Interval Bound Interpolation for Few-shot Learning with Few Tasks

Shounak Datta [*1] **Sankha Subhra Mullick** [*1] **Anish Chakrabarty** [2] **Swagatam Das** [1]

Abstract

Few-shot learning aims to transfer the knowledge acquired from training on a diverse set of tasks to unseen tasks from the same task distribution with a limited amount of labeled data. The underlying requirement for effective few-shot generalization is to learn a good representation of the task manifold. This becomes more difficult when only a limited number of tasks are available for training. In such a few-task few-shot setting, it is beneficial to explicitly preserve the local neighborhoods from the task manifold and exploit this to generate artificial tasks for training. To this end, we introduce the notion of *interval bounds* from the provably robust training literature to few-shot learning. The interval bounds are used to characterize neighborhoods around the training tasks. These neighborhoods can then be preserved by minimizing the distance between a task and its respective bounds. We then use a novel strategy to artificially form new tasks for training by interpolating between the available tasks and their respective interval bounds. We apply our framework to both model-agnostic meta-learning as well as prototype-based metric-learning paradigms. The efficacy of our proposed approach is evident from the improved performance on several datasets from diverse domains compared to current methods.

1. Introduction

Few-shot learning problems deal with diverse tasks consisting of subsets of data drawn from the same underlying data manifold and associated labels. The joint distribution of data and corresponding labels which governs the sampling of such tasks is often called the task distribution (Finn et al., 2017; Yao et al., 2022). Consequently, few-shot learning methods attempt to leverage the knowledge acquired by training on a large pool of such tasks to easily generalize to unseen tasks from the same distribution, using only a few labeled examples. We hereafter refer to the support of the task distribution as the task manifold, which is distinct from but closely related to the data manifold associated with the data distribution. Since the unseen tasks are sampled from the same underlying manifold governing the task distribution, we should ideally learn a good representation of the task manifold by preserving the neighborhoods from the high-dimensional manifold in the lower-dimensional feature embedding (Tenenbaum et al., 2000; Roweis & Saul, 2000; Van der Maaten & Hinton, 2008). However, the labels associated with a task can define any arbitrary partitioning of the data. Therefore, we may preserve the neighborhood for a task by simply conserving the neighborhoods for the corresponding subset of the data manifold in the feature embedding learned by the few-shot learner. This facilitates effective few-shot generalization to new tasks as the layers which conserve the neighborhoods would likely require very little adaptation, and only the subsequent layers of the network will need to be updated. However, existing few-shot learning methods lack an explicit mechanism for achieving this. Further, real-world few-shot learning scenarios like rare disease detection may have a smaller number of training tasks required for effective learning due to various constraints such as data collection costs, privacy concerns,

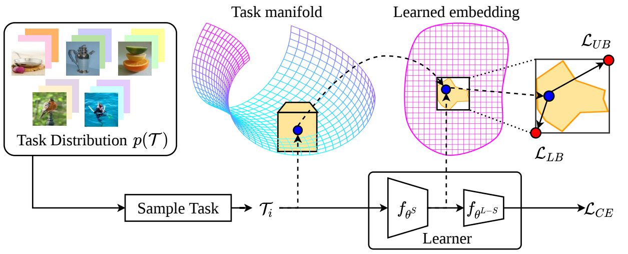


Figure 1. Illustration of the proposed interval bound propagation–aided few-shot learning setup (best viewed in color): We use interval arithmetic to define a small ϵ-neighborhood around a training task $\mathcal{T}_i$ sampled from the task distribution $p(\mathcal{T})$. IBP is then used to obtain the bounding box around the mapping of the said neighborhood in the embedding space f_{θ^S} given by the first S layers of the learner f_θ. While training the learner f_θ to minimize the classification loss $\mathcal{L}_{CE}$ on the query set $\mathcal{D}_i^q$, we additionally attempt to minimize the losses $\mathcal{L}_{LB}$ and $\mathcal{L}_{UB}$, forcing the ϵ-neighborhood to be compact in the embedding space as well.

[*] Equal contribution [1] Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata, India [2] Statistics and Mathematics Unit, Indian Statistical Institute, Kolkata, India. Correspondence to: Swagatam Das <swagatam.das@isical.ac.in>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

and/or data availability in newer domains (Yao et al., 2022). In such scenarios, few-shot learning methods are prone to overfit the training tasks, thus limiting the ability to generalization to unseen tasks. Therefore, in this work, we develop a strategy to explicitly constrain the feature embedding to preserve neighborhoods from the high-dimensional task manifold and to construct artificial tasks within these neighborhoods in the feature space, to improve the performance when a limited number of training tasks are available.

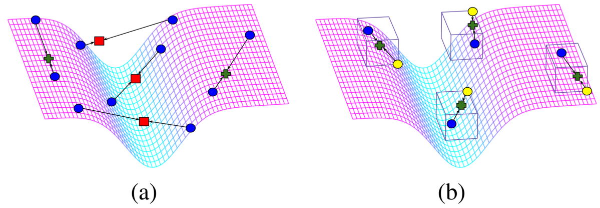


Figure 2. Interval bound–based task interpolation (best viewed in color): (a) Existing inter-task interpolation methods create new artificial tasks by combining pairs of original tasks (blue ball). However, depending on how flat the task-manifold embedding is at the layer where interpolation is performed, the artificial tasks may either be created close to the task-manifold (green cross) or away from the task-manifold (red box). (b) The proposed interval bound–based task interpolation creates artificial tasks by combining an original task with one of its interval bounds (yellow ball). Such artificial tasks are likely to be in the vicinity of the task manifold as the interval bounds are forced to be close to the task embedding by the losses $\mathcal{L}_{LB}$ and $\mathcal{L}_{UB}$.

The proposed approach relies on characterizing the neighborhoods from the high-dimensional task manifold and propagating them through the network with the intent to preserve the task neighborhood in the feature space. We achieve this by employing the concept of interval bounds from the provably robust training literature (Gowal et al., 2019; Morawiecki et al., 2020), i.e., the axis-aligned bounds for the activations in each layer, obtained using interval arithmetic (Sunaga, 1958). Concretely, as shown in Figure 1, we first define a small ϵ-neighborhood for each few-shot training task and then use Interval Bound Propagation (IBP; Gowal et al., 2019) to obtain the bounding box around the mapping of the corresponding neighborhood in the feature embedding space. We then explicitly attempt to preserve the ϵ-neighborhoods by minimizing the distance between a task and its respective interval bounds in addition to optimizing the few-shot classification objective. We further devise a mechanism to construct the artificial tasks by interpolating between a task and its corresponding IBP bounds. It is important to notice that this setup is distinct from provably robust training for few-shot learning in that we do not attempt to minimize (or calculate for that matter) the worst-case classification loss.

Various methods have been proposed to mitigate the few-task few-shot problem using approaches such as explicit regularization (Jamal & Qi, 2019; Yin et al., 2019), intra-task augmentation (Lee et al., 2020; Ni et al., 2021; Yao et al., 2021), and inter-task interpolation to construct new artificial tasks (Yao et al., 2022). While inter-task interpolation has been shown to be the most effective among these existing approaches, it suffers from the limitation that the artificially created tasks may be generated away from the task manifold depending on the curvature of the feature embedding space, as there is no natural way to select pairs of task which are close to each other on the manifold (Figure 2(a)). In contrast, the interval bounds obtained using IBP are likely to be close to the original task embedding as we explicitly minimize the distance between a task and its interval bounds. Thus, using them for interpolation will likely keep the generated tasks close to the manifold (Figure 2(b)).

In essence, the key contributions made in this article advance the existing literature in the following ways:

1. In Section 4.1, we present, for the first time, a novel method to synergize few-shot learning with interval bound propagation (Gowal et al., 2019) to explicitly lend the ability to preserve task neighborhoods in the feature embedding space of the few-shot learner.
2. In Section 4.3, we propose the interval bound–based task interpolation technique which can create new tasks (as opposed to augmenting each individual task (Lee et al., 2020; Ni et al., 2021; Yao et al., 2021)) by interpolating between a task sampled from the task distribution and its interval bounds.
3. Unlike existing inter-task interpolation methods that require paired tasks for interpolation (Yao et al., 2022), our framework generates new tasks from only a single task. This allows the proposed framework to be seamlessly integrated with few-shot learning paradigms.

In Section 5, we empirically demonstrate the effectiveness of our proposed approach, in comparison to the recent prior methods while making concluding remarks in Section 6.

2. Related works

Few-shot learning aims to generalize to new tasks using only a few examples (Wang et al., 2020) through three major strategies. First, one can augment the tasks at the data level (Hariharan & Girshick, 2017). Second, the hypothesis space can be constrained at the model level (Snell et al., 2017). Third, the hypothesis search strategy at the algorithm level can be improved (Finn et al., 2017). The problem of few-task learning can be even more difficult when training tasks are scarce in a few-task scenario. To train on few-task datasets, some works directly impose regularization on the few-shot learner (Jamal & Qi, 2019; Yin et al., 2019). Another line of work performs data augmentation on individual tasks (Lee et al., 2020; Ni et al., 2021; Yao et al., 2021). Finally, a third direction is to employ inter-task interpolation to mitigate task scarcity (Yao et al., 2022). Our approach is similar to the third category in that we create new artificial tasks directly. But, we also differ from all of the methods mentioned above in that we neither undertake intra-task augmentation nor inter-task interpolation. Moreover, our novel task augmentation strategy can work in conjunction with both algorithm-level meta-learning as well as model-level metric-learning methods.

Our technique relies on preserving the neighborhoods of the task manifold in the learned feature embedding space. This, in spirit, connects with the classical problem of manifold learning. Traditional methods like ISOMAP (Tenenbaum et al., 2000), LLE (Roweis & Saul, 2000), t-SNE (Van der Maaten & Hinton, 2008), etc. aim to represent high-dimensional data in lower-dimensional space while preserving the local neighborhoods through manifold learning (Abukmeil et al., 2021). Recent deep manifold learning methods mostly employ deep belief network (Lee et al., 2009), variational auto-encoders (Connor et al., 2021; Kumar & Poole, 2020), flow-based approaches (Brehmer & Cranmer, 2020; Caterini et al., 2021), etc. Similarly, here we repurpose IBP (Gowal et al., 2019) to define ϵ-neighborhoods for few-shot learning tasks and constrain the learned feature embedding to preserve the said neighborhoods. IBP was originally proposed to build robust neural networks. A way to build robust neural networks is to find a differentiable upper bound on the verifiable violation of specifications. Such upper bounds can then be directly optimized alongside the original loss (Mirman et al., 2018; Raghunathan et al., 2018; Wong et al., 2018). IBP (Gowal et al., 2019) follows this direction by explicitly minimizing the worst-case loss inside the ϵ-neighborhood of an input for an arbitrary network with some architectural constraints. However, in our work, instead of building robust networks, we repurpose IBP to characterize the ϵ-neighborhood to learn better representation such that the generalization to new tasks by a few-shot learner becomes easier. Moreover, the bounds of the ϵ-neighborhood obtained through IBP give us a direct way to construct new artificial tasks when the number of available tasks is scarce.

3. Preliminaries

In a few-shot learning problem, we deal with tasks $\mathcal{T}_i \sim p(\mathcal{T})$. Each task $\mathcal{T}_i$ is associated with a dataset $\mathcal{D}_i = (X_i, Y_i)$, that we further subdivide into a support set $\mathcal{D}_i^s = (X_i^s, Y_i^s) = \{(\mathbf{x}_{i,r}^s, y_{i,r}^s)\}_{r=1}^{N_s}$ and a query set $\mathcal{D}_i^q = (X_i^q, Y_i^q) = \{(\mathbf{x}_{i,r}^q, y_{i,r}^q)\}_{r=1}^{N_q}$. Given a learning model f_θ, where θ denotes the model parameters, few-shot learning algorithms attempt to learn θ to minimize the loss on the query set $\mathcal{D}_i^q$ for each of the sampled tasks using the data-label pairs from the corresponding support set $\mathcal{D}_i^s$. Thereafter, the trained model f_θ and the support set $\mathcal{D}_j^s$ for new tasks $\mathcal{T}_j$ can be used to perform inference on the corresponding query set $\mathcal{D}_j^q$. In the following, we discuss gradient-based meta-learning while the prototype-based metric-learning is detailed in Appendix A.

Gradient-based meta-learning: In gradient-based meta-learning, the aim is to learn initial parameters θ^* such that a typically small number of gradient update steps using the data-label pairs in the support set $\mathcal{D}_i^s$ results in a model f_{ϕ_i} that performs well on the query set of task $\mathcal{T}_i$. During the meta-training stage, first, a base learner is trained on multiple support sets $\mathcal{D}_i^s$, and the performance of the resulting models f_{ϕ_i} is evaluated on the corresponding query sets $\mathcal{D}_i^q$. The meta-learner parameters θ are then updated so that the base learner's expected loss on query sets is minimized. In the meta-testing stage, the final meta-trained model f_{θ^*} is fine-tuned on the support set $\mathcal{D}_j^s$ for the given test task $\mathcal{T}_j$ to obtain the adapted model f_{ϕ_j} that can then be used for inference on the corresponding query set $\mathcal{D}_j^q$. Considering Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017) as an example, the bi-level optimization of the gradient-based meta-learning is formulated as:

$$\theta^* \leftarrow \arg\min_{\theta} \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})}[\mathcal{L}(f_{\phi_i}; \mathcal{D}_i^q)], \tag{1}$$

where $\phi_i = \theta - \eta_0 \nabla_\theta \mathcal{L}(f_\theta; \mathcal{D}_i^s)$ while η_0 denotes the inner-loop learning rate used by the base learner to train on $\mathcal{D}_i^s$ for task $\mathcal{T}_i$, and $\mathcal{L}$ is the loss function, which is usually the cross-entropy loss for classification problems:

$$\mathcal{L}_{CE} = \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})}[-\sum\nolimits_r \log p(y_{i,r}^q | \mathbf{x}_{i,r}^q, f_{\phi_i})]. \tag{2}$$

Remark 3.1. For effective few-shot generalization to new tasks, gradient-based meta-learning methods (or prototype-based metric-learning methods) need to learn a good representation of the task manifold. Since the unseen tasks are sampled from the same task distribution supported by an underlying task manifold, this can ideally be achieved by preserving the neighborhoods from the high-dimensional manifold in the lower-dimensional feature embedding, similar to long-standing manifold learning methods (Abukmeil et al., 2021). However, the labels for a task may be constructed to define any arbitrary partition of the data, depending on the application domain. Therefore, it may be futile to retain information about the partitioning imposed by past tasks. One can instead choose to conserve the neighborhoods in the subset of the data manifold corresponding to a given task. This will encourage few-shot generalization as the layers of the network which preserve the neighborhoods are likely to require little update while only the subsequent layers need to be tuned.

Therefore, in the following section, we start by discussing IBP (Gowal et al., 2019) and show how it can be repurposed to define a neighborhood around the samples for a given task in the few-shot learning setup.

4. Proposed Method

In the following subsections, we describe the notion of an ϵ-neighborhood for a training task $\mathcal{T}_i$ using IBP and show how preserving that can aid a few-shot learner f_θ to learn an efficient feature embedding, especially in few-task case.

