Meta-Learning from Multimodal Task Distributions Using Multiple Sets of Meta-Parameters

Anna Vettoruzzo, Mohamed-Rafik Bouguelia, Thorsteinn Rögnvaldsson
Center for Applied Intelligent Systems Research (CAISR), Halmstad University, Sweden
{anna.vettoruzzo, mohamed-rafik.bouguelia, thorsteinn.rognvaldsson}@hh.se

***Abstract*—Meta-learning or learning to learn involves training a model on various learning tasks in a way that allows it to quickly learn new tasks from the same distribution using only a small amount of training data (i.e., few-shot learning). Current meta-learning methods implicitly assume that the distribution over tasks is unimodal and consists of tasks belonging to a common domain, which significantly reduces the variety of task distributions they can handle. However, in real-world applications, tasks are often very diverse and come from multiple different domains, making it challenging to meta-learn common knowledge shared across the entire task distribution. In this paper, we propose a method for meta-learning from a multimodal task distribution. The proposed method learns multiple sets of meta-parameters (acting as different initializations of a neural network model) and uses a task encoder to select the best initialization to fine-tune for a new task. More specifically, with a few training examples from a task sampled from an unknown mode, the proposed method predicts which set of meta-parameters (i.e., model's initialization) would lead to a fast adaptation and a good post-adaptation performance on that task. We evaluate the proposed method on a diverse set of few-shot regression and image classification tasks. The results demonstrate the superiority of the proposed method compared to other state-of-the-art meta-learning methods and the benefit of learning multiple model initializations when tasks are sampled from a multimodal task distribution.**

***Index Terms*—Meta-Learning, Few-Shot Learning, Transfer Learning, Task Representation, Multimodal Distribution**

I. Introduction

Human learning relies on accumulating knowledge across different experiences and taking advantage of this knowledge to solve new tasks efficiently. To do so, the human mind selects only the relevant information and adapts it to the specific task. Inspired by this approach, meta-learning methods aim to acquire general knowledge from multiple tasks to efficiently adapt to new, unseen tasks. This is typically achieved with neural networks by learning a representation that is easily adaptable for new tasks using a limited amount of labeled samples. Most existing methods (such as [1]–[3]) learn a single representation globally shared across all tasks, implicitly assuming that the task distribution is unimodal or that tasks are all closely related to the same application domain (e.g., classifying digits from different alphabets). However, real-world tasks are generally diverse and sampled from a more complex task distribution consisting of multiple unknown modes (either overlapping or far apart), thus requiring different representations. As an analogy, humans can learn a new task, e.g., a novel ice skating skill, exploiting not only the fundamental ice skating knowledge (i.e., tasks from the same mode) but also previous experience from roller skating, skiing, and dancing (i.e., related tasks from other modes); however, they may not benefit from knowledge gained through tasks related to singing or cooking (i.e., modes that are disjoint from the previous ones).

Model-Agnostic Meta-Learning (MAML) [1] is one of the most successful meta-learning methods. It learns a representation by optimizing the parameters of a neural network such that adapting (or fine-tuning) them on any training task would yield a good generalization on this latter. As such, the learned representation consists of a parameters' initialization that is easily adaptable to new tasks. Other methods, such as Reptile [2], follow a similar principle by finding an initialization that is close to each task's optimal parameters. While these methods work well when tasks are sampled from a unimodal distribution, they lose efficiency when dealing with heterogeneous tasks drawn from a multimodal task distribution. Indeed, learning a single model initialization, shared across the entire task distribution, may not be sufficient to capture the heterogeneity in the tasks and to achieve fast and efficient adaptation to new tasks. This is also true for other meta-learning methods such as [3]–[11]. One general way to address this issue is to learn a distinct initialization for tasks within each separate mode. This could lead to improved results compared to using one single global initialization. However, this requires knowing the mode of each task (i.e., the ground-truth task mode label), which is not feasible in real-world scenarios. Furthermore, this method hinders the transfer of useful information between tasks from related modes.

This paper proposes extending traditional meta-learning approaches to address meta-learning in cases where tasks are sampled from a multimodal distribution with unknown modes. To do so, the proposed method trains **mu**ltiple **se**ts of meta-parameters (hence the name MUSE), which would, for example, correspond to multiple model initializations in the case of MAML and Reptile. The proposed method also trains a task encoder network that takes training examples from a task, learns an embedding of the task, and uses it to predict the best initialization to fine-tune on that task (i.e., it selects the initialization that leads to the lowest generalization error after being adapted for that task).

Recently, a few approaches have been proposed to address meta-learning from multimodal task distributions. Some of these approaches [12], [13] learn to cluster tasks (based on

some similarity measure between tasks) or perform clustering in the parameter space [14], [15], while others [16], [17] learn to directly modulate the model parameters based on the task at hand. However, these approaches are difficult to train and highly dependent on the selection of various hyperparameters. Some hybrid methods [18]–[23] have also been proposed to solve few-shot classification problems by learning a universal representation that works well for tasks drawn from different datasets (corresponding to different modes) and using different techniques to specialize it towards each new task. However, learning such a representation beforehand is challenging and may lead to poor performance on completely different datasets.

Our main contribution is a framework that learns multiple model initializations and, when presented with a few training examples from a new task, selects the initialization that will result in a good performance after adaptation to that task. This approach allows the model to only consider relevant information for the task at hand, thereby speeding up the learning process and improving the accuracy of predictions. The experimental evaluation shows that the proposed approach outperforms traditional approaches where a single initialization is learned across all tasks. Despite its simplicity, it also outperforms other existing methods designed for meta-learning from multimodal task distributions.

