–Supplementary Materials–
DataDAM: Efficient Dataset Distillation with Attention Matching

Ahmad Sajedi[1]*, Samir Khaki[1]*, Ehsan Amjadian[2,3], Lucy Z. Liu[2], Yuri A. Lawryshyn[1], and Konstantinos N. Plataniotis[1]
[1]University of Toronto [2]Royal Bank of Canada (RBC) [3]University of Waterloo

Code: https://github.com/DataDistillation/DataDAM

Contents

1. Implementation Details

1.1. Datasets

We carried out experiments on the following datasets: CIFAR10/100 [14], TinyImageNet [16], ImageNet-1K [6], and subsets of ImageNet-1K including ImageNette [12], ImageWoof [12], and ImageSquawk [2]. CIFAR10/100 is a standard computer vision dataset consisting of natural images with colored 32x32 pixels. It has 10 coarse-grained labels (CIFAR10) and 100 fine-grained labels (CIFAR100), each with 50,000 training samples and 10,000 tests. The classes of the CIFAR10 are "Airplane", "Car", "Bird", "Cat", "Deer", "Dog", "Frog", "Horse", "Ship", and "Truck," which are mutually exclusive. TinyImageNet is a subset of the ImageNet-1K dataset with 200 classes. The dataset contains 100,000 high-resolution training images and 10,000 test examples that are downsized to 64x64. ImageNet-1K is a standard large-scale dataset with 1,000 classes, including 1,281,167 training examples and 50,000 testing images. Following [29, 27], we resize ImageNet-1K images to 64x64 resolution to match TinyImageNet. Compared to CIFAR10/100, TinyImageNet and ImageNet-1K are more challenging because of their diverse classes and higher image resolution. To further extend dataset distillation, we take a step forward by applying our method to even higher-resolution images, specifically 128x128 subsets of ImageNet. In previous dataset distillation research [2], subsets were introduced based on categories and aesthetics, encompassing birds, fruits, and cats. In this study, we utilize ImageNette (assorted objects), ImageWoof (dog breeds), and ImageSquawk (birds) to provide additional examples of our algorithm's effectiveness. For a detailed enumeration of ImageNet classes in each of our datasets, please refer to Table 1.

1.2. Data Preprocessing

We implemented a standardized preprocessing approach for all datasets, following the methodology outlined in [26]. To ensure optimal model performance during both training and evaluation, we utilized several popular transformations, including color jittering, cropping, cutout, scaling, and rotation, as differentiable augmentation strategies across all datasets. For the CIFAR10/100 datasets, we additionally applied Kornia zero-phase component analysis (ZCA) whitening, using the same setting as [2]. However, we refrained from using ZCA preprocessing for the medium- and high-resolution datasets due to the computational expense of the full-size ZCA transformation. As a result, the distilled images for these datasets display checkboard artifacts (see Figures 18, 19, 20, 21, and 22). It is worth noting that we visualized the distilled images by directly applying the reverse transformation based on the corresponding data preprocessing without any further modifications.

1.3. Implementations of Prior Works

To ensure fair comparisons with prior works, we obtained publicly available distilled data for each baseline method and trained models using our experimental setup. We utilized the same ConvNet architecture with three, four, or five layers, depending on the image resolutions, and applied the same

*Equal contribution

Dataset	0	1	2	3	4	5	6	7	8	9
ImageNette [12]	Tench	English Springer	Cassette Player	Chainsaw	Church	French Horn	Garbage Truck	Gas Pump	Golf Ball	Parachute
ImageWoof [12]	Australian Terrier	Border Terrier	Samoyed	Beagle	Shih-Tzu	English Foxhound	Rhodesian Ridgeback	Dingo	Golden Retriever	English Sheepdog
ImageSqauwk [2]	Peacock	Flamingo	Macaw	Pelican	King Penguin	Bald Eagle	Toucan	Ostrich	Black Swan	Cockatoo

Table 1: Class listings for our ImageNet subsets.

preprocessing technique across all methods. In cases where our results were comparable or inferior to those reported in the original papers, we presented their default numbers directly. Regarding the Kernel Inducing Points (KIP) method [19, 20], we made a slight modification by employing a 128-kernel ConvNet instead of the original 1024-kernel version. To ensure fairness in accordance with the Matching Training Trajectories (MTT) [2], we calculated performance based on the average test accuracy across 100 networks rather than relying on the best result reported in the paper. We did our best to reproduce prior methods that did not conduct experiments on some datasets by following the released author codes. However, for methods that encountered scalability issues on high-resolution datasets, we were unable to obtain the relevant performance scores.

1.4. Hyperparameters

In order to ensure that our methodology can be reproduced, we have included a Table 2 listing all the hyperparameters used in this work. For the baseline methods, we utilized the default parameters that the authors specified in their original papers. We used the same hyperparameter settings across all experiments, unless otherwise stated. Specifically, we employed an SGD optimizer with a learning rate of 1 for learning synthetic sets and a learning rate of 0.01 for training neural network models. For low-resolution datasets, we used a 3-layer ConvNet, while for medium- and high-resolution datasets, we followed the recommendation of [27] and used a 4-layer and 5-layer ConvNet, respectively. In all experiments, we used a mini-batch of 256 real images from each class to learn the synthetic set. Additionally, we conducted ablation studies on certain hyperparameters, such as task balance λ and the power parameter p in the Spatial Attention Matching (SAM) modules, which are discussed in Section 2.2.

2. Additional Results and Further Analysis

2.1. Comparison to More Baselines

We conducted a comparison between images created by the DataDAM and popular generative models such as variational auto-encoders (VAEs) [13, 21] and generative adversarial networks (GANs) [9, 18, 1, 17] to evaluate their data efficiency. For this purpose, we selected state-of-the-art models, including the DC-VAE [21], cGAN [18], BigGAN [1], and GMMN [17]. The DC-VAE generates a model with dual contradistinctive losses, which improves the generative autoencoder's inference and synthesis abilities simultaneously. The cGAN model is conditioned on both the generator and discriminator, while BigGAN uses differentiable augmentation techniques [26]. On the other hand, GMMN aims to learn an image generator that can map a uniform distribution to a real image distribution. We trained these models on the CIFAR10 dataset with varying numbers of images per class (1, 10, and 50 IPCs) using ConvNet's (3-layer) architecture [8] and evaluated their performance on real testing images. Our results, presented in Table 3, indicate that our proposed method significantly outperforms these generative models. The DataDAM generates superior training images that offer more informative data for training DNNs, while the primary goal of the generative models is to create realistic-looking images that can deceive humans. Therefore, the efficiency of images produced by generative models is similar to that of randomly selected coresets.

We also employed another baseline approach, which is learning synthetic images through distribution matching using vanilla maximum mean discrepancy [10] (MMD) in the pixel space. By utilizing MMD loss with a linear kernel, we achieved improved performance compared to randomly selected real images and generative models (see Table 3). However, DataDAM surpasses the results of vanilla MMD since it generates more informative synthetic images by utilizing the information of the feature extractor at various levels of representation.

IPC	Random	DC-VAE	cGAN	BigGAN	GMMN	MMD	DataDAM
1	14.4±2.0	15.7±2.1	16.3±1.4	15.8±1.2	16.1±2.0	22.7±0.6	**32.0±1.2**
10	26.0±1.2	29.8±1.0	27.9±1.1	31.0±1.4	32.2±1.3	34.9±0.3	**54.2±0.8**
50	43.4±1.0	44.0±0.8	43.8±0.9	46.2±0.9	45.3±1.0	50.9±0.3	**67.0±0.4**

Table 3: Comparison of the DataDAM's performance to popular generative models and the MMD baseline on the CIFAR10 dataset using ConvNets. The "Random" category denotes randomly selected real images.

