psi), \tag{20}$$

omitting $r(\mathcal{Y})$, $r(\Phi)$, and $r(\Psi)$ from joint optimization for the following reasons. First, we employ hard (one-hot) labels for $\mathcal{Y}$, which can be treated as a constant additive offset. Second, both the entropy and decoder networks are architected to be extremely lightweight. We quantize



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

Figure 3. Comparison of the rate–utility curves on CIFAR-10 and CIFAR-100. Our proposed joint rate–utility optimization method achieves superior accuracy across a wide range of bpc budgets.

and entropy-encode their weights offline—via a small grid search on uniform step sizes—and thereby fix $r(\Phi)$ and $r(\Psi)$ as constant rate offsets. This decoupling avoids back-propagating through network quantization and considerably reduces computational overhead and gradient variance. Algorithm 1 sketches the entire procedure.

3.5. Bits Per Class (bpc) Metric

The widely used ipc metric—counting the number of distilled images per category—fails to capture the true storage footprint of a synthetic dataset. First, raw-pixel storage grows linearly with image height, width, channel count, and bit-depth; hence, a fixed ipc can correspond to vastly different bitrates across datasets or methods. Second, ipc ignores any auxiliary overhead, such as the bits required to encode decoder parameters or soft-label representations.

To provide a unit-consistent, information-theoretic measure of storage cost, we define the bpc metric. Let $\#\text{bits}(\mathcal{S})$ be the total number of bits needed to (losslessly) encode the distilled dataset $\mathcal{S}$. If there are K distinct classes, then

$$\text{bpc}(\mathcal{S}) = \frac{\#\text{bits}(\mathcal{S})}{K}. \tag{21}$$

By averaging over classes, bpc directly reports the per-class storage requirement in bits, irrespective of image format, label representation (hard vs. soft), or decoder complexity. Thus, bpc enables fair comparisons across DD methods in trading off rate and utility.

4. Experiments

To validate the effectiveness of our joint rate–utility optimization method, we conduct extensive evaluations on CIFAR-10, CIFAR-100, and an ImageNet subset. We assess classification accuracy under varying bpc budgets, test robustness across network architectures, and verify compatibility with different distillation losses.

4.1. Experimental Setups

Datasets. We evaluate on CIFAR-10 and CIFAR-100 at 32×32 resolution, and on six subsets of ImageNet (at 128×128 resolution), namely Nette, Woof, Fruit, Yellow, Meow, and Squawk [44].

Baselines. Comparisons include TM [9], GLaD [10], H-GLaD [63], FRePo [64], HaBa [34], IDC [27], FreD [43], SPEED [50], RTP [14], NSD [54], HMN [62], and DDiF [44].

Implementation Details. We adopt the TM loss [9] as our default utility objective. To accelerate convergence, all parameters are initialized via an "overfitted" image compression pretraining [26] on the original dataset $\mathcal{D}$, with the following objective:

$$\min_{\mathcal{Z},\Phi,\Psi} r(\mathcal{Z};\Phi) + \beta \left\| g\left(z;\psi\right) - x \right\|_2^2, \tag{22}$$

where $z \in \mathcal{Z}$ is the latent code for input x, decoded by $g(\cdot;\psi)$ with $\psi \subset \Psi$. We then jointly optimize the entire method with Adam for the objective in Eq. (20) until convergence. We evaluate our approach by training downstream classifiers across five independent trials and reporting the mean performance. Further implementation details are provided in the Appendix.

Table 1. Classification accuracies (%) on the 128×128 ImageNet subsets (*i.e.*, Nette, Woof, Fruit, Yellow, Meow, and Squawk), evaluated under a bpc budget of $\leq$ 192 kB. The performance when trained on the original dataset is listed in the first row. A dash "–" denotes results not reported in the original work.

Method	ImageNet Subset (128×128)					
	Nette	Woof	Fruit	Yellow	Meow	Squawk
Original	87.4	67.0	63.9	84.4	66.7	87.5
TM (Vanilla) [9]	51.4	29.7	28.8	47.5	33.3	41.0
GLaD [10]	38.7	23.4	23.1	–	26.0	35.8
H-GLaD [63]	45.4	28.3	25.6	–	29.6	39.7
FRePo [64]	48.1	29.7	–	–	–	–
HaBa [34]	51.9	32.4	34.7	50.4	36.9	41.9
IDC [27]	61.4	34.5	38.0	56.5	39.5	50.2
FreD [43]	66.8	38.3	43.7	63.2	43.2	57.0
SPEED [50]	66.9	38.0	43.4	62.6	43.6	60.9
NSD [54]	68.6	35.2	39.8	61.0	45.2	52.9
DDiF [44]	72.0	42.9	48.2	69.0	47.4	67.0
TM-RUO (Ours)	**76.5**	**46.0**	**50.8**	**74.3**	**53.5**	**74.8**

Table 2. Classification accuracies (%) averaged across six 128×128 ImageNet subsets for different network architectures, including AlexNet, VGG-11, ResNet-18, and a variant of ViT-B, evaluated under a bpc budget of $\leq$ 192 kB.

Method	Classifier Architecture				Avg
	AlexNet	VGG-11	ResNet-18	ViT	
TM (Vanilla) [9]	10.8	14.5	29.0	19.7	18.5
IDC [27]	18.3	16.4	36.7	28.1	24.9
FreD [43]	24.3	16.9	40.0	32.9	28.5
DDiF [44]	**49.3**	38.6	49.6	**43.5**	45.2
TM-RUO (Ours)	46.7	**56.7**	**56.0**	43.4	**50.7**

4.2. Main Results

Results on CIFAR-10, CIFAR-100, and ImageNet Subsets. Fig. 3 shows the rate-utility curves on CIFAR-10 and CIFAR-100. Our method, coupled with the TM loss and termed as TM-RUO, consistently outperforms all competing methods, achieving 77.5% accuracy at 94 kB, representing up to a 2.5% absolute gain over the next best method. On CIFAR-100, TM-RUO attains 44.4% and 49.2% accuracy at only 8 kB and 53 kB, respectively, setting new state-of-the-art trade-offs. Similar conclusions can be drawn on ImageNet subsets (see Fig. 1 and Table 1).

Cross-architecture Generalization. To assess the generalizability of synthetic data across architectures, we train classifiers implemented by four widely used networks—AlexNet [29], VGG-11 [45], ResNet-18 [22], and a variant of ViT-B [16]—that differ from the network architecture employed during distillation. As shown in Table 2, under a stringent bitrate budget (*i.e.*, bpc $\leq$ 192 kB)[4], TM-RUO attains the highest accuracies across almost all architectures, with an average accuracy of 50.7%. These results demonstrate the robustness and adaptability of the proposed method in cross-architecture scenarios.

Table 3. Classification accuracies (%) averaged across six 128×128 ImageNet subsets for different utility losses, including gradient matching (GM) [61] and distribution matching (DM) [60], evaluated under a bpc budget of $\leq$ 192 kB. An asterisk "*" indicates results reproduced by us.

Method	ImageNet Subset (128×128)						Avg
	Nette	Woof	Fruit	Yellow	Meow	Squawk	
GM							
GM (Vanilla) [61]	34.2	22.5	21.0	37.1*	22.0	32.0	28.1
GLaD [10]	35.4	22.3	20.7	–	22.6	33.8	27.0
H-GLaD [63]	36.9	24.0	22.4	–	24.1	35.3	28.5
IDC [27]	45.4	25.5	26.8	–	25.3	34.6	31.5
FreD [43]	49.1	26.1	30.0	–	28.7	39.7	34.7
DDiF [44]	61.2	**35.2**	**37.8**	–	39.1	54.3	45.5
GM-RUO (Ours)	**67.2**	33.1	37.0	**58.9**	**42.0**	**58.6**	**49.5**
DM							
DM (Vanilla) [60]	30.4	20.7	20.4	36.0*	20.1	26.6	25.7
GLaD [10]	32.2	21.2	21.8	–	22.3	27.6	25.0
H-GLaD [63]	34.8	23.9	24.4	–	24.2	29.5	27.4
IDC [27]	48.3	27.0	29.9	–	30.5	38.8	34.9
FreD [43]	56.2	31.0	33.4	–	33.3	42.7	39.3
DDiF [44]	69.2	42.0	45.3	–	45.8	64.6	53.4
DM-RUO (Ours)	**71.9**	**46.4**	**49.0**	**69.2**	**48.8**	**69.0**	**59.1**

Compatibility with Multiple DD Losses. We also plug in gradient matching (GM) [61] and distribution matching (DM) [60] losses into our method. As shown in Table 3, GM-RUO and DM-RUO achieve 49.5% and 59.1%, respectively, both surpassing all competing methods and demonstrating broad compatibility.

Synthetic Sample Visualization. In Fig. 4, we visualize the optimized synthetic samples and their multiscale latent codes, alongside "overfitted" initialization from the Nette subset. It is clear that the explicit latents capture coarse-to-fine object contours, while the implicit synthesis injects category-specific details, validating the complementary roles of our hybrid representation.

Bit Allocation Analysis. Fig. 5 examines how explicit and implicit bits distribute across bpc regimes. As our entropy and decoder networks are held constant, at low bpc (*e.g.*, 3.04 kB) implicit bits dominate ($\approx$ 74%), whereas at high bpc (*e.g.*, 182 kB), explicit latents account for $\approx$ 93%. This trend suggests that tight budgets require balancing explicit diversity and implicit structure, while generous budgets benefit more from explicit feature compression.

[4]bpc = 192 kB corresponds to ipc = 1 of image dimensions $128 \times 128 \times 3$, where each pixel is represented at a 32-bit depth.

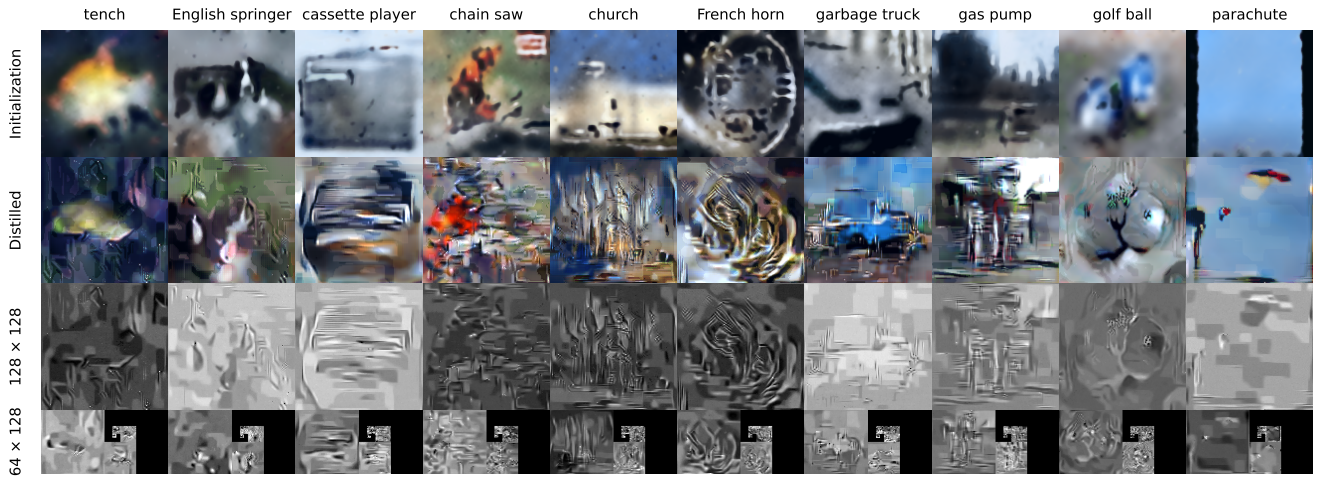


Figure 4. Visualization of synthetic samples on the Nette subset of ImageNet. The first row presents initialization by "overfitted" image compression, the second row shows distilled images (from the post-quantized decoder), and the third and fourth rows display the corresponding multiscale latent codes. Zoom in for improved visibility.