4.1. Few-shot learning with interval bounds

Let us consider a neural network f_θ consisting of a sequence of transformations $h_l, (l \in \{1, 2, \cdots, L\})$ for each of its L layers. We start from an initial input $\mathbf{z}_0 = \mathbf{x}$ to the network along with lower bound $\underline{\mathbf{z}}_0(\epsilon) = \mathbf{x} - \mathbf{1}\epsilon$ and upper bound $\overline{\mathbf{z}}_0(\epsilon) = \mathbf{x} + \mathbf{1}\epsilon$ for an ϵ-neighborhood around the input $\mathbf{x}$. In each of the subsequent layers $l \in \{1, 2, \cdots, L\}$ of the network, we get an activation $\mathbf{z}_l = h_l(\mathbf{z}_{l-1})$. IBP uses interval arithmetic to obtain the corresponding axis-aligned bounds of the form $\underline{\mathbf{z}}_l(\epsilon) \leq \mathbf{z}_l \leq \overline{\mathbf{z}}_l(\epsilon)$ on the activations for the l-th layer. Given the specific differentiable transformation h_l, interval arithmetic yields corresponding differentiable lower and upper bound transformations $\underline{\mathbf{z}}_l(\epsilon) = \underline{h}_l(\underline{\mathbf{z}}_{l-1}(\epsilon), \overline{\mathbf{z}}_{l-1}(\epsilon))$, and $\overline{\mathbf{z}}_l(\epsilon) = \overline{h}_l(\underline{\mathbf{z}}_{l-1}(\epsilon), \overline{\mathbf{z}}_{l-1}(\epsilon))$ (see Appendix C). This ensures that each of the coordinates $\underline{z}_{l,c}(\epsilon)$ and $\overline{z}_{l,c}(\epsilon)$ of $\underline{\mathbf{z}}_l(\epsilon)$ and $\overline{\mathbf{z}}_l(\epsilon)$ respectively, satisfies:

$$\underline{z}_{l,c}(\epsilon) = \min_{\underline{\mathbf{z}}_{l-1}(\epsilon) \leq \mathbf{z}_{l-1} \leq \overline{\mathbf{z}}_{l-1}(\epsilon)} \mathbf{e}_c^{\mathrm{T}} h_l(\mathbf{z}_{l-1}) \text{ and} \tag{3}$$

$$\overline{z}_{l,c}(\epsilon) = \max_{\underline{\mathbf{z}}_{l-1}(\epsilon) \leq \mathbf{z}_{l-1} \leq \overline{\mathbf{z}}_{l-1}(\epsilon)} \mathbf{e}_c^{\mathrm{T}} h_l(\mathbf{z}_{l-1}), \tag{4}$$

where $\mathbf{e}_c$ is the standard c-th basis vector. For multiple layers, such as f_{θ^S} having the first S layers of f_θ, the individual transformations $\underline{h}_l$ and $\overline{h}_l$ for $l \in \{1, 2, \cdots, S\}$ can be composed to obtain the corresponding functions $\underline{f}_{\theta^S}$ and $\overline{f}_{\theta^S}$, such that $\underline{\mathbf{z}}_S(\epsilon) = \underline{f}_{\theta^S}(\mathbf{z}_0, \epsilon)$, and $\overline{\mathbf{z}}_S(\epsilon) = \overline{f}_{\theta^S}(\mathbf{z}_0, \epsilon)$.

Now consider the network $f_\theta = f_{\theta^{L-S}} \circ f_{\theta^S}$ where S ($\leq L$) is a user-specified layer number that demarcates the boundary between the portion f_{θ^S} of the model that focuses on feature representation and the subsequent portion $f_{\theta^{L-S}}$ responsible for the classification. Given training task $\mathcal{T}_i$, the Euclidean distances between the embedding $f_{\theta^S}(\mathbf{x}^q_{i,r})$ for the query instances and their respective interval bounds $\underline{f}_{\theta^S}(\mathbf{x}^q_{i,r}, \epsilon)$ and $\overline{f}_{\theta^S}(\mathbf{x}^q_{i,r}, \epsilon)$ is a measure of how well the ϵ-neighborhood is preserved in the learned feature embedding:

$$\mathcal{L}_{LB} = \frac{1}{N_q} \sum\nolimits_{r=1}^{N_q} ||f_{\theta^S}(\mathbf{x}^q_{i,r}) - \underline{f}_{\theta^S}(\mathbf{x}^q_{i,r}, \epsilon)||_2^2 \text{ and} \tag{5}$$

$$\mathcal{L}_{UB} = \frac{1}{N_q} \sum\nolimits_{r=1}^{N_q} ||f_{\theta^S}(\mathbf{x}^q_{i,r}) - \overline{f}_{\theta^S}(\mathbf{x}^q_{i,r}, \epsilon)||_2^2. \tag{6}$$

To ensure that the small ϵ-neighborhoods get mapped to small interval bounds by the feature embedding f_{θ^S}, we can minimize the losses $\mathcal{L}_{LB}$ and $\mathcal{L}_{UB}$ in addition to the classification loss $\mathcal{L}_{CE}$ in (2). Notice that the losses $\mathcal{L}_{LB}$ and $\mathcal{L}_{UB}$ are never used for the support instances $\mathbf{x}^s_{i,r}$.

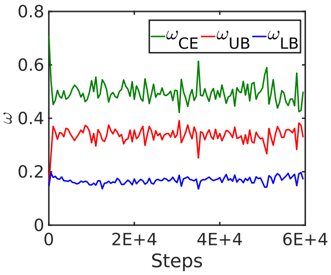


Figure 3. Dynamic weights for MAML+IBP on miniImageNet when γ is set to 1 for ease of visualisation.

4.2. Dynamic loss weighting

Attempting to minimize a naïve sum of the three losses can cause some issues. For example, weighing the classification loss $\mathcal{L}_{CE}$ too high essentially reduces the proposed method to vanilla few-shot learning. On the contrary, assigning very high weights to the interval losses $\mathcal{L}_{LB}$ and/or $\mathcal{L}_{UB}$ may diminish learnability as the preservation of ϵ-neighborhoods gets precedence over classification performance. Moreover, such static weighting approaches are not capable of adapting to (and consequently mitigating) situations where one of the losses comes to unduly dominate the others. Thus, we minimize a convex weighted sum $\mathcal{L}$ of the three losses:

$$\mathcal{L}(t) = \sum\nolimits_{e \in \{CE, LB, UB\}} w_e \mathcal{L}_e(t), \tag{7}$$

where t is the current training step and $w_e(t)$ is the weight for the corresponding loss $\mathcal{L}_e$, $e \in \{CE, LB, UB\}$ at the t-th training step, which is dynamically calculated based on a softmax across the current values of the three losses:

$$w_e(t) = \frac{\exp(\mathcal{L}_e(t)/\gamma)}{\sum_{e' \in \{CE, LB, UB\}} \exp(\mathcal{L}_{e'}(t)/\gamma)}. \tag{8}$$

The hyperparameter γ controls the relative importance of the losses. If any of the losses become too large, the dynamic weighing scheme strives to restore balance by assigning very high weightage to the concerned loss, thus prioritizing its minimization over that of the other losses. The changes in the dynamic weights over training steps for IBP-aided MAML (hereafter called MAML+IBP) using "4-CONV" network (Vinyals et al., 2016) on the miniImageNet dataset (Vinyals et al., 2016) is illustrated in Figure 3. We can observe that while there is an explicit ordering to the magnitude of the weights (and, therefore, the corresponding losses) throughout the entire training run, the weights can adapt to changes in loss values to maintain the status quo among the different losses.

Motivating results: In Table 1, we demonstrate the effect of employing IBP-aided training for MAML using the "4-CONV" network. Apart from vanilla MAML, we consider two other baselines, (1) MAML+GL that uses the distance between the original query set and its perturbed (by additive Gaussian noise) version as an extra loss, and (2)

Table 1. Accuracy and intra-task compactness of MAML+IBP.

Metric	Algorithm	miniImageNet	tieredImageNet
5-way 1-shot Accuracy	MAML (Finn et al., 2017)	48.70±1.75%	51.67±1.81%
	MAML+GL	48.70±0.97%	51.90±0.98%
	MAML+ULBL	49.43±0.90%	51.67±0.91%
	MAML+IBP (ours)	**50.76±0.83%**	**54.36±0.80%**
1-NN Distance	MAML (Finn et al., 2017)	0.97±0.02	0.98±0.02
	MAML+GL	0.96±0.02	0.98±0.02
	MAML+ULBL	0.94±0.02	0.97±0.02
	MAML+IBP (ours)	**0.90±0.02**	**0.96±0.02**

MAML+ULBL that considers the distance between the upper and lower interval bounds as an additional loss (Morawiecki et al., 2020) (further details in Appendix F.3). We see that MAML+IBP achieves higher 5-way 1-shot classification accuracy than the five contenders on the miniImageNet and tieredImageNet (Ren et al., 2018) datasets (supporting the conjecture in Remark 3.1). Moreover, we also illustrate that the feature embedding learned by IBP-aided training exhibits better intra-task compactness in terms of the mean Euclidean distances from the nearest neighbor in the same class for 100 query instances from 600 tasks in the feature space characterized by f_{θ^S}. Recent works (Ni et al., 2021; Yao et al., 2022) have shown that augmenting the training data with artificial tasks can improve performance in domains with a scarcity of tasks. Thus, while IBP-aided training improves the performance of vanilla MAML (as well as other baselines, see Appendix F.3), we are particularly interested in the added advantage that it lends by facilitating the generation of artificial tasks within the neighborhoods defined by the interval bounds.

4.3. Interval bound–based task interpolation

Since minimizing the additional losses $\mathcal{L}_{LB}$ and $\mathcal{L}_{UB}$ is expected to ensure that the ϵ-neighborhood around a task is mapped to a small interval in the feature embedding space, artificial tasks formed within such intervals are naturally expected to be close to the task manifold. Therefore, we create additional artificial tasks by interpolating between an original task and its corresponding interval bounds (i.e., either the upper or the lower interval bound). In other words, for a training task $\mathcal{T}_i$, a corresponding artificial task $\mathcal{T}_i'$ is characterized by a support set $\mathcal{D}_i^{s'} = \{(\mathbf{H}_{i,r}^{s'}, \mathbf{y}_{i,r}^{s})\}_{r=1}^{N_s}$ in the embedding space. The artificial support instances $\mathbf{H}_{i,r}^{s'}$ are obtained as a sum of $(1-\lambda_k)f_{\theta^S}(\mathbf{x}_{i,r}^s)$, $(1-\nu_k)\lambda_k\underline{f}_{\theta^S}(\mathbf{x}_{i,r}^s,\epsilon)$, and $\nu_k\lambda_k\overline{f}_{\theta^S}(\mathbf{x}_{i,r}^s,\epsilon)$, where k denotes the class to which $\mathbf{x}_{i,r}^s$ belongs, $\lambda_k \in [0,1]$ is sampled from a Beta distribution $Beta(\alpha,\beta)$, and the random choice of $\nu_k \in \{0,1\}$ dictates which of the bounds is chosen randomly for each class. The labels $\mathbf{y}_{i,r}^s$ for the artificial task remain identical to that of the original task. The query set $\mathcal{D}_i^{q'}$ for the artificial task is also constructed analogously. We then minimize the mean of the additional classification loss $\mathcal{L}_{CE}'$ for the artificial task $\mathcal{T}_i'$ and the classification loss $\mathcal{L}_{CE}$ for the original task $\mathcal{T}_i$ for query instances (also the support instances in case of meta-learning). As a reminder, the losses $\mathcal{L}_{LB}$ and $\mathcal{L}_{UB}$ are also additionally minimized for the query instances. The complete IBP-based task interpolation or Interval Bound Interpolation (IBI) training setup is illustrated in Figure 4. Since IBI does not play any part during the testing phase, the testing recipe remains identical to that of vanilla few-shot learning. The pseudocode of MAML+IBI (and the IBI variant of ProtoNet) can be found in Appendix B.

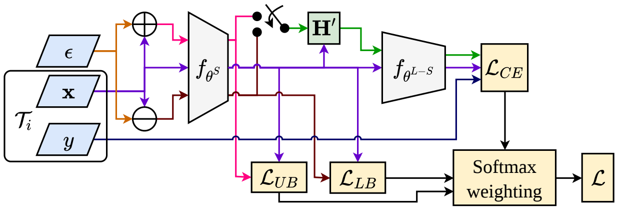


Figure 4. Training of IBI (best viewed in color): For each query data-label pair $(\mathbf{x}, y)$ in a given training task $\mathcal{T}_i$, we start by defining a ϵ-neighborhood $[\mathbf{x}-\mathbf{1}\epsilon, \mathbf{x}+\mathbf{1}\epsilon]$ around $\mathbf{x}$. The bounding box $[\underline{f}_{\theta^s}(\mathbf{x},\epsilon), \overline{f}_{\theta^s}(\mathbf{x},\epsilon)]$ around the embedding $f_{\theta^S}(\mathbf{x})$ after the first S layers of the learner is found using IBP. In addition to the classification loss $\mathcal{L}_{CE}$, we also minimize the losses $\mathcal{L}_{LB}$ and $\mathcal{L}_{UB}$ which respectively measure the distances of $f_{\theta^S}(\mathbf{x})$ to $\underline{f}_{\theta^s}(\mathbf{x},\epsilon)$ and $\overline{f}_{\theta^s}(\mathbf{x},\epsilon)$. A softmax across the three loss values is used to dynamically calculate the convex weights for the losses, so as to prioritize the minimization of the dominant loss(es) at any given training step. For IBP-based interpolation, artificial tasks $\mathcal{T}_i'$ are created with instances $\mathbf{H}'$ formed by interpolating both the support and query instances with their corresponding lower or upper bounds. The mean of the classification loss $\mathcal{L}_{CE}$ for the $\mathcal{T}_i$ and the corresponding extra loss $\mathcal{L}_{CE}'$ for $\mathcal{T}_i'$ is minimized.

Theoretical analysis: The data X_i $(i = 1, 2, \cdots, N)$ for tasks $\mathcal{T}_i$ can be thought of as i.i.d. observations from a marginal distribution $\mathbb{P}_X$ defined on a compact subset $\mathcal{X}$ of $\mathbb{R}^d$ $(d \geq 1)$, paired with corresponding Y_i drawn from the marginal distribution $\mathbb{P}_Y$. The map f_{θ^S} is bestowed with the task of producing a lower-dimensional representation of the input X. Let us denote the embedding space by $\mathcal{H} \subseteq \mathbb{R}^\kappa$, given that $\kappa \leq d$. The spaces $\mathcal{X}$ and $\mathcal{H}$ are endowed with l_2 norm for simplicity and conformity to our convention. One may observe that $f_{\theta^S} = h_1 \circ h_2 \circ \cdots \circ h_S$, where in general $h_l(\mathbf{z}) = \sigma(A_l\mathbf{z} + \mathbf{b}_l)$ given that $A_l \in \mathbb{R}^{d_{l+1}\times d_l}$ and $\mathbf{b}_l \in \mathbb{R}^{d_{l+1}}$, $l = 1, \cdots, S$. The function σ denotes the activation (such as ReLU), applied component-wise. Evidently, in our notation $d_1 = d$ and $d_{S+1} = \kappa$. With this setup, we proceed to the theoretical analysis of our approach. Please find the detailed proofs in Appendix C.

Definition 4.1 (Perturbation)**.** Given any $\mathbf{x}_1 \in \mathcal{X}$, an ε-perturbation corresponding to $\mathbf{x}_1$ is the set of points $\mathbf{x}_1(\varepsilon) \subset \mathcal{X}$ such that $\|\mathbf{x}_1 - \mathbf{x}_2\| = \varepsilon, \forall \mathbf{x}_2 \in \mathbf{x}_1(\varepsilon); \varepsilon > 0$.

For the particular choice of the l_2 norm, Definition 4.1 characterizes ε-perturbation as a hollow ball of radius $\varepsilon = \epsilon\sqrt{d}$ around a given point.

Lemma 4.2 (Lipschitz networks ensure bounded IBP)**.** *Let*

$\overline{\mathbf{x}}$ and $\underline{\mathbf{x}}$ be ε-perturbations of $\mathbf{x} \sim \mathbb{P}_X$ for an $\varepsilon > 0$ (i.e. $\overline{\mathbf{x}}, \underline{\mathbf{x}} \in \mathbf{x}(\varepsilon)$). Given that the activation σ is Lipschitz continuous (such as ReLU) with constant $c_\sigma > 0$, there exists a constant $D = D(c_\sigma; A_1, A_2, \cdots, A_S; \varepsilon)$ such that $\underline{f}_{\theta^S}(\mathbf{x}, \varepsilon)$ and $\overline{f}_{\theta^S}(\mathbf{x}, \varepsilon))$ will at most be an $\hat{\varepsilon}$-perturbed version of $f_{\theta^S}(\mathbf{x})$, where $\hat{\varepsilon} = \varepsilon D$.

The minimization objective function of IBI can be rephrased as $\mathcal{L} = \mathcal{L}_{CE} + \omega_1 \mathcal{L}_{LB} + \omega_2 \mathcal{L}_{UB}$, where $\omega_1, \omega_2 \geq 0$ are Lagrangian multipliers. The forthcoming result, however, relies on the constrained formulation of the objective, given as $\min\{\mathcal{L}_{CE}\}$ subject to $\mathcal{L}_{LB} \leq t_1$ and $\mathcal{L}_{UB} \leq t_2$, where $t_1, t_2 \geq 0$. This is motivated by the fact that the constrained formulation yields solutions upper bounding the ones obtained using its Lagrangian counterpart (Boyd & Vandenberghe, 2004). Lemma 4.2 implies that the losses $\mathcal{L}_{UB}$ and $\mathcal{L}_{LB}$ appearing in the constraints can always be made arbitrarily small, depending upon ε. As such, in the constrained regime, the remaining problem is to show that the multi-task sample classification loss can indeed be dealt with.

