Supplementary Material for Label-Augmented Dataset Distillation

A. Sub-Sampling Hyperparameter N and R

We perform the study on `LADD-GLaD(MTT)` using the ImageNette dataset at 5 IPC. It aims to determine the optimal size (R) and the number (N) of image sub-samplings. We test four different sizes R and quantities N, validating `LADD-GLaD(MTT)` with 5-CAE. Tab. S1 shows that an increase in N correlates with improved overall accuracy. This is expected, as a higher number of soft labels in a dense label encompasses more information. However, increasing N also results in greater memory inefficiency. For instance, comparing $N = 5$ with $N = 7$, the performance gain is a mere 1.1%, but the overhead rises by 94%. Therefore, balancing the performance-efficiency trade-off is crucial. Hence, we select N = 5 for our model, considering both performance and efficiency.

R represents the size of the sub-image. If R is too small, vital objects representing the target class may be absent in most sub-images. This results in performance degradation due to information loss. Conversely, if R is too large, label augmentation efficiency drops because of redundant information in each sub-image. Our observations indicate that R = 62.5% yields the most accurate results. Therefore, we choose R = 62.5% for our model.

B. Fair Comparison Settings for RDED

RDED [32] introduces an efficient approach for distilling large-scale datasets. It achieves a remarkable 42% top-1 validation accuracy with ResNet-18 [14] on the ImageNet-1K dataset [5]. RDED first generates diverse and realistic data through an optimization-free algorithm backed by ν-information theory [37], which is equivalent to the distillation step. In the deployment stage, the method augments the distilled images and computes the corresponding soft labels from the teacher model. Then, it trains the test model using the augmented images and soft labels.

Despite the remarkable performance of RDED, we identified that the method does not align with the purpose of dataset distillation. Dataset distillation aims to distill the knowledge from a given dataset into a terse data summary [25]. However, RDED uses a teacher model for soft label prediction of augmented images in the deployment stage. Specifically, RDED generates an unlimited number of images and labels via image augmentation that fully exploits the teacher model's knowledge. Thus, RDED aligns more with knowledge distillation rather than dataset distillation in the deployment stage.

Therefore, we assess the performance of RDED while ensuring it complies with the purpose of dataset distillation by eliminating the labeling process that relies on the teacher model during the deployment stage.

C. Performance Degradation in TESLA

TESLA consistently depicts low accuracy in both Tab. 3 and Tab. 5. Although we used the official code and tuned the hyperparameters, we could not successfully train TESLA. Thus, we investigated the reason for this result.

TESLA introduces a method to reduce the high GPU memory issue arising from the bi-loop nested optimization problem in MTT [2]. Through a formulaic improvement, it reduces unnecessary computation graphs while achieving the same objective. Specifically, TESLA claims that the gradient for each batch image only depends on the iteration involving the images. Thereby, the model can remove the computation graph after computing the gradient for each image.

We found an oversight in TESLA's formulation: it does not consider the inner-loop model parameters as dependent variables of the image from different iterations. This means TESLA simplifies the objective of MTT by ignoring the feedback from different training iterations to reduce computations. This explains why TESLA is incapable of achieving a similar high accuracy to MTT in Tab. 3. Detailed proof can be found in Sec. E.

D. Experiments on Small Dataset: CIFAR-10

We evaluate LADD on the small-sized image dataset, CIFAR-10 [18]. We adopt the same hyperparameters (i.e., R and N) defined in Sec. 4.1, with an image size of 32×32. We apply LADD to the distilled dataset from DATM [13], which is the current state-of-the-art method for small-sized datasets. To account for the small-sized image, we use a 3-layer convolutional network (ConvNetD3) for both the distillation and deployment stages. Tab. S2 reports the deployment stage performance at 1 and 10 IPC. The results demonstrate that our method improves `DATM` and achieves the highest performance compared to other methods. Therefore, we conclude that LADD also boosts performance in small-sized datasets.

E. Mathematical Analysis on TESLA

In this section, we derive the mathematical differences between TESLA and MTT to explain the performance difference in Tab. 3 and Tab. 5.

