

# Learning an Efficient Optimizer via Hybrid-Policy Sub-Trajectory Balance

Yunchuan Guan<sup>a,b</sup>, Yu Liu<sup>a,c,\*</sup>, Ke Zhou<sup>a</sup>, Hui Li<sup>d</sup>, Sen Jia<sup>e</sup>, Zhiqi Shen<sup>b</sup>, Ziyang Wang<sup>f</sup>, Xinglin Zhang<sup>g</sup>, Tao Chen<sup>h</sup>, Jenq-Neng Hwang<sup>e</sup> and Lei Li<sup>e,i,\*</sup>

<sup>a</sup>Huazhong University of Science and Technology, Wuhan, China

<sup>b</sup>Nanyang Technological University, Singapore

<sup>c</sup>National University of Singapore, Singapore

<sup>d</sup>Jinan Inspur Data Technology Co., Ltd, Jinan, China

<sup>e</sup>University of Washington, Seattle, United States

<sup>f</sup>University of Oxford, Oxford, United Kingdom

<sup>g</sup>Shanghai Medical Image Insights Intelligent Technology Co., Shanghai, China

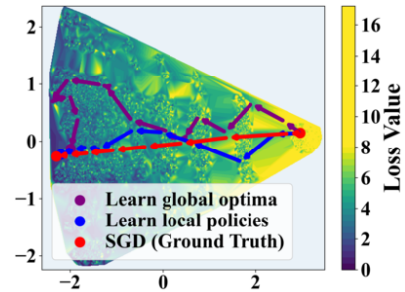
<sup>h</sup>University of Waterloo, Waterloo, Canada

<sup>i</sup>University of Copenhagen, Copenhagen, Denmark

**Abstract.** Recent advances in generative modeling enable neural networks to generate weights without relying on gradient-based optimization. However, current methods are limited by issues of over-coupling and long-horizon. The former tightly binds weight generation with task-specific objectives, thereby limiting the flexibility of the learned optimizer. The latter leads to inefficiency and low accuracy during inference, caused by the lack of local constraints. In this paper, we propose Lo-Hp, a decoupled two-stage weight generation framework that enhances flexibility through learning various optimization policies. It adopts a hybrid-policy sub-trajectory balance objective, which integrates on-policy and off-policy learning to capture local optimization policies. Theoretically, we demonstrate that learning solely local optimization policies can address the long-horizon issue while enhancing the generation of global optimal weights. In addition, we validate Lo-Hp’s superior accuracy and inference efficiency in tasks that require frequent weight updates, such as transfer learning, few-shot learning, domain generalization, and large language model adaptation.

## 1 Introduction

Generative models have rapidly advanced and become central to AI research, driving breakthroughs in areas including vision, audio, language, and structured data [10]. Recent work has extended the generative model to weight generation for neural networks. They predict downstream neural network weights  $\theta$  [20, 31, 15, 16] using a learned forward-only optimizer  $f_\phi^G$ , enabling efficient weight adaptation to downstream tasks without computing gradients. Since this innovation can reduce the cost of weight updates, it shows promise in scenarios that require frequent weight updates, such as transfer learning, few-shot learning, domain generalization, and LLM fine-tuning. However, existing methods are limited by two issues: over-coupling and long horizon.



**Figure 1.** Inference trajectory of generative models in CIFAR-10’s 2D weight-reduced space. Darker regions indicate lower downstream task loss, and the red trajectory represents the ground truth generated by the real-world optimizer SGD.

**Over-Coupling:** Previous studies, such as MetaHyperNetwork [35] and the GHN series [32, 15, 16], adopted an end-to-end approach that directly models downstream task performance. These methods define the optimization objective as

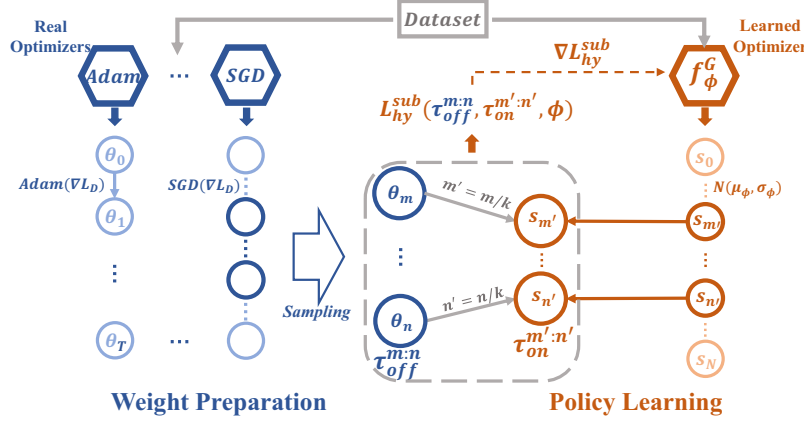
$$\arg \min_{\phi} E_{(x,y) \in \mathcal{D}} [L_D(f_\theta(x; \theta = f_\phi^G(x; \phi)), y)], \quad (1)$$

where  $L_D$  refers to the loss function of downstream tasks and  $\mathcal{D}$  denotes the dataset. Although straightforward, this approach suffers from over-coupling. It binds the objectives of weight generation and downstream task optimization, thus limiting the flexibility for the learned optimizer  $f_\phi^G$ .<sup>1</sup> Specifically, the inference process of  $f_\phi^G$  must be differentiable and can only be unrolled over a short horizon. This prevents the learned  $f_\phi^G$  from capturing more expressive policies over long horizons.

**Long-Horizon:** Recent studies, such as OCD [20], MetaDiff [31], and D2NWG [29], leverage models such as diffusion to model the downstream neural network weights  $\theta$ , enabling more faithful simulation of optimizer behaviors over a long-horizon inference. However, these methods focus on a single optimization policy (e.g.,

\* Co-corresponding authors. Emails: liu\_yu@hust.edu.cn, lilei@di.ku.dk

<sup>1</sup> The terms generative model and learned optimizer are used interchangeably.



**Figure 2.** Overview of Lo-Hp. It consists of two decoupled stages: weight preparation and policy learning. In the weight preparation stage, it utilizes learned optimizers such as Adam, SGD, etc., to update the neural network weights  $\theta$ . Then, it samples and records the offline sub-trajectory  $\tau_{off}^{m:n}$ . In the policy learning stage, the generative model  $f_\phi^G$  adopts a Gaussian policy to generate the online trajectory. A uniform sub-trajectory matching strategy is used to align the online sub-trajectory  $\tau_{on}^{m':n'}$  and offline sub-trajectory  $\tau_{off}^{m:n}$ , and the proposed hybrid-policy sub-trajectory balance is applied to learn local optimization policies.

SGD), thereby failing to explore the flexibility of this paradigm. More importantly, they focus solely on the global optima  $\theta_*$ , neglecting the local policy details in the sub-trajectories  $\theta_m \rightarrow \theta_n$ . As shown by the purple trajectory in Figure 1, such an unconstrained learning process leads to low efficiency and low accuracy in the inference trajectories over a long horizon.