2.2. More Ablation Studies

Evaluation of power parameter p in the SAM module. This section examines how the p-norm impacts the efficiency of spatial-wise attention maps in the SAM module. In Figure 1, we evaluate the testing accuracy of the DataDAM on CIFAR10 with IPC 10 for various values of p. Our method proves to be robust across a broad range of p values, indi-

Hyperparameters			Options/ Range	Value
Category	**Parameter Name**	**Description**		
Optimization	**Learning Rate η_S (images)**	Step size towards global/local minima	$(0, 10.0]$	IPC $\leq$ 50: 1.0 IPC $>$ 50: 10.0
	Learning Rate η_θ (network)	Step size towards global/local minima	$(0, 1.0]$	0.01
	Optimizer (images)	Updates synthetic set to approach global/local minima	SGD with Momentum	Momentum: 0.5 Weight Decay: 0.0
	Optimizer (network)	Updates model to approach global/local minima	SGD with Momentum	Momentum: 0.9 Weight Decay: $5e-4$
	Scheduler (images)	-	-	-
	Scheduler (network)	Decays the learning rate over epochs	StepLR	Decay rate: 0.5 Step size: 15.0
	Iteration Count	Number of iterations for learning synthetic data	$[1, \infty)$	8000
Loss Function	**Task Balance λ**	Regularization Multiplier	$[0, \infty)$	Low Resolution: 0.01 High Resolution: 0.02
	Power Value p	Exponential power for amplification in the SAM module	$[1, \infty)$	4
	Loss Configuration	Type of error function used to measure distribution discrepancy	-	Mean Squared Error
	Normalization Type	Type of normalization used in the SAM module on attention maps	-	L2
DSA Augmentations	**Color**	Randomly adjust (jitter) the color components of an image	brightness saturation contrast	1.0 2.0 0.5
	Crop	Crops an image with padding	ratio crop pad	0.125
	Cutout	Randomly covers input with a square	cutout ratio	0.5
	Flip	Flips an image with probability p in range:	$(0, 1.0]$	0.5
	Scale	Shifts pixels either column-wise or row-wise	scaling ratio	1.2
	Rotate	Rotates image by certain angle	$0° - 360°$	$[-15°, +15°]$
Encoder Parameters	**Conv Layer Weights**	The weights of convolutional layers	$\mathbb{R}$ bounded by kernel size	Uniform Distribution
	Activation Function	The non-linear function at the end of each layer	-	ReLU
	Normalization Layer	Type of normalization layer used after convolutional blocks	-	InstanceNorm

Table 2: Hyperparameters Details.

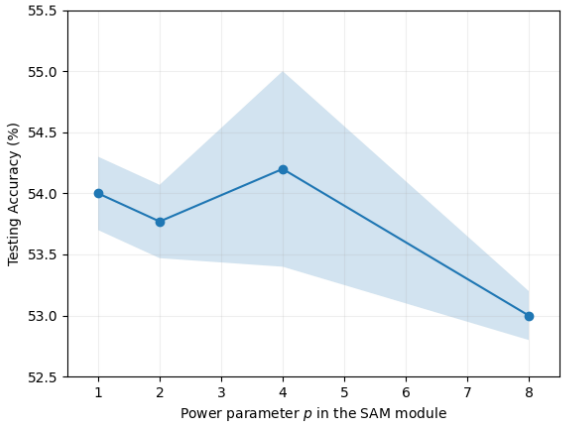


Figure 1: The effect of the power parameter p on the final testing accuracy (%) for the CIFAR10 dataset with IPC 10 configuration.

cating that it is not significantly affected by changes in the degree of discrepancy measured by $\mathcal{L}_{\text{SAM}}$. However, when the power is raised to 8, the DataDAM gives more weight to spatial locations that correspond to the neurons with the highest activations. In other words, it prioritizes the most discriminative parts, potentially ignoring other important components that may be crucial in approximating the data distribution. This could negatively impact the testing performance to some extent.

Exploring the effect of Gaussian noise initialization for synthetic images on DataDAM. To augment our results in the main paper, we present an extended training configuration for initialization from Gaussian noise. We conducted this experiment on CIFAR10 with IPC 50. As seen in Figure 2, the Gaussian noise initialization scheme takes longer to converge to a competitive accuracy level. Despite underperforming in comparison to Random and K-Center initialization, it still demonstrates the ability of our proposed method to distill information from the real dataset onto pure random noise. Moreover, it is capable of outperforming competitive methods, particularly KIP [19] and DSA [26]. In Figure 3, we provide visualizations of the synthetic data generated from random noise during different iterations. These visualizations highlight how our method successfully transfers information from the real dataset to the random noise, especially when comparing the initial noise image with the final iteration.

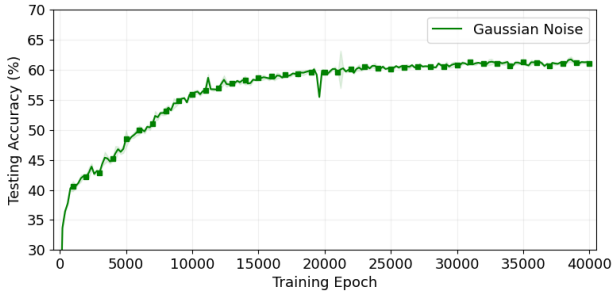


Figure 2: Test accuracy evolution of synthetic image learning on CIFAR10 with IPC 50 under Gaussian noise initialization.

(a) Iteration 0 (b) Iteration 600 (c) Iteration 2000 (d) Iteration 5000 (e) Iteration 15000 (f) Iteration 25000 (g) Iteration 35000 (h) Iteration 40000

Figure 3: The learning process of all classes in the CIFAR10 dataset (IPC 50) initialized from Gaussian noise. We take two random images for each class and visualize their progression over the 40,000 training epochs.

Exploring the effect of different augmentation strategies in DataDAM. In this section, we explore the impact of augmentation methods on the effectiveness of our approach when evaluated on the CIFAR10 dataset with an IPC 10 configuration. We treat our method as a black box, as in the work of [5], and assess the effects of various augmentation techniques such as AutoAugment [3], RandAugment [4], DSA [26], and no augmentation on the distilled datasets during the evaluation phase. The results are presented in Figure 4. Our observations reveal that DSA delivers significantly better performance as it is integrated into the training process of the synthetic dataset and is more compatible with the learning phase of the distilled images. Additionally, our findings indicate that augmentation is vital for training on synthetic data, as evidenced by the substantial differences between different augmentation methods and no augmentation. Therefore, applying augmentation techniques to our distilled images during evaluation can substantially enhance model performance.

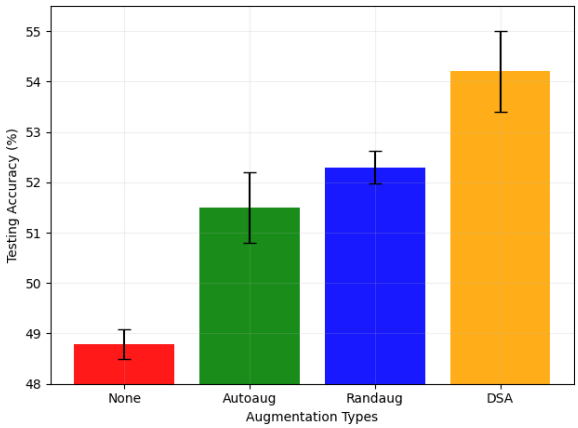


Figure 4: The effect of different augmentation strategies during the evaluation phase on the final testing accuracy (%) for the CIFAR10 dataset with IPC 10 configuration.

Exploring the effect of different loss configurations in $\mathcal{L}_{\text{SAM}}$. In this section, we explore the impact of different loss configurations on attention loss ($\mathcal{L}_{\text{SAM}}$). To conduct this evaluation, we employed mean absolute error (MAE), cosine dissimilarity, and mean square error (MSE) as objective functions for $\mathcal{L}_{\text{SAM}}$ to train a synthetic dataset on CIFAR10 with IPC 10. The results presented in Figure 5 demonstrate that MSE yields the best results. Nonetheless,

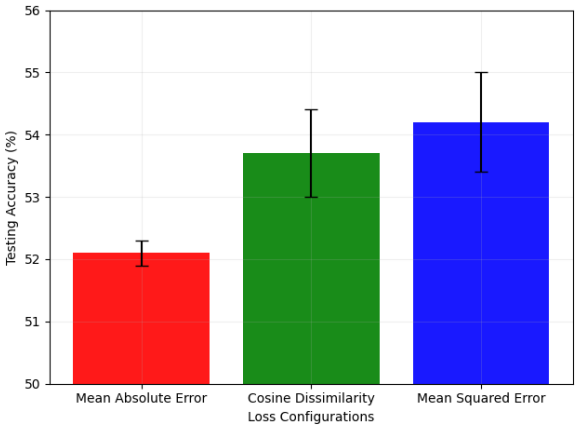


Figure 5: The effect of loss configurations of $\mathcal{L}_{\text{SAM}}$ on the final testing accuracy (%) for the CIFAR10 dataset with IPC 10 configuration.

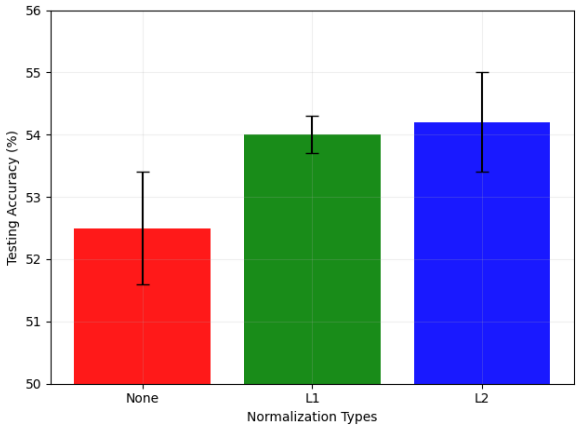


Figure 6: The effect of different normalization blocks of the SAM module on the final testing accuracy (%) for the CIFAR10 dataset with IPC 10 configuration.

it is crucial to note that even with any of these configurations, our method still outperforms most of the competitive methods (except the MTT [2]). Therefore, we can conclude that our approach performs well with any loss configuration, but a well-designed configuration can result in a substantial performance improvement of up to 2.0% in our ablation study.