II. Background and Related Work

Meta-learning, or learning-to-learn, has gained significant attention recently due to its effectiveness in solving few-shot classification, regression, and reinforcement learning problems. To perform meta-training, a collection of training tasks $\{\mathcal{T}_i\}_{i=1}^{T}$ is sampled from a task distribution $\mathrm{P}(\mathcal{T})$. Each task $\mathcal{T}_i \sim \mathrm{P}(\mathcal{T})$ corresponds to data generating distributions $\mathcal{T}_i \triangleq \{p_i(x), p_i(y|x)\}$. The data generated from a training task $\mathcal{T}_i$ are split into a small set of K training examples called *support set* or $\mathcal{D}_i^{(sp)}$, and a test set called *query set* or $\mathcal{D}_i^{(qr)}$ which is typically large. At meta-test time, a completely new task $\mathcal{T}_{new}$ is sampled from $\mathrm{P}(\mathcal{T})$, and only a few training examples are observed in the form of a support set $\mathcal{D}_{new}^{(sp)} \triangleq \{x_k, y_k\}_{k=1}^{K}$. The goal is to train a model on $\mathcal{D}_{new}^{(sp)}$ while leveraging and adapting the prior knowledge gained during meta-training in order to achieve a good generalization performance on unseen test examples from $\mathcal{T}_{new}$.

Optimization-based meta-learning methods (such as MAML [1]) use a bi-level optimization process to embed learning procedures, such as gradient descent, into the meta-optimization problem. By doing so, MAML meta-learns an initial set of parameters θ that can be efficiently adapted (or fine-tuned) on new tasks. Specifically, MAML meta-trains a model parameterized by θ via a two-stage procedure consisting of an inner and outer loop. For each training task $\mathcal{T}_i$, the inner loop adapts the initial set of parameters θ to obtain a set of task-specific parameters ϕ_i, by taking $Q \geq 1$ (i.e., one or a few) gradient descent steps on the support set $\mathcal{D}_i^{(sp)}$. This is illustrated in Eq. 1 when a single gradient step is used:

$$\phi_i = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_i^{(sp)}), \tag{1}$$

where $\mathcal{L}(.,.)$ denotes the loss function. Then, the outer loop optimizes the initial parameters θ to minimize the post-adaptation loss (achieved by task-specific parameters ϕ_i) on the query sets. This is illustrated in Eq. 2:

$$\theta \leftarrow \theta - \epsilon \nabla_\theta \sum_{\mathcal{T}_i \sim \mathrm{P}(\mathcal{T})} \mathcal{L}(\phi_i, \mathcal{D}_i^{(qr)}). \tag{2}$$

The result is a model initialization θ that can effectively be adapted to new tasks using only a few (K) training examples and a few gradient updates.

One simpler and less computationally expensive variant is Reptile [2], which aims to learn an initial set of parameters θ that is close to each task's optimal parameters, thus, easily adaptable to new tasks. To do so, Reptile repeatedly samples a training task $\mathcal{T}_i \sim \mathrm{P}(\mathcal{T})$, performs $Q > 1$ gradient descent steps on $\mathcal{T}_i$ (starting with parameters θ and resulting in task-specific parameters ϕ_i), then moves θ closer to ϕ_i by interpolating between the two as shown in Eq. 3:

$$\theta \leftarrow \theta + \epsilon(\phi_i - \theta). \tag{3}$$

Despite their success, the ability of these methods to handle more general meta-learning problems has recently been questioned, as a single initialization θ may not be enough, especially when tasks are sampled from a diverse, wide, or multimodal task distribution $\mathrm{P}(\mathcal{T})$. This has led to the development of variants that incorporate task-specific information [12]–[17]. One such approach, MMAML [16] tries to identify the mode of tasks sampled from a multimodal task distribution $\mathrm{P}(\mathcal{T})$ and modulates the meta-parameters θ according to the identified mode. However, training such a model end-to-end is challenging, and its performance highly depends on the number of modes. Another approach, TSA-MAML [15] uses a vanilla MAML and clusters task-specific parameters by applying k-means in the parameters space, with the number of clusters being the same as the number of modes. Each cluster centroid serves as a group-specific initialization. However, this centroid-based clustering fails to take advantage of negative correlations between tasks (e.g., **w** and **-w** may be assigned to different clusters) and fails to handle tasks that are distant from all clusters. CAVIA [17] separates the initial set of parameters into parameters that are shared across all tasks and context parameters that are specific to individual tasks. At meta-test time, only the context parameters are adapted for each new task. This approach has inspired a series of related works aiming to build a "universal representation", i.e., a set of robust features that lead to strong performance across multiple datasets in a multi-task learning setup [24]. Building on this, the authors in [18]–[20] propose to use meta-learning to specialize the universal representation towards each new task. However, learning such a representation in advance is challenging and may result in overfitting. To overcome these issues, SUR [21] and URL [22] train a separate feature extractor for each dataset (or mode) and combine the learned representations to solve a new task at test time. Similarly, FLUTE [23] aims to learn a shared model across all datasets while allowing for specialization to each individual dataset by

learning a small set of parameters specific to each dataset. However, these approaches are implemented only for few-shot classification problems, and they don't benefit from meta-learning to adapt fast, i.e., using a few adaptation steps at test time.

Inspired by these ideas, we introduce in the following section a method that addresses the scenario where the distribution over tasks is multimodal, i.e., consists of tasks derived from different datasets (corresponding to the modes). Our proposed method operates under the general assumption that the mode from which each task is sampled is unknown, both during meta-training and at meta-test time. This problem is closely related to the one addressed by MMAML [16] and TSA-MAML [15], hence the comparison with these two methods in the experiments. It is also worth noting that this is different from multimodality in data type [25], where tasks represent the same concept but in different modalities, such as a combination of images and text.