R (pixels) \ N	3	5	7	9	Avg.
50.0% (64)	47.0±1.0	53.4±0.9	55.0±0.7	56.3±0.8	52.9±0.9
62.5% (80)	48.8±1.3	53.9±0.9	55.2±1.3	54.2±1.0	**53.0±1.1**
75.0% (96)	48.9±0.9	52.1±1.5	51.4±1.5	52.0±1.2	51.1±1.3
88.5% (128)	48.4±1.8	50.5±1.2	50.9±1.1	51.6±1.0	50.4±1.3
Avg.	48.3±1.3	52.5±1.1	53.1±1.2	**53.5±1.0**	
Overhead (%)	7.5	20.7	40.2	66.3	

Table S1. **Ablation Study on Sub-Image Size $R(\%)$ and the Number of Axis Split N.** Each accuracy indicates `LADD-GLaD(MTT)` results on ImageNette at 5 IPC. Underline depicts chosen parameter for other experiments.

IPC	1	10
Random	15.4±0.3	31.0±0.5
DC [43]	28.3±0.5	44.9±0.5
DM [42]	26.0±0.8	48.9±0.6
DSA [41]	28.8±0.7	52.1±0.5
CAFE [35]	30.3±1.1	46.3±0.6
FRePo [45]	46.8±0.7	65.5±0.4
MTT [2]	46.2±0.8	65.4±0.7
FTD [8]	46.0±0.4	65.3±0.4
DATM [13]	46.9±0.5	66.8±0.2
DATM†	47.6±0.3	65.5±0.5
LADD-DATM (ours)	**48.6±0.7**	**67.2±0.4**

Table S2. **Performance on CIFAR-10 Dataset.** DATM† indicates the performance of the reproduced image which is used in LADD-DATM.

E.1. Objective Function of MTT

We briefly review the mathematical expression of MTT to understand the oversight in TESLA. MTT defines the $\mathcal{L}_{sim}$ through the parameter distance:

$$\mathcal{L}_{sim} = \|\hat{\theta}_{t+T} - \theta^*_{t+M}\|_2^2 / \|\theta^*_t - \theta^*_{t+M}\|_2^2, \tag{S1}$$

where θ^*_t and θ^*_{t+M} are the model parameters trained on source dataset D_s for t and $t+M$ steps, respectively. Starting from the θ^*_t, MTT trains the model for $i \in [0, T)$ steps on the distilled dataset D following the SGD rule and cross-entropy loss. The trained parameter is denoted as:

$$\hat{\theta}_{t+i+1} = \hat{\theta}_{t+i} - \beta \nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i), \tag{S2}$$

where $\tilde{X}_i$ is sub-batch of D and $\ell(\hat{\theta}_{t+i}; \tilde{X}_i)$ is the cross-entropy loss. β indicates the learning rate for the inner-loop. We can expand $\hat{\theta}_{t+T}$ as:

$$\begin{aligned}\hat{\theta}_{t+T} = \theta^*_t &- \beta \nabla_\theta \ell(\theta^*_t; \tilde{X}_0) - \beta \nabla_\theta \ell(\hat{\theta}_{t+1}; \tilde{X}_1) - \dots \\ &- \beta \nabla_\theta \ell(\hat{\theta}_{t+T-1}; \tilde{X}_{T-1}).\end{aligned} \tag{S3}$$

Eqn. S1 is expanded as:

$$\|\hat{\theta}_{t+T} - \theta^*_{t+M}\|_2^2 = \|\theta^*_t - \beta \sum_{i=0}^{T-1} \nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i) - \theta^*_{t+M}\|_2^2. \tag{S4}$$

We omit the constant denominator of $\mathcal{L}_{sim}$ for brevity. We then further expand the Eqn. S4 as:

$$\begin{aligned}\|\hat{\theta}_{t+T} - \theta^*_{t+M}\|_2^2 &= 2\beta(\theta^*_{t+M} - \theta^*_t)^T \Big(\sum_{i=0}^{T-1} \nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i)\Big) \\ &+ \beta^2 \Big\| \sum_{i=0}^{T-1} \nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i)\Big\|^2 + C,\end{aligned} \tag{S5}$$

where $C = \|\theta^*_t - \theta^*_{t+M}\|_2^2$ is a constant and a negligible term in the gradient computation. For convenience, we represent $G = \sum_{i=0}^{T-1} \nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i)$.

