Retaining Privileged Information for Multi-Task Learning

Fengyi Tang
Michigan State University
East Lansing, MI, USA
tangfeng@msu.edu

Cao Xiao
Analytics Center of Excellence, IQVIA
Cambridge, MA, USA
cao.xiao@iqvia.com

Fei Wang
Weill Cornell Medical College
New York, NY, USA
feiwang.cornell@gmail.com

Jiayu Zhou
Michigan State University
East Lansing, MI, USA
jiayuz@msu.edu

Li-wei H. Lehman
MIT-IBM Watson AI Lab &
Massachusetts Institute of Technology
Cambridge, MA, USA
lilehman@mit.edu

ABSTRACT

Knowledge transfer has been of great interest in current machine learning research, as many have speculated its importance in modeling the human ability to rapidly generalize learned models to new scenarios. Particularly in cases where training samples are limited, knowledge transfer shows improvement on both the learning speed and generalization performance of related tasks. Recently, *Learning Using Privileged Information* (LUPI) has presented a new direction in knowledge transfer by modeling the transfer of prior knowledge as a Teacher-Student interaction process. Under LUPI, a Teacher model uses Privileged Information (PI) that is only available at training time to improve the sample complexity required to train a Student learner for a given task. In this work, we present a LUPI formulation that allows privileged information to be retained in a multi-task learning setting. We propose a novel feature matching algorithm that projects samples from the original feature space and the privilege information space into a *joint latent space* in a way that informs similarity between training samples. Our experiments show that useful knowledge from PI is maintained in the latent space and greatly improves the sample efficiency of other related learning tasks. We also provide an analysis of sample complexity of the proposed LUPI method, which under some favorable assumptions can achieve a greater sample efficiency than brute force methods.

CCS CONCEPTS

• **Computing methodologies** → **Multi-task learning**; • **Applied computing** → **Health informatics**.

KEYWORDS

Privileged Information; Multi-Task Learning; Electronic Health Records

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330907

ACM Reference Format:
Fengyi Tang, Cao Xiao, Fei Wang, Jiayu Zhou, and Li-wei H. Lehman. 2019. Retaining Privileged Information for Multi-Task Learning. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'19), June 22–24, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330907

1 INTRODUCTION

In classical supervised learning, the learner is presented with the training tuple $\{(x_i, y_i)\}_{i=1}^m$ and performs an optimization task of finding the best model in a hypothesis space $h : X \to Y$ to approximate some true $f : X \to Y$ which explains the data. Given a new task, knowledge transfer [21] is often applied to accelerate the learning process by distilling and transferring relevant knowledge from previous tasks to the unseen one. Under classical formulations, the learner incorporates prior information in one of several ways:

- Direct transfer of parameters from old hypothesis models to the new task and *fine-tuning* [21] the parameters.
- Learning multiple tasks (online or batched) related to the current task [4, 11].
- Using the prior knowledge (i.e. a knowledge graph) to constrain the hypothesis space by regularization [6].
- Using representations (i.e. *embeddings*) of X and / or Y from previous tasks for new tasks [19, 29].
- Accelerate learning rate and model compression by Distillation as typically seen in Teacher-Student models [12].

In each of these settings, knowledge transfer operates directly within the X, Y and $\mathcal{H}$ spaces to improve generalization of information from old models to the new task.

Recently, *Learning Using Privileged Information* (LUPI) [27] has provided a new paradigm for knowledge transfer. Under LUPI, the learner now interacts with a Teacher who provides *privileged information* (PI), which is available exclusively at training time. From the learner's perspective, the training set is now extended to the tuple $\{(x_i, x_i^*, y_i)\}_{i=1}^m$, and the test set stays the same. Some examples of PI include: 1) Future information that relates X and Y. For example, using future stock prices beyond the prediction window during training. 2) Auxiliary information describing the label space that is available only to a subset of samples. For example, physician notes that accompany diagnostic predictions which is only available after the diagnosis is made.

At a high-level, PI provides some similarity information between training samples from the original feature space, and the Teacher hypothesis serves as additional "explanations" of the hypothesis space [26, 27]. As a result, [26] showed that the LUPI Teacher provides a principled way to improve the *generalization error* of Student learners using agnostic PAC models, providing some theoretical improvements in the number of samples required to generalize to test set data (i.e. improves sampling efficiency).

However, under the current state-of-art LUPI formulations such as [16] and [15], PI is incorporated by means of support vectors and dropout schemes, both of which fail to explore the underlying similarity structure between examples in the PI space X^*. For example the mode distribution and pairwise similarity between points in the X^* space is largely unused. The PI contributes as auxiliary training features and kernel information, but much of the LUPI information is lost at inference time and beyond. A significant question remains: **can privileged information be *retained* for future tasks?**

Ideally, we want the LUPI Teacher to incorporate PI in a way that is specific enough to inform similarity between training samples yet general enough to be retained across future tasks. As a motivating example, consider the medical setting, where electronic health records (EHR) are often sparse, noisy, and full of missing data. Complex tasks such as multi-task learning of many diseases are often difficult to do because of the *long-tail property of diseases* – that is, diseases with very few training samples (i.e. < 100) are very difficult to learn using EHR features alone. On the other hand, medical research on rare diseases are often plenty – large volumes of clinical journals focus on text descriptions of rare diseases in the medical setting. As a result, clinical texts such as discharge notes are *unavailable* at inference time, but when used retrospectively during training can serve as a source of PI that allows for rare diseases to be learned with few examples.

