A D^3 ALGORITHM

Algorithm 1 Distributional Dataset Distillation

Require: Distillation Algorithm `MTT`; Distribution Matching Algorithm `DM`; Expert trajectories set $\mathcal{E}$. Total number of classes C; Generator Neural Network $f(\cdot;\boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$; number of training steps N; Per-class latent distribution parameters $\{(\mu_c^i, \Sigma_c^i)_{i=1}^{\text{PPC}}\}_{c=1}^{C}$

1: **for** i = 1, ... N **do**
2: $\quad\mathcal{L}_{\text{MTT}} = \texttt{MTT}(P_{\mathcal{S}}, \mathcal{E})$ ▷ Compute MTT and KL Loss
3: $\quad(\mu_c, \Sigma_c, \boldsymbol{\theta}) \leftarrow (\mu_c, \Sigma_c, \boldsymbol{\theta}) - \nabla_{\mu_c,\Sigma_c,\boldsymbol{\theta}}\mathcal{L}_{\text{MTT}}$ ▷ Update distilled distribution w.r.t $\mathcal{L}_{\text{MTT}}$
4: $\quad\mathcal{L}_{\text{MMD}} = \texttt{MMD}(P_{\mathcal{S}}, \mathcal{D})$ ▷ Compute DM Loss
5: $\quad(\mu_c, \Sigma_c, \boldsymbol{\theta}) \leftarrow (\mu_c, \Sigma_c, \boldsymbol{\theta}) - \nabla_{\mu,\Sigma,\boldsymbol{\theta}}\mathcal{L}_{\text{DM}}$ ▷ Update distilled distribution w.r.t $\mathcal{L}_{\text{DM}}$
6: **end for**
7: **Return:** latent prior distribution per class $\{(\mu_c^i, \Sigma_c^i)_{i=1}^{\text{PPC}}\}_{c=1}^{C}$ and posterior distribution $f(\cdot;\boldsymbol{\theta})$

B DETAILS ON THE DECODER

Our decoder is adopted from the decoder part of the VAE designed by Li et al. (2017), with small modifications. First, we project the 64 dimensional latent z in to a 256 dimension feature vector, which is then fed into a sequence of 2D `ConvTranspose` blocks. Each of the decoder block contains a `ConvTranspose` layer followed by a `BatchNorm` layer and a LeakyReLU activation. We inherent the default parameter choice for those layers: starting from the dimension of feature vector(256 channels), each `ConvTranspose` layer reduces the channel number from the previous block by half. After the those blocks, there is a 2D convlutional layer followed by a tanh activation. The exact dimension of the convolution layer differs by image output size. The original VAE was designed only for images with size 32×32, and used only 3 blocks. We also increase the number of deconv blocks for larger datasets: 4 for TinyImageNet and 5 for ImageNet subsets.

C DETAILS ON EXPERIMENT SETUP

In this section, we provide a detailed description on experiment setups for all experiment resuls presented in the paper.

Dataset **CIFAR-10** stands as the smallest dataset in our suite of experiments. It encompasses a corpus of 50,000 training images, each with dimensions of 32×32. **TinyImageNet**, in contrast, presents a more extensive dataset, consisting of 100,000 images distributed across 200 classes. The images within TinyImageNet are characterized by larger dimensions, measuring 64×64. In comparison to CIFAR-10, TinyImageNet exhibits enlargement across all three dimensions: image size, class count, and image quantity. Finally, we extend our investigation to include two renowned subsets of **ImageNet**: ImageNette and ImageWoof. In line with established practices from prior work, we resize the images within both subsets of ImageNet to dimensions of 128×128. Each subset comprises 10 classes in their respective training sets, with ImageNette possessing a total size of 12,894 images, and ImageWoof containing 12,454 images.

Dataset preprocessing For all three datasets, only a simple channel-based mean-variance scaling is performed as the preprocessing step. For ImageNet subsets, we crop the data into size 128×128. For CIFAR-10 we perform ZCA whitening as done in all data distillaion work (Nguyen et al. (2020), Nguyen et al. (2021)) using Kornia implementation with default parameters (Riba et al. (2020). To generated experts used in MTT, we also perform random simple augmentations to the images, including rotations, flip, crop, and color changes. The preprocessing step is chosen to mirror the baselines we make direct comparisons to.

