TAdaNet: Task-Adaptive Network for Graph-Enriched Meta-Learning

Qiuling Suo
State University of New York at Buffalo, NY, USA
qiulings@buffalo.edu

Jingyuan Chou
University of Virginia, VA, USA
jc2wv@virginia.edu

Weida Zhong
State University of New York at Buffalo, NY, USA
weidazho@buffalo.edu

Aidong Zhang
University of Virginia, VA, USA
aidong@virginia.edu

ABSTRACT

Annotated data samples in real-world applications are often limited. Meta-learning, which utilizes prior knowledge learned from related tasks and generalizes to new tasks of limited supervised experience, is an effective approach for few-shot learning. However, standard meta-learning with globally shared knowledge cannot handle the task heterogeneity problem well, i.e., tasks lie in different distributions. Recent advances have explored several ways to trigger task-dependent initial parameters or metrics, in order to customize task-specific information. These approaches learn task contextual information from data, but ignore external domain knowledge that can help in the learning process. In this paper, we propose a task-adaptive network (TAdaNet) that makes use of a domain-knowledge graph to enrich data representations and provide task-specific customization. Specifically, we learn a task embedding that characterizes task relationships and tailors task-specific parameters, resulting in a task-adaptive metric space for classification. Experimental results on a few-shot image classification problem show the effectiveness of the proposed method. We also apply it on a real-world disease classification problem, and show promising results for clinical decision support.

CCS CONCEPTS

• **Information systems → Data mining**; • **Computing methodologies** → *Neural networks.*

KEYWORDS

Meta learning; Few-shot learning; Predictive healthcare

ACM Reference Format:
Qiuling Suo, Jingyuan Chou, Weida Zhong, and Aidong Zhang. 2020. TAdaNet: Task-Adaptive Network for Graph-Enriched Meta-Learning. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3394486.3403230

1 INTRODUCTION

With the availability of big data and the growing computational power, machine learning has experienced tremendous growth in past decades. However, in many application scenarios, the number of annotated data can be extremely limited, where labeling data is typically very expensive and even unrealistic. For example, many healthcare tasks such as rare disease detection, where the prevalence of certain diseases in population is very low, suffer from limited training data. To tackle the data scarcity issue, few-shot learning has attracted great interest of the community.

Meta-learning, which is to acquire meta-knowledge across many tasks, has become an important and effective approach for few-shot learning in recent years. It assumes that some internal representations are transferable among tasks, so that the model learned from training tasks can be adapted to new tasks and produce good generalization performance. Most of existing meta-learning methods [6, 15, 28, 29, 31] rely on a globally shared meta-optimizer, initial parameter, or metric space across tasks. In the training process, a batch of tasks are sampled from a distribution, and transferable knowledge is learned in the form of optimization strategies, initialization, and embedding functions. However, a task distribution can be complex and has far apart mode, making it difficult to find a global condition that can be adapted quickly to desired parameters for all tasks. For example, if an intelligent system learns how to detect Alzheimer's disease, one can imagine that it is easier to be modulated to detect Parkinson's disease than other less related diseases such as heart failure. Therefore, it is desired that a meta-learner can accommodate tasks sampled from complex task distributions and customize parameters according to task relationships. Recent work [13, 14, 24, 32, 36, 39] has incorporated task contextual information to trigger task-specific initialization and embedding. Specifically, [36] proposes to cluster tasks based on task similarity, and promotes knowledge customization to different clusters. These models rely on the training data itself to learn task contextual information.

In many application scenarios, domain knowledge exhibits in the form of graphs, which can provide useful information to learn data representations and characterize task relationships. For example, in the healthcare domain, there are well organized disease ontologies such as the International Classification of Diseases[1](ICD) and Clinical Classifications Software[2](CCS), which provide the hierarchical relationships of diseases; and ImageNet is organized according to

[1]http://www.icd9data.com
[2]https://www.hcup-us.ahrq.gov/toolssoftware/ccs/ccs.jsp

the WordNet hierarchy[3]. This kind of graphs are often known in advance or can be extracted from a knowledge base. Nodes (e.g., diseases or words) sharing the same parent nodes are likely to be associated with similar data examples (e.g., patients or images), allowing knowledge transfer between them. The graph-enriched information helps representations learning especially when the data is limited in the setting of few-shot learning. Since a task contains several classes that are represented as nodes in the graph, each task can be regarded as a subset of nodes. Thus tasks are related through the paths that link their nodes. A task is considered more similar to another task sharing some nodes in the graph, than the one that has disjoint classes with it. Therefore, it is likely that the information in the domain-knowledge graph can enrich data representations and help in identifying task relationships.

In this paper, to incorporate such domain knowledge, we propose a task-adaptive meta learning framework named TAdaNet that allows message passing across nodes of a domain-knowledge graph and promotes meta knowledge customization for different tasks. Our model learns task embeddings by organizing task knowledge from historical tasks in a memory network, and produces task-aware parameter adjustment to customize the learner parameters conditioned on the task embeddings. The learner of each task produces a prototype per class for the classification task. Specifically, we use the class relationships on a given graph to learn prototypes by combining the neighborhood information through an attention mechanism. The data representations can be enriched by aggregating the information from their neighbors, and task relationships are captured by the paths linking classes on the graph. Our main contributions can be summarized as follows:

- We propose a novel meta-learning framework that allows message passing through external knowledge graphs to enrich data representations, and extracts task relationships reflected in the graph, in order to customize task-aware meta-learners for new tasks.
- We demonstrate that, comparing with the state-of-the-art approaches, our method effectively utilizes the graph-enriched information of ImageNet, and tailors the learned structure knowledge to the metric space per task for classification.
- We demonstrate that the proposed method can be effectively applied to a real-world application, i.e., disease prediction, using a large healthcare dataset by incorporating the domain knowledge of disease ontology. The results are promising for solving the data scarcity problem in healthcare decision support.

The paper is organized as follows. We first review the recent progress in meta-learning and its applications in related work, and then introduce the details of our proposed framework. In the experiments, we validate the proposed method on an extracted dataset from ImageNet, and extend the method to an application of disease prediction on a real-world healthcare dataset.

2 RELATED WORK

Meta-learning learns new tasks quickly with a few training examples. There are mainly three lines of approaches on meta-learning: (1) black-box amortized methods design meta-learners to quickly optimize the model parameters [1, 22, 23, 25]; (2) gradient-based methods learn an optimized initialization across tasks, allowing quick adaptation to new tasks by a one or more steps of gradient descent [6, 8, 15]; and (3) non-parametric learners aim to learn an appropriate distance metric between query and support examples [2, 7, 13, 28, 29, 31]. Meanwhile, there are a few hybrid work from the above categories [27, 33]. However, the tasks may be diverse and the generalization across the entire task distribution may be unuseful or even harmful to unrelated tasks.