In this paper, we rethink weight generation as an optimization policy learning problem. Our objective is to develop methodologies for generating global optimal weights as well as modeling local optimization policies. We propose to **Learn an efficient Optimizer via Hybrid-Policy Sub-Trajectory Balance**, i.e., Lo-Hp. (1) To address the limited flexibility caused by over-coupling, Lo-Hp adopts a decoupled two-stage learning process that enables the learning of diverse local optimization policies. Formally, the decoupled learning framework can be defined as

$$\begin{aligned} \{\theta_t\}_0^T &= \arg \min_{\theta} E_{(x,y) \in \mathcal{D}} [L_D(x, y, f_\theta)] \\ \phi &= \arg \min_{\phi} E_{x, m, n} [\mathcal{L}_{hy}^{sub}(x, \{\theta_t\}_m^n, f_\phi^G)]. \end{aligned} \quad (2)$$

As shown in Figure 2, the weight preparation stage uses multiple optimizers (e.g., SGD and Adam) to construct diverse offline trajectories, enhancing the policy flexibility of the learned  $f_\phi^G$ . In the policy learning stage, the generative model  $f_\phi^G$  adopts Hybrid-Policy Sub-Trajectory Balance to constrain the inference trajectory at the local level. (2) Hybrid-Policy Sub-Trajectory Balance is designed to address the inefficiency in inference caused by the long horizon. It is a hybrid learning strategy that lies between on-policy and off-policy learning [2]. It introduces supervision signals from the offline sub-trajectory, i.e.,  $\{\theta_t\}_m^n$ , into the learning process of the online sub-trajectory, i.e.,  $\{s_i\}_{m'}^{n'}$ ,<sup>2</sup> thereby enabling  $f_\phi^G$  to acquire optimization policies at the local level. As shown by the blue trajectory in Figure 1, this approach improves both the efficiency and accuracy of the weight generation process. (3) We theoretically show that our method, though focusing solely on local optimization policy learning, remains capable of promoting the generation of global optimal weights. In addition, we analyze the convergence of our method and introduce Sharpness Aware Minimization (SAM) [9]. It improves the

overall convergence efficiency of the framework. Extensive experiments demonstrate Lo-Hp’s superior accuracy and inference efficiency in tasks that require frequent weight updates. Our contributions can be summarized as follows:

- We propose a decoupled framework Lo-Hp, which consists of weight preparation and policy learning stages. It enhances the flexibility of the learned optimizer by learning various optimization policies.
- We propose a hybrid-policy sub-trajectory balance, which captures local policy details while simultaneously facilitating the generation of globally optimal weights.
- We analyze the convergence of the decoupled weight generation framework and introduce SAM to enhance convergence efficiency.

## 2 Method

The proposed Lo-Hp decouples the learning process into two stages: weight preparation and policy learning. This framework enhances the flexibility of the learned optimizer. During the weight preparation stage, Lo-Hp constructs offline trajectories using various optimization policies. During the policy learning stage, the generative model  $f_\phi^G$  adopts Hybrid-Policy Sub-Trajectory Balance to capture local optimization policies.

### 2.1 Weight Preparation

As shown on the left side of Figure 2, the target of the weight preparation stage is to collect optimization trajectories constructed by different optimization policies. Formally, this process can be defined as

$$\tau_{of}^{0:T} = \{\theta_t\}_0^T = \arg \min_{\theta} E_{(x,y) \in \mathcal{D}} [L_D(x, y, f_\theta)]. \quad (3)$$

Since these trajectories are built on real-world optimizers, we refer to them as offline trajectories, i.e.,  $\tau_{of}^{0:T} = \{\theta_t\}_0^T$ . Specifically, multiple optimizers are used to solve Equation 3, from which we collect the checkpoint weights  $\theta_t$  and the associated data  $x$ . Within an offline trajectory,  $\theta_0$  denotes the Gaussian-initialized weight,  $\theta_T$  represents the global optimum, and  $T$  is the total number of training epochs. In our implementation, we use two optimizers, i.e., Adam and SGD, and an auxiliary optimization policy, i.e., Sharpness Aware Minimization (SAM). Specifically, SGD is well-suited for large, clean datasets such

<sup>2</sup> The terms online trajectory, inference trajectory, and sampling trajectory are used interchangeably.

**Algorithm 1** Weight Preparation

---

**Require:** Downstream task weights  $\theta_0$ , downstream task loss  $L_D$ , perturbation  $\rho$ , optimization policy  $Op$ , dataset  $\mathcal{D}$ .

- 1: **for**  $t$  in  $\text{range}(T)$  **do**
- 2:   Sample batch of data  $S_i \sim \mathcal{D}$
- 3:    $\epsilon \leftarrow \rho \frac{\nabla L_D(S_i; \theta_t)}{\|\nabla L_D(S_i; \theta_t)\|}$
- 4:    $g_{SAM} \leftarrow \nabla_{\theta} L_D(S_i; \theta_t + \epsilon)$
- 5:    $\theta_{t+1} = Op(\theta_t, g_{SAM})$
- 6:   Append  $\theta_t$  to  $\tau_{off}$
- 7: **end for**
- 8: Randomly sample sub-trajectories  $\tau_{off}^{m:n}$  from  $\tau_{off}$
- 9: **return**  $\{\tau_{off}^{m:n}\}_{m \in [0, T], n \in (m, T]}$

---

as ImageNet, while Adam excels in fast convergence tasks like few-shot learning on Mini-ImageNet. The motivation and role of SAM are detailed in Section 3.3.

Compared to the end-to-end optimization objective given by Equation 1, the decoupled weight preparation offers more choices for the optimization policy. By improving the flexibility of optimization policies, it further enhances the robustness of the policy learned by the generative model  $f_{\phi}^G$ . Our experiments in Section 4.1.2 support the above claim. The concern about overhead is detailed in the Supplementary Material C.3. The specific process of weight preparation is detailed in Algorithm 1.

## 2.2 Policy Learning

As illustrated on the right side of Figure 2, we adopt a learnable Gaussian policy for inference in continuous weight space. Formally, the online inference trajectory, i.e.,  $\tau_{on}^{0:N} = \{s_i\}_0^N$ , starts from  $s_0 = \theta_0 \sim N(0, 1)$  and is driven by

$$s_{t+1} \sim \mathcal{N}(\mu_{\phi}(s_t), \sigma_{\phi}(s_t)). \quad (4)$$

It can be found that the inference trajectory is determined solely by the Gaussian policy, ignoring the policy details behind real-world offline trajectories. We refer to this on-policy method, which models only the global optima, as Lo-Op. As shown by the purple trajectory in Figure 1, Lo-Op's unconstrained inference trajectory exhibits low efficiency and poor accuracy over a long horizon. Previous methods, such as OCD, MetaDiff, and D2NWG, fall into this category. In this paper, we propose Hybrid-Policy Sub-Trajectory Balance to introduce a supervision signal from offline sub-trajectories to the learning process of online sub-trajectories. It captures local optimization policies and enables better efficiency and accuracy. For the generative model  $f_{\phi}^G$ , we use the commonly used U-Net architecture, conditioning on the unlabeled samples  $\{x_i\} \in \mathcal{D}$  to differentiate among tasks (Equation 2). The specific process of policy learning is detailed in Algorithm 2.

