Revisit Multimodal Meta-Learning through the Lens of Multi-Task Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Multimodal meta-learning is a recent problem that extends conventional few-shot meta-learning by generalizing its setup to diverse multimodal task distributions. This setup mimics how humans make use of a diverse set of prior skills to learn new skills. Previous work has achieved encouraging performance. In particular, in spite of the diversity of the multimodal tasks, previous work claims that a single meta-learner trained on a multimodal distribution can sometimes outperform multiple specialized meta-learners trained on individual unimodal distributions. The improvement is attributed to *knowledge transfer* between different modes of task distributions. However, there is no deep investigation to verify and understand the knowledge transfer between multimodal tasks. Our work makes two contributions to multimodal meta-learning. First, we propose a method to analyze and quantify knowledge transfer across different modes at a micro-level. Our quantitative, task-level analysis is inspired by the recent transference idea from multi-task learning. Second, inspired by hard parameter sharing in multi-task learning and a new interpretation of related work, we propose a new multimodal meta-learner that outperforms existing work by considerable margins. While the major focus is on multimodal meta-learning, our work also attempts to shed light on task interaction in conventional meta-learning.

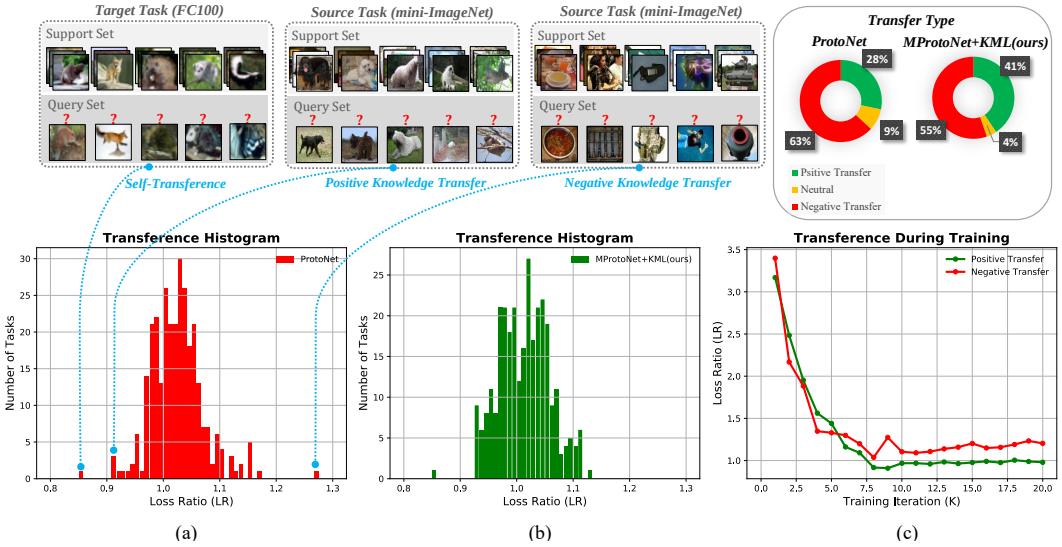
1 Introduction

Multimodal meta-learning was recently proposed as an extension of conventional few-shot meta-learning. In [1], it is defined as a meta-learning problem that involves classification tasks from multiple different input and label domains. An example in their work is a 3-mode few-shot image classification which includes tasks to classify characters (Omniglot) and natural objects of different characteristics (FC100, mini-ImageNet). The multimodal extension of meta-learning is proposed with two objectives. First, it generalizes the conventional meta-learning setup for more diverse task distributions. Second, it mimics humans' ability to acquire a new skill via prior knowledge of a set of diverse skills. For example, humans can quickly learn a novel snowboarding trick by exploiting not only fundamental snowboarding knowledge but also skiing and skateboarding experience [1].

Multimodal Model-Agnostic Meta-Learning (MMAML) [1] proposes a framework to better handle multimodal task distributions and achieves encouraging performance. As one of the most intriguing findings, MMAML claims that a single meta-learner trained on a multimodal distribution can sometimes outperform multiple specialized meta-learners trained on individual unimodal distributions [1]. This was observed in spite of the diversity of the multimodal tasks. This observation is attributed to knowledge transfer across different modes of multimodal task distribution [1].

In our work, we delve into understanding knowledge transfer in multimodal meta-learning. While improved performance using a multimodal task distribution is reported in [1], there is no deep

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.



(a) (b) (c)

Figure 1: Transference from 300 mini-ImageNet tasks to a FC100 task: (a) Transference Histogram for ProtoNet. For both positive knowledge transfer ($LR < 1$), and negative knowledge transfer ($LR > 1$) an exemplar task is shown. (b) Transference Histogram for Proposed method where the positive transfer is increased from 28% to 41%. Here, we simply use the LR threshold to class a task as positive or negative. (c) Transference value for two specified tasks during training.

investigation to verify and understand how tasks from different modes benefit from each other. Towards understanding knowledge transfer in multimodal meta-learning at a micro-level, we propose a new quantification method inspired by the idea of *transference* recently proposed in multi-task learning (MTL) [2]. Despite the large number of meta-learning algorithms proposed in the literature, we remark that little work has been done in analyzing meta-learning at the task sample level. In particular, to the best of our knowledge, there is no previous work on understanding knowledge transfer among tasks and quantitative analysis of task samples in the context of meta-learning. Because of the lack of such study, the interaction between task samples remains rather opaque in meta-learning.