Theorem 4.3 (Generalization bound)**.** *Let $\tilde{\mathbb{P}}$ be the joint distribution of $(f_{\theta^S}(X), Y)$, supported on $\mathcal{H} \times \mathbb{R}$. In the multi-task regime, let I denote the set of tasks, each consisting of N samples. Define $\hat{\mathcal{R}}(N, |I|) = \mathbb{E}_{\mathcal{T}_i \sim \hat{p}(\mathcal{T})} \mathbb{E}_{(X_j, Y_j) \sim \hat{p}(\mathcal{T}_i)} [\mathcal{L}_{CE}(f_{\theta^{L-S}}(\mathbf{H}_j^*), Y_j)]$ and $\mathcal{R} = \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} \mathbb{E}_{(X_j, Y_j) \sim \mathcal{T}_i} [\mathcal{L}_{CE}(f_{\theta^{L-S}}(f_{\theta^S}(X_j)), Y_j)]$. For a bounded loss function $\mathcal{L}_{CE} : \mathbb{R} \times \mathbb{R} \to [0, a] (a \geq 0)$, if the neural network-induced map $f_{\theta^{L-S}}$ is such that $\left|\nabla f_{\theta^{L-S}}(\cdot)\right| < \infty$, we ensure:*

$$\left|\hat{\mathcal{R}}(N, |I|) - \mathcal{R}\right| - \tilde{\lambda} \precsim 2^{L-S+1} \sqrt{2 \log(2\kappa + 2)} \times \left[\left(\frac{1}{\sqrt{N}} + \frac{1}{\sqrt{|I|}} \right) + \sqrt{\frac{\log(2 \left|I\right| / \delta)}{N}} + \sqrt{\frac{\log(2/\delta)}{|I|}} \right]$$

holds with probability at least $1 - \delta$, where $\tilde{\lambda} = \tilde{\lambda}(\hat{\varepsilon}, \lambda)$.

Theorem 4.3 suggests that the excess risk (absolute difference between the population risk and the empirical counterpart obtained by our method) behaves approximately similar to a linear function of the perturbation parameter $\varepsilon > 0$. The rate of convergence we obtain also turns out to be sharp (compared to (Yao et al., 2022)) as the RHS vanishes when both N and $|I| \to \infty$, such that $\log(|I|)/N = o(1)$. Another key highlight of Theorem 4.3 is that it circumvents the curse of dimensionality, often present in classical generalization bounds, by incorporating the dimension of the embedding space (κ) in the constants instead.

5. Experiments

The experiments are conducted on few-task few-shot image classification datasets, viz. a subset of the miniImageNet dataset called miniImageNet-S (Yao et al., 2022), and two medical images datasets namely DermNet-S (Yao et al., 2022), and ISIC (Codella et al., 2018; Li et al., 2020). We begin our experiments with a few analyses and ablations to better understand the properties of our proposed method. We then empirically demonstrate the effectiveness of our proposed IBI method on the gradient-based meta-learning method MAML (Finn et al., 2017) as well as the prototype-based metric-learner ProtoNet (Snell et al., 2017) to show that IBI can be seamlessly integrated with multiple few-shot learning paradigms. For our experiments, we employ the commonly used "4-CONV" network (Vinyals et al., 2016) as well as the larger ResNet-12 network (Lee et al., 2019) to demonstrate the scalability of the proposed method (further details on scalability in Appendix E). We perform 5-way 1-shot and 5-way 5-shot classification on all the above datasets (except ISIC where we use 2-way classification problems, similar to (Yao et al., 2021), due to the lack of sufficient training classes). Further discussion on the datasets and implementation details of IBI along with the choice of hyperparameters can be found in the Appendix while the code is available at `https://github.com/SankhaSubhra/maml-ibp-ibi`.

Ablation studies on task interpolation: We undertake an ablation study to highlight the importance of generating artificial tasks using IBP bound–based interpolation by comparing IBI with (1) inter-task interpolation on images, (2) inter-task interpolation in the feature embedding learned by f_θ^S, (3) Worst-Case Loss (WCL) on the ϵ-neighborhood (Gowal et al., 2019) along with IBP losses, (4) inter-task interpolation while minimizing ULBL (Morawiecki et al., 2020), (5) Gaussian noise–based perturbation (GA) in the image space with IBP losses, (6) Gaussian noise–based perturbation in the feature embedding space f_{θ^S} with IBP losses, (7) MLTI (Yao et al., 2022), which performs MixUp (Zhang et al., 2017) at randomly chosen layers of the learner, and (8) IBP bound–based interpolation without minimizing the $\mathcal{L}_{UB}$ and $\mathcal{L}_{LB}$ while only optimizing $\mathcal{L}_{CE}$ (more results in Appendix F.3). We perform the ablation study on 5-way 1-shot classification with the "4-CONV" network on miniImageNet-S, ISIC, and DermNet-S. From Table 2, we observe that IBI performs best in all cases. Moreover, inter-class interpolation at the same fixed layer S as IBI and at randomly selected task-specific layers in MLTI shows worse performance, demonstrating the superiority of the proposed interval bound–based interpolation mechanism. Further, it is interesting to observe that IBI, when performed without minimizing the $\mathcal{L}_{UB}$ and $\mathcal{L}_{LB}$, performs the worst. This behavior is not unexpected as the neighborhoods are no longer guaranteed to be preserved by the learned embedding in this case, thus potentially resulting in the generation of out-of-manifold artificial tasks.

To further check whether the tasks generated by IBI indeed follow the distribution, we undertake a comparison based on

Table 2. Ablation on task interpolation strategies in terms of mean Accuracy and average median distance between original and interpolated tasks over 600 tasks on miniImageNet-S (mIS), ISIC and DermNet-S (DS).

Algorithm	Accuracy			Average median distance		
	mIS	ISIC	DS	mIS	ISIC	DS
MAML+Inter-task interpolation in image space	40.90%	55.25%	48.30%	N/A	N/A	N/A
MAML+Inter-task interpolation after $f_{\theta S}$	41.00%	61.33%	47.43%	3.08	1.23	2.99
MAML+IBP+WCL	41.56%	64.75%	48.90%	NA	NA	NA
MAML+ULBL+Inter-task interpolation after $f_{\theta S}$	40.37%	64.91%	48.23%	3.10	0.97	2.83
MAML+IBP+GA (Image Space)	41.83%	62.67%	48.83%	NA	NA	NA
MAML+IBP+GA (after $f_{\theta S}$)	41.66%	63.75%	47.60%	NA	NA	NA
MAML+MLTI (Yao et al., 2022)	41.58%	61.79%	48.03%	3.24	1.36	3.05
MAML+IBI without $\mathcal{L}_{UB}$ and $\mathcal{L}_{LB}$ losses	35.26%	48.94%	41.30%	N/A	N/A	N/A
MAML+IBI (ours)	**42.20%**	**68.58%**	**49.13%**	**2.74**	**0.60**	**2.65**

The Average median distance is calculated with features after the third block for all cases.

Table 3. Average loss weights for MAML+IBP and MAML+IBI, comparison of the static weighting and dynamic weighting versions, and transferability of static weights across variants.

	MAML+IBP	MAML+IBI
Average of dynamic loss weights calculated for IBP and IBI.		
w_{CE}	0.8600	0.8658
w_{UB}	0.1369	0.1314
w_{LB}	0.0029	0.0027
Accuracy of algorithms with different weight choices.		
Dynamic weighting	**41.30±0.79%**	**42.20±0.82%**
Static average weights for MAML+IBP	40.55±0.81%	N/A
Static average weights for MAML+IBI	N/A	40.72±0.79%

the similarity of the artificial tasks with the corresponding original tasks. Concretely, we define the distance between a task and its artificial counterpart as the median of the pairwise distances between the corresponding data instances in the two tasks. If an artificial task is created by combining two tasks, *a la* MLTI (Yao et al., 2022), we consider the smaller of the two median distances. We observe from Table 2, that the average median distance over 600 tasks is smaller for the proposed method compared to MLTI, as well as inter-task interpolation in the feature embedding learned by f_θ^S. This indicates that the tasks generated by IBI are more likely to lie close to the original task distribution.

Importance of dynamic loss weighting: To validate the usefulness of softmax-based dynamic weighting of the three losses for both IBP and IBI, we first find the average weights for each loss in a dynamic weight run and then plug in the respective values as static weights for new runs. All experiments in Table 3 are conducted on the miniImageNet-S dataset. From the upper half of Table 3, we can see that the three average weights are always distinct with a definite trend in that $\mathcal{L}_{CE}$ gets maximum importance followed by $\mathcal{L}_{UB}$ while $\mathcal{L}_{LB}$ contributes very little to the total loss $\mathcal{L}$. This may be due to the particular "4-CONV" architecture used in this study which employs ReLU activations, thus implicitly limiting the spread of the lower bound (Gowal et al., 2019). Further, the average weights of IBP and IBI are similar for a particular learner highlighting their commonalities, while they are distinct over different learners stressing their learner-dependent behavior. Further, in the lower half of Table 3, we explore the effect of using static weights as well as the transferability of the loss weights across learners. In all cases, the softmax-based dynamic weighting outperforms static weighting, thus demonstrating the importance of dynamic weighting.

Results on few-task few-shot classification problems: For evaluating the few-shot classification performance of IBI in few-task situations for MAML we compare against (1) regularization-based meta-learning methods TAML (Jamal & Qi, 2019), Meta-Reg (Yin et al., 2019), and Meta-Dropout (Lee et al., 2020) (2) recent task augmentation techniques Meta-Interpolation (Lee et al., 2022), TU, and ATU (Wu et al., 2022). We also compare IBI against data augmentation–based methods like MetaMix (Yao et al., 2021), Meta-Maxup (Ni et al., 2021), and MLTI (Yao et al., 2022) for both MAML and ProtoNet. The results in Table 4 show that in keeping with the observation in Table 1, IBP without task interpolation can improve upon the corresponding baselines. Incorporating IBP-based task interpolation in IBI generally improves the results even further. Overall, we observe that both IBP and IBI outperform the competitors in case of DermNet-S and ISIC datasets. Even though, IBI achieves slightly lower accuracy on miniImageNet-S compared to TU and ATU, the large gains over these methods on the significantly challenging ISIC dataset (Wu et al., 2022) establishes the usefulness of the proposed technique.

Cross-domain transferability analysis: The DermNet-S and miniImageNet-S datasets both allow 5-way 1-shot classification. Moreover, miniImageNet-S contains images from natural scenes, while DermNet-S consists of medical images. Therefore, we undertake a cross-domain transferability study in Table 5. We summarize the Accuracy values obtained by a source model trained on DermNet-S but tested on miniImageNet-S and vice-versa (denoted DS → mIS and mIS → DS, respectively). In most cases, the IBP variant can improve upon the corresponding baseline. Further, the interpolation-based methods, i.e. MLTI and IBI, are able to further enhance performance, with IBI achieving the best performance in most cases, validating that IBI training can improve cross-domain transferability.

Table 4. Performance comparison of the two proposed methods with baselines and contending algorithms in terms of 5-way, 1-shot and 5-shot mean Accuracy over 600 tasks.

Backbone Network	Algorithm	miniImageNet-S		ISIC		DermNet-S	
		1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
4-CONV	MAML (Finn et al., 2017; Yao et al., 2022)	38.27%	52.14%	57.59%	65.24%	43.47%	60.56%
	MAML+Meta-Reg (Yin et al., 2019; Yao et al., 2022)	38.35%	51.74%	58.57%	68.45%	45.01%	60.92%
	TAML (Jamal & Qi, 2019; Yao et al., 2022)	38.70%	52.75%	58.39%	66.09%	45.73%	61.14%
	MAML+Meta-Dropout (Lee et al., 2020; Yao et al., 2022)	38.32%	52.53%	58.40%	67.32%	44.30%	60.86%
	MAML+MetaMix (Yao et al., 2021; 2022)	39.43%	54.14%	60.34%	69.47%	46.81%	63.52%
	MAML+Meta-Maxup (Ni et al., 2021; Yao et al., 2022)	39.28%	53.02%	58.68%	69.16%	46.10%	62.64%
	MAML+MLTI (Yao et al., 2022)	41.58%	55.22%	61.79%	70.69%	48.03%	64.55%
	MAML+Meta-Interpolation (Lee et al., 2022)	40.28%	53.06%	-	-	-	-
	MAML+TU (Wu et al., 2022)	42.16%	56.33%	62.03%	73.97%	48.07%	64.81%
	MAML+ATU (Wu et al., 2022)	**42.60%**	**56.78%**	62.84%	74.50%	48.33%	65.16%
	MAML+IBP (ours)	41.30%	54.36%	64.91%	78.75%	48.33%	63.33%
	MAML+IBI (ours)	42.20%	55.23%	**68.58%**	**79.75%**	**49.13%**	**65.43%**
	ProtoNet* (Snell et al., 2017; Yao et al., 2022)	36.26%	50.72%	58.56%	66.25%	44.21%	60.33%
	ProtoNet (Snell et al., 2017)	40.70%	53.16%	65.58%	75.25%	46.86%	62.03%
	ProtoNet*+MetaMix (Yao et al., 2021; 2022)	39.67%	53.10%	60.58%	70.12%	47.71%	62.68%
	ProtoNet*+Meta-Maxup (Ni et al., 2021; Yao et al., 2022)	39.80%	53.35%	59.66%	68.97%	46.06%	62.97%
	ProtoNet*+MLTI (Yao et al., 2022)	41.36%	55.34%	62.82%	71.52%	49.38%	65.19%
	ProtoNet+IBP (ours)	41.46%	55.00%	**70.75%**	81.01%	48.66%	**67.26%**
	ProtoNet+IBI (ours)	**43.30%**	**55.73%**	70.25%	**81.16%**	**51.13%**	65.93%
ResNet-12	MAML (Finn et al., 2017; Yao et al., 2022)	40.02%	52.56%	59.41%	67.66%	47.58%	63.13%
	MAML+MetaMix (Yao et al., 2021; 2022)	42.26%	54.65%	62.06%	72.18%	51.40%	64.82%
	MAML+MetaMaxup (Ni et al., 2021; Yao et al., 2022)	41.97%	53.92%	61.64%	72.04%	50.82%	64.24%
	MAML+MLTI (Yao et al., 2022)	43.35%	54.89%	62.16%	73.56%	52.03%	65.12%
	MAML+IBP (ours)	43.50%	55.13%	**64.50%**	73.91%	50.40%	65.40%
	MAML+IBI (ours)	**43.90%**	**57.00%**	63.25%	**75.66%**	**52.10%**	**66.50%**
	ProtoNet* (Snell et al., 2017; Yao et al., 2022)	40.96%	53.77%	61.91%	72.97%	48.65%	64.61%
	ProtoNet (Snell et al., 2017)	42.60%	55.00%	63.01%	75.91%	50.66%	65.40%
	ProtoNet*+MetaMix (Yao et al., 2021; 2022)	42.95%	56.95%	65.55%	78.33%	51.18%	66.80%
	ProtoNet*+MetaMaxup (Ni et al., 2021; Yao et al., 2022)	42.68%	56.07%	64.17%	77.62%	50.96%	66.38%
	ProtoNet*+MLTI (Yao et al., 2022)	44.08%	57.14%	66.02%	79.15%	52.01%	67.28%
	ProtoNet+IBP (ours)	43.33%	57.40%	66.66%	81.00%	51.33%	67.57%
	ProtoNet+IBI (ours)	**45.33%**	**58.23%**	**66.75%**	**81.83%**	**52.53%**	**68.00%**

* ProtoNet implementation as per (Yao et al., 2022).

Table 5. Transferability comparison of MAML and ProtoNet, with their MLTI, IBP, and IBI variants in terms of 5-way, 1-shot mean Accuracy over 600 tasks.

Algorithms	Accuracy	
	DS → mIS	mIS → DS
MAML	25.06%	33.40%
MAML+MLTI (Yao et al., 2022)	30.03%	**36.74%**
MAML+IBP (ours)	27.06%	33.90%
MAML+IBI (ours)	**30.23%**	36.21%
ProtoNet	28.76%	34.03%
ProtoNet*+MLTI (Yao et al., 2022)	30.06%	35.46%
ProtoNet+IBP (ours)	29.60%	34.13%
ProtoNet+IBI (ours)	**30.32%**	**35.63%**

*: ProtoNet implementation as per Yao et al. (2022).

6. Conclusion and future works

We explore the utility of IBP beyond its originally-intended usage for building and verifying classifiers that are provably robust against adversarial attacks. We identify the potential of IBP to conserve a neighborhood from the input image space to the learned feature space through the layers of a deep neural network by minimizing the distances of the feature embedding from the two bounds. This can be effective in few-shot classification problems to obtain feature embeddings where task neighborhoods are preserved, thus enabling easy adaptability to unseen tasks. Further, interpolating between training tasks and their corresponding IBP bounds can yield artificial tasks with a higher chance of lying on the task manifold, that are likely to prevent overfitting to seen tasks in the few-task scenario. The resulting IBI is shown to be effective in both the meta-learning and metric-learning paradigms of few-shot learning.