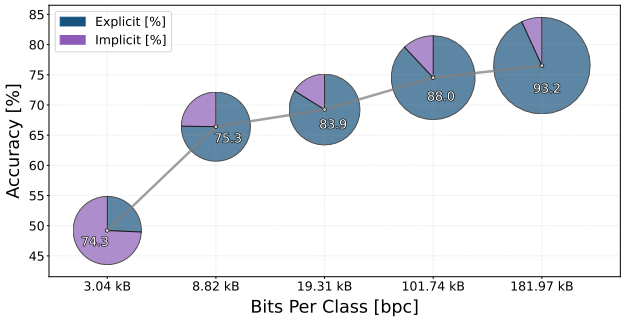


Figure 5. Bit allocation analysis across bpc regimes, with post-quantization error $< 5 \times 10^{-7}$ in terms of mean squared error.

5. Conclusion

In this work, we have reframed DD as a joint rate-utility optimization problem. By introducing a differentiable objective that incorporates the bitrate of latent codes, class labels, and decoder parameters alongside a task utility loss, we achieved state-of-the-art trade-offs on CIFAR-10, CIFAR-100, and ImageNet-128. Our multiscale latent representations, decoded via lightweight networks, not only decouple sample-specific details from shared network structures but also enable transparent comparison through a consistent bpc metric. Empirically, this formulation yields up to $170\times$ compression relative to vanilla distillation while maintaining—or even improving—test accuracy under tight storage budgets.

Looking ahead, we see three main avenues for extending this paradigm. First, fully joint optimization of soft labels and network architectures promises to tighten the rate-utility frontier: by learning continuous label embeddings rather than one-hot encodings, and by integrating differentiable neural architecture search to co-design entropy and decoder network topologies, one can further reduce bitrate without sacrificing accuracy. Second, scalability remains a practical bottleneck for high-resolution and large-scale datasets. Incorporating recent decoupling techniques, such as staged pipelines like SRe2L [56], will drastically cut computation and memory costs.

Finally, we believe our joint rate-utility optimization method can be generalized far beyond images. Extending to video, text, and graph modalities will require spatiotemporal or relational decoder architectures and entropy models attuned to temporal and structural dependencies. Moreover, adaptive schemes—where bit allocations and loss weights vary per sample, class, or downstream task—could enable conditional or multi-task distillation, dynamically trading storage for performance where it matters most. By pursuing these directions—soft label compression, neural architecture search, efficiency-focused decoupling tricks, and richer compression tools—we anticipate that rate-utility optimization will become a versatile foundation for data-efficient learning across domains.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (62472124), Hong Kong ITC Innovation and Technology Fund (9440379 and 9440390), Shenzhen Colleges and Universities Stable Support Program (GXWD20220811170130002), and Major Project of Guangdong Basic and Applied Basic Research (2023B0303000010).

References

[1] Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc Van Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. In *Advances in Neural Information Processing Systems*, pages 1141–1151, 2017. 5

[2] 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

[3] Nasir Ahmed, Raj Natarajan, and Kamisetty R. Rao. Discrete cosine transfom. *IEEE Transactions on Computers*, 23 (1):90–93, 1974. 3

[4] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. Density modeling of images using a generalized normalization transformation. In *International Conference on Learning Representations*, 2016. 3

[5] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. End-to-end optimized image compression. In *International Conference on Learning Representations*, 2017. 2, 3, 5

[6] 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

[7] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *ArXiv preprint arXiv:1308.3432*, 2013. 5

[8] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, pages 1877–1901, 2020. 1

[9] 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, 2022. 1, 2, 4, 6, 7, 5

[10] 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. 1, 2, 6, 7, 4

[11] 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

[12] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, USA, 2006. 2, 3

[13] Justin Cui, Ruochen Wang, Si Si, and Cho-Jui Hsieh. Scaling up dataset distillation to ImageNet-1K with constant memory. In *International Conference on Machine Learning*, pages 6565–6590, 2023. 5

[14] Zhiwei Deng and Olga Russakovsky. Remember the past: Distilling datasets into addressable memories for neural networks. In *Advances in Neural Information Processing Systems*, 2022. 2, 6, 5

[15] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear independent components estimation. In *International Conference on Learning Representations*, 2015. 3

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 1, 7

[17] 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

[18] Yunzhen Feng, Ramakrishna Vedantam, and Julia Kempe. Embarrassingly simple dataset distillation. In *International Conference on Learning Representations*, 2024. 1, 2, 3

[19] Robert M. Gray. Vector quantization. *IEEE ASSP Magazine*, 1(2):4–29, 1984. 5

[20] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander J. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773, 2012. 4

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

[22] 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. 3, 7

[23] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory F. Diamos, Heewoo Jun, Hassan Kianinejad, Mostofa Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *ArXiv preprint arXiv:1712.00409*, 2017. 1

[24] David A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9): 1098–1101, 1952. 2

[25] 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 arXiv:2001.08361*, 2020. 1

[26] Hyunjik Kim, Matthias Bauer, Lucas Theis, Jonathan R. 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. 2, 3, 5, 6, 1

[27] 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. 1, 6, 7, 4, 5

[28] Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014. 3

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1106–1114, 2012. 7

[30] 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, 5

[31] 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

[32] 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

[33] Ping Liu and Jiawei Du. The evolution of dataset distillation: Toward scalable and generalizable solutions. *ArXiv preprint arXiv:2502.05673*, 2025. 1, 5

[34] Songhua Liu, Kai Wang, Xingyi Yang, Jingwen Ye, and Xinchao Wang. Dataset distillation via factorization. In *Advances in Neural Information Processing Systems*, 2022. 2, 4, 6, 7, 5

[35] 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. 1

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

[37] David Minnen, Johannes Ballé, and George Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Advances in Neural Information Processing Systems*, pages 10794–10803, 2018. 3

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

[39] Timothy Nguyen, Roman Novak, Lechao Xiao, and Jaehoon Lee. Dataset distillation with infinitely wide convolutional networks. In *Advances in Neural Information Processing Systems*, pages 5186–5198, 2021. 2, 3

[40] Aäron V. Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756, 2016. 1

[41] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. 5

[42] Claude E. Shannon. Coding theorems for a discrete source with a fidelity criterion. *IRE National Convention Record*, 4 (1):142–163, 1959. 3

[43] Donghyeok Shin, Seungjae Shin, and Il-Chul Moon. Frequency domain-based dataset distillation. In *Advances in Neural Information Processing Systems*, 2023. 1, 2, 6, 7, 4, 5

[44] Donghyeok Shin, HeeSun Bae, Gyuwon Sim, Wanmo Kang, and Il-Chul Moon. Distilling dataset into neural field. In *International Conference on Learning Representations*, 2025. 2, 6, 7, 4, 5

[45] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 7

[46] Duo Su, Junjie Hou, Weizhi Gao, Yingjie 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

[47] Benigno Uria, Iain Murray, and Hugo Larochelle. RNADE: The real-valued neural autoregressive density-estimator. In *Advances in Neural Information Processing Systems*, pages 2175–2183, 2013. 3

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017. 3

[49] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *ArXiv preprint arXiv:1811.10959*, 2018. 1, 2

[50] Xing Wei, Anjia Cao, Funing Yang, and Zhiheng Ma. Sparse parameterization for epitomic dataset distillation. In *Advances in Neural Information Processing Systems*, 2023. 1, 6, 7, 4, 5

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

[52] Ian H. Witten, Radford M. Neal, and John G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987. 2

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

[54] 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. 1, 6, 7, 4, 5

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

[56] Zeyuan Yin, Eric P. Xing, and Zhiqiang Shen. Squeeze, recover and relabel: Dataset condensation at ImageNet scale from a new perspective. In *Advances in Neural Information Processing Systems*, 2023. 2, 3, 5, 8

[57] 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*, pages 1–17, 2024. 2, 3

[58] 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. 1, 2

[59] Bo Zhao and Hakan Bilen. Synthesizing informative training samples with GAN. In *Advances in Neural Information Processing Systems Workshops*, 2022. 2

[60] 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, 7, 6

[61] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *International Conference on Learning Representations*, 2021. 1, 2, 4, 7, 6

[62] Haizhong Zheng, Jiachen Sun, Shutong Wu, Bhavya Kailkhura, Zhuo M. 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, 5

[63] 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, 6, 7, 4

[64] Yongchao Zhou, Ehsan Nezhadarya, and Jimmy Ba. Dataset distillation using neural feature regression. In *Advances in Neural Information Processing Systems*, 2022. 6, 7, 4, 5

[65] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977. 2

Appendix

This appendix elaborates on 1) the information-theoretic cost of encoding soft labels, 2) full hyperparameter and architecture configurations in our joint rate-utility optimization method, 3) expanded quantitative results, 4) runtime statistics for synthetic dataset generation, and 5) more synthetic data visualizations.

A1. Bitrate for Soft Labels

Let $\mathcal{Y} = \{\tilde{y}^{(i)}\}_{i=1}^{N}$ be a collection of soft classes, where each $\tilde{y}^{(i)}$ is a probability vector

$$\tilde{y}^{(i)} = [\tilde{y}_1^{(i)}, \ldots, \tilde{y}_K^{(i)}], \quad \tilde{y}_k^{(i)} \geq 0, \quad \sum_{k=1}^{K} \tilde{y}_k^{(i)} = 1, \tag{A1}$$

lying in the $(K-1)$-simplex Δ^{K-1}. To encode these continuous vectors, we quantize Δ^{K-1} into B cells (*i.e.*, "bins") of side length ϵ. Denote by $\{\Omega_i\}_{i=1}^{B}$ the partition cells, each with volume $\mathrm{Vol}(\Omega_i) \approx \epsilon^{K-1}$, and let

$$P_i = \int_{\Omega_i} p(\xi) \mathrm{d}\xi \tag{A2}$$

be the probability mass of soft labels falling in bin Ω_i under the true (non-uniform) density $p(\xi)$. The optimal bitrate per label, using entropy coding, is given by the Shannon entropy of the mass distribution:

$$r(\mathcal{Y}) \approx \mathrm{Entropy}(P) = -\sum_{i=1}^{B} P_i \log_2 P_i, \tag{A3}$$

which accounts for non-uniform concentrations of soft labels across the simplex. As entropy is maximized when $P_i = 1/B$ for all i, we have

$$\begin{aligned} r(\mathcal{Y}) \leq \log_2 B &\approx \log_2 \left(\frac{\mathrm{Vol}(\Delta^{K-1})}{\epsilon^{K-1}} \right) \\ &= \log_2 \left(\frac{1}{(K-1)! \epsilon^{K-1}} \right) \\ &= -\log_2 (K-1)! - (K-1) \log_2 \epsilon. \end{aligned} \tag{A4}$$

where $\mathrm{Vol}(\Delta^{K-1}) = 1/(K-1)!$. This recovers the bound in the main text for worst-case (uniform) quantization.

