A. Theorem 1 and Proof

Readers can refer to the derivation process of DDPM to understand the following derivation. To align with the subscript ordering of weights $\theta_0 \to \theta_M$, we reverse the index order of diffusion states x_i mentioned in standard diffusion method.

Theorem 1. *Given the number of diffusion steps T, an increasing noise schedule $\{\alpha_0, ..., \alpha_T\}$, local target weights $\{\theta_d, ..., \theta_{k*d}\}$, and let the inference process align with the vanilla diffusion algorithm,* i.e.*,*

$$x_{t+1} = \frac{1}{\sqrt{\bar{\alpha}_t}} \left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right). \tag{1}$$

Then, the denoiser ϵ_ϕ can recover the target sequence $\{\theta_d, ..., \theta_{k*d=M}\}$ from standard Gaussian noise x_0 with evenly T/k steps intervals, when ϵ_ϕ is trained by local consistency loss L^{loc} as follows:

$$\begin{aligned} L^{loc} &= \underset{i \in (0,k]}{E} L_i^{loc}, \\ L_i^{loc} &= \underset{t \in [0, i*T/k)}{E} || \sqrt{1-\bar{\alpha}_t^i} \epsilon_\phi(x_t, t) - \sqrt{1-\bar{\alpha}_t} \epsilon ||^2, \\ x_t &= \sqrt{\bar{\alpha}_t^i} \theta_{i*d} + \sqrt{1-\bar{\alpha}_t^i} \epsilon, \end{aligned} \tag{2}$$

where $\bar{\alpha}_t = \prod_{j=t}^{T-1} \alpha_j$, $\bar{\alpha}_t^i = \prod_{j=t}^{i*T/k-1} \alpha_j$, and ϵ denotes standard Gaussian noise.

Proof. First, consider a single local target $\theta_{i*d=i*M/k} = x_{i*T/k}$, its inference chain $\{x_0, ..., x_{i*T/k}\}$ and its increasing schedule $\{\alpha_0, ..., \alpha_{i*T/k}\}$. Let $x_{i*T/k}$ be $x_{T'}$ and diffusion steps be $i * T/K$.

According to vanilla diffusion model, to maximize the likelihood $p(x'_T)$ of observed data x'_T, we need to minimize denoising matching term

$$\underset{t \in [0,T'), q(x_t|x_{T'})}{E} \left[\mathrm{D_{KL}} \left(q(x_{t+1}|x_t, x'_T) \parallel p_\phi(x_{t+1}|x_t) \right) \right]. \tag{3}$$

In the KL divergence bracket, the left term can be expanded by the Bayesian Theorem. The right term is the inference process to be modeled with ϕ, whose expectation is given by Equation 1.

According to Bayes Theorem,

$$q(x_{t+1} \mid x_t, x_{T'}) = \frac{q(x_t \mid x_{t+1}, x_{T'})\, q(x_{t+1} \mid x_{T'})}{q(x_t \mid x_{T'})}. \tag{4}$$

According to the Markov Rule and standard diffusion process,

$$\begin{aligned} q(x_t \mid x_{t+1}, x_{T'}) &= q(x_t \mid x_{t+1}) \\ &\sim \mathcal{N}(x_t; \sqrt{\alpha_t} x_{t+1} + \sqrt{1-\alpha_t} \mathbf{I}). \end{aligned} \tag{5}$$

Recursively using the diffusion process on $\{x_{T'}, ..., x_{t+1}\}$, we have

$$q(x_t|x_{T'}) \sim \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t^{T'}}\, x_{T'}, (1-\bar{\alpha}_t^{T'})\, \mathbf{I}), \tag{6}$$

where $\bar{\alpha}_t^{T'} = \prod_{j=t}^{T'-1} \alpha_j$. Note that the coefficients here differ from those in the vanilla diffusion algorithm.

According to Equation 6, $q(x_{t+1}|x_{T'})$ can be written as

$$q(x_{t+1}|x_{T'}) \sim \mathcal{N}(x_{t+1}; \sqrt{\bar{\alpha}_{t+1}^{T'}}\, x_{T'}, (1-\bar{\alpha}_{t+1}^{T'})\, \mathbf{I}). \tag{7}$$