Our results demonstrate that IBI can be effectively scaled to relatively large networks like ResNet-12 as IBP is typically needed in a few initial layers (see Appendix E). This may still add some extra computational cost (see Appendix E for a comparative study), which scales linearly with the number of layers subjected to IBP. Thus, one may investigate the applicability of advanced provably robust training methods that yield more efficient and tighter bounds (Lyu et al., 2021). Few-shot learners can also be improved with adaptive hyperparameters (Baik et al., 2020), feature reconstruction (Lee & Chung, 2021), knowledge distillation (Tian et al., 2020), embedding propagation (Rodríguez et al., 2020), etc. One can observe the performance gains from these orthogonal techniques when coupled with IBI. However, this may not be a straightforward endeavor, given the complex dynamic nature of such frameworks.

References

Abukmeil, M., Ferrari, S., Genovese, A., Piuri, V., and Scotti, F. A survey of unsupervised generative models for exploratory data analysis and representation learning. *Acm computing surveys (csur)*, 54(5):1–40, 2021.

Allen, K., Shelhamer, E., Shin, H., and Tenenbaum, J. Infinite mixture prototypes for few-shot learning. In *International Conference on Machine Learning*, pp. 232–241. PMLR, 2019.

Baik, S., Choi, M., Choi, J., Kim, H., and Lee, K. M. Meta-learning with adaptive hyperparameters. *Advances in Neural Information Processing Systems*, 33:20755–20765, 2020.

Bertinetto, L., Henriques, J. F., Torr, P., and Vedaldi, A. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2019.

Boyd, S. and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

Brehmer, J. and Cranmer, K. Flows for simultaneous manifold learning and density estimation. *Advances in Neural Information Processing Systems*, 33:442–453, 2020.

Caterini, A. L., Loaiza-Ganem, G., Pleiss, G., and Cunningham, J. P. Rectangular flows for manifold learning. *Advances in Neural Information Processing Systems*, 34: 30228–30241, 2021.

Codella, N. C., Gutman, D., Celebi, M. E., Helba, B., Marchetti, M. A., Dusza, S. W., Kalloo, A., Liopyris, K., Mishra, N., Kittler, H., et al. Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (isbi), hosted by the international skin imaging collaboration (isic). In *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*, pp. 168–172. IEEE, 2018.

Connor, M., Canal, G., and Rozell, C. Variational autoencoder with learned latent structure. In *International Conference on Artificial Intelligence and Statistics*, pp. 2359–2367. PMLR, 2021.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Gowal, S., Dvijotham, K. D., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Arandjelovic, R., Mann, T., and Kohli, P. Scalable verified training for provably robust image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Grant, E., Finn, C., Levine, S., Darrell, T., and Griffiths, T. Recasting gradient-based meta-learning as hierarchical bayes. In *International Conference on Learning Representations*, 2018.

Hariharan, B. and Girshick, R. Low-shot visual recognition by shrinking and hallucinating features. In *Proceedings of the IEEE international conference on computer vision*, pp. 3018–3027, 2017.

Jamal, M. A. and Qi, G.-J. Task agnostic meta-learning for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11719–11727, 2019.

Kumar, A. and Poole, B. On implicit regularization in $\beta - vaes$. In *International Conference on Machine Learning*, pp. 5480–5490. PMLR, 2020.

Lee, D. H. and Chung, S.-Y. Unsupervised embedding adaptation via early-stage feature reconstruction for few-shot classification. In *International Conference on Machine Learning*, pp. 6098–6108. PMLR, 2021.

Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning*, pp. 609–616, 2009.

Lee, H. B., Nam, T., Yang, E., and Hwang, S. J. Meta dropout: Learning to perturb latent features for generalization. In *International Conference on Learning Representations*, 2020.

Lee, K., Maji, S., Ravichandran, A., and Soatto, S. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10657–10665, 2019.

Lee, S., Andreis, B., Kawaguchi, K., Lee, J., and Hwang, S. J. Set-based meta-interpolation for few-task meta-learning. *arXiv preprint arXiv:2205.09990*, 2022.

Li, X., Yu, L., Jin, Y., Fu, C.-W., Xing, L., and Heng, P.-A. Difficulty-aware meta-learning for rare disease diagnosis. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 357–366. Springer, 2020.

Li, Z., Zhou, F., Chen, F., and Li, H. Meta-SGD: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

Lyu, Z., Guo, M., Wu, T., Xu, G., Zhang, K., and Lin, D. Towards evaluating and training verifiably robust neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4308–4317, June 2021.

Meir, R. and Zhang, T. Generalization error bounds for bayesian mixture algorithms. *Journal of Machine Learning Research*, 4(Oct):839–860, 2003.

Mirman, M., Gehr, T., and Vechev, M. Differentiable abstract interpretation for provably robust neural networks. In *International Conference on Machine Learning*, pp. 3578–3586. PMLR, 2018.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Morawiecki, P., Spurek, P., Śmieja, M., and Tabor, J. Fast and stable interval bounds propagation for training verifiably robust models. In *ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2020.

Ni, R., Goldblum, M., Sharaf, A., Kong, K., and Goldstein, T. Data augmentation for meta-learning. In *International Conference on Machine Learning*, pp. 8152–8161. PMLR, 2021.

Nichol, A., Achiam, J., and Schulman, J. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Oh, J., Yoo, H., Kim, C., and Yun, S.-Y. {BOIL}: Towards representation change for few-shot learning. In *International Conference on Learning Representations*, 2021.

Prabhu, V., Kannan, A., Ravuri, M., Chaplain, M., Sontag, D., and Amatriain, X. Few-shot learning for dermatological disease diagnosis. In *Proceedings of the 4th Machine Learning for Healthcare Conference*, volume 106, pp. 532–552. PMLR, 2019.

Raghunathan, A., Steinhardt, J., and Liang, P. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018.

Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J. B., Larochelle, H., and Zemel, R. S. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*, 2018.

Rodríguez, P., Laradji, I., Drouin, A., and Lacoste, A. Embedding propagation: Smoother manifold for few-shot classification. In *European Conference on Computer Vision*, pp. 121–138. Springer, 2020.

Roweis, S. T. and Saul, L. K. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290 (5500):2323–2326, 2000.

Satorras, V. G. and Estrach, J. B. Few-shot learning with graph neural networks. In *International Conference on Learning Representations*, 2018.

Shalev-Shwartz, S. and Ben-David, S. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

Sunaga, T. Theory of an interval algebra and its application to numerical analysis. *RAAG memoirs*, 2:29–46, 1958.

Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H., and Hospedales, T. M. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Tenenbaum, J. B., Silva, V. d., and Langford, J. C. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

Tian, Y., Wang, Y., Krishnan, D., Tenenbaum, J. B., and Isola, P. Rethinking few-shot image classification: a good embedding is all you need? In *European Conference on Computer Vision*, pp. 266–282. Springer, 2020.

Van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

Wang, Y., Yao, Q., Kwok, J. T., and Ni, L. M. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020.

Wojtowytsch, S. and E, W. On the banach spaces associated with multi-layer relu networks: Function representation, approximation theory and gradient descent dynamics. *arXiv preprint arXiv:2007.15623*, 2020.

Wong, E., Schmidt, F., Metzen, J. H., and Kolter, J. Z. Scaling provable adversarial defenses. *Advances in Neural Information Processing Systems*, 31, 2018.

Wu, Y., Huang, L.-K., and Wei, Y. Adversarial task upsampling for meta-learning. In *Advances in Neural Information Processing Systems*, 2022.

Yao, H., Huang, L.-K., Zhang, L., Wei, Y., Tian, L., Zou, J., Huang, J., et al. Improving generalization in meta-learning via task augmentation. In *International Conference on Machine Learning*, pp. 11887–11897. PMLR, 2021.

Yao, H., Zhang, L., and Finn, C. Meta-learning with fewer tasks through task interpolation. In *International Conference on Learning Representations*, 2022.

Yin, M., Tucker, G., Zhou, M., Levine, S., and Finn, C. Meta-learning without memorization. *arXiv preprint arXiv:1912.03820*, 2019.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

A. Prototype-based metric-learning

Metric-based few-shot learning aims to obtain a feature embedding of the task manifold suitable for non-parametric classification. Prototype-based metric-learning, specifically Prototypical Network (ProtoNet) (Snell et al., 2017), assigns a query point to the class having the nearest (in terms of Euclidean distance) prototype in the learned embedding space. Given the model f_θ and a task $\mathcal{T}_i$, we compute class prototypes $\{\mathbf{c}_k\}_{k=1}^K$ as the mean of $f_\theta(\mathbf{x}^s_{i,r})$ for the instances $\mathbf{x}^s_{i,r}$ belonging to class k:

$$\mathbf{c}_k = \frac{1}{N_s} \sum\nolimits_{(\mathbf{x}^s_{i,r}, y^s_{i,r}) \in \mathcal{D}_i^{s,k}} f_\theta(\mathbf{x}^s_{i,r}), \tag{9}$$

where $\mathcal{D}_i^{s,k} \subset \mathcal{D}_i^s$ represents the subset of N_s support samples from class k. Given a sample $\mathbf{x}^q_{i,r}$ from the query set, the probability $p(y^q_{i,r} = k | \mathbf{x}^q_{i,r})$ of assigning it to the k-th class is calculated using the distance function $d(.,.)$ between the representation $f_\theta(\mathbf{x}^q_{i,r})$ and the prototype $\mathbf{c}_k$:

$$p(y^q_{i,r} = k | \mathbf{x}^q_{i,r}, f_\theta) = \frac{\exp(-d(f_\theta(\mathbf{x}^q_{i,r}), \mathbf{c}_k))}{\sum_{k'} \exp(-d(f_\theta(\mathbf{x}^q_{i,r}), \mathbf{c}_{k'}))}. \tag{10}$$

Thereafter, the parameters θ for the model f_θ can be trained by minimizing cross-entropy loss (2). During testing, each query sample $\mathbf{x}^q_{j,r}$ is assigned to the class having the maximal probability, i.e., $\mathbf{y}^q_{j,r} = \arg\max_k p(y^q_{j,r} = k | \mathbf{x}^q_{j,r})$.

B. Algorithms of MAML and ProtoNet coupled with IBP and IBI

The steps for MAML+IBP/IBI and ProtoNet+IBP/IBI are respectively presented in Algorithm 1 and 2. Please consult the main paper for various notations and equations used in the algorithms. Also recall from Section 4.3, that an artificial task in IBI is created as:

$$\mathbf{H}^{s'}_{i,r} = (1-\lambda_k) f_{\theta^S}(\mathbf{x}^s_{i,r}) + (1-\nu_k)\lambda_k \underline{f}_{\theta^S}(\mathbf{x}^s_{i,r}, \epsilon) + \nu_k \lambda_k \overline{f}_{\theta^S}(\mathbf{x}^s_{i,r}, \epsilon). \tag{11}$$

Remark B.1. The way in which the training support set $\mathcal{D}_i^s$ informs the loss calculation on the corresponding query set $\mathcal{D}_i^q$ differs between the MAML and ProtoNet variants. While a limited number of training steps on the support set is undertaken to obtain the model f_{ϕ_i} where the loss is calculated on the query set for MAML, the support set is used to calculate the prototypes $\{\mathbf{c}_k\}_{k=1}^K$ for the loss calculation on the query set for ProtoNet.

C. Detailed Theoretical Analysis

Interval bound propagation for networks with affine layer: Let us assume a network f with L layers where the 0-th layer denotes the initial input. Let us also consider a layer $l \leq L$ that is not the 0-th input layer. The 0-th layer of f takes the input along with its perturbed counterparts, as shown in Section 3 in the main paper. If at the end of $l-1$-th layer, the activation, upper bound, and lower bound are respectively $\mathbf{z}_{l-1}$, $\overline{\mathbf{z}}_{l-1}$ and $\underline{\mathbf{z}}_{l-1}$. Suppose the l-th layer performs an affine transformation (such as a convolutional, fully connected, batch normalization, etc.) followed by a monotonic activation function (such as ReLU, sigmoid, tanh, etc.), i.e. $\mathbf{z}_l = \sigma(A_l \mathbf{z}_{l-1} + \mathbf{b}_l)$, then as per Gowal et al. (2019). In that case, we can calculate the interval bounds for the subsequent l-th layer as follows:

$$\underline{\mathbf{z}}_l = \sigma(\mu_l - \psi_l), \tag{12}$$

$$\overline{\mathbf{z}}_l = \sigma(\mu_l + \psi_l), \tag{13}$$

where $\psi_l = |A_l|\psi_{l-1}$ and $\mu_l = A_l \mu_{l-1} + \mathbf{b}_l$ given $\mu_{l-1} = \frac{\underline{\mathbf{z}}_{l-1} + \overline{\mathbf{z}}_{l-1}}{2}$ and $\psi_{l-1} = \frac{\underline{\mathbf{z}}_{l-1} - \overline{\mathbf{z}}_{l-1}}{2}$.

Lemma 4.2 (Lipschitz networks ensure bounded IBP). Let $\overline{\mathbf{x}}$ and $\underline{\mathbf{x}}$ be ε-perturbations of $\mathbf{x} \sim \mathbb{P}_X$ for an $\varepsilon > 0$ (i.e. $\overline{\mathbf{x}}, \underline{\mathbf{x}} \in \mathbf{x}(\varepsilon)$). Given that the activation σ is Lipschitz continuous (such as ReLU) with constant $c_\sigma > 0$, there exists a constant $D = D(c_\sigma; A_1, A_2, \cdots, A_S; \varepsilon)$ such that $\underline{f}_{\theta^S}(\mathbf{x}, \varepsilon)$ and $\overline{f}_{\theta^S}(\mathbf{x}, \varepsilon)$) will at most be an $\hat{\varepsilon}$-perturbed version of $f_{\theta^S}(\mathbf{x})$, where $\hat{\varepsilon} = \varepsilon D$.

Proof. Given that $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$

$$\begin{aligned} \|h_1(\mathbf{x}_1) - h_1(\mathbf{x}_2)\| &= \|\sigma(A_1\mathbf{x}_1 + \mathbf{b}_1) - \sigma(A_1\mathbf{x}_2 + \mathbf{b}_1)\| \\ &\leq c_\sigma \|A_1(\mathbf{x}_1 - \mathbf{x}_2)\| \\ &\leq c_\sigma \|A_1\| \|\mathbf{x}_1 - \mathbf{x}_2\| \end{aligned} \tag{14}$$

Algorithm 1 IBP/IBI for MAML training

Requires: Task distribution $p(\mathcal{T})$, batch size B, learning rates η_0 and η_1, interval coefficient ϵ.

```
Randomly initialize the meta-learner parameters $\theta$.
while not converged do
    Sample a batch of $B$ tasks from the distribution $\rho(\mathcal{T})$.
    For IBI, randomly sample an index $1 \leq m \leq B$ to perform the interpolation.
    for all $i \in \{1, 2, \cdots, B\}$ do
        Initialize base learner to meta-learner state.
        Sample a support set $\mathcal{D}_i^s$ of data-label pairs $\{(\mathbf{x}_{i,r}^s, \mathbf{y}_{i,r}^s)\}_{r=1}^{N_s}$ from task $\mathcal{T}_i$.
        Calculate the classification loss $\mathcal{L}_{CE}$ using $f_\theta(\mathbf{x}_{i,r}^s)$ and $\mathbf{y}_{i,r}^s$.
        if $i = m$ then
            Generate interpolated support and query instances $\mathbf{H}_{i,r}^{s'}$ and $\mathbf{H}_{i,r}^{s'}$ using (11).
            Calculate classification loss $\mathcal{L}'_{CE}$ using $f_{\theta^{L-S}}(\mathbf{H}_{i,r}^{s'})$ and $\mathbf{y}_{i,r}^s$.
            Set $\mathcal{L}_{CE} = \frac{1}{2}(\mathcal{L}_{CE} + \mathcal{L}'_{CE})$.
        end if
        Update base learner parameters to $\phi_i = \theta - \eta_0 \nabla_\theta \mathcal{L}_{CE}$.
        Sample a query set $\mathcal{D}_i^q$ of data-label pairs $\{(\mathbf{x}_{i,r}^q, \mathbf{y}_{i,r}^q)\}_{r=1}^{N_q}$ from task $\mathcal{T}_i$.
        Calculate the classification loss $\mathcal{L}_{CE}$ with $f_{\phi_i}(\mathbf{x}_{i,r}^q)$ and $\mathbf{y}_{i,r}^q$.
        Calculate $\mathcal{L}_{LB}$ and $\mathcal{L}_{UB}$ respectively using (5) and (6).
        if $i = m$ then
            Calculate classification loss $\mathcal{L}'_{CE}$ using $f_{\phi^{L-S}}(\mathbf{H}_{i,r}^{q'})$ and $\mathbf{y}_{i,r}^q$.
            Set $\mathcal{L}_{CE} = \frac{1}{2}(\mathcal{L}_{CE} + \mathcal{L}'_{CE})$.
        end if
        Calculate $\mathcal{L}$ by accumulating $\mathcal{L}_{CE}$, $\mathcal{L}_{LB}$ and $\mathcal{L}_{UB}$ using (7).
    end for
    Update meta-learner parameters $\theta = \theta - \eta_1 \frac{1}{B} \sum_{i=1}^{B} \nabla_\theta \mathcal{L}$.
end while
```