Practical illustration. For single-precision floats (*i.e.*, `fp32`) the 24-bit mantissa roughly yields a machine precision of $\epsilon \approx 2^{-24}$. For a $K = 1,000$-class problem such as ImageNet-1K classification, the continuous label space $\mathcal{Y}$ incurs an encoding cost of $r(\mathcal{Y}) \approx 15,456$ bits/vector, which dwarfs the $\log_2 K = \log_2 1,000 \approx 10$ bits needed for representing a hard (one-hot) label, even surpassing the size of synthetic samples themselves.

A2. Hyperparameter Settings

Implementation Details. For synthetic dataset parameterization, we employ the C3 defaults [26]: $L = 6$ latent scales with an upsampling kernel of size 8. We customize both the entropy and decoder networks to accommodate different datasets and varying synthetic samples per class (spc). The entropy network is implemented by a multilayer perceptron, whose detailed layer-by-layer specification is provided in Table A2. All hidden layers employ ReLU activations, and the final layer outputs two values—$\mu, \log \sigma$—which parameterize the conditional Laplace distribution of each latent code. To enforce causality, latent codes are processed in raster-scan order [40]. For each code m at scale l, we extract its causal context from a fixed-size neighborhood of C previously encoded codes, experimenting with $C \in \{8, 16, 24, 32, 64\}$ (see Table A2 for more details). A single entropy network then processes all scales simultaneously by concatenating context tensors $\{c_1, \ldots, c_L\}$, where $c_l \in \mathbb{R}^{(\lfloor \frac{H}{2^{l-1}} \rfloor \times \lfloor \frac{W}{2^{l-1}} \rfloor) \times C}$, to form $c \in \mathbb{R}^{(\sum_{l=1}^{L} \lfloor \frac{H}{2^{l-1}} \rfloor \times \lfloor \frac{W}{2^{l-1}} \rfloor) \times C}$.

Fig. 2 in the main text depicts our decoder architecture, which comprises five convolution layers with hyperparameters $\{D_1, D_2, D_3\}$. Two residual connections link the last two convolutions. Following the guidelines of [26], we implement eight distinct decoder configurations (see Table A1), where setting $D_2 = 0$ indicates the identity mapping (*i.e.*, no learnable parameters) at that layer. ReLU is adopted as the nonlinear activation function. We further denote "slice size" as the number

Table A1. Configuration of the decoder network across model variants, denoted by "v⟨depth⟩-⟨channels⟩," where the suffix denotes the network depth and first-layer output channels. D_1 to D_3 list the output channel counts for layers one through three, and the last column gives the total number of trainable parameters.

Version	D_1	D_2	D_3	#Parameters
v4-40	40	0	3	571
v4-160	160	0	3	1,771
v4-240	240	0	3	2,571
v4-480	480	0	3	4,971
v4-960	960	0	3	9,771
v4-1200	1,200	0	3	12,171
v5-240	240	40	3	11,611
v5-320	320	40	3	15,371

of synthetic samples handled by each entropy or decoder network. By default, we allocate one network per class. However, for CIFAR-10 with spc $= 718$, we set the slice size to 359, resulting in two slices per class.

For downstream classifiers, we follow FreD [43] and employ convolutional neural networks for both dataset distillation and evaluation. The architecture comprises identical blocks, each containing a 3×3 convolution with 128 filters, instance normalization, ReLU, and 2×2 average pooling (stride 2). A final linear layer produces the output logits. We configure three blocks for 32×32 datasets (*i.e.*, CIFAR-10 and CIFAR-100), and five blocks for 128×128 ImageNet subsets.

We use TM [9] as our default distillation loss, while supporting GM [61] and DM [60] identically. All experiments use differentiable siamese augmentation [59] for data augmentation; zero-phase component analysis as a form of signal whitening is only applied to CIFAR-10 at spc $= 64$ or 240. Hyperparameters for each loss generally follow those of FreD [43] and DDiF [44] (see Table A2 for details), with synthetic minibatch sizes adjusted for higher spc in TM.

Training Protocols. In the *initialization* phase, we adopt the default warm-up schedule from C3 [26], setting the learning rate to 0.01. The rate-distortion trade-off coefficient β is chosen per dataset and distillation loss. For instance, $\beta = 10$ when applying the TM loss on ImageNet subsets, while $\beta = 10^6$ for both the GM and DM losses.

In the *joint rate-utility optimization* phase, training is carried out using the Adam optimizer at a fixed learning rate of 10^{-3}. We run $15,000$ iterations for TM, 800 for GM, and $20,000$ for DM, respectively. Bit-budget enforcement is governed by a two-stage schedule for the rate-utility coefficient λ: during the first half of training, a generally higher λ prioritizes distillation performance; in the second half, we are inclined to decrease λ to strictly impose the bitrate constraint. In the *post-quantization* phase, we enforce the mean squared error between the pre- and post-quantized synthetic samples to lie within $\{5 \times 10^{-5}, 5 \times 10^{-6}, 5 \times 10^{-7}, 5 \times 10^{-8}\}$, then select the threshold that maximizes rate-utility performance for the target bpc budget. In the *evaluation* phase, we train five independent classifiers for $1,000$ epochs and report the average classification accuracy. For cross-architecture comparisons, we utilize FreD's implementations [43], applying a fixed Adam learning rate of 0.02 for AlexNet, VGG-11, ResNet-18, and 0.015 (with a dropout ratio of 0.01) for a variant of ViT-B. All experiments are performed on NVIDIA A100 ($4\times$) and A800 ($4\times$) GPUs.

A3. Expanded Quantitative Results

We provide detailed performance comparisons across multiple datasets and distillation losses to underscore the generality and efficiency of our joint rate-utility optimization method.

Table A3 supplements the raw data for the rate-utility curve in Fig. 1 of the main text.

Table A4 summarizes results on CIFAR-10 and CIFAR-100 over a range of bpc budgets. Under each constraint, our method not only achieves state-of-the-art classification accuracy but also requires substantially fewer bits—for example, on CIFAR-100 with a 120 kB budget, we attain superior accuracy using only 53 kB.

Table A5 extends this analysis by applying the TM loss [9] across four distinct backbone architectures (*i.e.*, AlexNet, VGG-11, ResNet-18, and a variant of ViT-B), confirming cross-architecture generalization.

Table A6 reports averaged accuracies ($\pm$ standard deviation) on the ImageNet subsets, when using the GM [61] and DM [60] distillation losses under bpc $= 192$ kB with spc $= 96$, revealing consistent gains over vanilla baselines.

Table A2. Overview of hyperparameter settings for different DD loss functions.

(a) Hyperparameter settings for the TM loss.

Dataset	spc	TM				Entropy Network		Decoder Network		Optimization	
		t_1	t_2	t	Synthetic minibatch size	Width	Depth	Version	Slice Size	β	λ
CIFAR-10	64	60	392	1,960	256	16	4	v4-480	64	10^6	$\{2 \times 10^1\}$
	240	60	392	7,840	410	16	4	v4-960	240	10^6	$\{10^2, 2 \times 10^2\}$
	718	60	392	7,840	720	16	4	v4-1200	360	10^6	$\{2 \times 10^2\}$
CIFAR-100	48	60	392	7,840	256	16	4	v4-240	48	10^8	$\{3 \times 10^2\}$
	120	40	392	9,408	960	16	4	v4-960	120	10^6	$\{1.5 \times 10^3\}$
ImageNet Subset	1	20	102	510	102	16	2	v4-40	1	10	$\{10^2, 10^3\}$
	8	20	102	510	102	16	2	v4-160	8	10	$\{10^3, 8.5 \times 10\}$
	15	20	102	510	102	16	2	v4-240	15	10	$\{10^3, 8.5 \times 10\}$
	51	40	102	1,020	80	32	4	v5-240	51	10	$\{10^3, 8.5 \times 10\}$
	102	40	102	1,020	80	32	4	v5-240	102	10	$\{10^3, 6.6 \times 10\}$

(b) Hyperparameter settings for the GM and DM losses.

Dataset	ℓ	spc	Synthetic batch size	Entropy Network		Decoder Network		Optimization	
				Width	Depth	Version	Slice Size	β	λ
ImageNet Subset	GM	96	960	32	4	v5-320	96	10^6	$\{2.8 \times 10^{-2}, 1.2 \times 10^{-2}\}$
	DM	96	960	32	4	v5-320	96	10^6	$\{2 \times 10^1, 6.7 \times 10^{-1}\}$

A4. Wall-Clock Time

We evaluate decoding speed on a single NVIDIA A100 80 GB GPU, averaged across $10,000$ runs. From Table A7, we find that total latency grows roughly linearly with spc, but the slope varies by dataset due to the differences in the entropy and decoder networks. For instance, increasing spc from 64 to 718 on CIFAR-10 raises average latency from 55.30 ms to 318.57 ms ($5.8\times$), while on ImageNet an increase from 1 to 102 spc yields a $5.7\times$ rise (51.95 ms to 295.87 ms). Notably, per-sample latency improvements taper off at high spc: beyond spc $= 51$, ImageNet's per-sample time only drops from 0.32 ms to 0.29 ms, and CIFAR-10 stabilizes at 0.04 ms past spc $= 240$, indicating hardware or algorithmic limits on batching efficiency.

A5. Expanded Qualitative Results

Figs. A1 to A6 showcase final synthetic samples optimized with the TM loss, whereas Figs. A7 and A8 juxtapose multiscale latent-code visualizations and their decoded reconstructions using the GM and DM objectives. Each panel's first row presents initialization by "overfitted" image compression, the second row shows distilled images, and the last six rows display the corresponding multiscale latent codes. Finally, Figs. A9 to A14 extend these visualizations across all six ImageNet subsets, demonstrating consistently structured latent representations and sample synthesis.

Table A3. Detailed experimental results on the Nette subset of ImageNet (128×128). Classification accuracies (%) are reported as mean $\pm$ standard deviation.