According to Equation 5 6 7, Equation 4 can be written as

$$
\begin{aligned}
q(x_{t+1} \mid x_t, x_{T'}) &= \frac{q(x_t \mid x_{t+1}, x_{T'})\, q(x_{t+1} \mid x_{T'})}{q(x_t \mid x_{T'})} \\
&\sim \frac{\mathcal{N}\big(x_t; \sqrt{\alpha_t}\, x_{t+1}, (1-\alpha_t)\mathbf{I}\big)\, \mathcal{N}\big(x_{t+1}; \sqrt{\bar{\alpha}_{t+1}^{T'}}\, x_{T'}, (1-\bar{\alpha}_{t+1}^{T'})\mathbf{I}\big)}{\mathcal{N}\big(x_t; \sqrt{\bar{\alpha}_t^{T'}}\, x_{T'}, (1-\bar{\alpha}_t^{T'})\mathbf{I}\big)} \\
&\propto \exp\left\{ -\tfrac{1}{2}\Big(\frac{1}{\frac{(1-\alpha_t)\,(1-\bar{\alpha}_{t+1}^{T'})}{1-\bar{\alpha}_t^{T'}}} \Big) \left[x_{t+1}^2 - 2\, \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t+1}^{T'})\, x_t + \sqrt{\bar{\alpha}_{t+1}^{T'}}\,(1-\alpha_t)\, x_{T'}}{1-\bar{\alpha}_t^{T'}}\, x_{t+1} \right] \right\}.
\end{aligned}
$$

According to the definition of Gaussian distribution, the variance of Equation 4 can be written as

$$
\sigma_q(t) \propto \frac{(1-\alpha_t)(1-\bar{\alpha}_{t+1}^{T'})}{1-\bar{\alpha}_t^{T'}}\mathbf{I},
$$

the expectation can be written as

$$
\mu(x_t, x_{T'}) \propto \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t+1}^{T'})x_t + \sqrt{\bar{\alpha}_{t+1}^{T'}}(1-\alpha_t)x_{T'}}{1-\bar{\alpha}_t^{T'}}. \tag{8}
$$

Recursively apply the reparameterization trick to Equation 5 in an iterative manner, we have:[4]

$$
x_t = \sqrt{\bar{\alpha}_t^{T'}}\, x_{T'} + \sqrt{1-\bar{\alpha}_t^{T'}}\, \epsilon. \tag{9}
$$

Reorganize Equation 9, we have

$$
x_{T'} = \frac{1}{\sqrt{\bar{\alpha}_t^{T'}}}\big(x_t - \sqrt{1-\bar{\alpha}_t^{T'}}\, \epsilon\big).
$$

Substitute $x_{T'}$ into Equation 8, we have

$$
\begin{aligned}
\mu(x_t, x_{T'}) &\propto \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t+1}^{T'})x_t + \sqrt{\bar{\alpha}_{t+1}^{T'}}(1-\alpha_t)\frac{1}{\sqrt{\bar{\alpha}_t^{T'}}}(x_t - \sqrt{1-\bar{\alpha}_t^{T'}}\epsilon)}{1-\bar{\alpha}_t^{T'}} \\
&= \frac{1}{\sqrt{\alpha_t}}x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t^{T'}}\sqrt{\alpha_t}}\epsilon.
\end{aligned} \tag{10}
$$

Returning to Equation 3, minimizing KL divergence is equivalent to minimizing the difference between the expectations of the two terms in the bracket. Equation 1 gives the expectation of the right term, *i.e.*,

$$
\mu_\phi(x_t, t) = \frac{1}{\sqrt{\alpha_t}}x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\,\sqrt{\alpha_t}}\,\epsilon_\phi,
$$

where $\bar{\alpha}_t = \prod_{j=t}^{T-1} \alpha_j$.

Equation 10 is the expectation of the left term. As a result, Equation 3 can be simplified to the form of the left-term' expectation minus the right-term's expectation:

$$
\begin{aligned}
L_{T'} &= \underset{t\in[0,T')}{E} ||\mu(x_t, x_{T'}) - \mu_\phi(x_t, t)||^2 \\
&= \underset{t\in[0,T')}{E} \frac{(1-\alpha_t)^2}{\alpha_t(1-\bar{\alpha}_t^{T'})(1-\bar{\alpha}_t)} ||\sqrt{1-\bar{\alpha}_t^{T'}}\epsilon_\phi - \sqrt{1-\bar{\alpha}_t}\epsilon_{T'}||^2 \\
&\propto \underset{t\in[0,T')}{E} ||\sqrt{1-\bar{\alpha}_t^{T'}}\epsilon_\phi - \sqrt{1-\bar{\alpha}_t}\epsilon||^2.
\end{aligned} \tag{11}
$$

[4]This step is relatively complex; it is recommended to refer to [36].