Exploring the effect of normalization in the SAM module. In this section, we aim to evaluate the impact of the normalization block in the internal structure of the SAM module on testing accuracy. We conducted experiments by training distilled images for CIFAR10 with IPC 10 and testing three normalization techniques: L_1 normalization, L_2 normalization, and no normalization. The results, as shown in Figure 6, indicate that L_2 normalization is the most effective in terms of testing accuracy. By adding normalization, we reduce the magnitude of the attention loss $\mathcal{L}_{\text{SAM}}$ in backpropagation, thus decreasing the chance of overshooting the global minima in the optimization space when modifying the input image's pixels. We can observe that both normalization schemes work well, but the absence of normalization leads to significant performance degradation. Therefore, we conclude that while the appropriate use of normalization is critical for the performance of the DataDAM, the type of normalization is not as significant.

2.3. More Experiments and Analysis on Neural Architecture Search

Taking inspiration from [28, 26, 27], we define a search space consisting of 720 ConvNets on the CIFAR10 dataset. We evaluate the models using our distilled data with IPC 50 as a proxy set under the neural architecture search (NAS) framework. We start with a base ConvNet and construct a uniform grid that varies in depth $D \in \{1, 2, 3, 4\}$, width $W \in \{32, 64, 128, 256\}$, activation function $A \in \{\text{Sigmoid, ReLu, LeakyReLu}\}$, normalization technique $N \in \{\text{None, BatchNorm, LayerNorm, InstanceNorm, GroupNorm}\}$, and pooling operation $P \in \{\text{None, MaxPooling, AvgPooling}\}$. These candidates are then evaluated based on their validation performance and ranked accordingly.

In Figure 7, we displayed the performance rank correlation between the proxy set, generated using various methods, and the whole training dataset using Spearman's correlation across all 720 architectures. Each point in the graph represents a selected architecture. The x-axis represents the test accuracy of the model trained on the proxy set, while the y-axis represents the accuracy of the model trained on the whole dataset. Our analysis shows that all methods perform well. However, DataDAM has a higher concentration of dots close to the straight line, indicating a better proxy set for obtaining more reliable performance rankings of candidate architectures. These results are on par with the DataDAM's performance correlation (0.72), which is higher than other prior works. To further assess the effectiveness of our approach, we conducted an analysis of the top 20% of the search space, selecting 144 architectures with the highest validation accuracy. As depicted in Figure 8, our method outperforms most of the state-of-the-art methods, except for early stopping, where we only beat it by a small margin. Our evaluation of the correlation graphs indicates that DataDAM is capable of accurately correlating the performance of models trained on the proxy dataset with their performance on the whole training dataset. We substantiate these findings by presenting quantitative results of performance and Spearman's correlation in Table 4.

	Random	DSA	DM	CAFE	Ours	Early-stopping	Whole Dataset
Performance (%)	88.9	87.2	87.2	83.6	**89.0**	88.9	89.2
Correlation Top 20%	0.44	0.57	0.51	0.36	**0.69**	0.64	1.00
Time cost (min)	33.0	31.2	32.2	**30.7**	34.8	37.1	5168.9
Storage (imgs)	**500**	**500**	**500**	**500**	**500**	5×10^4	5×10^4

Table 4: Neural architecture search on CIFAR10 with a search space of the top 20% of the sample space with the highest validation accuracy.

Experiments on NAS-Bench-201. To conduct a more comprehensive analysis of the neural architecture search, we expanded the search space by including NAS-Bench-201 [7] as recommended in [5]. Our aim is to compare the performance of DataDAM against other methods using the CIFAR10 dataset with IPC 50 as the proxy set. To create a search space, we randomly selected 100 networks from the 15,635 available models in NAS-Bench-201. We followed the configuration and settings presented in [5], which involve training all models using five random seeds and ranking them based on their average accuracy on a validation set comprising 10,000 images. We used two metrics to evaluate the effectiveness of NAS: the performance correlation ranking between models trained on synthetic and real datasets and the top-1 performance in the search space. In contrast to the previous search space that concentrated on 720 ConvNet architectures, we observed a distinct trend in this larger NAS benchmark with modern architectures. According to Table 5, while most methods achieved negative correlations between performance on the proxy set and the entire dataset, our method had a small positive correlation and obtained competitive outcomes on the original dataset. This implies that DataDAM preserves the true strength of the underlying model more effectively than previous works. Nevertheless, despite the encouraging performance gains achieved by the best single model, utilizing the distilled data to guide model design remains a significant challenge. It is important to mention that the rank correlation presented in Table 5 for the original real dataset is not 1.0. This is because a smaller architecture was used and the ranking was based on a validation set, as pointed out in [5].

	Random	DC	DSA	DM	KIP	MTT	DataDAM	Whole Dataset
Correlation	-0.06	-0.19	-0.37	-0.37	-0.50	-0.09	**0.07**	0.7487
Top 1 (%)	91.9	86.44	73.54	92.16	92.91	73.54	**93.96**	93.5

Table 5: Spearman's rank correlation results were obtained using NAS-Bench-201. The best performance achieved on the test set is 94.36% [5].

3. Additional Visualizations and Analysis

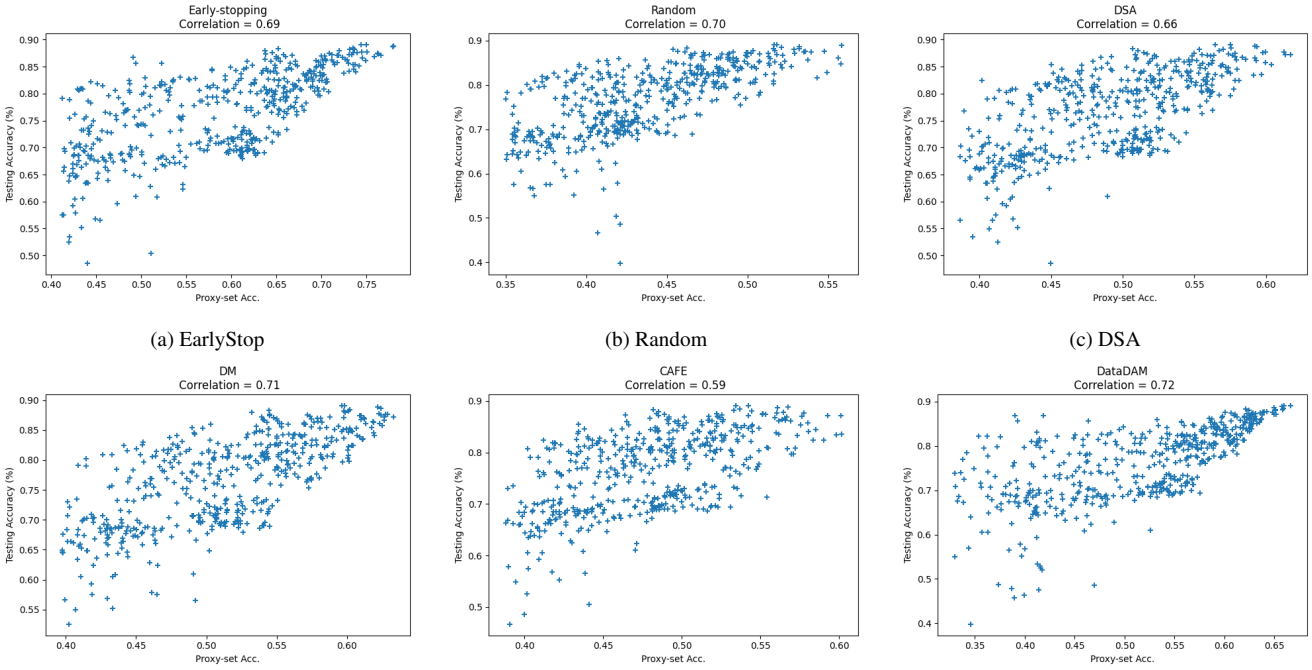
3.1. More Analysis on Data Distribution

To complement the data distribution visualization results presented in the main paper, we have included t-SNE [24] illustrations for all categories in Figure 9. We utilized t-SNE to show the features of real and synthetic sets generated by DC [28], DSA [26], DM [27], CAFE [25], and DataDAM in the embedding space of the ResNet-18 [11] architecture. The visualizations were applied to the CIFAR10 dataset with IPC 50 for all methodologies. As depicted in Figure 9, our approach, similar to DM, preserves the distribution of data with a well-balanced spread over the entire dataset. Conversely, other methods, such as DC, DSA, and CAFE, exhibit a significant bias toward the boundaries of certain clusters and have high false-positive rates for the majority of the classes. To put it simply, the t-SNE visualization validates that our method maintains a considerable degree of impartiality in accurately capturing the dataset distribution uniformly across all categories.