Handling task heterogeneity in meta-learning has attracted substantial interests in recent advances. A handful of works [13, 14, 24, 32, 39] try to leverage task-specific information to tailor the shared knowledge for each task. [24] learns task-dependent metric space by scaling and shifting the feature extractors conditioned on the task sample set. [14] generates functional weights for the target prediction network of each task. [39] linearly projects sample embeddings to a task adaptive space, and calculates the distance metric in the projection space. [13] enables meta-learner to learn on each layer's activation space, so that task-specific learners perform gradient descent on their corresponding subspaces. [32] modulates the meta-learned prior parameters according to the mode of each task sampled from a multimodal task distribution. These approaches have shown the benefits of customizing task-specific representations compared to the globally shared parameters. However, they ignore the relationships between tasks, which may limit the model expressiveness and impair knowledge generalization. Considering task-relatedness, [36] proposes to cluster tasks into several states through hierarchical clustering, and trigger the initialization of each task through a cluster-specific parameter gate; and [37] constructs meta-knowledge graph to extract cross-task relations. The two methods learn task relationship from the training data, but ignore the inherent relationship expressed by external knowledge.

Our work is also related to graph neural network [30], message passing [16, 18] and graph meta-learning [16, 17, 38]. Specifically, [17] learns a global embedding network for all tasks to update the prototypes, and it covers the classification of any classes from the graph. In contrast, we perform classifications on some target classes from the graph, and utilize the graph structure to relate tasks and customize the metric space for different tasks.

Owing to the ability of learning with only a small amount of data, meta-learning has been applied to deal with the data scarcity problem in various fields, such as natural language processing [9], computer vision [11], recommendation systems [5, 12] and spatial-temporal prediction [35]. In the field of healthcare, [40] applied gradient-based meta-learning to predict the risk of target diseases with limited data samples from longitudinal patient health records, and has shown promising results of meta-learning in healthcare compared with training on limited samples directly, transfer learning and multi-task learning. Previous approaches [4, 20] on healthcare representation learning show that incorporating domain knowledge into the predictive model enriches data representations. Our proposed meta-learning method can be naturally applied into disease prediction tasks with limited training data, and improve prediction performance using well-organized disease ontology (e.g., CCS and ICD).

[3]https://wordnet.princeton.edu

3 METHODOLOGY

In this section, we first introduce the data structure and some basic notations, and then introduce details of the proposed framework on learning task-adaptive metric space with domain-knowledge graph for few shot classification.

3.1 Problem Setup

In the training phase of meta-learning, we are given a set of tasks $\{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, ...\}$ sampled from the task distribution $p(\mathcal{T})$. In each task $\mathcal{T}_i \sim p(\mathcal{T})$, we have a set of support examples $\mathcal{D}_i^{tr}$ and query examples $\mathcal{D}_i^{te}$. The support set $\mathcal{D}_i^{tr} = \{\mathbf{x}_{i,j}, y_{i,j}\}_{j=1}^{n^{tr}}$ contains N classes randomly selected from the meta dataset and K examples of each class, where $n^{tr} = N \times K$. The query set $\mathcal{D}_i^{te} = \{\mathbf{x}'_{i,j}, y'_{i,j}\}_{j=1}^{n^{te}}$ contains unseen examples from the classes in $\mathcal{D}_i^{tr}$, and the labels are to be learned. The goal is to learn a meta-learner that can produce a predictive model for each task. The optimal parameters are obtained by minimizing the expected empirical loss on training tasks. At the test time, the same strategy of task sampling is applied on meta-test data, which contains a disjoint set of target classes. The meta-test data is used to evaluate model generalization on unseen tasks.

In our problem, a graph $\mathcal{G}(\mathcal{Y}, \mathcal{E})$ indicating the relationship of classes is available. In the graph, each node $y \in \mathcal{Y}$ denotes a class and each directed edge $y_i \longrightarrow y_j \in \mathcal{E}$ connects a parent class y_i to its child class y_j on the graph. We assume that there are two types of nodes in $\mathcal{G}$: the leaf nodes $\mathcal{Y}^t$ are the target fine-grained classes; the ancestor nodes $\mathcal{Y}^c$ form the coarse classes. The fine-grained classes are the target classes which we are interested in for classification, and the coarse classes are expected to provide additional information that can help the few-shot learning process. For example, in the graph of disease ontology, "Parkinson's disease" has two children "Paralysis agitans" and "Secondary parkinsonism", while its ancestor classes are "Hereditary and degenerative nervous system conditions" and "Diseases of nervous system and sense organs". Following the settings of few-shot learning, we draw a task $\mathcal{T}_i \sim p(\mathcal{T})$ which contains classes in the leaf nodes as the target classes, and extract the ancestors for the target classes, which forms a subset of nodes in $\mathcal{G}$.

3.2 The Proposed Framework

The overall learning framework is shown in Figure 1. Our goal is to facilitate the learning process by leveraging the information on hierarchy graph and transferable knowledge from related tasks. The proposed framework contains two key modules: task context embedding and customized target prediction. During the meta-training phase, we first map the support and query examples to an embedding space through a feature extractor f_θ. Prototypes of the few-shot classes are calculated in the metric space and updated by propagating the message from ancestor nodes in the hierarchy graph. A task context encoder network g_φ encodes the task representations through aggregating the related information from a task-context memory $\mathcal{M}$. Meanwhile, the task representation is restricted to not deviate much from the one obtained by the graph-enriched prototypes. With the enhanced task representation, the initial parameter θ in the feature extractor is customized to a task-specific one θ_i by utilizing a set of modulating functions F_c. Few-shot classification is performed on the customized metric space by comparing the distances between the query example representation and prototypes of support classes.

In the following part of this section, we first introduce a graph-based method to learn class prototypes which enable message passing across classes, and then propose a task embedding method to learn the task relationship. Finally, parameters of the classification network are adapted to each task conditioning on its embedding.

3.2.1 Graph-based Prototype Learning. A prototype can be regarded as the representation of a certain class. Following [28], we obtain the initial prototype $\mathbf{c}_i^{k,0}$ of class k in task $\mathcal{T}_i$ by calculating the mean vector of sample embeddings in the class:

$$\mathbf{c}_i^{k,0} = \frac{1}{n_i^{tr,k}} \sum_{(\mathbf{x}_{i,j}, y_{i,j}) \in \mathcal{D}_i^{tr,k}} f_\theta(\mathbf{x}_{i,j}), \tag{1}$$

where $\mathcal{D}_i^{tr,k}$ is the support example set of the k-th class in task $\mathcal{T}_i$, $n_i^{tr,k}$ is the number of examples in the set, $\mathbf{x}_{i,j}$ is the j-th sample, and $f_\theta(\cdot)$ is the embedding function with learnable parameters θ. Eq. (1) calculates the prototype using limited samples in a class, which may be inaccurate and sensitive to outliers due to data scarcity of each class.