Note that $\bar{\alpha}_t^{T'} = \prod_{j=t}^{T'-1} \alpha_j = \prod_{j=t}^{i*T/k-1} \alpha_j = \bar{\alpha}_t^i$. So we substitute $T' = i * T/k$ into to Equation 11, it yield:

$$L_i^{loc} = \underset{t\in[0,i*T/k)}{E} ||\sqrt{1-\bar{\alpha}_t^i}\epsilon_\phi(x_t,t) - \sqrt{1-\bar{\alpha}_t}\epsilon||^2.$$

According to $x_{T'} = \theta_{i*d}$ and Equation 9, we have

$$x_t = \sqrt{\bar{\alpha}_t^{T'}}\theta_{i*d} + \sqrt{1-\bar{\alpha}_t^{T'}}\epsilon.$$

The loss function derived above is consistent with the intended formulation L_i^{loc} in Equation 2. This means that L_i^{loc} enables the diffusion model to generate the inference chain from Gaussian noise to $\theta_{i*d=i*M/k}$ in $i * T/k$ steps.

Moreover, since the inference process is fixed, *i.e.*, Equation. 1, $L^{loc} = \underset{i\in(0,k]}{E} L_i^{loc}$ allows any θ_{i*d} and $\theta_{(i+1)*d}$ to share the same inference chain. Their interval is

$$(i+1) * T/k - i * T/k = T/k.$$

Thus, the proof is complete. □

B. Proposition 1 and Proof

Proposition 1. *When $k = 1$, local consistency diffusion is equivalent to the vanilla diffusion algorithm.*

Proof. When $k = 1$, $d = M/k = M$, so $i \in (0, k] \rightarrow i = 1$. At this point, only the optimal weight $\theta_{i \cdot d = M}$ is modeled.

Moreover, for $k = 1$, we have

$$\bar{\alpha}_t^i = \prod_{j=t}^{i*T/k-1} \alpha_j = \prod_{j=t}^{T-1} \alpha_j = \bar{\alpha}_t,$$

hence the two coefficients in Equation 2 are identical and can be simplified, making L_i equivalent to L_i^{loc}. This makes local consistency diffusion be equivalent to the vanilla diffusion algorithm. □

C. Theorem 2 and Proof

Lemma 1. *Assume that the loss function $L^D(\cdot)$ is l-smooth and satisfies μ-strongly convex. Then, the sequence $\{\theta_i\}_{i=0}^M$ generated by the gradient descent update with step size $\frac{1}{l}$ satisfies*

$$\|\theta_M - \theta_*\|^2 \leq \frac{2[L^D(\theta_0) - L^D(\theta_*)]}{\mu}\left(1 - \frac{\mu}{l}\right)^M.$$

Proof. Since $L^D(\theta)$ is l-smooth, for any θ and θ',

$$L^D(\theta') \leq L^D(\theta) + \nabla L^D(\theta)^\top(\theta' - \theta) + \frac{l}{2}\|\theta' - \theta\|^2.$$

Applying this to the gradient descent update $\theta_{k+1} = \theta_k - \frac{1}{l}\nabla L^D(\theta_k)$, we have

$$\begin{aligned} L^D(\theta_{k+1}) &\leq L^D(\theta_k) + \nabla L^D(\theta_k)^\top(\theta_{k+1} - \theta_k) + \frac{l}{2}\|\theta_{k+1} - \theta_k\|^2 \\ &= L^D(\theta_k) - \frac{1}{2l}\|\nabla L^D(\theta_k)\|^2. \end{aligned} \tag{12}$$

Since $L^D(\theta)$ is μ-strongly convex, it satisfies the Polyak–Lojasiewicz condition:

$$\frac{1}{2}\|\nabla L^D(\theta)\|^2 \geq \mu[L^D(\theta) - L^D(\theta_*)].$$

Substituting this inequality into Equation 12, we have

$$L^D(\theta_{k+1}) \leq L^D(\theta_k) - \frac{\mu}{l}(L^D(\theta_k) - L^D(\theta_*)).$$

The above equation can be reorganized to

$$L^D(\theta_{k+1}) - L^D(\theta_*) \leq \left(1 - \frac{\mu}{l}\right)(L^D(\theta_k) - L^D(\theta_*)).$$

Start from $k = 0$, and recursively apply the above equation with M times. It follows that

$$L^D(\theta_M) - L^D(\theta_*) \leq \left(1 - \frac{\mu}{l}\right)^M (L^D(\theta_0) - L^D(\theta_*)).$$

Since $L^D(\theta)$ is μ-strongly convex, it satisfies

$$\|\theta - \theta_*\|^2 \leq \frac{2}{\mu}(L^D(\theta) - L^D(\theta_*)).$$

So we have

$$\begin{aligned} \|\theta_M - \theta_*\|^2 &\leq \frac{2}{\mu}(L^D(\theta_M) - L^D(\theta_*)) \\ &\leq \frac{2(L^D(\theta_0) - L^D(\theta_*))}{\mu}\left(1 - \frac{\mu}{l}\right)^M. \end{aligned}$$