Interestingly, despite the notion of a task in meta-learning and MTL, there has not been much intersections between these two branches of research, perhaps due to several fundamental differences between meta-learning and MTL[1] (e.g., a significantly much larger number of tasks in meta-learning and unavailability of meta-test tasks during training; this will be further discussed). We note that because of these fundamental differences, we propose adaptations to develop our method to quantify knowledge transfer for multimodal tasks in meta-learning.

Another contribution of our work is a new multimodal meta-learner. Our idea is inspired by *hard parameter sharing* in multitask learning [4]. Furthermore, we discuss our own interpretation of the modulation mechanism in [1]. These lead us to propose a new method that achieves substantial improvement over the best results in multimodal meta-learning [1]. While our major focus in this work is on multimodal meta-learning, we have also performed experiments on conventional meta-learning for our proposed knowledge transfer quantification.

Our work makes an attempt to shed light on task interaction in conventional meta-learning.

- Focusing on multimodal meta-learning, we propose a method to understand and quantify knowledge transfer across different modes at a micro-level.
- We propose a new multimodal meta-learner that outperforms existing state-of-the-art methods by substantial margins.

[1]Here we follow the MTL definition in Zhang and Yang [3] instead of a loose definition of MTL: learning problems that involve more that one task.

2 Related Work

Few-Shot Learning. In a few-data regime, conventional learning methods mostly fail due to overfitting. Fine-tuning a pre-trained network [5, 6, 7, 8] sometimes prevents overfitting but at the cost of computation [9]. So, recent successful approaches tackle this problem by meta-learning [10]. These methods can be classified into several categories. In *metric-based* approaches, a similarity metric between support and query samples is learned by learning an embedding space, in which samples from similar classes are close and samples from different classes are further apart [11, 12, 13, 14, 15, 16]. *Optimization-based* methods focus on learning an optimizer, including an LSTM meta-learner for replacing stochastic gradient descent optimizer [17], a mechanism to update weights using an external memory [18], or finding a good initialization point for model parameters for fast adaptation [19, 20, 21, 22]. *Augmentation-based* methods learn a generator from the existing labeled data to further use it for data augmentation in novel classes [23, 24]. Finally, *weight-generation* methods directly generate the classification weights for unseen classes [25, 26].

Multi-Task Learning. MTL algorithms can generally be divided into *hard* or *soft parameter sharing* methods [4]. Hard parameter sharing is the most prevalent design to MTL where a subset of the hidden layers are shared among all tasks, while several task-specific layers are stacked on top of the shared base [27]. Hard parameter sharing reduces the risk of overfitting and enables parameter efficiency across tasks [3, 28]. In soft parameter sharing, each task has a separate model. Then, either the parameters of the models are encouraged to be similar by regularizing a distance [29, 30] or the knowledge of tasks are linearly combined to produce output for each task [31]. Other works try to improve the multi-task performance by addressing *what to share* [32, 33, 34] or *which tasks to train together* [35, 36]. Recently, the *transference* is proposed to analyze the information transfer in a general MTL framework [2]. This metric is then used for selecting the tasks for updating shared parameters.

3 Preliminaries

In the *N-way*, *K-shot* image classification task, given *N* classes and *K* labeled examples per class, the final goal is to learn a model that can generalize well to unseen examples from the same classes. Each task $\mathcal{T}$ consists of a *support* set $\mathcal{S}_\mathcal{T} = \{(x_i, y_i)\}_{i=1}^{N\times K}$ and a query set $\mathcal{Q}_\mathcal{T} = \{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^{N\times M}$. The model learns from the samples in the support set $\mathcal{S}_\mathcal{T}$ and is evaluated on the query set $\mathcal{Q}_\mathcal{T}$. As a solution, meta-learning assumes that tasks are sampled from a distribution of tasks $p(\mathcal{T})$, and are split into meta-training and meta-test sets. The meta-learner learns the prior knowledge about the underlying structure of tasks using the meta-training set, such that later, the meta-test tasks can benefit from this prior knowledge. While early meta-learning algorithms directly minimize the average training error of a set of training tasks, Vinyals et al. [12] propose a novel training strategy called *episodic training*. Episodic training utilizes sampled mini-batches called *episodes* during training, where each episode is designed to mimic the few-shot task and generated by subsampling of classes and samples from the training dataset. The use of episodes improves the generalization by making the training process more faithful to the test environment and has become a dominant approach for a few-shot learning [37, 19, 11]. Meta-learning algorithms usually suppose the task distribution $p(\mathcal{T})$ is *unimodal*, meaning that all generated classification tasks belong to a single input-label domain (e.g. classification tasks with different combination of digits). Then, a *multimodal* counterpart can be considered where it contains classification tasks from multiple different input-label domains (e.g. few-shot digits classification and few-shot birds classification tasks).

4 Information Transfer Among Tasks

Transference in Episodic Training. The episodic training process can be viewed from a multi-task learning point of view, where multiple tasks (episodes) collaborate to build a shared feature representation that is broadly suitable for many tasks. Consequently, training episodes implicitly transfer information to each other by updating this shared feature representation with successive gradient updates. Then, the information transfer (transference) in episodic training can be viewed as the effect of gradients update from one episode (or a group of episodes) to the network parameters on the generalization performance on other episodes. Like multi-task learning framework, in an episodic

Algorithm 1: Measuring Transference on a Target Task.

