Few-Shot Dataset Distillation
–Supplementary Materials–

Songhua Liu Xinchao Wang*
National University of Singapore
songhua.liu@u.nus.edu, xinchao@nus.edu.sg

In this document, we provide additional materials on the proposed method for few-shot dataset distillation (DD) that cannot be accommodated in the main paper due to the page limitation. We first provide detailed algorithms for both our method and baselines for comparison. Then, more quantitative and qualitative results are included.

A. More Implementing Details

A.1. Our Method

Pre-training. Here, we provide detailed algorithmic implementation to Alg. 2 of the main paper, the translative pre-training algorithm for distillation space. The key design of the translative pre-training pipeline lies in the separation of two stages: dataset distillation in an arbitrary but fixed neural network and the translation to the desired space. In practice, instead of the version shown in the main paper that nests the two steps in one loop, the two stages are also conducted independently. Results distilled in the predefined network are cached for multiple random subsets so that they can be loaded directly and used repeatedly in the pre-training stage, which improves the training efficiency. The detailed algorithms for the caching stage and the pre-training stage are elaborated in Alg. 1 and Alg. 2 respectively. Readers can refer to Tab. 1 of the main paper for the configurations of hyper-parameters.

Adaptation. Given a pre-trained translator, it requires a small number of adaptation steps in general for the target dataset. The detailed adaptation algorithm is shown in Alg. 3. The hyper-parameters are summarized in Tab. 1 of the main paper.

Translator. The detailed architecture of the translator in this paper is shown in Fig. 1. It adopts an auto-encoder structure. To better inherit the useful information in the input, the network learns the difference between desired output and input. Notably, due to the fully-convolutional structure, the pre-trained translator can also be adapted to datasets with different resolutions beyond the one used during pre-training. Please refer to the following section for details.

Algorithm 1 Caching Stage.

Input: $\mathcal{Z}$: A Large Dataset; θ: An Arbitrary Random Neural Network; T: Number of Update Steps for Synthetic Data; α: Learning Rate for Synthetic Data; N: Number of Cached Subsets; C_{max} and C_{min}: Maximal and Minimal Number of Classes in a Subset; M: Maximal Number of Images in a Subset; I_{max}: Maximal Number of Images in a Synthetic Dataset.
Output: $\mathcal{D}$: A Dataset Used for Pre-training.

$\mathcal{D} \leftarrow \emptyset$;
for N subsets in parallel **do**
  ▷ Data preparation.
  Select an integer C randomly from $[C_{min}, C_{max}]$;
  Select C classes randomly from $\mathcal{Z}$;
  Select an integer I randomly from $[1, \lfloor \frac{I_{max}}{C} \rfloor]$;
  $X_\mathcal{T}, Y_\mathcal{T}, X_\mathcal{S}, Y_\mathcal{S} = [], [], [], []$;
  for $1 \le c \le C$ **do**
    $X_{\mathcal{T},c} \leftarrow \lfloor \frac{M}{C} \rfloor$ random images in class c;
    $Y_{\mathcal{T},c} \leftarrow OneHot([c] \times \lfloor \frac{M}{C} \rfloor)$;
    $X_{\mathcal{S},c} \leftarrow X_{\mathcal{T},c}[:I], Y_{\mathcal{S},c} \leftarrow Y_{\mathcal{T},c}[:I]$;
    $X_\mathcal{T} \leftarrow [X_\mathcal{T}; X_{\mathcal{T},c}], Y_\mathcal{T} \leftarrow [Y_\mathcal{T}; Y_{\mathcal{T},c}]$;
    $X_\mathcal{S} \leftarrow [X_\mathcal{S}; X_{\mathcal{S},c}], Y_\mathcal{S} \leftarrow [Y_\mathcal{S}; Y_{\mathcal{S},c}]$;
  end for
  ▷ Dataset distillation in one network.
  for T steps **do**
    $w^*_{\mathcal{S},\theta} \leftarrow f_\theta(X_\mathcal{S})^\top (f_\theta(X_\mathcal{S}) f_\theta(X_\mathcal{S})^\top)^{-1} Y_\mathcal{S}$;
    $\mathcal{L} = \|f_\theta(DSA(X_\mathcal{T})) w^*_{\mathcal{S},\theta} - Y_\mathcal{T}\|^2$;
    $X_\mathcal{S} \leftarrow X_\mathcal{S} - \alpha \nabla_{X_\mathcal{S}} \mathcal{L}$;
    $Y_\mathcal{S} \leftarrow Y_\mathcal{S} - \alpha \nabla_{Y_\mathcal{S}} \mathcal{L}$;
  end for
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(Idx(X_\mathcal{T}), X_\mathcal{S}, Y_\mathcal{S})\}$;
end for

A.2. Baselines

Benchmark. The baseline algorithm, used in Fig. 1 and Tab. 2 in the main paper, and Tab. 1 here, is shown in Alg. 4.