Method	spc	#Parameters Per Class ($\times 10^4$)	bpc (kB)	Accuracy (%)
TM [9]	1	4.92	192.0	51.4$\pm$2.3
	10	49.15	1,920.0	63.0$\pm$1.3
	50	245.76	9,600.0	72.8$\pm$0.8
	51	250.68	9,792.0	73.0$\pm$0.7
GLaD [10]	–	4.92	192.0	38.7$\pm$1.6
H-GLaD [63]	–	4.92	192.0	45.4$\pm$1.1
FRePo [64]	–	4.92	192.0	48.1$\pm$0.7
	–	49.15	1,920.0	66.5$\pm$0.8
HaBa [34]	–	4.92	192.0	51.9$\pm$0.7
	–	49.15	1,920.0	64.7$\pm$1.6
IDC [27]	–	4.92	192.0	61.4$\pm$1.0
	–	49.15	1,920.0	70.8$\pm$0.5
FReD [43]	8	4.92	192.0	66.8$\pm$0.4
	16	9.83	384.0	69.0$\pm$0.9
	40	49.15	1,920.0	72.0$\pm$0.8
SPEED [50]	15	4.92	192.0	66.9$\pm$0.7
	111	49.15	1,920.0	72.9$\pm$1.5
NSD [54]	–	4.92	192.0	68.6$\pm$0.8
DDiF [44]	1	0.10	3.8	49.1$\pm$2.0
	8	0.77	30.1	67.1$\pm$0.4
	15	1.44	56.4	68.3$\pm$1.1
	51	4.92	192.0	72.0$\pm$0.9
	510	49.15	1,920.0	74.6$\pm$0.7
	697	245.55	9,591.2	75.2$\pm$1.3
TM-RUO (Ours)	1	2.31	3.2	49.2$\pm$1.2
	8	17.71	8.4	66.4$\pm$1.6
	15	33.08	18.2	69.2$\pm$1.5
	51	112.98	101.7	74.5$\pm$0.8
	102	224.36	179.7	76.5$\pm$0.7

Table A4. Classification accuracies (mean ± standard deviation) on CIFAR-10 and CIFAR-100 under five different bpc budgets. The first row ("Original") shows accuracy when training on the original dataset. The budget rows give the compressed-size budgets (in kB) with the corresponding number of synthetic spc (in parentheses).

Method	CIFAR-10			CIFAR-100	
	12 kB	120 kB	600 kB	12 kB	120 kB
Original		84.8±0.1			56.2±0.3
TM [9]	46.3±0.8	65.3±0.7	71.6±0.2	24.3±0.3	40.1±0.4
FRePo [64]	46.8±0.7	65.5±0.4	71.7±0.2	28.7±0.1	42.5±0.2
IDC [27]	50.0±0.4	67.5±0.5	74.5±0.2	–	–
HaBa [34]	48.3±0.8	69.9±0.4	74.0±0.2	–	–
FreD [43]	60.6±0.8	70.3±0.3	75.8±0.1	34.6±0.4	42.7±0.2
RTP [14]	66.4±0.4	71.2±0.4	73.6±0.5	34.0±0.4	42.9±0.7
NSD [54]	68.5±0.8	73.4±0.2	75.2±0.6	36.5±0.3	46.1±0.2
SPEED [50]	63.2±0.1	73.5±0.2	77.7±0.4	40.4±0.4	45.9±0.3
HMN [62]	65.7±0.3	73.7±0.1	76.9±0.2	36.3±0.2	45.4±0.2
DDiF [44]	66.5±0.4	74.0±0.4	77.5±0.3	42.1±0.2	46.0±0.2
Method	CIFAR-10			CIFAR-100	
	13 kB (64)	94 kB (240)	246 kB (718)	8 kB (48)	53 kB (120)
TM-RUO (Ours)	**70.3**±0.7	**77.5**±0.3	**79.7**±0.3	**44.4**±0.3	**49.2**±0.4

Table A5. Classification accuracies (mean ± standard deviation) on six 128×128 ImageNet subsets across different network architectures, including AlexNet, VGG-11, ResNet-18, and a variant of ViT-B, evaluated under a bpc budget of $\leq$ 192 kB.

Classifier	Method	Nette	Woof	Fruit	Yellow	Meow	Squawk
AlexNet	TM [9]	13.2±0.6	10.0±0.0	10.0±0.0	11.0±0.2	9.8±0.0	–
	IDC [27]	17.4±0.9	16.5±0.7	17.9±0.7	20.6±0.9	16.8±0.5	20.7±1.0
	FreD [43]	35.7±0.4	23.9±0.7	15.8±0.7	19.8±1.2	14.4±0.5	36.3±0.3
	DDiF [44]	60.7±2.3	**36.4**±2.3	**41.8**±0.6	**56.2**±0.8	**40.3**±1.9	**60.5**±0.4
	TM-RUO (Ours)	**64.1**±1.3	32.6±1.8	40.7±0.5	49.9±2.4	36.9±1.6	55.9±2.5
VGG-11	TM [9]	17.4±2.1	12.6±1.8	11.8±1.0	16.9±1.1	13.8±1.3	–
	IDC [27]	19.6±1.5	16.0±2.1	13.8±1.3	16.8±3.5	13.1±2.0	19.1±1.2
	FreD [43]	21.8±2.9	17.1±1.7	12.6±2.6	18.2±1.1	13.2±1.9	18.6±2.3
	DDiF [44]	53.6±1.5	29.9±1.9	33.8±1.9	44.2±1.7	32.0±1.8	37.9±1.5
	TM-RUO (Ours)	**70.3**±2.5	**42.1**±2.9	**44.6**±1.4	**66.9**±0.4	**49.6**±1.4	**66.8**±2.8
ResNet-18	TM [9]	34.9±2.3	20.7±1.0	23.1±1.5	43.4±1.1	22.8±2.2	–
	IDC [27]	43.6±1.3	23.2±0.8	32.9±2.8	44.2±3.5	28.2±0.5	47.8±1.9
	FreD [43]	48.8±1.8	28.4±0.6	34.0±1.9	49.3±1.1	29.0±1.8	50.2±0.8
	DDiF [44]	63.8±1.8	37.5±1.9	42.0±1.9	55.9±1.0	35.8±1.8	62.6±1.5
	TM-RUO (Ours)	**70.6**±1.2	**39.9**±1.8	**44.6**±0.6	**67.1**±1.5	**47.8**±0.9	**66.1**±1.6
Variant of ViT-B	TM [9]	22.6±1.1	15.9±0.4	23.3±0.4	18.1±1.3	18.6±0.9	–
	IDC [27]	31.0±0.6	22.4±0.8	31.1±0.8	30.3±0.6	21.4±0.7	32.2±1.2
	FreD [43]	38.4±0.7	25.4±1.7	31.9±1.4	37.6±2.0	19.7±0.8	44.4±1.0
	DDiF [44]	**59.0**±0.4	**32.8**±0.8	39.4±0.8	47.9±0.6	27.0±0.6	54.8±1.1
	TM-RUO (Ours)	57.5±1.2	29.5±0.3	**41.4**±1.1	**48.2**±0.8	**28.0**±0.9	**55.8**±1.4

Table A6. Classification accuracies (mean ± standard deviation) on six 128×128 ImageNet subsets for different utility losses, including gradient matching (GM) [61] and distribution matching (DM) [60], evaluated under a bpc budget of ≤ 192 kB.

Method	Nette	Woof	Fruit	Yellow	Meow	Squawk	Avg
GM							
GM (Vanilla) [61])	34.2±1.7	22.5±1.0	21.0±0.9	37.1*±1.1	22.0±0.6	32.0±1.5	28.1
GLaD [10]	35.4±1.2	22.3±1.1	20.7±1.1	–	22.6±0.8	33.8±0.9	26.9
H-GLaD [63]	36.9±0.8	24.0±0.8	22.4±1.1	–	24.1±0.9	35.3±1.0	28.5
IDC [27]	45.4±0.7	25.5±0.7	26.8±0.4	–	25.3±0.6	34.6±0.5	31.5
FreD [43]	49.1±0.8	26.1±1.1	30.0±0.7	–	28.7±1.0	39.7±0.7	34.7
DDiF [44]	61.2±1.0	**35.2**±1.7	**37.8**±1.1	–	39.1±1.3	54.3±1.0	45.5
GM-RUO (Ours)	**67.2**±1.1	33.1±1.1	37.0±1.0	**58.9**±1.4	**42.0**±1.7	**58.6**±1.6	**49.5**
DM							
DM (Vanilla) [60]	30.4±2.7	20.7±1.0	20.4±1.9	36.0*±0.8	20.1±1.2	26.6±2.6	25.7
GLaD [10]	32.2±1.7	21.2±1.5	21.8±1.8	–	22.3±1.6	27.6±1.9	25.0
H-GLaD [63]	34.8±1.0	23.9±1.9	24.4±2.1	–	24.2±1.1	29.5±1.5	27.4
IDC [27]	48.3±1.3	27.0±1.0	29.9±0.7	–	30.5±1.0	38.8±1.4	34.9
FreD [43]	56.2±1.0	31.0±1.2	33.4±0.5	–	33.3±0.6	42.7±0.8	39.3
DDiF [44]	69.2±1.0	42.0±0.4	45.3±1.8	–	45.8±1.1	64.6±1.1	53.4
DM-RUO (Ours)	**71.9**±0.8	**46.4**±1.7	**49.0**±0.5	**69.2**±1.0	**48.8**±1.4	**69.0**±1.1	**59.1**

Table A7. Wall-clock time of synthetic dataset generation.

Dataset	spc	Average Time (ms)	Time Per Sample (ms)
CIFAR-10	64	55.30	0.09
	240	105.04	0.04
	718	318.57	0.04
CIFAR-100	48	677.13	0.14
	120	706.31	0.06
ImageNet Subset	1	51.95	5.19
	8	52.32	0.65
	15	60.58	0.40
	51	161.51	0.32
	102	295.87	0.29

