MetaGB: A Gradient Boosting Framework for Efficient Task Adaptive Meta Learning

Manqing Dong[*], Lina Yao[*], Xianzhi Wang[†], Xiwei Xu[‡], and Liming Zhu[‡]
[*]*Computer Science and Engineering, University of New South Wales*
[†]*School of Computer Science, University of Technology Sydney*
[‡]*Data61, CSIRO*
Sydney, Australia

***Abstract*—Deep learning frameworks generally require sufficient training data to generalize well while fail to adapt on small or few-shot datasets. Meta-learning offers an effective means of tackling few-shot scenarios and has drawn increasing attention in recent years. Meta-optimization aims to learn a shared set of parameters across tasks for meta-learning while facing challenges in determining whether an initialization condition can be generalized to tasks with diverse distributions. In this regard, we propose a meta-gradient boosting framework that can fit diverse distributions based on a base learner (which learns shared information across tasks) and a series of gradient-boosted modules (which capture task-specific information). We evaluate the model on several few-shot learning benchmarks and demonstrate the effectiveness of our model in modulating task-specific meta-learned priors and handling diverse distributions.**

***Index Terms*—Meta Learning, Few-shot Learning, Optimization**

I. Introduction

While humans can learn fast with very few samples using prior knowledge and experiences, artificial intelligence algorithms face challenges in dealing with such situations. Learning to learn (or meta-learning) [1] leverages transferable knowledge learned from previous tasks to improve learning on new tasks [2]; it emerges as an effective means of address the few shot challenge.

In meta-learning, meta-optimization [3]–[5], a.k.a., model-agnostic meta-learning (MAML) [6] is a fundamental task that learns initial model parameters from past tasks so as to adapt fast and perform well on new tasks. Meta-optimization frameworks offer flexibility in model choices and exhibit excellent performance in various domains, such as image classification [6], [7], language modeling [8], and reinforcement learning [9], [10]. Generally, they define a target model $\mathcal{F}_\theta$ and a meta-learner $\mathcal{M}$. The learning tasks $\mathcal{T} = \{\mathcal{T}^{train}, \mathcal{T}^{test}\}$ are divided into training and testing tasks, where $\mathcal{T}$ are generated from the meta-dataset $\mathcal{D}$, i.e., $\mathcal{T} \sim P(\mathcal{D})$. Each task contains a support set D^S and a query set D^Q for training and evaluating a local model. The initialization of the model parameter θ is learned by the meta learner, i.e., $\theta \leftarrow \mathcal{M}(\mathcal{T}^{train})$. We denote the meta-learned parameter by ϕ, i.e., $\theta \leftarrow \phi$. For each task, the model obtains locally optimal parameter $\hat{\theta}$ by minimizing the loss $\mathcal{L}(\mathcal{F}_\theta(D^S), y^S)$, where y^S stands for the label set for the support set. The meta parameter ϕ will be updated across all training tasks by minimizing the loss $\Sigma_{T \in \mathcal{T}^{train}}(\mathcal{L}(\mathcal{F}_{\hat{\theta}}(D^Q), y^Q))$. The goal of meta-optimization approaches is to take only a small number of epochs to learn locally optimal parameters across training tasks while the learned parameter ϕ can quickly converge to an optimal parameter for new tasks.

Most methods assume some transferable knowledge across all tasks and rely on a single shared meta parameter. However, the success of the meta-learners are limited within similar task families, and the single shared meta parameter cannot well support fast learning on diverse tasks (e.g., a large meta-dataset) or task distributions (e.g., $\mathcal{T}$ are generated from multiple meta-datasets) due to conflicting gradients for those tasks [2]. Recent efforts have studied multiple initial conditions to solve the above challenges. Some employ probabilistic models [5], [11], [12] while others incorporate task-specific information [3], [13], [14]. The former learns to obtain an approximate posterior of an unseen task yet needs sufficient samples to get reliable data distributions; the latter conducts task-specific parameter initialization using multiple meta-learners yet requires expensive computation and cannot transfer knowledge across different modes of task distributions.

In this work, we aim to resolve the above challenges from a novel perspective by proposing a meta gradient boosting framework. Gradient boosting [15] aims to build a new learner towards the residuals of the previous prediction result for each step. We call the learner for each step as *weak learner* and make predictions based on summing up the weak learners. Recent research [16], [17] has demonstrated the potential of decomposing deep neural nets into an ensemble of sub-networks, each achieving low training errors. We propose to use the first or first few weak learners as the base learner, followed by a series of gradient-boosting modules to cope with a diverse array of tasks—the base learner is responsible for inferring transferable knowledge by learning across all tasks; the gradient-boosting modules are designed to make task-specific updates to the base learner. Moreover, we operate in the embedding space to inherit the advantages of metric learning, whose simple inductive bias is evidently effective for very-few-shot learning [18]. In a nutshell, our contributions include:

- We propose a novel framework in enabling task-awareness of the traditional meta-optimization approaches: a meta

gradient boosting framework with gradient-boosting modules inferring the task-specific priors.
- The framework is flexible and efficient. It takes fewer resources in setting the initialization conditions and does not require a large number of gradient-boosting modules.
- Our extensive experiments on few-shot learning scenarios for both regression and classification tasks suggest the model's ability in capturing task-aware priors with very few cases.

II. Related Work

Meta-learning has the potential of replicating the human ability to learn new concepts from one or very few instances. It has recently drawn increasing attention, given its broad applicability to different fields [2].

Pioneers [4], [6] in this topic propose optimization algorithms with learned parameters to automate the exploitation to the structures of learning problems. However, most of them initialize the same set of model parameters for all tasks that may have different distributions, thus resulting in over-fitting. Recent studies either model the mixture of multiple initial conditions via probabilistic modeling [11], [12] or incorporate task-specific knowledge [3], [14], to address the above issues. [12] and [11] use variational approximation to enable probabilistic extensions to MAML. But it is unclear how to extend MAML for a wide range of task distributions. [5] consider multiple conditions by borrowing the idea of variational autoencoders [19], which encodes inputs to a low-dimensional latent embedding space and then decodes the learned latent code to generalize task-specific parameters. Another line of research defines a set of initialization modules and incorporates task-specific information to select task-specific modules; this way, it can identify the mode of tasks sampled from a multimodal task distribution and adapt quickly through gradient updates [13]. [20] propose a Hierarchically Structured Meta-Learning (HSML) framework to perform soft clustering on tasks. HSML first learns the inputs and then obtains clustering results by the hierarchical clustering structure. HSML tailors the globally shared parameter initialization for each cluster via a parameter gate to initialize all tasks within the clusters. The above approaches have common limitations in 1) requiring sufficient data samples to generalize task distribution, thus may fail in few-shot cases; 2) being computationally expensive due to the globally stored initialization modules; 3) facing challenges in exhaustively listing every possible initial condition.