Therefore, we utilize the hierarchy graph to align related classes, in order to learn better class prototypes. In the hierarchy graph, the target few-shot classes come from the leaf nodes, while the coarse nodes from the ancestors indicate the distance between two target classes. For example, if two few-shot classes share some ancestor nodes in the hierarchy graph $\mathcal{G}$, they are likely to have some common characteristics, so that their prototypes should be similar. To incorporate the class relatedness information from graph $\mathcal{G}$, inspired by [17, 30], we update the prototype in the $(l+1)$-th iteration for each target class in the leaves, using its ancestors' embeddings in the last iteration as below,

$$\mathbf{c}_i^{k,l+1} = \sum_{p \in \mathcal{N}_k \cup k} att(\mathbf{c}_i^{p,l}, \mathbf{c}_i^{k,l}) \mathbf{c}_i^{p,l}, \tag{2}$$

where $att(a, b) = \frac{\exp(-d(a,b))}{\sum_{p'} \exp(-d(a,b))}$ is the attention score, $d(a, b)$ is the distance between a and b, and $\mathcal{N}_k$ denotes the neighbors of class k in $\mathcal{G}$. $\mathbf{c}_i^p$ contains the prototypes of a target class and its neighbors which are the parent nodes for leaves. We use cosine distance in our experiments. After getting the initial class embedding $\mathbf{c}_i^{k,0}$ in Eq. (1), we update it by aggregating the neighbor embeddings using Eq. (2). The updated prototype of class k in task $\mathcal{T}_i$ is,

$$\mathbf{c}_i^k = (1 - \lambda)\mathbf{c}_i^{k,0} + \lambda \mathbf{c}_i^{k,L}, \tag{3}$$

where the weight $\lambda \in [0, 1]$ is a hyperparameter to adjust the weight between information from the examples in class k and its ancestor classes, and L is the number of propagation iterations. We empirically find that the performance is insensitive to L, so we set L to 1 in the experiments for simplicity. Similarly, the prototype of a coarse node is calculated by aggregating the information from its ancestors, descendants and itself via graph attention.

3.2.2 Task Representation Learning. Metric-based few-shot learning methods [7, 28, 29, 31] typically learn a task-invariant metric for all the tasks. This may limit the model's expressive ability since

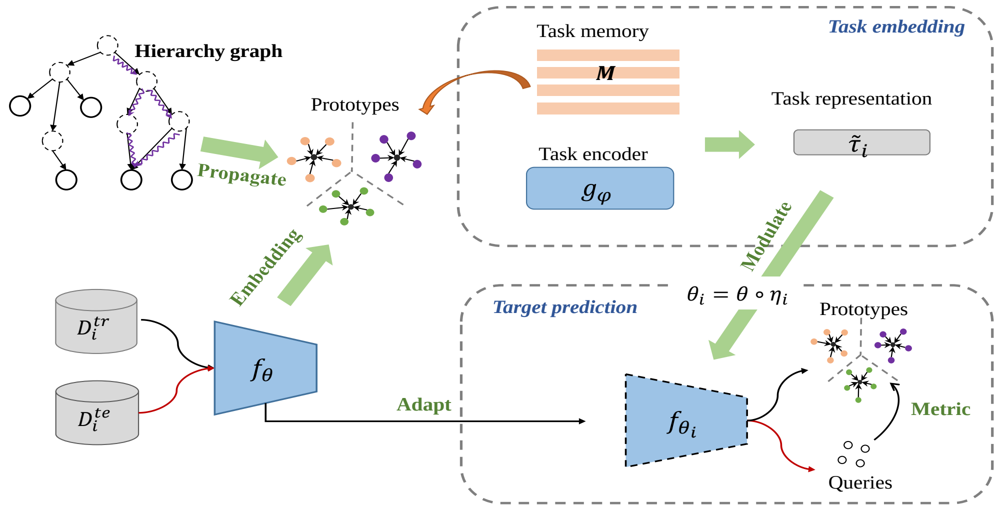

Figure 1: The overall framework. For each task $\mathcal{T}_i$, the support and query data samples are first embedded to a metric space through network $f_\theta(\cdot)$, and then the task encoder $g_\varphi(\cdot)$ extracts the task representation which is enhanced through memory network. A set of modulate networks $F_c(\cdot)$ generate the gate η for each parameter θ conditioned on the enhanced task representations. The customized parameter θ_i is used to obtain the prototype embeddings for input classes, and perform classification for query examples. In the process of learning prototypes, information from the hierarchy graph is passed to enhance the prototype representations.

the optimal metric may vary across different tasks. To improve the model expressiveness, we introduce parameter customization to make feature extractor behaviour task-adaptive. The task-specific parameters are conditioned on task representations. Learning a task representation is expected to account for the following properties: enough distinctions between different tasks, sufficient similarities between related tasks, and insensitivity to the number and order of samples in a certain task.

We use an auxiliary network $g_\varphi(\cdot)$ to learn task representations. Given an N-way K-shot task $\mathcal{T}_i$ with support examples $\mathcal{D}_i^{tr} = (\mathbf{x}_{i,j}, y_{i,j})$, we obtain the task representation by,

$$\tau_i = \frac{1}{NK}\sum_{j=1}^{NK} g_\varphi(\mathbf{x}_{i,j}). \tag{4}$$

The task representation τ_i is obtained by aggregating the example embeddings in the task with a mean pooling operation, e.g., $g_\varphi(\mathbf{x}_{i,j})$ is the representation of the j-th example.

We leverage the underlying knowledge obtained from historical tasks to enhance the task representation, in order to better characterize task relationships. To extract knowledge from historical learning process, we construct a parameterized memory matrix $\mathcal{M} \in \mathbb{R}^{S\times d}$, where S denotes the predefined knowledge types and d is the dimension of prototype embeddings. We utilize the knowledge patterns of task relations stored in $\mathcal{M}$ via attention mechanism. The attention score between prototype $\mathbf{c}_i^k$ of task i and the stored memory patterns is calculated by,

$$\alpha_s^k = \frac{\exp(\mathbf{c}_i^k \cdot \mathcal{M}(s))}{\sum_{s'=1}^{S}\exp(\mathbf{c}_i^k \cdot \mathcal{M}(s'))}, \tag{5}$$

where $\mathcal{M}(s)$ denotes the s-th pattern in the memory, and $\cdot$ is the dot-product operation. α_s^k measures the importance of each knowledge type in $\mathcal{M}$ to the k-th prototype in current task. The information passing from the memory is aggregated by,

$$\mathbf{z}_i^k = \sum_{s=1}^{S}\alpha_s^k \mathcal{M}(s), \tag{6}$$

where $\mathbf{z}_i^k$ is the information-propagated prototype. Therefore, the enhanced task representation $\tilde{\tau}_i$ can be obtained by concatenating the task embedding and the memory updated one as,

$$\tilde{\tau}_i = [\tau_i; \tau_i'], \tag{7}$$

where τ_i' is the mean vector of the updated prototypes of $\{\mathbf{z}_i^k\}_{k=1}^K$. Now the task representation $\tilde{\tau}_i$ is learned based on example embeddings within the task and historical tasks. Since the graph $\mathcal{G}$ provides class connections in the graph and tasks can be related via classes, we expect to leverage the graph information to better evaluate the task relatedness and allow knowledge shared across tasks by message passing through the graph. To incorporate the class connections in $\mathcal{G}$, we minimize the difference between the prototypes obtained by $g_\varphi(\cdot)$ and Eq. (3) as below:

$$\mathcal{L}_c(\mathcal{T}_i) = \sum_{k=1}^{K}\left\|\mathbf{c}_i^k - \frac{1}{n_i^{tr,k}}\sum_{\mathbf{x}_{i,j}\in\mathcal{D}_i^{tr,k}} g_\varphi(\mathbf{x}_{i,j})\right\|_F^2. \tag{8}$$

In this way, information from hierarchy graph is incorporated into task representation, so that task representations can reflect the class relatedness. If two tasks have similar class prototypes, they should have close task representations. With the learned task representation, we then perform task-specific customization for the prediction model of each task.