Require: task distribution $p(\mathcal{T})$, learning rate α, current state of network parameters θ^t

Sample a batch of tasks $\xi = \{\mathcal{T}_i\} \sim p(\mathcal{T})$
Sample a target task $\mathcal{T}_j$
Use support set of target task $\mathcal{S}_j$ for adaption
Use query set $\mathcal{Q}_j$ to evaluate the adapted model and compute the loss $\mathcal{L}_{\mathcal{T}_j}(\mathcal{Q}_j; \theta^t, \mathcal{S}_j)$
for *all* $\mathcal{T}_i \in \xi$ **do**
  Use support set $\mathcal{S}_i$ to adapt to that task
  Use query set $\mathcal{Q}_i$ to evaluate the adapted model and compute the loss $\mathcal{L}_{\mathcal{T}_i}(\mathcal{Q}_i; \theta^t, \mathcal{S}_i)$
  Update model parameters with respect to task i using (1)
  Use $\mathcal{S}_j$ and $\mathcal{Q}_j$ to compute the loss of target task using updated parameters $\mathcal{L}_{\mathcal{T}_i}(\mathcal{Q}_i; \theta_i^{t+1}, \mathcal{S}_i)$
  Compute transference from task i to target task $LR_{i \to j}$ using (2)
end

training scenario, some learning episodes can be constructive regarding a target task and some can be destructive [38].

Analysis of Transference. To gain insights on the interaction between different episodes during episodic training, we adapt the *transference* idea [2] from multi-task learning to episodic training scenario. Consider a meta-learner parameterized by θ. At time-step t of episodic training, a batch of tasks are sampled from task distribution $p(\mathcal{T})$. Then for task i, the model uses its support set $\mathcal{S}_i$, to adapt to it, and the quality of adaption is evaluated by the loss on query set $\mathcal{Q}_i$ denoted by $\mathcal{L}_{\mathcal{T}_i}$. The total loss can be considered as sum of losses for tasks within batch. The quantity $\theta_{\text{ß}}^{t+1}$ is defined as updated model parameters after a SGD step with respect to task i:

$$\theta_i^{t+1} = \theta^t - \alpha \nabla_{\theta^t} \mathcal{L}_{\mathcal{T}_i}(\mathcal{Q}_i; \theta^t, \mathcal{S}_i) \tag{1}$$

We can use this quantity to calculate a loss which reflects the effect of task i on the performance of others. More specifically, in order to assess the effect of the gradient update of task i on a given target task j, we calculate the ratio between loss of task j before and after applying the gradient update on the shared parameters with respect to i:

$$LR_{i \to j} = \frac{\mathcal{L}_{\mathcal{T}_j}(\mathcal{Q}_j; \theta_i^{t+1}, \mathcal{S}_j)}{\mathcal{L}_{\mathcal{T}_j}(\mathcal{Q}_j; \theta^t, \mathcal{S}_j)} \tag{2}$$

In episodic training, the loss ratio $LR_{i \to j}$ can be considered as a measure of transference from meta-train task i to meta-test task j. If $LR_{i \to j}$ has a value smaller than one, the update on the network parameters results in a lower loss on target task j than the original parameter values, meaning that task i has a constructive effect on the generalization of the model on task j. On the other hand, if $LR_{i \to j}$ has a value greater than one, it indicates the destructive effect of task i on the target task. For $i = j$, the loss ratio denotes the *self-transference*, i.e. the effect of a task's gradient update in its own loss which can be used as a baseline for transference. The overall procedure of calculating transference within an episodic training framework is summarized in Algorithm 1.

Experiments. To investigate the interaction between tasks during training in a multimodal task distribution, we have sampled 300 meta-train tasks from the mini-ImageNet dataset as source tasks. Then we have analyzed the transference from these tasks to a single FC100 meta-test task using algorithm 1. For this experiment, we have used a ProtoNet [11] as a met-learner, and the analysis is performed in the middle of training on the combination of mini-ImageNet and FC100 datasets. The details of the experimental setup can be found in the supplementary material. The histogram of the transference is shown in Figure 1a which indicate both positive and negative knowledge transfer from mini-ImageNet tasks to the target task. An exemplar task for both positive and negative knowledge transfer is shown in figure 1. The task including animal classes has positive knowledge transfer to target task while the task including non-animal classes has negative transfer. The transference value for these two tasks during training (figure 1c) shows a consistent behavior in terms of knowledge transfer.