Compared with optimization-based meta-learning approaches, metric-based meta-learning approaches [21], [22] learn a projection network that maps the support and the query samples into the same embedding space for comparing the similarities, which aims at optimizing the representation learner $\mathcal{F}_\theta$ so that samples with different labels locate in distinct regions in the embedding space. Metric-based learning approaches are non-parametric, easier for implementation, and less computationally expensive. They have shown superior performance on the case of very-few-shot learning [18], thus achieve state-of-the-art performance on most few-shot learning benchmarks [23]. Recent studies [18], [24] suggest that the Prototypical Networks [21] can be re-interpreted as a linear classifier, where the label is represented by the prototypical representation vector instead of one-hot encoding, i.e., $y_n = \frac{1}{K}\Sigma_{k=1}^{K}\mathcal{F}_\theta(x_{nk}), x_{nk} \in D^S$ for label n as the average of the representations obtained by $\mathcal{F}_\theta$ for K support samples. Optimizing the representation learner $\mathcal{F}_\theta$ in a meta-optimization way can improve the results. We follow this idea in formulating the learning target by combining the strengths of Prototypical Networks and MAML [18].

Two other closely related topics to meta-learning are modular approaches [25] and multi-task learning [26]. Modular approaches are similar to meta-learning in that the input signal gives relatively direct information about a good structural decomposition of the problem. For example, [14] adopt the modular structure and parameter adaptation method for learning reusable modules. Multi-task learning aims to learn a good shared parameter or make the parameter for each task as similar as possible [27]. For example, [28] propose two task networks that share the first few layers for the generic information before applying different prediction layers to different tasks. These approaches differ from meta-learning in requiring fine-tuning on all training samples; thus, they cannot adapt well to new tasks.

Our framework Meta Gradient Booting (MetaGB) neural network is based on the idea of gradient boosting [15], which aims to build a new learner towards the residuals of the previous prediction result for each step. The learner for each step is called a weak learner, and the prediction is based on the summation of weak learners. Weak learners may vary from traditional decision trees [29] to neural networks [16], [30]. A recent study [16] proposes a general framework for gradient boosting on neural networks, which work for both regression and classification tasks. It uses the deep layers of neural nets as a bagging mechanism in a similar spirit to random forest classifier [31]. After only slight tuning, deep neural nets can perform well on a wide range of small real-world datasets [17]. These findings demonstrate the potential of decomposing deep neural nets into an ensemble of sub-networks, each achieving low training errors. In our framework, we use the first weak learner or the first few weak learners as the base learner for learning the shared initialization parameter across tasks. The output for each weak learner is then aggregated to the inputs for the next step for constructing an end-to-end learning strategy until the last gradient-boosting module. This way, the base learner serves as transferable knowledge, and the gradient-boosting modules following it are trained for task-specific predictions.

III. Method

A. Task Formulation

We explore the problem in the context of supervised learning, where input-output pairs are available in both training and validation sets. Similar to previous meta-optimization based approaches [4], [6], we assume the tasks are generated from an underlying distribution $\mathcal{T} \sim P(\mathcal{D})$, where $\mathcal{D}$ is the meta-dataset, which is either a uni-mode dataset or multi-mode datasets. Given a set of tasks $\mathcal{T} = \{\mathcal{T}^{train}, \mathcal{T}^{test}\}$, each task $T \in \mathcal{T}$ contains a support dataset D^S and a query dataset D^Q, both representing input-output pairs (x, y). The support dataset D^S includes $N \times K$ samples for N classes each with K samples, which forms the N-way K-shot learning task. The query dataset includes samples belong to those N classes. We aim to learn a meta-learner $\mathcal{M}$ to guide the initialization for a target model $\mathcal{F}_\theta$ so that the target model can quickly adapt and perform well on a given new task. We propose a Meta Gradient Boosting (MetaGB) framework as the target model $\mathcal{F}_\theta$, which consists of several weak learners and can be represented as $\mathcal{F}_\theta \sim \Sigma_{i=0}^{I} f_{\theta_i}$. The first weak learner f_{θ_0} or the first few weak learners are regarded as the base learner for learning the shared information ϕ across tasks; the following weak learners $\{f_{\theta_i} | i = 1, \ldots, I\}$ are gradient-boosting modules for capturing task-specific information. The meta learner $\mathcal{M}$ aims to learn transferable knowledge and provides initialization details for the base learner, i.e., $\theta_0 \leftarrow \mathcal{M}(\mathcal{T}^{train})$, so that the model can quickly adapt to task-specific predictions with a few gradient-boosting steps. The meta-learner in this paper denotes the global sharing parameter ϕ. Figure 1 shows an example of our MetaGB framework when $I = 1$, where we update the model locally for task-specific predictions and update the meta-learner globally for all tasks. For a classification task, the output y is the representative vector for its corresponding label class instead of a one-hot vector; for a regression task, the output $y \in \mathbb{R}$ is a numerical value. We will discuss more details in the following.

B. Local learning: task-adaptive updating via gradient-boosting

Gradient boosting machines hold out optimization in the function space [15]. In particular, the target model is in an addition form

$$\mathcal{F}_\theta = \Sigma_{i=0}^{I} \alpha_i f_{\theta_i}, \tag{1}$$

where I is the number of adaptions (gradient boosts), f_{θ_0} is the first weak learner, which provides initial prediction of the inputs, f_{θ_i} are the function increments (gradient-boosting modules), and α_i is the boosting rate.

To start, the base-learner f_{θ_0} minimizes the prediction loss $\mathcal{L}(f_{\theta_0}, y)$ to predict the outputs $\hat{y} \leftarrow f_{\theta_0}(x)$, which looks very similar to the traditional supervised approaches. Then, at gradient boosting step i, the gradient boost module f_{θ_i} minimizes the loss $\mathcal{L}(f_{\theta_i}, g_i)$, where $g_i = -\frac{\partial \mathcal{L}(y, \mathcal{F}_\theta^{i-1}(x))}{\partial \mathcal{F}_\theta^{i-1}(x)}$ denotes the negative gradient for the previous step's prediction, and $\mathcal{F}_\theta^i = \Sigma_{*=0}^{i} \alpha_* f_{\theta_*}$ denotes the ensemble of functions at gradient step i. Traditional boosting frameworks learn each weak learner greedily; therefore, only parameters of i-th weak learner are updated at boosting step i while all the parameters of previous $i-1$ weak learners remain unchanged. This, together with the single shared meta parameter, make it easy for the model to stuck in a local minimum. The fixed boosting rate α_i further aggravates the issue.