III. Proposed Approach

As previously discussed, the proposed MUSE approach considers tasks sampled from a multimodal task distribution $\mathrm{P}(\mathcal{T})$ and trains N distinct sets of meta-parameters $\{\theta_n\}_{n=1}^{N}$ that serve as initializations for the base model. Therefore, the question arises: *Given a task $\mathcal{T}$ with support set $\mathcal{D}^{(sp)}$, which one of the N initializations is best suited for adapting to task $\mathcal{T}$* ? At meta-training time, a large enough query dataset $\mathcal{D}_i^{(qr)}$ can be generated from each training task $\mathcal{T}_i$. These data can be used to evaluate each initialization θ_n by adapting it on $\mathcal{D}_i^{(sp)}$ and computing the post-adaptation loss on $\mathcal{D}_i^{(qr)}$, as described in subsection III-A. However, at meta-test time, only a small support set $\mathcal{D}_{new}^{(sp)}$ is available from a task $\mathcal{T}_{new}$, and it can't be used to compute a reliable post-adaptation loss (e.g., using cross-validation) due to its small size. To address this issue and answer the previous question, the proposed approach trains a task encoder that learns a task embedding and uses it to predict which one of the N initializations is the best to fine-tune for the given task. Figure 1 presents a conceptual illustration of the task encoder, which is further explained in subsection III-B.

A. Meta-learning multiple model initializations

Algorithm 1 meta-trains N sets of meta-parameters $\{\theta_n\}_{n=1}^{N}$ with the purpose of using them as initializations[1] of the model parameters at meta-test time. At lines 3-4 of Algorithm 1, a training task $\mathcal{T}_i$ (or a mini-batch of training tasks) is sampled from a multimodal distribution $\mathrm{P}(\mathcal{T})$, with its associated support set $\mathcal{D}_i^{(sp)}$ and query set $\mathcal{D}_i^{(qr)}$. At line 5, the RANK function (Algorithm 2) is called to assign a rank r_n to each set of meta-parameters θ_n with respect to the current task $\mathcal{T}_i$. Each rank[2] r_n represents how good the corresponding θ_n is if we fine-tune it on a small dataset from task $\mathcal{T}_i$. This is done in Algorithm 2 by adapting θ_n on $\mathcal{D}_i^{(sp)}$ via a small number, L, of gradient descent steps. The generalization loss of the adapted parameters on $\mathcal{D}_i^{(qr)}$ is then computed and used to assign a rank r_n to each θ_n (with the best being at rank 0 and the worst at rank $N-1$). At this stage, one alternative is to only pick the top-ranked θ_n (having rank $r_n = 0$) to update it. However, a more general approach is to update all N meta-parameters to different degrees using different step sizes ε_n. To do so, at line 7 of Algorithm 1, we define $\varepsilon_n = \varepsilon \cdot e^{-r_n/\lambda}$, where ε is a fixed number, and $\lambda > 0$ is the so-called neighborhood range as in Neural Gas [26]. The specific approach that picks only the top-ranked set of meta-parameters, can be obtained by setting λ close to 0. This results in a nearly zero value for ε_n for all sets of meta-parameters except for the top-ranked one, which would have $\varepsilon_n = \varepsilon$. At line 8, task-specific parameters[2], ϕ_n, are computed for each θ_n independently with a few, i.e., Q, adaptation steps, and each set of meta-parameters θ_n is updated at line 9 similarly to the standard approach (according to Eq. 2), using ε_n instead of ϵ.

It is worth noting that Algorithm 1 outlines the case in which the proposed approach is applied on top of MAML, but the same concept can be easily applied to Reptile with the following minor changes: (1) calculating the task-specific parameters in line 8 starting from the best-ranked set of meta-parameters θ^* as $\phi \leftarrow \mathrm{ADAPT}(Q, \mathcal{D}_i, \theta^*)$; and (2) updating each θ_n in line 9 according to Eq. 3 using ε_n instead of ϵ. Using the best initialization θ^* at line 8 is desirable; however, it can be done with Reptile but not with MAML since the latter involves second-order derivatives, requiring backpropagating through the gradient operator computed in the adaptation phase.

Algorithm 1 Training Multiple Sets of Meta-Parameters

Require: Number of sets of meta-parameters N, adaptation steps to rank the meta-parameters L, adaptation steps Q

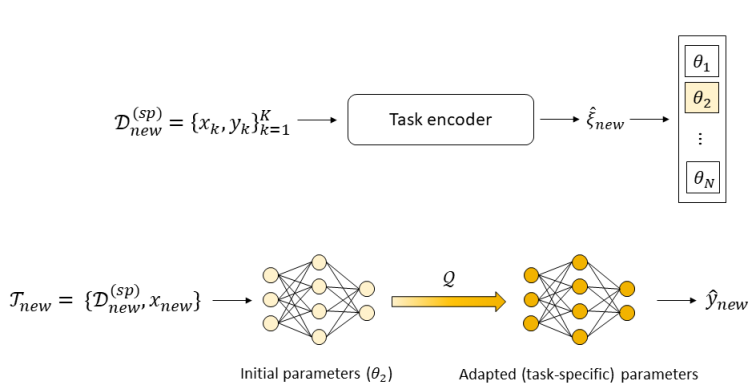
1: Randomly initialize $\{\theta_n\}_{n=1}^{N}$
2: **while** not done **do**
3: Sample a task $\mathcal{T}_i \sim \mathrm{P}(\mathcal{T})$ (or a mini-batch of tasks)
4: Sample data $\mathcal{D}_i = \mathcal{D}_i^{(sp)} \cup \mathcal{D}_i^{(qr)}$ from $\mathcal{T}_i$
5: $\{r_n\}_{n=1}^{N} \leftarrow \mathrm{RANK}(L, \mathcal{D}_i^{(sp)}, \mathcal{D}_i^{(qr)}, \{\theta_n\}_{n=1}^{N})$
6: **for each** set of meta-parameters θ_n **do**
7: Define a step size $\varepsilon_n = \varepsilon \cdot e^{-r_n/\lambda}$
8: Compute $\phi_n \leftarrow \mathrm{ADAPT}(Q, \mathcal{D}_i^{(sp)}, \theta_n)$
9: Update $\theta_n \leftarrow \mathrm{UPDATE}(\mathcal{D}_i^{(qr)}, \varepsilon_n, \theta_n, \phi_n)$
10: **end for**
11: **end while**
12: **return** $\{\theta_n\}_{n=1}^{N}$

B. Task-encoder training

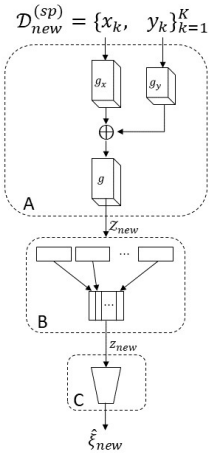
At meta-training time, we have access to support sets (small training sets of size K) but also to query sets (large enough test sets) generated from the training tasks. These are used in Algorithm 2 to evaluate and rank the N sets of meta-parameters with respect to each task. However, this is not

[1] Therefore, the terms "initialization" and "set of meta-parameters" will be used interchangeably to refer to θ_n.