In this work, we propose a LUPI formulation that achieves precisely this. First, we introduce the idea of building a *vocabulary* of PI features by unsupervised learning using external data sources. We then propose a mechanism for learning a joint representation between the PI information and the original set of example features by exploiting their co-occurrence statistics in the training data. We finally learn a *shared* decision function using a contrastive-loss to distinguish between samples drawn from the joint latent space based on their labels for each task. In experiments, we demonstrate the effectiveness of our method in retaining PI obtained from external data sources to support multi-task prediction tasks in the EHR setting against other transfer learning methods. We demonstrate that such an approach both improve the prediction accuracy as well as decrease the samples required to train an accurate model, especially for rare-diseases.

2 RELATED WORKS

Our methods provide several advantages over traditional transfer learning schemes. In the following section, we highlight some of these differences as well as improvements over related LUPI works.

Limitations of incorporating PI as auxiliary targets. When considering retraining PI in the multi-task setting, an intuitive approach may be to incorporate PI as auxiliary prediction targets. For example, suppose a multi-task learning (MTL) model considers a set of $\mathcal{T}_1, ...\mathcal{T}_t$ tasks. PI can potentially be used as the T_{t+1} task and discarded during testing when it is not available. This approach utilizes *inductive learning* and retains PI information by allowing for *parameter sharing* among the multi-task models during training [21, 28].

A major disadvantage of this inductive approach is the constraint on relatedness of tasks. For example, it is hard to know beforehand which subset of tasks will contribute positively to the target, and poor selection of parameter transfer can actually lead to *negative transfer*, leading to poor performance on the target task [21]. By contrast, LUPI uses PI that is by definition *specific to the current task*. However, unlike MTL, LUPI does not use a shared feature space between the PI and original data, i.e. $X \neq X^*$ and cannot directly incorporate the Teacher hypothesis via direct parameter sharing.

Limitations of incorporating PI as prior constraints. In recent years, *relational-knowledge* provides an alternative approach to inductive learning by incorporating domain-specific knowledge in the form of regularized priors for target tasks [6, 9]. In contrast to parameter-sharing, relational-knowledge is agnostic to the collection of source tasks and should apply universally to all learning tasks given the same feature space X. This formulation overcomes the limitation of task similarity in the former case, but it is also very expensive to construct reliable relational-knowledge, such as knowledge graphs. By constraining the hypothesis space of the target task with relational-knowledge, the speed of convergence can be improved. However, inductive learning techniques generally focus on improving either accuracy of prediction or convergence rate rather than sample complexity. The main focus of LUPI, on the other hand, is to provide some guarantees on improving the sample efficiency of the Student learner.

Limitations of incorporating PI with Data Fusion. Another approach of incorporating PI for multi-task learning is to learn a model that uses both the original set of features and PI features for prediction. For example, PI may be seen as an auxiliary source of information, similar to the *Multi-modal Learning* (MML) setting [1, 20]. However, the main drawback of this approach is that the PI features are *unavailable at test time*, leading to poor generalization of the hypothesis model since it is conditioned on both X and X^*.

For example, suppose we have some data fusion model $g : X \times X^* \rightarrow Z$, and a hypothesis function $h : Z \rightarrow Y$. At training time, both the PI and the original features are utilized to train $h(g(x, x^*)) = y$. At test time, however, since only X is available, $g(x, 0)$ may actually map to a completely different set of features in Z, leading to a biased $h(z) = y$. In other words, if X is under-utilized during training, $h(z)$ will likely lead to poor generalization at test time. Unfortunately, since PI is by definition a more task-specific descriptor of Y, this is the most likely case and presents a limitation for data fusion methods for incorporating PI.

Similarly, generative models also offer a possible way of incorporating PI into the modeling process by learning a domain-invariant latent representation or transformation function between X and X^*. For example, recent works such as VRADA and RadialGAN both attempt to learn a *domain-invariant* latent representation between datasets so that examples from multiple sources can be used for a target task [8, 30]. However, these models improve the hypothesis function by means of *data augmentation* (i.e. increasing the samples

available) rather than decreasing the sample complexity required to train an accurate model. This subtle difference becomes important when "big data" is not available for complex data problems, for example modeling rare diseases in healthcare records.

Knowledge Transfer by LUPI: Finally, LUPI provides some performance guarantees with regard to the sample efficiency of the Student learner, so long as the PI and the Teacher model satisfy some conditions [26, 27]. However, the main drawback of current formulations of LUPI is that the PI used is highly specific to the task at hand – if PI information for one task can be carried over for other related tasks, no current mechanisms exist to exploit this advantage. Our work applies elements of both *inductive* and *transductive learning* to alleviate this limitation of LUPI. Although recent works such as [15] has generalized the LUPI framework to deep learning settings, our work extends LUPI to allow for multi-task and transfer learning, enabling the generalization of a PI source to accelerate the sample efficiency of many tasks.