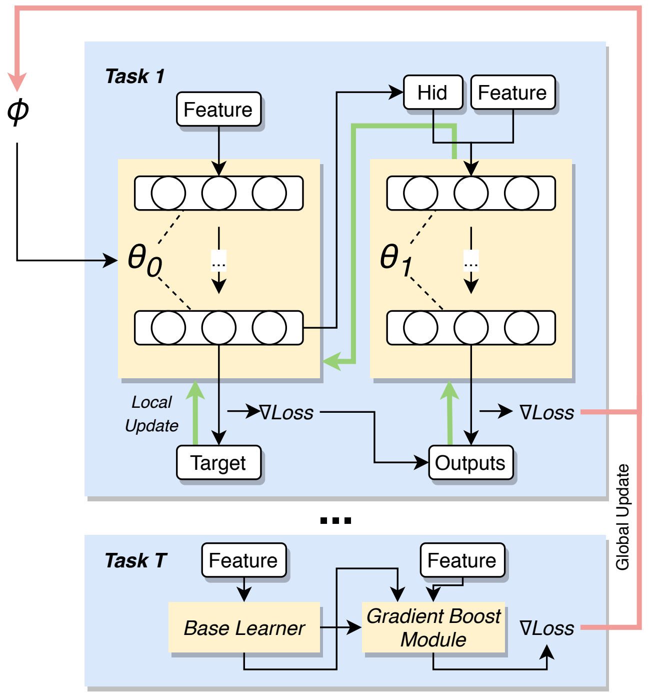


Fig. 1. Example of the model with only one gradient-boosting module. Green lines are for local updates and red lines are for global updates.

In response, we construct the gradient boosting neural network in a cascading structure [32]. Similar to [16], at each step i, we take the concatenation of the inputs x and the hidden layer of the previous weak learner $h_{i-1} = \sigma_{\theta_{i-1}}(x)$ as the inputs for the current step gradient boost module, i.e., $g_i \leftarrow f_{\theta_i}([h_{i-1}, x])$. But our approach differs in optimizing the gradient boosting neural networks in a macroscopic view—for each step i, we learn the ensemble of the weak learners $\mathcal{F}_\theta^i$ by minimizing the loss function

$$\arg\min_\theta \mathcal{L}(y, \mathcal{F}_\theta^i) \to \arg\min_\theta \mathcal{L}(y, \Sigma_{*=0}^{i} \alpha_* f_{\theta_*}), \tag{2}$$

We update parameters of both weak learners and gradient boost module via back-propagation. Generally, the parameter θ of $\mathcal{F}_\theta$ is locally updated via

$$\theta \leftarrow \theta - \beta \nabla_\theta \mathcal{L}(y, \mathcal{F}_\theta) \tag{3}$$

where β is the task learning rate. The boosting rate α_i can be set in various forms—in the simplest case, we can use an increasing or decreasing boosting rate, e.g. $\alpha_i = \alpha_{i-1}/c$ (c is a constant), to decrease or increase the contribution of the base learner. We will discuss model performance under different settings of the boosting rate later. Both the boosting rate and the number of gradient boost modules affect the sharing ability and prediction performance of the base learner. [33] found that the gradient for the earlier weak learners decays with the increasing number of gradient boost modules. On balance, we use the base learner of our proposed gradient boosting framework as a weak learner and a series of gradient-boosting modules as strong learners for a specific task.

Algorithm 1: Task Training by MetaGB

```
Input: $D^S = (x^S, y^S)$, $D^Q = (x^Q, y^Q)$, $\phi$
/* Training on the support sets */
initialize the base learner $f_{\theta_0}$ by $\theta_0 \leftarrow \phi$;
for i in I do
    $\mathcal{F}_\theta^i = \Sigma_{*=0}^i \alpha_* f_{\theta_*}$ ;
    minimize the loss $\mathcal{L}(y^S, \mathcal{F}_\theta^i(x^S))$ by equation (2)
end
for iter in iterations do
    $\theta \leftarrow \theta - \beta \nabla_\theta \mathcal{L}(y^S, \mathcal{F}_\theta(x^S))$ ;
end
/* Testing on the query sets */
get the prediction by $\hat{y}^Q = \mathcal{F}_\theta(x^Q)$
```

Algorithm 2: Meta Training by MetaGB

```
Input: Training tasks $\mathcal{T}$
Output: The updated meta parameter $\hat{\phi}$
random initialize the meta parameter $\phi$;
for iter in iterations do
    for T in $\mathcal{T}$ do
        initialize the task base learner by $\theta_0 \leftarrow \phi$;
        task training on the support set by Algorithm 1;
        obtain the updated task parameters $\hat{\theta}$;
        task testing on the query set;
    end
    update the meta parameter $\phi$ by equation (5)
end
```

Algorithm 1 shows the pseudocode of the task training process by MetaGB. Specially, for the classification tasks, the labels are initialized by the representative vector for its corresponding label class, i.e., $y_n = \frac{1}{K}\Sigma_k f_{\theta_0}(x^S_{nk})$, so that to introduce inductive bias to the model.

C. Global learning: meta-optimization for transferable knowledge learning

The learning and updating strategy of the gradient boosting framework ensure a weak base learner. Since the base learner could be the first weak learner or the first few weak learners, we use f_{θ_0} to represent both conditions for ease of illustration. We take the meta-optimization approach for initializing the base learner so that the model can provide an initial guess for the prediction based on the transferable knowledge over a wide arrange of tasks. Specifically, we learn a global sharing parameter ϕ s.t. $\theta_0 \leftarrow \phi$. Similar to other MAML-based approaches [3], [6], we minimize the expected loss on the local query set D^Q for tasks $T \in \mathcal{T}$ in meta optimization, i.e., minimizing the loss

$$\arg\min_\phi \mathcal{L}_{meta} \rightarrow \arg\min_\phi \Sigma_{T\in\mathcal{T}} \mathcal{L}(y^Q, \mathcal{F}_{\hat{\theta}}(x^Q)) \quad (4)$$

where $T = \{D^S, D^Q\}$, $D^Q = (x^Q, y^Q)$, and $\hat{\theta}$ is the updated model parameters by learning on the support set D^S we introduced in the previous section.