□

Theorem 2. *Assume that the reconstruction error of the generative model is bounded by c, the downstream loss is bounded by ψ, and the loss function is both l-smooth and μ-strongly convex, with the eigenvalues of the Hessian matrix around the optimum θ_* bounded by λ. Then, the cumulative empirical error of the weight generation paradigm can be bounded as follows:*

$$L^D(\hat{\theta}) - L^D(\theta_*) \leq \frac{\lambda}{2}\left[c + \frac{2\psi}{\mu}\left(1 - \frac{\mu}{l}\right)^M\right], \tag{13}$$

where $\hat{\theta}$ is the weight predicted by the generative model.

Proof. Using the Taylor expansion around the optimal point θ_*, we have

$$\begin{aligned} L^D(\hat{\theta}) - L^D(\theta_*) &= \nabla L^D(\theta_*)^T(\hat{\theta} - \theta_*) + \frac{1}{2}(\hat{\theta} - \theta_*)^T \nabla^2 L^D(\xi)(\hat{\theta} - \theta_*) \\ &= \frac{1}{2}(\hat{\theta} - \theta_*)^T \nabla^2 L^D(\xi)(\hat{\theta} - \theta_*). \end{aligned} \tag{14}$$

According to a constraint on the Hessian matrix, we have

$$\frac{1}{2}(\hat{\theta} - \theta_*)^T \nabla^2 L^D(\xi)(\hat{\theta} - \theta_*) \leq \frac{\lambda}{2}||\hat{\theta} - \theta_*||^2. \tag{15}$$

Decomposing $|\hat{\theta} - \theta_*||^2$ into weight preparation error and reconstruction error, we have

$$\begin{aligned} ||\hat{\theta} - \theta_*||^2 &\leq ||\hat{\theta} - \theta_M||^2 + ||\theta_M - \theta_*||^2 \\ &\leq c + \frac{2\left(L^D(\theta_0) - L^D(\theta_*)\right)}{\mu}\left(1 - \frac{\mu}{l}\right)^M \text{ (Using Lemma 1)} \\ &\leq c + \frac{2\psi}{\mu}\left(1 - \frac{\mu}{l}\right)^M. \end{aligned} \tag{16}$$

Substituting Equation 16 and Equation 15 into Equation 14 we obtain

$$L^D(\hat{\theta}) - L^D(\theta_*) \leq \frac{\lambda}{2}\left[c + \frac{2\psi}{\mu}\left(1 - \frac{\mu}{l}\right)^M\right].$$

□

D. Preliminary

D.1. Diffusion Model

The core idea of the diffusion model is to model data through a two-stage process:

- **Diffusion Process**: Starting with data x_T, noise is added at each step to generate x_0, eventually approaching a standard normal distribution.
- **Inference Process**: Starting with noise x_0, a denoising model ϕ generates $x_1, x_2, \ldots, x_T$ step by step.

By precisely modeling the reverse process, the diffusion model ϕ can generate new samples that match the original data distribution. Diffusion models define an increasing schedule $\{\alpha_i\}_{i=0}^{T}$ to control the noise level at each step. In the diffusion process, noise is added at each step t, transforming the data x_{t+1} into x_t

$$q(x_t|x_{t+1}) = \mathcal{N}(x_t; \sqrt{\alpha_t}x_{t+1}, (1-\alpha_t)\mathbf{I}).$$

The direct transition from x_T to x_t can be written as

$$q(x_t|x_T) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_T, (1-\bar{\alpha}_t)\mathbf{I}),$$

where $\bar{\alpha}_t = \prod_{i=t}^{T-1} \alpha_i$ is the cumulative noise schedule, controlling the overall noise level from x_0 to x_t. In the inference process, the denoiser iteratively reconstructs the data by

$$x_{t+1} = \frac{1}{\sqrt{\alpha_t}}x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\epsilon_\phi + \sigma_z\epsilon,$$

We omit the additional $\sigma_z\epsilon$ for stable weight generation. The key point of training a variational model is to maximize the Evidence Lower Bound (ELBO). In the diffusion algorithm, optimizing the ELBO is essentially equivalent to minimizing the denoising match term

$$\underset{t\in[0,T'),q(x_t|x_{T'})}{E}\left[\mathrm{D}_{\mathrm{KL}}\left(q(x_{t+1}|x_t, x'_T) \parallel p_\phi(x_{t+1}|x_t)\right)\right].$$

which is also the objective optimized by our local consistency diffusion.