3.2.3 Task-specific Prediction. Previous gradient-based method [36] designed a cluster-specific parameter gate and allows similar tasks to be initialized by similar meta-parameters and then adapts the customized parameters to new tasks. Inspired by this, we learn a series of task-specific parameter gates:

$$\eta_i^j = F_c^j(\tilde{\tau}_i), \forall j \in 1, ..., |\theta|, \tag{9}$$

to enforce the task-specific property on the embedding network, where F_c^j is the j-th fully-connected network, and $|\theta|$ is the number of parameters in θ. Note that $\eta_i = \{\eta_i^j\}_{j=1}^{|\theta|}$ has the same dimension as the feature embedding parameter θ, so that each element in η_i controls the weight of neurons to be adapted to each task. With the parameter gates, the globally transferable knowledge is adapted to the task-specific embedding parameter $\theta_i = \theta \circ \eta_i$, where $\circ$ is the element-wise multiplication. Therefore, similar task embeddings will trigger similar parameter gates, resulting in similar model parameters and allowing more information to be shared, while different tasks are controlled to share less information.

With the customized feature extractor $f_{\theta_i}(\cdot)$, we embed the support and query examples in $\mathcal{T}_i$ to obtain the data representations conditioned on the task. We can then obtain the task-adaptive prototype $\tilde{\mathbf{c}}_i^k$ for class k following the same procedure as Section 3.2.1 with θ substituted by θ_i: first calculating the prototype per class by calculating the mean of supports to obtain the initial value, and then aggregating the information propagated through the ancestor nodes in $\mathcal{G}$. The prototype $\tilde{\mathbf{c}}_i^k$ is used to assign the predicted class label for a query data point $\mathbf{x}'_{i,j}$ based on a softmax function over distances between prototypes and the query embedding:

$$p(y'_{i,j} = k|\mathbf{x}'_{i,j}) = \frac{\exp(-d(f_{\theta_i}(\mathbf{x}'_{i,j}), \tilde{\mathbf{c}}_i^k))}{\sum_{k'} \exp(-d(f_{\theta_i}(\mathbf{x}'_{i,j}), \tilde{\mathbf{c}}_i^{k'}))}, \tag{10}$$

where the distance function $d(\cdot)$ is Euclidean distance. The classification process is to minimize the cross-entropy loss function as,

$$\mathcal{L}_p(\mathcal{T}_i) = -\sum_{k=1}^{K} \sum_{\mathbf{x}'_{i,j} \in \mathcal{D}_i^{te,k}} \log p(y'_{i,j} = k|\mathbf{x}'_{i,j}). \tag{11}$$

The overall loss is a weighted sum of the classification loss $\mathcal{L}_p$ and task encoder loss $\mathcal{L}_c$, i.e., $\mathcal{L}_p + \alpha\mathcal{L}_c$, where α is a hyperparameter. The training procedure is shown in Algorithm 1.

Algorithm 1 TAdaNet for meta-learning

```
Require: task distribution $p(\mathcal{T})$; hierarchy graph $\mathcal{G}$; meta training data.
 1: Randomly initialize $\theta$, $\varphi$ and $\mathcal{M}$
 2: for number of training iterations do
 3:     Sample a batch of tasks from $p(\mathcal{T})$ and sample $\mathcal{D}_i^{tr}$ and $\mathcal{D}_i^{te}$ for each task $\mathcal{T}_i$
 4:     for all $\mathcal{T}_i$ do
 5:         Calculate the prototype $\mathbf{c}_i^k$ for each support class $k$ by Eq. (1), Eq. (2) and Eq. (3)
 6:         Compute task representation $\tilde{\tau}_i$ using Eq. (4), Eq. (5), Eq. (6) and Eq. (7)
 7:         Compute the task encoder loss $\mathcal{L}_c(\mathcal{T}_i)$ by Eq. (8)
 8:         Compute $\eta_i^j$ by Eq. (9) and update parameters $\theta_i \leftarrow \eta_i\theta$
 9:         Compute the customized prototype $\tilde{\mathbf{c}}_i^k$ by $f_{\theta_i}(\cdot)$, and the classification loss $\mathcal{L}_p(\mathcal{T}_i)$ by Eq. (11)
10:     end for
11:     Update $\theta$, $\varphi$ and $\mathcal{M}$ by minimizing $\mathcal{L}_p + \alpha\mathcal{L}_c$.
12: end for
```

4 EXPERIMENTS

We conduct experiments on two datasets, including an image dataset extracted from *tiered*ImageNet [26] and a medical dataset extracted from MIMIC[4]. We compare the proposed method with state-of-the-art meta-learning approaches on the N-way K-shot classification problem, and present the quantitative and qualitative experimental results.

4.1 Experimental Setup

In this section, we first describe the data preprocessing steps on the two databases to extract the data specifically designed for graph-based meta-learning, and then introduce the state-of-the-art meta-learning approaches which are used as baselines. Finally, we provide the implementation details of the experiments.

4.1.1 Datasets. We carry out the experiments on two datasets separately and utilize their knowledge graphs.

Image-graph We extract a subset from *tiered*ImageNet [26], which is a subset of ILSVRC0-12 and has a total of 608 classes (named leaf categories) and 779,165 images. Each non-leaf category (named coarse node) contains 10 to 30 classes. A whole directed acyclic graph is firstly built from root node to leaf nodes based on WordNet. Starting from each class whose distance from root node is 4, we build a subgraph that contains all its descendants. Then for each subgraph, we randomly select 80% of its leaf classes as training and the rest 20% as testing, and divide the subgraph into two graphs, i.e., train-graph that only contains training leaf classes and their ancestors, and test-graph which only contains testing leaf classes and their ancestors. For each leaf class in train/test-graph, we randomly sample 20 images belonging to that category. Then for each coarse class, we sample 20 images from each of its descendants that are leaf classes. Each image will be picked up at most once. To ensure that there is enough difference between different subgraphs, we remove leaf classes that are too close to other subgraphs. Here, the distance between a leaf node and a subgraph is defined as the minimum hops between the node and all leaves in the subgraph.

MIMIC Multiparameter Intelligent Monitoring in Intensive Care (MIMIC-III) is a large publicly available dataset for medical analysis. We select 446 diagnosis codes which serve as the labels for disease prediction, and ensure that different target disease cohorts have no overlapped patients. For each patient, we extract two types of features: time series variables and discrete variables. The time series are informative physiological variables suggested in [10, 34]

[4]https://mimic.physionet.org/

such as heart rate, systolic blood pressure, oxygen saturation, and respiratory rate. We use the records of 31 variables during the first 48 hours after patients' admission to the ICU. Therefore, the time series record of each patient can be viewed as a matrix, where the horizontal dimension corresponds to timestamps and the vertical dimension is physiological variables. The discrete variables include 3,103 ICD-9 codes with the 446 selected disease codes being removed. The discrete variables are represented as a binary vector, where each element indicates the absence or occurrence of a variable. We use ICD ontology as the external graph which provides category relationship among diseases. Detailed statistics of two datasets are shown in Table 1.

Table 1: Data statistics of two datasets.