The meta-gradient for updating ϕ may involve higher-order derivatives, which are computationally expensive for deep neural nets. Some work [4], [6] take one-step gradient descent (FOMAML), i.e.,

$$\phi \leftarrow \phi - \gamma \frac{1}{|\mathcal{T}|} \Sigma_{T\in\mathcal{T}} \nabla_{\hat{\theta}_0} \mathcal{L}(y^Q, \mathcal{F}_{\hat{\theta}}(x^Q)) \quad (5)$$

or parameter difference (Reptile), i.e.,

$$\phi \leftarrow \phi + \gamma \frac{1}{|\mathcal{T}|} \Sigma_{T\in\mathcal{T}} (\hat{\theta}_0^T - \phi) \quad (6)$$

for meta-optimization, where γ is the global learning rate. Specially, We use FOMAML (equation 5) to update the global parameter ϕ for its generally better performance. After the meta training process, we obtain the updated meta parameter $\hat{\phi}$, which can be utilized to initialize the model for new tasks. The pseudocode of meta training is described in Algorithm 2.

IV. Experiments

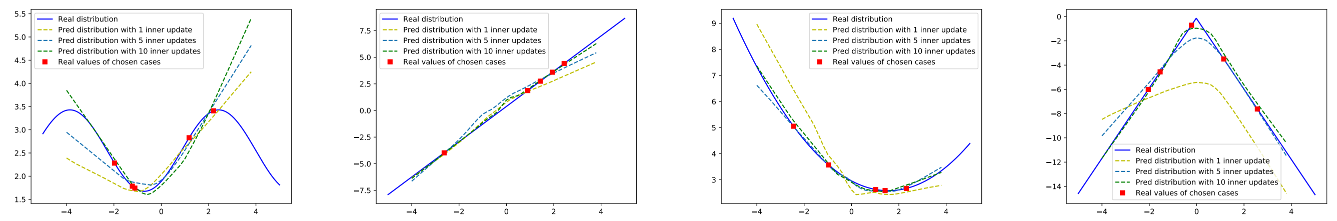
A. Regression tasks

We adopt the simple regression task with similar settings in [11]. In the 1D regression problem, we assume different tasks correspond to different underlying functions, and we aim to predict the outputs for new tasks by learning on very few samples. Specially, we consider four types of functions, i.e., sinusoidal, linear, quadratic, and abstract value functions. To ensure each function can generate a variety of tasks, we define the four functions with three variables as follows:

- Sinusoidal function: $v_2 \cdot \sin(x + v_1) + v_3$
- Linear function: $v_2 \cdot (x + v_1) + v_3$
- Quadratic function: $v_1^2 \cdot x + v_2 \cdot x + v_3$
- Absolute value function: $v_2 \cdot |x + v_1| + v_3$

where v_1, v_2, and v_3 are sampled uniformly from ranges [0, π], [-3,3], and [-3,3], respectively. The inputs are within [-5,5], and the outputs are perturbed with Gaussian noise with the standard deviation of 0.3.

1) Single-mode Task Adaptation: Under the case of single-mode learning, the training tasks $\mathcal{T}^{tr}$ and testing tasks $\mathcal{T}^{te}$ are generated from a single type of function. For each task T, the function is given by randomly selecting the coefficients v_1, v_2, v_3, e.g., the slope for the linear function, then generates the input-output pairs for support set D^S and query set D^Q. Following [6], we use two fully connected layers of size 40 as the weak learner for regression tasks. The activation function is Leaky ReLU [34], which provides good and stable results over the four functions. By default, the MetaGB framework includes one weak learner as the base learner and three gradient-boosting modules, where the gradient boost modules share the same structure as the base learner but with different parameter initialization. Recall that we use a cascading structure [32] in constructing the whole framework: at each step i, we take the concatenation of the inputs x and the hidden layer of the previous weak learner $h_{i-1} = \sigma_{\theta_{i-1}}(x)$ as the inputs of gradient boost module at the current step, i.e., $g_i \leftarrow f_{\theta_i}([h_{i-1}, x])$, where σ is the embedding layer. Notice



(a) Sinusoid function (b) Linear function (c) Quadratic function (d) Absolute value function

Fig. 2. Performance on uni-mode regression tasks with three gradient boost modules. Blue lines show the real distribution. Red nodes stand for the real values of the support set samples when **n-shot = 5**. The dashed yellow, blue, and green lines represent for model performance with updating 1, 5, or 10 times, respectively.

that the input for the regression task is a single numerical value, we introduce shortcut layers to map the input x to a latent embedding h_x so that to be with the same shape as h_{i-1}, i.e., we feed $[h_{i-1}, h_x]$ into the i-th gradient boost module. We have also tested the model performance with forwarding $[h_{i-1}, h_{i-2}]$ for each gradient boost module and found it only has a slight impact on the model performance. The inputs are batch normalized [35] when fed into the whole framework. Then, we update ten times for both the base learner and gradient-boosting modules for task learning. The task learning rate β is set to 0.01, and the boosting rate α is initialized as 0.5 that will be automatically updated during the task training. The meta sharing parameter ϕ will be learned on the training tasks with a meta-learning rate γ as 0.01.

Figure 2 shows the performance of the proposed MetaGB framework with three gradient boost modules in different task adaptation steps on the four types of functions. We have the following observations. 1) Generally, more updating steps improve the model performance. For the relatively simple regression tasks, the model can generalize well with few updating steps. 2) Too many updates (i.e., updating ten times for a module) can only slightly improve the performance, which suggests the MetaGB can obtain an acceptable result in a computationally inexpensive way. 3) We notice that the model cannot handle well with the unseen cases for the sinusoid function, as the trace from the five seen samples could form either quadratic function or absolute value functions. To improve the performance, we can predict the coefficients v_1, v_2, v_3 in forming the function distribution other than directly predict the regression output y. However, this will weaken the generalization ability of the model in handling a wide variety of regression functions. Overall, the proposed framework can accurately predict the regression outputs within the input domain of the few observed samples.

2) Multi-mode Task Adaptation: Under the case of multi-mode learning, the training tasks and the testing tasks are generated from multiple types of functions, i.e., we generate multi-mode training tasks by first selecting the function type for each task T and then initializing random parameters v_1, v_2, v_3 for the function to generate samples. Other settings for multi-mode learning are similar to the settings for single-mode learning.

Learning from the above four types of distribution is challenging. First, the function values may vary across the distributions. For example, the output of the quadratic function can range from zero to dozens (see Figure 2) while the other three functions are more likely to produce outputs within [-10,10]. Second, the few-shot samples that sit on a line might be generated from non-linear functions, which makes it difficult to learn the real modality from such samples. Updating more steps for task-specific models could solve the first challenge yet may cause over-fitting. The second challenge can be mitigated by producing a set of models for task learning. Our proposed MGB can well handle the two challenges by a series of task-specific gradient boost modules and providing flexibility in updating the framework.