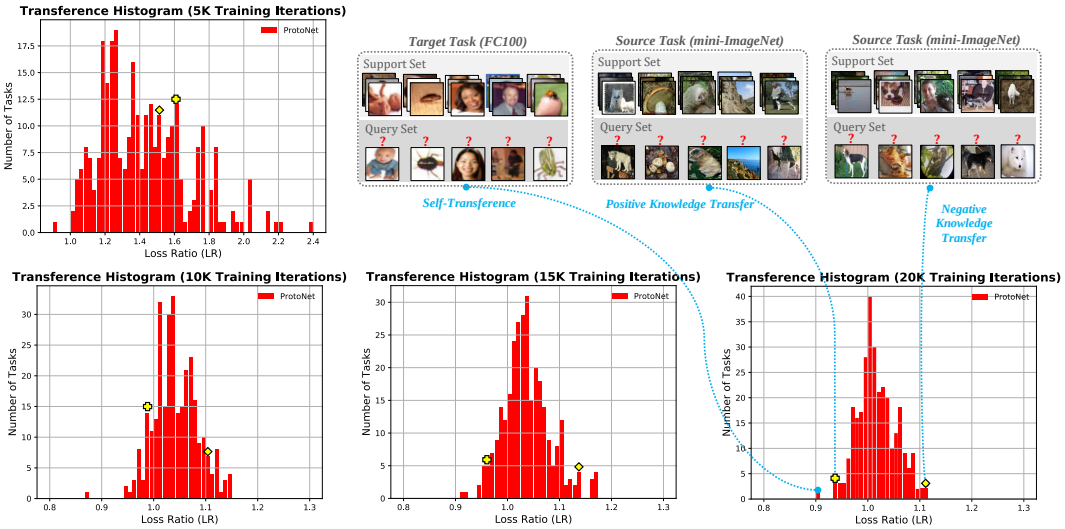


Figure 2: Transference from 300 mini-ImageNet meta-train tasks to a FC100 meta-test task. In the beginning of the training, when network learns low-level features, both tasks transfer negative knowledge. As the training proceeds, one becomes more and more positive, and then, consistently transfers positive knowledge to target task.

In figure 2, the transference to a different meta-test FC100 target task from mini-ImageNet meta-train tasks is shown. While the target task includes classification from people and insect classes, two source tasks with animal classes are among the best and worst knowledge transferring source tasks. This can be attributed to the quality of samples in these tasks. When a task includes noisy data samples, it is much harder to solve. With this point of view, the transference can also happen based on task hardness [39]. Considering two meta-train tasks both from animal classes, the one with lower hardness transfers better to a target task with lower hardness. The transference analysis can be extended to the unimodal scenario when some tasks within a single dataset (e.g., mini-ImageNet) can have a positive or negative impact on the others. More experiments can be found in the supplementary.

5 Proposed Multimodal Meta-Learner

In the previous section, the transference analysis shows that different tasks in episodic training can have a positive or negative impact on the learning process of other tasks. Recent works have analyzed the compatibility of tasks in multi-task learning [36, 2] during training to select some grouping of tasks for co-training that can improve performance. In our episodic training scenario, ideally, we would like to select the grouping of learning episodes that are compatible with the meta-test task. However, direct application of the methods in MTL to episodic training is not possible, due to two main reasons. *First*, the novel tasks in meta-testing are unseen and unknown during meta-training. *Second,* even if the tasks in meta-testing were known, it can be very computationally expensive to determine the group of cooperative tasks, and then assign a task-specific layer for each task, because episodic training involves tens of thousands of training tasks. Here, we propose a new interpretation of the modulation mechanism in MMAML. These lead us to propose pseudo-task-specific (task-aware) layers inspired by hard parameter sharing in MTL.

5.1 Multi-Modal Model-Agnostic Meta-Learning

To tackle multimodal few-shot tasks, MMAML [1] proposes a multimodal meta-learning framework which consists of a modulation network and a base network f_θ (Figure 3a). The modulation network, predicts the mode of a task and generates a set of task-specific modulation parameters to help the base network to better fit to the identified mode. It includes a task encoder network and a modulation parameter generator. Task encoder takes the support samples of a task $\mathcal{T}$ as input and produces an embedding vector $\boldsymbol{v}_\mathcal{T} = h_\varphi(\mathcal{S}_\mathcal{T})$ to encode its characteristics. Then, task-specific modulation parameters are generated using the embedding vector of task by the parameter generator network

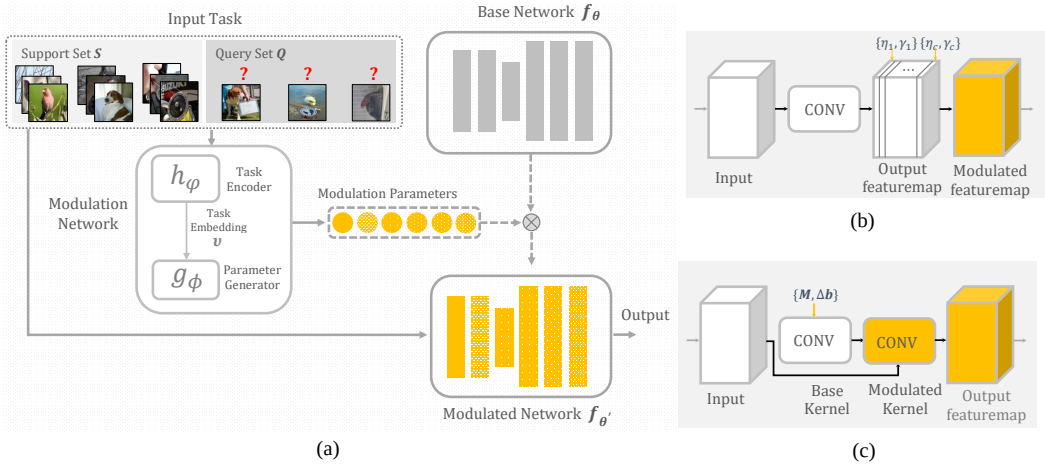


Figure 3: (a) General multimodal meta-Learning framework; Modulation scheme in (b) MMAML, (c) Proposed KML method.

$\boldsymbol{\omega}_{\mathcal{T}} = g_\phi(\boldsymbol{\upsilon}_{\mathcal{T}})$. Since MMAML uses feature-wise linear modulation (FiLM) [40], the generated modulation parameters are split into scaling and shifting parameters $\boldsymbol{\omega}_{\mathcal{T}} = \{\boldsymbol{\eta}_{\mathcal{T}}, \boldsymbol{\gamma}_{\mathcal{T}}\}$. Let $\mathbf{Y}_i$ denote the ith output channel of a layer in the base network f_θ. Then, the corresponding modulated output channel in MMAML is calculated as:

$$\hat{\mathbf{Y}}_i = \eta_i \mathbf{Y}_i + \gamma_i \tag{3}$$

where $\eta_i \in \boldsymbol{\eta}_{\mathcal{T}}$ and $\gamma_i \in \boldsymbol{\gamma}_{\mathcal{T}}$ are scalar values. For task encoder, MMAML uses a 4-layer convolutional network which is followed by multiple MLPs as parameter generator. MMAML uses the model-agnostic meta-learning (MAML) algorithm [19] as meta-learner in base network.