[2] Normally, an additional subscript i is used in r_n and ϕ_n to indicate that these are specific for task $\mathcal{T}_i$. However, here, the subscript i is omitted for simplicity.



(a) Illustration of *MUSE*.



(b) Task encoder architecture

Fig. 1: (a) Illustration of *MUSE*. (Top) The task encoder takes in input K training examples from a new task $\mathcal{T}_{new}$ (i.e., $\mathcal{D}_{new}^{(sp)}$) and predicts as output $\hat{\xi}_{new}$ indicating which of the N meta-parameters (e.g., θ_2) constitutes the best initialization to adapt on task $\mathcal{T}_{new}$. (Bottom) The model's parameters are then initialized to θ_2 and adapted with $\mathcal{D}_{new}^{(sp)}$ using Q adaptation steps. The adapted parameters can then be used to make accurate predictions on new test examples x_{new}. (b) Task encoder architecture which consists of a feature extractor (A), an averaging layer (B), and a classifier (C).

Algorithm 2 RANK(L, $\mathcal{D}_i^{(sp)}$, $\mathcal{D}_i^{(qr)}$, $\{\theta_n\}_{n=1}^N$)

Require: Adaptation steps to rank the meta-parameters L, support and query sets of task $\mathcal{T}_i$, sets of meta-parameters $\{\theta_n\}_{n=1}^N$

1: **for each** set of meta-parameters θ_n **do**
2: Compute $\psi_n \leftarrow$ ADAPT$(L, \mathcal{D}_i^{(sp)}, \theta_n)$
3: Compute post-adaptation loss: $l_n = \mathcal{L}(\psi_n, \mathcal{D}_i^{(qr)})$
4: **end for**
5: Sort $\{l_n\}_{n=1}^N$ in ascending order
6: Assign a rank r_n to each θ_n based on sorted losses
7: **return** $\{r_n\}_{n=1}^N$

Algorithm 3 ADAPT(Q, $\mathcal{D}$, θ)

Require: Adaptation steps Q, dataset $\mathcal{D}$, initial parameters θ, learning rate α

$\phi \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D})$
for $q = 1, \ldots, Q-1$ **do**
$\phi \leftarrow \phi - \alpha \nabla_\phi \mathcal{L}(\phi, \mathcal{D})$
end for
return ϕ

possible to do at meta-test time since only K examples are observed from a completely new test task. To address this issue, a task encoder network, parameterized by ρ, is trained to predict which set of meta-parameters provides the lowest generalization error on the target task. As shown in Figure 1b, the task encoder architecture consists of three parts: a feature extractor (A), an averaging layer (B), and a classifier (C).

Specifically, the feature extractor takes as input a support set $\mathcal{D}_i^{(sp)} = \{x_k, y_k\}_{k=1}^K$ consisting of inputs x_k and labels y_k (represented as one-hot-vectors in classification) sampled from a task $\mathcal{T}_i$. It then transforms them to get $\mathcal{Z}_i = \{g(\ g_x(x_k) \ \oplus \ g_y(y_k)\ )\}_{k=1}^K$, where $g_x(.)$ denotes an input-specific feature extractor, $g_y(.)$ a label-specific feature extractor, and $\oplus$ denotes the concatenation operator. In other words, it transforms each sample $\mathbf{x}_k = g_x(x_k)$ and label $\mathbf{y}_k = g_y(y_k)$, then concatenates and transforms them as $g(\mathbf{x}_k \oplus \mathbf{y}_k)$. The averaging layer simply performs an averaging of the transformed data $\mathcal{Z}_i$ along the K examples to produce a vector representation z_i corresponding to an embedding of the task $\mathcal{T}_i$. Finally, the last part of the task encoder network acts as an N-class classifier with a softmax activation at the final layer. It takes as input a task's embedding z_i and outputs an N-dimensional vector $\hat{\xi}_i \in [0,1]^N$ indicating which one of the N initializations is the best for task $\mathcal{T}_i$ (i.e., ranked at the top).

The task encoder uses the meta-learned initializations $\{\theta_n\}_{n=1}^N$ (returned by Algorithm 1) and is trained as described in Algorithm 4. At lines 3-7, a mini-batch of training tasks $\{\mathcal{T}_1, \mathcal{T}_2, \ldots\}$ is sampled, and new labels $\{\xi_1, \xi_2, \ldots\}$ are computed and assigned to them. Each label $\xi_i \in \{1, \ldots, N\}$ corresponds to the index of the top-ranked initialization for the task $\mathcal{T}_i$ (i.e., the one ranked at the top according to Algorithm 2). The mini-batch used to train the task encoder is then presented at line 9, with inputs corresponding to the support sets $\{\mathcal{D}_1^{(sp)}, \mathcal{D}_2^{(sp)}, \ldots\}$ and outputs corresponding to the labels $\{\xi_1, \xi_2, \ldots\}$. The task encoder's parameters ρ are then updated at line 10 by minimizing a classification loss computed using the outputs $\{\hat{\xi}_1, \hat{\xi}_2, \ldots\}$ predicted by the task encoder and the actual labels $\{\xi_1, \xi_2, \ldots\}$. Here, any optimizer of choice, e.g., Adam, can also be used (not necessarily gradient descent).