Dataset		training		testing	
		#classes	#examples	#classes	#examples
Image-graph	coarse	388	55,800	190	13,580
	target	441	8,820	113	2,260
	total	829	64,620	303	15,840
MIMIC	coarse	155	11,099	74	4,731
	target	171	11,428	46	2,902
	total	326	22,527	120	7,651

4.1.2 Baseline Approaches. We compare the proposed method with two types of baselines, including gradient-based meta-learning methods and metric-based methods. For the gradient-based methods, we compare with MAML [6], MMAML [32] and HSML [36]. Among the methods, MAML learns globally shared initial parameters across tasks and then adapt the parameters to new tasks through a few gradient steps, while MMAML and HSML learn task-specific initialization to customize the learning process. The metric-based methods to be compared include MatchingNet [31], ProtoNet [28], RelationNet [29] and PPN [16]. These methods measure distances between query and support examples using a shared metric for all tasks. Among the methods, MatchingNet produces a weighted k nearest neighbor classifier; ProtoNet computes distances between a query point and prototypes; RelationNet learns a nonlinear comparison on top of metric space; and PPN learns the propagated prototypes by accumulating the level-wise classification loss on each level of classes sampled from the graph.

4.1.3 Implementation Details. We divide the target classes for few-shot classification into training and testing sets in an 8:2 ratio. For each target class, we retrieve its ancestor nodes from the hierarchy graph to form the coarse classes. The target classes in the training and testing sets are disjoint, while we allow them to share ancestors according to the hierarchy. During the training phase, we sample N target classes from the leaf nodes to form a task, and retrieve their corresponding coarse nodes. For evaluation purpose, we randomly sample 600 tasks from the meta-test dataset.

For the image data, we adopt the widely used backbone CNN architecture as in previous work [6, 28, 31]. It has 4 convolutional layers, each with 64 filters of kernel size 3×3, followed by batch normalization, ReLU activation and 2×2 max-pooling. For metric-based methods, $f_\theta(\cdot)$ is used to extract features from the input images, and the learned feature representations are used for distance measurement between query and support images. For gradient-based methods, we add a fully-connected layer on top of the feature extractor to produce the probabilities as a prediction for each data example. For the proposed method, the task encoder $g_\varphi(\cdot)$ contains two 3×3 convolutional layers and each with 64 filters. The modulation network $F_c(\cdot)$ is a set of one-layer projections with sigmoid activation to customize each parameter in $f_\theta(\cdot)$. We empirically find that more layers of $F_c(\cdot)$ do not make significant improvement.

For the medical data, the network $f_\theta(\cdot)$ contains a CNN-based sequence learning structure for time series and a multilayer perceptron (MLP) for discrete variables. We adopt the one-directional convolution operations in risk prediction [3, 19, 40] to capture patterns across the temporal dimension, followed by max pooling and a fully connected layer. We use a one-layer CNN network with 64 filters for time series and a two-layer MLP for discrete data, and combine the learned vector representations through a fully-connected layer. The task encoder $g_\varphi(\cdot)$ has the same structure as $f_\theta(\cdot)$ to learn example embeddings and we use mean pooling to obtain the task embedding. The modulation network $F_c(\cdot)$ is a set of one-layer projections.

4.2 Experimental Results

4.2.1 Performance Evaluation. To evaluate the model's generalization performance, tasks are generated using two sampling strategies: random sampling and subgraph sampling. In random sampling, the few-shot classes are randomly selected from the leaf nodes of the whole graph. In subgraph sampling, we split the graph into several subgraphs without overlapping target classes, and each task is sampled from nodes within one subgraph. In this setting, tasks are constructed heterogenously, and tasks sampled from the same subgraph can be considered more similar to each other, while tasks from different subgraphs are considered dissimilar. Subgraph sampling corresponds to an application scenario which requires discriminating fine-grained classes (e.g., Parkinson's, Alzheimer's, and Huntington's disease) from the same category (e.g., neurodegenerative diseases).

We conduct experiments on N-way K-shot few-shot classification problems. Results under two sampling strategies on Image-graph and MIMIC datasets are shown in Table 2 and Table 3 respectively. In Table 2, we compare the proposed method with baseline approaches of the few-shot image classification problem on 5-way 1-shot, 5-way 5-shot and 10 way 1-shot. We set λ in Eq. (3) to 0.5. Our method significantly outperforms baseline approaches, especially on 1-shot classification. This is due to the fact that 1-shot learning is more difficult than 5-shot learning for lack of training data, so that the information from ancestors of the graph can be more helpful to learn representations for the target classes.

We observe that MMAML usually achieves higher accuracy than MAML, especially under subgraph sampling. The reason is that MMAML augments MAML with the capability to identify the mode of tasks sampled from complex task distributions. This indicates that task-specific initialization can produce an improvement over the global initialization. PPN propagates the the prototype of each class to its child classes on the graph and buffers the prototypes for further updates. It outperforms most of the baseline approaches,

Table 2: Comparison between the proposed and baseline approaches on Image-graph of image classification problem. Accuracy ±95% confidence intervals under two sampling strategies are reported.

Methods	Subgraph sampling			Random sampling		
	5-way 1-shot	5-way 5-shot	10-way 1-shot	5-way 1-shot	5-way 5-shot	10-way 1-shot
MAML	38.12±1.06%	53.82±1.14%	21.15±0.91%	47.50±1.04%	60.90±1.10%	32.70±0.67%
MMAML	39.60±0.84%	54.23±1.19%	23.05±1.16%	47.34±1.05%	61.96±0.88%	32.40±1.20%
HSML	38.14±1.01%	54.85±1.08%	21.36±1.03%	46.56±1.01%	62.05±0.98%	32.68±0.66%
MatchingNet	38.21±1.08%	50.64±1.07%	25.16±0.75%	47.37±1.04%	64.03±0.81%	33.87±0.64%
ProtoNet	38.50±1.03%	54.31±1.06%	25.29±0.63%	46.95±1.04%	65.73±0.96%	33.65±0.69%
RelationNet	36.46±1.06%	51.87±1.01%	23.40±0.73%	49.94±1.06%	65.89±0.88%	34.16±0.64%
PPN	45.08±1.04%	53.32±1.05%	36.32±1.23%	56.35±1.01%	65.90±0.92%	49.47±1.31%
TAdaNet	**48.85±1.17%**	**55.29±1.13%**	**38.55±1.31%**	**60.01±0.98%**	**69.38±0.90%**	**52.20±1.24%**

Table 3: Comparison between the proposed and baseline approaches on MIMIC of disease classification problem. Accuracy ±95% confidence intervals under two sampling strategies are reported.