### 2.2.1 Hybrid-Policy Sub-Trajectory Balance

For an online sub-trajectory  $\tau_{on}^{m':n'} = \{s_t\}_{m'}^{n'}$ , the loss function of vanilla sub-trajectory balance [23](sub-TB) can be written as

$$\begin{aligned} \mathcal{L}^{sub} = E_{m', n'} \left\| \log F_{\phi}(s_{m'}) + \sum_{t=m'}^{n'-1} \log P_{\phi}^F(s_{t+1} | s_t) \right. \\ \left. - \log F_{\phi}(s'_{n'}) - \sum_{t=m'}^{n'-1} \log P_{\phi}^B(s_t | s_{t+1}) \right\|_2^2, \quad (5) \end{aligned}$$

**Algorithm 2** Policy Learning

---

**Require:** Generative model  $f_{\phi}^G$ , learning rate  $\alpha$ , sub-trajectories  $\{\tau_{off}^{m:n}\}_{m \in [0, T], n \in (m, T]}$

- 1: **while** not converged **do**
- 2:   Select offline sub-trajectory  $\tau_{off}^{m:n}$
- 3:   Start from  $s_0 = \theta_0$ , and sample online trajectory  $\tau_{on}^{0:N}$  (Equation 4)
- 4:   Match  $\tau_{on}^{m':n'}$  for  $\tau_{off}^{m:n}$  (Equation 9)
- 5:   Compute  $\nabla_{\phi} \mathcal{L}_{hy}^{sub}(\tau_{on}^{m':n'}, \tau_{off}^{m:n}; \phi)$
- 6:   Update  $\phi \leftarrow \phi - \alpha \nabla \mathcal{L}_{hy}^{sub}$
- 7: **end while**

---

where the generative model is defined as  $f_{\phi}^G := \{P_{\phi}^F, P_{\phi}^B, F_{\phi}\}$ ,  $P_{\phi}^F$  is the learnable forward Gaussian policy,  $P_{\phi}^B$  is the learnable backward Gaussian policy, and  $F_{\phi}$  is the flow function. Following sub-TB,  $P_{\phi}^F$ ,  $P_{\phi}^B$ , and  $F_{\phi}$  are parameterized by  $\phi$ , utilizing a shared bottleneck for representation learning and separate heads to distinguish the forward policy, backward policy, and flow function, respectively.<sup>3</sup> The conditional probability  $P_{\phi}(s_{t+1} | s_t)$  is given by

$$\begin{aligned} \log P_{\phi}(s_{t+1} | s_t) = & -\frac{1}{2}(s_{t+1} - \mu_{\phi}(s_t))^{\top} \sigma_{\phi}^{-1}(s_t)(s_{t+1} - \mu_{\phi}(s_t)) \\ & -\frac{1}{2} \log \left[ (2\pi)^d \det(\sigma_{\phi}(s_t)) \right]. \quad (6) \end{aligned}$$

Given an offline sub-trajectory  $\tau_{off}^{m:n} = \{\theta_t\}_m^n$ , we define our hybrid-policy loss as<sup>4</sup>

$$\begin{aligned} \mathcal{L}_{hy}^{sub} = E_{m', n'} \left\| \log C_{\phi}(s_{m'}) R_n(s_{m'}) + \sum_{t=m'}^{n'-1} \log P_{\phi}^F(s_{t+1} | s_t) \right. \\ \left. - \log C_{\phi}(s_{n'}) R_n(s_{n'}) - \sum_{t=m'}^{n'-1} \log P_{\phi}^B(s_t | s_{t+1}) \right\|_2, \quad (7) \end{aligned}$$

where  $C_{\phi}(\cdot)$  is a learnable coefficient and  $R_n$  is the reward function

$$R_n(s_t) = e^{-\|s_t - \theta_n\|_2^2}. \quad (8)$$

We replace the flow function  $F_{\phi}(s_t)$  in vanilla sub-TB with  $C_{\phi}(s_t) R_n(s_t)$ , since the reward for intermediate states can be directly evaluated in our setting (i.e., weight generation).  $R_n(s_t)$  describes how close the current state is to the local target  $\theta_n$ . The matching strategy for the local target is detailed in Section 2.2.2.

Moreover, according to the principle of TB, the replaced  $F(s_t)$  is proportional to  $R_n(s_t)$ . To ensure consistency with this principle, we introduce a learnable coefficient  $C_{\phi}(\cdot)$ .<sup>5</sup> The theoretical soundness of the above design is established through the following theorems.

**Theorem 1.** Suppose that  $\mathcal{L}_{hy}^{sub} = 0$ . Then, the expected cumulative probability of the sub-inference trajectories  $\tau_{on}^{m':n'} = \{s_{m'}, \dots, s_{n'}\}$  satisfies

$$E_{\tau \in \tau_{on}^{m':n'}} \prod_{t=m'}^{n'-1} P_{\phi}^F(s_{t+1} | s_t) \propto R_n(s_{n'}).$$

<sup>3</sup> For simplicity, we use the same symbol  $\phi$  to represent the parameters of  $P^F$ ,  $P^B$ , and  $F$ , disregarding the differences in their heads.

<sup>4</sup> It is worth noting that we adopt  $\|\cdot\|_2$  instead of  $\|\cdot\|_2^2$  to enable the use of the triangle inequality in the proof of Theorem 2, thereby guaranteeing the desired global property.

<sup>5</sup> Similar to  $P^F$  and  $P^B$ , we use the same symbol  $\phi$  to denote the parameters, and omit the differences between the heads.

**Theorem 2.** Suppose that  $\mathcal{L}_{hy}^{sub} = 0$ . Then, the expected cumulative probability of the full inference trajectory  $\tau_{on}^{0:N} = \{s_0, \dots, s_N\}$  satisfies

$$E_{\tau \in \mathcal{T}_{0:N}} \prod_{t=1}^{N-1} P_{\phi}^F(s_{t+1} | s_t) \propto R_T(s_N).$$

The proof is detailed in the Supplementary Material. The above theorems indicate that our proposed objective  $\mathcal{L}_{hy}^{sub}$  possesses the following properties:

1. For sub-trajectories, their cumulative probability is proportional to  $R_n(s_{n'})$ . This indicates that the inferred sub-trajectory’s endpoint is very close to the local target  $\theta_n$ , suggesting that the learned optimizer  $f_{\phi}^G$  captures the local optimization policy.
2. For full trajectories, their cumulative probability is proportional to  $R_T(s_N)$ . This indicates that the inferred full trajectory’s endpoint is very close to the global optimum  $\theta_T$ , suggesting that the learned optimizer enables the generation of optimal weights.

Therefore, Lo-Hp can learn a local policy to guide sampling trajectory while facilitating the generation of global optimal weights.

### 2.2.2 Trajectory Matching

As shown in Figure 2, the lengths of the online trajectory  $T$  and offline trajectory  $N$  are inconsistent. To achieve better sub-trajectory matching, we adopt a uniform assignment strategy in our implementation. More precisely, we enforce the length of the online trajectory to satisfy  $T = kN$ , where  $k$  is a segmentation factor. Formally, any offline sub-trajectory  $Tra_{off}^{sub}$  and its corresponding online sub-trajectory  $Tra_{on}^{sub}$  satisfy the following relation:

$$\tau_{off}^{m:n} = \{\theta_m, \dots, \theta_n\} \xrightarrow{\text{match}} \tau_{on}^{m':n'} = \{s_{m'=\frac{m}{k}}, \dots, s_{n'=\frac{n}{k}}\}. \quad (9)$$

In addition to gradient-free optimization, this design allows Lo-Hp to solve in the weight space  $k$  times faster than real-world optimizers. As a result, during the weight preparation stage, we can use a very small learning rate to ensure the stability and accuracy of offline trajectories. On the other hand, we can avoid the overhead caused by overly long online trajectories.