We compare the proposed framework on multi-mode regression tasks with the baseline MAML [6] and two other state-of-the-arts: Multimodal Model-Agnostic Meta-Learning (MMAML) [13], and Meta-learning with Latent Embedding Optimization (LEO) [5]. Both MMAML and LEO model a mixture of multiple initial conditions. MMAML modulates the meta-learned prior parameters via an external modular network to enable more efficient adaptation; LEO solves the problem via probabilistic modeling that learns a stochastic latent space with an information bottleneck conditioned on the input data, from which the high-dimensional parameters are generated. Metric-based meta-learning approaches are discarded for comparison since they are not applicable for regression tasks. We use mean absolute error (MAE) as the evaluation metric to evaluate the model performance in training on one-mode (sinusoidal function), two-mode (sinusoidal function and linear function), or four-mode (all the four functions) tasks. To ensure a fair comparison, the same basic settings of the network structure are configured for all comparison methods, and the models are learned within certain training epochs. The results are shown in Table I.

We have the following observations. 1) Capturing task-specific patterns on multi-mode tasks is difficult, where the MAE is larger when more function modes are considered. This makes sense considering the randomness in choosing functions and function parameters for all models. 2) Comparing MAML with other methods, we can see that incorporating task identities, e.g., modifies the initialization parameter by an external network

TABLE I
RESULTS FOR REGRESSION TASKS (MAE)

Method	1 Mode		2 Modes		4 Modes	
	5-shot	10-shot	5-shot	10-shot	5-shot	10-shot
MAML [6]	1.234±0.174	1.054±0.077	1.548±0.223	1.356±0.109	2.044±0.472	1.698±0.267
LEO [5]	0.957±0.123	0.789±0.042	1.127±0.175	0.899±0.084	1.234±0.248	1.095±0.163
MMAML [13]	0.638±0.053	0.526±0.027	0.783±0.096	0.709±0.048	1.016±0.181	0.920±0.099
MetaGB-1	0.674±0.009	0.524±0.013	0.999±0.103	0.906±0.038	1.213±0.173	0.928±0.084
MetaGB-2	0.629±0.004	0.512±0.005	0.822±0.032	0.734±0.021	1.046±0.106	0.899±0.027
MetaGB-5	**0.615±0.005**	**0.476±0.005**	**0.704±0.042**	**0.672±0.077**	**0.985±0.089**	**0.825±0.043**

as in MMAML or introduces a group of task-aware gradient boost modules as in our MetaGB, can significantly improve the performance for multi-mode learning. MAML has the highest error for both uni-mode tasks and multi-mode tasks, which suggests the single sharing parameter is hard to capture task identities and needs more training information to generalize well. 3) Comparing the performance of the proposed MetaGB with one (MetaGB-1), two (MetaGB-2), and five (MetaGB-5) gradient-boosting modules, the performance improves with more gradient boost modules, while the improvement decreases as the number of gradient-boosting modules increases, which suggests the MetaGB can obtain well performance with a small number of gradient boost modules. Overall, the proposed MGB framework achieves the best performance in all settings and shows more stable performance compared with LEO and MMAML on multi-mode regression tasks, which proves the effectiveness of the gradient boost modules in capturing the task-specific multi-modality information.

B. Classification tasks

We use four datasets to evaluate multi-mode few-shot classification tasks: Omniglot [36], miniImageNet [37], FC100 [24], and CUB [38]. Basic information about the datasets are listed in Table II, where **# samples** denotes the number of samples for each class, and **# channels** denotes the number of image channels. We divide tasks in each dataset into train, validation, and test tasks. Omniglot is a dataset of 1,623 handwritten characters from 50 different alphabets, where we use the characters from the first 30 alphabets, the following 10 alphabets, and the last 10 alphabets as training, validating, and testing tasks, respectively. The miniImageNet dataset is a subset of the ImageNet[1]. Following [6], we divide 100 classes into 64, 16, and 20 classes for meta-training, meta-validation, and meta-testing, respectively. Fewshot-CIFAR100 (FC100) is based on the widely applied CIFAR 100 dataset[2]. Similar to [24], we divide 100 classes into 60 training classes, 20 validation classes, and 20 testing classes. As for the Caltech-UCSD Birds 200 dataset, i.e., the CUB dataset, we follow the settings in [13] and use 140, 30, and 30 classes for training, validation, and testing, respectively. In general, we use 20 batches of 50 tasks to train the model for both uni-mode tasks and multi-mode tasks.

[1]http://www.image-net.org/

[2]https://www.cs.toronto.edu/~kriz/cifar.html

TABLE II
BASIC INFORMATION OF THE IMAGE DATASETS

Dataset	# classes	# samples	# channels	Image size
Omniglot	1,623	20	1	28×28
miniImageNet	100	600	3	84×84
FC100	100	600	3	32×32
CUB	200	>40	3	84×84

1) Single-mode Task Adaptation: Under the case of single-mode image classification, the training, validation, and testing tasks are generated from a single dataset and have no intersection in the label set. For each N-way K-shot learning task, we randomly draw N classes and sample K images for each class to form the support set. We further sample M images for each class to form the query set, which includes $N \times M$ images. By default, MetaGB includes one weaker learner as the base learner and uses either CONV4 [6] or ResNet18 [32] as the backbone. For the gradient boost modules, we use CONV4 [6] as the backbone for saving the computation cost. Similar to the settings of regression tasks, we batch normalize the inputs and use shortcut layers before feeding them into the gradient modules. For classification, instead of using one-hot vectors, we use the prototype vector, i.e., $y_n = \frac{1}{K}\Sigma_{k=1}^{K}\mathcal{F}_\theta(x_{nk}), x_{nk} \in D^S$, to initialize the representation for each class for task training. We update five times for both the base learner and the gradient-boosting modules in task learning. Then, the class representation will be updated with the learned task-specific model parameter $\hat{\theta}$, which will be further used for evaluation on the query set. The boosting rate (α) for each gradient boost module is automatically learned and is initialized as 0.5. The task learning rate β and the meta-learning rate γ are set to 0.001 and 0.005, respectively.