*Corresponding Author.

Algorithm 2 Pre-training Stage.

Input: $\mathcal{Z}$: A Large Dataset; Θ: Distribution for Initializing Neural Networks; η: Learning Rate for Translator; $\mathcal{D}$: Dataset Cached by Alg. 1; B: Batch Size.
Output: ω: A Pre-trained Translator.

Initialize ω randomly;
repeat
  for B samples in parallel **do**
    $(Idx, X_{\mathcal{S}'}, Y_{\mathcal{S}}) \leftarrow$ a random sample from $\mathcal{D}$;
    Fetch $X_{\mathcal{T}}, Y_{\mathcal{T}}$ from $\mathcal{Z}$ with Idx;
    $X_{\mathcal{S}} \leftarrow G_\omega(X_{\mathcal{S}'})$;
    Randomly sample $\theta' \sim \Theta$;
    $w^*_{\mathcal{S},\theta'} \leftarrow f_{\theta'}(X_{\mathcal{S}})^\top (f_{\theta'}(X_{\mathcal{S}}) f_{\theta'}(X_{\mathcal{S}})^\top)^{-1} Y_{\mathcal{S}}$;
    $\nabla^i_\omega \leftarrow \nabla_\omega \| f_{\theta'}(X_{\mathcal{T}}) w^*_{\mathcal{S},\theta'} - Y_{\mathcal{T}} \|^2$;
  end for
  $\omega \leftarrow \omega - \eta \frac{1}{B} \sum_{i=1}^{B} \nabla^i_\omega$;
until Converge

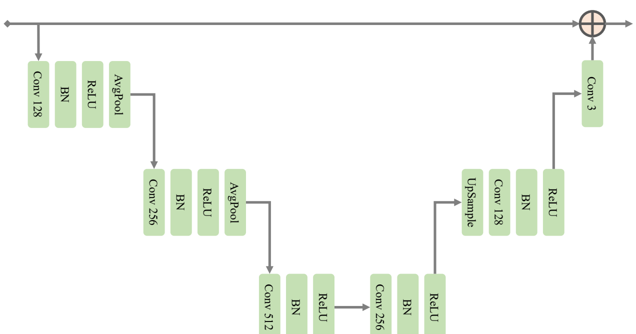


Figure 1. Architecture of our translator.

In experiments, we maintain the same total number of update steps for synthetic data. When the number of networks R is 1, it corresponds to the setting of *1 Net* in Tab. 1. And when $R = T$, it is the setting of *Baseline*.

Relationships with state of the arts. The algorithm shown in Alg. 4 is based on neural feature regression (NFR) [6]. It first uses a neural network as a feature extractor to map the original input to a feature space and then performs kernel ridge regression. On the one hand, it behaves essentially the same as RFAD [5] and the only difference is that it uses a new random neural network in each step of updating synthetic data. On the other hand, compared with FRePo, it does not update networks via current synthetic data or store the networks in a pool. That is why FRePo can further improve performance in many cases. However, these related works have no constraint on the total number of networks, which negatively affects the efficiency of dataset distillation, as illustrated in Fig. 1 of the main paper. We are thus motivated by this observation and dedicated to studying the problem of few-shot dataset distillation which only adopts a limited number of networks.

Algorithm 3 Adaptation Stage.

Input: $(X_{\mathcal{T}}, Y_{\mathcal{T}})$: A Target Dataset to be Distilled; Θ: Distribution for Initializing Neural Networks; θ: The Random Neural Network Used in the Cachine Stage; η: Learning Rate for Translator; $\mathcal{I}$: A List of Numbers of Images per Class for Adaptation; T: Number of Update Steps for Synthetic Data; α: Learning Rate for Synthetic Data; ω_0: A Pre-trained Translator; S: Number of Adaptation Steps for Translator; B: Batch Size.
Output: ω: An Adapted Translator.