3 PRELIMINARIES

LUPI Preliminaries. Traditionally, LUPI is applied to training data of the form:

$$(x_1, x_1^*, y_1), (x_2, x_2^*, y_2)...(x_m, x_m^*, y_m) \in \mathcal{D}_{\text{train}},$$

where $x_i \in \mathbb{R}^n$ denotes example feature (EF) vectors from the original feature space, and $x_i^* \in \mathbb{R}^{n*}$ denotes privileged information (PI) vectors from the privileged information space. $\mathcal{D}_{train}$ indicates that the PI inputs are only available during training. $y_i \in \{-1, +1\}$ denotes the ground truth labels for inputs (x_i, x_i^*). LUPI then considers two pattern recognition problems:

- Using $\{(x_i, y_i)\}_{i=1}^m$, find rule $h(x_i) = y_i$.
- Using $\{(x_i^*, y_i)\}_{i=1}^m$, find rule $f^*(x_i^*) = y_i$

Suppose that the $f^*(x_i^*) = y$ can produce low generalization error, the LUPI task is to transfer knowledge of rules in the X^* space to improve learning in the X space. In [27] and [26], knowledge transfer between $h(x)$ and $f^*(u_s^*)$ is achieved by by *minimizing the difference between* them over a subset of *frames* in the privileged Kernel space:

$$y = f^*(x^*) = \sum_{i=1}^m \beta_s^* K^*(u_s^*, x^*) + b^* \approx \sum_{i=1}^m \alpha_i K(x_i, x) \quad (1)$$

where $s = 1, ...k$, denotes the number of frames, and $K^*(u_s^*, x^*)$ denotes the *frames* u_s^*, which are vectors in the Kernel space of the privileged information that support the PI. In an ideal situation where the Student learner matches the Teacher perfectly, we will have $\alpha \approx \beta^*$ in the respective Kernel parameters.

Problem Setting and Assumptions. We consider a similar problem setting where we are given $\mathcal{D}_{train} = \{(x_i, x_i^*, y_i)\}_{i=1}^m$. We assume that there exists a set of *vocabulary* with size d that define the PI space. $x_{ij}^* = 1$ if the j^{th} term in the PI vocabulary is contained in the x_i^* sample, $x_{ij}^* = 0$ otherwise. We also denote $x_i^* = \{w_1, ..., w_k\}$ as a decomposition of x_i^* into k individual components such that each $w_j \in \{0, 1\}^d$ is a one-hot vector, corresponding to a non-zero component in x_i^*, and $x_i^* = w_1 + w_2 +, ..., +w_k$. In the following sections, we denote $\{w_j\}_{j=1}^d$ as the set of "words" that compose the PI vocabulary. Thus, x_i^* gives the *co-occurrence* label of each word $w_j \in \{w_j\}_{j=1}^d$ with respect to the sample x_i.

We make no assumptions on the example features (X) with regards to data type. However, for simplicity, let us take $x_i = \{x_i^{(t)}\}_{t=1}^T, x_i^{(t)} \in \mathbb{R}^n$ to be the time-series data type, where each sample x_i has T time-steps, and each $x_i^{(t)}$ is a real-valued vector of n-dimensions. Finally, we define the multi-task learning objective as learning $y_i = \{0, 1\}^C$ to be a set of C binary classification tasks.

4 PROPOSED METHOD

At a high-level, the main intuition behind our proposed method is to decompose the LUPI process into three parts:

(1) Build a dictionary of PI features and learn a distributed representation [19] over the PI vocabulary.
(2) Find a joint representation space (Φ) between the PI and example features.
(3) Jointly learn the decision functions $h^* : \Phi \rightarrow Y$ by feature-matching in the joint representation space.

The first process uses unsupervised learning to embed the PI vocabulary into a vector space. The second process allows for some of the privileged information to be *retained* at inference time, despite not having direct access to the PI vectors. The third process allows for PI information for one task to be transferred for other $C - 1$ tasks in the label space. In the following subsections, we will examine how to achieve (1) – (3) in detail. We also provide analysis of how (3) can maintain the favorable LUPI sample efficiency.