We evaluate the performance of the proposed MetaGB framework with one, two, or five gradient boost modules and compare with a series of baseline methods (e.g., MAML [6] and ProtoNet [21]) and several recent task-adaptive frameworks (e.g., LEO [5] and TADAM [24]) for the single-mode image classification tasks. The results are presented in Table III, where the comparison results are originated from the published papers. First, we can see that the options of the model backbone can

TABLE III
RESULTS FOR SINGLE-MODE CLASSIFICATION TASKS (ACCURACY)

Method	Backbone	**miniImageNet**		**FC100**		**CUB**	
		5-way 1-shot	5-way 5-shot	5-way 1-shot	5-way 5-shot	5-way 1-shot	5-way 5-shot
MAML [6]	CONV4	48.70±1.84	63.11±0.92	-	-	69.96±1.01	82.70±0.65
ProtoNet [21]	CONV4	49.42±0.78	68.20±0.66	35.30±0.60	48.60±0.60	-	-
MatchingNet [8]	CONV4	43.56±0.84	55.31±0.73	-	-	72.36±0.90	83.64±0.60
TADAM [24]	ResNet-12	58.50±0.30	76.70±0.30	40.10±0.40	56.10±0.40	-	-
ProtoNet [21]	ResNet-18	-	-	37.50±0.60	52.50±0.60	71.88±0.91	87.42±0.48
LEO [5]	WRN28-10	61.76±0.08	77.59±0.12	-	-	-	-
MetaGB-1	CONV4	49.04±1.58	60.58±1.23	36.68±1.32	54.37±1.04	68.24±1.68	81.35±1.12
MetaGB-2	CONV4	50.18±1.37	63.29±0.95	39.46±0.98	57.68±0.90	68.68±1.17	83.49±0.78
MetaGB-5	CONV4	50.88±1.01	65.11±0.88	40.61±0.89	58.89±0.99	69.72±1.03	84.12±0.65
MetaGB-1	ResNet18	56.78±0.97	70.74±0.98	38.44±1.27	55.14±1.12	70.37±1.32	82.86±1.21
MetaGB-2	ResNet18	58.84±0.72	74.51±0.89	40.52±1.01	58.76±1.01	71.68±1.12	84.37±1.07
MetaGB-5	ResNet18	59.31±0.61	77.17±0.63	40.84±1.08	59.85±0.97	72.31±1.16	86.19±0.99

TABLE IV
RESULTS FOR MULTI-MODE CLASSIFICATION TASKS (ACCURACY)

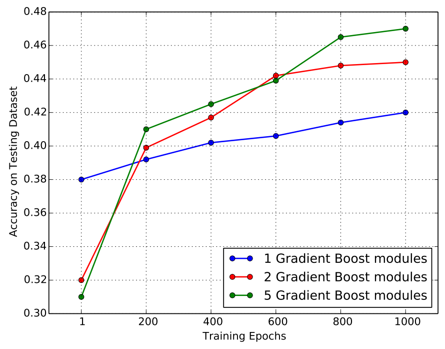
Method	**2 Modes**			**3 Modes**			**4 Modes**		
	5-way		20-way	5-way		20-way	5-way		20-way
	1-shot	5-shot	1-shot	1-shot	5-shot	1-shot	1-shot	5-shot	1-shot
MAML [6]	0.6381	0.7524	0.4296	0.5235	0.6481	0.2986	0.4223	0.5172	0.2415
LEO [5]	0.6676	0.7689	0.4318	0.5129	0.6413	0.3106	0.3948	0.4757	0.2301
MMAML [13]	0.6797	0.7738	0.4521	0.5536	0.6728	0.3534	0.4812	0.5528	0.2949
MetaGB-1	0.6394	0.7579	0.4228	0.5241	0.6435	0.3002	0.4277	0.5209	0.2533
MetaGB-2	0.6501	0.7633	0.4270	0.5503	0.6750	0.3459	0.4531	0.5374	0.2682
MetaGB-5	0.6834	0.7830	0.4426	0.5611	0.6897	0.3568	0.4725	0.5530	0.2956

significantly affect the prediction results. There is a noticeable improvement with using ResNet [32] on miniImageNet dataset, which contributes to the pretraining strategy of the ResNet that may contain the prior information of the datasets. Such effects become less significant on the other two datasets, while a deeper and more complex model structure can still achieve better performance with more parameters capturing the information from the datasets. Second, the task-adaptive frameworks obtain better results than the traditional meta-learning frameworks, which indicates the task-adaptive frameworks secure more task-specific information by introducing task-aware parameter space. And last, the proposed MetaGB framework performs well on nearly all datasets, and generally, such improvements are coming with more gradient boost modules. Overall, the results suggest the proposed MetaGB works efficiently on the uni-mode classification tasks.

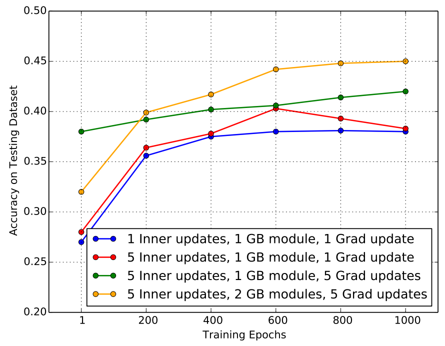
2) Multi-mode Task Adaptation: For multi-mode image classification, each task of the training and testing tasks can be generated from either one of the four image datasets. We resize the image for each dataset into 84×84 so that the model can share across the four datasets. Besides, we map images in the Omniglot dataset (which only has one image channel) to a layer with three channels so that they can be learned by the sharing model. Other settings are the same as in single-mode learning.

We compare the proposed MetaGB with three optimization-based methods, i.e., MAML [6], LEO [5], and MMAML [13]. MAML is one of the pioneer frameworks for meta optimization. For each task, the parameter θ of the task model $\mathcal{F}_\theta$ is initialized by the global sharing parameter ϕ and is updated by learning from the support set D^S. The meta parameter ϕ will be updated by learning from the loss on the query set $\mathcal{L}(D^Q)$. LEO uses an external encoder-decoder to initialize the parameter θ of the task model $\mathcal{F}_\theta$. The external encoder encodes the inputs to the latent code z and will be updated by learning on the support set. The decoder will decode the updated latent code $\hat{z}$ to obtain the task-specific parameter θ. In particular, LEO uses pretrained embedding as input to statistically generate means and variances for the predictor in its encoder-decoder structure. The predictor can be regarded as one fully-connected layer that takes the pretrained embedding [39] as input. The weights of the predictor are sampled from a distribution specified by the means and variances generated by LEO. Since LEO cannot be updated via classic back-propagation, we use similar measures as presented by [5]. But we additionally use our strategy for generating the embeddings. MMAML uses a modulation network g_{ϕ_h} to obtain the mode information v of a sampled task, i.e., $v \leftarrow g_{\phi_h}(x)$, which will be further used to provide the modulation vectors τ by $\tau \leftarrow g_{\phi_g}(v)$. Suppose the task model $\mathcal{F}_\theta$ includes I sub-parameters $\{\theta^i | i \in I\}$, then each sub-parameter will be initialized by $\theta^i \leftarrow \theta^i \odot \tau^i$.