$\mathcal{D} \leftarrow \emptyset$;
for I in $\mathcal{I}$ in parallel **do**
  ▷ Synthetic data initialization.
  $X_{\mathcal{S}}, Y_{\mathcal{S}} = [], []$;
  for $1 \le c \le C$ **do**
    $X_{\mathcal{S},c} \leftarrow I$ random images in class c;
    $Y_{\mathcal{S},c} \leftarrow OneHot([c] \times I)$;
    $X_{\mathcal{S}} \leftarrow [X_{\mathcal{S}}; X_{\mathcal{S},c}], Y_{\mathcal{S}} \leftarrow [Y_{\mathcal{S}}; Y_{\mathcal{S},c}]$;
  end for
  ▷ Dataset distillation in one network.
  $X^\theta_{\mathcal{T}} \leftarrow f_\theta(X_{\mathcal{T}})$;
  for T steps **do**
    $w^*_{\mathcal{S},\theta} \leftarrow f_\theta(X_{\mathcal{S}})^\top (f_\theta(X_{\mathcal{S}}) f_\theta(X_{\mathcal{S}})^\top)^{-1} Y_{\mathcal{S}}$;
    $X_{\mathcal{S}} \leftarrow X_{\mathcal{S}} - \alpha \nabla_{X_{\mathcal{S}}} \| X^\theta_{\mathcal{T}} w^*_{\mathcal{S},\theta} - Y_{\mathcal{T}} \|^2$;
    $Y_{\mathcal{S}} \leftarrow Y_{\mathcal{S}} - \alpha \nabla_{Y_{\mathcal{S}}} \| X^\theta_{\mathcal{T}} w^*_{\mathcal{S},\theta} - Y_{\mathcal{T}} \|^2$;
  end for
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(X_{\mathcal{S}}, Y_{\mathcal{S}})\}$;
end for
▷ Adapt the translator to the desired space.
$\omega \leftarrow \omega_0$;
for S steps **do**
  for B samples in parallel **do**
    $(X_{\mathcal{S}'}, Y_{\mathcal{S}}) \leftarrow$ a random sample from $\mathcal{D}$;
    $X_{\mathcal{S}} \leftarrow G_\omega(X_{\mathcal{S}'})$;
    Randomly sample $\theta' \sim \Theta$;
    $w^*_{\mathcal{S},\theta'} \leftarrow f_{\theta'}(X_{\mathcal{S}})^\top (f_{\theta'}(X_{\mathcal{S}}) f_{\theta'}(X_{\mathcal{S}})^\top)^{-1} Y_{\mathcal{S}}$;
    $\nabla^i_\omega \leftarrow \nabla_\omega \| f_{\theta'}(X_{\mathcal{T}}) w^*_{\mathcal{S},\theta'} - Y_{\mathcal{T}} \|^2$;
  end for
  $\omega \leftarrow \omega - \eta \frac{1}{B} \sum_{i=1}^{B} \nabla^i_\omega$;
end for

B. More Experimental Results

Results on larger datasets. Although trained with only 100 classes and 32×32 resolution at most during pre-training, we demonstrate that it is also feasible for the pre-trained translator to be adapted to target datasets with more classes and higher resolutions. Following the settings in the main paper, Tab. 1 shows experimental results on ImageNet1k [2] with 1k classes under 32×32 resolution and ImageNette [3] with 10 classes under 128×128 resolution. For ImageNet1k, although the performance is bottle-necked