Algorithm 2 IBP/IBI for ProtoNet training

Requires: Task distribution $p(\mathcal{T})$, learning rate η, interval coefficient ϵ.

```
Randomly initialize the learner parameters $\theta$.
while not converged do
    For IBI, randomly select if interpolation is to be performed.
    Sample a support set $\mathcal{D}_i^s$ of data-label pairs $\{(\mathbf{x}_{i,r}^s, \mathbf{y}_{i,r}^s)\}_{r=1}^{N_s}$ from task $\mathcal{T}_i$.
    Calculate the features $f_{\theta^L}(\mathbf{x}_{i,r}^s)$ and find the prototypes $\{\mathbf{c}_k\}_{k=1}^K$ using (9).
    if interpolation to be performed then
        Generate interpolated support and query instances $\mathbf{H}_{i,r}^{s'}$ and $\mathbf{H}_{i,r}^{s'}$ using (11).
        Calculate features $f_{\theta^{L-S}}(\mathbf{H}_{i,r}^{s'})$ and find prototypes $\{\mathbf{c}'_k\}_{k=1}^K$.
    end if
    Sample a query set $\mathcal{D}_i^q$ of data-label pairs $\{(\mathbf{x}_{i,r}^q, \mathbf{y}_{i,r}^q)\}_{r=1}^{N_q}$ from task $\mathcal{T}_i$.
    Calculate the loss $\mathcal{L}_{CE}$ using (10) and (2).
    Calculate $\mathcal{L}_{LB}$ and $\mathcal{L}_{UB}$ using (5) and (6).
    if interpolation to be performed then
        Calculate classification loss $\mathcal{L}'_{CE}$ with $f_{\theta^{L-S}}(\mathbf{H}_{i,r}^{q'})$, $\{\mathbf{c}'_k\}_{k=1}^K$ and $\mathbf{y}_{i,r}^q$ by (10) and (2).
        Set $\mathcal{L}_{CE} = \frac{1}{2}(\mathcal{L}_{CE} + \mathcal{L}'{}_{CE})$.
    end if
    Calculate $\mathcal{L}$ by accumulating $\mathcal{L}_{CE}$, $\mathcal{L}_{LB}$ and $\mathcal{L}_{UB}$ using (7).
    Update learner parameters $\theta = \theta - \eta \nabla_\theta \mathcal{L}$.
end while
```

where $\|A_1\| = \sup_{\|\mathbf{x}\|=1} \|A_1 \mathbf{x}\|$. The inequality 14 is due to the Lipschitz continuity of σ. Commonly used activation functions, such as ReLU, tend to satisfy this condition. In particular, for ReLU, $c_\sigma = 1$. As such, the map h_1 also turns out to be Lipschitz continuous. A similar argument also proves that $h_l, l = 2, \cdots, S$ all follow the same trait. As a result, f_{θ^S} also becomes Lipschitz continuous with accompanying constant $(c_\sigma A)^S$, where $A = \max\{\|A_l\|\}$.

The recurrence relation of extremities in IBP, as suggested by Gowal et al. (2019), can be written as:

$$\begin{aligned} \overline{f}_{\theta^l}(\mathbf{x}, \varepsilon) &= \sigma\left\{ \frac{(A_l + |A_l|)}{2} \overline{f}_{\theta^{l-1}}(\mathbf{x}, \varepsilon) + \frac{(A_l - |A_l|)}{2} \underline{f}_{\theta^{l-1}}(\mathbf{x}, \varepsilon) + \mathbf{b}_l \right\}, \\ \text{and } \underline{f}_{\theta^l}(\mathbf{x}, \varepsilon) &= \sigma\left\{ \frac{(A_l - |A_l|)}{2} \overline{f}_{\theta^{l-1}}(\mathbf{x}, \varepsilon) + \frac{\{A_l + |A_l|\}}{2} \underline{f}_{\theta^{l-1}}(\mathbf{x}, \varepsilon) + \mathbf{b}_l \right\}, \end{aligned}$$

where the $|\cdot|$ operator results in a matrix with all elements replaced by their corresponding absolute values, and $l = 1, 2, ..., S$. Thus,

$$\begin{aligned} &\left\| \overline{f}_{\theta^l}(\mathbf{x}, \varepsilon) - f_{\theta^l}(\mathbf{x}) \right\| \\ =& \left\| \sigma\left\{ \frac{(A_l + |A_l|)}{2} \overline{f}_{\theta^{l-1}}(\mathbf{x}, \varepsilon) + \frac{(A_l - |A_l|)}{2} \underline{f}_{\theta^{l-1}}(\mathbf{x}, \varepsilon) + \mathbf{b}_l \right\} - \sigma\left(A_l f_{\theta^{l-1}}(\mathbf{x}) + \mathbf{b}_l \right) \right\| \\ \leq& c_\sigma \left\| \frac{(A_l + |A_l|)}{2} \overline{f}_{\theta^{l-1}}(\mathbf{x}, \varepsilon) + \frac{(A_l - |A_l|)}{2} \underline{f}_{\theta^{l-1}}(\mathbf{x}, \varepsilon) - A_l f_{\theta^{l-1}}(\mathbf{x}) \right\| \\ =& c_\sigma \left\| \frac{(A_l + |A_l|)}{2} \left(\overline{f}_{\theta^{l-1}}(\mathbf{x}, \varepsilon) - f_{\theta^{l-1}}(\mathbf{x}) \right) + \frac{(A_l - |A_l|)}{2} \left(\underline{f}_{\theta^{l-1}}(\mathbf{x}, \varepsilon) - f_{\theta^{l-1}}(\mathbf{x}) \right) \right\| \\ \leq& c_\sigma \left\{ \left\| \frac{(A_l + |A_l|)}{2} \left(\overline{f}_{1(\theta^{l-1}}(\mathbf{x}, \varepsilon) - f_{\theta^{l-1}}(\mathbf{x}) \right) \right\| + \left\| \frac{(|A_l| - A_l)}{2} \left(f_{\theta^{l-1}}(\mathbf{x}) - \underline{f}_{\theta^{l-1}}(\mathbf{x}, \varepsilon) \right) \right\| \right\} \\ \leq& c_\sigma \left\{ \left\| \frac{A_l + |A_l|}{2} \right\| \left\| \overline{f}_{\theta^{l-1}}(\mathbf{x}, \varepsilon) - f_{\theta^{l-1}}(\mathbf{x}) \right\| + \left\| \frac{|A_l| - A_l}{2} \right\| \left\| f_{\theta^{l-1}}(\mathbf{x}) - \underline{f}_{\theta^{l-1}}(\mathbf{x}, \varepsilon) \right\| \right\}. \end{aligned}$$

Observe that, in particular for $l = 1$

$$\begin{aligned} \left\| \overline{f}_{\theta^1}(\mathbf{x}, \varepsilon) - f_{\theta^1}(\mathbf{x}) \right\| &\leq c_\sigma \left\{ \left\| \frac{A_1 + |A_1|}{2} \right\| \|\overline{\mathbf{x}} - \mathbf{x}\| + \left\| \frac{|A_1| - A_1}{2} \right\| \|\mathbf{x} - \underline{\mathbf{x}}\| \right\} \\ &= c_\sigma \varepsilon \left\{ \left\| \frac{A_1 + |A_1|}{2} \right\| + \left\| \frac{|A_1| - A_1}{2} \right\| \right\} = \varepsilon_1 \text{ say,} \end{aligned}$$

i.e., the deviation in the first layer can be made arbitrarily small based on ε. The quantity $\left\| f_{\theta^1}(\mathbf{x}) - \underline{f}_{\theta^1}(\mathbf{x}, \varepsilon) \right\|$ can be shown to be upper bounded using a similar argument. In other words, both $\overline{f}_{\theta^1}(\mathbf{x}, \varepsilon)$ and $\underline{f}_{\theta^1}(\mathbf{x}, \varepsilon)$ are at most ε_1-perturbed from $f_{\theta^1}(\mathbf{x})$. By the method of induction we eventually get a $D = D(c_\sigma; A_1, A_2, \cdots, A_S; \varepsilon) > 0$ for which the lemma holds. □

Theorem 4.3 (Generalization bound). Let $\tilde{\mathbb{P}}$ be the joint distribution of $(f_{\theta^S}(X), Y)$, supported on $\mathcal{H} \times \mathbb{R}$. In the multi-task regime, let I denote the set of tasks, each consisting of N samples. Define $\hat{\mathcal{R}}(N, |I|) = \mathbb{E}_{\mathcal{T}_i \sim \hat{p}(\mathcal{T})} \mathbb{E}_{(X_j, Y_j) \sim \hat{p}(\mathcal{T}_i)} [\mathcal{L}_{CE}(f_{\theta^{L-S}}(\mathbf{H}_j^*), Y_j)]$ and $\mathcal{R} = \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} \mathbb{E}_{(X_j, Y_j) \sim \mathcal{T}_i} [\mathcal{L}_{CE}(f_{\theta^{L-S}}(f_{\theta^S}(X_j)), Y_j)]$. For a bounded loss function $\mathcal{L}_{CE} : \mathbb{R} \times \mathbb{R} \to [0, a] (a \geq 0)$, if the neural network-induced map $f_{\theta^{L-S}}$ is such that $\left| \nabla f_{\theta^{L-S}}(\cdot) \right| < \infty$, we ensure:

$$\left| \hat{\mathcal{R}}(N, |I|) - \mathcal{R} \right| - \tilde{\lambda} \precsim 2^{L-S+1} \sqrt{2 \log(2\kappa + 2)} \left\{ \frac{1}{\sqrt{N}} + \frac{1}{\sqrt{|I|}} \right\} + \sqrt{\frac{\log(\frac{2|I|}{\delta})}{N}} + \sqrt{\frac{\log(\frac{2}{\delta})}{|I|}}$$

holds with probability at least $1 - \delta$, where $\tilde{\lambda} = \tilde{\lambda}(\hat{\varepsilon}, \lambda)$.

Proof. Before beginning with the proof, we point out that, based on Definition 4.1, given $\varepsilon > 0$ and $\mathbf{x} \in \mathcal{X}$, any $\mathbf{x}^{'} \in \mathbf{x}(\varepsilon)$ can be written as $\mathbf{x}^{'} = \mathbf{x} + \eta(\varepsilon)$. For example, in the simplest case, $\eta(\varepsilon)$ can be a function in the family $\pm\epsilon\mathbf{1}$. Thus, in case of IBI, the $\underline{f}_{\theta^S}(\mathbf{x}_i, \varepsilon)$ and $\overline{f}_{\theta^S}(\mathbf{x}_i, \varepsilon)$ can both be expressed as $f_{\theta^S}(\mathbf{x}_i) + \eta(\hat{\varepsilon})$ with corresponding $\eta(\hat{\varepsilon})$. In essence $\mathbf{H}_i^* = (1-\lambda) f_{\theta^S}(\mathbf{x}_i) + \lambda(f_{\theta^S}(\mathbf{x}_i) + \eta(\hat{\varepsilon}))$, where $\lambda \in [0, 1]$. Now, we can observe that,

$$\begin{aligned}
f_{\theta^{L-S}}(\mathbf{H}_i^*) &= f_{\theta^{L-S}}\left((1-\lambda) f_{\theta^S}(\mathbf{x}_i) + \lambda\left[f_{\theta^S}(\mathbf{x}_i) + \eta(\hat{\varepsilon})\right]\right) \\
&= f_{\theta^{L-S}}\left(f_{\theta^S}(\mathbf{x}_i) + \lambda\eta(\hat{\varepsilon})\right) \\
&= f_{\theta^{L-S}}\left(f_{\theta^S}(\mathbf{x}_i)\right) + \lambda\nabla f_{\theta^{L-S}}\left(f_{\theta^S}(\mathbf{x_i})\right)\eta(\hat{\varepsilon}),
\end{aligned} \tag{15}$$

where $\eta(\hat{\varepsilon}) \in \mathbb{R}^{\kappa}$, $\hat{\varepsilon}$ being as mentioned in lemma 4.2. We obtain (15) by using the Taylor expansion of $f_{\theta^{L-S}}$ up to the first order. Given that $\left|\nabla f_{\theta^{L-S}}(\cdot)\right| < \infty$, the second term $\lambda\nabla f_{\theta^{L-S}}\left(f_{\theta^S}(\mathbf{x}_i)\right)\eta(\hat{\varepsilon})$ can be made arbitrarily small. The higher-order terms in the expansion all follow suit, which justifies their omission. Now,

$$\begin{aligned}
&\left|\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_{CE}(f_{\theta^{L-S}}(\mathbf{H}_i^*), y_i) - \int_{\mathcal{H}\times\mathbb{R}}\mathcal{L}_{CE}(f_{\theta^{L-S}}(\mathbf{x}), y)d\tilde{\mathbb{P}}(\mathbf{x}, y)\right| \\
=&\left|\frac{1}{N}\sum_{i=1}^{N}\left[\mathcal{L}_{CE}(f_{\theta^{L-S}}(\mathbf{H}_i^*), y_i) - \mathcal{L}_{CE}(f_{\theta^{L-S}}(f_{\theta^S}(\mathbf{x}_i)), y_i)\right]\right. \\
&\left. +\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_{CE}(f_{\theta^{L-S}}(f_{\theta^S}(\mathbf{x}_i)), y_i) - \int_{\mathcal{H}\times\mathbb{R}}\mathcal{L}_{CE}(f_{\theta^{L-S}}(\mathbf{x}), y)d\tilde{\mathbb{P}}(\mathbf{x}, y)\right| \\
\leq&\frac{1}{N}\sum_{i=1}^{N}\left|\mathcal{L}_{CE}(f_{\theta^{L-S}}(\mathbf{H}_i^*), y_i) - \mathcal{L}_{CE}(f_{\theta^{L-S}}(f_{\theta^S}(\mathbf{x}_i)), y_i)\right| \\
&+\left|\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_{CE}(f_{\theta^{L-S}}(f_{\theta^S}(\mathbf{x}_i)), y_i) - \int_{\mathcal{H}\times\mathbb{R}}\mathcal{L}_{CE}(f_{\theta^{L-S}}(\mathbf{x}), y)d\tilde{\mathbb{P}}(\mathbf{x}, y)\right|.
\end{aligned} \tag{16}$$

Since our networks use ReLU activation, the map induced by $f_{\theta^{L-S}}$ can be shown to be continuous. Given $\mathcal{H}$ is compact, the output space also becomes compact. Restricted to such a space, the cross-entropy loss $\mathcal{L}_{CE}$ (similarly, regularized cross-entropy loss) turns out to be Lipschitz continuous. Consequently,

$$\begin{aligned}
&|\mathcal{L}_{CE}(f_{\theta^{L-S}}(\mathbf{H}_i^*), y_i) - \mathcal{L}_{CE}(f_{\theta^{L-S}}(f_{\theta^S}(\mathbf{x}_i)), y_i)| \\
&\qquad\leq c_L\left\|f_{\theta^{L-S}}(\mathbf{H}_i^*) - f_{\theta^{L-S}}\left(f_{\theta^S}(\mathbf{x}_i)\right)\right\| = \tilde{\lambda}(\hat{\varepsilon}, \lambda),
\end{aligned} \tag{17}$$

where $c_L > 0$ is the Lipschitz constant associated with $\mathcal{L}_{CE}$. Without loss of generality we can construct the map $f_{\theta^{L-S}}$ such that $\|f_{\theta^{L-S}}\| \leq 1$. Now, in case there are $|I|$ tasks involved, namely $\{\mathcal{T}_i\}_{i=1}^{|I|}$ (i.e., the multi-task regime), the population risk turns out to be

$$\begin{aligned}
\mathcal{R} &= \mathbb{E}_{\mathcal{T}_i\sim p(\mathcal{T})}\mathbb{E}_{(X_j, Y_j)\sim\mathcal{T}_i}\left[\mathcal{L}_{CE}\left(f_{\theta^{L-S}}(f_{\theta^S}(X_j)), Y_j\right)\right] \\
&= \mathbb{E}_{\mathcal{T}_i\sim p(\mathcal{T})}\mathbb{E}_{(X_j, Y_j)\sim\mathcal{T}_i}\left[\mathcal{L}_{CE}\left(f_{\theta^{L-S}}(\mathbf{H}_j), Y_j\right)\right].
\end{aligned}$$

We are interested in observing the deviation of the same from the realized risk. In other words,

$$\left|\hat{\mathcal{R}}(N, |I|) - \mathcal{R}\right| \leq \underbrace{\left|\hat{\mathcal{R}}(N, |I|) - \mathcal{J}\right|}_{\text{(i)}} + \underbrace{\left|\mathcal{J} - \mathcal{R}\right|}_{\text{(ii)}}, \tag{18}$$

where $\mathcal{J} = \mathbb{E}_{\mathcal{T}_i\sim\hat{p}(\mathcal{T})}\mathbb{E}_{(X_j, Y_j)\sim\mathcal{T}_i}\left[\mathcal{L}_{CE}\left(f_{\theta^{L-S}}(\mathbf{H}_j), Y_j\right)\right]$ and $\hat{p}$ is the empirical counterpart of the task distribution. Using the Jensen's inequality, (i) can be upper bounded by

$$\begin{aligned}
&\mathbb{E}_{\mathcal{T}_i\sim\hat{p}(\mathcal{T})}\left|\mathbb{E}_{(X_j,Y_j)\sim\hat{p}(\mathcal{T}_i)}\left[\mathcal{L}_{CE}\left(f_{\theta^{L-S}}(\mathbf{H}_j^*),Y_j\right)\right]-\mathbb{E}_{(X_j,Y_j)\sim\mathcal{T}_i}\left[\mathcal{L}_{CE}\left(f_{\theta^{L-S}}(\mathbf{H}_j),Y_j\right)\right]\right|\\
&\leq\tilde{\lambda}+\mathbb{E}_{\mathcal{T}_i\sim\hat{p}(\mathcal{T})}\left|\mathbb{E}_{(X_j,Y_j)\sim\hat{p}(\mathcal{T}_i)}\left[\mathcal{L}_{CE}\left(f_{\theta^{L-S}}(\mathbf{H}_j),Y_j\right)\right]-\mathbb{E}_{(X_j,Y_j)\sim\mathcal{T}_i}\left[\mathcal{L}_{CE}\left(f_{\theta^{L-S}}(\mathbf{H}_j),Y_j\right)\right]\right|, \qquad (19)
\end{aligned}$$

where we utilize arguments (16) and (17) to reach (19). Using the union bound based on $|I|$ tasks on top of Corollary 3.14 of Wojtowytsch & E (2020) we can show that the second term in the right-hand side of (19) becomes $\precsim 2^{L-S+1}\sqrt{\frac{2\log(2\kappa+2)}{N}}+a\sqrt{\frac{2\log(\frac{2|I|}{\delta})}{N}}$, with probability at least $1-\delta$.