3.2. Extended Visualizations of Synthetic Images

Visualization of the synthetic images trained with different model architectures in DataDAM. In this section, we present a qualitative comparison of the generated distilled images using different architectures to demonstrate how the choice of architecture influences the quality of the synthetic set. We assess the efficacy of the distilled data trained using ConvNet [8], AlexNet [15], and VGG-11 [23] architectures on the CIFAR10 dataset with IPC 50. Our results, as depicted in Figure 10, reveal that the distilled data can encode the inductive bias of the chosen architecture. Specifically, the distilled images produced by the simplest architecture, i.e., ConvNet [8], exhibit a natural appearance and can transfer well to other architectures (see Table 3 of the main paper). In contrast, the distilled images generated by modern architectures like VGG-11 [23] exhibit different brightness and contrast than natural images. We found that increasing the complexity and number of convolutional layers in the feature extraction process led to brighter and more contrasting distilled images. This is likely because the attention loss ($\mathcal{L}_{\text{SAM}}$) becomes more potent, resulting in a more substantial modulation effect on the input image pixels during backpropagation. This trend is noticeable in the distilled images generated by AlexNet [15] and VGG-11 [23]. We note that the synthetic images may reflect the similarity between architectures, as evidenced by the similarity between the images produced by AlexNet and ConvNet. This finding suggests that the inductive biases of these two architectures are comparable.

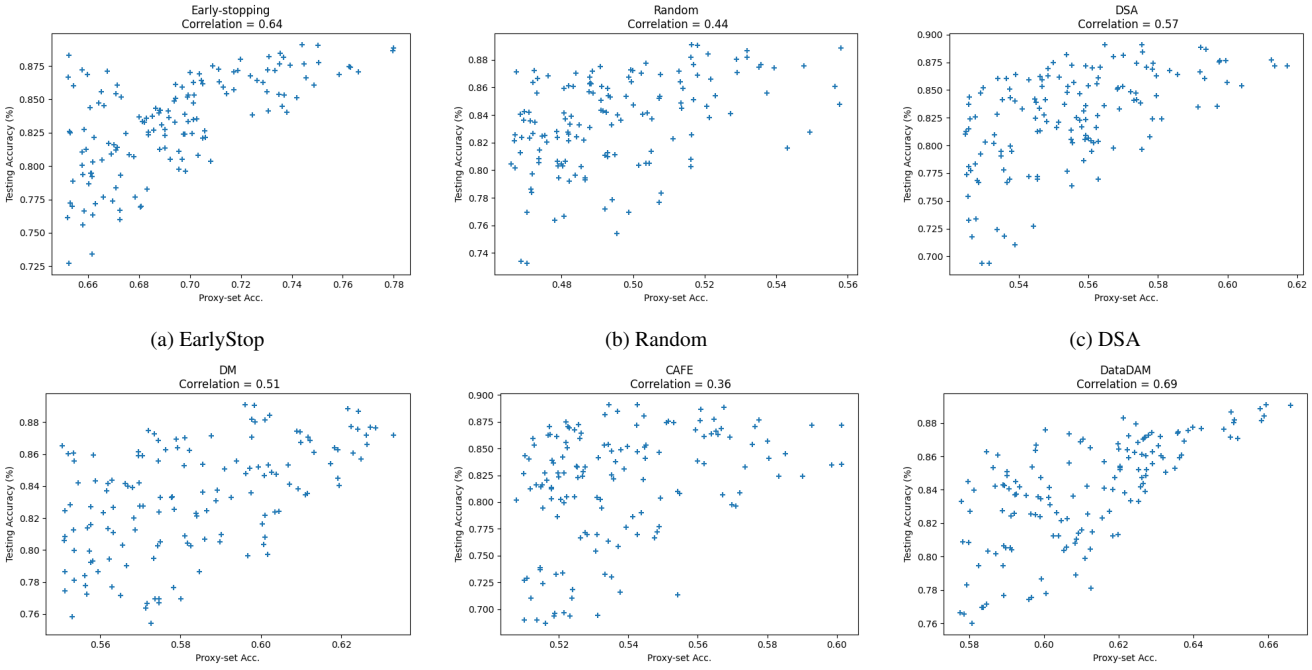
Visualization of the synthetic images trained with different loss components in DataDAM. This section involves a comparison of the synthetic images generated by utilizing different loss objectives, namely only $\mathcal{L}_{\text{MMD}}$, only $\mathcal{L}_{\text{SAM}}$, layer-wise feature map transfer loss, and the DataDAM loss. The CIFAR10 dataset with IPC 10 was used for this evaluation to qualitatively assess the contribution of each loss component. As shown in Figure 11, the visualization of DataDAM is a linear combination of the $\mathcal{L}_{\text{SAM}}$ and $\mathcal{L}_{\text{MMD}}$ visualizations, resulting in a brighter and more contrasted image compared to each loss component individually. The



(a) EarlyStop (b) Random (c) DSA

(d) DM (e) CAFE (f) DataDAM

Figure 7: Performance rank correlation between proxy set and whole dataset training across all 720 architectures.



(a) EarlyStop (b) Random (c) DSA

(d) DM (e) CAFE (f) DataDAM

Figure 8: Performance rank correlation between proxy-set and whole-dataset training across the top 20% of the search space (selecting 144 architectures with the highest validation accuracy).

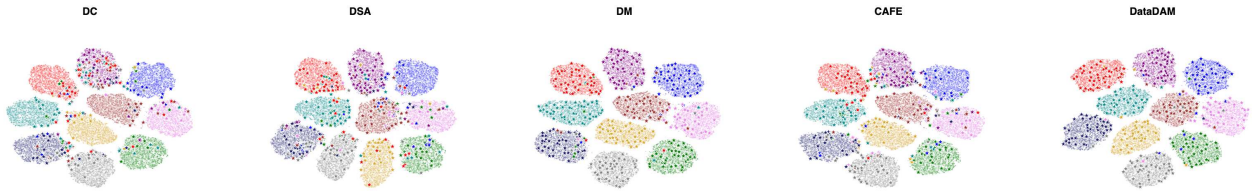


Figure 9: Distributions of the synthetic images learned by five methods on the CIFAR10 dataset with IPC 50. The stars represent the synthetic data dispersed amongst the original dataset. The classes are as follows: plane, car, bird, cat, deer, dog, frog, horse, ship, truck.

generated synthetic sets by $\mathcal{L}_{\text{SAM}}$ and layer-wise feature transfer loss are somewhat similar since both losses match the information of feature maps generated by the real and synthetic datasets. However, the images distilled by $\mathcal{L}_{\text{SAM}}$ are brighter and more contrasted due to the matching of the most discriminative parts of the images.

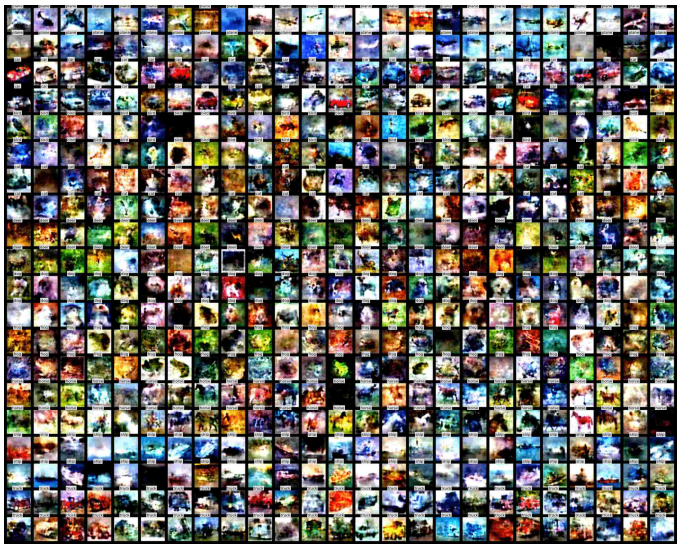
Visualization of the synthetic images trained with different layers in DataDAM. We conducted an experiment to analyze the distilled images produced by matching different layers of the ConvNet on real and synthetic datasets. Our study focused specifically on the CIFAR10 dataset with IPC 10. Figure 12 demonstrates that the layers performed differently as each layer conveyed distinct information regarding the data distributions. Our approach, DataDAM, utilizes all intermediate and final layers, resulting in distilled images that possess greater brightness and contrast. This is primarily due to the matching of attention maps in each layer as well as the embedding representation of the final layer.