Dataset	IPC	Metric	Baseline	1 Net	Bi-Level	w/o Pre-Train	w AE	w/o Ada	w Ada
ImageNet1k (32×32)	1	Acc. (%)	$6.9_{\pm 0.1}$	$6.4_{\pm 0.1}$	$4.4_{\pm 0.1}$	$6.7_{\pm 0.1}$	$6.8_{\pm 0.1}$	$7.0_{\pm 0.1}$	$7.2_{\pm 0.1}$
		Time (sec.)	5393.1	788.3	789.8	2158.0	2158.0	789.4	$2158.0_{\times 2.5}$
	2	Acc. (%)	$7.7_{\pm 0.1}$	$7.1_{\pm 0.1}$	$5.8_{\pm 0.1}$	$7.0_{\pm 0.1}$	$7.1_{\pm 0.1}$	$7.8_{\pm 0.1}$	$8.3_{\pm 0.1}$
		Time (sec.)	6134.2	1529.3	1530.9	3205.6	3205.6	1530.6	$3205.6_{\times 1.9}$
ImageNette (128×128)	1	Acc. (%)	$33.8_{\pm 1.0}$	$27.0_{\pm 0.7}$	$25.6_{\pm 0.3}$	$26.7_{\pm 1.3}$	$34.3_{\pm 0.3}$	$25.4_{\pm 1.6}$	$41.4_{\pm 0.8}$
		Time (sec.)	745.6	676.9	37.4	229.7	229.7	38.0	$229.7_{\times 3.2}$
	10	Acc. (%)	$57.5_{\pm 0.3}$	$48.4_{\pm 1.4}$	$39.3_{\pm 0.6}$	$50.3_{\pm 0.1}$	$52.1_{\pm 0.8}$	$42.0_{\pm 0.2}$	$59.1_{\pm 0.6}$
		Time (sec.)	1010.9	942.2	80.5	275.0	275.0	81.2	$275.0_{\times 3.7}$

Table 1. Results of the proposed method in more challenging datasets with more classes and higher resolutions. The black subscript indicates the standard deviation over multiple evaluations. The red subscript denotes the times of acceleration compared with *Baseline*.

Algorithm 4 Baseline for Comparison.

Input: $(X_\mathcal{T}, Y_\mathcal{T})$: A Target Dataset to be Distilled; Θ: Distribution for Initializing Neural Networks; I: Number of Images per Class; R: Number of Networks; T: Number of Update Steps for Synthetic Data; α: Learning Rate for Synthetic Data.
Output: $(X_\mathcal{S}, Y_\mathcal{S})$: A Distilled Dataset.

▷ Synthetic data initialization.
$X_\mathcal{S}, Y_\mathcal{S} = [], []$;
for $1 \leq c \leq C$ **do**
    $X_{\mathcal{S},c} \leftarrow I$ random images in class c;
    $Y_{\mathcal{S},c} \leftarrow OneHot([c] \times I)$;
    $X_\mathcal{S} \leftarrow [X_\mathcal{S}; X_{\mathcal{S},c}], Y_\mathcal{S} \leftarrow [Y_\mathcal{S}; Y_{\mathcal{S},c}]$;
end for
▷ Dataset distillation in R networks.
for R networks **do**
    Randomly sample $\theta \sim \Theta$;
    $X_\mathcal{T}^\theta \leftarrow f_\theta(X_\mathcal{T})$;
    for $\lfloor \frac{T}{R} \rfloor$ steps **do**
        $w_{\mathcal{S},\theta}^* \leftarrow f_\theta(X_\mathcal{S})^\top (f_\theta(X_\mathcal{S}) f_\theta(X_\mathcal{S})^\top)^{-1} Y_\mathcal{S}$;
        $\mathcal{L} = \|X_\mathcal{T}^\theta w_{\mathcal{S},\theta}^* - Y_\mathcal{T}\|^2$;
        $X_\mathcal{S} \leftarrow X_\mathcal{S} - \alpha \nabla_{X_\mathcal{S}} \mathcal{L}$;
        $Y_\mathcal{S} \leftarrow Y_\mathcal{S} - \alpha \nabla_{Y_\mathcal{S}} \mathcal{L}$;
    end for
end for

by the challenge of scalability in DD [1], the improvement over baseline methods is consistent and it achieves $\sim 2\times$ acceleration compared with *Baseline*. For ImageNette, we first conducted distillation under the 32×32 resolution in the pre-defined network and then up-sample the distilled results to 128×128 resolution with bi-linear interpolation as input to the translator. Due to the discrepancy in resolutions, the performance of results after translation is worse than that of *1 Net* in this case, which is distilled in a network directly under 128×128 resolution. However, after a small number of adaptation steps, with $3 \sim 4\times$ acceleration, it can even surpass *Baseline* which is distilled by enormous networks under the large resolution.

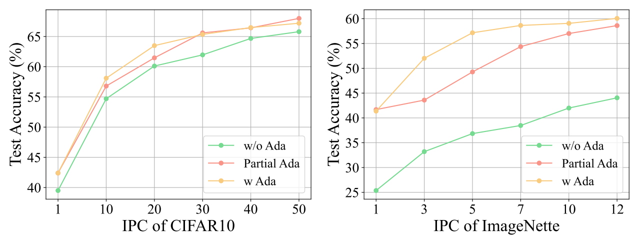


Figure 2. It is feasible for translators after being adapted on some IPCs, *e.g.*, 1 IPC and 50 IPC in CIFAR10 (left), and 1 IPC and 10 IPC in ImageNette (right), to be generalized to the unseen IPCs during adaptation.