Building the PI vocabulary: First, we can define $g^*(w_j; \theta_{g*})$ as an embedding function that maps $g^* : X^* \rightarrow \Phi$. Note that x_i^* consists of individual words, $\{w_1, ..., w_k\}$. So $g^*(w_j; \theta_{g*})$ embeds each individual word in the PI vocabulary rather than the PI samples (i.e., x_i^*). The rationale behind $g^*(.)$ is to encode each word in the PI vocabulary into a vector space so vector operations can be applied to the PI. We specifically consider embedding function of the form,

$$g^*(w_j; \theta_{g*}) = w_j^T \theta_{g*}. \quad \text{(PI Embedding)}$$

Since each $w_j \in \{0, 1\}^d$ has $w_{jk} = 1$ only when $j = k$, the w_j vector simply selects the j^{th} column in θ_{g*}. We restrict $\theta_{g*} \in \mathbb{R}^{d \times k}$ so that θ_{g*} produces a lower-dimensional representation of each word in the PI vocabulary. For this first step, we do not restrict the PI to come from the original dataset $\{(x_i, x_i^*, y_i)\}$. In fact, we can learn the *embedding* θ_{g*} for our PI using any data source by applying the following word-model:

$$Score(w_1, w_2) = \begin{cases} 1 & \text{if } w_1, w_2 \in x_i^* \\ 0 & \text{otherwise} \end{cases} \quad \text{(Co-occurence)}$$

$$g(w_1, w_2) = \sigma\{(\theta_{g*} w_1)^T (\theta_{g*} w_2)\} \quad \text{(Word Model)}$$

$$\mathcal{L}_{emb} = BCE(Score(w_1, w_2), g(w_1, w_2)). \quad (2)$$

Here, $\sigma(.)$ denotes the Sigmoid activation function, and $BCE(.)$ denotes the binary cross-entropy loss: $-\sum_i [a_i (log b_i) + (1 - a_i) log(1 - b_i)]$. The cross-term $(\theta_{g*} w_1)^T (\theta_{g*} w_2)$ gives the *similarity* between w_1 and w_2 in the embedding space, which is then scored against $Score(w_1, w_2)$ based on whether w_1, w_2 both appear in x_i^*. We note that the embedding loss $\mathcal{L}_{emb}$ is trained separately from the rest of the LUPI model since it is not specific to the dataset.

Learning the Joint Representation: Next, let us define $g(x_i;\theta_g)$ to be the embedding function that maps $g: X \rightarrow \Phi$. In this work, we consider the following for $g(.)$ and $g^*(.)$:

- *Distributed representation* of PI vocabulary, which captures its underlying manifold structure and is obtained by unsupervised learning [19].
- Encoding of X into a fixed length vector by deep embedding methods such as [5, 17].

For time-series X, we take the embedding functions to be a recurrent encoder neural network:

$$g(x_i;\theta_g) = RNN(x_i;\theta_g) \quad \text{(EF Embedding)}$$

The motivation behind using the embedding functions $g(x)$ and $g^*(w)$ is to extend the idea of *Student and Teacher kernels*, which allow for the privileged information to provide information about similarity between training samples in the feature space [26]. Using neural encoding for $g(x;\theta_g)$ allows such feature spaces to be represented by a fixed-length vector without losing the underlying spatio-temporal information.

To find commonality between example features and PI, we introduce a *matching function* (μ) that maps each $(g(x_i), g^*(w_j))$ pair onto the interval $[0,1]$, i.e., $\mu: \Phi \times \Phi \rightarrow [0,1]^d$:

$$\mu(g(x_i), g^*(w_j); A) = \frac{\exp(\max\{0, [g(x_i); g^*(w_j)]^T A\})}{\sum_{p=1}^{d} \exp(\max\{0, [g(x_i); g^*(w_p)]^T A\})}. \quad (3)$$

Here, $[g(x_i); g^*(w_j)] \in \mathbb{R}^{2k}$ denotes the concatenation of the $g(x_i)$ and $g^*(w_j)$ embeddings in the joint latent space. The parameter matrix A projects the pairs $(g(x_i), g^*(w_j))$ onto $\mathbb{R}$, and the softmax activation normalizes the pairwise scores against other word-pairs in the PI vocabulary. Thus, for each sample mapped from the feature space X, the matching function μ produces a set of corresponding weights over all of the words in the PI vocabulary.

We make the key observation that for each word w_j, the output weight of the matching function should correspond to the j^{th} component of the x_i^* sample in the training data. That is, $\mu(g(x_i), g^*(w_j); A) \approx x_{ij}^*$. Using this fact, we can learn the matching function by minimizing over the following objective:

$$\mathcal{L}_\phi(\theta_g, \theta_{g^*}, A) = -\frac{1}{md}\sum_{i=1}^{m}\sum_{j=1}^{d}[x_{ij}^* \log \mu_i(g(x_i), g^*(w_j); A) + (1 - x_{ij}^*)\log(1 - \mu_i(g(x_i), g^*(w_j); A))]. \quad (4)$$

Each component of the PI vector x_i^* can thus be interpreted as providing an indicator label for likelihood of the (x_i, w_j) pair to occur together. The similarity control mechanism highlighted in Eqn. 4 differs from the Kernel-matching mechanism mentioned previously in Eqn. 1. The limitation of Eqn. 1 is that two sets of Kernel weights need to be learned simultaneously: α for $K(x_i, x)$ and β for $K^*(x_i^*, x^*)$. By contrast, our joint representation for x_i and x_i^* encourages a single hypothesis model to be used to map $h^*: \Phi \rightarrow Y$. Since matrix A captures the $g(x_i)$ and $g^*(w_j^*)$ interactions, it preserves the PI in the space of Φ and allows relevant PI to be retrieved at test time.