We evaluate the model on two modes (Omniglot and miniImageNet), three modes (Omniglot, miniImageNet, and FC100), and four modes (all four datasets) tasks with one (MetaGB-1), two (MetaGB-2), or five (MetaGB-5) gradient-boosting modules. For each task, we consider 1-shot and 5-shot learning for 5-way classification and 1-shot learning for 20-way classification. According to the results in Table III and the reports in [40], the model backbone can significantly affect the classification results. To ensure a fair comparison, we use CONV4 [6] as the backbone for all comparison methods on multi-mode classification tasks. Table IV shows the comparison results. Overall, the proposed MetaGB performs well on multi-mode tasks. Compared with MMAML, our method works better on most scenarios except on 1-shot 20-way classifications, where MMAML includes an external network that can store more parameters. Similar to the regression tasks, MetaGB with more gradient-boosting modules shows better performance, while MetaGB-1 can make only a slight improvement over MAML because images contain more complex information than real numbers. More modes of tasks increase the performance gap between MAML and the other methods, which suggests the other methods (which consider multiple conditions) can handle multi-mode tasks better than MAML. Under the same experimental settings, i.e., with the same image embedding modules, LEO does not perform well on tasks with more modes, partially because it is largely impacted by the quality of the learned image embedding—first, LEO's learning strategy [5] pre-trains the dataset-specific image embedding [39] before meta-learning; then, LEO uses an encoder-decoder structure to generate parameters for the classifier from the learned embedding.



(a)



(a)

Fig. 3. Model performance under different settings about (a) the number of gradient boost modules and (b) the updating strategy of the gradient boost modules.

V. Discussion

Our experimental results on both regression and classification tasks suggest our method can adapt to the optimal results with few gradient-boosting modules. In this section, we take a further step to discuss i) the configuration of the gradient-boosting modules and ii) the sharing ability of the base learner during the back-propagation through meta-gradient-boosting modules. The results are presented for 5-way 1-shot 4-mode image classification tasks. There are two gradient boost modules of the MetaGB framework by default. All experiments are implemented on a single NVIDIA Tesla V100 32GB GPU, which will take seconds to tens of seconds for running over a single task and require few hours to obtain the optimal results.

A. Configuration of the gradient-boosting modules

1) Settings for the weak learner: Our MetaGB framework consists of a series of weak learners, where the first or the first few weak learners serve as the base learner to be shared among tasks. Type and dimension of the weak learner are two key factors that may affect the final results. [13] find that LSTM models perform better than linear models in regression tasks. [40] show the choice of feature extractor for images has a strong correlation with the final results, and using a pre-trained network or network structure can improve the results significantly. For example, it improves the accuracy by about 6% [24], [41] to use ResNet-12 as the feature extractor on the MiniImageNet dataset (see Table III. In this work, we use linear layers for regression tasks and either CONV4 [6] or ResNet [32] for classification tasks. For classification tasks, each weaker learner includes a feature extractor and a predictor. We use several fully connected layers as the feature extractor for mapping the extracted feature representation to an embedding space. We find that a larger embedding size (e.g., 400-800) provides better performance than a relatively small embedding size (e.g., 100-200). All these findings suggest a deeper and wider model delivers better prediction results, whereas the model will take more computation cost for those extended network structures. For the proposed MetaGB, the previous experimental results indicate the improvements by many gradient boost modules become insignificant with the growing number of gradient boost modules, which demonstrates the model requires only few gradient-boosting steps to obtain optimal results. Under the case of a model that includes three gradient boost modules with each contains a CONV4 feature extractor and several fully connected layers, the whole framework will take about 6G video memory during the training process.

2) Settings for the gradient-boosting modules: The number of gradient-boosting modules and the updating strategy for the gradient-boosting modules are two important factors in

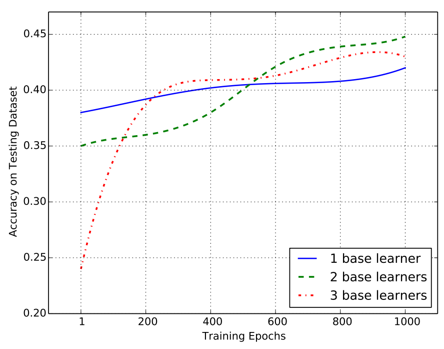


Fig. 4. Model performance with using different number of weak learners as the base learner.

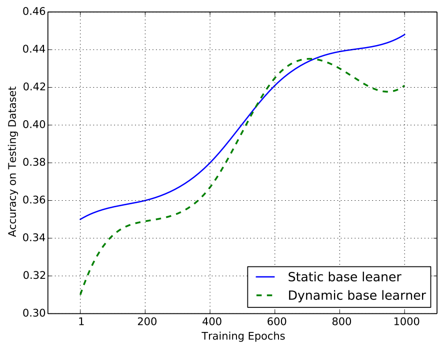


Fig. 5. Model performance with static or dynamic base learner during the task training

constructing the whole framework. Since the prediction is based on the summation of a series of weak learners, more gradient-boosting modules will reduce the contribution of each weak learner. According to the results shown in Figure 3 (a), the model with only one gradient boost module cannot well handle multi-mode tasks; more gradient boost modules improves the results while takes more time to learn the task information. The traditional gradient boost module greedily updates the whole framework at each step, while for MetaGB, we can adjust the contribution of the base learner and gradient-boosting modules to the final results by allowing them to update with different iterations in each learning step. Intuitively, if we fine-tune on the base learner (i.e., updating multiple times on the base learner), the model may get stuck in a local optimal and decrease the impact of gradient-boosting modules on the model performance; conversely, if we conduct more updates on the gradient-boosting modules, the model may need more training epochs from all training tasks to grasp the general sharing information that assists the model in fast-adaptation. The results under different settings of the updating times for base learner and gradient-boosting modules are shown in Figure 3 (b). We can see the updating strategy significantly affects model performance, where updating more steps on the gradient modules raises the model's robustness. The performance tends to become unstable if we conduct more updates for the base learner, which observation aligns with our analysis above.

B. Sharing ability of the base learner

The base learner of our MetaGB framework is shared across tasks for capturing the general sharing knowledge. Three factors may affect the sharing ability of the base learner: 1) which part to share; 2) how to share; 3) how the shared base learner contributes to MetaGB. We discuss these three components as follows.

