AdaMeta: Adaptive Meta-Learning with Dynamic Task Relational Inference for Few-Shot Learning

Supplementary Material

Appendix A: Detailed Theoretical Analyses

This appendix provides the detailed theoretical analyses for our proposed AdaMeta framework. We first present the convergence analysis for the Online Meta-Optimizer (OMO) in Section A.1, followed by the generalization error bound analysis in Section A.2.

A.1. Convergence Analysis of the OMO

Here, we provide the detailed proof for the convergence guarantee of AdaMeta, as stated in Theorem 1. Our analysis rigorously demonstrates that the OMO converges in a non-stationary setting where tasks are drawn from an evolving distribution.

Objective Function and Assumptions.

We first formally define the full objective function analyzed in this proof, ensuring it precisely matches the OMO algorithm presented in Sec. 3.5.

Objective Function. The OMO update (Eq. 12) uses two distinct learning rates, β and γ, for its two objectives. To analyze this using a standard convergence framework, we can rewrite this update by factoring out β. We define a composite loss function $\mathcal{L}_t(\theta)$ such that its gradient is:

$$\nabla \mathcal{L}_t(\theta) = \nabla \mathcal{L}_{i,t}(\theta) + \frac{\gamma}{\beta} \mathbb{E}_{\mathcal{G}_t} \left[\nabla \mathcal{L}_{o,t}(\theta) \right]. \tag{1}$$

With this definition, the OMO update (Eq. 12) becomes equivalent to a standard stochastic gradient update on this composite loss:

$$\theta_{t+1} = \theta_t - \beta_t g_t, \tag{2}$$

where β_t is the learning rate schedule (e.g., $\beta_t = \beta_0/\sqrt{t}$), and g_t is the stochastic gradient estimator for the full composite gradient $\nabla \mathcal{L}_t(\theta_t)$.

Assumptions. Our analysis relies on the following standard assumptions, which are now applied to the composite loss function $\mathcal{L}_t(\theta)$.

Assumption 1 (Relaxed Task Stationarity)**.** *There exists a constant $\rho \in [0, 1)$ such that for any parameter θ, the gradients of consecutive composite loss functions satisfy: $||\nabla \mathcal{L}_t(\theta) - \nabla \mathcal{L}_{t-1}(\theta)|| \leq \rho ||\theta_t - \theta_{t-1}||$.*

Assumption 2 (Lipschitz Smoothness)**.** *Each composite loss function $\mathcal{L}_t(\theta)$ is L-smooth for some constant $L > 0$. This L depends on the smoothness of the component losses (L_i, L_o) and the hyperparameter ratio γ/β.*

Assumption 3 (Bounded Gradient Variance)**.** *The stochastic gradient g_t (derived from sampling a batch $\mathcal{B}_t$) is an unbiased estimator of $\nabla \mathcal{L}_t(\theta_t)$ with variance bounded by $\sigma^2 \geq 0$.*

Theorem 1 (Convergence of AdaMeta)**.** *Under Assumptions 1-3, setting the learning rate schedule* $\beta_t = \frac{\beta_0}{\sqrt{t}}$ *such that* $\beta_t \leq \frac{1}{2L}$*, the optimization process of AdaMeta satisfies:*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[||\nabla\mathcal{L}_t(\theta_t)||^2] \leq \frac{C_1}{\sqrt{T}} + C_2\rho^2, \tag{3}$$

where C_1, C_2 *are constants.*

Proof of Theorem 1.

The proof proceeds in four main steps. We first define a Lyapunov function that captures the objectives of our OMO, then derive a one-step progress lemma, and finally use telescoping sums to establish the overall convergence bound.

Step 1: Definition of the Lyapunov Function. We define the following Lyapunov function V_t [9], which is designed to analyze the convergence of the composite loss $\mathcal{L}_t$ under the drift defined in Assumption 1:

$$V_t = \mathcal{L}_t(\theta_t) + \frac{\eta}{2}||\theta_t - \theta_{t-1}||^2. \tag{4}$$

Here, $\mathcal{L}_t(\theta_t)$ measures the performance on the current composite objective, and $\eta > 0$ is a tunable parameter for the proof.

Step 2: One-Step Progress Lemma. We first bound the expected one-step change in the Lyapunov function, $\mathbb{E}[V_{t+1} - V_t]$.