D.2. REPTILE

REPTILE is a first-order optimization-based meta-learning algorithm that simplifies training while retaining strong adaptability across tasks. It eliminates the need for second-order gradients, making it computationally efficient compared to algorithms like MAML. The training process consists of two loops: the inner-loop and the outer-loop.

In the inner loop, REPTILE performs gradient descent on a sampled task T_i using the task's support set. Starting from the meta-parameters ϕ, the task-specific parameters ϕ_i are updated for K steps using

$$\phi_i^{(t+1)} = \phi_i^{(t)} - \eta\nabla_{\phi_i^{(t)}} L_{T_i}(\phi_i^{(t)}),$$

where η is the inner-loop learning rate and L_{T_i} is the loss for the taskT_i.

In the outer-loop, the meta-parameters ϕ are updated by moving them toward the task-specific parameters ϕ_i obtained from the inner loop. This meta-update is given by

$$\phi \leftarrow \phi + \zeta(\phi_i - \phi),$$

where ζ is the outer-loop learning rate.

By iteratively repeating the inner and outer loops across multiple tasks drawn from the task distribution $\mathcal{T}$, REPTILE optimizes the meta-parameters ϕ to find an initialization that enables fast adaptation to new tasks with minimal gradient steps. Its simplicity lies in avoiding second-order derivatives, while its effectiveness is demonstrated across diverse applications such as few-shot learning and domain generalization.

E. Setup Details

E.1. Dataset

Omniglot. The raw Omniglot dataset contains 1623 handwritten characters from 50 alphabets, each with 20 instances in 28×28 grayscale format. We partition the classes of the training set, evaluation set, and testing set into 800:400:432. We use Omniglot in three scenarios. We used the Omniglot dataset in our preliminary experiments, ablation experiments, and comparative experiments. For the construction of the classification task, we referred to the experimental setup by MAML.

Mini-ImageNet. The raw Mini-ImageNet contains 100 classes, each containing 600 instances in 84×84 grayscale format. We partition classes of training set, evaluation set, and testing set into 64:16:20. The usage of Mini-ImageNet is the same as Omniglot, and we also follow the setup given by MAML.

Tiered-ImageNet. The Tiered-Imagenet dataset is a larger-scale few-shot learning benchmark derived from ImageNet, containing 608 classes grouped into 34 higher-level categories. Each image is in 84×84 resolution with RGB channels. Following the standard protocol (Meta-Baseline), we partition the dataset into 351 classes for training, 97 for validation, and 160 for testing, ensuring no class overlap across phases. We use Tiered-ImageNet in all experimental stages, including ablation and comparative evaluations, to assess the generalization ability of our method on a more diverse and semantically structured dataset.

ImageNet-1K. The raw ImageNet-1K is a benchmark dataset with 1000 classes, 1.2 million training images, and 50000 validation images, typically resized to a resolution of 224×224 pixels. We partitioned the dataset into 20k subsets, each containing 50 classes with 50 images per class. We use this dataset for pre-training and perform transfer learning evaluation on other unseen datasets.

CIFAR-10 CIFAR-100 STL-10 Aircraft Pets. CIFAR-10 and CIFAR-100 are image datasets introduced by Alex Krizhevsky, containing 60000 images resized to 32×32 pixels. CIFAR-10 includes 10 classes, while CIFAR-100 features 100 fine-grained classes. STL-10, derived from ImageNet, consists of 10 classes with 13000 labeled images and 100000 unlabeled images, with a resolution of 96×96 pixels. The Aircraft dataset includes 10000 images across 100 aircraft models, with hierarchical labels for manufacturer, family, and variant. The Pets dataset consists of 7349 images of 37 pet breeds, with annotations for class labels, bounding boxes, and pixel-level segmentation. We use these datasets to evaluate the model's transfer learning capabilities, which means the labels of these datasets are not visible to the model.

DomainNet. DomainNet is a dataset for multi-domain generalization. We use it to evaluate algorithms' ability for few-shot domain generalization. It consists of 345 classes from 6 domains, with a resolution of 224×224 pixels. We use Clipart, Infograph, Painting, Quickdraw, and Real domains for training, while Sketch domains are for testing. The tasks we constructed are 5-way 1-shot and 20-way 5-shot. Note that the testing set shares the same 345 classes as the training set.