To put a deterministic upper bound on (ii) let us first define the class of functions

$$\mathcal{G}=\left\{g:g(\mathcal{T})=\mathbb{E}_{\left(f_{\theta^S}(X),Y\right)\sim\tilde{\mathbb{P}}}\left[\mathcal{L}_{CE}\left(f_{\theta^{L-S}}(\mathbf{H}),Y\right)\right];f_{\theta^{L-S}}\in W^{L-S}\right\},$$

where W^{L-S} is the function space induced by networks with $L-S$ hidden layers (Wojtowytsch & E, 2020). Let us now calculate the Rademacher complexity of the class functions $\mathcal{G}$:

$$\begin{aligned}
Rad\left(\mathcal{G},\{\mathcal{T}_i\}_{i=1}^{|I|}\right)&=\mathbb{E}_\xi\sup_{g\in\mathcal{G}}\frac{1}{|I|}\left|\sum_{i=1}^{|I|}\xi_i g(\mathcal{T}_i)\right|=\mathbb{E}_\xi\sup_{g\in\mathcal{G}}\frac{1}{|I|}\left|\sum_{i=1}^{|I|}\xi_i\mathbb{E}_{\mathcal{T}_i}\left[\mathcal{L}_{CE}\left(f_{\theta^{L-S}}(\mathbf{H}),Y\right)\right]\right|\\
&\leq\mathbb{E}_{\mathcal{T}_i}\mathbb{E}_\xi\sup_{g\in\mathcal{G}}\frac{1}{|I|}\left|\sum_{i=1}^{|I|}\xi_i\mathcal{L}_{CE}\left(f_{\theta^{L-S}}(\mathbf{H}),Y\right)\right| \qquad (20)\\
&\leq c_L\mathbb{E}_{\mathcal{T}_i}\mathbb{E}_\xi\sup_{f_{\theta^{L-S}}\in W^{L-S}}\frac{1}{|I|}\left|\sum_{i=1}^{|I|}\xi_i f_{\theta^{L-S}}(\mathbf{H})\right| \qquad (21)\\
&\leq c_L 2^{L-S+1}\sqrt{\frac{2\log(2\kappa+2)}{|I|}}, \qquad (22)
\end{aligned}$$

where (21) is due to the Lipschitz property of $\mathcal{L}_{CE}(\cdot,y)$ [Lemma 26.9 of (Shalev-Shwartz & Ben-David, 2014) or Theorem 7 of (Meir & Zhang, 2003)]. We arrive at (22) using lemma 3.13 of (Wojtowytsch & E, 2020). The inequality (20) is based on the fact that $\sup_{u\in\mathcal{U}}\left|\mathbb{E}[u(X)]\right|\leq\mathbb{E}[\sup_{u\in\mathcal{U}}\left|u(X)\right|]$, given the expectation exists for the class of functions $\mathcal{U}$ and random variable X.

Thus we obtain the deterministic bound on (ii) given by

$$\begin{aligned}
&\left|\mathbb{E}_{\mathcal{T}_i\sim\hat{p}(\mathcal{T})}\mathbb{E}_{(X_j,Y_j)\sim\mathcal{T}_i}\left[\mathcal{L}_{CE}\left(f_{\theta^{L-S}}(\mathbf{H}_j),Y_j\right)\right]-\mathbb{E}_{\mathcal{T}_i\sim p(\mathcal{T})}\mathbb{E}_{(X_j,Y_j)\sim\mathcal{T}_i}\left[\mathcal{L}_{CE}\left(f_{\theta^{L-S}}(\mathbf{H}_j),Y_j\right)\right]\right|\\
&\precsim\sqrt{\frac{2\log(2\kappa+2)}{|I|}}+\sqrt{\frac{\log(\frac{2}{\delta})}{|I|}},
\end{aligned}$$

that holds with probability at least $1-\delta$. The bounds on (i) and (ii) together prove the theorem. □

D. Details of datasets used in this study

miniImageNet: The miniImageNet dataset (Vinyals et al., 2016) is a commonly used subset of ImageNet (Deng et al., 2009) for evaluating few-shot classifiers. The dataset contains a total of 100 classes, each containing 600 images of resolution $84\times84\times3$. Following the directives of Vinyals et al. (2016) from the total 100 classes, 64 are kept in the Training set, 16 are retained for validation, and the rest of the 20 classes are used for testing.

tieredImageNet: In (Ren et al., 2018) the authors proposed a new larger subset of ImageNet (Deng et al., 2009) for addressing the limitations of miniImageNet. In miniImageNet it is not ensured that the classes used for training are distinct from those contained in the Test set. Evidently, this contains the risk of information leakage and may not provide a fair evaluation of the few-shot classifier. As a remedy Ren et al. (2018) proposed to go higher in the class hierarchy in ImageNet. This enables tieredImageNet to use higher-level categories in the Training, Validation, and Test sets, maintaining significant diversity between the three. In essence, a total of 608 ImageNet leaf-level classes are considered that can be categorized into 34 groups. Among these 34 higher-level groups, 20 are used for training, six are kept for validation, and the rest eight are included in the Test set.

miniImageNet-S: This dataset is created by only using a subset of the original miniImageNet Training set for training the few-shot learner in a few-task scenario Yao et al. (2022).

Training Classes: n03017168, n07697537, n02108915, n02113712, n02120079, n04509417, n02089867, n03888605, n04258138, n03347037, n02606052, n06794110

Validation and Test sets are kept as same as those used in miniImageNet.

DermNet-S: DermNet-S (Yao et al., 2022) is a subset of the ”Dermnet Skin Disease Atlas” publicly available at http://www.dermnet.com/. The dataset, after discarding the duplicates, contains more than 22,000 medical images spread across 625 classes of dermatological diseases. Following the preprocessing suggested by Prabhu et al. (2019) the authors of (Yao et al., 2022) created DermNet-S by first extracting the 203 classes containing more than 30 images. Then from the long-tailed data distribution of the 203 disease classes, the top 30 larger classes are kept for training while the smaller 53 bottom classes are considered for meta-testing. The images are resized to $84 \times 84 \times 3$ to match the resolution of miniImageNet. We follow the same dataset construction strategy in our case. Moreover, we use random classes not included in the Training or Test set as the Validation set. The complete list of classes in the Training and Test sets are listed as follows:

Training Classes: Seborrheic Keratoses Ruff, Herpes Zoster, Atopic Dermatitis Adult Phase, Psoriasis Chronic Plaque, Eczema Hand, Seborrheic Dermatitis, Keratoacanthoma, Lichen Planus, Epidermal Cyst, Eczema Nummular, Tinea (Ringworm) Versicolor, Tinea (Ringworm) Body, Lichen Simplex Chronicus, Scabies, Psoriasis Palms Soles, Malignant Melanoma, Candidiasis large Skin Folds, Pityriasis Rosea, Granuloma Annulare, Erythema Multiforme, Seborrheic Keratosis Irritated, Stasis Dermatitis and Ulcers, Distal Subungual Onychomycosis, Allergic Contact Dermatitis, Psoriasis, Molluscum Contagiosum, Acne Cystic, Perioral Dermatitis, Vasculitis, Eczema Fingertip

Testing Classes: Warts, Ichthyosis Sex Linked, Atypical Nevi, Venous Lake, Erythema Nodosum, Granulation Tissue, Basal Cell Carcinoma Face, Acne Closed Comedo, Scleroderma, Crest Syndrome, Ichthyosis Other Forms, Psoriasis Inversus, Kaposi Sarcoma, Trauma, Polymorphous Light Eruption, Dermagraphism, Lichen Sclerosis Vulva, Pseudomonas, Cutaneous Larva Migrans, Psoriasis Nails, Corns, Lichen Sclerosus Penis, Staphylococcal Folliculitis, Chilblains Perniosis, Psoriasis Erythrodermic, Squamous Cell Carcinoma Ear, Basal Cell Carcinoma Ear, Ichthyosis Dominant, Erythema Infectiosum, Actinic Keratosis Hand, Basal Cell Carcinoma Lid, Amyloidosis, Spiders, Erosio Interdigitalis Blastomycetica, Scarlet Fever, Pompholyx, Melasma, Eczema Trunk Generalized, Metastasis, Warts Cryotherapy, Nevus Spilus, Basal Cell Carcinoma Lip, Enterovirus, Pseudomonas Cellulitis, Benign Familial Chronic Pemphigus, Pressure Urticaria, Halo Nevus, Pityriasis Alba, Pemphigus Foliaceous, Cherry Angioma, Chapped Fissured Feet, Herpes Buttocks, Ridging Beading

ISIC: Following Yao et al. (2022) for “ISIC 2018: Skin Lesion Analysis Towards Melanoma Detection” (Codella et al., 2018; Li et al., 2020), we select the third task where 10,015 medical images are categorized into seven classes based on lesion types. We first resize the images to $84 \times 84 \times 3$ to match the miniImageNet resolution. Then among the seven classes in the ISIC dataset, we select the four classes containing a higher number of samples for training while considering the rest for meta-testing as per the directives of Yao et al. (2022). Since there are only four classes in the Training set, setting the number of ways to 2 results in six possible class combinations in a task. This, in consequence, offers an extreme few-task scenario. For hyper-parameter tuning, random classes are used as a Validation set following the cross-validation-based approach employed in (Yao et al., 2022). The list of classes in the Training and the Test sets are listed as follows:

Training Set: Nevus, Melanoma, Benign Keratoses, Basal Cell Carcinoma

Testing Set: Dermatofibroma, Pigmented Bowen’s, Vascular

E. Implementation details

Scheduling of ϵ: In their paper Gowal et al. (2019) recommended starting with an initial perturbation $\epsilon_0 = 0$ and gradually increasing it to the intended perturbation ϵ over the training steps. In our case, we follow a similar approach for scheduling the value of perturbation ϵ_t at the t-th training step. We have observed that a rapid increase in perturbation usually slows down training while a very slow increment fails to aid the learner. We have found that the following strategy works well in practice from extensive experimentation with various scheduling techniques such as linear, cosine, etc.. If the maximum allowed number of training steps is set to T then for the step t, the perturbation ϵ_t is calculated as:

$$\epsilon_t = \begin{cases} \epsilon & \text{if } t > \lceil 0.9T \rceil \\ \frac{t}{0.9T}\epsilon & \text{otherwise} \end{cases} . \tag{23}$$

In essence, we linearly increase ϵ_t starting from 0 up to ϵ over 90% of the maximum training steps T and keep it fixed at ϵ for the remainder of the training.

Frequency of interpolation for IBI variants: Performing IBP bound–based interpolation for every task during training may not be beneficial and may instead mislead the learner. For MAML, we have seen that performing interpolation once in every batch of B tasks aids the training process. In the case of ProtoNet, we have found that performing IBP bound–based interpolation with a 25% probability results in the best outcome.

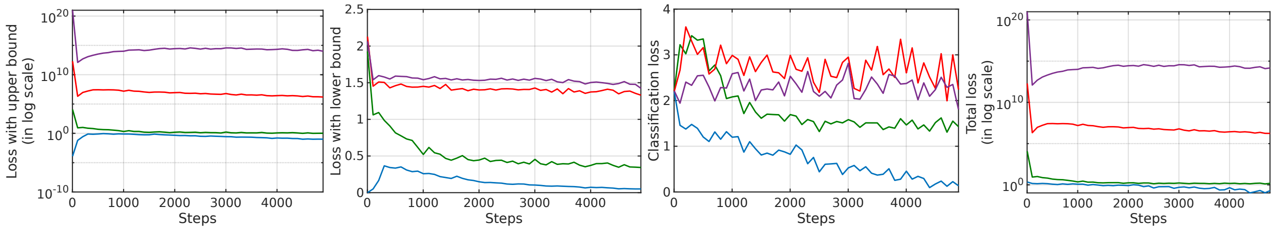
Modifications to network architecture: We have used two networks for our experiments namely “4-CONV” and “ResNet-12”. The “4-CONV” network can be seamlessly integrated with IBI for both MAML and ProtoNet. This network consists of 4 blocks, each having a convolution, batch normalization, max pooling, and ReLU in sequential order. IBI can be performed after any one of the blocks. The ResNet-12 network also consists of 4 blocks where a block (except the first one) receives inputs from (1) the output of the preceding block and (2) the input of the preceding block through a skip connection. While the idea of applying IBI after any of the blocks seems appealing, the presence of skip connections may hinder a straightforward integration of IBI in this case. To understand how ResNet-12 can be customized to accommodate MAML+IBP (and consequently MAML+IBI) we undertake an ablation study on the miniImageNet-S dataset in a 5-way 1-shot classification problem as described in Table 6. We can observe that in our initial hyperparameter tuning experiment, MAML+IBP can not match the performance of vanilla MAML on ResNet-12. Moreover, the performance gap increases as IBP is applied deeper into the network. This may be explained by the fact that the interval bounds become gradually loose as they progress through the network. Thus, with increasing depth, the magnitude of the bound losses (especially $\mathcal{L}_{UB}$ as the ReLU activations prevent $\mathcal{L}_{LB}$ from becoming too large) will largely outscale the classification loss and consequently affect convergence (see Remark E.1). Applying IBP after only the first block still fails to achieve parity with the baseline because IBP induces a distortion in the feature space due to its regularization effect. While the sequential part of the blocks after IBP can adapt to this distortion due to their complexity, the simpler skip paths can not do so. Hence, the effect of the distortion keeps propagating to the deeper blocks via skip connections. To aid the network in such a situation, we investigate three approaches to modify the skip connection immediately after the block(s) subjected to IBP, viz. (1) remove the skip connection for the subsequent block, (2) introduce additional layers in the skip connection for the subsequent block to make it deeper, and more complex (3) use a skip after one or more of the initial sub-block(s) (consisting sequentially of one convolution, one batch normalization, and one ReLU layer) of the next block. Among the three approaches, we empirically found that MAML+IBP (consequently MAML+IBI) performs best when the skip connection starts after the second sub-block in block 2. Due to the comparatively powerful learning strategy of ProtoNet, no such modifications to ResNet-12 are necessary for ProtoNet+IBI.