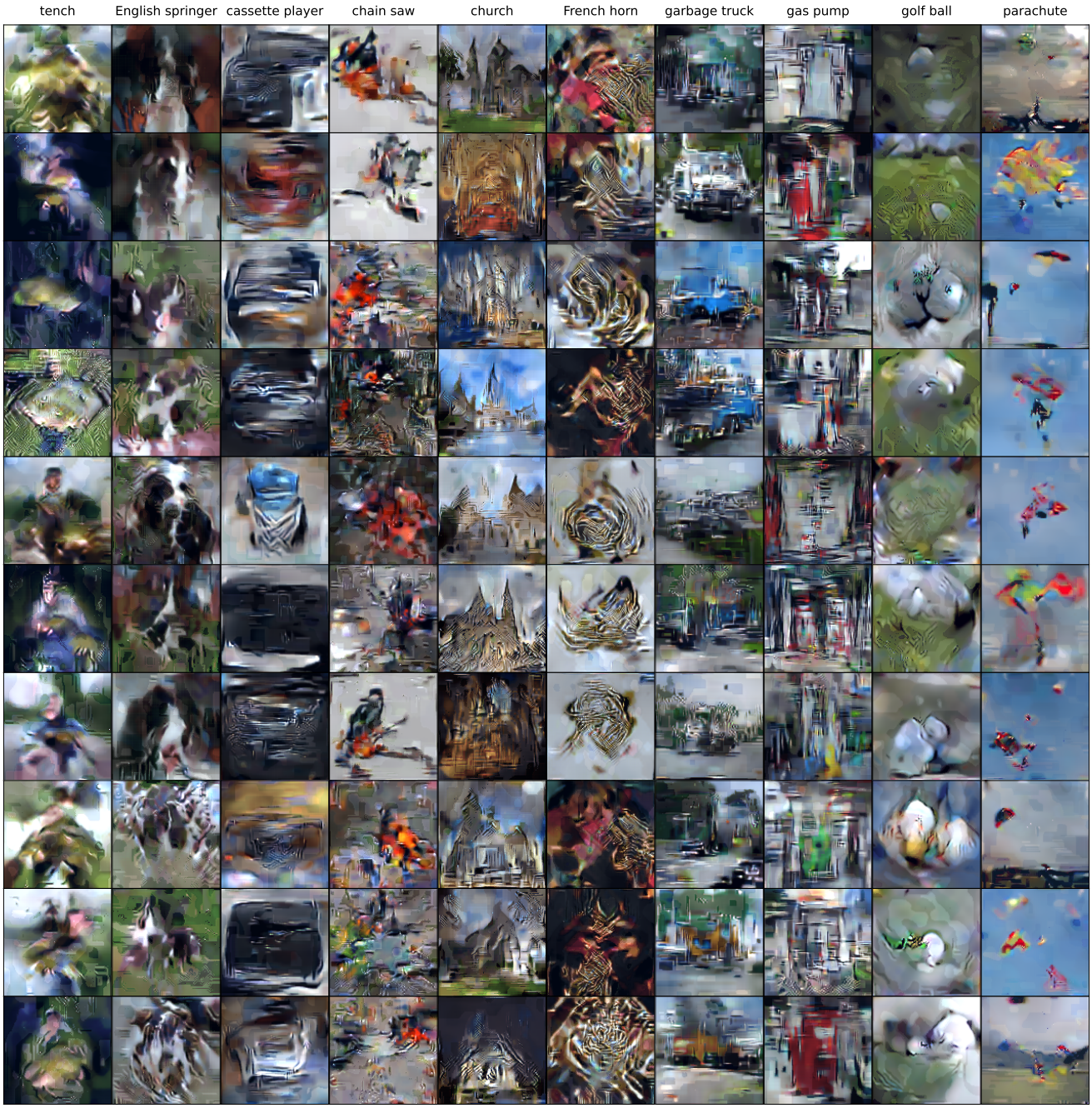


Figure A1. Visualization of final synthetic samples on the Nette subset of ImageNet.

Figure A2. Visualization of final synthetic samples on the Woof subset of ImageNet.

Figure A3. Visualization of final synthetic samples on the Fruit subset of ImageNet.

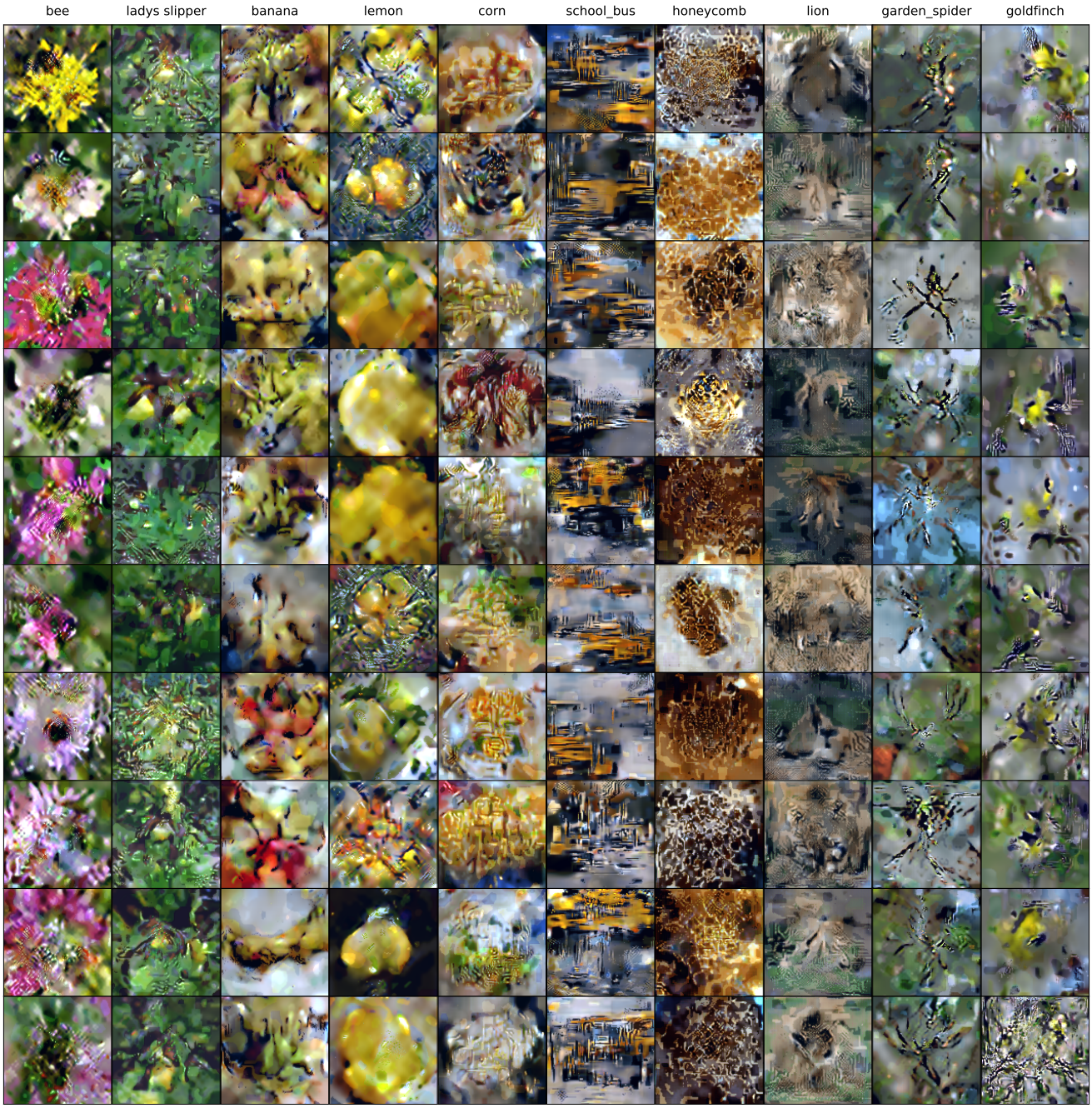


Figure A4. Visualization of final synthetic samples on the Yellow subset of ImageNet.

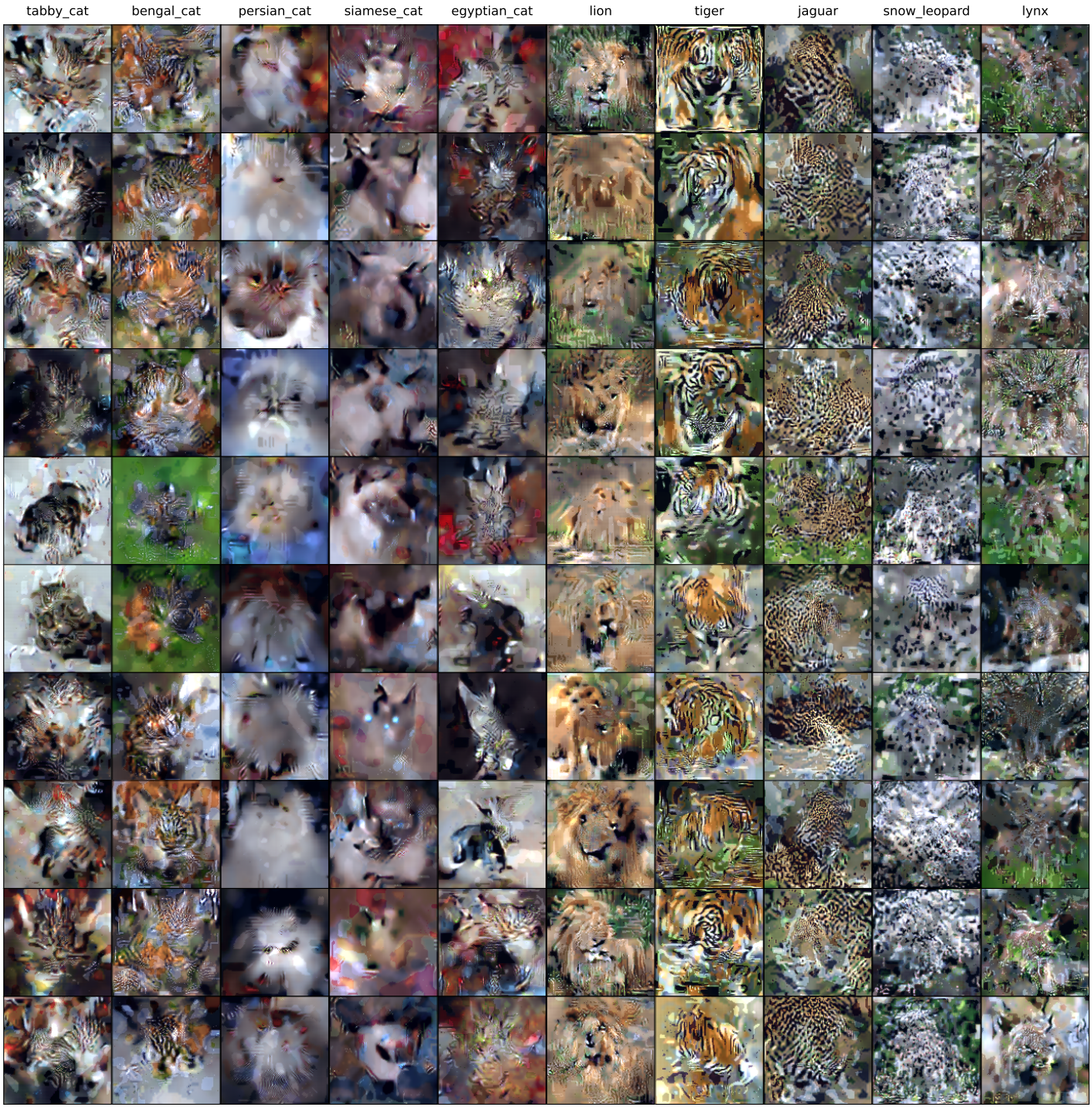


Figure A5. Visualization of final synthetic samples on the Meow subset of ImageNet.