Visualization of the synthetic images trained with different initialization strategies in DataDAM. In this section, we presented the distilled images for the CIFAR10 dataset generated by IPC 50 using three distinct initialization methods: Random, K-Center [22, 5], and Gaussian noise. Figure 13 illustrates the learned representations of the synthesis images produced using each initialization strategy. We observed a striking resemblance between the distilled images obtained through Random and K-Center initialization, which further confirms the results presented in the main paper. In contrast, the images generated using Gaussian noise initialization have noticeable differences in comparison to others, but they have still been learned effectively, and they contain crucial information for each class. In summary, these qualitative observations provide additional evidence that our model is robust enough to handle variations in initialization conditions.

More distilled image visualization. We provide additional visualizations of the distilled images for all five datasets used in this work, namely CIFAR10 (Figures 14, 15), CIFAR100 (Figures 16, 17), TinyImageNet (Figure 18), ImageNet-1K (Figure 19), ImageNette (Figure 20), ImageWoof (Figure 21), and ImageSquawk (Figure 22).

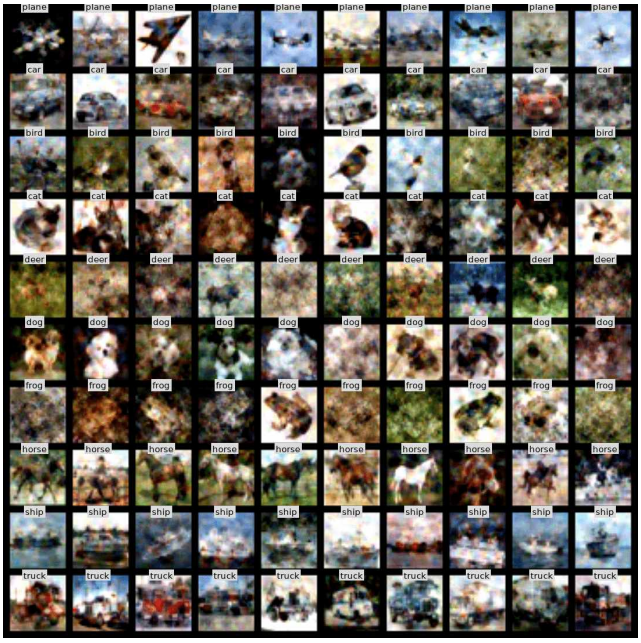
(a) ConvNet

(b) AlexNet



(c) VGG-11

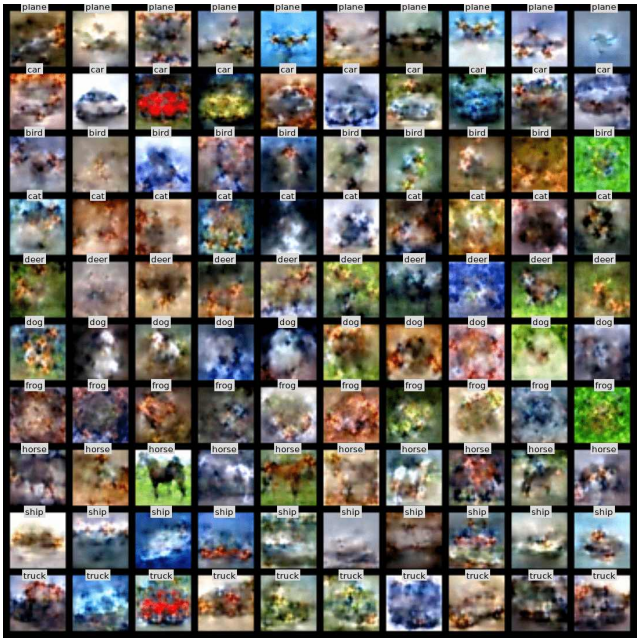
Figure 10: Learned synthetic images with different model architectures on the CIFAR10 dataset with IPC 50.



(a) $\mathcal{L}_{\text{MMD}}$

(b) $\mathcal{L}_{\text{SAM}}$

(c) Feature Map Transfer Loss



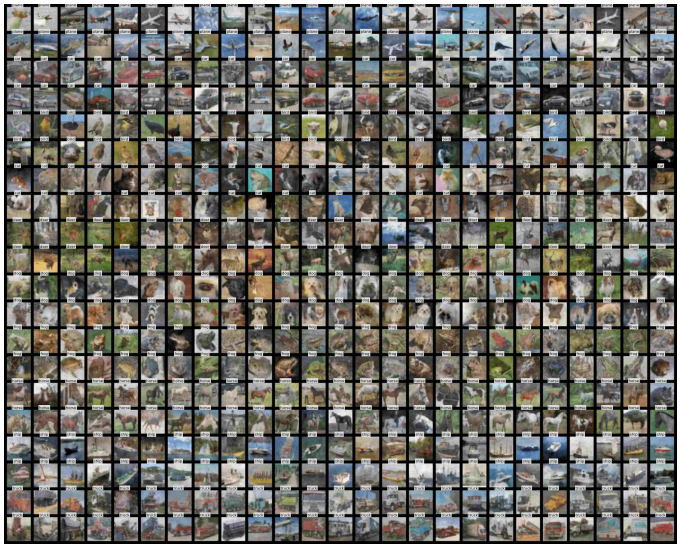
(d) DataDAM

Figure 11: Learned synthetic images with different loss functions on the CIFAR10 dataset with IPC 10.

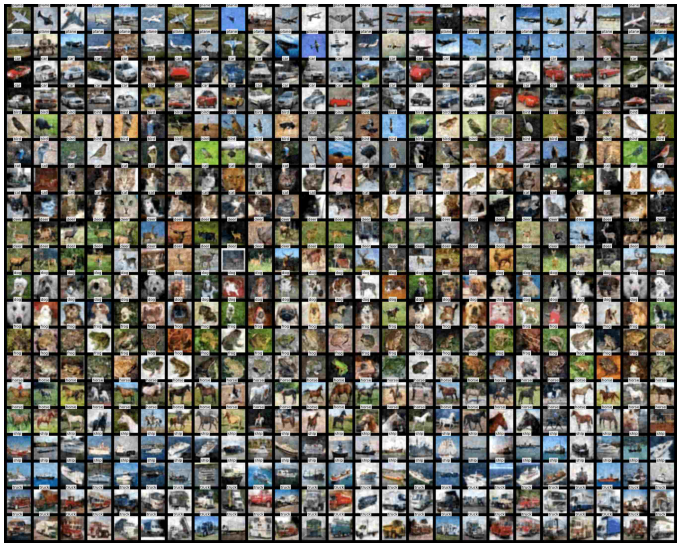
(a) Last layer ($\mathcal{L}_{\text{MMD}}$) (b) Layer 1 and Last Layer (c) Layer 2 and Last Layer

(d) Layer 1 and Layer 2 ($\mathcal{L}_{\text{SAM}}$) (e) All layers (DataDAM)

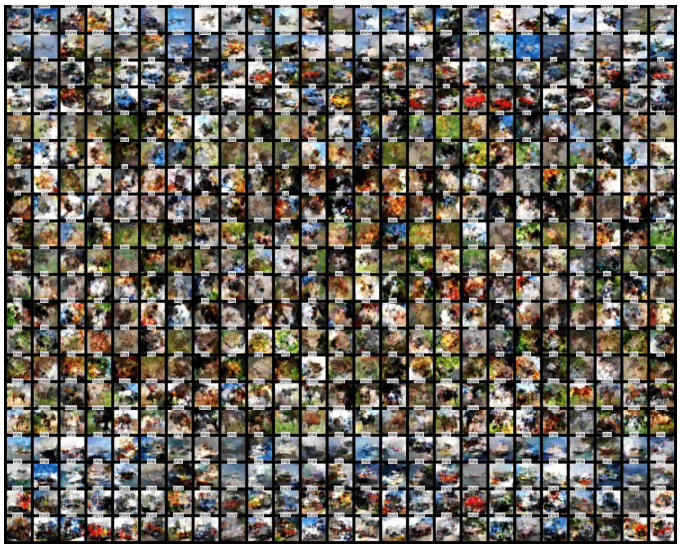
Figure 12: Learned synthetic images with different matching layers on the CIFAR10 dataset with IPC 10.



(a) Random Initialization



(b) K-Center Initialization



(c) Gaussian noise Initialization

Figure 13: Learned synthetic images with different initialization strategies on the CIFAR10 dataset with IPC 50.