Finally, we obtain the *augmented representation* of x_i as a weighted combination of $\mathcal{G}^* = \{g(w_j)\}_{j=1}^d$ and $g(x_i)$:

$$\phi(x_i) = g(x_i) + \sum_{j=1}^{d} \mu_i(g(x_i), g^*(w_j); A) \cdot g^*(w_j) \quad (5)$$

One can think of $\mathcal{G}^*$ as the set of *basis vectors* supporting the PI space (similar to *frames* for the PI Kernel [26]). The augmented representation $\phi(x_i)$ contains both information from the original x_i as well as relevant information retrieved from $\mathcal{G}^*$. Note that since the PI vectors $x_j^* \in X^*$ are not directly used at testing time, each sample $x_i \in \mathcal{D}_{test}$ is mapped into Φ using $g(.)$, and the trained $\mu(.)$ selects the corresponding bases in $\mathcal{G}^*$ to construct ϕ_i.

This is quite different from *representation fusion* methods [22, 29], which only try to learn a shared representation space for input modalities $X_1, ..., X_k$, without a matching function to control the contribution of each modality to the hypothesis. For example, we can take X_1 to be the original feature space and X_2 to be the privileged information. At test time, when X_2 is unavailable, X_1 inputs with masked X_2 components may be projected into a completely different location in the shared representation space than if the X_2 information were available. Furthermore, *model fusion* methods [22] may also under-utilize the original feature space during training, as the PI contain more information related to the target task.

Coupling Decision Functions with Feature Matching: LUPI typically considers 2 hypothesis functions: the Student hypothesis $h: X \rightarrow Y$, and Teacher hypothesis $f^*: X^* \rightarrow Y$. Since we have already addressed the problem of finding a common "frame of reference" between the original feature space and the PI space by the matching function μ, the main focus for this portion of our method has to do with finding an efficient f^* that relates the privileged information to the labels. Fortunately, we can directly approximate $f^*: X^* \rightarrow Y$ by a function $h^*: \Phi \rightarrow Y$ that maps samples from the joint representation space to the label space. This is because Φ is constructed by embedding function $g^*(.)$ on X^* and is an approximation of the Kernel space for the privileged information.

In the case that the target task is *classification*, we can formulate h^* as a *feature matching* problem between samples from Φ and Y. Specifically, we can use contrastive loss [10] to find an invariant representation $h^*: \Phi \rightarrow Y$ and vice versa, by minimizing the distance between similar samples drawn from the joint embedding space based on signals from the label space:

$$L(W, y_i, x_i^+, x_i^-) = (1 - y_i)\mathcal{M}_W(x_i^+, x_i^-) + (y_i)(\max\{0, C - \mathcal{M}_W(x_i^+, x_i^-)\}) \quad \text{(Contrastive)}$$

$$\mathcal{L}(W) = \frac{1}{S}\sum_{i=1}^{S} L(W, (y_i, x_i^+, x_i^-)^i) \quad (6)$$

where $\mathcal{M}_W(x^+, x^-) = ||\phi(x^+; W) - \phi(x^-; W)||_2$ refers to a parameterized distance metric with respect to projections $\phi(x^+)$ and $\phi(x^-)$, and C is the *slack variable* which defines the margin of separation between them. $\phi(.)$ is simply the projection function from eqn. 5, which is parameterized by $W = [\theta_g, \theta_{g^*}, A]$ from eqn. 4. Intuitively, $\mathcal{M}_W$ finds the distance between *augmented projections* of x_i^+ and x_i^-, i.e., $\phi(x_i^+)$ and $\phi(x_i^-)$, which are compared by their

labels y_i. Given a training pair (x_i, x_i^*, y_i), a set of k *similarity samples* $S_i = \{(x_i^+, x_i^-, y_i)^j\}_{j=1}^k$ is constructed around the (x_i, y_i) pair, whereby x_i^+ denotes samples with the same label as $y_i = y^+$, and x_i^- denotes samples with a different label than $y_i \neq y^-$. Thus, x^+ and x^- denote *positive samples* (similar) and *negative samples* (dissimilar), respectively.

A variety of negative sampling techniques can be used to obtain the set S [3, 10, 14]. In practice, we found picking $5-10$ negative samples that are close to $\phi(x_i)$ and $5-10$ samples that are far from $\phi(x_i)$ to be sufficient in creating S for each training triplet (x_i, x_i^*, y_i). We refer the reader to [10] and [3] for more information about the contrastive loss and the construction of S.

Finally, we combine the two portions of our learning task (i.e. representation learning and joint hypothesis) into the optimization task:

$$\min_{\Theta} \mathcal{L}(W) + \lambda \mathcal{L}_\phi(\theta_g, \theta_{g^*}, A) + \Omega(W), \tag{7}$$

where $W = [\theta_g, \theta_{g^*}, A]$ is the total set of parameters for the learning task. λ is the hyperparameter which controls the trade-off between the contrastive loss to learn h^* and the representation loss in eqn. 4. $\Omega(.)$ is the regularization term used to constrain the hypothesis space of the joint model.