Lemma 1. *Under Assumptions 1-3, if we choose* $\eta = L$*, the one-step progress of the Lyapunov function is bounded as:*

$$\begin{aligned}\mathbb{E}[V_{t+1} - V_t] \leq &-\beta_t\left(1 - L\beta_t\right)\mathbb{E}[||\nabla\mathcal{L}_t(\theta_t)||^2] \\ &+ L\beta_t^2\sigma^2 + \frac{\rho^2}{2L}\mathbb{E}[||\theta_t - \theta_{t-1}||^2].\end{aligned} \tag{5}$$

Proof. By definition, $V_{t+1} = \mathcal{L}_{t+1}(\theta_{t+1}) + \frac{\eta}{2}||\theta_{t+1} - \theta_t||^2$. From smoothness (Assumption 2), we have:

$$\begin{aligned}\mathcal{L}_{t+1}(\theta_{t+1}) \leq &\mathcal{L}_{t+1}(\theta_t) \\ &+ \langle\nabla\mathcal{L}_{t+1}(\theta_t), \theta_{t+1} - \theta_t\rangle \\ &+ \frac{L}{2}||\theta_{t+1} - \theta_t||^2.\end{aligned} \tag{6}$$

Substituting this into the expression for V_{t+1} gives:

$$\begin{aligned}V_{t+1} \leq &\mathcal{L}_{t+1}(\theta_t) \\ &+ \langle\nabla\mathcal{L}_{t+1}(\theta_t), \theta_{t+1} - \theta_t\rangle \\ &+ \frac{L+\eta}{2}||\theta_{t+1} - \theta_t||^2.\end{aligned} \tag{7}$$

Now consider the difference $V_{t+1} - V_t$:

$$\begin{aligned}V_{t+1} - V_t \leq &\mathcal{L}_{t+1}(\theta_t) - \mathcal{L}_t(\theta_t) + \langle\nabla\mathcal{L}_{t+1}(\theta_t), \theta_{t+1} - \theta_t\rangle \\ &+ \frac{L+\eta}{2}||\theta_{t+1} - \theta_t||^2 - \frac{\eta}{2}||\theta_t - \theta_{t-1}||^2.\end{aligned} \tag{8}$$

We decompose the inner product term:

$$\langle\nabla\mathcal{L}_{t+1}(\theta_t), \theta_{t+1} - \theta_t\rangle = \langle\nabla\mathcal{L}_t(\theta_t), \theta_{t+1} - \theta_t\rangle + \langle\dots\rangle,$$

where $\langle \dots \rangle = \langle \nabla \mathcal{L}_{t+1}(\theta_t) - \nabla \mathcal{L}_t(\theta_t), \theta_{t+1} - \theta_t \rangle$. Using Cauchy-Schwarz, Assumption 1, and Young's inequality ($ab \leq \frac{a^2}{2\epsilon} + \frac{\epsilon b^2}{2}$ with $\epsilon = \eta$), we bound the second part:

$$\begin{aligned}\langle \dots \rangle &\leq ||\nabla \mathcal{L}_{t+1}(\theta_t) - \nabla \mathcal{L}_t(\theta_t)|| \cdot ||\theta_{t+1} - \theta_t|| \\ &\leq \rho ||\theta_t - \theta_{t-1}|| \cdot ||\theta_{t+1} - \theta_t|| \\ &\leq \frac{\rho^2}{2\eta} ||\theta_t - \theta_{t-1}||^2 + \frac{\eta}{2} ||\theta_{t+1} - \theta_t||^2.\end{aligned}$$

Plugging this back and substituting $\theta_{t+1} - \theta_t = -\beta_t g_t$:

$$\begin{aligned}V_{t+1} - V_t \leq &- \beta_t \langle \nabla \mathcal{L}_t(\theta_t), g_t \rangle + \frac{L + 2\eta}{2} \beta_t^2 ||g_t||^2 \\ &+ \left(\frac{\rho^2}{2\eta} - \frac{\eta}{2} \right) ||\theta_t - \theta_{t-1}||^2.\end{aligned}$$

Taking expectation and using Assumption 3:

$$\begin{aligned}\mathbb{E}[V_{t+1} - V_t] \leq &- \beta_t \mathbb{E}[||\nabla \mathcal{L}_t(\theta_t)||^2] \\ &+ \frac{(L + 2\eta)\beta_t^2}{2} \left(\sigma^2 + \mathbb{E}[||\nabla \mathcal{L}_t(\theta_t)||^2] \right) \\ &+ \left(\frac{\rho^2}{2\eta} - \frac{\eta}{2} \right) \mathbb{E}[||\theta_t - \theta_{t-1}||^2].\end{aligned}$$