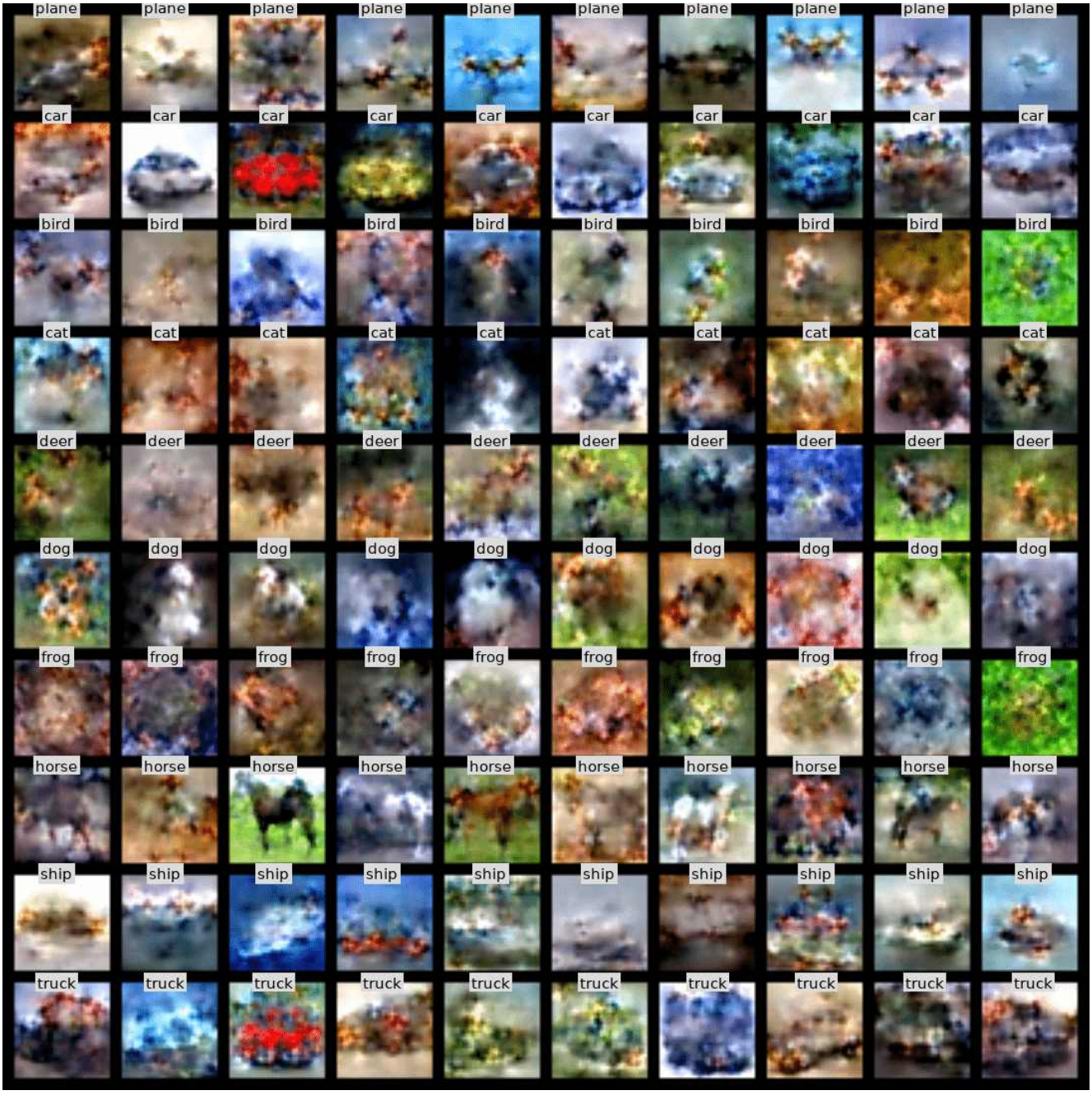


Figure 14: Distilled Image Visualization: CIFAR10 dataset with IPC 10.

Figure 15: Distilled Image Visualization: CIFAR10 dataset with IPC 50.

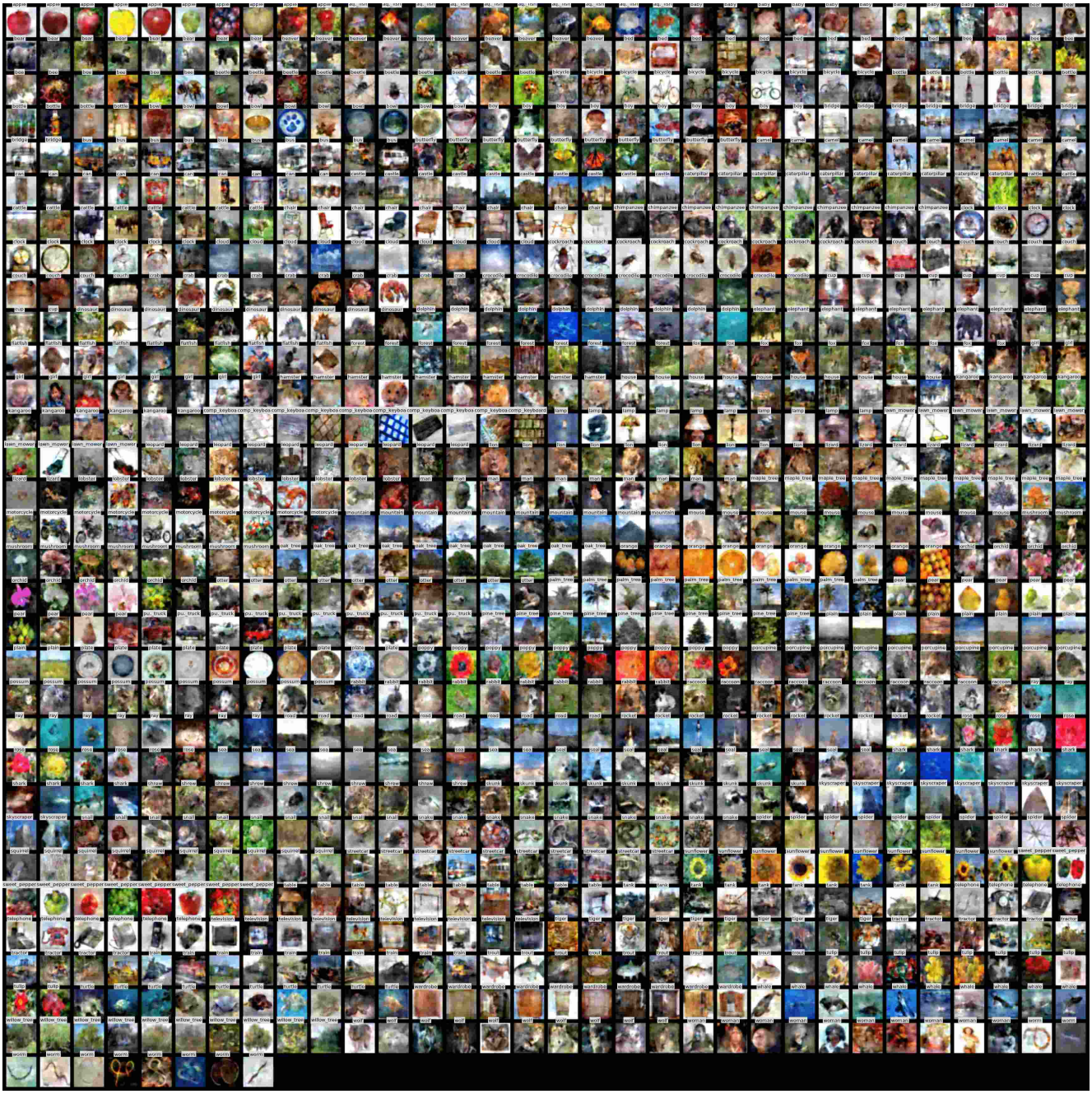


Figure 16: Distilled Image Visualization: CIFAR100 dataset with IPC 10.