## 3 Discussion

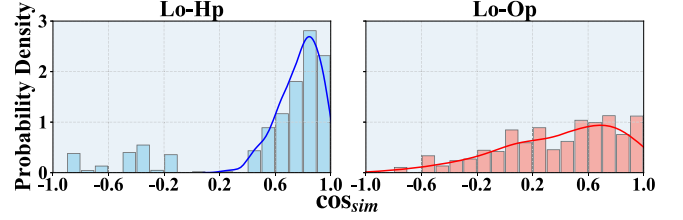
In this section, we discuss how to improve Lo-Hp’s efficiency by  $k$ . We also analyze Lo-Hp’s improvement on the local optimization policy. Furthermore, we provide a convergence analysis of such a decoupled weight generation framework and introduce SAM to improve convergence efficiency.

### 3.1 Efficiency Improvement

According to the matching strategy provided by Equation 9, we can increase  $k$  to reduce the sampling trajectory length  $N$ , thereby further improving inference efficiency. Table 3.1 shows how the accuracy and inference latency of Lo-Hp vary with different values of  $k$ . We evaluate on the CIFAR-10 transfer learning task and the Mini-ImageNet 5-way 1-shot task (see Section 4.2 for setup details). As  $k$  increases, inference latency decreases due to fewer online states  $s_t$  being used to estimate the offline sub-trajectories (Equation 9). However, this comes at the cost of reduced accuracy. To balance accuracy and latency, we set  $k = 2$  in the following experiments, achieving a  $2\times$  speedup.

**Table 1.** Accuracy and inference latency under different values of  $k$  on CIFAR-10 transfer learning task and Mini-ImageNet 5-way 1-shot task.

|           | CIFAR-10         | Mini-ImageNet    | Latency(ms) |
|-----------|------------------|------------------|-------------|
| $k = 1/2$ | $65.11 \pm 0.25$ | $66.19 \pm 0.27$ | 12.6        |
| $k = 1$   | $64.97 \pm 0.22$ | $66.04 \pm 0.29$ | 6.1         |
| $k = 2$   | $64.25 \pm 0.28$ | $65.21 \pm 0.36$ | 2.9         |
| $k = 3$   | $62.58 \pm 0.56$ | $63.87 \pm 0.60$ | 2.1         |



**Figure 3.** Similarity statistics between generated online sub-trajectories and target offline sub-trajectories on CIFAR-10.

### 3.2 Local Policy Details

The proposed Lo-Hp aims to capture local policy details through hybrid policy sub-trajectory learning. Therefore, beyond the final optimal weight, we are also interested in whether Lo-Hp exhibits optimizer-like behavior at the local level. To measure whether a learned optimizer  $f_{\phi}^G$  behaves like a real-world optimizer, we compute the cosine similarity between its generated online sub-trajectories  $\tau_{on}^{m' \rightarrow n'}$  and the offline target  $\tau_{off}^{m \rightarrow n}$  as

$$\text{cos}_{sim} = \frac{(s_{n'} - s_{m'})(\theta_n - \theta_m)}{\|s_{n'} - s_{m'}\| \cdot \|\theta_n - \theta_m\|}. \quad (10)$$

Figure 3 shows the distribution of Lo-Hp, which learns local policy details, and Lo-Op, which models only global optima. Lo-Hp yields a more concentrated distribution near 1, while Lo-Op is more dispersed, indicating that Lo-Hp aligns better with real optimizers even at the local level.

### 3.3 Convergence Analysis and Improvement

Unlike previous end-to-end frameworks, the decoupled framework used here involves multiple independent losses and models, making convergence harder to guarantee. We next derive its empirical error bound and introduce improvements.

**Theorem 3.** When the reconstruction error of the generative model is bounded by  $c$ , the downstream loss satisfies  $L_d(\cdot) \leq \psi$ , and the loss function is both  $l$ -smooth and  $\mu$ -strongly convex, with the eigenvalues of the Hessian matrix around the optimum  $\theta_*$  bounded by  $\lambda$ , the cumulative empirical error of the decoupled weight generation framework can be bounded as follows

$$L_D(\hat{\theta}) - L_D(\theta^*) \leq \frac{\lambda}{2} \left[ c + \frac{2\psi}{\mu} \left( 1 - \frac{\mu}{l} \right)^T \right], \quad (11)$$

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

The proof, provided in the Supplementary Material, relies on the triangle inequality to decompose the accumulated error into weight preparation error and reconstruction error. We make a  $\mu$ -strong convex assumption here, but subsequent analysis and improvement do not rely on this property, thus preserving the practicality of our derivation. Theorem 3 shows that, compared to direct learning methods, the reconstruction error of weight generation algorithms affects



**Table 2.** Ablation main components on CIFAR-10 transfer learning task and Mini-Imagenet 5-way 1-shot task. Metric by accuracy.

|              | C1 | C2 | C3 | CIFAR-10         | Mini-Imagenet    |
|--------------|----|----|----|------------------|------------------|
| Hypernetwork |    |    |    | 43.71 $\pm$ 0.20 | 45.29 $\pm$ 0.28 |
| Lo-Di        | ✓  |    |    | 61.67 $\pm$ 0.14 | 61.84 $\pm$ 0.30 |
| Lo-Op        | ✓  | ✓  |    | 61.28 $\pm$ 0.32 | 62.53 $\pm$ 0.37 |
| Lo-Hp        | ✓  | ✓  | ✓  | 64.25 $\pm$ 0.28 | 65.21 $\pm$ 0.36 |

the upper bound of cumulative error in only a linear manner. Furthermore, this upper bound can be effectively improved by reducing the maximum eigenvalue  $\lambda$ . Penalizing the Hessian matrix is the simplest way to accelerate convergence, but it is computationally unacceptable.

In this paper, we penalize  $\lambda$  by constraining the curvature near the neighborhood of the optimal solution. We use Sharpness-Aware Minimization (SAM) [9] in the weight preparation stage to achieve the above target. The process of SAM is shown in Algorithm 1.

## 4 Experiment

Our experimental platform includes two A100 GPUs, one Intel Xeon Gold 6348 processor, and 512 GB of DDR4 memory. For all experiment results, we report the mean and standard deviation over 5 independent repeated experiments. We present the basic experimental results, with more setup details provided in the Supplementary Material.