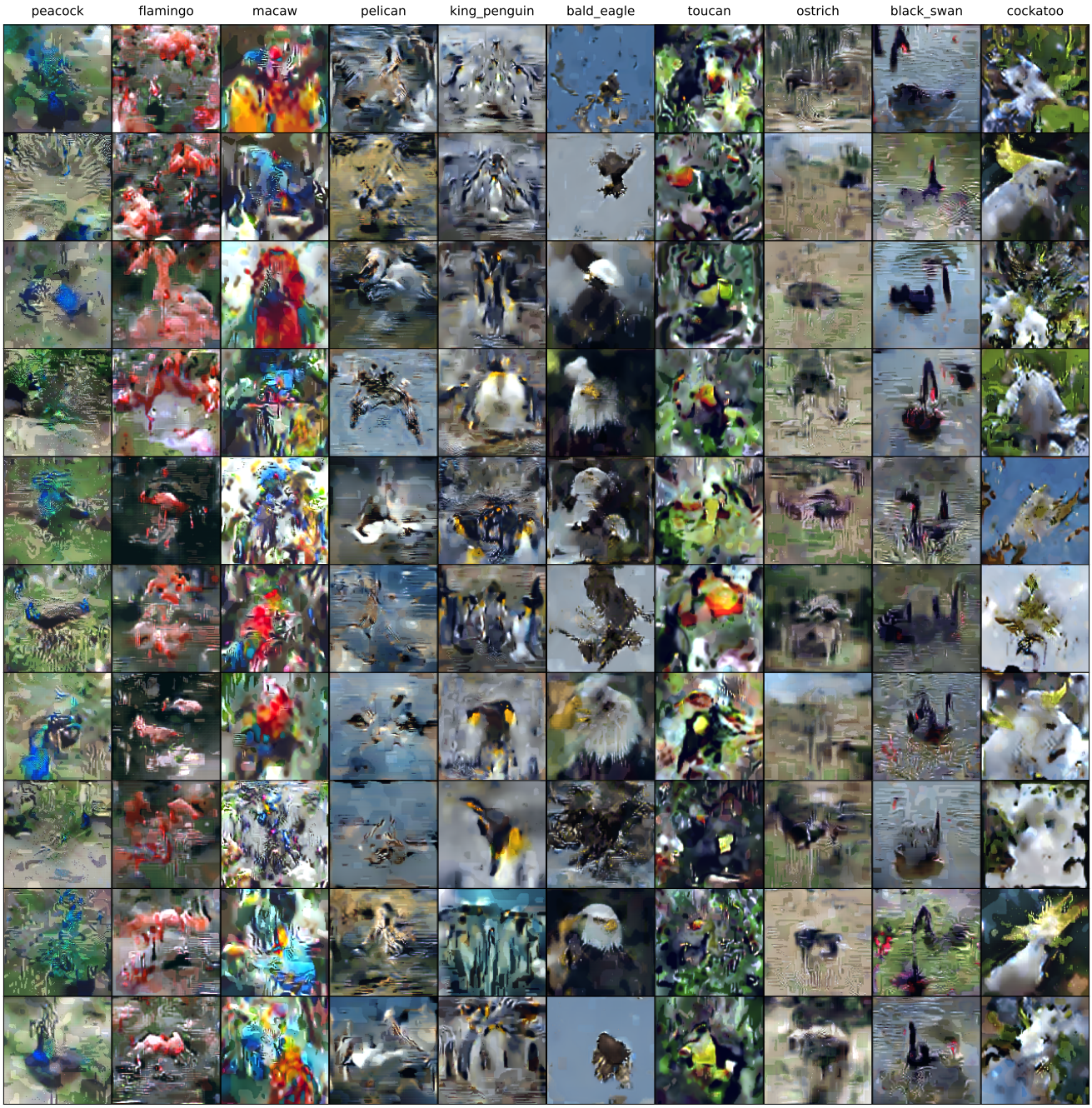


Figure A6. Visualization of final synthetic samples on the Squawk subset of ImageNet.

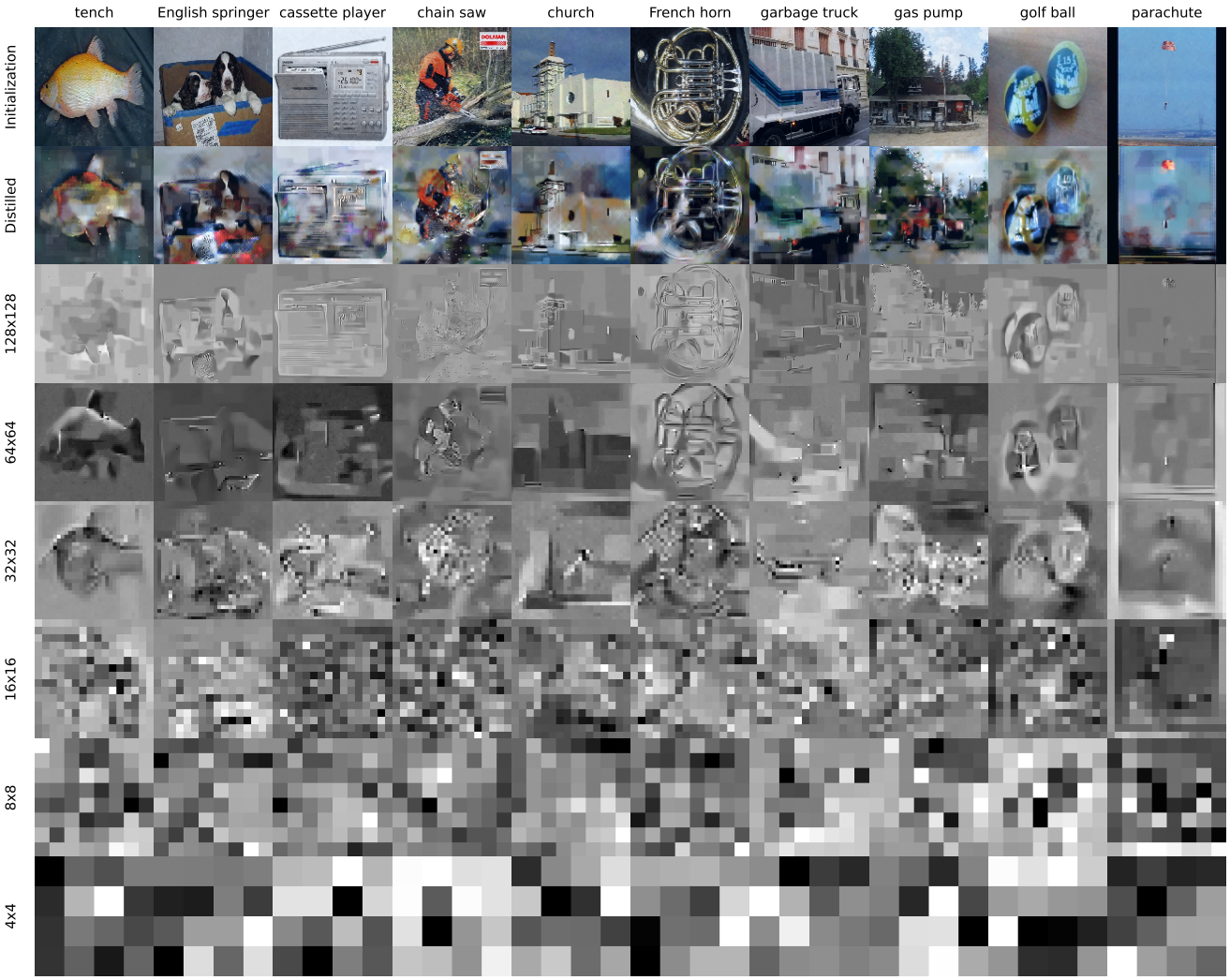


Figure A7. Visualization of synthetic samples on the Nette subset of ImageNet using the GM loss.

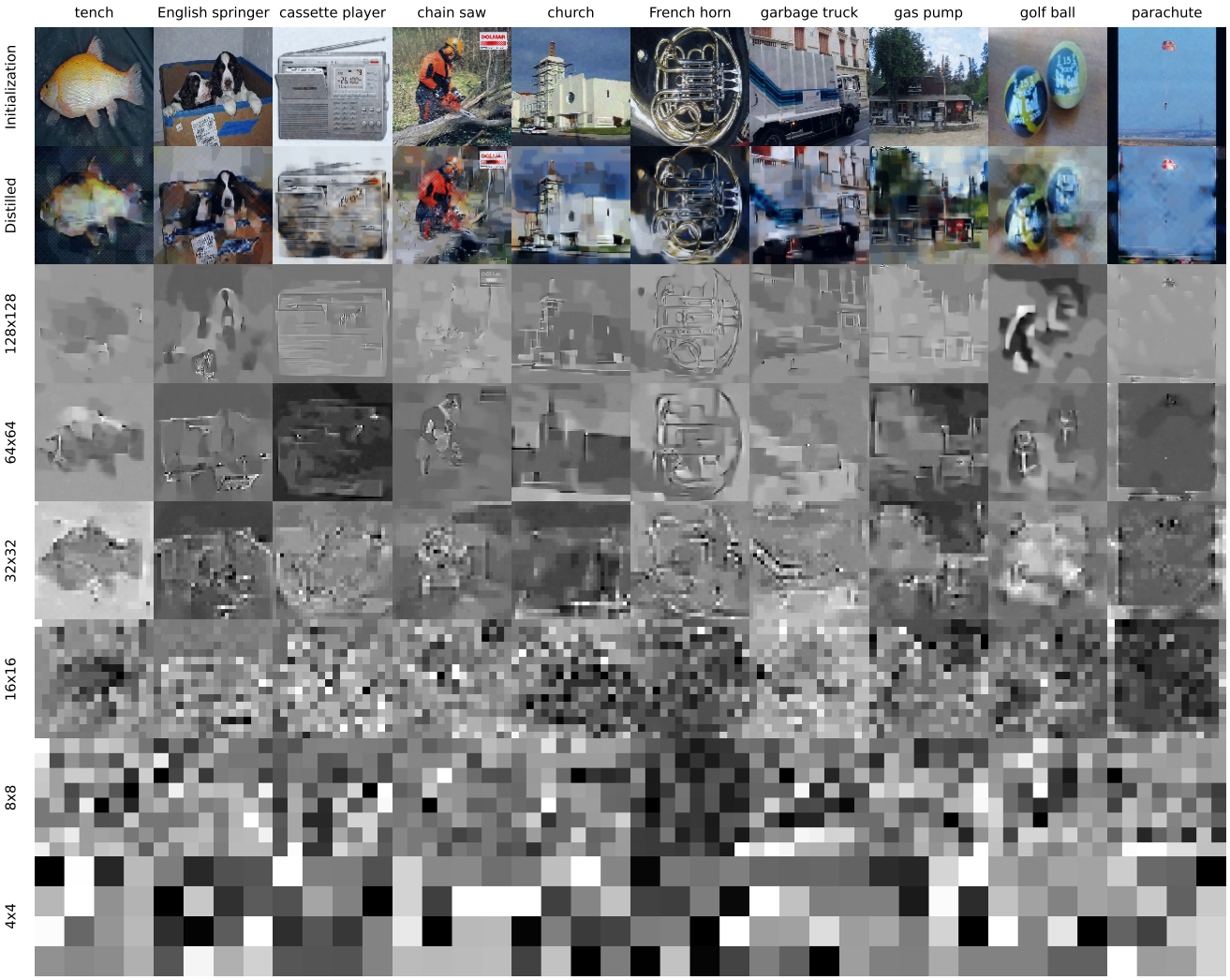


Figure A8. Visualization of synthetic samples on the Nette subset of ImageNet using the DM loss.