5 ANALYSIS OF SAMPLING EFFICIENCY

Results from Existing Agnostic Models

For an agnostic hypothesis model, such as non-linearly separable SVM, the generalization error bound holds with $1-\delta$ probability:

$$R(h) \leq R_{emp}(h) + O^*\left(\sqrt{\frac{\Delta_H \log(m/\Delta_H) - \log \delta}{m}}\right), \quad \text{(VC-bound)}$$

where $|R(h) - R_{emp}(h)| = \epsilon \in [0, 1]$ is the generalization error represented by the difference between expected and empirical training risks. Δ_H is the VC-dimension of the given SVM model class, m is the sample size, and $\delta \in (0, 1)$, whereas under the SVM+ formulation in [26], the generalization error is given by:

$$R(h) \leq R_{emp}(f^*) + O^*\left(\frac{(\Delta_H + \Delta_{F^*})\log(\frac{m}{\Delta_H + \Delta_{F^*}}) - \log \delta}{m}\right), \tag{SVM+}$$

where $R_{emp}(f^*)$ denotes the error rate of the Teacher's hypothesis $f^* : X^* \to Y$, and Δ_{F^*} denotes the VC-dimension of the Teacher model. In the original SVM, the model needs to re-estimate m slack variables for each training sample, in addition to the n parameters in w. At a high-level, the hypothesis function f^* of the LUPI Teacher serves as a *slack function* which approximates these slack variables for each x_i, eliminating the need for the Student to estimate them during training [26]. The number of estimations in the latter case reduces to $O(m+n)$, rather than $O(mn)$. As a result, the sampling efficiency improves from $m \leq O(\frac{\Delta_H + \log(1/\delta)}{\epsilon^2})$ to $m \leq O(\frac{\Delta_H \log \Delta_H + \log(1/\delta)}{\epsilon})$ in the number of samples required to achieve the same generalization error ϵ.

Complexity of Proposed LUPI Method

In this section, we examine the sample complexity of our proposed LUPI method. For simplicity, let us consider the classification setting where we are given a hypothesis class $\mathcal{H}$ of finite VC-dimensions which define a set of functions mapping $\mathcal{X}$ to a label set $\{0, 1\}$, and let the $0-1$ loss function define the empirical risk. Let $\Delta_{\mathcal{H}} = d < \infty$. By **the fundamental theorem of PAC learning** (Thm. 6.7 in [23]), there exist $C_1, C_2 \in \mathbb{R}$ such that:

(1) $\mathcal{H}$ is agnostic PAC learnable and has the *uniform convergence property* with sample complexity

$$C_1 \frac{d + \log(1/\delta)}{\epsilon^2} \leq m(\epsilon, \delta) \leq C_2 \frac{d + \log(1/\delta)}{\epsilon^2} \quad \text{(Agnostic)}$$

(2) There exists a Realizable subset $\mathcal{H}_r$ such that the sample complexity is defined by

$$C_1 \frac{d + \log(1/\delta)}{\epsilon} \leq m(\epsilon, \delta) \leq C_2 \frac{d \log(1/\epsilon) + \log(1/\delta)}{\epsilon}. \quad \text{(Realizable)}$$

The main difference between *agnostic* and *realizable* PAC models lies in whether the classifier can completely classify a training set $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^m$. That is, the training error $R_{emp}(h) = \frac{1}{m}\sum_{i=1}^m L(h(x_i), y_i) = 0$ for the particular hypothesis class under some empirical risk minimization (ERM) algorithm. On the other hand, when we have $y_i \neq h(x_i)$ for some training data, there exist some examples for which the current hypothesis class can not successfully separate (i.e. cannot realize an accurate ERM hypothesis with 0 empirical risk), suggesting that the optimal solution is either not contained in the span of the given hypothesis class, or the ERM algorithm cannot converge to the optimal solution in the hypothesis space. Fortunately,[25] introduces some conditions for which a sample complexity between $O(\frac{\log(1/\delta)}{\epsilon})$ and $O(\frac{\log(1/\delta)}{\epsilon^2})$ is possible for some classes of models. Specifically, Tsybakov showed that $m(\epsilon, \delta) \leq C\frac{d\log(1/\delta) + \log(1/\delta)}{\epsilon^n}, 1 < n < 2$ exists under two general scenarios:

- When there is zero training error (ERM realizable).
- When the classification *margin* between the given hypothesis class and another realizable hypothesis class is bounded.

LUPI qualifies the second Tsybakov condition by leveraging a realizable Teacher that provides a mechanism to bound the margins of the Student ERM classifier. Specifically, [27] considers the case where the Student hypothesis class $\mathcal{H}$ is non-realizable and the Teacher hypothesis $f^* \in \mathcal{F}^*$ is a realizable classifier that approximates an *Oracle* classifier with zero-training error. [27] showed that using kernel alignment between the Teacher and the Student, the latter can satisfy the Tsybakov conditions, leading to a sample complexity that is comparable to the realizable case. [26] further showed that, under some assumptions on $\mathcal{H}^*$, one can achieve $O(\frac{\log(1/\delta)}{\epsilon}) \leq m(\epsilon, \delta) \leq O(\frac{\log(1/\delta)}{\epsilon^2})$ so long as the Teacher classifier $\mathcal{H}^*$ has lower VC-dimension and training error than $\mathcal{H}$.