Algorithm 4 Task Encoder Training

Require: Learning rate η, adaptation steps to rank the meta-parameters L, meta-learned initializations $\{\theta_n\}_{n=1}^N$ (returned by Algorithm 1)

1: Randomly initialize ρ
2: **while** not done **do**
3: Sample a mini-batch of tasks $\{\mathcal{T}_1, \mathcal{T}_2, \dots\}$ from $\mathrm{P}(\mathcal{T})$
4: **for** all $\mathcal{T}_i$ **do**
5: Sample disjoint datasets $\mathcal{D}_i^{(sp)}, \mathcal{D}_i^{(qr)}$ from $\mathcal{T}_i$
6: $\{r_n\}_{n=1}^N \leftarrow \mathrm{RANK}(L, \mathcal{D}_i^{(sp)}, \mathcal{D}_i^{(qr)}, \{\theta_n\}_{n=1}^N)$
7: Assign a label $\xi_i = \arg\min_n \{r_n\}$
8: **end for**
9: Mini-batch of training data for the task encoder: $\mathbf{D} = \{(\mathcal{D}_1^{(sp)}, \xi_1), (\mathcal{D}_2^{(sp)}, \xi_2), \dots\}$
10: Update $\rho \leftarrow \rho - \eta \nabla_\rho \mathcal{L}(\rho, \mathbf{D})$
11: **end while**
12: **return** ρ

C. Meta-test time

At meta-test time, only a set $\mathcal{D}_{new}^{(sp)}$ of K training examples, is observed from a completely new task $\mathcal{T}_{new} \sim \mathrm{P}(\mathcal{T})$. The goal is to efficiently train a model that would generalize well to future unseen examples from that task. To do so, the task encoder takes $\mathcal{D}_{new}^{(sp)}$ as input and predicts the label $\hat{\xi}_{new}$ corresponding to the best initialization (θ_{best}) for the task $\mathcal{T}_{new}$. The ADAPT function (Algorithm 3) is then called using a few (i.e., Q) adaptation steps as follows $\mathrm{ADAPT}(Q, \mathcal{D}_{new}^{(sp)}, \theta_{best})$, to get the task-specific parameters. These latter can then be used to make accurate predictions on unseen data from this new task.

IV. Experiments

In this section, the proposed MUSE approach, applied on top of MAML (MUSE-M) and Reptile (MUSE-R), is evaluated using tasks from various few-shot regression and image classification problems. The proposed MUSE-M and MUSE-R are compared against various methods listed below:

- MAML [1] and Reptile [2]: The respective unimodal counterparts of MUSE-M and MUSE-R.
- MMAML [16] and TSA-MAML [15]: Two existing methods for meta-learning from multimodal task distributions.
- Multi-MAML and Multi-Reptile: A trivial extension of MAML and Reptile for multimodal task distributions. This baseline consists of meta-training a distinct model (using MAML or Reptile) for tasks within each separate mode. At meta-test time, the mode of each new task is also assumed to be known, and it is used to select the corresponding initial model. Note that directly comparing the previous approaches to Multi-MAML (or Multi-Reptile) is not fair as this latter uses additional information (mode label of each task) that is usually unavailable in real-world situations. However, this baseline can provide useful insights into whether or not it is helpful to transfer knowledge across different modes.
- "Scratch": A naive approach that consists of training a model on each new task from scratch, i.e., with a random parameters' initialization instead of meta-learning it. This baseline is used as a lower bound on the performance.
- "Oracle": A baseline proposed to verify the performance of MUSE in the ideal case in which the task encoder always predicts the best model initialization (among the N initializations) for any given task. This oracle baseline is not intended to be compared against the proposed approach, but it serves as an upper bound for its performance.

All methods are evaluated using 20 test tasks from each mode, and the average performance is computed after fine-tuning the meta-learned initializations for 100 adaptation steps. To ensure a fair and consistent comparison, we used similar hyperparameters and model architectures for all methods. The final reported results are the average over three full runs of all the algorithms (including meta-training and meta-testing).

A. Regression Problem

As a proof of concept, we start with a simple regression problem where a multimodal task distribution is constructed considering five different families of functions from which tasks are generated. These families are: (1) sinusoidal $y(x) = a\sin(x-b)$, where $a \sim U[0.1, 5.0]$, $b \sim U[0, \pi]$; (2) linear $y(x) = ax + b$, where $a \sim U[0,1]$, $b \sim U[0,5]$; (3) quadratic $y(x) = ax^2 + bx + c$, where $a, b, c \sim U[0, 0.5]$; (4) l1 norm $y(x) = a|x - c| + b$, where $a, b, c \sim U[0, 0.5]$; (5) hyperbolic tangent $y(x) = a\tanh(x-c) + b$, where $a, b, c \sim U[0, 0.5]$. Each task is randomly sampled from one of the five underlying families and consists of inputs x sampled uniformly in $[-5, 5]$. We tried different combinations of hyperparameters and selected the ones that provided the best results on a validation set across all methods. As in [1], the base model for this experiment is a neural network with two hidden layers of size 40 and ReLU nonlinearities. The adaptation phase (line 8 of Algorithm 1) consists of $Q = 8$ gradient descent steps with a fixed learning rate $\alpha = 0.005$. The ranking procedure (line 5 in Algorithm 1) is performed with $L = 8$ and the step size ε_n is computed with $\varepsilon = 0.1$ and $\lambda = 10^{-5}$. The task encoder architecture comprises a total of 6 layers, 3 for the feature extractor part and 3 for the classifier part, with sizes of 32 and 128, respectively. The training of the task encoder network (as described in Algorithm 4) is performed with a mini-batch size of 35 tasks using the Adam optimizer with $\eta = 10^{-4}$, and the cross-entropy loss function.