Student network architecture The student network is a neural network consists of multiple convet blocks, and we call them ConvNet. The ConvNet configuration consists of multiple convolutional blocks, each housing a convolutional layer, a normalization layer, ReLU activation, and an average pooling layer. For larger datasets, we increase the number of convolutional blocks used in the

ConvNet. For CIAR10 we use ConvNet with 3 convolutional blocks, and for TinyImageNet we use 4 convolution blocks. For ImageNet subsets, we use 5 convolutional blocks. In our DM objective, we use the features generated by those convolutional blocks to compute MMD. Finally, a linear layer with Softmax activation is used to map the features generated by convolutional blocks into class prediction.

Training The distillation time is not the primary concern for data distillation tasks since it only needs to be done once for all downstream tasks. However, methods that are overly expensive to train might become infeasible when distilling large datastes. Because we compute and back propagate on both MTT and DM losses, our compute time is comparable to both method combined. For CIFAR-10, our method converges around 10,000 steps, totaling around 10 GPU hours. For TinyImageNet, our method converges around 20,000 steps, totalling around 160 GPU hours. Finally, for the two ImageNet subsets, our method converges around 3,000 steps, totalling around 20 GPU hours.

Evaluation Our evaluation is systematically structured as follows:

1. Learning Distilled Distributions: We execute the method delineated in Algorithm 1 to learn distilled distributions tailored to each dataset under consideration.
2. Training Student Networks: We initiate the training of five neural networks with randomly initialized parameters from scratch. These student networks may adhere to the same architectural encountered during distillation (i..e, ConvNet, see section 4.1), or they may deviate into different choices (i.e., cross architecture generalization section 4.2).
3. Validation of Performance: To train each student network, we draw data samples from the distilled distribution as the training dataset. We assess their performance on the original validation dataset.

For evaluation, we use SGD optimizer with momentum 0.9 and weight decay 5×10^{-4}. We only allow hyper-parameter tuning on the learning rate and we train student networks until convergence.

D Ablation Study

D.1 Loss function contribution

To understand the contribution of each loss terms, we repeat Algo. 1 to performance distillation on CIFAR-10 with subsets of different loss terms. Table 7 exhibits test performance of 1 PPC and Table 8 exhibits test performance for 2 PPC, where there is an additional diversity term in the loss function. Both tables indicate that MTT loss generate distributions that work well on seen architecture, but fail to generalize to new architectures. On the other hand, DM loss generates distributions that generalize well to unseen architectures but overall converge to a lower distillation quality. The sub-par performance for DM on seen architecture is more prominent at higher PPC case, as reflected in Table 8. However, our when we use MTT and DM together, the distilled distributions show a clear improvement from both loss terms alone. Table 9 shows that diversity loss contributed to a small but consistent increase in the overall distillation quality.

Table 7: Results (Test Accuracy%) of ablation study on the contribution of each loss terms at 1 PPC scenarios using CIFAR-10

	Evaluation Model			
Loss Terms Used	ConvNet	ResNet18	VGG11	AlexNet
DM	57.4 (0.23)	48.1 (0.33)	42.6 (0.12)	35.6 (0.43)
MTT	56.5 (0.22)	43.7 (0.15)	48.6 (0.2)	27.7 (0.22)
MMT and DM	61.2 (0.14)	56.5 (0.35)	53.5 (0.47)	42.4 (0.68)

D.2 DM Feature Space

By using the same experts from MTT, the distillation algorithm has only seen one type of architecture throughout the whole distillation process. However, our experiments have shown that the DM loss term provides significant improvements to the cross-architecture generalizability of the distilled data.

Table 8: Results (Test Accuracy%) of ablation study on the contribution of each loss terms at 2 PPC scenarios using CIFAR-10

	Evaluation Model			
Loss Terms Used	ConvNet	ResNet18	VGG11	AlexNet
DM	55.6 (0.34)	44.9 (0.62)	43.9 (0.30)	42.4 (0.66)
MTT	61.5 (0.22)	47.3 (0.47)	47.6 (0.29)	27.6 (0.50)
MMT, DM	63.1 (0.51)	57.5 (0.39)	55.5 (0.22)	45.2 (0.24)
MMT, DM, Diversity	64.98 (0.14)	61.07 (0.30)	59.50 (0.53)	51.80 (1.80)