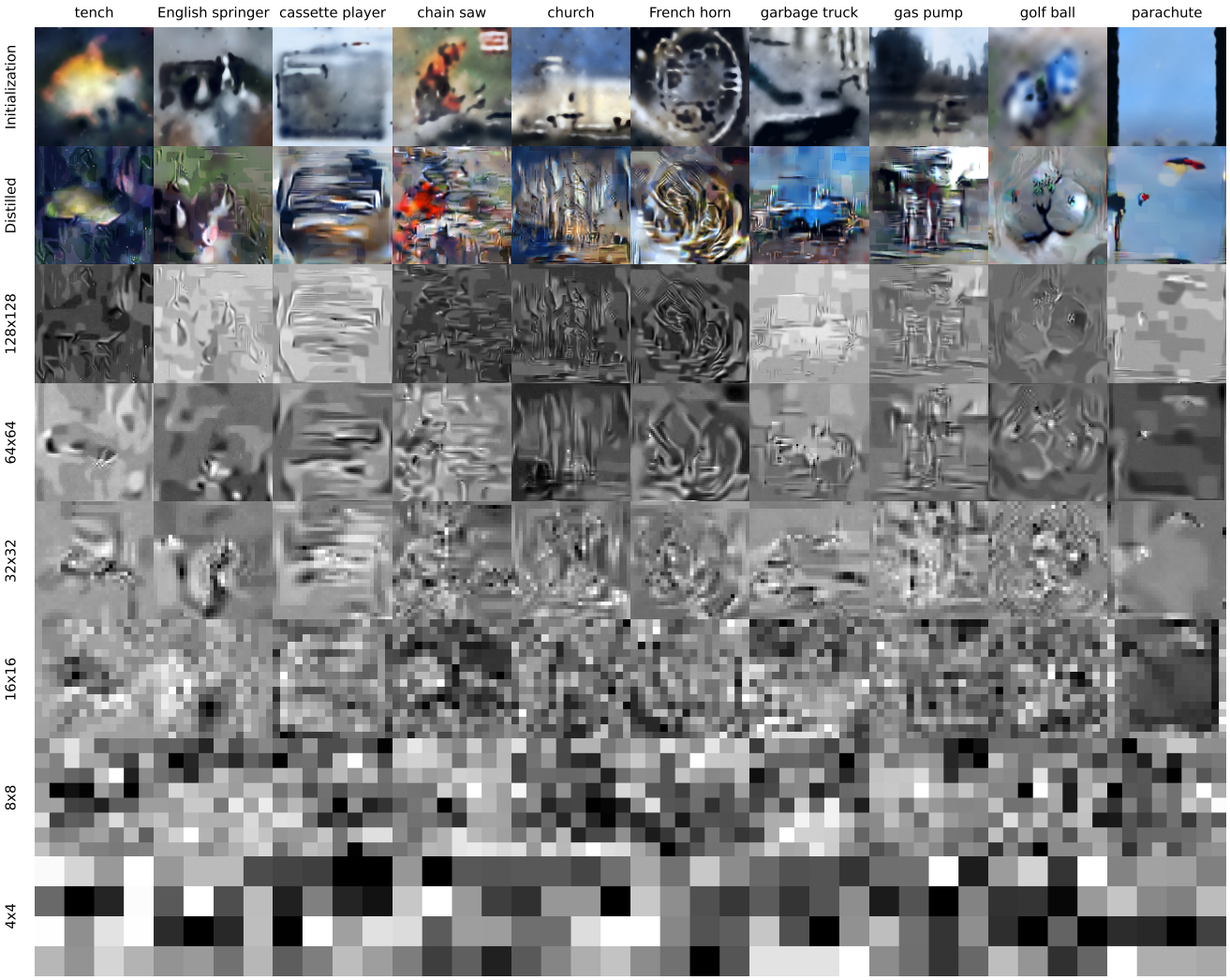


Figure A9. Visualization of synthetic samples on the Nette subset of ImageNet using the TM loss.

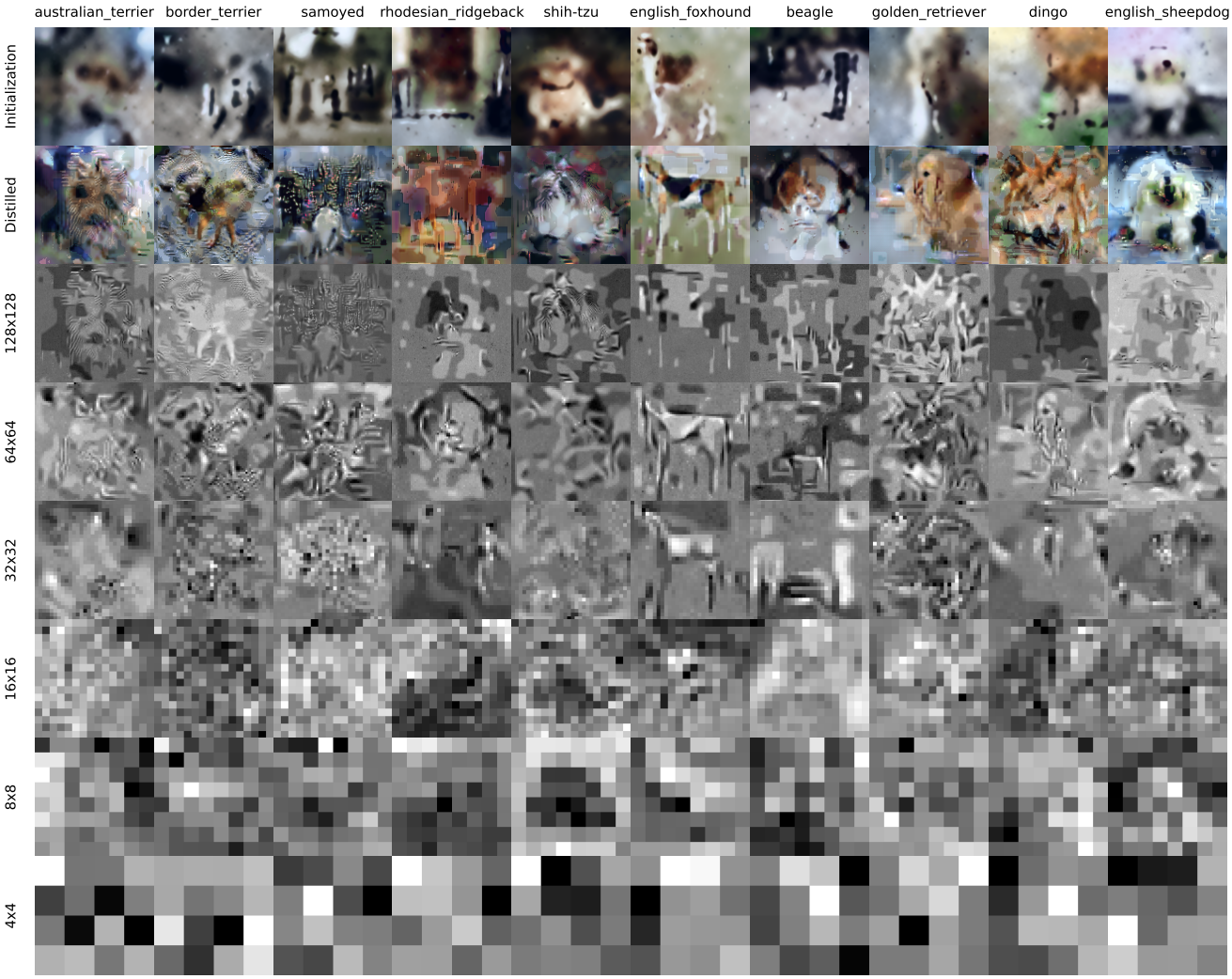


Figure A10. Visualization of synthetic samples on the Woof subset of ImageNet using the TM loss.

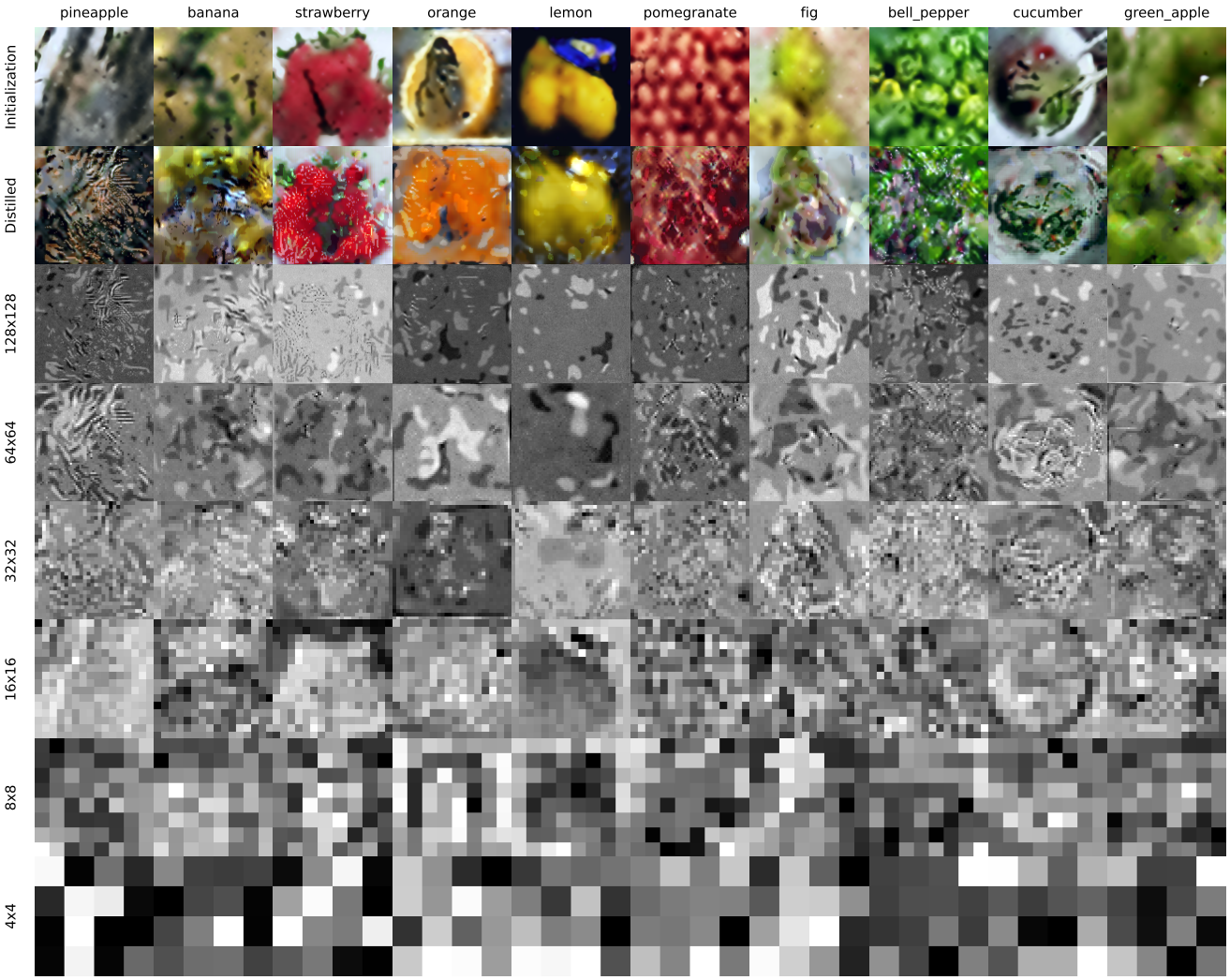


Figure A11. Visualization of synthetic samples on the Fruit subset of ImageNet using the TM loss.