The quantitative results showing the average MSE (mean squared error) are reported in Table I (for MUSE-M) and Table II (for MUSE-R). As expected, usual meta-learning approaches (e.g., MAML and Reptile) have high errors when the distribution over tasks is multimodal, and they require more steps to adapt to the new tasks. This is confirmed by Figure 2, where the result of MUSE-R (MUSE on top of Reptile) is compared against Reptile when adapting to a

test task consisting of a new sine wave with five training examples (shown as red triangles). After a few adaptation steps ($Q = 10$) using only these five data-points, MUSE-R was able to adapt much more effectively, as indicated by the predictions represented by the dashed green line. This enhanced adaptation is a result of the task encoder's ability to select the most suitable model initialization for the task at hand. Moreover, TSA-MAML shows poor performance in this regression setting, likely due to the limitations related to directly applying k-means clustering in the parameters space and the fact that parameters specific to various tasks might not constitute clearly separable clusters (e.g., when tasks from various modes are similar or related). The proposed approach, instead, achieves a good performance (low MSE) both when applied on top of MAML (MUSE-M) and Reptile (MUSE-R). In the 10-shot learning scenario ($K = 10$), MUSE-M (resp. MUSE-R) demonstrates comparable results to the Multi-MAML (resp. Multi-Reptile) baselines. This suggests that MUSE can effectively generalize across a multimodal task distribution, even in the absence of ground-truth task mode labels. Besides, it may seem, in this experiment, that the performance of MUSE is better with a larger number of initializations N. Nevertheless, Figure 3 shows that there is a wide range of values (between $N = 4$ to 15) that lead to good results, demonstrating the flexibility in the choice of N. However, as we further increase N, the performance tends to decrease (i.e., MSE increases) due to the increased difficulty for the task encoder to correctly predict the best model initialization.

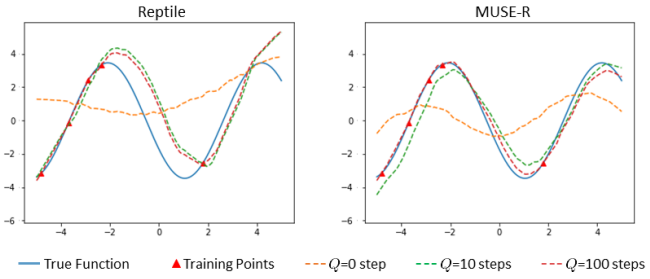


Fig. 2: Few-shot adaptation (with $K = 5$ training points) for a new task consisting of a sine wave, after $Q = 0$, $Q = 10$, $Q = 100$ adaptation steps at meta-test time. Results are presented for Reptile and MUSE-R.

B. Image Classification

For the image classification problem, a task is defined by randomly selecting $\mathcal{N}$ classes and K labeled images per class from a given dataset, i.e., an $\mathcal{N}$-way K-shot classification problem. To create a multimodal task distribution, multiple well-established datasets, each representing a different mode, are combined (including Omniglot [27], Mini-Imagenet [5], FC100 [9], Aircraft [28], FGVCx Fungi [29], CUB Birds [30]) and the images are converted to RGB format with a resolution of 84×84 pixels. The classes within each

TABLE I: Regression results (average MSE) with MUSE-M using tasks sampled from 3 (sinusoidal, linear, quadratic) and 5 (sinusoidal, linear, quadratic, l1norm, hyperbolic tangent) modes. The results are reported in the 5-shot ($K = 5$) and the 10-shot ($K = 10$) learning scenarios. The number of initializations ($N = 3$ or 5) is shown between brackets. For simplicity, standard deviations are not reported in the table, but they are in the order of 10^{-2} for all methods.

	3 modes		5 modes	
Method	5-shot	10-shot	5-shot	10-shot
Scratch	1.88	1.2	1.06	0.74
MAML	1.03	0.26	1.13	0.20
MMAML	0.56	0.26	0.98	0.18
TSA-MAML	3.05	1.14	1.98	1.31
MUSE-M (3)	0.61	0.20	0.65	0.16
MUSE-M (5)	**0.47**	**0.08**	**0.44**	**0.09**
Multi-MAML	0.27	0.07	0.22	0.03
MUSE-M oracle (3)	0.45	0.22	0.55	0.16
MUSE-M oracle (5)	0.29	0.07	0.37	0.11

TABLE II: Regression results (average MSE) with MUSE-R using tasks sampled from 3 (sinusoidal, linear, quadratic) and 5 (sinusoidal, linear, quadratic, l1norm, hyperbolic tangent) modes. The results are reported in the 5-shot ($K = 5$) and the 10-shot ($K = 10$) learning scenarios. The number of initializations ($N = 3$ or 5) is shown between brackets. For simplicity, standard deviations are not reported in the table, but they are in the order of 10^{-2} for all methods.

	3 modes		5 modes	
Method	5-shot	10-shot	5-shot	10-shot
Scratch	1.88	1.2	1.06	0.74
Reptile	2.23	0.41	1.32	0.45
MMAML	0.56	0.26	0.98	0.18
TSA-MAML	3.05	1.14	1.98	1.31
MUSE-R (3)	0.65	0.18	0.74	0.11
MUSE-R (5)	**0.63**	**0.14**	**0.55**	**0.05**
Multi-Reptile	0.56	0.13	0.65	0.16
MUSE-R oracle (3)	0.61	0.13	0.52	0.10
MUSE-R oracle (5)	0.34	0.09	0.38	0.05

dataset are split into two sets, one used to generate tasks for meta-training and the other to generate tasks for meta-testing, following the train/test splits outlined in [31]. On this benchmark, the base model is composed of four modules, each consisting of a convolutional layer with 32 filters, followed by batch normalization, ReLU nonlinearities, and max-pooling, as described in [5]. Additionally, two linear layers with a size of 128 are used to complete the classification model. The number of adaptation steps (used in line 8 of Algorithm 1) is set to $Q = 5$ and the learning rate $\alpha = 0.005$. The step size ε_n is computed in line 7 of Algorithm 1 with $\varepsilon = 0.1$ and a value of λ that iteratively decreases from 0.5 to 0.05, as suggested in [26], for training MUSE-R. For MUSE-M, the value of λ is set to a small value, i.e., 10^{-5}. Also in this setting, we tried different sets of hyperparameters and we selected the one that provided the best performance on the validation set.