GLUE. The GLUE Benchmark (General Language Understanding Evaluation) tests models on 9 diverse NLP tasks, including CoLA for grammatical acceptability, SST-2 for sentiment classification, MRPC for paraphrase detection, STS-B for sentence similarity, QQP for duplicate question detection, MNLI for natural language inference, QNLI for question-answer validation, RTE for entailment classification, and WNLI for pronoun resolution. We use this dataset to test the efficiency of different algorithms for multi-task fine-tuning on LLM models. Specifically, we use five binary classification tasks, *i.e.*, SST-2, QQP, RTE, WNIL, and CoLA for the training of Mc-Di. Then we use the other two tasks, *i.e.*, MRPC and QNIL, to evaluate the performance of different fine-tuning algorithms.

E.2. Model Configuration

For the downstream network θ, we adopt different architectures for different tasks. In tasks on the Omniglot and Mini-ImageNet datasets, we follow the standard setup used by most meta-learning methods, employing 4 convolution blocks with batch normalization and ReLU with a linear probe for classification. For zero-shot tasks on CIFAR-10, CIFAR-100, STL-10, Aircraft, and Pets, we use the commonly adopted ResNet-12 with a linear probe for classification. In multi-domain generalization tasks, we maintain the same setup with Hierarchical Meta-Learning [17]. In the LLM multi-task fine-tuning scenario, the downstream network is the LLM model itself, *i.e.*, RoBERTa-base. For the diffusion model f_ϕ^G, Mc-Di employs the same

U-Net architecture given by Meta-Diff [51]. The task embedding Emb_{T_i} is calculated by a ResNet101 backbone used by ICIS [5]. In the weight preparation stage, the real-world optimizer (Adam) uses a fixed learning rate of 0.005 and an automatic early-stopping strategy [43] to determine the downstream task training epoch M. In the meta-training stage, we set the learning rate η, meta-learning rate ζ and training epochs to 0.005, 0.001, and 6000, respectively. The segment number k is set to 3, and the diffusion step T is set to 20.

F. Additional Experiments

F.1. Sensitivity Study

Compared to existing weight generation methods, the two hyperparameters of Mc-Di, *i.e.*, segment number and diffusion steps, may affect the model's sensitivity. To evaluate the robustness of our approach, we analyze the impact of segment number and diffusion steps on classification accuracy for Mini-ImageNet and Omniglot. As shown in Figure 8 and Figure 9, the accuracy remains stable across different parameter settings, with only minor variations. While increasing segment number and diffusion steps can slightly improve performance, excessive changes do not lead to significant degradation. The consistency across both datasets indicates that Mc-Di is insensitive to these hyperparameters, demonstrating robustness. Note that we do not use the parameter combination that achieves the highest accuracy in this experiment as the default setting. We aim to maintain performance above the state-of-the-art levels while reducing computational costs during inference.

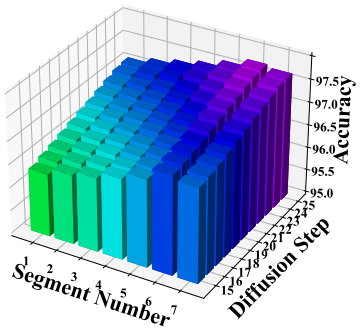

Figure 8. Sensitive study on Omniglot

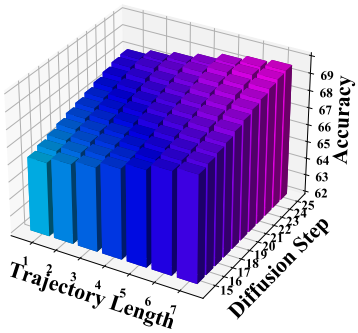

Figure 9. Sensitive study on Mini-Imagenet

F.2. Scalability to Different Architectures

To verify the effectiveness of Mc-Di under different scale network structures, we compare them with their degraded versions, *i.e.*, REPTILE, TW-Di, and Mv-Di. Table 6 and Table 7 show the effectiveness of our method on Swin Transformer, ResNet18, and MobileNetV2 architectures. The results demonstrate that each component of our method remains effective across neural network architectures of different scales.

Table 6. Comparison of accuracy across different network structures on Omniglot 5-way 1-shot tasks.

Method	Swin Transformer	ResNet18	MobileNetV2
REPTILE	98.27	97.11	93.79
TW-Di	97.43	96.60	93.61
Mv-Di	98.93	98.44	94.92
Mc-Di	99.90	98.82	95.84

Table 7. Comparison of accuracy across different network structures on Mini-ImageNet 5-way 1-shot tasks.