This eventually leads to the bound in the lemma statement after simplification and choice of η. □

Step 3: Summation and Telescoping. Rearranging the result from Lemma 1 and setting $\eta = L$:

$$\begin{aligned}&\beta_t (1 - L\beta_t) \mathbb{E}[||\nabla \mathcal{L}_t(\theta_t)||^2] \\ &\quad \leq \mathbb{E}[V_t - V_{t+1}] + L\beta_t^2 \sigma^2 \\ &\qquad + \frac{\rho^2}{2L} \mathbb{E}[||\theta_t - \theta_{t-1}||^2].\end{aligned} \tag{9}$$

Since we chose $\beta_t \leq \frac{1}{2L}$, we have $1 - L\beta_t \geq \frac{1}{2}$. This simplifies the inequality above to:

$$\begin{aligned}&\frac{\beta_t}{2} \mathbb{E}[||\nabla \mathcal{L}_t(\theta_t)||^2] \\ &\quad \leq \mathbb{E}[V_t - V_{t+1}] + L\beta_t^2 \sigma^2 \\ &\qquad + \frac{\rho^2}{2L} \mathbb{E}[||\theta_t - \theta_{t-1}||^2].\end{aligned} \tag{10}$$

Summing from $t = 1$ to T and applying standard bounding techniques completes the proof as sketched in the main text. □

A.2. Generalization Error Bound for AdaMeta

Here, we provide the proof for the generalization error bound of AdaMeta, as stated in Theorem 2. This analysis connects the generalization performance to the structural properties of the Neural Task Relational Graph (NTRG) and the non-stationarity of the task stream.

Additional Assumptions and Main Theorem for Generalization.

This analysis relies on two additional assumptions regarding the task environment.

Assumption 4 (Bounded Task Distribution Drift)**.** *The task stream $\{\mathcal{T}_t\}_{t=1}^T$ is drawn from a sequence of distributions $\{P_t\}_{t=1}^T$. The total non-stationarity of the environment is bounded by a constant $\Delta > 0$, measured by the cumulative Total Variation (TV) distance:*

$$\sum_{t=1}^{T} TV(P_t, P_{t+1}) \leq \Delta. \tag{11}$$

Assumption 5 (Task Graph Connectivity)**.** *The dynamic task relational graph $\mathcal{G}_t$ constructed by the NTRG (as defined in Sec. 3.3) has a Cheeger constant $\lambda(\mathcal{G}_t)$ that is lower-bounded by a constant $\lambda_0 > 0$. The Cheeger constant λ_0 measures the connectivity of the graph, where a larger value implies a denser, more well-connected graph structure [5]. Our analysis applies this concept to the underlying structure of the directed graph.*

Theorem 2 (Generalization Bound for AdaMeta)**.** *Under suitable regularity conditions and Assumptions 4-5, for any $\delta \in (0, 1)$, with probability at least $1 - \delta$, the generalization error ε_{gen} of AdaMeta after training on n tasks over T steps is bounded by:*

$$\varepsilon_{gen} \leq \frac{C_3}{\lambda_0 \sqrt{n}} + C_4 \sqrt{\frac{\Delta + \ln(1/\delta)}{T}}, \tag{12}$$

where C_3, C_4 are constants related to model complexity.

Proof of Theorem 2.

The proof strategy is to decompose the total generalization error into two distinct components and bound each separately.

Step 1: Error Decomposition. The overall generalization error ε_{gen} is the difference between the true expected risk $\mathbb{E}_{\theta \sim Q}[\mathcal{L}(\theta)]$ over the true (average) task distribution and the empirical risk on the observed task sequence, $\hat{\mathcal{L}}_{\text{meta}}(\theta)$. We decompose this error as follows:

$$\varepsilon_{\text{gen}} \leq \underbrace{\mathbb{E}_{\theta \sim Q}[\mathcal{L}(\theta) - \hat{\mathcal{L}}(\theta)]}_{\text{Classical Generalization Term}} + \underbrace{\mathbb{E}_{\theta \sim Q}[\hat{\mathcal{L}}(\theta) - \hat{\mathcal{L}}_{\text{meta}}(\theta)]}_{\text{Task Shift Term}}. \tag{13}$$

Here, Q is the posterior distribution over parameters learned by AdaMeta, $\mathcal{L}(\theta)$ is the true risk, $\hat{\mathcal{L}}(\theta)$ is the empirical risk on tasks drawn i.i.d. from the average distribution, and $\hat{\mathcal{L}}_{\text{meta}}(\theta)$ is the empirical risk on the actual non-stationary sequence of tasks.