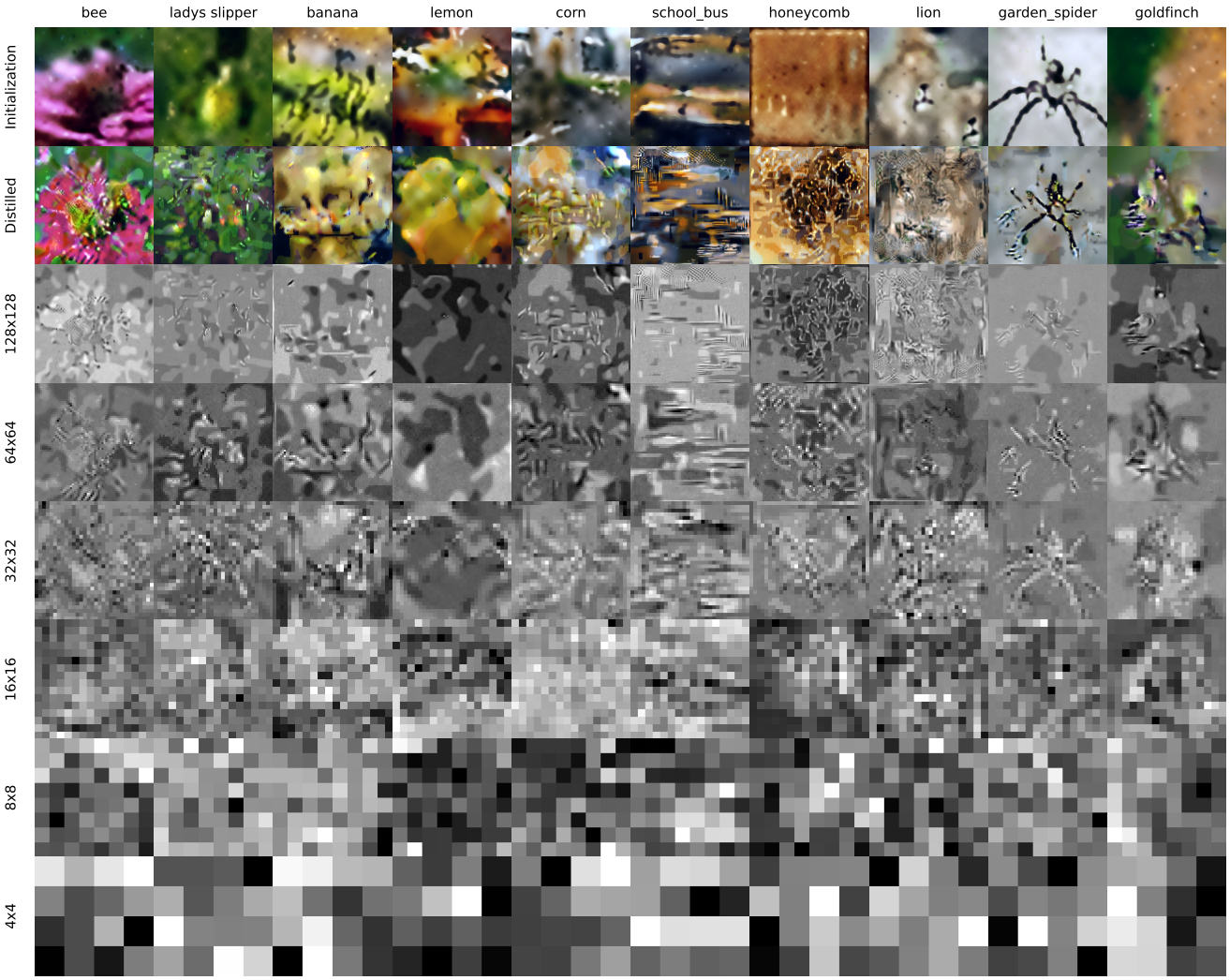


Figure A12. Visualization of synthetic samples on the Yellow subset of ImageNet using the TM loss.

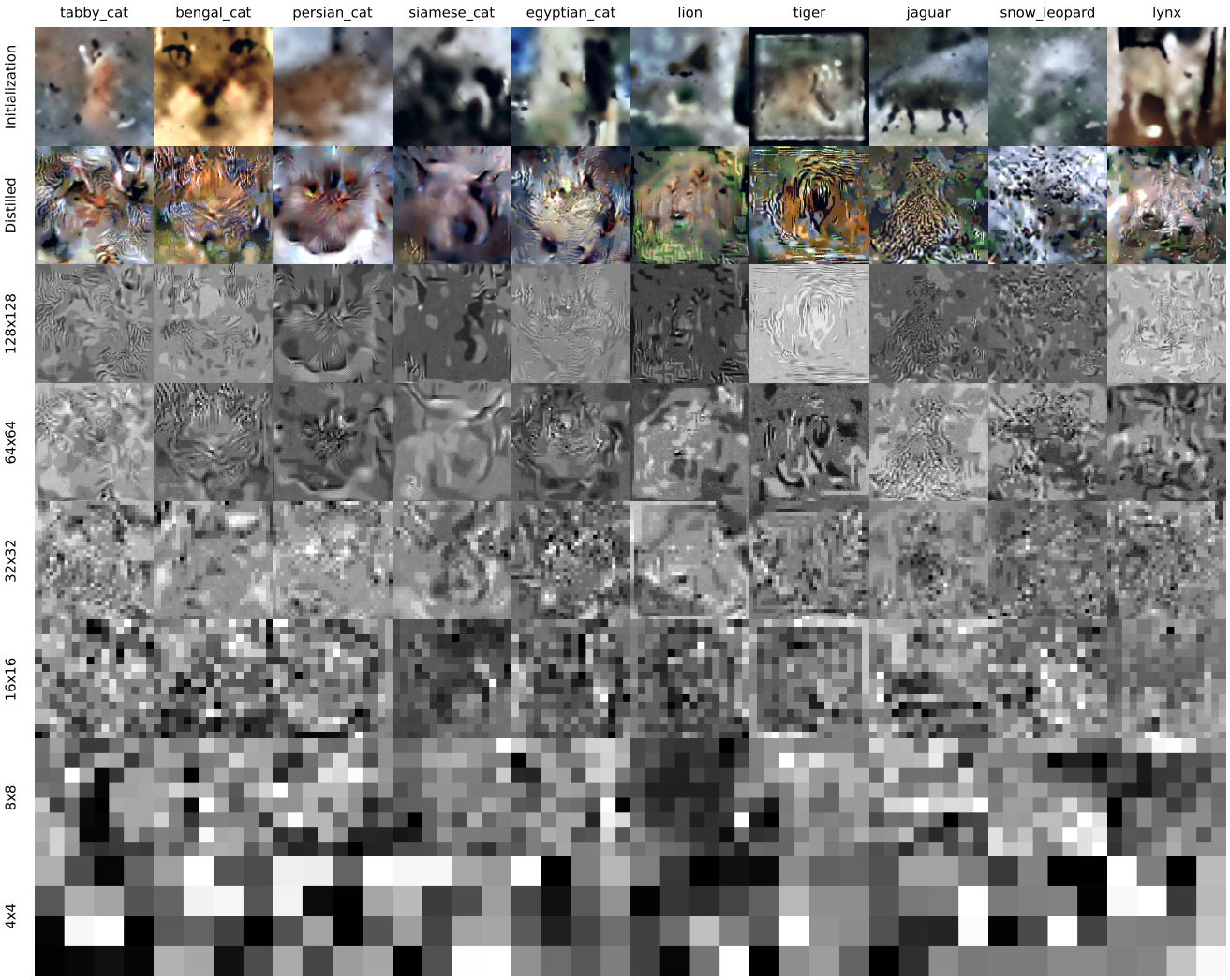


Figure A13. Visualization of synthetic samples on the Meow subset of ImageNet using the TM loss.

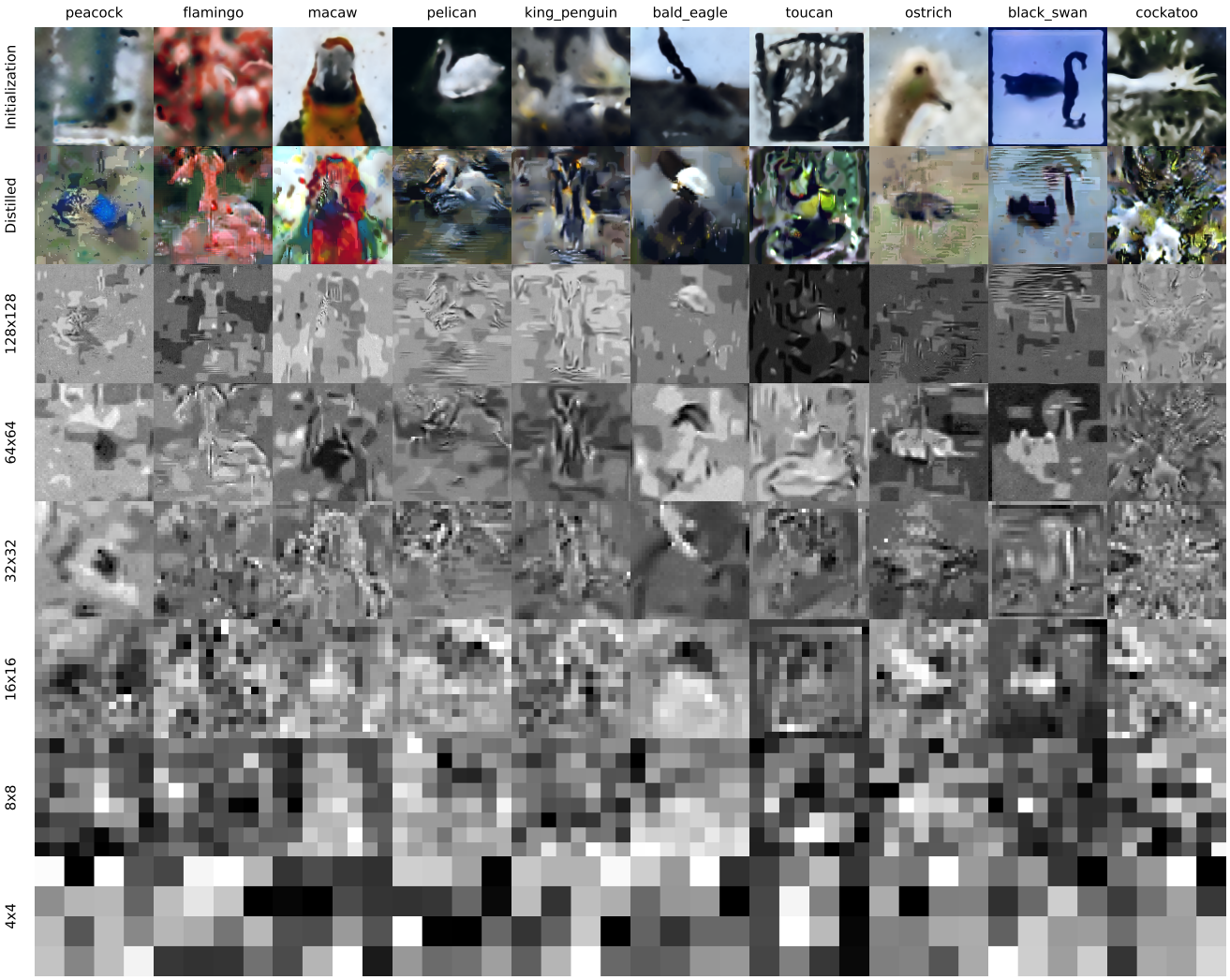


Figure A14. Visualization of synthetic samples on the Squawk subset of ImageNet using the TM loss.