Method	Swin Transformer	ResNet18	MobileNetV2
REPTILE	57.27	52.55	40.67
TW-Di	66.26	60.32	53.27
Mv-Di	76.75	70.76	59.33
Mc-Di	82.38	77.85	64.81

F.3. Scalability to Larger Datasets

In the following section, we evaluate the scalability of our method when applied to significantly larger and more diverse datasets. To this end, we conduct a 5-way 1-shot domain generalization experiment on Meta-Dataset [49], a large-scale benchmark containing heterogeneous image domains with substantial variation in style, resolution, and semantic distribution. Unlike standard few-shot benchmarks, Meta-Dataset requires models to generalize to unseen domains whose distributions differ substantially from the training data, making it a strong indicator of cross-domain robustness and real-world applicability.

Table 8 reports the results on the last five unseen datasets of Meta-Dataset. Our method outperforms better than other non-consistency weight generation method across all datasets. Notably, Mc-Di achieves the highest average accuracy while simultaneously maintaining the lowest latency, demonstrating the effectiveness of meta-learning and local consistency mechanisms on larger datasets with larger distribution differences.

Table 8. Comparison of 5-way 1-shot performance on the last 5 unseen datasets of Meta-Dataset. These methods are trained on the first 8 datasets of Meta-Dataset.

Method	**Traffic Sign**	**MSCOCO**	**MNIST**	**CIFAR-100**	**CIFAR-10**	**Avg**	**Latency (ms)**
REPTILE	43.32 ± 0.88	39.62 ± 1.02	68.50 ± 0.88	40.60 ± 0.98	43.73 ± 0.94	47.15	26.8
Meta-Baseline	47.52 ± 0.92	41.20 ± 1.04	68.47 ± 0.85	43.07 ± 0.85	46.02 ± 0.90	49.26	26.2
Meta-Hypernetwork	55.91 ± 0.91	48.47 ± 0.91	80.55 ± 0.91	50.67 ± 0.91	54.14 ± 0.91	57.95	10.3
GHN3	44.22 ± 0.93	47.27 ± 1.14	76.44 ± 0.91	36.38 ± 0.83	42.52 ± 1.06	49.37	20.1
OCD	52.14 ± 0.90	57.74 ± 1.12	75.23 ± 0.92	47.51 ± 0.91	55.54 ± 1.13	57.63	9.7
Meta-Diff	57.19 ± 0.94	60.01 ± 1.04	78.47 ± 0.94	55.29 ± 0.89	61.38 ± 1.01	62.47	10.9
Mv-Di (ours)	58.48 ± 1.02	60.58 ± 1.00	80.82 ± 0.93	55.13 ± 0.93	62.93 ± 1.04	63.59	7.0
Mc-Di (ours)	59.46 ± 0.99	62.70 ± 1.01	83.65 ± 0.94	57.07 ± 0.91	65.11 ± 1.05	**65.60 (+2.01)**	**5.2 (× 1.35)**

F.4. Ablation on the overhead of SAM

To address the concern regarding the the computational cost of SAM, we provide additional evaluations covering both the weight preparation stage and Mc-Di's overall training stage. Tables 9 and Table 10 further validate the impact of SAM on both the weight preparation stage and Mc-Di's overall training stage (following the DomainNet 5-way 1-shot setup in Section 4.2.3).

Weight preparation impact: In Table 9, SAM increases the time required to compute each weight trajectory. However, this overhead is relatively small compared to Mc-Di's overall training cost reported in Table 10.

Overall impact: Moreover, Table 9 shows that SAM effectively constrains the weight update behavior. As guaranteed by Theorem 2, it enables faster convergence (fewer GPU-hours) at the same level of accuracy, as reflected in Table 10. This benefit outweighs the additional cost.

Table 9. Impact of SAM on the weight preparation stage (DomainNet 5-way 1-shot Task).

Metric	**Adam**	**Adam+SAM (Ours)**
Avg GPU-Second / per-trajectory	1.3	1.6
Avg Early-stopping steps / per-trajectory	32	21
Avg Final weight's sharpness	0.3062	0.1794

Table 10. Impact of SAM on overall Mc-Di (DomainNet 5-way 1-shot Task).

Accuracy (%)	**Mc-Di w/o SAM (GPU-Hours)**	**Mc-Di with SAM (GPU-Hours) (Ours)**
20	0.00	0.00
30	0.29	0.25
40	0.56	0.52
50	1.51	1.18
60	3.62	2.65
Coverage	7.14	5.97

F.5. Comprehensive Overhead Analysis Across Architectures

To address the concern regarding memory complexity across architectures, we provide a unified analysis of all architectures used in this work under the same experimental setup (NVIDIA A100, Mini-ImageNet 84×84, batch size = 32).

As shown in Table 11, within our few-shot setting that requires frequent adaptation, the medium-scale architectures used in the main text incur acceptable overhead. For Mc-Di, where GPU memory for weight inference is the main concern, the peak usage is around 3 GB. Moreover, even when applied to larger architectures, the memory overhead of Mc-Di remains within the single-GPU memory capacity of an NVIDIA A100. In future work, we aim to leverage multi-GPU training environments to directly generate much larger neural network architectures, including models on the scale of GPT-3 and GPT-4.