Limitations of Feature-Wise Modulation. We believe the major limitation of MMAML results from using FiLM modulation scheme. For each convolutional layer of a CNN, the ith output channel (feature map) is computed by convolving the ith kernel of that layer $\mathbf{W}_i$ (the layer index is removed for the sake of simplicity) with input to the layer $\mathbf{X}$ and adding the bias term b_i:

$$\mathbf{Y}_i = \mathbf{W}_i * \mathbf{X} + b_i \tag{4}$$

where $*$ denotes the convolution operator. Considering this, the modulation in (3) can be rewritten as:

$$\hat{\mathbf{Y}}_i = (\eta_i \mathbf{W}_i) * \mathbf{X} + (\eta_i b_i + \gamma_i) \tag{5}$$

We define the modulated kernel, and the modulated bias as $\hat{\mathbf{W}}_i = \eta_i \mathbf{W}_i$, and $\hat{b}_i = \eta_i b_i + \gamma_i$. Note that the modulated kernel is calculated by scaling the whole elements of the original kernel with a single scalar number. Then considering (3)-(5), the modulation in MMAML can be interpreted as convolving the input with this modulated kernel and adding the modulated bias term. The validity of this new interpretation is verified with experimental results (please refer to supplementary material). Based on this new interpretation, the modulated kernels used for different tasks are heavily bound together by $\mathbf{W}$ and are just scaled versions of it. Since the base kernel is shared across all tasks and scaling gives not many degrees of freedom for each task to produce its desired modulated kernel, different tasks fight for the model capacity, and the negative transfer between tasks from different distributions can lead to training a compromised base kernel.

5.2 Kernel Modulation

We propose to modulate the base network by changing the whole parameters within each layer using the modulation parameters conditioned on the task. This gives more capacity for each task to generate

Algorithm 2: Kernel Modulation Algorithm

Require: task distribution $p(\mathcal{T})$, learning rate α
Randomly initialize θ, φ, ϕ
while *not done* **do**
  Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
  for *all* $\mathcal{T}_i$ **do**
    Infer $\boldsymbol{v}_{\mathcal{T}_i} = h_\varphi(\mathcal{S}_i)$
    Generate $M^{(l)}_{\mathcal{T}_i}$ and $\Delta b^{(l)}_{\mathcal{T}_i}$ using (8) and (9)
    Modulate θ using (6) and (7) to obtain $\hat{\theta}_{\mathcal{T}_i}$
    Compute $\mathcal{L}_{\mathcal{T}_i}(\mathcal{Q}_i; \hat{\theta}_{\mathcal{T}_i}, \mathcal{S}_i)$ using $f_{\hat{\theta}_{\mathcal{T}_i}}$
  end
  Update $\theta \leftarrow \theta - \alpha \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(\mathcal{Q}_i; \hat{\theta}_{\mathcal{T}_i}, \mathcal{S}_i)$
  Update $\varphi \leftarrow \varphi - \alpha \nabla_\varphi \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(\mathcal{Q}_i; \hat{\theta}_{\mathcal{T}_i}, \mathcal{S}_i)$
  Update $\phi \leftarrow \phi - \alpha \nabla_\phi \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(\mathcal{Q}_i; \hat{\theta}_{\mathcal{T}_i}, \mathcal{S}_i)$
end

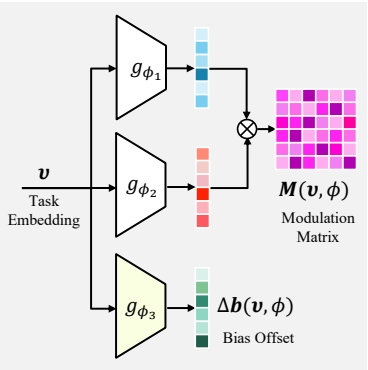


Figure 4: Simplified design for modulation parameter generator.

its desired kernel. To do this, for every single parameter within the network, a modulation parameter is generated. For a task $\mathcal{T}$, and for each layer $l = 1, \ldots, L$ of the network, let $\mathbf{M}^l(\boldsymbol{v}_\mathcal{T}, \phi)$ denote the modulation matrix generated for modulating the base kernel $\mathbf{W}^l$ of that layer. Among different design choices, intuitively, if we consider the modulated kernels as perturbations around the base kernel, by learning residual weights, a more powerful base kernel can be learned via enabling proper knowledge transfer across different modes of the task distribution. In our case, this can be achieved by adding the unit matrix $\mathbf{J}$ (all-ones matrix) to the modulation matrix and then perform Hadamard multiplication between the result and the base kernel as follows:

$$\hat{\mathbf{W}}^l_\mathcal{T} = \mathbf{W}^l \odot (\mathbf{J} + \mathbf{M}^l(\boldsymbol{v}_\mathcal{T}, \phi)) \tag{6}$$