More results on cross-IPC generalization. Here, we provide more results on cross-IPC generalization for an adapted translator on some seen IPCs. As shown in the red curve of Fig. 2(left), on CIFAR10 [4], we adapt the translator on 1 and 50 IPCs and report the performance for a variety of IPCs including both seen and unseen ones. As references, we also report the results without adapting the translator and adapting the translator on the specific IPC only in the green and yellow curves respectively. We can find that the performance on unseen IPCs is close to or sometimes even slightly better than that by adapting specifically. Similarly, in Fig. 2(right), we conduct experiments in the same way on ImageNette. For the red curve, the translator is adapted on 1 and 12 IPCs. We observe that the generalization is satisfactory and can approach the yellow curve when IPC is large. In all cases, it works significantly better than no adaptation, which indicates that the adaptation stage helps the translator encode useful knowledge of the whole target dataset, not limited to the knowledge required by the seen IPCs.

More qualitative visualization. We provide more qualitative visualizations on CIFAR10 and ImageNette datasets in Fig. 3 and Fig. 4 respectively, including results distilled by the pre-defined network, results produced by the pre-trained translator without adaptation, and results produced by the translator after adaptation as seen and unseen IPCs.

The observations are consistent with those in the main paper: the pre-trained translator mainly modifies global styles and colors while the adapted translator adds more local details, and the learned textures in the translator are also transferable to unseen IPCs.

References

[1] Justin Cui, Ruochen Wang, Si Si, and Cho-Jui Hsieh. DC-BENCH: Dataset condensation benchmark. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 3

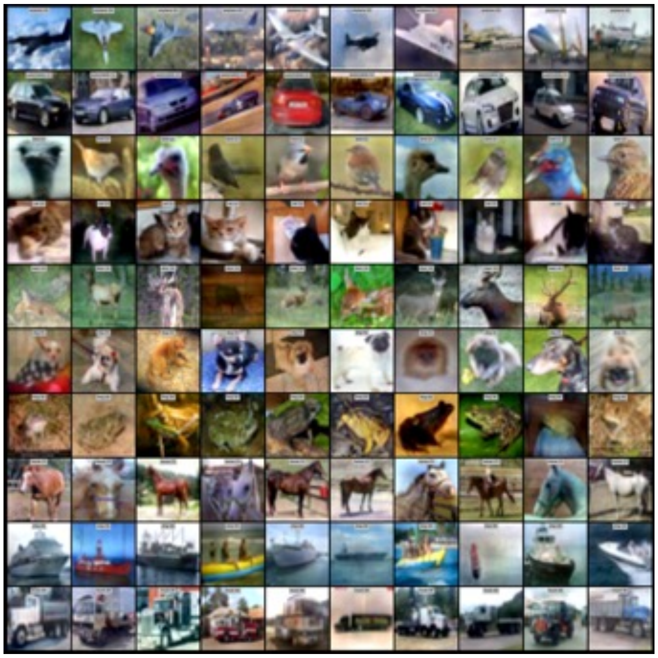
[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 2

[3] Fastai. Fastai/imagenette: A smaller subset of 10 easily classified classes from imagenet, and a little more french. 2

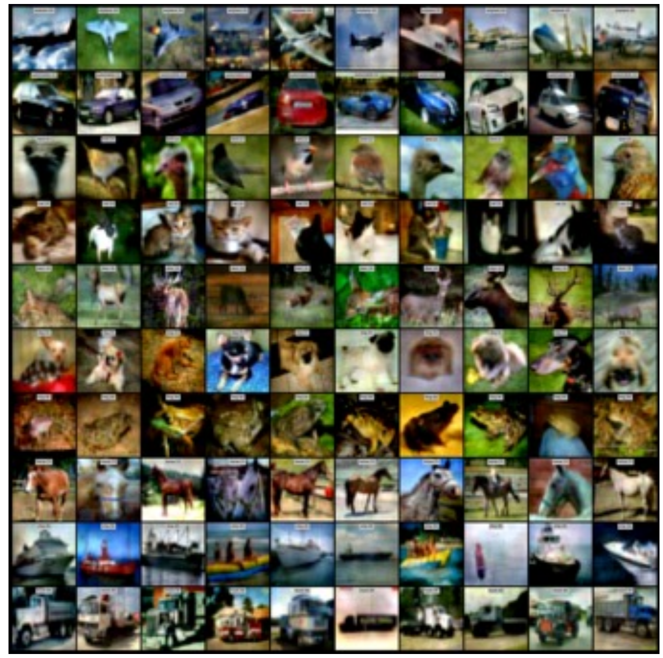
[4] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 3