Methods	Subgraph sampling			Random sampling		
	3-way 1-shot	3-way 5-shot	5-way 1-shot	3-way 1-shot	3-way 5-shot	5-way 1-shot
MAML	40.54±0.95%	45.47±0.94%	26.29±0.61%	46.20±0.97%	53.70±0.94%	31.97±0.74%
MMAML	41.21±0.79%	46.32±1.01%	26.94±0.62%	45.91±0.98%	54.93±0.90%	32.25±0.78%
HSML	40.97±0.41%	45.76±0.86%	27.01±0.70%	45.94±0.84%	53.14±0.41%	31.04±0.70%
MatchingNet	39.12±0.70%	43.26±0.80%	26.08±0.57%	43.87±0.77%	50.92±0.92%	29.64±0.63%
ProtoNet	38.68±0.68%	46.24±0.97%	25.79±0.54%	42.92±0.72%	54.08±1.00%	29.61±0.65%
RelationNet	39.00±0.87%	42.46±0.96%	21.38±0.43%	43.85±0.89%	52.23±0.95%	28.67±0.61%
PPN	45.59±0.85%	50.55±0.99%	30.67±0.67%	51.54±0.90%	58.16±0.93%	38.59±0.68%
TAdaNet	**49.74±0.92%**	**52.05±0.91%**	**32.56±0.67%**	**54.06±0.94%**	**59.05±0.92%**	**40.31±0.72%**

especially under 1-shot learning. This indicates the effectiveness of learning from external information in the graph. Besides allowing message passing through graphs for prototype learning, the proposed method TAdaNet also learns task relationships and customizes model parameters for different tasks, so that it outperforms baseline approaches. We also observe that the performances of all methods under subgraph sampling are generally lower than those under random sampling. The reason is that in subgraph sampling, classes that form a task come from the same subgraph and share some similarities. Distinguishing these classes is more difficult than distinguishing irrelevant classes randomly sampled from the data. For example, distinguishing "heart failure" from "myocardial infarction" is more difficult than from bone disease. However, learning task relationships and customizing task-specific model parameters can be more helpful under subgraph sampling compared with baseline approaches, as our method can trigger similar initial parameters for tasks sampled from the same subgraph.

In Table 3, we show the performance comparison of different methods on disease classification problem under 3-way 1-shot, 3-way 5-shot and 5-way 1-shot settings. λ is set to 0.8. In this problem, each task is comprised of sampling 3 or 5 diseases from the target nodes of the disease ontology, and each disease contains a few patient examples. Due to the sparsity and complexity of healthcare data, disease classification is a more difficult problem than image classification. The relative comparison between methods is consistent with Table 2, and our method achieves the best performance.

4.2.2 *Ablation Study.*

To evaluate the contribution of different components in the proposed framework, we conduct the following experiments. TAdaNet-ta is the proposed model without task-specific customization part, and the method learns task invariant prototypes for prediction. TAdaNet-mem is the one without memory network, so that task embeddings do not retrieve historical stored information. TAdaNet-reg has no task-relational regularization term $\mathcal{L}_c$, so that task embeddings are learned independently, without explicitly considering task relations in the graph.

The results on two datasets are shown in Table 4 and Table 5. Comparing TAdaNet-ta with TAdaNet, we observe that the performance drops dramatically after removing the task-specific customization part. This shows the importance of customizing task-specific parameters. With the customized parameters, tasks can be quickly adapted to their optimized parameters, resulting better results than using global parameters. Comparing TAdaNet-reg with TAdaNet, we see that using graph structures to regularize the task representations which better identifies the relationships among tasks, is beneficial to the classification. Also, retrieving information from historical tasks helps to enhance task representations, as indicated by comparing with TAdaNet-mem.

4.2.3 *Visualization.*

In Figure 2, we randomly sample 600 tasks from three subgraphs on Image-graph, and use t-SNE [21] to visualize the task embedding vectors learned by HSML and TAdaNet. The root nodes of the three subgraphs which can be viewed as the

Table 4: Ablation study on Image-graph.

Methods	Subgraph sampling	Random sampling
	5-way 1-shot	5-way 1-shot
TAdaNet-ta	44.36±1.07%	54.48±1.05%
TAdaNet-mem	45.38±1.15%	56.87±1.01%
TAdaNet-reg	47.29±1.15%	57.50±0.98%
TAdaNet	48.85±1.17%	60.01±0.98%

Table 5: Ablation study on MIMIC.

Methods	Subgraph sampling	Random sampling
	3-way 1-shot	5-way 1-shot
TAdaNet-ta	45.98±0.85%	32.56±0.67%
TAdaNet-mem	45.36±1.20%	38.25±0.66%
TAdaNet-reg	47.30±0.90%	37.05±0.66%
TAdaNet	49.74±0.92%	40.31±0.72%

semantics of tasks are "domestic animal", "chordate" and "instrumentation". Since tasks sampled from the same subgraph are more related to each other, their embeddings are closer to each other. As in Eq. (9), tasks with closer representations in the embedding space can produce similar parameter gates η and trigger similar model parameters for classification; while far apart tasks will make the model parameters more different, allowing less information to be shared among each other. The results indicate the our method is able to learn the similarity relations among tasks. HSML is a method that can identify tasks in different clusters using hierarchical clustering. Since HSML relies on handcrafted design of hierarchical clustering structure, the task relationship learned by HSML may not accurately reflect the graphical structure of the data.

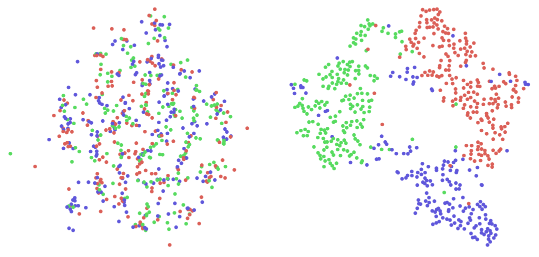

Figure 2: t-SNE visualization of task embeddings under subgraph sampling on Image-graph dataset. Dots with the same color are the tasks sampled from the same subgraph.

4.2.4 Parameter Study. We evaluate the performance of the proposed method with respect to different values of two hyperparameters, i.e., λ and α.

Parameter λ. It is used in Eq. (3) for balancing the contribution of the initial prototype to the final updated one. The trends on Image-graph and MIMIC datasets are depicted in Figure 3a and Figure 3b, respectively. In Figure 3a, under the setting of 5-way 1-shot, the accuracy value improves with respect to λ for both random sampling and subgraph sampling, while it does not change much under 5-way 5-shot. Similar observations can be found in Figure 3b. This indicates that more information from the ancestors can help the learning process, especially for 1-shot learning. The parameter study on λ suggests that graph information can be more helpful when there are very limited data examples.

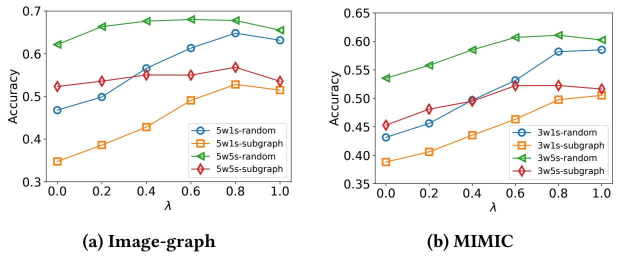

Figure 3: Accuracy with respect to λ on two datasets.

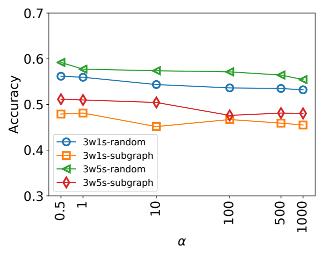

Figure 4: Accuracy with respect to α on MIMIC.

Parameter α. We also investigate the performance in terms of different values of the loss balancing term α on MIMIC dataset. α controls the weight of the task encoder loss obtained from Eq. (8) in the overall loss. The results are shown in Figure 4. λ is set to 0.8 in the experiments. Various values of α have been selected in the range of $[0.5, 1000]$. To make the figure readable, log 10 is applied to the X-axis. From the figure, we can see that the model is insensitive to α. However, the performances under four settings slightly decrease when α is large. This may be due to the fact that a large α restricts the task embeddings to be the mean of class prototypes, while it could be hard to identify task relationships if the class representations have large variance. Therefore, using the task encoder network rather than directly using the mean of prototypes brings more flexibility to incorporate task characteristics.