For the bias of each layer we simply add an offset term $\Delta\mathbf{b}^l(\boldsymbol{v}_\mathcal{T}, \phi)$:

$$\hat{\mathbf{b}}^l_\mathcal{T} = \mathbf{b}^l + \Delta\mathbf{b}^l(\boldsymbol{v}_\mathcal{T}, \phi) \tag{7}$$

We denote the modulated parameters for task $\mathcal{T}$ as $\hat{\theta}_\mathcal{T}$. The modulation parameters for each task are generated through processing task embeddings of that task $\boldsymbol{v}_\mathcal{T}$ by parameter generator network $\mathbf{g}_\phi$ (Figure 3). Similar to [1], assuming $\mathbf{g}_\phi$ to be MLP, saves lots of computations where a separate parameter generation network $\mathbf{g}^l_\phi$ used for each layer. The main concern with using an MLP is the large number of parameters required in $\mathbf{g}^l_\phi$ to generate the modulation parameters $\Delta\mathbf{b}^l$ and $\mathbf{M}^l$. Because we want to produce the modulation parameters for the whole parameters of a deep neural network, we need to employ some sort of parameter reduction in $\mathbf{g}_\phi$ to prevent an explosion of parameters. Instead of following current complicated algorithms such as pruning the weights of network [41] or redesigning the operations [42], we propose a simple design for $\mathbf{g}_\phi$ which substitutes an MLP module with three smaller ones (Figure 4). In this design, the modulation matrix is generated by processing task embeddings $\boldsymbol{v}_\mathcal{T}$ using two of modules, and then performing the outer product operation on the output of these two modules:

$$\mathbf{M}^l(\boldsymbol{v}_\mathcal{T}, \phi) = \mathbf{g}^l_{\phi_1}(\boldsymbol{v}_\mathcal{T}) \otimes \mathbf{g}^l_{\phi_2}(\boldsymbol{v}_\mathcal{T}) \tag{8}$$

The produced matrix is then reshaped to match the kernel shape in that layer. The bias offset of each layer is generated by processing the embedding vector $\boldsymbol{v}_\mathcal{T}$ using the third module:

$$\Delta\mathbf{b}^l(\boldsymbol{v}_\mathcal{T}, \phi) = \mathbf{g}^l_{\phi_3}(\boldsymbol{v}_\mathcal{T}) \tag{9}$$

Our experiments show that the proposed method decreases the number of parameters in $\mathbf{g}_\phi$ by a factor of *150*, and also provides better generalization performance. A detailed discussion can be found in supplementary. The overall multimodal meta-learning algorithm using the proposed **K**ernel **M**odu**L**ation (KML) scheme is summarized in Algorithm 2.

Table 1: Meta-test accuracies on the multimodal few-shot image classification with 2, 3, and 5 modes. The accuracy values are mean of 1000 randomly generated test tasks, and the $\pm$ shows 95% confidence interval over tasks. 2Mode† and 2Mode indicate the combination of mini-ImageNet with FC100 and Omniglot, respectively. * Results produced by code provided in [1]. ** Our Implementation.

Setup		**Method**			
		MAML [19]*	**Multi-MAML**	**MMAML [1]***	**MMAML+KML (ours)**
2Mode†	1-shot	40.53±68%	39.27±0.76%	39.11±0.62%	**40.73±0.66%**
	5-shot	**54.11±0.63%**	53.51±0.72%	52.02±0.63%	53.72±0.60%
2Mode	1-shot	65.18±0.61%	66.77±0.68%	67.67±0.63%	**68.01±0.59%**
	5-shot	74.18±0.57%	73.07±0.61%	73.52±0.71%	**77.02±0.66%**
3 Mode	1-shot	54.40±0.56%	56.01±0.66%	57.35±0.61%	**57.68±0.59%**
	5-shot	66.51±0.54%	65.92±0.62%	64.21±0.57%	**67.12±0.55%**
5Mode	1-shot	47.19±0.49%	48.33±0.58%	49.53±0.50%	**50.31±0.49%**
	5-shot	58.13±0.48%	59.20±0.52%	58.89±0.47%	**60.51±0.47%**
		ProtoNet [11]**	**Multi-ProtoNet**	**MProtoNet [1]****	**MProtoNet+KML (ours)**
2Mode†	1-shot	43.05±0.58%	43.42±0.56%	43.57±0.59%	**44.40±0.65%**
	5-shot	57.70±0.59%	56.73±0.64%	56.03±0.64%	**59.31±0.62%**
2Mode	1-shot	69.55±0.54%	70.17±0.61%	70.60±0.56%	**73.69±0.52%**
	5-shot	75.12±0.41%	75.33±0.46%	75.72±0.47%	**79.82±0.40%**
3 Mode	1-shot	58.14±0.49%	59.89±0.50%	59.62±0.54%	**62.08±0.54%**
	5-shot	66.84±0.44%	67.03±0.44%	67.51±0.47%	**70.03±0.43%**
5Mode	1-shot	49.31±0.53%	50.69±0.57%	51.75±0.52%	**56.72±0.46%**
	5-shot	58.91±0.51%	59.88±0.54%	59.95±0.42%	**64.91±0.38%**

6 Experiments and Analysis

We evaluate the proposed model in both multimodal and unimodal few-shot learning scenarios. Following [1], to create a multimodal few-shot image classification meta-dataset, we combine multiple widely used datasets (Omniglot [43], mini-Imagenet [12], FC100 [44], CUB [45], and Aircraft [46]). Most of our experiments were performed by modifying the code accompanying [1], and for a fair comparison, we follow their experimental protocol unless specified. The details of all datasets, experimental setups, and network structure can be found in the supplementary material.