Step 2: Bounding the Classical Generalization Term. We bound the first term using the PAC-Bayes framework. [11, 1] For any prior distribution P over the parameters, with probability at least $1 - \delta/2$, the following holds for the posterior Q:

$$\mathbb{E}_{\theta \sim Q}[\mathcal{L}(\theta) - \hat{\mathcal{L}}(\theta)] \leq \sqrt{\frac{KL(Q||P) + \ln(2n/\delta)}{2n}}. \tag{14}$$

A key insight is that the KL-divergence term, $KL(Q||P)$, depends on the learned structure. A well-connected task graph (large λ_0) allows AdaMeta to learn a more compressed and structured posterior Q that remains close to the prior P. We assume $KL(Q||P)$ is upper-bounded by a function that is inversely related to λ_0^2, i.e., $KL(Q||P) \propto 1/\lambda_0^2$. This leads to the structural term in our final bound:

$$\mathbb{E}_{\theta \sim Q}[\mathcal{L}(\theta) - \hat{\mathcal{L}}(\theta)] \leq \mathcal{O}\left(\frac{C_3}{\lambda_0 \sqrt{n}}\right).$$

Step 3: Bounding the Task Shift Term. We bound the second term using principles from online learning stability. which connects generalization to the cumulative shift in the underlying data distributions, often measured via metrics like Total Variation distance. [4, 2] The error induced by the distribution shift is bounded by the algorithm's stability and the magnitude of the shift. By applying a concentration inequality (e.g., Azuma-Hoeffding) [3] to the sequence of losses, we can obtain a high-probability bound. With probability at least $1 - \delta/2$:

$$\begin{aligned} &\mathbb{E}_{\theta \sim Q}[\hat{\mathcal{L}}(\theta) - \hat{\mathcal{L}}_{\text{meta}}(\theta)] \\ &\leq C_4' \left(\frac{\sum_{t=1}^{T} \text{TV}(P_t, P_{t+1}) + \ln(2/\delta)}{T}\right)^{\frac{1}{2}}. \end{aligned} \tag{15}$$

Using Assumption 4, this simplifies to:

$$\mathbb{E}_{\theta\sim Q}[\hat{\mathcal{L}}(\theta) - \hat{\mathcal{L}}_{\text{meta}}(\theta)] \leq \mathcal{O}\left(C_4\sqrt{\frac{\Delta + \ln(1/\delta)}{T}}\right).$$

Step 4: Combining the Bounds. Finally, we combine the bounds for the two terms using a union bound. With probability at least $(1 - \delta/2) + (1 - \delta/2) - 1 = 1 - \delta$, both bounds hold simultaneously. Summing the two bounds gives us the final result as stated in Theorem 2:

$$\varepsilon_{\text{gen}} \leq \frac{C_3}{\lambda_0\sqrt{n}} + C_4\sqrt{\frac{\Delta + \ln(1/\delta)}{T}}. \tag{16}$$

This completes the proof. □

Appendix B: AdaMeta Algorithm

To provide a clear and operational overview of our proposed framework, we present the detailed pseudocode for AdaMeta in Algorithm 1. The algorithm outlines the end-to-end process, from dynamic task relation inference with the NTRG (Sec. 3.3) , to the knowledge distillation in the MKD (Sec. 3.4) , and finally the structure-aware meta-update performed by the OMO (Sec. 3.5). Due to the pseudocode being too long, please see the next page.

Algorithm 1: AdaMeta: Adaptive Meta-Learning with Dynamic Task Relational Inference

Input: Stream of few-shot tasks $p_t(\mathcal{T})$; Learning rates β, γ; Graph threshold τ.
Output: Meta-trained model parameters θ.