E.2. Cause of Performance Degradation

TESLA claims two points. First, the elements of the first term G only involve the gradients in a single batch and thus can be pre-computed. Second, the computation graph of $\nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i)$ is not required in the derivative of any other batch $\tilde{X}_{j \neq i}$. Based on these points, TESLA computes the gradient for each batch $\tilde{X}_i$ as:

$$\begin{aligned}\frac{\partial \|\hat{\theta}_{t+T} - \theta^*_{t+M}\|_2^2}{\partial \tilde{X}_i} &= 2\beta(\theta^*_{t+M} - \theta^*_t)^T \frac{\partial}{\partial \tilde{X}_i} \nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i) \\ &+ 2\beta^2 G^T \frac{\partial}{\partial \tilde{X}_i} \nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i).\end{aligned} \tag{S6}$$

Since Eqn. S6 can be computed for each batch, TESLA asserts that the memory requirement can be significantly reduced by not retaining the computation graph for all batches.

Here, we found the missing point in the second claim. The computation graph of $\nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i)$ is required in the derivative of any other batch $\tilde{X}_{j \neq i}$. For example, we can compute the gradient for $\tilde{X}_{T-2}$ from the Eqn. S5:

$$\frac{\partial \|\hat{\theta}_{t+T} - \theta^*_{t+M}\|_2^2}{\partial \tilde{X}_{T-2}} = 2\beta(\theta^*_{t+M} - \theta^*_t)^T \frac{\partial}{\partial \tilde{X}_{T-2}} \Big[\nabla_\theta \ell(\hat{\theta}_{t+T-1}; \tilde{X}_{T-1}) + \nabla_\theta \ell(\hat{\theta}_{t+T-2}; \tilde{X}_{T-2})\Big] + 2\beta^2 G^T \frac{\partial}{\partial \tilde{X}_{T-2}} \Big[\nabla_\theta \ell(\hat{\theta}_{t+T-1}; \tilde{X}_{T-1}) + \nabla_\theta \ell(\hat{\theta}_{t+T-2}; \tilde{X}_{T-2})\Big]. \quad \text{(S7)}$$

We can omit other $\nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i)$ where $i < T-2$ because they are independent of $\tilde{X}_{T-2}$. However, the term $\nabla_\theta \ell(\hat{\theta}_{t+T-1}; \tilde{X}_{T-1})$ cannot be ignored. Following Eqn. S2, $\hat{\theta}_{t+T-1}$ depends on the synthetic image $\tilde{X}_{T-2}$. The derivative for $\hat{\theta}_{t+T-1}$ with respect to the image is:

$$\frac{\partial}{\partial \tilde{X}_{T-2}} \hat{\theta}_{t+T-1} = -\beta \frac{\partial}{\partial \tilde{X}_{T-2}} \nabla_\theta \ell(\hat{\theta}_{t+T-2}; \tilde{X}_{T-2}). \quad \text{(S8)}$$

Then, we can compute the derivative for the term $\nabla_\theta \ell(\hat{\theta}_{t+T-1}; \tilde{X}_{T-1})$:

$$\begin{aligned} &\frac{\partial}{\partial \tilde{X}_{T-2}} \nabla_\theta \ell(\hat{\theta}_{t+T-1}; \tilde{X}_{T-1}) \\ &\quad = \nabla^2_\theta \ell(\hat{\theta}_{t+T-1}; \tilde{X}_{T-1}) \frac{\partial}{\partial \tilde{X}_{T-2}} \hat{\theta}_{t+T-1} \\ &\quad = -\beta \nabla^2_\theta \ell(\hat{\theta}_{t+T-1}; \tilde{X}_{T-1}) \frac{\partial}{\partial \tilde{X}_{T-2}} \nabla_\theta \ell(\hat{\theta}_{t+T-2}; \tilde{X}_{T-2}). \end{aligned} \quad \text{(S9)}$$