### 4.1 Ablation Study

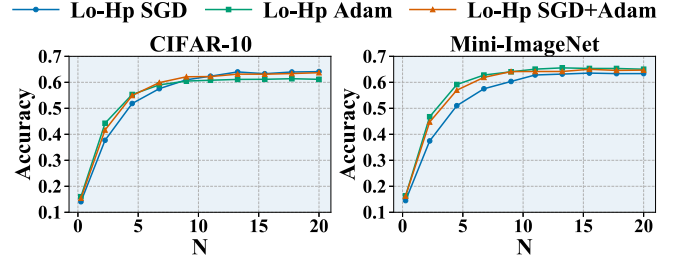
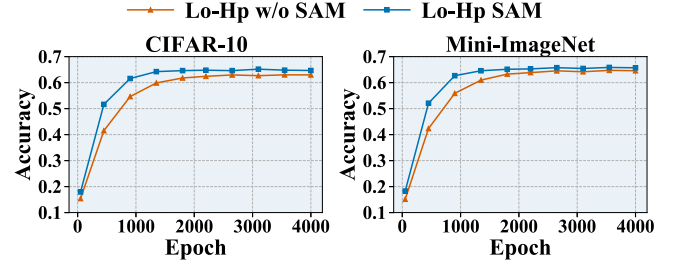
#### 4.1.1 Main Components

The advantages of Lo-Hp stem from three main components:

- C1: Using a decoupled weight generation framework to learn a more flexible optimization policy.
- C2: Using the trajectory balance loss to model the global optimal weights.
- C3: Using the proposed hybrid policy sub-trajectory balance loss to introduce offline supervision signals, thereby enabling the learning of local optimization policies.

Note that C2 builds upon C1, and C3 builds upon C2. Therefore, we conduct incremental ablation experiments here. As shown in Table 2, we validate the effectiveness of each component on CIFAR-10 transfer learning and Mini-ImageNet 5-way 1-shot tasks. We use four convolution blocks with a linear probe for classification. When none of the components are used, our method degrades to a Hypernetwork optimized by an end-to-end objective, i.e., Equation 1. When only C1 is used, we employ a commonly used diffusion algorithm (rather than trajectory balance) to model the global optimal weights, which we refer to as Lo-Di. When only C1 and C2 are used and the trajectory balance is applied to model global optimal weights, our method reduces to Lo-Op.

As shown in Table 2, Lo-Di and Lo-Op demonstrate higher accuracy compared to Hypernetwork, suggesting the necessity for a decoupled framework. The comparison between Lo-Di and Lo-Op shows that using different generative models does not significantly affect the accuracy on downstream tasks. However, once hybrid-policy sub-trajectory learning is incorporated, Lo-Hp surpasses both Lo-Op and Lo-Di, which can be attributed to its capability of learning local optimization policies.

**Figure 4.** The impact of different offline optimization policies on Lo-Hp's inference curve.**Figure 5.** The impact of SAM on Lo-Hp's learning curve.

#### 4.1.2 Offline Optimization Policies

In the weight preparation stage, we use two optimizers, i.e., SGD and Adam, to improve policy flexibility. Specifically, SGD is well-suited for large, clean datasets such as ImageNet, while Adam excels in fast-convergence tasks like few-shot learning on Mini-ImageNet. Figure 4 shows the learned optimizer's inference curves produced by three weight preparation schemes, i.e., SGD, Adam, and SGD+Adam, on CIFAR-10 and Mini-ImageNet. Although not consistently superior in all periods, the offline optimization trajectories that the combination of SGD and Adam enables Lo-Hp to achieve a more balanced inference speed and accuracy across diverse tasks.

To improve convergence efficiency, we introduce SAM. Figure 5 shows the learning curves of two schemes, i.e., Lo-Hp w/o SAM and standard Lo-Hp. It can be observed that SAM not only improves the overall convergence efficiency of Lo-Hp during training but also enhances downstream task accuracy on the CIFAR-10 transfer learning task.

## 4.2 Comparison Experiments

### 4.2.1 Transfer Learning

**Task.** In this task, we train and evaluate models on disjoint pre-training and evaluation datasets. During evaluation, we do not use any labeled data to adjust the models. We evaluated the transfer learning capability of the models using both accuracy and average per-sample latency.

**Dataset.** We partitioned ImageNet-1k [6] into 20k subsets of 50 classes each with 50 images per class per task for pre-training. The evaluation datasets are CIFAR-10, CIFAR-100 [17], STL-10 [5], Aircraft [22], and Pets [24].

**Baselines and Setups.** We categorize the baselines into three types: gradient-based methods using conventional optimizers, end-to-end weight generation methods, and decoupled weight generation methods. We compare Lo-Hp with representative baselines, including the gradient-based ICIS [4], the end-to-end GHN3 [16], and the decoupled method D2NWG [29]. For all the aforementioned models, the

**Table 3.** Transfer learning accuracy comparison on various datasets with average per-sample evaluation latency.

| Learning Type  | Method       | CIFAR-10                                           | CIFAR-100                          | STL-10                                             | Aircraft                                           | Pets                                               | Latency (ms)                         |
|----------------|--------------|----------------------------------------------------|------------------------------------|----------------------------------------------------|----------------------------------------------------|----------------------------------------------------|--------------------------------------|
| Gradient-Based | ICIS [4]     | 61.75 $\pm$ 0.31                                   | 47.66 $\pm$ 0.24                   | 80.59 $\pm$ 0.12                                   | 26.42 $\pm$ 1.56                                   | 28.71 $\pm$ 1.60                                   | 9.2                                  |
| End-to-End     | GHN3 [16]    | 51.80 $\pm$ 0.42                                   | 11.90 $\pm$ 0.45                   | 75.37 $\pm$ 0.19                                   | 23.19 $\pm$ 1.38                                   | 27.16 $\pm$ 1.08                                   | 14.5                                 |
| Decoupled      | D2NWG [29]   | 60.42 $\pm$ 0.75                                   | <b>51.50 <math>\pm</math> 0.25</b> | 82.42 $\pm$ 0.04                                   | 27.70 $\pm$ 3.24                                   | 32.17 $\pm$ 6.30                                   | 6.7                                  |
| Decoupled      | Lo-Op (ours) | 61.28 $\pm$ 0.32                                   | 48.42 $\pm$ 0.36                   | 79.63 $\pm$ 0.38                                   | 28.90 $\pm$ 1.22                                   | 30.71 $\pm$ 1.29                                   | 4.3                                  |
| Decoupled      | Lo-Hp (ours) | <b>64.25 <math>\pm</math> 0.28</b> $\uparrow$ 2.50 | 50.85 $\pm$ 0.49 $\downarrow$ 0.65 | <b>84.66 <math>\pm</math> 0.26</b> $\uparrow$ 2.24 | <b>30.08 <math>\pm</math> 1.02</b> $\uparrow$ 2.38 | <b>35.75 <math>\pm</math> 1.18</b> $\uparrow$ 3.58 | <b>2.2</b> $\downarrow$ $\times$ 3.0 |

**Table 4.** Few-shot task accuracy comparison on Omniglot, Mini-ImageNet, and Tiered-ImageNet datasets with average per-sample evaluation latency.