1) Single weak learner v.s. Multiple weak learners: Instead of choosing a single weak learner as the base learner, we can choose the first few weak learners as the base learner to be shared across tasks. We present the results of MetaGB with one, two, or three weak learners as the base learner and one gradient-boosting module in Figure 4. Generally, when more weak learners used as the base learner (more than the number of gradient boost modules), MetaGB faces difficulties in capturing multi-mode patterns and achieves degraded generalization performance at the beginning of the training process; the performance increases with more training epochs but fluctuates more than the one with only one weak learner as the base learner.

2) Static base learner v.s. Dynamic base learner: The base learner is initialized using the global sharing parameter ϕ. It can be either static (if we keep its parameter unchanged) or dynamic (if we update the ensemble of base learner and gradient boost modules during the training process). The results (shown in Figure 5) reveal that keeping the shared information (i.e., using static base learner) unchanged can improve the stability of the model.

3) Boosting rate α: The boosting rate α is probably the most vital component for the MetaGB framework because it directly indicates the contribution of each weak learner to the final prediction. We test the performance of MetaGB under various settings of the boosting rate α, where the rate is either decayed (i.e., $\alpha_k = \alpha_{k-1}/c$, where c is a constant), automatically learned, or equally contributed (i.e., $\alpha_* = 1$ for all weak learners). The result suggests that using the automatically learned α or equally contributed α leads the better performance, while a decayed α causes difficulty in model convergence, which indicates the importance of the gradient boost modules in capturing the task information.

TABLE V
SETTINGS FOR THE BOOSTING RATE α

Settings	Accuracy
Decayed, c=2	0.3923
Automatically learned	0.4525
Equally contributed	0.4267

VI. CONCLUSION

In this work, we aim to learn an optimized set of initial parameters that fit diverse tasks, which is a key challenge confronting current meta-optimization approaches. We propose a meta gradient boosting framework that uses a series of weak learners to make predictions and a base learner to grasp shared information across all tasks. Our extensive experiments show the effectiveness of our framework in handling diverse tasks, demonstrating the model's ability to capture meta information and task-specific information on both regression and classification tasks. Our results also reveal the necessity of selecting the weak learner carefully according to task types. An example is that the pretrained image processing networks outperform convolutional neural networks (CNNs) on image classification problems. The extended parameter space engages more task-aware knowledge but takes more computation cost. In future work, we will analyze the weak learner options in-depth to discover how to obtain well and stable performance with the least effort in tuning on external networks.

References

[1] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Artificial intelligence review*, vol. 18, no. 2, pp. 77–95, 2002.

[2] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *arXiv preprint arXiv:2004.05439*, 2020.

[3] Y. Lee and S. Choi, "Gradient-based meta-learning with learned layerwise metric and subspace," in *International Conference on Machine Learning*, 2018, pp. 2927–2936.

[4] A. Nichol and J. Schulman, "Reptile: a scalable metalearning algorithm," *arXiv preprint arXiv:1803.02999*, vol. 2, p. 2, 2018.

[5] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell, "Meta-learning with latent embedding optimization," *ICLR*, 2019.

[6] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1126–1135.

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

[8] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, "Matching networks for one shot learning," in *Advances in neural information processing systems*, 2016, pp. 3630–3638.

[9] C. Fernando, J. Sygnowski, S. Osindero, J. Wang, T. Schaul, D. Teplyashin, P. Sprechmann, A. Pritzel, and A. Rusu, "Meta-learning by the baldwin effect," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2018, pp. 1313–1320.

[10] M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castaneda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman *et al.*, "Human-level performance in 3d multiplayer games with population-based reinforcement learning," *Science*, vol. 364, no. 6443, pp. 859–865, 2019.

[11] C. Finn, K. Xu, and S. Levine, "Probabilistic model-agnostic meta-learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 9516–9527.

[12] J. Yoon, T. Kim, O. Dia, S. Kim, Y. Bengio, and S. Ahn, "Bayesian model-agnostic meta-learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 7332–7342.

[13] 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*, pp. 1–12, 2019.

[14] F. Alet, T. Lozano-Peréz, and L. P. Kaelbling, "Modular meta-learning," *arXiv preprint arXiv:1806.10166*, 2018.

[15] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[16] S. Badirli, X. Liu, Z. Xing, A. Bhowmik, and S. S. Keerthi, "Gradient boosting neural networks: Grownet," *arXiv preprint arXiv:2002.07971*, 2020.

[17] M. Olson, A. Wyner, and R. Berk, "Modern neural networks generalize on small data sets," in *Advances in Neural Information Processing Systems*, 2018, pp. 3619–3628.

[18] 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 *International Conference on Learning Representations*, 2019.

[19] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *stat*, vol. 1050, p. 1, 2014.

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

[21] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 4080–4090.

[22] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2. Lille, 2015.

[23] Y. Tian, Y. Wang, D. Krishnan, J. B. Tenenbaum, and P. Isola, "Rethinking few-shot image classification: a good embedding is all you need?" *arXiv preprint arXiv:2003.11539*, 2020.

[24] B. Oreshkin, P. R. Lopez, and A. Lacoste, "Tadam: Task dependent adaptive metric for improved few-shot learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 721–731.

[25] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein, "Neural module networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 39–48.

[26] Y. Zhang and Q. Yang, "A survey on multi-task learning," *arXiv preprint arXiv:1707.08114*, 2017.

[27] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–34, 2020.

[28] Y. Zhang, H. Tang, and K. Jia, "Fine-grained visual categorization using meta-learning optimization with sample selection of auxiliary data," in *Proceedings of the european conference on computer vision (ECCV)*, 2018, pp. 233–248.

[29] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[30] P. Tannor and L. Rokach, "Augboost: gradient boosting enhanced with step-wise feature augmentation," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 2019, pp. 3555–3561.

[31] A. Veit, M. J. Wilber, and S. Belongie, "Residual networks behave like ensembles of relatively shallow networks," in *Advances in neural information processing systems*, 2016, pp. 550–558.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[33] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber *et al.*, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," 2001.

[34] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[35] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[36] 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, 2011.

[37] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," 2016.

[38] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The caltech-ucsd birds-200-2011 dataset," 2011.

[39] S. Qiao, C. Liu, W. Shen, and A. L. Yuille, "Few-shot image recognition by predicting parameters from activations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7229–7238.

[40] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele, "Meta-transfer learning for few-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 403–412.

[41] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," in *International Conference on Learning Representations*, 2018.