In our model, we provide the margin bounding mechanism by contrastive loss in Eqn. Contrastive. Under the condition that the Teacher model $f^*(x^*)$ has a lower VC-dimension and $y \cdot h(x) > c_0 C - \mathcal{M}(x^+, x^-)$, the $C - \mathcal{M}(x^+, x^-)$ term from the Contrastive eqn. serves as the approximate slack margin of the oracle classifier. To see this, let $p \in \mathbb{R}$ and $q \in \mathbb{R}$ be random values. Suppose that $p < 0$, then either $q < 0$ or $p - q < 0$ is true. We then also have $P(p < 0) \leq P(q < 0) + P(p - q < 0)$. If we take $p = y \cdot h(x)$ and $q = y \cdot f^*(x^*)$, then with $1-\delta$ probability, we can express the error

bounds of the Student and Teacher models as follows,

$$P(y \cdot h(x) < 0) \leq P(y \cdot f^*(x^*) < 0) + P(y \cdot h(x) < y \cdot f^*(x^*)).$$

Under the contrastive loss in eqn. (Contrastive), we can re-formulate the above as:

$$\begin{aligned} P(y \cdot h(x) < 0) &= P(y \cdot f^*(x^*) < 0) + P(y \cdot h(x) < C - \mathcal{M}(x^+, x^-)) \\ &\leq P(y \cdot f^*(x^*) < 0) + O\left(\frac{\Delta_H + \Delta_{f^*} - \ln \delta}{m}\right) \\ \implies m &\leq O\left(\frac{(\Delta_H + \Delta_{f^*} \log(1/\delta) + \log(1/\delta)}{\epsilon^n}\right), \end{aligned}$$

where $1 < n < 2$ if $P(y \cdot f^*(x^*) < 0)$ is satisfied, i.e. if the Teacher model is realizable. We note that although our proposed LUPI model is PAC learnable, i.e. has the uniform convergence property, we cannot bound the computational complexity of learning. Specifically, if we allow the embedding components $g(.)$ and $g^*(.)$ to be non-convex functions, then finding the global optimum for $\phi(x)$ becomes an NP-hard problem. In other words, the above analysis only examine the sample complexity bounds, but it does not provide insight into the computational runtime of learning, or the *actualization* of the uniform convergence property.

6 EXPERIMENTS

We empirically assess the effectiveness of our LUPI formulation for improving sample efficiency and generalization performance in a multi-task setting. First, we consider performance accuracy on numerous *diagnostic prediction tasks*, which are individually binary classification problems. This allows us to evaluate the ability of our LUPI formulation to actually transfer the privileged information to improve the learning efficiency in a multi-task setting. We benchmark the learning accuracy of our method against the performance of various transfer learning baselines. We also perform an ablation study on the privilege information components of our model to ascertain its necessity. Finally, we perform prediction tasks on classes with only sparse examples – as defined by ≤ 100 training samples, and compare the sample efficiency of our model against select models from other transfer learning paradigms.

For PI, we consider *physician notes* in the form of discharge summaries, linking standard medical terminologies (i.e., UMLS codes [2]) with diagnostic findings in the EHR. *UMLS codes* are a set of standardized medical concepts used by clinicians to describe physical findings of diseases and are used widely in both the EHR as well as medical research [2]. [26] alluded at the idea that medical datasets also contain vast amounts of privileged information in the *physician notes*, which serve to explain the qualities of diseases that can greatly aid decision rules. For experiments, we consider the following set of data for example features, PI, and labels:

- Example Features X: continuous time-series data (i.e. lab values, blood tests, imaging) and discrete static variables (i.e. demographics information) that describe a patient.
- Privileged Information X^*: physician notes containing descriptions in natural language and medical terms (UMLS concepts [2]) that summarize a particular visit for a patient.
- Target Task Y: prediction tasks of interest, such as mortality (binary classification), disease prediction (multi-task and transfer learning), ... etc.

Datasets and Setup: Table 1 provides a brief summary of data sources for our experiments. For each data source, we extract unique data modalities available in the dataset. MIMIC-III (Medical Information Mart for Intensive Care) is a publicly available benchmark dataset for predictive modeling and clinical decision support in the intensive care unit (ICU) setting [13]. It should be noted that MIMIC-III and STRIDE datasets are EHR datasets, although STRIDE is comprised of clinical notes (PI vocabulary) obtained from multiple EHR datasets over 19 years of data collection. *Documents* in table 1 refer to literature sources, including medical claims [7] and research articles [9] that heavily utilize UMLS codes. We refer to the clinical notes from EHR as the PI source, which we decomposed into lists of UMLS codes. For example, a clinician's note may contain a text description of *pneumonia* which may utilize several UMLS codes such as (*Lower Lobe Consolidation, Staph Aureus, Productive Cough*) as keywords.