Remark E.1. [Scalability of IBI] IBP (and consequently IBI) requires the propagation of the two interval bounds along with the input data. This introduces a computational overhead, especially in deeper networks. However, in practice, even in a deeper network, we may only need to perform IBP in the initial few layers, as the bound losses will otherwise overwhelm the classification loss and consequently impact convergence. To demonstrate this, we plot the losses (up to 5000 training steps for the ease of visualization) in the following Figure 5 for MAML+IBI using a ResNet-12 network for 5-way 1-shot miniImageNet-S classification, when IBP is applied up to blocks 1-4. We can see that the three losses have comparable scales only when IBP is applied after block 1. In all other cases, $\mathcal{L}_{UB}$ heavily dominates the total loss. But, due to its sheer magnitude, the optimizer is unable to minimize it. Thus, in practice, IBP should only be limited to a few initial layers in

Table 6. Ablation study of ResNet-12 modifications for MAML+IBP on miniImageNet-S in terms of mean Accuracy over 600 tasks with 95% confidence interval.

Algorithm	IBP position	Accuracy
MAML	None (Baseline)	40.02±0.78%
MAML+IBP	after block 4	21.24±0.54%
MAML+IBP	after block 3	23.77±0.59%
MAML+IBP	after block 2	29.62±0.62%
MAML+IBP	after block 1	37.95±0.83%
MAML+IBP	after block 1 with no-skip at block 2	37.81±0.85%
MAML+IBP	after block 1 with deeper skip at block 2	38.54±0.81%
MAML+IBP	after block 1 with skip and output combination at block 2	40.50±0.83%
MAML+IBP	after block 1 with skip after one sub-block in block 2	42.18±0.82%
MAML+IBP	after block 1 with skip after two sub-blocks in block 2	**43.50±0.86%**

deeper networks. Consequently, IBI easily scales to deeper networks despite the computational overhead.



(a) (b) (c) (d)

Figure 5. In the four plots above of losses against training steps, the Blue, Green, Red, and Magenta lines, respectively, denote IBI applied after blocks 1,2,3, and 4 in ResNet-12 without any additional modifications. (a) The plot of $\mathcal{L}_{UB}$ in log scale for ease of visualization. (b) Plot of $\mathcal{L}_{LB}$. (c) Plot of $\mathcal{L}_{CE}$. (d) The plot of $\mathcal{L}$ in log scale for ease of visualization.

Remark E.2. To show that IBP and IBI variants are well-scalable as their vanilla counterpart, we list the actual training costs in the following Table 7 in terms of the average time in seconds to execute a single training step of the algorithm. All the experiments are performed in the same environment using an RTX 3090 GPU. From Table 7, we can observe that, in the case of MAML, the IBI and IBP variants only takes about 40%-70% additional time when "4-CONV" is used. The difference in cost reduces further if ResNet-12 is used as the backbone. This is expected as we only need to apply IBP in the first few layers of ResNet-12 to gain its full advantage. For ProtoNet, the increment in computational cost for the proposed techniques is slightly higher than that of MAML.

Table 7. Actual computational cost in seconds for IBP and IBI variants of MAML and ProtoNet with "4-CONV" and ResNet-12 backbone.

Algorithm	4-CONV		ResNet-12	
	1-shot	5-shot	1-shot	5-shot
MAML	0.244	0.432	1.408	3.742
MAML + IBP (ours)	0.407	0.615	1.994	4.324
MAML + IBI (ours)	0.412	0.616	2.001	4.326
ProtoNet	0.067	0.073	0.075	0.091
ProtoNet + IBP (ours)	0.129	0.144	0.196	0.221
ProtoNet + IBI (ours)	0.133	0.156	0.202	0.233

F. Hyperparameters used in IBP and IBI

F.1. Names and functions of hyperparameters

The following Table 8 describes the hyperparameters used in the vanilla MAML, MAML+IBP, and MAML+IBI.

Table 8. Descriptions of hyperparameters used in vanilla MAML, MAML+IBP, MAML+IBI

Hyperarameter name	Hyperparameter description
Hyperparameters used in MAML	
Meta-shots	Number of shots in the query set in the training phase.
Inner loop iterations	Number of iterations of the inner loop during training on support set.
Inner loop learning rate η_0	Learning rate for SGD in the inner loop during training on support set.
Meta-step size η_1	Learning rate for ADAM in the meta-learner update during training.
Meta-batch B	Batch size of task during training.
Meta-iterations T	Number of training steps.
Evaluation iterations	Number of fine-tuning steps on the support set during meta-testing.
Additional hyperparameters introduced in MAML+IBP	
Interval coefficient ϵ	Perturbation required for IBP.
Softmax coefficient γ	Controls the relative importance of the three losses used in MAML+IBP during softmax-based weighting in training phase.
Layer S	A layer in the network where IBP losses will be calculated.
Additional hyperparameters introduced in MAML+IBI	
α and β	Hyperparameters associated with the $Beta$ distribution required for performing IBP bounds-based interpolation.

The following Table 9 describes the hyperparameters used in the vanilla ProtoNet, ProtoNet+IBP, and ProtoNet+IBI.

Table 9. Descriptions of hyperparameters used in vanilla ProtoNet, ProtoNet+IBP, ProtoNet+IBI

Hyperarameter name	Hyperparameter description
Hyperparameters used in MAML	
Number of ways in training	Traditional ProtoNet (Snell et al., 2017) usually considers a higher number of ways during training.
Meta-shots	Number of shots in the query set in the training phase.
Meta-step size η	Learning rate for ADAM in the learner update during training.
Meta-iterations T	Number of training steps.
Distance metric	Choice of distance measure, Euclidean or Cosine.
Additional hyperparameters introduced in ProtoNet+IBP	
Interval coefficient ϵ	Perturbation required for IBP.
Softmax coefficient γ	Controls the relative importance of the three losses used in ProtoNet+IBP during softmax-based weighting in the training phase.
Layer S	A layer in the network where IBP losses will be calculated.
Additional hyperparameters introduced in ProtoNet+IBI	
α and β	Hyperparameters associated with the $Beta$ distribution required for performing IBP bounds-based interpolation.

F.2. Hyperparameter search space and tuning

For hyperparameter tuning, we employ a grid search. In Table 10, we list the search spaces for each of the hyperparameters used in MAML+IBP and MAML+IBI. Moreover, in Table 11, we also detail the search spaces for each of the hyperparameters used in ProtoNet+IBP and ProtoNet+IBI. For all other learners used in Tables 1 and 2 in the main paper, the results are either taken from the corresponding article or reproduced using the originally recommended hyperparameter settings.

In Tables 12 and 13, we report the optimal dataset-specific hyperparameters for MAML+IBP and MAML+IBI. Similarly, Tables 14 and 15 detail the optimal dataset-specific hyperparameter choices for ProtoNet+IBP and ProtoNet+IBI.

For Table 3 in the main paper, the methods using static weights share the same hyperparameter settings with their dynamic weighted counterpart except for γ, which is not used for the static weight runs. For Table 4 in the main paper, all the MAML variants use the same settings as vanilla MAML. Further, for all the different interpolation strategies, $Beta$ distribution is used with the choices of α and β matching those of the MAML+IBI settings.

Table 10. Grid search space of hyperparameters used in vanilla MAML, MAML+IBP, MAML+IBI

Hyperarameter name	Hyperparameter search space
Hyperparameters used in MAML	
Meta-shots	Set to 15 following (Finn et al., 2017).
Inner loop iterations	Set to 5 following (Finn et al., 2017).
Inner loop learning rate η_0	Set to 0.01 following (Finn et al., 2017).
Meta-step size η_1	Set to 0.001 following (Finn et al., 2017).
Meta-batch B	Set to 4 following (Finn et al., 2017).
Meta-iterations T	Set to 60000 for miniImageNet and tieredImageNet following (Finn et al., 2017). Set to 50000 for miniImageNet-S, DermNet-S, and ISIC following (Yao et al., 2022).
Evaluation iterations	Set to 10 following (Finn et al., 2017).
Additional hyperparameters introduced in MAML+IBP	
Interval coefficient ϵ	Searched in the set $\{0.05, 0.1, 0.2\}$.
Softmax coefficient γ	Searched in the set $\{0.01, 1, 10\}$.
Layer S	For the "4-CONV" learner containing 4 blocks of Convolution, Batch normalization, Max pooling, and ReLU, S is searched at the block level in the set $\{1, 2, 3, 4\}$. For example, $S = 2$ means IBP losses are calculated after the second block. For the "ResNet-12" network the ablation study in Appendix E provides the optimum choice of S.
Additional hyperparameters introduced in MAML+IBI	
α and β	Search space contains three pairs of choices $(0.1, 1)$, $(0.25, 1)$, and $(0.5, 0.5)$ where a tuple contains the value of α and β in order.

Table 11. Grid search space of hyperparameters used in vanilla ProtoNet, ProtoNet+IBP, ProtoNet+IBI

Hyperarameter name	Hyperparameter search space
Hyperparameters used in ProtoNet	
Number of ways in training	Set to 30 for miniImageNet and tieredImageNet following (Snell et al., 2017). Set to 5 for miniImageNet-S and DermNet-S, and 2 for ISIC as the benefit of training using higher ways cannot be leveraged in the few-task scenario (Yao et al., 2022).
Meta-shots	Set to 15 following (Snell et al., 2017).
Meta-step size η	Set to 0.001 following (Snell et al., 2017).
Meta-iterations T	Set to 20000 for miniImageNet and tieredImageNet following (Snell et al., 2017). Our implementation of ProtoNet, unlike (Yao et al., 2022), does not require an additional hyperparameter B, analogous to MAML, for IBP or IBI training. Thus, for miniImageNet-S, DermNet-S, and ISIC also we set T to 20000.
Distance metric	Set to Euclidean following (Snell et al., 2017).
Additional hyperparameters introduced in ProtoNet+IBP	
Interval coefficient ϵ	Searched in the set $\{0.05, 0.1, 0.2\}$.
Softmax coefficient γ	Searched in the set $\{0.01, 1, 10\}$.
Layer S	For the "4-CONV" learner containing 4 blocks of Convolution, Batch normalization, Max pooling, and ReLU, S is searched at the block level in the set $\{1, 2, 3, 4\}$. For example, $S = 2$ means IBP losses are calculated after the second block. For the "ResNet-12" network the ablation study in Appendix E provides the optimum choice of S.
Additional hyperparameters introduced in ProtoNet+IBI	
α and β	Search space contains three pairs of choices $(0.1, 1)$, $(0.25, 1)$, and $(0.5, 0.5)$ where a tuple contains the value of α and β in order.

Table 12. Optimal hyperparamter setting for MAML+IBP, MAML+IBI in 1-shot settings when "4-CONV" network is used.

Hyperarameter	Hyperparameter settings for datasets				
	miniImageNet	tieredImageNet	miniImageNet-S	DermNet-S	ISIC
Additional hyperparameters introduced in MAML+IBP					
ϵ	0.1	0.05	0.1	0.2	0.05
γ	0.1	0.1	0.1	0.1	0.1
S	3	3	3	3	3
Additional hyperparameters introduced in MAML+IBI					
α and β	(0.25, 1)	(0.25, 1)	(0.5, 0.5)	(0.5, 0.5)	(0.25, 1)

Table 13. Optimal hyperparamter setting for MAML+IBP, MAML+IBI in 5-shot settings when "4-CONV" network is used.

Hyperarameter	Hyperparameter settings for datasets				
	miniImageNet	tieredImageNet	miniImageNet-S	DermNet-S	ISIC
Additional hyperparameters introduced in MAML+IBP					
ϵ	0.1	0.05	0.1	0.2	0.05
γ	0.1	0.1	0.1	0.1	0.1
S	3	3	3	3	3
Additional hyperparameters introduced in MAML+IBI					
α and β	(0.1, 1)	(0.1, 1)	(0.5, 0.5)	(0.5, 0.5)	(0.25, 1)

Table 14. Optimal hyperparamter setting for ProtoNet+IBP, ProtoNet+IBI in 1-shot settings when "4-CONV" network is used.

Hyperarameter	Hyperparameter settings for datasets				
	miniImageNet	tieredImageNet	miniImageNet-S	DermNet-S	ISIC
Additional hyperparameters introduced in ProtoNet+IBP					
ϵ	0.05	0.05	0.1	0.1	0.05
γ	1	1	1	1	1
S	1	1	1	1	1
Additional hyperparameters introduced in ProtoNet+IBI					
α and β	(0.1, 1)	(0.25, 1)	(0.5, 0.5)	(0.25, 1)	(0.1, 1)

Table 15. Optimal hyperparamter setting for ProtoNet+IBP, ProtoNet+IBI in 5-shot settings when "4-CONV" network is used.

Hyperarameter	Hyperparameter settings for datasets				
	miniImageNet	tieredImageNet	miniImageNet-S	DermNet-S	ISIC
Additional hyperparameters introduced in ProtoNet+IBP					
ϵ	0.05	0.05	0.1	0.1	0.05
γ	1	1	1	1	1
S	1	1	1	1	1
Additional hyperparameters introduced in ProtoNet+IBI					
α and β	(0.1, 1)	(0.1, 1)	(0.5, 0.5)	(0.5, 0.5)	(0.25, 1)

F.3. Full results

Contenders in Motivating Example: For the contenders in Table 1 the settings are as follows:

1. MAML+SN on f_{θ^S}: This variant of MAML applies Spectral Normalization (Miyato et al., 2018) up to the S-th layer

Table 16. Optimal hyperparameter settings for MAML+IBP/IBI and ProtoNet+IBP/IBI when “ResNet-12” is used as the network.

			Datasets		
Learner	Shots	Parameter	miniImageNet-S	DermNet-S	ISIC
MAML+IBP	1 and 5	ϵ	0.1	0.1	0.1
MAML+IBP	1 and 5	γ	0.1	0.1	0.1
MAML+IBP	1 and 5	S^*	1	1	1
MAML+IBI (Additional)	1 and 5	α and β	(0.1, 1)	(0.1, 1)	(0.1, 1)
ProtoNet+IBP	1 and 5	ϵ	0.05	0.05	0.05
ProtoNet+IBP	1 and 5	γ	0.1	0.1	0.1
ProtoNet+IBP	1 and 5	S^*	1	1	1
ProtoNet+IBI (Additional)	1 and 5	α and β	(0.1, 1)	(0.1, 1)	(0.1, 1)

*: Set as per Appendix E with necessary modifications.

of the “4-CONV” network. Here similar to the MAML+IBP the value of S is set to 3.

2. MAML+SN on f_θ: Here Spectral Normalization is applied on the full network.
3. MAML+GL: In this variant, we calculate a Gaussian regularization loss instead of IBP. Here we send the query set along with its perturbed version and attempt to minimize their norm after the S-th layer alongside $\mathcal{L}_{CE}$. The extra loss $\mathcal{L}_{GL}$ can be expressed as follows:

$$\mathcal{L}_{GL} = \frac{1}{N_q}\sum_{r=1}^{N_q} ||f_{\theta^S}(\mathbf{x}^q_{i,r}) - f_{\theta^S}(\mathbf{x}^q_{i,r} + \zeta)||_2^2,$$

where $\zeta \sim \mathcal{N}(0, \sigma)$, and the standard deviation σ is scheduled similar to ϵ with starting from 0 and slowly increasing to $\epsilon/2$.

4. MAML+ULBL: Following (Morawiecki et al., 2020) we replace the two bound losses with a single one that calculates the distance between the upper and lower interval bounds. The loss $\mathcal{L}_{ULBL}$ in this case can be written as:

$$\mathcal{L}_{ULBL} = \frac{1}{N_q}\sum_{r=1}^{N_q} ||\overline{f}_{\theta^S}(\mathbf{x}^q_{i,r}, \epsilon) - \underline{f}_{\theta^S}(\mathbf{x}^q_{i,r}, \epsilon)||_2^2$$

The full version of Table 1 is provided in the following Table 17.

Table 17. Effect of IBP on MAML for miniImageNet and tieredImageNet datasets in terms of 5-way 1-shot Accuracy and intra-task compactness. This is the full version of Table 1.