[5] Noel Loo, Ramin Hasani, Alexander Amini, and Daniela Rus. Efficient dataset distillation using random feature approximation. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 2

[6] Yongchao Zhou, Ehsan Nezhadarya, and Jimmy Ba. Dataset distillation using neural feature regression. *arXiv preprint arXiv:2206.00719*, 2022. 2

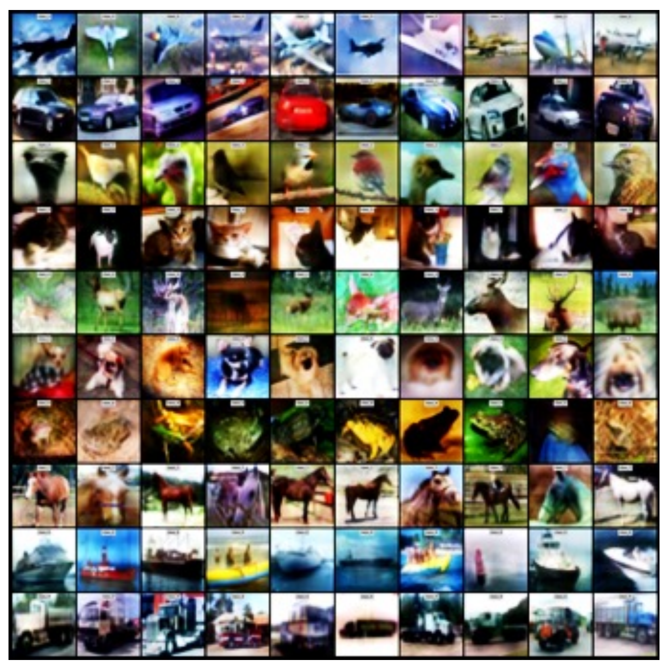


An Arbitrary Net
Acc. 50.5%



Translated Results w/o Ada
Acc. 54.7%

Translated Results w Ada
Acc. 59.2%



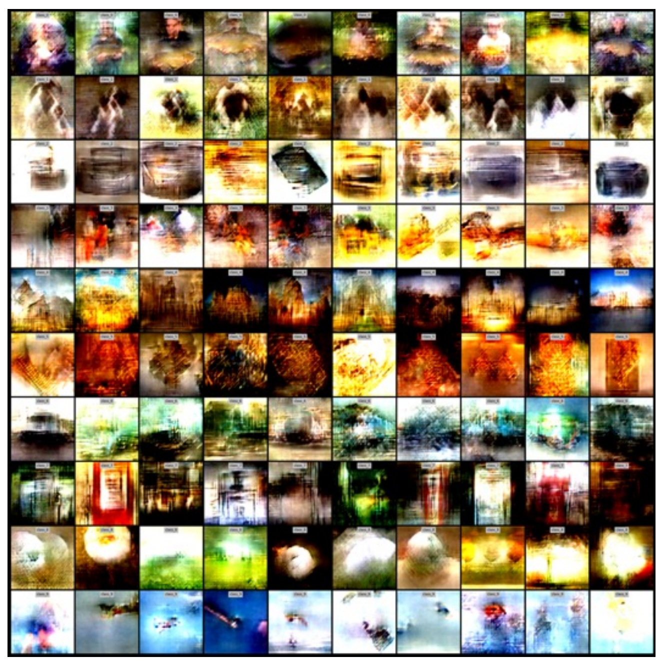
Translated Results w Ada (Generalized)
Acc. 56.8%

Figure 3. Qualitative analysis of results produced by our method for IPC 10 on CIFAR10.

An Arbitrary Net
Acc. 40.1%

Translated Results w/o Ada
Acc. 42.0%

Translated Results w Ada
Acc. 59.1%



Translated Results w Ada (Generalized)
Acc. 58.6%

Figure 4. Qualitative analysis of results produced by our method for IPC 10 on ImageNette.