5 CONCLUSION

In this paper, we proposed a task-adaptive metric-based meta-learning framework called TAdaNet, to facilitate meta-learning for handling tasks sampled from complex distributions. In the problem, a hierarchical graph whose leaf nodes are the target classes for few-shot learning is given. The coarse nodes in the graph provide extra information which can help in handling the data insufficiency issue. Moreover, the paths that relate tasks on the graph are helpful in identifying the task relationships. The proposed approach utilizes the graph structure in two ways: refining each class's prototype by aggregating the information from its ancestors' prototypes on the graph; and regularizing the task representations to trigger task-specific customization for model parameters. In the experiments, we compared the proposed method with various state-of-the-art approaches under the setting of complex task distributions. Experimental results show that TAdaNet can effectively utilize the graph-enriched information to learn more accurate class prototypes, and identify the task relations to adapt task-specific metrics for classification. We also applied the proposed approach to a real-world problem, i.e., disease prediction. We utilized the well-organized disease ontology to provide graphical information for TAdaNet. The experimental analysis on two completely different applications demonstrates the generality of the proposed approach.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments and helpful suggestions. This work is supported in part by the US National Science Foundation under grants IIS-1924928, IIS-1938167 and OAC-1934600. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. 2016. Learning to learn by gradient descent by gradient descent. In *Advances in neural information processing systems*. 3981–3989.

[2] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. 2019. A Closer Look at Few-shot Classification. In *ICLR*.

[3] Yu Cheng, Fei Wang, Ping Zhang, and Jianying Hu. 2016. Risk prediction with electronic health records: A deep learning approach. In *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM, 432–440.

[4] Edward Choi, Mohammad Taha Bahadori, Le Song, Walter F Stewart, and Jimeng Sun. 2017. GRAM: graph-based attention model for healthcare representation learning. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 787–795.

[5] Zhengxiao Du, Xiaowei Wang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Sequential Scenario-Specific Meta Learner for Online Recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2895–2904.

[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1126–1135.

[7] Victor Garcia and Joan Bruna. 2018. Few-shot learning with graph neural networks. In *ICLR*.

[8] Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. 2018. Recasting gradient-based meta-learning as hierarchical bayes. *arXiv preprint arXiv:1801.08930* (2018).

[9] Jiatao Gu, Yong Wang, Yun Chen, Kyunghyun Cho, and Victor OK Li. 2018. Meta-learning for low-resource neural machine translation. In *EMNLP*.

[10] Hrayr Harutyunyan, Hrant Khachatrian, David C Kale, Greg Ver Steeg, and Aram Galstyan. 2017. Multitask learning and benchmarking with clinical time series data. *arXiv preprint arXiv:1703.07771* (2017).

[11] Bingyi Kang, Zhuang Liu, Xin Wang, Fisher Yu, Jiashi Feng, and Trevor Darrell. 2019. Few-shot object detection via feature reweighting. In *Proceedings of the IEEE International Conference on Computer Vision*. 8420–8429.

[12] Hoyeop Lee, Jinbae Im, Seongwon Jang, Hyunsouk Cho, and Sehee Chung. 2019. MeLU: Meta-Learned User Preference Estimator for Cold-Start Recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1073–1082.

[13] Yoonho Lee and Seungjin Choi. 2018. Gradient-based meta-learning with learned layerwise metric and subspace. In *ICML*.

[14] Huaiyu Li, Weiming Dong, Xing Mei, Chongyang Ma, Feiyue huang, and Baogang Hu. 2019. LGM-Net: Learning to Generate Matching Networks for Few-Shot Learning. In *ICML*.

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

[16] Lu Liu, Tianyi Zhou, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. 2019. Prototype propagation networks (PPN) for weakly-supervised few-shot learning on category graph. In *IJCAI*.

[17] Lu Liu, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Learning to propagate for graph meta-learning. In *Advances in Neural Information Processing Systems*. 1037–1048.

[18] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sung Ju Hwang, and Yi Yang. 2019. Learning to propagate labels: Transductive propagation network for few-shot learning. In *ICLR*.

[19] Fenglong Ma, Jing Gao, Qiuling Suo, Quanzeng You, Jing Zhou, and Aidong Zhang. 2018. Risk prediction on electronic health records with prior medical knowledge. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1910–1919.

[20] Fenglong Ma, Quanzeng You, Houping Xiao, Radha Chitta, Jing Zhou, and Jing Gao. 2018. Kame: Knowledge-based attention model for diagnosis prediction in healthcare. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 743–752.

[21] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[22] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2018. A simple neural attentive meta-learner. In *ICLR*.

[23] Tsendsuren Munkhdalai, Xingdi Yuan, Soroush Mehri, and Adam Trischler. 2018. Rapid adaptation with conditionally shifted neurons. In *ICML*.

[24] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. 2018. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems*. 721–731.

[25] Sachin Ravi and Hugo Larochelle. 2017. Optimization as a model for few-shot learning. In *ICLR*.

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

[27] Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. 2019. Meta-learning with latent embedding optimization. In *ICLR*.

[28] Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*. 4077–4087.

[29] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. 2018. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1199–1208.

[30] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.

[31] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, and Daan Wierstra. 2016. Matching networks for one shot learning. In *Advances in neural information processing systems*. 3630–3638.

[32] Risto Vuorio, Shao-Hua Sun, Hexiang Hu, and Joseph J Lim. 2019. Multimodal Model-Agnostic Meta-Learning via Task-Aware Modulation. In *Advances in Neural Information Processing Systems*. 1–12.

[33] Duo Wang, Yu Cheng, Mo Yu, Xiaoxiao Guo, and Tao Zhang. 2019. A Hybrid Approach with Optimization and Metric-based Meta-Learner for Few-Shot Learning. *NeuroComputing* (2019).

[34] Lu Wang, Wei Zhang, Xiaofeng He, and Hongyuan Zha. 2018. Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2447–2456.

[35] Huaxiu Yao, Yiding Liu, Ying Wei, Xianfeng Tang, and Zhenhui Li. 2019. Learning from multiple cities: A meta-learning approach for spatial-temporal prediction. In *The World Wide Web Conference*. 2181–2191.

[36] Huaxiu Yao, Ying Wei, Junzhou Huang, and Zhenhui Li. 2019. Hierarchically Structured Meta-learning. In *ICML*.

[37] Huaxiu Yao, Xian Wu, Ruirui Li, Zhiqiang Tao, Yaliang Li, Bolin Ding, and Zhenhui Li. 2020. Automated Relational Meta-learning. In *ICLR*.

[38] Huaxiu Yao, Chuxu Zhang, Ying Wei, Meng Jiang, Suhang Wang, Junzhou Huang, Nitesh V Chawla, and Zhenhui Li. 2019. Graph Few-shot Learning via Knowledge Transfer. *arXiv preprint arXiv:1910.03053* (2019).

[39] Sung Whan Yoon, Jun Seo, and Jaekyun Moon. 2019. TapNet: Neural Network Augmented with Task-Adaptive Projection for Few-Shot Learning. In *ICML*.

[40] Xi Sheryl Zhang, Fengyi Tang, Hiroko H Dodge, Jiayu Zhou, and Fei Wang. 2019. Metapred: Meta-learning for clinical risk prediction with limited patient electronic health records. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2487–2495.