TABLE III: Classification results (average accuracy) with MUSE-M when tasks are sampled from multimodal task distributions with three modes. One distribution consists of a combination of Omniglot, Mini-Imagenet, and FC100, while the other consists of Aircraft, FGVCx Fungi, and CUB Birds. The columns "All Datasets" report the average performance when tasks are randomly sampled from all datasets, while "Average" contains the average result over the six datasets. For simplicity, standard deviations are not reported in the table, but they are in the order of 10^{-2} for all methods.

	Omniglot + Mini-Imagenet + FC100				**Aircraft + FGVCx Fungi + CUB Birds**				**Average**
Method	Omniglot	Mini-Imagenet	FC100	All Datasets	Aircraft	Fungi	Birds	All Datasets	
Scratch	0.89	0.48	0.44	0.63	0.43	0.33	0.49	0.46	0.51
MAML	0.95	0.52	0.58	0.66	0.53	0.39	0.52	0.51	0.58
MMAML	0.90	0.42	0.54	0.63	0.48	0.45	0.61	0.51	0.56
TSA-MAML	0.92	0.39	0.46	0.57	0.60	**0.48**	**0.66**	**0.58**	0.59
MUSE-M (3)	**0.97**	0.59	0.66	0.72	**0.68**	0.42	0.58	0.54	**0.65**
MUSE-M (5)	0.96	**0.60**	**0.68**	**0.73**	0.67	0.41	0.55	0.51	0.65
Multi-MAML	0.97	0.50	0.59	0.66	0.61	0.40	0.60	0.51	0.61
MUSE-M oracle (3)	0.98	0.60	0.67	0.73	0.68	0.40	0.56	0.51	0.65
MUSE-M oracle (5)	0.98	0.62	0.69	0.74	0.61	0.41	0.59	0.54	0.65

TABLE IV: Classification results (average accuracy) with MUSE-R when tasks are sampled from multimodal task distributions with three modes. One distribution consists of a combination of Omniglot, Mini-Imagenet, and FC100, while the other consists of Aircraft, FGVCx Fungi, and CUB Birds. The columns "All Datasets" report the average performance when tasks are randomly sampled from all datasets, while "Average" contains the average result over the six datasets. For simplicity, standard deviations are not reported in the table, but they are in the order of 10^{-2} for all methods.

	Omniglot + Mini-Imagenet + FC100				**Aircraft + FGVCx Fungi + CUB Birds**				**Average**
Method	Omniglot	Mini-Imagenet	FC100	All Datasets	Aircraft	Fungi	Birds	All Datasets	
Scratch	0.89	0.48	0.44	0.63	0.43	0.33	0.49	0.46	0.51
Reptile	0.91	0.48	0.54	0.64	0.44	0.43	0.61	0.49	0.56
MMAML	0.90	0.42	0.54	0.63	0.48	0.45	0.61	0.51	0.56
TSA-MAML	0.92	0.39	0.46	0.57	**0.60**	**0.48**	0.66	**0.58**	0.59
MUSE-R (3)	**0.96**	0.49	0.55	0.65	0.50	0.41	0.61	0.51	0.59
MUSE-R (5)	0.94	**0.50**	**0.56**	**0.65**	0.54	0.42	**0.67**	0.54	**0.61**
Multi-Reptile	0.95	0.48	0.54	0.64	0.45	0.44	0.63	0.49	0.58
MUSE-R oracle (3)	0.96	0.51	0.56	0.66	0.51	0.43	0.64	0.52	0.60
MUSE-R oracle (5)	0.96	0.52	0.58	0.66	0.54	0.45	0.67	0.54	0.62

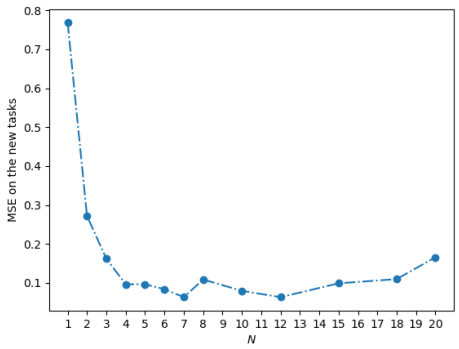


Fig. 3: Average test error (MSE) after adapting to new test tasks, with respect to various values of N (the number of initializations). The result is obtained for MUSE-R in the 10-shot regression scenario with three modes.

Results showing the average accuracy are presented in Tables III and IV. Overall, considering the average result over all six datasets, MUSE demonstrates superior performance compared to other meta-learning approaches. Additionally, MUSE outperforms the Multi-MAML (or Multi-Reptile) approach, highlighting the importance of sharing knowledge across tasks from various modes. This intuition is particularly evident when looking at the results obtained with the combination of Omniglot, Mini-Imagenet, and FC100. Indeed, Mini-Imagenet and FC100 have some commonalities (e.g., similar types of images and some classes in common), which allows MUSE to learn an initialization using tasks generated from both datasets. In other words, the same initialization θ_n can be shared between some tasks generated from Mini-Imagenet, but also some other tasks generated from FC100. This helps MUSE achieve a better result on new tasks generated from these two datasets (or modes) as opposed to other methods (like Multi-MAML and Multi-Reptile) that try to learn a separate initialization using only tasks from the same dataset (or mode). However, when applied to Aircraft, FGVCx Fungi, and CUB Birds, MUSE's performance is not always better than TSA-MAML, but still generally better than Multi-MAML and Multi-Reptile. This could be due to the fact that tasks generated from these datasets represent completely different concepts with no overlap between classes. For this reason,

applying clustering in the parameters' space (as in TSA-MAML) may better capture the differences between tasks than learning to do so via tasks' embeddings. To statistically assess whether the proposed approach significantly outperforms the best-performing baseline (i.e., MMAML for Omniglot, Mini-Imagenet, and FC100 and TSA-MAML for Aircraft, FGVCx Fungi, and CUB Birds), a t-test was conducted comparing the performance of MUSE-M (5) to that of MMAML and TSA-MAML. The results of the t-test indicate that MUSE-M (5) performs significantly better than MMAML with a p-value of approximately 0.03 (less than the significance level 0.05), indicating that the difference in performance is unlikely to be due to chance alone. In other words, the results suggest that the proposed approach is superior to the best-performing baseline on Omniglot, Mini-Imagenet, and FC100 datasets. In contrast, there was no significant difference in performance between TSA-MAML and MUSE-M (5) on Aircraft, FGVCx Fungi, and CUB Birds (i.e., p-value ≈ 0.06), indicating that TSA-MAML was not significantly better than MUSE-M (5) in this case.