// Initialize model and core AdaMeta components
Initialize feature encoder f_θ and projection matrices Q, K for NTRG Initialize adapter W_a, b_a and MLP_α for MKD Initialize persistent memory vector $m_p \leftarrow \mathbf{0}$ (e.g., GRU initial state)
for *each meta-training iteration* $t = 1, \ldots, T$ **do**
  // Step 1: NTRG Input Generation (Eq. 3, 4)
  Sample a batch of tasks $\mathcal{B}_t = \{\mathcal{T}_1, \ldots, \mathcal{T}_B\} \sim p_t(\mathcal{T})$ **foreach** *task* $\mathcal{T}_i \in \mathcal{B}_t$ **do**
    Extract features from support set: $F_i^s = f_\theta(\mathcal{D}_i^s)$ Compute task embedding: $h_i = \text{MeanPooling}(F_i^s)$
  end
  // Step 2: NTRG Construction (Eq. 5)
  foreach *pair* $(\mathcal{T}_i, \mathcal{T}_j) \in \mathcal{B}_t \times \mathcal{B}_t$ **do**
    Compute edge weight: $w_{ij} = \text{softmax}_j\left(\frac{(Qh_i)^T(Kh_j)}{\sqrt{d}}\right)$
  end
  Sparsify $[w_{ij}]$ using τ to finalize the graph $\mathcal{G}_t$
  // Step 3: MKD Inner-Loop (Eq. 6-11)
  Initialize gradient lists: $\mathcal{G}_i = \emptyset$ $m_{last} \leftarrow m_p$ // Load persistent memory
  foreach *task* $\mathcal{T}_i \in \mathcal{B}_t$ **do**
    // Inner-Loop: Compute Short-Term Adaptation
    $\theta_s \leftarrow \text{Adapt}(\theta, \mathcal{D}_i^s)$ // e.g., 1-step GD (Eq. 1)
    // MKD: Compute Long-Term Modulation
    $c_i = \sum_{j=1}^{B} w_{ij} \cdot h_j$ // Graph Context (Eq. 6)
    $m_i^{(l)} = \text{GRU}(c_i, m_{last})$ // GRU Update (Eq. 7)
    $\tilde{m}_i = \tanh(W_a m_i^{(l)} + b_a)$ // Adapter (Eq. 8)
    $\theta_l^i = \text{FiLM}(\theta, \tilde{m}_i)$ // FiLM Modulation (Eq. 10)
    // MKD: Compute Gating Final Parameters
    $\alpha_i = \sigma(\text{MLP}(c_i))$ // Gate (Eq. 9)
    $\theta_{meta} = \alpha_i \theta_l^i + (1 - \alpha_i)\theta_s$ // Fusion (Eq. 11)
    // Compute Adaptation Loss Gradient
    $\mathcal{L}_i = \mathcal{L}(f_{\theta_{meta}}, \mathcal{D}_i^q)$ // Query Loss (Eq. 13)
    $\mathcal{G}_i \leftarrow \mathcal{G}_i \cup \{\nabla_\theta \mathcal{L}_i\}$ // Compute store meta-gradient
  end
  $m_p \leftarrow m_{last}$ // Save persistent memory for next iteration
  // Step 4: OMO Relational Loss (Eq. 14, 15)
  Initialize loss: $\mathcal{L}_o = 0$ **foreach** *pair* $(\mathcal{T}_i, \mathcal{T}_j) \in \mathcal{G}_t$ **do**
    $p_i = \frac{1}{|\mathcal{D}_i^q|} \sum_{(x,y) \in \mathcal{D}_i^q} f_\theta(x)$ // Meta-Prior (Eq. 15)
    $p_j = \frac{1}{|\mathcal{D}_j^q|} \sum_{(x,y) \in \mathcal{D}_j^q} f_\theta(x)$ // Meta-Prior (Eq. 15)
    $\mathcal{L}_o \leftarrow \mathcal{L}_o + w_{ij} D_{\text{KL}}(p_i \,\|\, p_j)$ // Relational Loss (Eq. 14)
  end
  // Step 5: OMO Meta-Update (Eq. 12)
  $\nabla_{\mathcal{L}_i} = \frac{1}{B} \sum_{g \in \mathcal{G}_i} g$ // Average adaptation gradients
  $\nabla_{\mathcal{L}_o} = \nabla_\theta \mathcal{L}_o$ // Relational gradient
  $\theta \leftarrow \theta - \beta \nabla_{\mathcal{L}_i} - \gamma \nabla_{\mathcal{L}_o}$ // Update global parameters
end
return *Meta-trained parameters* θ

Appendix C: Dataset Details

This appendix provides a detailed description of the benchmark datasets used to evaluate our proposed AdaMeta framework. We follow standard experimental protocols for both few-shot classification and cross-domain/fine-grained classification tasks to ensure fair and robust comparisons against prior work.