Table 11. Comprehensive memory complexity analysis under unified settings: one NVIDIA A100 GPU, Swin-Transformer architecture, Mini-ImageNet dataset (84×84), and batch size = 32.

Architecture	**Parameter Number (M)**	**Data Load GPU I/O (MB/s)**	**Peak Meta-Training GPU Memory (MB)**	**Peak Weight Inference GPU Memory (MB)**
4-CNN	0.11	12.18	764	174
ResNet-12	1.55	58.36	5887	1221
MobileNetV2	2.23	15.55	7063	1465
ResNet-18	11.17	127.44	15815	3275
Swin-Trans	87.77	151.96	34293	7462

F.6. Comparison with NiNo and WNN

To clarify how our approach differs from the latest developments in weight-prediction methods such as NiNo [31] and WNN [24], we summarize the key conceptual distinctions as follows. NiNo and WNN focus on short-horizon weight evolution by predicting near-future updates from past weights, and both operate jointly with Adam in an iterative optimization loop. Mc-Di, in contrast, samples weights from the full training trajectory and performs gradient-free inference toward the final weight, capturing global rather than local weight dynamics. This introduces the challenge of maintaining optimization consistency across diverse local weights, which is addressed by our theoretical analysis (Theorems 1 and 2). As shown in Table 13, NiNo and WNN exhibit strong performance on large-scale architectures with abundant data, whereas Mc-Di places greater emphasis on computational efficiency through forward-only training.

In contrast, Table 12 highlights that Mc-Di is particularly well suited for rapid adaptation under limited data or resource constraints, such as in few-shot, multi-domain, and transfer learning scenarios. These tasks typically benefit less from short-horizon weight predictors like NiNo and WNN, while Mc-Di is explicitly designed around frequent model updates in such environments. Empirically, Mc-Di outperforms weight-prediction and Adam-based baselines in these adaptive settings and provides significantly lower inference overhead. Collectively, these results indicate that although NiNo and WNN are effective for full-scale supervised training, Mc-Di offers clearer advantages in data-efficient and fast-adaptation learning regimes.

Table 12. Accuracy comparison on ResNet-12 for few-shot and multi-domain tasks.

Method	Mini-ImageNet (1-shot)	Tiered-ImageNet (5-shot)	DomainNet	Latency (ms)
WNN	42.53	62.46	41.75	11.90
REPTILE (Adam FT)	47.07	65.99	48.13	22.10
Mc-Di	**64.97 (+17.90)**	**88.58 (+22.59)**	**69.05 (+20.92)**	**5.06 (×4.37)**

Table 13. Accuracy comparison on ResNet-50 for ImageNet re-training.

Method	Best Accuracy (%)	Reach 70% Accuracy	Epochs
WNN	76.02	70.00	49
Adam Baseline	75.36	70.00	64
Mc-Di	**65.74 (-10.28)**	-	**20 (forward-only)**

F.7. Scalability to Decoder-Only Models

To address the concern regarding the applicability of Mc-Di to decoder-only architectures such as GPT models, we provide additional evaluations beyond the encoder-focused experiments in the main text. Although Section 4.2.4 explores only a single text classification architecture due to space constraints, the results from Section 4.2.1 and Section 4.2.3 indicate that Mc-Di is architecture-agnostic and should extend naturally to decoder-based models.

To further verify this, we conducted experiments on GPT-2 under the same text classification setup described in Section 4.2.4. As shown in Table 14, Mc-Di achieves substantial reductions in fine-tuning latency on the GLUE MRPC and QNLI tasks, while maintaining competitive accuracy. Specifically, Mc-Di reduces the fine-tuning latency by an average factor of 4.1, with only a 0.87% average decrease in accuracy. These results demonstrate that Mc-Di adapts effectively to decoder-only architectures and remains efficient and practical for lightweight LLM adaptation scenarios.

Table 14. Accuracy and fine-tuning latency comparison on MRPC and QNLI using GPT-2 under the same fine-tuning setup.

Method	**MRPC Acc (%)**	**Latency (h)**	**QNLI Acc (%)**	**Latency (h)**
Full Fine-Tuning	82.74	1.17	86.58	2.52
LoRA	81.92	0.65	86.12	1.41
Mv-Di	80.38	0.20	85.05	0.38
Mc-Di	81.69 (-1.05)	0.17 ($\times$3.8)	85.97 (-0.68)	0.32 ($\times$4.4)