| Learning Type  | Method                 | Omniglot                           |                                    | Mini-ImageNet                                      |                                                    | Tiered-ImageNet                                    |                                                    | Latency (ms)                         |
|----------------|------------------------|------------------------------------|------------------------------------|----------------------------------------------------|----------------------------------------------------|----------------------------------------------------|----------------------------------------------------|--------------------------------------|
|                |                        | (5, 1)                             | (5, 5)                             | (5, 1)                                             | (5, 5)                                             | (5, 1)                                             | (5, 5)                                             |                                      |
| Gradient-Based | MAML [8]               | <b>98.70 <math>\pm</math> 0.40</b> | <b>99.90 <math>\pm</math> 0.10</b> | 48.70 $\pm$ 1.84                                   | 63.11 $\pm$ 0.92                                   | 48.95 $\pm$ 0.89                                   | 62.71 $\pm$ 0.77                                   | 20.9                                 |
| Gradient-Based | Meta-Baseline [3]      | 97.75 $\pm$ 0.25                   | 99.68 $\pm$ 0.18                   | 58.10 $\pm$ 0.31                                   | 74.50 $\pm$ 0.29                                   | 68.62 $\pm$ 0.29                                   | 83.29 $\pm$ 0.51                                   | 19.4                                 |
| End-to-End     | Meta-Hypernetwork [35] | 96.57 $\pm$ 0.22                   | 98.83 $\pm$ 0.16                   | 52.50 $\pm$ 0.28                                   | 67.76 $\pm$ 0.34                                   | 53.80 $\pm$ 0.35                                   | 69.98 $\pm$ 0.42                                   | 13.1                                 |
| End-to-End     | GHN3 [16]              | 95.23 $\pm$ 0.23                   | 98.65 $\pm$ 0.19                   | 63.22 $\pm$ 0.29                                   | 76.79 $\pm$ 0.33                                   | 64.72 $\pm$ 0.36                                   | 78.40 $\pm$ 0.46                                   | 17.5                                 |
| Decoupled      | OCD [20]               | 95.04 $\pm$ 0.18                   | 98.74 $\pm$ 0.14                   | 59.76 $\pm$ 0.27                                   | 75.16 $\pm$ 0.35                                   | 60.01 $\pm$ 0.38                                   | 76.33 $\pm$ 0.47                                   | 8.4                                  |
| Decoupled      | Meta-Diff [31]         | 94.65 $\pm$ 0.65                   | 97.91 $\pm$ 0.53                   | 55.06 $\pm$ 0.81                                   | 73.18 $\pm$ 0.64                                   | 57.77 $\pm$ 0.90                                   | 75.46 $\pm$ 0.69                                   | 8.9                                  |
| Decoupled      | D2NWG [29]             | 96.77 $\pm$ 6.13                   | 98.94 $\pm$ 7.49                   | 61.13 $\pm$ 8.50                                   | 76.94 $\pm$ 6.04                                   | 65.33 $\pm$ 6.50                                   | 85.05 $\pm$ 8.25                                   | 10.4                                 |
| Decoupled      | Lo-Op (ours)           | 96.65 $\pm$ 0.19                   | 99.34 $\pm$ 0.23                   | 62.53 $\pm$ 0.37                                   | 76.25 $\pm$ 0.28                                   | 64.72 $\pm$ 0.17                                   | 83.26 $\pm$ 0.49                                   | 6.7                                  |
| Decoupled      | Lo-Hp (ours)           | 98.25 $\pm$ 0.19 $\downarrow$ 0.45 | 99.67 $\pm$ 0.13 $\downarrow$ 0.23 | <b>65.21 <math>\pm</math> 0.36</b> $\uparrow$ 1.99 | <b>80.17 <math>\pm</math> 0.16</b> $\uparrow$ 3.23 | <b>69.88 <math>\pm</math> 0.21</b> $\uparrow$ 1.26 | <b>88.36 <math>\pm</math> 0.26</b> $\uparrow$ 3.31 | <b>3.6</b> $\downarrow$ $\times$ 2.3 |

downstream network uses ResNet12 [12] with a linear probe for classification. For the generative model  $f_{\phi}^G$ , Lo-Hp employs the same U-Net architecture given by Meta-Diff [31]. In the weight preparation stage, the real optimizers SGD and Adam use a fixed learning rate of 0.005 and an automatic early-stopping strategy [25] to determine the downstream task training epoch  $T$ . In the policy learning stage, we set the learning rate  $\alpha$  and training epochs to 0.001 and 6000, respectively. The acceleration coefficient  $k$  is set to 2, and the inference step  $N$  is computed as  $N = T/k$  for each task. **We maintain this setup across all experiments in this paper.**

**Results.** Table 3 shows that Lo-Hp achieves the highest accuracy on five out of six tasks while also reducing average latency. Compared to the second-best method on each task, it yields an average accuracy improvement of 2.68%. On CIFAR-100, its accuracy is only 0.65% lower than the best-performing method. Compared to the fastest existing method, D2NWG, Lo-Hp reduces inference latency by  $3.0\times$ . As discussed in Section 4.1.1, Lo-Hp also outperforms its simplified variant Lo-Op, demonstrating the effectiveness of hybrid-policy sub-trajectory balance.

#### 4.2.2 Few-shot Learning

**Task.** Following the setup provided by MAML [8], we train and evaluate models on disjoint meta-training and meta-testing tasks. During the evaluation stage, we fine-tune the models on the support set of each meta-test task and measure the per-sample fine-tuning latency. We then evaluate the accuracy on the corresponding query set.

**Dataset.** We use Omniglot [18], Mini-ImageNet [7], and Tiered-ImageNet [26] datasets for the construction of 5-way 1-shot, 5-way 5-shot tasks. To evaluate generalization capabilities, we maintain distinct and separate class sets for training and evaluating phases.

**Baselines.** Our benchmarks include gradient-based methods MAML [8] and Meta-Baseline [3], end-to-end weight generation approaches Meta-Hypernetwork [35], GHN3 [16], and decoupled frameworks OCD [20], Meta-Diff [31], D2NWG [29]. Following the setting given by MAML [8], the downstream neural network uses four convolution blocks with a linear probe for classification.

**Results.** Table 3 shows that Lo-Hp can improve performance on almost all tasks. Since the 5-way task of Omniglot is relatively easy to learn, the gradient-based method MAML algorithm can achieve slightly higher accuracy compared to gradient-free methods. On the winning tasks, compared to the second-best method, Lo-Hp achieves

an average improvement of 2.45% in accuracy. Compared to the current fastest weight generation algorithm, i.e., OCD, Lo-Hp achieves a  $2.3\times$  reduction in inference latency. Note that the gradient-based methods, MAML and Meta-Baseline, exhibit high latency here due to the need for gradient computation during fine-tuning.

#### 4.2.3 Multi-Domain Generalization

**Task.** In this task, we explore the domain generalization ability of our method. We follow the few-shot task setting given by H-Meta [11] to evaluate the model’s performance.

**Dataset.** We use DomainNet [19] for the construction of 5-way 1-shot and 20-way 5-shot tasks. Specifically, we use Clipart, Infograph, Painting, Quickdraw, and Real domains for training, while Sketch domains are used for evaluation. Under this setting, the tasks in the training set may come from different domains, and the tasks in the testing set come from another unseen domain.

**Baselines.** We benchmark against MAML, Meta-baseline, Meta-Hypernetwork, GHN3, OCD, Meta-Diff, and D2NWG. The downstream network uses ResNet12 with a linear probe for classification.