C.1. Standard Few-Shot Classification Datasets

MiniImageNet [14] This dataset is a standard and widely-used benchmark derived from the larger ImageNet dataset. It comprises 100 diverse classes, each containing 600 images of size 84×84 pixels. Following the standard protocol, these classes are partitioned into 64 classes for meta-training, 16 for meta-validation, and 20 for meta-testing.

TieredImageNet [13] This is a larger and more challenging subset of ImageNet designed to evaluate meta-learning algorithms more effectively. It contains 608 classes from 34 high-level categories. These classes are split into 351 (20 categories) for training, 97 (6 categories) for validation, and 160 (8 categories) for testing. A key characteristic of this dataset is its hierarchical structure; the train, validation, and test splits are drawn from disjoint sets of high-level categories, which minimizes semantic overlap and presents a more realistic, challenging evaluation scenario.

CIFAR-FS [10] This dataset is a modification of the CIFAR-100 dataset for few-shot learning. It contains all 100 classes from CIFAR-100, each with 600 images of 32×32 resolution. The standard split allocates 64 classes for training, 16 for validation, and 20 for testing.

FC100 [12] Also derived from CIFAR-100, the Fewshot-CIFAR100 (FC100) dataset is designed to test a model's ability to learn from classes with minimal conceptual overlap. It contains 100 classes grouped into 20 superclasses. The dataset is partitioned into 60 training classes (from 12 superclasses), 20 validation classes (from 5 superclasses), and 20 testing classes (from 5 superclasses), ensuring that the superclasses are disjoint across splits.

C.2. Fine-Grained and Cross-Domain Datasets

To evaluate the robustness and adaptability of our model on more specialized and challenging tasks, we employ the following fine-grained and cross-domain benchmarks.

CUB-200-2011 [15] The Caltech-UCSD Birds-200-2011 (CUB) dataset is a standard benchmark for fine-grained visual categorization. It consists of 11,788 images across 200 different bird species. For our few-shot experiments, we follow the standard split of 100 species for training, 50 for validation, and 50 for testing. The images are typically used with the provided bounding-box annotations to focus the model on the object of interest.

Cars [8] The Stanford Cars dataset is another key benchmark for fine-grained classification, containing 16,185 images of 196 classes of cars, categorized by make, model, and year. For cross-domain evaluation, the dataset is split into 130 training classes, 17 validation classes, and 49 testing classes. This dataset tests a model's ability to discern subtle differences between visually similar objects.

C.3. Experimental Protocol

Across all datasets, we adhere to the standard few-shot learning protocol. The training set of classes is used exclusively for the meta-training phase, where the model learns how to learn from a variety of tasks. The validation set is used for hyperparameter selection and tuning of our model. Finally, the testing set is held out and used only for the final evaluation of the model's performance on novel, unseen classes. This strict separation ensures a fair and unbiased assessment of the model's few-shot generalization capabilities.

Table 1. Detailed hyperparameter settings for all benchmark experiments. The 'Tasks per Epoch' column refers to the number of tasks sampled during meta-training, while 'Val Tasks' indicates the total number of tasks used for validation after each epoch.

Dataset	Setting	Backbone	Epochs	Train Tasks	Val Tasks	β	γ
MiniImageNet	5-way 1-shot	ResNet-50	200	1000	200	0.01	0.0005
	5-way 5-shot	ResNet-50	200	1000	200	0.05	0.001
TieredImageNet	5-way 1-shot	ResNet-50	200	3000	500	0.01	0.0002
	5-way 5-shot	ResNet-50	200	3000	500	0.015	0.0003
CIFAR-FS	5-way 1-shot	Mod. ResNet-18	200	5000	500	0.01	0.0002
	5-way 5-shot	Mod. ResNet-18	200	5000	500	0.05	0.0005
FC100	5-way 1-shot	Mod. ResNet-18	200	5000	500	0.005	0.0002
	5-way 5-shot	Mod. ResNet-18	200	5000	500	0.05	0.001
CUB-200-2011	5-way 1-shot	ResNet-50	200	2000	400	0.01	0.001
	5-way 5-shot	ResNet-50	200	2500	400	0.05	0.003

Appendix D: Implementation and Hyperparameter Details

This appendix provides a comprehensive overview of the implementation details for our experiments, including network architectures, training protocols, and hyperparameter settings. Our framework was implemented using PyTorch 2.4.1 and all experiments were conducted on a single NVIDIA GeForce RTX 4060 GPU.