Algorithm	Accuracy		1-NN distance	
	miniImageNet	tieredImageNet	miniImageNet	tieredImageNet
MAML (Finn et al., 2017)	48.70±1.75%	51.67±1.81%	0.97±0.02	0.98±0.02
MAML+SN on f_{θ^S}	44.90±1.12%	45.26±1.05%	1.38±0.04	1.41±0.04
MAML+SN on f_θ	42.83±0.94%	43.06±0.96%	1.52±0.04	1.53±0.04
MAML+GL	48.70± 0.97%	51.90±0.98%	0.96±0.02	0.98±0.02
MAML+ULBL	49.43±0.90%	51.67±0.91%	0.94±0.02	0.97±0.02
MAML+IBP (ours)	**50.76±0.83%**	**54.36±0.80%**	**0.90±0.02**	**0.96±0.02**

Comparison of IBP with other few-shot learners: As contending meta-learning algorithms, we choose the vanilla MAML along with notable meta-learners such as Meta-SGD (Li et al., 2017), Reptile (Nichol et al., 2018), LLAMA (Grant et al., 2018), R2-D2 (Bertinetto et al., 2019), and BOIL (Oh et al., 2021). Moreover, considering the regularizing effect of IBP and IBI, we also include meta-learners such as TAML (Jamal & Qi, 2019), Meta-Reg (Yin et al., 2019), and Meta-Dropout (Lee et al., 2020) which employ explicit regularization. We further include data augmentation–reliant learners such as MetaMix

(Yao et al., 2021), Meta-Maxup (Ni et al., 2021), as well as the inter-task interpolation method MLTI (Yao et al., 2022). In case of metric-learners, we compare against the vanilla ProtoNet in addition to other notable methods like MatchingNet (Vinyals et al., 2016), RelationNet (Sung et al., 2018), IMP (Allen et al., 2019), and GNN (Satorras & Estrach, 2018). We also compare against ProtoNet coupled with data augmentation methods such as MetaMix, Meta-Maxup, and MLTI, as done in (Yao et al., 2022). While (Yao et al., 2022) had to modify the training strategy of the canonical ProtoNet to accommodate the changes introduced by MetaMix, Meta-Maxup, and MLTI, the flexibility of IBP and IBI imposes no such requirements. We summarize the findings in Table 18. We can observe that either IBP or IBI or both achieve better Accuracy than the competitors in all cases. The slightly better performance of IBP with ProtoNet seems to imply that IBP-based task interpolation is often unnecessary for ProtoNet when a large number of tasks is available.

Table 18. Performance comparison of the two proposed methods with baselines and competing algorithms on miniImageNet and tieredImageNet datasets. The results are reported in terms of mean Accuracy over 600 tasks with 95% confidence interval.

Dataset	Learner type	Algorithm	1-shot	5-shot
miniImageNet	Meta-learners	MAML (Finn et al., 2017)	48.70±1.75%	63.11±0.91%
		Meta-SGD (Li et al., 2017)	50.47±1.87%	64.03±0.94%
		Reptile (Nichol et al., 2018)	49.97±0.32%	65.99±0.58%
		LLAMA (Grant et al., 2018)	49.40±0.84%	-
		R2-D2 (Bertinetto et al., 2019)	49.50±0.20%	65.40±0.20%
		TAML (Jamal & Qi, 2019; Yao et al., 2022)	46.40±0.82%	63.26±0.68%
		BOIL (Oh et al., 2021)	49.61±0.16%	66.45±0.37%
		MAML+Meta-Reg (Yin et al., 2019; Yao et al., 2022)	47.02±0.77%	63.19±0.69%
		MAML+Meta-Dropout (Lee et al., 2020; Yao et al., 2022)	47.47±0.81%	64.11±0.71%
		MAML+MetaMix (Yao et al., 2021; 2022)	47.81±0.78%	64.22±0.68%
		MAML+Meta-Maxup (Ni et al., 2021; Yao et al., 2022)	47.68±0.79%	63.51±0.75%
		MAML+MLTI (Yao et al., 2022)	48.62±0.76%	64.65±0.70%
		MAML+IBP (ours)	50.76±0.83%	67.13±0.81%
		MAML+IBI (ours)	**52.16±0.84%**	**67.56±0.86%**
	Metric-learners	MatchingNet (Vinyals et al., 2016)	43.44±0.77%	55.31±0.73%
		RelationNet (Sung et al., 2018)	50.44±0.82%	65.32±0.70%
		IMP (Allen et al., 2019)	49.60±0.80%	68.10±0.80%
		GNN (Satorras & Estrach, 2018)	49.02±0.98%	63.50±0.84%
		ProtoNet (Snell et al., 2017)	49.42±0.78%	68.20±0.66%
		ProtoNet*+MetaMix (Yao et al., 2021; 2022)	47.21±0.76%	64.38±0.67%
		ProtoNet*+Meta-Maxup (Ni et al., 2021; Yao et al., 2022)	47.33±0.79%	64.43±0.69%
		ProtoNet*+MLTI (Yao et al., 2022)	48.11±0.81%	65.22±0.70%
		ProtoNet+IBP (ours)	50.48±0.83%	68.33±0.79%
		ProtoNet+IBI (ours)	**51.79±0.81%**	**68.46±0.79%**
tieredImageNet	Meta-learners	MAML (Finn et al., 2017)	51.67±1.81%	70.30±0.08%
		Meta-SGD (Li et al., 2017)	48.97±0.21%	66.47±0.21%
		Reptile (Nichol et al., 2018)	49.97±0.32%	65.99±0.58%
		BOIL (Oh et al., 2021)	49.35±0.26%	69.37±0.12%
		MAML+IBP (ours)	**54.36±0.80%**	**71.30±0.77%**
		MAML+IBI (ours)	54.16±0.79%	71.00±0.84%
	Metric-learners	MatchingNet (Vinyals et al., 2016)	54.02±0.79%	70.11±0.82%
		RelationNet (Sung et al., 2018)	54.48±0.93%	71.32±0.78%
		ProtoNet (Snell et al., 2017)	53.31±0.20%	72.69±0.74%
		ProtoNet+IBP (ours)	53.83±0.81%	**75.26±0.83%**
		ProtoNet+IBI (ours)	**55.16±0.77%**	74.96±0.82%

* ProtoNet implementation as per (Yao et al., 2022).

Notes on contenders used in Table 2: The extra parameter settings required for the contenders in Table 2 are as follows:

1. MAML+WCL: Here given a task its worst-case loss in the ϵ-neighborhood (Gowal et al., 2019) is added with the original loss. In essence, this acts similar to augmentation with the worst-case logits. We tune the relative contribution of the original task and the worst-case task to the final $\mathcal{L}_{CE}$ following the recommendations made by (Gowal et al., 2019).

2. MAML+GA (image space): Here the original task is perturbed with Gaussian noise to form the augmented task in the image space. The noise is sampled from a Gaussian with mean 0 and standard deviation $\sigma = \epsilon/2$. The value of σ is scheduled similarly to ϵ.

3. MAML+GA (at f_{θ^S} feature space): Here the embedding of the original task after f_{θ^S} is perturbed with Gaussian noise. Similar to the image space, the mean of the normal distribution used for sampling noise can be set to 0. However,

finding a good σ may not be straightforward as the f_{θ^S} feature space is continuously updating. In our implementation, we take σ as half of the median distance between the original task and its bounds over a MAML+IBI run.

The full version of Table 2 is detailed in Table 19. The full version of Table 4 in the main paper is provided here across Tables 20 and 21. Moreover, the full version of Table 5 in the main paper is presented in Table 22.

Table 19. Full version of Table 2 for performance comparison of MAML+IBI against 11 augmentation strategies, in the 5-way 1-shot setting. The results are reported in terms of mean Accuracy over 600 tasks along with the 95% confidence intervals.

Algorithm	mIS	ISIC	DS
MAML+Inter-task interpolation in image space	40.90±0.86%	55.25±1.58%	48.30±0.81%
MAML+Inter-task interpolation after f_{θ^S}	41.00±0.83%	61.33±1.52%	47.43±0.78%
MAML+WCL	41.56±0.88%	66.83±1.64%	48.20±0.81%
MAML+ULBL+WCL	41.27±0.84%	63.50±1.48%	48.43±0.80%
MAML+IBP+WCL	41.56±0.85%	64.75±1.61%	48.90±0.83%
MAML+ULBL+Intra-task Interpolation	40.37±0.80%	64.91±1.45%	48.23±0.77%
MAML+GA (Image Space)	41.33±0.85%	63.25±1.68%	47.67±0.86%
MAML+IBP+GA (Image Space)	41.83±0.82%	62.67±1.59%	48.83±0.82%
MAML+IBP+GA (after f_{θ^S})	41.66±0.84%	63.75±1.63%	47.60±0.82%
MAML+MLTI (Yao et al., 2022)	41.58±0.72%	61.79±1.00%	48.03±0.80%
MAML+IBI without $\mathcal{L}_{UB}$ and $\mathcal{L}_{LB}$ losses	35.26±0.79%	48.94±1.36%	41.30±0.81%
MAML+IBI (Ours)	**42.20±0.82%**	**68.58±0.93%**	**49.13±0.80%**

Table 20. Full results for MAML variants on miniImageNet-S, DermNet-S, and ISIC in Table 4 of the main paper. All results are reported in terms of Accuracy over 600 tasks along with 95% confidence level.

Algorithm	4-CONV		ResNet-12	
	5-way 1-shot	5-way 5-shot	5-way 1-shot	5-way 5-shot
miniImageNet-S				
MAML (Finn et al., 2017)	38.27±0.74%	52.14±0.65%	40.02±0.78%	52.56±0.85%
MAML+Meta-Reg (Yin et al., 2019; Yao et al., 2022)	38.35±0.76%	51.74±0.68%	-	-
TAML (Jamal & Qi, 2019; Yao et al., 2022)	38.70±0.77%	52.75±0.70%	-	-
MAML+Meta-Dropout (Lee et al., 2020; Yao et al., 2022)	38.32±0.75%	52.53±0.69%	-	-
MAML+MetaMix (Yao et al., 2021; 2022)	39.43±0.77%	54.14±0.73%	42.26±0.75%	54.65±0.87%
MAML+Meta-Maxup (Ni et al., 2021; Yao et al., 2022)	39.28±0.77%	53.02±0.72%	41.97±0.78%	53.92±0.85%
MAML+MLTI (Yao et al., 2022)	41.58±0.72%	55.22±0.76%	43.35±0.90%	54.89±0.88%
MAML+IBP (ours)	41.30±0.79%	54.36±0.81%	43.50±0.86%	55.13±0.90%
MAML+IBI (ours)	**42.20±0.82%**	**55.23±0.81%**	**43.90±0.90%**	**57.00±0.88%**
ISIC				
MAML (Finn et al., 2017)	57.59±0.79%	68.24±0.77%	59.41±1.98%	67.66±1.92%
MAML+Meta-Reg (Yin et al., 2019; Yao et al., 2022)	58.57±0.94%	68.45±0.81%	-	-
TAML (Jamal & Qi, 2019; Yao et al., 2022)	58.39±1.00%	66.09±0.71%	-	-
MAML+Meta-Dropout (Lee et al., 2020; Yao et al., 2022)	58.40±1.02%	67.32±0.92%	-	-
MAML+MetaMix (Yao et al., 2021; 2022)	60.34±1.03%	69.47±0.60%	62.06±1.77%	72.18±1.75%
MAML+Meta-Maxup (Ni et al., 2021; Yao et al., 2022)	58.68±0.86%	69.16±0.61%	61.64±1.81%	72.04±1.79%
MAML+MLTI (Yao et al., 2022)	61.79±1.00%	70.69±0.68%	62.16±1.88%	73.56±1.82%
MAML+IBP (ours)	64.91±0.92%	78.75±0.94%	**64.50±1.48%**	73.91±1.42%
MAML+IBI (ours)	**68.58±0.93%**	**79.75±0.91%**	63.25±1.51%	**75.66±1.56%**
DermNet-S				
MAML (Finn et al., 2017)	43.47±0.83%	60.56±0.74%	47.58±0.93%	63.13±0.85%
MAML+Meta-Reg (Yin et al., 2019; Yao et al., 2022)	45.01±0.83%	60.92±0.69%	-	-
TAML (Jamal & Qi, 2019; Yao et al., 2022)	45.73±0.84%	61.14±0.72%	-	-
MAML+Meta-Dropout (Lee et al., 2020; Yao et al., 2022)	44.30±0.84%	60.86±0.73%	-	
MAML+MetaMix (Yao et al., 2021; 2022)	46.81±0.81%	63.52±0.73%	51.40±0.89%	64.82±0.87%
MAML+Meta-Maxup (Ni et al., 2021; Yao et al., 2022)	46.10±0.82%	62.64±0.72%	50.82±0.85%	64.24±0.86%
MAML+MLTI (Yao et al., 2022)	48.03±0.79%	64.55±0.74%	52.03±0.90%	65.12±0.88%
MAML+IBP (ours)	48.33±0.83%	63.33±0.84%	50.40±0.88%	65.40±0.89%
MAML+IBI (ours)	**49.13±0.80%**	**65.43±0.79%**	**52.10±0.87%**	**66.50±0.92%**

Table 21. Full results for ProtoNet variants on miniImageNet-S, ISIC, and DermNet-S in Table 4 of the main paper. All results are reported in terms of Accuracy over 600 tasks along with 95% confidence level.

Algorithm	4-CONV		ResNet-12	
	5-way 1-shot	5-way 5-shot	5-way 1-shot	5-way 5-shot
miniImageNet-S				
ProtoNet* (Snell et al., 2017; Yao et al., 2022)	36.26±0.70%	50.72±0.70%	-	-
ProtoNet (Snell et al., 2017)	40.70±0.79%	53.16±0.77%	40.96±0.75%	55.00±0.86%
ProtoNet*+MetaMix (Yao et al., 2021; 2022)	39.67±0.71%	53.10±0.74%	42.95±0.87%	56.95±0.89%
ProtoNet*+Meta-Maxup (Ni et al., 2021; Yao et al., 2022)	39.80±0.73%	53.35±0.68%	42.68±0.78%	56.07±0.85%
ProtoNet*+MLTI (Yao et al., 2022)	41.36±0.75%	55.34±0.74%	44.08±0.83%	57.14±0.90%
ProtoNet+IBP (ours)	41.46±0.79%	55.00±0.81%	43.33±0.82%	57.40±0.90%
ProtoNet+IBI (ours)	**43.30±0.81%**	**55.73±0.80%**	**45.33±0.85%**	**58.23±0.92%**
ISIC				
ProtoNet* (Snell et al., 2017; Yao et al., 2022)	58.56±1.01%	66.25±0.96%	-	-
ProtoNet (Snell et al., 2017)	65.58±0.91%	75.25±0.90%	61.91±1.94%	75.91±1.92%
ProtoNet*+MetaMix (Yao et al., 2021; 2022)	60.58±1.17%	70.12±0.94%	65.55±1.80%	78.33±1.76%
ProtoNet*+Meta-Maxup (Ni et al., 2021; Yao et al., 2022)	59.66±1.13%	68.97±0.83%	64.17±1.85%	77.62±1.86%
ProtoNet*+MLTI (Yao et al., 2022)	62.82±1.13%	71.52±0.89%	66.02±1.88%	79.15±1.87%
ProtoNet+IBP (ours)	**70.75±0.95%**	81.01±0.93%	66.66±1.52%	81.00±1.49%
ProtoNet+IBI (ours)	70.25±0.91%	**81.16±0.94%**	**66.75±1.63%**	**81.83±1.58%**
DermNet-S				
ProtoNet* (Snell et al., 2017; Yao et al., 2022)	44.21±0.75%	60.33±0.70%	-	-
ProtoNet (Snell et al., 2017)	46.86±0.77%	62.03±0.79%	48.65±0.85%	65.40±0.81%
ProtoNet*+MetaMix (Yao et al., 2021; 2022)	47.71±0.83%	62.68±0.71%	51.18±0.90%	66.80±0.83%
ProtoNet*+Meta-Maxup (Ni et al., 2021; Yao et al., 2022)	46.06±0.78%	62.97±0.74%	50.96±0.88%	66.38±0.85%
ProtoNet*+MLTI (Yao et al., 2022)	49.38±0.85%	65.19±0.73%	52.01±0.93%	67.28±0.87%
ProtoNet+IBP (ours)	48.06±0.81%	**67.26±0.84%**	51.33±0.91%	67.57±0.88%
ProtoNet+IBI (ours)	**51.13±0.80%**	65.93±0.82%	**52.53±0.94%**	**68.00±0.88%**

* ProtoNet implementation as per (Yao et al., 2022).

Table 22. Full result for Table 5 describing transferability comparison of MAML and ProtoNet, with their MLTI, IBP and IBI variants. All results are reported in terms of Accuracy over 600 tasks along with the 95% confidence intervals. Here, $A \to B$ indicates the model trained on dataset A is tested on dataset B.

Algorithms	Accuracy	
	DermNet-S → miniImageNet-S	miniImageNet-S → DermNet-S
MAML	25.06±0.79%	33.40±0.77%
MAML+MLTI	30.03±0.58%	**36.74±0.64%**
MAML+IBP (ours)	27.06±0.78%	33.90±0.81%
MAML+IBI (ours)	**30.23±0.82%**	36.21±0.84%
ProtoNet	28.76±0.82%	34.03±0.80%
ProtoNet*+MLTI	30.06±0.56%	35.46±0.63%
ProtoNet+IBP (ours)	29.60±0.81%	34.13±0.82%
ProtoNet+IBI (ours)	**30.32±0.84%**	**35.63±0.83%**

*: ProtoNet implementation as per Yao et al. (2022).