**Results.** Table 5 shows that Lo-Hp significantly outperforms current methods on few-shot domain generalization tasks. Compared to the second-best baselines in each task, Lo-Hp achieved an average improvement of 4.64% in accuracy. It can be observed that, compared to gradient-based methods MAML and Meta-Baseline, gradient-free methods (i.e., end-to-end and decoupled methods) exhibit a significant advantage gap. This is attributed to the generalization capability brought by the indirect weight generation method. Extending these approaches, Lo-Hp incorporates local optimization policy learning, which leads to further performance improvements. Lo-Hp’s generalization capability stems from our learning objective, local optimization policies, which remain invariant across different datasets. In terms of overhead, Lo-Hp achieves a  $2.3\times$  reduction in latency compared to OCD, showing the same advantage as in transfer learning and few-shot learning tasks.

#### 4.2.4 Large Language Model fine-tuning

**Task.** We demonstrate that Lo-Hp can be applied to the fine-tuning of Large Language Models by learning to generate LoRA [14] matrices for new tasks. We compared the algorithms in terms of their fine-tuning accuracy upon convergence and the latency required to achieve it.

**Table 5.** Multi-domain generalization accuracy comparison on DomainNet with 5-way 1-shot and 20-way 5-shot tasks.

| Learning Type  | Method                 | DomainNet                 |                           | Latency(ms)       |
|----------------|------------------------|---------------------------|---------------------------|-------------------|
|                |                        | (5, 1)                    | (20, 5)                   |                   |
| Gradient-Based | MAML [8]               | 45.92 ± 0.39              | 50.18 ± 0.51              | 26.8              |
| Gradient-Based | Meta-Baseline [3]      | 50.54 ± 0.47              | 54.45 ± 0.40              | 26.2              |
| End-to-End     | Meta-Hypernetwork [35] | 59.00 ± 0.39              | 63.32 ± 0.35              | 10.3              |
| End-to-End     | GHN3 [16]              | 63.11 ± 0.36              | 66.42 ± 0.33              | 30.1              |
| Decoupled      | OCD [20]               | 64.58 ± 0.42              | 67.10 ± 0.38              | 9.7               |
| Decoupled      | Meta-Diff [31]         | 64.24 ± 0.41              | 67.58 ± 0.39              | 10.9              |
| Decoupled      | D2NKG [29]             | 63.68 ± 3.38              | 65.72 ± 2.85              | 10.9              |
| Decoupled      | Lo-Op (ours)           | 65.96 ± 0.47              | 67.13 ± 0.40              | 7.5               |
| Decoupled      | Lo-Hp (ours)           | <b>70.29 ± 0.45</b> †5.71 | <b>72.98 ± 5.40</b> †3.58 | <b>4.0</b> ↓ ×2.3 |

**Datasets.** We conduct a case study to demonstrate the generalizability and efficiency of Lo-Hp. We use five binary classification tasks, i.e., SST-2, QQP, RTE, WNIL, and CoLA from the GLUE [30] benchmark for pre-training. Then we use the other two tasks, i.e., MRPC and QNLI, to evaluate the performance of the methods.

**Baselines.** We benchmark against Full-fine-tuning baseline, LoRA [14], AdaLoRA [33], DyLoRA [36], and FourierFT [21], which are all gradient-based fine-tuning algorithms. The large language model we fine-tuned is RoBERTa-base [21] and the LoRA matrices are generated following the fine-tuning process given by FourierFT [21]. The experimental results in Section 4.2.3 suggest that the Lo-Hp exhibits strong generalization capability, enabling it to learn across all training tasks. In contrast, gradient-based methods are single-task fine-tuning approaches that operate on one task at a time. Note that Lo-Hp requires additional time to pre-train the generative model  $f_{\phi}^G$ ; however, this is a one-time cost and demonstrates better potential in multi-task fine-tuning scenarios.

**Results.** Table 6 shows that Lo-Hp achieves comparable binary classification accuracy on two evaluation tasks compared to other gradient-based fine-tuning algorithms while significantly accelerating the fine-tuning speed by ×5.7 to ×5.9. By implementing a decoupled framework with multiple optimizers, Lo-Hp demonstrates remarkable efficiency in capturing shared local optimization policies, enabling gradient-free generation of LoRA matrices.

## 5 Related Work

### 5.1 Weight Generation.

The task of weight generation aims to generate neural network weights without gradient-based updates directly. Early approaches such as Meta-HyperNetwork [35], GHN2 [15], and GHN3 [16] adopt an end-to-end framework, where a meta-network is trained to generate weights optimized for downstream task performance. These methods, though efficient, are constrained by over-coupling between the weight generation and the task-specific objectives. This reduces the flexibility of the learned weight generator and limits the inference process to short horizons due to constraints on differentiability and tractable computation. Recent developments extend weight generation using generative models such as OCD [20], Meta-Diff [31], and D2NKG [29], which model optimal downstream task weights (rather than downstream task performance) via diffusion algorithms. These approaches support a long-horizon inference by treating the weight optimization process as a sampling process.

Despite their innovations, these methods focus solely on global optimal weights while neglecting the rich dynamics of intermediate optimization steps. As a result, they fail to model the local optimization policy—i.e., how weights update over time during optimization.

**Table 6.** Accuracy and fine-tuning latency comparison on GLUE-MRPC and GLUE-QNLI tasks with different fine-tuning algorithms.

| fine-tuning Type | Method         | MRPC                |                    | QNLI                |                    |
|------------------|----------------|---------------------|--------------------|---------------------|--------------------|
|                  |                | Acc                 | Latency (h)        | Acc                 | Latency (h)        |
| Gradient-Based   | Full-fine-tune | <b>90.24 ± 0.57</b> | 1.47               | 92.84 ± 0.26        | 3.15               |
| Gradient-Based   | LoRA [14]      | 89.76 ± 0.69        | 0.81               | <b>93.32 ± 0.20</b> | 1.76               |
| Gradient-Based   | AdaLoRA [33]   | 88.71 ± 0.73        | 0.74               | 93.17 ± 0.25        | 1.68               |
| Gradient-Based   | DyLoRA [36]    | 89.59 ± 0.81        | 0.76               | 92.21 ± 0.32        | 1.63               |
| Gradient-Based   | FourierFT [21] | 90.03 ± 0.54        | 0.68               | 92.25 ± 0.15        | 1.55               |
| Gradient-Free    | Lo-Op (ours)   | 87.59 ± 0.57        | 0.25               | 90.48 ± 0.27        | 0.51               |
| Gradient-Free    | Lo-Hp (ours)   | 89.62 ± 0.66        | ↓0.62              | 92.38 ± 0.29        | ↓0.94              |
|                  |                |                     | <b>0.12</b> ↓ ×5.7 |                     | <b>0.26</b> ↓ ×5.9 |

### 5.2 Generative Model

Diffusion-based generative models, such as DDPM [13], Score-based Generative Models [28], and Latent Diffusion [27], generate data through iterative denoising and have been widely applied to continuous or structured data domains. Their forward-backward refinement processes enable strong sample quality and controllability. Their strength lies in the multi-step refinement mechanism, which offers controllability and robustness in modeling complex, structured distributions. Bridging the conceptual gap between diffusion and discrete generative models, Zhang et al. [34] shows that GFlowNets can be viewed as a generalized form of diffusion models, where sampling follows learned stochastic policies over compositional trajectories rather than fixed noise schedules. In this context, GFlowNets [1] generate discrete trajectories leading to final states, sampling them with probabilities proportional to a reward function. To ensure consistent distributions over trajectories, the Trajectory Balance (TB) and sub-Trajectory Balance (sub-TB) objectives [23] were proposed. TB and sub-TB ensure equality between the forward and backward trajectory probabilities, scaled by a learnable flow term.