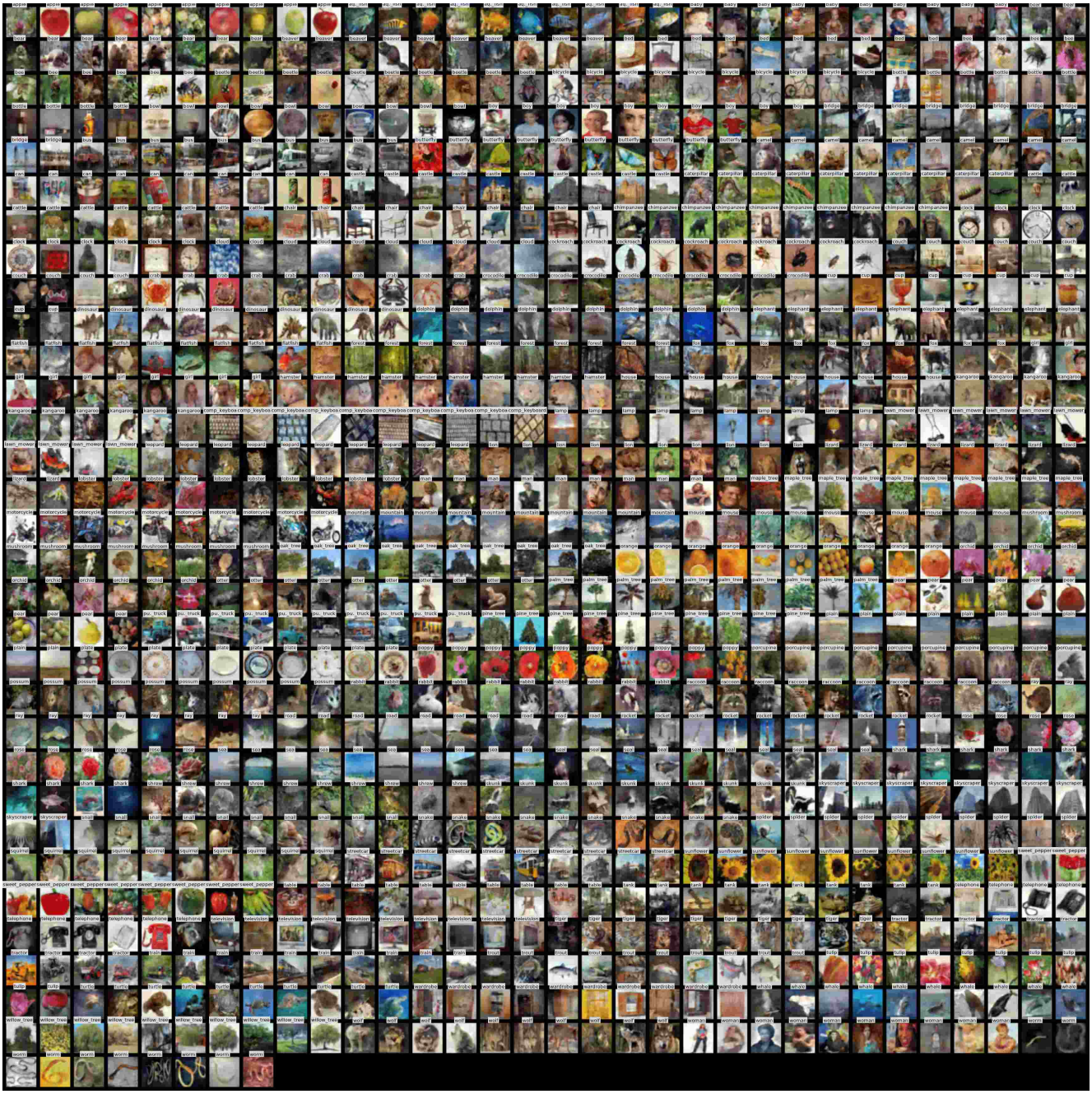


Figure 17: Distilled Image Visualization: CIFAR100 dataset with IPC 50 (10 randomly selected images for each class).

Figure 18: Distilled Image Visualization: TinyImageNet dataset with IPC 1.

Figure 19: Distilled Image Visualization: ImageNet-1K dataset with IPC 1.

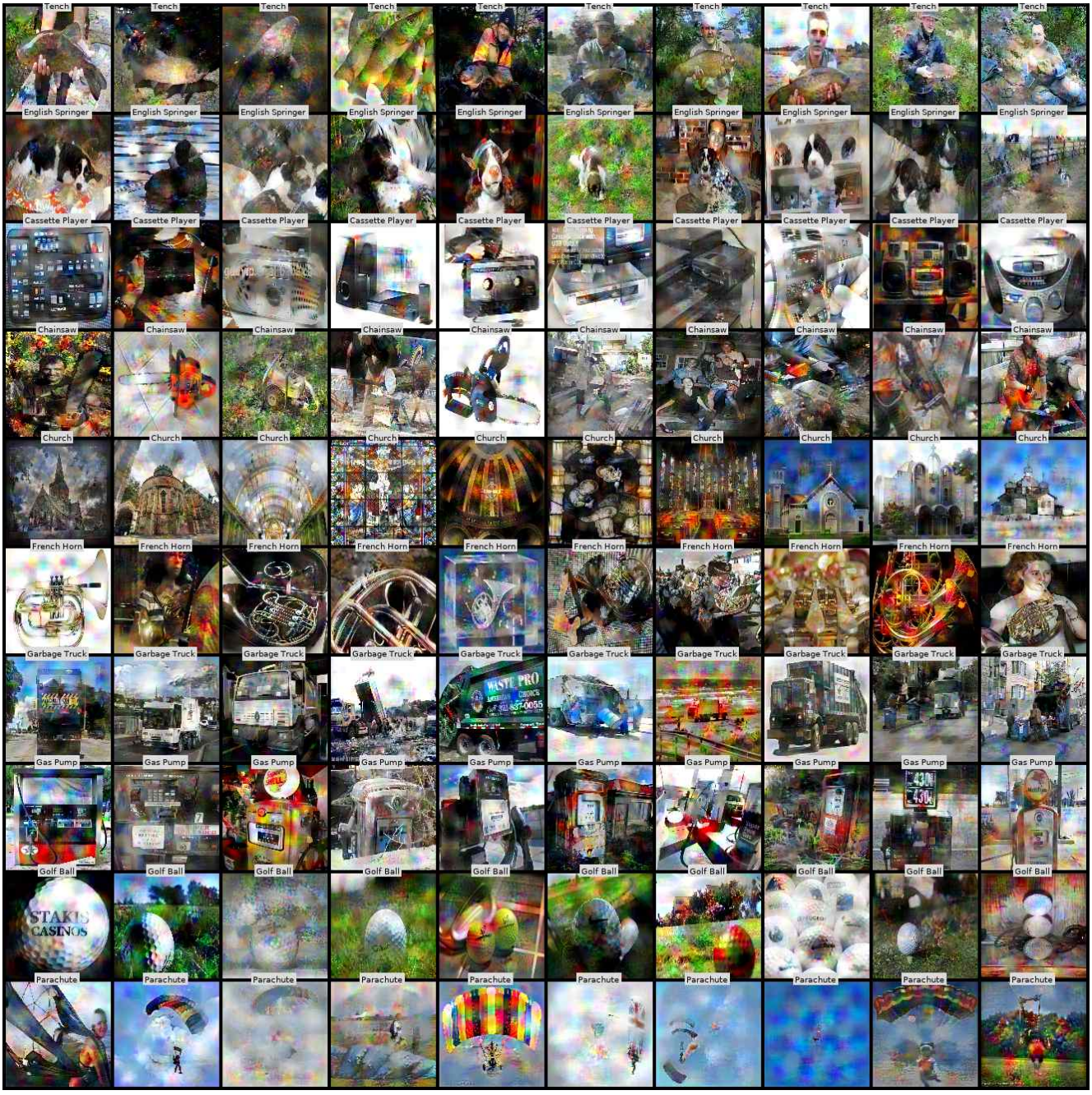


Figure 20: Distilled Image Visualization: ImageNette dataset with IPC 10.