Multimodal Few-shot Classification Results. For multimodal few-shot classification, similar to [1], we train and evaluate models on the meta-datasets with two modes (Omniglot and mini-Imagenet), three modes (Omniglot, mini-Imagenet, and FC100), and five modes (all the five datasets). This meta-dataset includes a few-shot classification of characters, natural objects with different statistics and also fine-grained classification of different bird and aircraft types. Due to the large discrepancies between datasets, the possibility of negative transfer increases which makes few-shot classification more challenging. We also combine the mini-ImageNet and FC100 datasets to construct another two-mode meta-dataset. Both mini-ImageNet and FC100 include samples of natural objects and few-shot tasks generated from them probably have a similar underlying structure. However, the few-shot classification of FC-100 tasks is a bit challenging due to the smaller image size. So, here we aim to investigate the possibility of knowledge transfer from mini-ImageNet to FC100. We use two different meta-learners namely ProtoNet [11], and MAML [19], and several baselines considering each meta-learner. The first baseline is training and testing multiple separate meta-learners, one for each dataset. The second baseline is considered when meta-learner has no access to task mode information and simply trained and tested in the combination of tasks from different modes. The third baseline is the MMAML [1] which to the best of our knowledge is the only proposed algorithm for multimodal few-shot learning. A variant of this baseline (MProtoNet) is also used here by replacing the ProtoNet meta-learner with MAML in the general framework.

Table 2: Meta-test accuracies including the performance on each dataset.

Method	Datasets					
	mini-ImageNet		FC100		Overall	
	1shot	5shot	1shot	5shot	1shot	5shot
MAML1	41.70±0.91%	**60.03±0.88%**	—	—	39.27±0.86%	53.51±0.82%
MAML2	—	—	36.84±0.81%	47.08±0.74%		
MAML [19]	**44.11±0.88%**	59.34±0.78%	37.01±0.82%	**49.01±0.73%**	40.53±0.68%	**54.11±0.63%**
MMAML [1]	42.70±0.86%	56.93±0.78%	35.50±0.75%	47.11±0.77%	39.11±0.72%	52.02±0.63%
MAML+KML(ours)	44.01±0.82%	58.95±0.76%	**37.47±0.86%**	48.45±0.76%	**40.73±0.66%**	53.72±0.60%
ProtoNet1	47.19±0.58%	61.11±0.53%	—	—	43.42±0.56%	56.73±0.64%
ProtoNet2	—	—	39.61±0.54%	52.35±0.57%		
ProtoNet [11]	45.62±0.82%	**61.74±0.75%**	40.36±0.76%	53.66±0.77%	43.05±0.58%	57.70±0.59%
MProtoNet [1]	47.06± 0.80	60.56± 0.77	40.09± 0.77	51.50± 0.83	43.57± 0.59	56.03± 0.64
MProto+KML(ours)	**48.34± 0.86%**	61.20± 0.77%	**40.48± 0.77%**	**54.42± 0.76%**	**44.40± 0.65%**	**59.31± 0.62%**

Table 3: Meta-test accuracies on unimodal few-shot classification.

Method	Backbone	mini-ImageNet		tiered-ImageNet	
		1-shot	5-shot	1-shot	5-shot
ProtoNet [11]	Conv4-64	51.19± 0.45%	65.59± 0.28 %	52.83± 0.25%	70.41± 0.21%
MProtoNet [1]	Conv4-64	51.55±0.51%	65.83±0.36 %	53.01±0.33%	70.11±0.29%
MProtoNet+KML(ours)	Conv4-64	**54.10± 0.61%**	**67.31± 0.35%**	**54.67± 0.39%**	**72.09± 0.27%**

Multimodal few-shot image classification results are shown in Table 1. Comparing MMAML with MProtoNet, we can see that replacing ProtoNet with MAML considerably improves the classification accuracy. We believe that most of the extra accuracy gained in the multimodal framework by replacing MAML with ProtoNet results from the proper training of modulation network due to improved gradient flow from meta-learner. A detailed experiment can be found in supplementary. Additionally, as the results in Table 1 suggest, the proposed KML scheme considerably improves the performance of multimodal few-shot classification with both meta-learners. The improvement in *5-shot* scenarios is higher because more clues about the task provide more accurate embedding vectors to generate the modulation parameters. We have also included the detailed accuracy for 2Mode† scenario consisting of mini-ImageNet and FC100 in table 2. This shows on average a positive knowledge transfer from mini-ImageNet to FC100 dataset in terms of increased classification accuracy compared to a single meta-learner trained and tested on FC100.

Unimodal Few-shot Classification Results. While our KML scheme is designed for multimodal task distribution, it can also be used in conventional unimodal scenarios. In a conventional few-shot classification scenario there can also be negative transfer between tasks generated by sampling from classes with large discrepancy. We compare the performance of ProtoNet with MProtoNet and MProtoNet+KML on mini-ImageNet and tiered-ImageNet [47] datasets (Table 3). While MProtoNet gives not much improvement over ProtoNet, MProtoNet+KML provides evident improvement. These results recommend that increasing the capacity of learning for each task can also bring improvement to conventional meta-learning by reducing the fight between different tasks for capacity in shared parameters [48]. More results and details about the experimental setup can be found in supplementary.