Despite these advances, diffusion models and GFlowNet-like methods typically rely on on-policy learning: models are supervised only by self-sampled trajectories and final reward, limiting their capacity to align generated intermediate states with off-policy data or external supervision signals.

## 6 Conclusion

In this work, we target the problem of weight generation by viewing it as an optimization policy learning problem. We propose Lo-Hp, a novel weight generation framework for the issues of over-coupling and long horizon. To address the limited flexibility caused by over-coupling, our method adopts a decoupled two-stage learning process that enables the learning of diverse local optimization policies. To address the inefficiency in inference caused by the long-horizon issue, Lo-Hp introduces Hybrid-Policy Sub-Trajectory Balance to capture local optimization policies. Theoretically, we demonstrate that focusing solely on local optimization policy learning addresses the long-horizon issue while also enhancing the generation of global optimal weights. Empirically, we demonstrate Lo-Hp’s superior accuracy and inference efficiency in tasks that require frequent weight updates, such as transfer learning, few-shot learning, domain adaptation, and large language model adaptation.

## Acknowledgements

This work was supported by the China Scholarship Council (CSC) under Grant No. 202406160071, the Pioneer Centre for AI, DNRG grant number P1, the National Key Research and Development Program of China under Grant No. 2023YFB4502701, and the National Natural Science Foundation of China under Grant No. 62232007.



## References

- [1] Y. Bengio, S. Subramanian, Y. Xu, T. Zhang, O. Teytaud, A. Madani, S. Liu, R. Zhang, E. Bengio, and O. Delalleau. Gflownet foundations. *arXiv preprint arXiv:2305.17136*, 2023.
- [2] J. Chen, Y. Jiang, W. Lee, Y. Yang, H. Liu, X. B. Peng, A. Lee, Y.-X. Zhang, K. Lin, A. Zhou, et al. Deep generative models for offline policy learning: Tutorial, survey, and perspectives on future directions. *arXiv preprint arXiv:2402.13777*, 2024.
- [3] Y. Chen, Z. Liu, H. Xu, T. Darrell, and X. Wang. Meta-baseline: Exploring simple meta-learning for few-shot learning. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 9042–9051. IEEE, 2021.
- [4] A. Christensen, M. Mancini, A. S. Koepke, O. Winther, and Z. Akata. Image-free classifier injection for zero-shot classification. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 19026–19035. IEEE, 2023.
- [5] A. Coates, A. Y. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 215–223, 2011.
- [6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. IEEE, 2009.
- [7] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32, pages 647–655. JMLR.org, 2014.
- [8] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70, pages 1126–1135. PMLR, 2017.
- [9] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.
- [10] R. Gozalo-Brizuela and E. C. Garrido-Merchán. A survey of generative AI applications. *CoRR*, abs/2306.02781, 2023.
- [11] Y. Guan, Y. Liu, K. Zhou, and J. Huang. Hierarchical meta-learning with hyper-tasks for few-shot learning. In I. Frommholz, F. Hopfgartner, M. Lee, M. Oakes, M. Lalmas, M. Zhang, and R. L. T. Santos, editors, *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, CIKM 2023, Birmingham, United Kingdom, October 21-25, 2023*, pages 587–596. ACM, 2023.
- [12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE, 2016.
- [13] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
- [14] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.
- [15] B. Knyazev, M. Drozdal, G. W. Taylor, and A. Romero-Soriano. Parameter prediction for unseen deep architectures. In M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 29433–29448, 2021.
- [16] B. Knyazev, D. Hwang, and S. Lacoste-Julien. Can we scale transformers to predict parameters of diverse imagenet models? In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 17243–17259. PMLR, 2023.
- [17] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.
- [18] B. M. Lake, R. Salakhutdinov, J. Gross, and J. B. Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the 33th Annual Meeting of the Cognitive Science Society, CogSci 2011, Boston, Massachusetts, USA, July 20-23, 2011*. cognitivesciencesociety.org, 2011.
- [19] A. Leventidis, L. D. Rocco, W. Gatterbauer, R. J. Miller, and M. Riedewald. Domainnet: Homograph detection for data lake disambiguation. In *Proceedings of the 24th International Conference on Extending Database Technology, EDBT 2021, Nicosia, Cyprus, March 23 - 26, 2021*, pages 13–24. OpenProceedings.org, 2021.
- [20] S. Lutati and L. Wolf. OCD: learning to overfit with conditional diffusion models. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 23157–23169. PMLR, 2023.
- [21] S. Ma, C. Zhou, S. Xie, X. Chen, J. Liu, and J. Gao. Fourier frequency tuning for parameter-efficient fine-tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3456–3467, 2023.
- [22] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. In *arXiv preprint arXiv:1306.5151*, 2013.
- [23] N. Malkin, M. Jain, E. Bengio, C. Sun, and Y. Bengio. Trajectory balance: Improved credit assignment in gflownets. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.
- [24] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. Jawahar. Cats and dogs. *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.
- [25] L. Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 2002.
- [26] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel. Meta-learning for semi-supervised few-shot classification. In *International Conference on Learning Representations (ICLR)*, 2018. URL <https://arxiv.org/abs/1803.00676>.
- [27] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022.
- [28] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.
- [29] B. Soro, B. Andreis, H. Lee, W. Jeong, S. Chong, F. Hutter, and S. J. Hwang. Diffusion-based neural network weights generation. *arXiv preprint arXiv:2402.18153*, 2024.
- [30] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.
- [31] B. Zhang, C. Luo, D. Yu, X. Li, H. Lin, Y. Ye, and B. Zhang. Metadiff: Meta-learning with conditional diffusion for few-shot learning. In M. J. Wooldridge, J. G. Dy, and S. Natarajan, editors, *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2024, February 20-27, 2024, Vancouver, Canada*, pages 16687–16695. AAAI Press, 2024.
- [32] C. Zhang, M. Ren, and R. Urtasun. Graph hypernetworks for neural architecture search. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.
- [33] R. Zhang, J. Li, Y. Xie, X. Liu, Y. Zhao, Q. Wang, and Z. Liu. AdaLoRA: Towards efficient adaptive low-rank adaptation for large language models. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, pages 5678–5689, 2023.
- [34] T. Zhang, Y. Bengio, Y. Xu, S. Subramanian, R. Zhang, O. Teytaud, A. Madani, E. Bengio, O. Delalleau, S. Liu, et al. Unifying generative models with gflownets and beyond. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
- [35] D. Zhao, S. Kobayashi, J. Sacramento, and J. von Oswald. Meta-learning via hypernetworks. In *4th Workshop on Meta-Learning at NeurIPS 2020 (MetaLearn 2020)*. NeurIPS, 2020.
- [36] X. Zhuang, Y. Cheng, Z. Gan, J. Liu, L. Liu, G. Chen, H. Zhang, M. Wang, S. Liu, and J. Gao. DyLoRA: Parameter-efficient tuning of pre-trained models via dynamic low-rank adaptation. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 2345–2356, 2023.