A APPENDIX

A.1 MIMIC Data Preprocessing

MIMIC database contains medical records of around 60,000 patients in critical care units, involving over 6,000 diseases represented by ICD-9 diagnosis codes. To extract patient cohorts for disease prediction task, we first remove common disease codes according to their frequency of occurrence, and then select target diseases based on their ICD-9 code distribution. We ensure that the selected disease cohorts do not have overlapped patients, and for each disease cohort, there are at least 20 patients. Table 6 shows a subset of our selected diseases. In MIMIC, each patient may have more than one hospital admissions. In this work, we consider each visit as a unique data example.

The time series variables are the physiological features extracted from tables in MIMIC including lab values, chart events and vital signs. We select informative variables following medical references, and remove those variables with more than 50% missing rate. Finally, there are 31 physiological variables used in this work including: HR, systolic blood pressure, DBP, mean BP, respiratory rate, oxygen saturation, temperature, partial pressure of carbon dioxide, ph, lactic acid, co2, pos end pressure set, tidal volume observed, peak inspiratory pressure, hemoglobin, hematocrit, white blood cell count, chloride, creatinine, glucose, magnesium, sodium, blood urea nitrogen, calcium, phosphorous, platelets, prothrombin time pt, partial thromboplastin time, prothrombin time, weight and potassium. To make sure that all patients are on the same page, we extract data from the first 48 hours duration after patients' admission to the ICU. We impute the missing variables with k-nearest neighbors first, and fill the remaining missing values with the mean values of corresponding variables.

The discrete variables are the remaining ICD-9 codes with disease indicator codes being removed. The discrete variables contain symptoms, complications and related procedures which provide useful information for disease identification. As in MIMIC-III, most patients have only one visit and we consider each visit as a data example, the discrete features can be represented by a binary vector. We extract 3,103 ICD-9 codes in this work, so that each example is a 3,103 dimensional binary vector, with "1" indicating the occurrence of a certain code, and "0" otherwise.

A.2 Experimental Settings

A.2.1 Backbone Network for MIMIC Dataset. For the disease prediction task on MIMIC, we use a CNN-based sequence learning structure for time series and a multilayer perceptron (MLP) for discrete variables. For time series, we adopt the one-directional convolution operations to capture patterns across the temporal dimension. For all the baseline approaches and proposed method, we use a one-layer CNN network with 64 filters for time series and a two-layer MLP for discrete data, and combine the learned vector representations through a fully-connected layer to produce the prediction. The network structure is shown in Figure 5.

A.2.2 Baseline Methods Description. The details of the baseline methods are as follows:

• MAML [6] learns a shared initialization of a backbone model's parameters and achieves fast adaptation on new tasks. For inner gradient updates, we set the step size to 0.05 and the number of gradient updates to 5. For outer loop, we set the learning rate to 0.001.

• MMAML [32] augments MAML with the capability to identify the mode of tasks sampled from a heterogeneous task distribution. It modulates the meta-learned prior parameters through task-specific

Table 6: A subset of target diseases for classification.

Disease	Category	ICD-9
Osteomyelitis	musculoskeletal	730
Osteitis deformans	musculoskeletal	731
Curvature of spine	musculoskeletal	737
Contact dermatitis	skin and subcutaneous	692
Erythematous	skin and subcutaneous	695
Acute renal failure	genitourinary system	584
Chronic renal failure	genitourinary system	585
Disorders from renal failure	genitourinary system	588
Vascular insufficiency of intestine	digestive system	557
Gastritis and duodenitis	digestive system	535
Herpetic whitlow	infectious and parasitic	546
Hypotension	circulatory system	458
Atherosclerosis	circulatory system	440
Cerebrovascular disease	circulatory system	433
Acute pulmonary heart disease	circulatory system	415
Hypertensive heart and renal disease	circulatory system	404
Diseases of mitral valve	circulatory	394
Vertiginolls syndrome	nervous and sense	386
Disorders of conjunctiva	nervous and sense	372
Blindness	nervous and sense	369
Myoneural disorders	nervous and sense	358
paralytic syndromes	nervous and sense	344
cerebral degenerations	nervous and sense	331
Bacterial meningitis	nervous and sense	320
Mild mental retardation	mental disorders	317
Drug psychoses	mental disorders	292
Disorders of adrenal glands	immunity disease	255

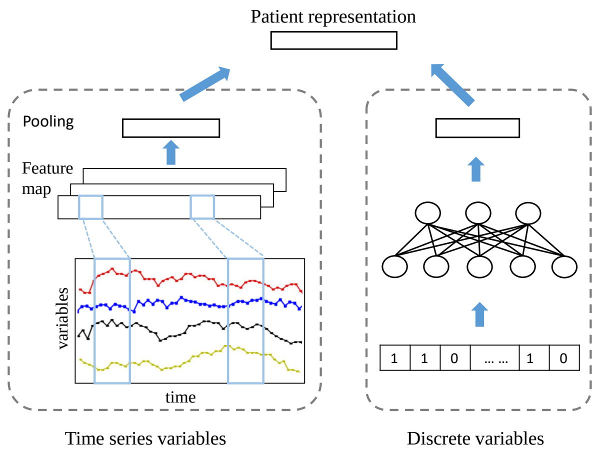

Figure 5: Backbone model structure for MIMIC dataset.

ebmeddings, to better adapt to tasks with different modes. The task embedding network we adopt is LSTM aggregator, and the modulation operation is feature-wise linear modulation (FiLM).

• HSML [36] pursues a hierarchically structured meta learning framework to balance generalization and customization of transferable knowledge. It preserves knowledge generalization among a cluster of similar tasks and customizes knowledge to different clusters of tasks. We adopt the pooling autoencoder aggregator to learn task representations for simplicity.

• MatchingNet [31] employs the idea of metric learning and uses attention mechanism to enable rapid learning.

• ProtoNet [28] learns a metric space in which classification is performed by computing distances between a query vector to prototypes of each class. A prototype is the mean vector of the embedded support points belonging to a certain class.

• RelationNet [29] performs classification by computing the pairwise relation scores between query images and the few support examples of each class. The feature maps obtained from the CNN backbone are concatenated and further learned through two CNN layers and a fully connected (FC) layer to obtain the relation score. It is particularly designed for image classification, so we modify the relation learning module for MIMIC dataset. We concatenate the learned feature maps of the time series part, followed by a CNN layer and an FC layer. Meanwhile, we also concatenate the vector representations of discrete variables, followed by an FC layer. The relation score is obtained by learning from the two parts.

• PPN [16] uses weakly-labeled data for meta-learning on few-shot classification. It learns an attention mechanism which propagates the prototype of one class to another on a given graph. Both PPN and TAdaNet adopt prototype propagation. However, we do not rely on level-wise training on subgraphs and buffered prototypes for update. TAdaNet learns task-adaptive metric by considering the task relationships provided on the given graph.

All deep models in this paper are implemented by PyTorch 1.4[5]. Since the original HSML is based on Tensorflow[6], we re-implement it to account for our experimental setting. All the models use the same backbone structure. For all methods, adam optimizer is used with learning rate 0.001. Early stopping strategy is adopted when the validation accuracy does not improve for 10 epochs.

[5]https://pytorch.org/
[6]https://www.tensorflow.org/