V. Conclusion and future work

We presented a meta-learning approach that handles tasks sampled from a multimodal distribution with unknown modes. Our method uses a task encoder to learn an embedding of the target task and predict which of the meta-trained model initializations will lead to the best post-adaptation performance. Our experiments showed that the proposed method outperforms existing meta-learning methods designed for unimodal and multimodal task distributions. While the approach is simple, it highlights the importance of leveraging relevant past experience to make accurate predictions on new tasks. Future work could include incorporating the task encoder into the meta-training process or allowing the meta-learner to directly learn the best initialization for each task.

References

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

[2] A. Nichol and J. Schulman, "Reptile: a scalable metalearning algorithm," *arXiv preprint arXiv:1803.02999*, vol. 2, no. 3, p. 4, 2018.

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

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

[5] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," *International Conference on Learning Representations (ICLR),*, 2017.

[6] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," *International Conference on Learning Representations (ICLR)*, 2017.

[7] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1199–1208.

[8] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell, "Meta-learning with latent embedding optimization," *arXiv preprint arXiv:1807.05960*, 2018.

[9] B. Oreshkin, P. Rodríguez López, and A. Lacoste, "Tadam: Task dependent adaptive metric for improved few-shot learning," *Advances in neural information processing systems*, vol. 31, 2018.

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

[11] A. Rajeswaran, C. Finn, S. M. Kakade, and S. Levine, "Meta-learning with implicit gradients," *Advances in neural information processing systems*, vol. 32, 2019.

[12] H. Yao, Y. Wei, J. Huang, and Z. Li, "Hierarchically structured meta-learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7045–7054.

[13] W. Jiang, J. Kwok, and Y. Zhang, "Subspace learning for effective meta-learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 10 177–10 194.

[14] G. Jerfel, E. Grant, T. Griffiths, and K. A. Heller, "Reconciling meta-learning and continual learning with online mixtures of tasks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[15] P. Zhou, Y. Zou, X.-T. Yuan, J. Feng, C. Xiong, and S. Hoi, "Task similarity aware meta learning: Theory-inspired improvement on maml," in *Uncertainty in Artificial Intelligence*. PMLR, 2021, pp. 23–33.

[16] R. Vuorio, S.-H. Sun, H. Hu, and J. J. Lim, "Multimodal model-agnostic meta-learning via task-aware modulation," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[17] L. Zintgraf, K. Shiarli, V. Kurin, K. Hofmann, and S. Whiteson, "Fast context adaptation via meta-learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7693–7702.

[18] J. Requeima, J. Gordon, J. Bronskill, S. Nowozin, and R. E. Turner, "Fast and flexible multi-task classification using conditional neural adaptive processes," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[19] P. Bateni, R. Goyal, V. Masrani, F. Wood, and L. Sigal, "Improved few-shot visual classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 493–14 502.

[20] L. Liu, W. Hamilton, G. Long, J. Jiang, and H. Larochelle, "A universal representation transformer layer for few-shot image classification," *Proceedings of the International Conference on Learning Representations*, 2021.

[21] N. Dvornik, C. Schmid, and J. Mairal, "Selecting relevant features from a multi-domain representation for few-shot classification," in *European Conference on Computer Vision*. Springer, 2020, pp. 769–786.

[22] W.-H. Li, X. Liu, and H. Bilen, "Universal representation learning from multiple domains for few-shot classification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9526–9535.

[23] E. Triantafillou, H. Larochelle, R. Zemel, and V. Dumoulin, "Learning a universal template for few-shot dataset generalization," in *International Conference on Machine Learning*. PMLR, 2021, pp. 10 424–10 433.

[24] H. Bilen and A. Vedaldi, "Universal representations: The missing link between faces, text, planktons, and cat breeds," *arXiv preprint arXiv:1701.07275*, 2017.

[25] Y. Ma, S. Zhao, W. Wang, Y. Li, and I. King, "Multimodality in meta-learning: A comprehensive survey," *Knowledge-Based Systems*, p. 108976, 2022.

[26] B. Fritzke, "A growing neural gas network learns topologies," *Advances in neural information processing systems*, vol. 7, 1994.

[27] B. Lake, R. Salakhutdinov, J. Gross, and J. Tenenbaum, "One shot learning of simple visual concepts," in *Proceedings of the annual meeting of the cognitive science society*, vol. 33, no. 33, 2011.

[28] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi, "Fine-grained visual classification of aircraft," *arXiv preprint arXiv:1306.5151*, 2013.

[29] Y. C. beejisbrigit, "2018 fgcvx fungi classification challenge," 2018. [Online]. Available: https://kaggle.com/competitions/fungi-challenge-fgvc-2018

[30] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The caltech-ucsd birds-200-2011 dataset," California Institute of Technology, Tech. Rep. CNS-TR-2011-001, 2011.

[31] E. Triantafillou, T. Zhu, V. Dumoulin, P. Lamblin, U. Evci, K. Xu, R. Goroshin, C. Gelada, K. Swersky, P.-A. Manzagol *et al.*, "Meta-dataset: A dataset of datasets for learning to learn from few examples," in *Meta-Learning Workshop at Neural Information Processing Systems,*, 2018.