D.1. Network Architectures

To ensure robust and meaningful feature extraction, the backbone network for each dataset was carefully chosen and initialized with weights pre-trained on a large-scale image dataset (e.g., ImageNet).

ResNet-50 for High-Resolution Datasets. For the high-resolution datasets, namely MiniImageNet, TieredImageNet, and CUB-200-2011, we employed a standard ResNet-50 backbone [6]. The deep architecture and powerful feature extraction capabilities of ResNet-50 provide rich representations that are essential for our NTRG module to effectively model complex inter-task relationships.

Modified ResNet-18 for Low-Resolution Datasets. For the CIFAR-FS and FC100 datasets, which are characterized by lower-resolution images (32×32), we utilized a modified ResNet-18 backbone to better suit the data constraints. Standard ResNet architectures are often optimized for larger ImageNet-scale images and can lose critical spatial information on smaller images due to their initial large-stride convolution and max-pooling layers. To mitigate this, we introduced the following specific modifications to the ResNet-18 architecture:

- We altered the first convolutional layer to use a kernel size of 3, a stride of 1, and a padding of 1.
- We completely removed the subsequent max-pooling layer.

These targeted modifications allow the network to preserve spatial information and enhance the granularity of learned features without imposing excessive computational overhead, which is crucial for adaptation in scenarios with limited data.

Backbone Generality. To further validate the generality of AdaMeta, we also experimented with other common backbones, including Conv-4, ResNet-12, and ViT. Our framework demonstrated robust performance across these architectures, indicating its wide applicability. The ResNet-50 and modified ResNet-18 backbones were selected for the main paper as they consistently yielded the most stable and effective results during our validation phase.

D.2. Training Protocol and Hyperparameters

General Training Setup. All models were trained using the Adam optimizer [7] with a learning rate of 1×10^{-4} and a batch size of 16. We employed an episodic training methodology to simulate the few-shot learning process. During meta-

training, each "task" or "episode" was constructed by sampling N classes (N-way) and K examples per class (K-shot) for the support set, plus a set of query examples from the same classes. The model was trained to minimize the classification loss on the query examples based on the information from the support set.

Hyperparameter Settings. The key hyperparameters in our AdaMeta framework include the task adaptation loss weight β and the relational consistency loss weight γ. These were specifically tuned for each dataset and setting using a grid search on the meta-validation set.

Our general tuning principle, observed during the grid search, was that the 1-shot setting typically requires a smaller relational consistency weight γ compared to the 5-shot setting. This is intuitive, as the 1-shot scenario has fewer samples per task, necessitating a more cautious application of the relational prior to avoid over-regularization.

To ensure full reproducibility, we report the specific values used for each benchmark experiment in Table 1. The number of training epochs and the number of tasks sampled for training and validation also varied per dataset to accommodate their different scales and complexities.

References

[1] Pierre Alquier. User-friendly introduction to pac-bayes bounds. *arXiv preprint arXiv:2110.11216*, 2021. 4

[2] Omar Besbes, Yonatan Gur, and Assaf Zeevi. Stochastic multi-armed-bandit problem with non-stationary rewards. *Advances in neural information processing systems*, 27, 2014. 4

[3] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, 02 2013. 4

[4] Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006. 4

[5] Fan RK Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997. 4

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 8

[7] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 8

[8] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013. 7

[9] Harold J Kushner and G George Yin. *Stochastic approximation and recursive algorithms and applications*. Springer, 2003. 2

[10] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10657–10665, 2019. 7

[11] David A McAllester. Some pac-bayesian theorems. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 230–234, 1998. 4

[12] Boris Oreshkin, Pau Rodríguez López, Alexandre Lacoste, et al. Tadam: Task dependent adaptive metric for improved few-shot learning. *Advances in neural information processing systems*, 31, 2018. 7

[13] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. *arXiv preprint:1803.00676*, 2018. 7

[14] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016. 7

[15] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 7