Finally, the Eqn. S7 becomes:

$$\frac{\partial \|\hat{\theta}_{t+T} - \theta^*_{t+M}\|_2^2}{\partial \tilde{X}_{T-2}} = A\left(1 - \beta \nabla^2_\theta \ell(\hat{\theta}_{t+T-1}; \tilde{X}_{T-1})\right) \frac{\partial}{\partial \tilde{X}_{T-2}} \nabla_\theta \ell(\hat{\theta}_{t+T-2}; \tilde{X}_{T-2}), \quad \text{(S10)}$$

where $A = 2\beta(\theta^*_{t+M} - \theta^*_t)^T + 2\beta^2 G^T$. It is obvious that the computation graph of $\nabla_\theta \ell(\hat{\theta}_{t+T-1}; \tilde{X}_{T-1})$ is required to compute the gradient for $\tilde{X}_{T-2}$. In general, the correct gradient for each batch $\tilde{X}_i$ is:

$$\frac{\partial \|\hat{\theta}_{t+T} - \theta^*_{t+M}\|_2^2}{\partial \tilde{X}_i} = A \prod_{j=i}^{T-1} \left(1 - \beta \nabla^2_\theta \ell(\hat{\theta}_{t+j}; \tilde{X}_j)\right) \frac{\partial}{\partial \tilde{X}_i} \nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i). \quad \text{(S11)}$$

Due to the product term in Eqn. S11, the computation graphs for other steps are required to compute the gradient of $\tilde{X}_i$.

In conclusion, the assumption in Eqn. S6 of TESLA neglects that the $\tilde{X}_i$ affects the other batch gradients. We also empirically confirm that the gradients for distilled images computed on MTT and TESLA are not identical when all other parameters (such as input distilled images, starting parameters, and learning rates) are equal. We conjecture that the low performance of TESLA is due to this observation.

F. Visualization of Sub-Samples

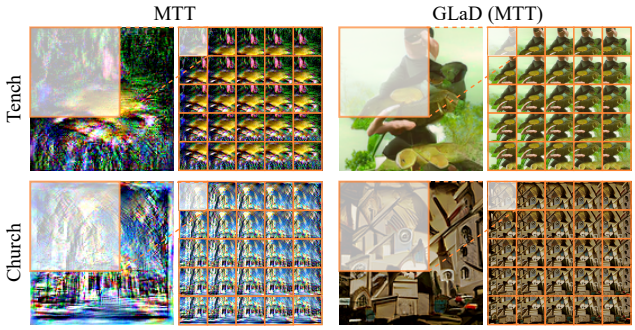

Figure S1. **The result of sub-sampling of MTT and GLaD.** Visualization of the sub-sampling results for the Tench and Church classes from the Imagenette dataset, distilled using the MTT and GLaD methods. For each sample, the image on the left is the original distilled image, and the images on the right are the sub-images after sub-sampling. The original images selected are the first index images from each class.

Fig. S1 demonstrates examples of the results after applying sub-sampling to the distilled dataset. After distilling the Imagenette dataset using the MTT and GLaD methods, the images from the Tench and Church classes were extracted, and these are the original images shown on the left of each sample. Sub-sampling is then performed with hyperparameters set to $N = 5$ and $R = 62.5\%$, starting from the top-left corner of the original image. As a result, 25 sub-images are generated for each original image, which are displayed on the right of each sample.

G. Future Works

We aim to quantize the LADD to reduce storage requirements and improve training efficiency. Furthermore, we plan to explore the application of LADD in tasks that require higher computational costs, such as vision-language models. We will optimize the balance between dense and hard labels through ablation studies or by learning a weight parameter. Additionally, we intend to experiment with alternative static sub-sampling methods to enhance overall performance and scalability across diverse tasks.