7 Discussion and Conclusion

In this work, we propose a new quantification method for understanding knowledge transfer in multimodal meta-learning. We then propose a new interpretation of the modulation mechanism in MMAML. These lead us to propose a new multi-modal meta-learner inspired by hard parameter sharing in multi-task learning. Our extensive experimental results show that the proposed method achieves substantial improvement over the best results in multimodal meta-learning. While our major focus is on multimodal meta-learning, our work also attempts to shed light on conventional unimodal meta-learning. Limitations are discussed in the supplementary.

References

[1] Risto Vuorio, Shao-Hua Sun, Hexiang Hu, and Joseph J. Lim. Multimodal model-agnostic meta-learning via task-aware modulation. In *Neural Information Processing Systems*, 2019.

[2] Christopher Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. Measuring and harnessing transference in multi-task learning. *arXiv preprint arXiv:2010.15413*, 2020.

[3] Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[4] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[5] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

[6] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[7] Touba Malekzadeh, Milad Abdollahzadeh, Hossein Nejati, and Ngai-Man Cheung. Aircraft fuselage defect detection using deep neural networks. *arXiv preprint arXiv:1712.09213*, 2017.

[8] Xiaopeng Yan, Ziliang Chen, Anni Xu, Xiaoxi Wang, Xiaodan Liang, and Liang Lin. Meta r-cnn: Towards general solver for instance-level low-shot learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[9] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *International Conference on Learning Representations*, 2019.

[10] Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

[11] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 4080–4090, 2017.

[12] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 3637–3645, 2016.

[13] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1199–1208, 2018.

[14] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *International Conference on Learning Representations*, 2019.

[15] Seong Min Kye, Hae Beom Lee, Hoirin Kim, and Sung Ju Hwang. Meta-learned confidence for few-shot learning. *arXiv e-prints*, pages arXiv–2002, 2020.

[16] Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Few-shot learning with embedded class models and shot-free meta training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 331–339, 2019.

[17] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017.

[18] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *International Conference on Machine Learning*, pages 2554–2563. PMLR, 2017.

[19] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[20] Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. In *International Conference on Learning Representations*, 2018.

[21] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

[22] Sungyong Baik, Seokil Hong, and Kyoung Mu Lee. Learning to forget for meta-learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[23] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. 2017.

[24] Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3018–3027, 2017.

[25] Siyuan Qiao, Chenxi Liu, Wei Shen, and Alan L Yuille. Few-shot image recognition by predicting parameters from activations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7229–7238, 2018.

[26] Yiluan Guo and Ngai-Man Cheung. Attentive weights generation for few shot learning via information maximization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13499–13508, 2020.

[27] Richard Caruana. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 41–48. Morgan Kaufmann, 1993.

[28] Jonathan Baxter. A model of inductive bias learning. *Journal of artificial intelligence research*, 12:149–198, 2000.

[29] Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd annual meeting of the Association for Computational Linguistics and the 7th international joint conference on natural language processing (volume 2: short papers)*, pages 845–850, 2015.

[30] Yongxin Yang and Timothy M Hospedales. Trace norm regularised deep multi-task learning. *arXiv preprint arXiv:1606.04038*, 2016.

[31] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3994–4003, 2016.

[32] Pengsheng Guo, Chen-Yu Lee, and Daniel Ulbricht. Learning to branch for multi-task learning. In *International Conference on Machine Learning*, pages 3854–3863. PMLR, 2020.

[33] Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. Adashare: Learning what to share for efficient deep multi-task learning. *arXiv preprint arXiv:1911.12423*, 2019.

[34] Simon Vandenhende, Stamatios Georgoulis, Bert De Brabandere, and Luc Van Gool. Branched multi-task networks: deciding what layers to share. *arXiv preprint arXiv:1904.02920*, 2019.

[35] Joachim Bingel and Anders Søgaard. Identifying beneficial task relations for multi-task learning in deep neural networks. *arXiv preprint arXiv:1702.08303*, 2017.

[36] Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In *International Conference on Machine Learning*, pages 9120–9132. PMLR, 2020.

[37] Jiaxin Chen, Xiao-Ming Wu, Yanke Li, Qimai Li, Li-Ming Zhan, and Fu-lai Chung. A closer look at the training strategy for modern meta-learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[38] Shengchao Liu, Yingyu Liang, and Anthony Gitter. Loss-balanced task weighting to reduce negative transfer in multi-task learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9977–9978, 2019.

[39] Anh T Tran, Cuong V Nguyen, and Tal Hassner. Transferability and hardness of supervised classification tasks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1395–1405, 2019.

[40] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[41] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *arXiv preprint arXiv:2003.03033*, 2020.

[42] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[43] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the annual meeting of the cognitive science society*, volume 33, 2011.

[44] Boris N Oreshkin, Pau Rodriguez, and Alexandre Lacoste. Tadam: task dependent adaptive metric for improved few-shot learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 719–729, 2018.

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

[46] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[47] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. In *International Conference on Learning Representations*, 2018.

[48] Zirui Wang, Zachary C Lipton, and Yulia Tsvetkov. On negative interference in multilingual language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4438–4450, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] *Please see supplementary material.*
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] *Please see supplementary material.*
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] *Please see Tables 1. All the results of the few-shot classification are provided by a 95% confidence interval.*
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] *All results are obtained by a machine with a single GPU. Please see supplementary material for more details.*
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] *The datasets are cited in the main body of the paper. For code, please see supplementary material.*
   (b) Did you mention the license of the assets? [Yes] *Please see supplementary material.*
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] *Please see supplementary material.*
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] *Please see supplementary material.*

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] *All assets are publicly available and contain no personally identifiable information or offensive content.*

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]