Table 9: Results (Test Accuracy%) of ablation study on the contribution of diversity loss terms at various PPC scenarios using CIFAR-10

	PPC		
Loss Terms Used	2	3	5
MMT, DM	63.1 (0.31)	66.9 (0.23)	65.6 (0.12)
MMT, DM, Diversity	64.98 (0.14)	67.6 (0.2)	68.1 (0.2)

In our model, we used the feature space mapped by the pre-trained convolutional blocks in ConvNet. Note that those features are directly passed into a single FC-layer for the classification task. In general, one can take any intermediate layer output as the image feature. In this ablation task, we test whether a shallower feature space mapping would impact the quality of the DM loss. In Table 10, we used feature mapping from only two convolutions blocks out of three. We see that a shallower feature mapping produces a much worse distillation outcome.

Table 10: Results (Test Accuracy%) of ablation study on the feature mapping of MMD using CIFAR-10

	Evaluation Model			
DM feature space	ConvNet	ResNet18	VGG11	AlexNet
Deep	61.2 (0.14)	56.5 (0.35)	53.5 (0.47)	42.4 (0.68)
Shallow	57.27 (0.18)	38.68 (0.50)	41.65 (0.33)	26.14 (0.27)

E GENERATED SAMPLES FROM THE DISTILLED DISTRIBUTION

Figure 3: ImageWoof 1 PPC average representations (column 1) followed by their corresponding variations (colum 2 -10)

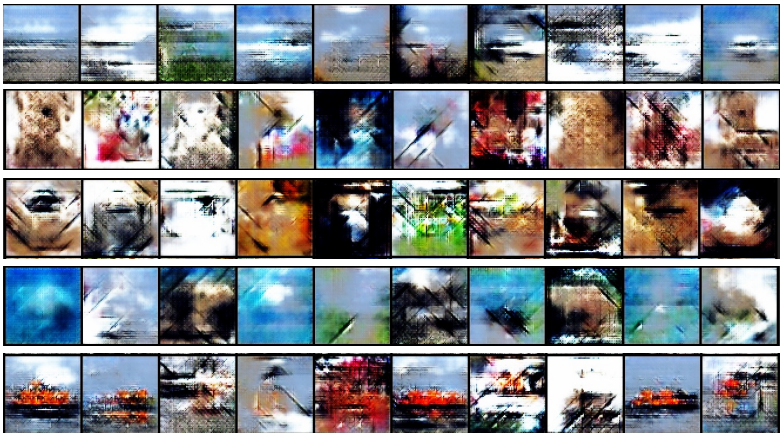

Figure 4: TinyImageNet 1 PPC average representations (column 1) followed by their corresponding variations (colum 2 -10)

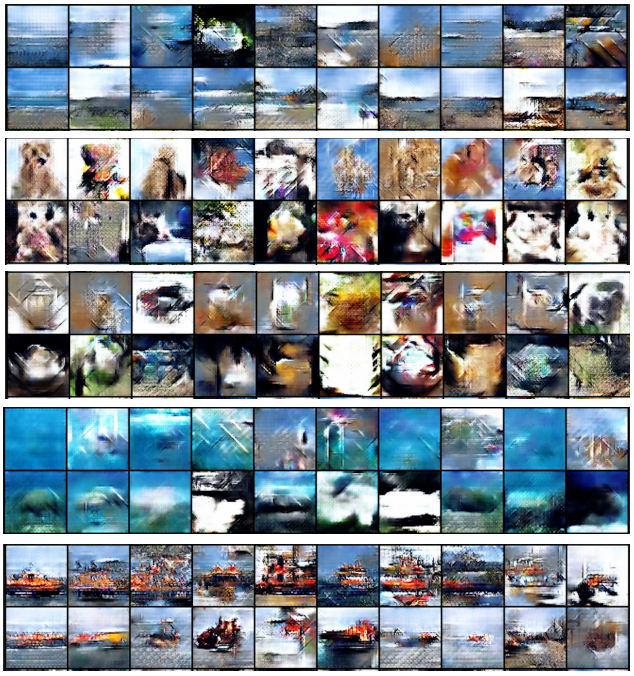

Figure 5: TinyImageNet 2 PPC average representations (column 1) followed by their corresponding variations (colum 2 -10). Same classes as above.

Figure 6: TinyImageNet 1 PPC Average Representations