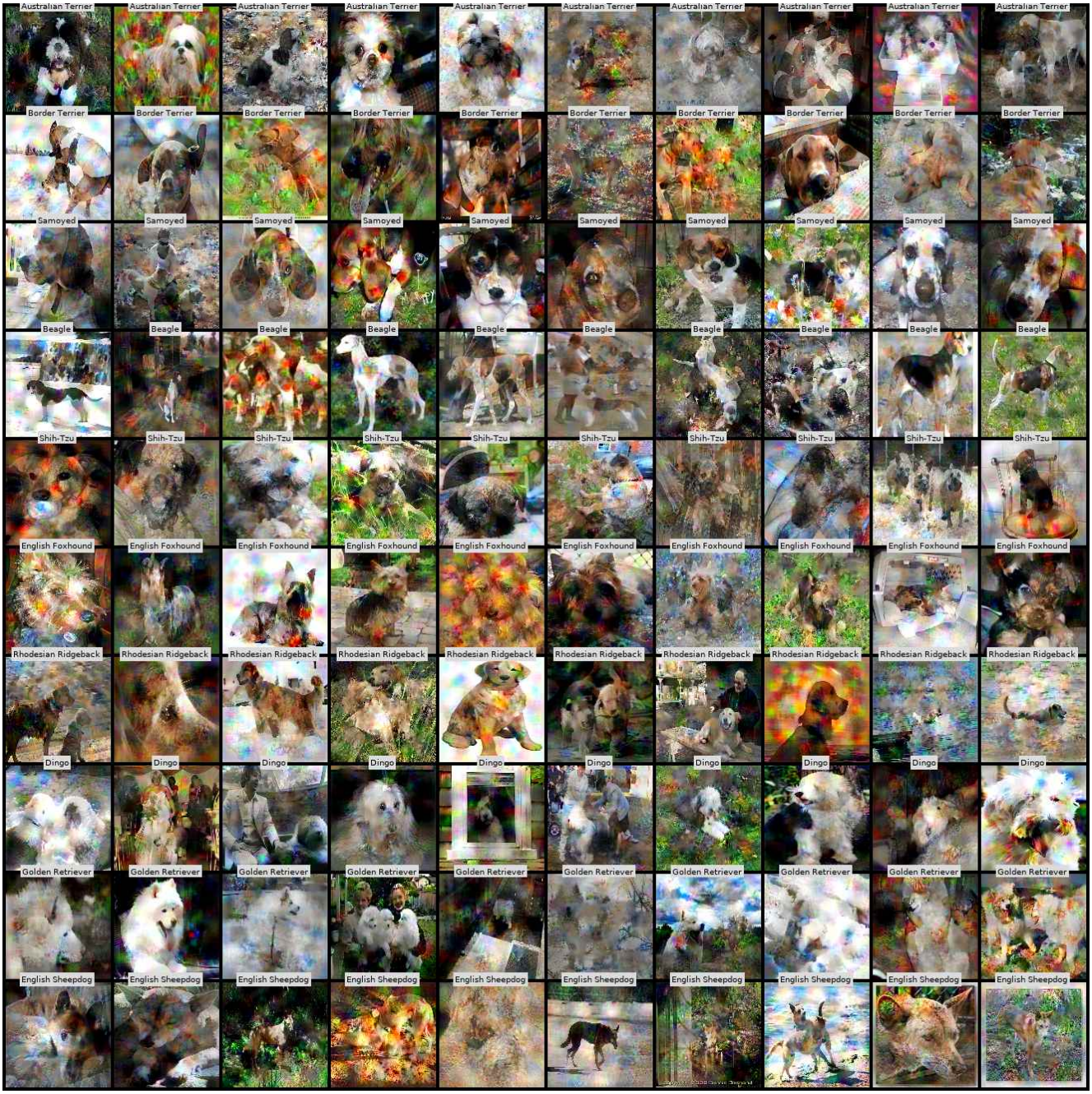


Figure 21: Distilled Image Visualization: ImageWoof dataset with IPC 10.

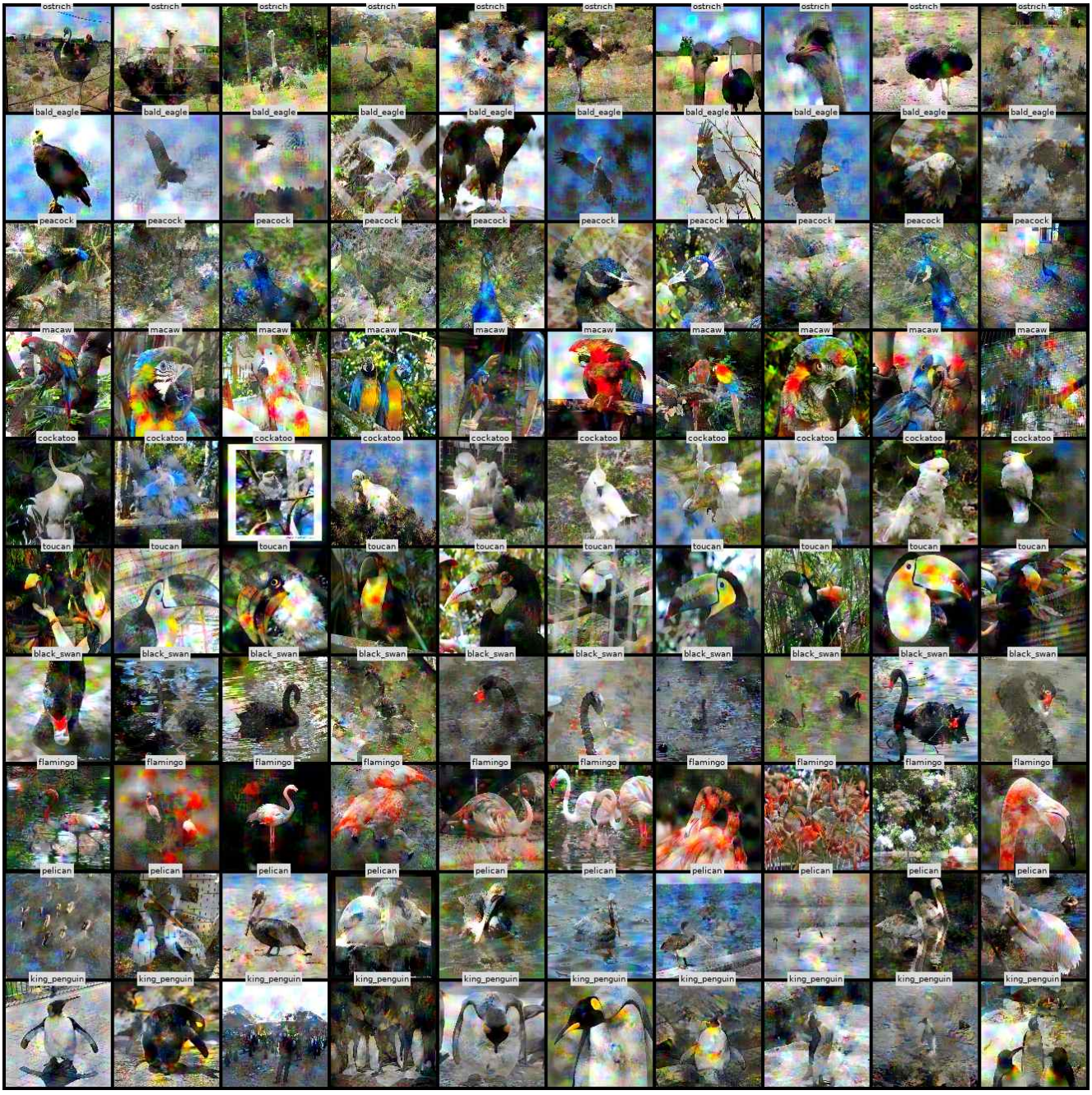


Figure 22: Distilled Image Visualization: ImageSquawk dataset with IPC 10.

References

[1] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019. 2

[2] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4750–4759, 2022. 1, 2, 5

[3] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018. 4

[4] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 702–703, 2020. 4

[5] Justin Cui, Ruochen Wang, Si Si, and Cho-Jui Hsieh. Dc-bench: Dataset condensation benchmark. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. 4, 6, 8

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1

[7] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations*, 2020. 6

[8] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4367–4375, 2018. 2, 6

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. 2

[10] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012. 2

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[12] J Howard. Imagenette: A smaller subset of 10 easily classified classes from imagenet, and a little more french, 2019. 1, 2

[13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[14] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 1

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 6

[16] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015. 1

[17] Yujia Li, Kevin Swersky, and Rich Zemel. Generative moment matching networks. In *International conference on machine learning*, pages 1718–1727. PMLR, 2015. 2

[18] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 2

[19] Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. Dataset meta-learning from kernel-ridge regression. In *International Conference on Learning Representations*, 2021. 2, 3

[20] Timothy Nguyen, Roman Novak, Lechao Xiao, and Jaehoon Lee. Dataset distillation with infinitely wide convolutional networks. *Advances in Neural Information Processing Systems*, 34:5186–5198, 2021. 2

[21] Gaurav Parmar, Dacheng Li, Kwonjoon Lee, and Zhuowen Tu. Dual contradistinctive generative autoencoder. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 823–832, 2021. 2

[22] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018. 8

[23] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 6

[24] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 6

[25] Kai Wang, Bo Zhao, Xiangyu Peng, Zheng Zhu, Shuo Yang, Shuo Wang, Guan Huang, Hakan Bilen, Xinchao Wang, and Yang You. Cafe: Learning to condense dataset by aligning features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12196–12205, 2022. 6

[26] Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation. In *International Conference on Machine Learning*, pages 12674–12685. PMLR, 2021. 1, 2, 3, 4, 5, 6

[27] Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6514–6523, 2023. 1, 2, 5, 6

[28] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *Ninth International Conference on Learning Representations 2021*, 2021. 5, 6

[29] Yongchao Zhou, Ehsan Nezhadarya, and Jimmy Ba. Dataset distillation using neural feature regression. In *Advances in Neural Information Processing Systems*, 2022. 1