Table 1: Summary of datasets used in this study.

Database	No. Patients	UMLS	ICD-9s	Temporal
MIMIC-III	22, 043	928	148	40
STRIDE	4M	14, 256	None	None
Documents	1.2M	None	11, 245	None

MIMIC-III provides a rich source of temporal data, ranging from laboratory tests, vital signs and respiratory parameters, all of which provide hourly resolution of descriptive features. For example features, we use 40 physiologic features, including vital signs (i.e. `heart rate, blood pressure, oxygen saturation, temperature`), blood tests (i.e. `WBC count, platelets, INR`) and respiratory parameters: (i.e. `PaO2/FiO2 ratio, PEEP`). These temporal features are the source of example features (EF) for our experiments, i.e. $X = \{x_i\}_{i=1}^m, x_i = \{x_i^t\}_{t=1}^T$. Preprocessing of these features include binning the time-series by hourly average of each feature and standardized feature values across all adult patients.

Physician notes (the source of PI) in MIMIC-III exist in the form of *discharge notes*, which are physician documentation of key findings relating to the patient's hospital visit. We can represent the PI as $X^* = \{x_i^*\}_{i=1}^m$, where each $x_i^* \in \{0,1\}^d$ represents a discharge note for the i-th patient, in the form of a d dimensional one-hot vector. Here, d is the total number of UMLS codes that are found in all of our data sources (MIMIC-III, STRIDES, and Documents). One can think of the UMLS codes as a set of basis features for the PI vectors. The rationale of using physician notes as PI is that they are *only available at the end of the hospital stay* and contain copious amount of valuable information regarding a wide array of clinical decision support tasks, such as physical findings, periodic nurse observations, medical or surgical complications, and indicators for mortality risk. During training, we can incorporate these notes into the learning regime, but they become unavailable at inference time.

For labels, MIMIC-III provides a wide range of potential tasks. We focus on the prediction of ICD-9 diagnostic codes, where are a set of diagnosis labels given to patients that identifies their disease states. Each patient has a set codes that can be described by a label vector $y_i \in \{0,1\}^C$, where C denotes the total number of disease classes considered. ICD-9 prediction is in fact a difficult *multi-label classification* problem among other clinical benchmark tasks due

the fact that the distribution of diseases often contain *long tails* [24]. In the typical case, a few diseases dominate in high frequency while most diagnostic codes appear only a few times among all patients. As a result, training samples are sparse for most diseases, leading to poor prediction beyond the most frequent cases. Our experimental task is to leverage information from PI under a *multi-task learning setting* to improve the learning efficiency for a large set of ICD-9 codes, especially ones in the tail distribution (i.e. occurring with few samples in the dataset). We consider the diagnoses appearing in at least 1% of admissions, leaving $C = 148$ ICD-9 group codes to formulate our multi-task prediction as C classification tasks. We consider UMLS terms appearing at least 50 times in discharge notes, leaving $d = 928$ UMLS terms to construct the PI vocabulary.

Initial Baselines: We establish some baseline performance of various hypothesis models for our prediction tasks under 3 settings:

- Using only example features (EF only) to predict ICD-9 labels.
- Using only PI information to predict ICD-9 labels.
- Using both EF and PI information to predict ICD-9 labels.

For each setting, a diverse set of hypothesis classes are used, including a standard recurrent neural network (RNN) and feed forward perceptrons (MLP). The rationale behind these baselines is to determine whether the PI indeed offers more information than the original feature-set based on noisy timer-series data. Ideally, the Teacher hypothesis class F^* should obtain lower empirical risk while using lower model complexity (lower VC-dimension) compared to the Student hypothesis class $\mathcal{H}$ without LUPI. Otherwise, the Student learner will not improve its sample complexity by LUPI, and any improvements in prediction accuracy will likely result from a variance-reduction mechanism (i.e. ensemble) rather than the LUPI mechanism. As a sanity check, we also included a comprehensive Teacher model using EF + PI features, which should provide the best performance. We note, however, that because the Teacher models use PI at test time, they are used to assess the quality of PI rather than benchmark Student performance. In practice, PI is unavailable at test time, so the Teacher models cannot be used for inference in a real-world setting. Table 2 summarizes the performance of these baselines on held-out test set data.

Since ICD-9 predictions involve a large number of classes, we take both micro-averaged and macro-averaged AUC as evaluation metrics. Macro-averaged AUC takes *per-class average* of AUC scores, while micro-averaged AUC considers a single AUC score based on a roll-out of label classes for each test set sample. We also include micro-averaged F1-score and micro-averaged area under PRC to quantify the trade-off between precision and recall.

- **RNN Student** denotes the Student learner using the RNN model class conditioned exclusively on EF, using the LSTM architecture as mentioned in [18].
- **MLP Student** denotes a feed-forward network conditioned on the final time-step of EF.
- **MLP Teacher** denotes the Teacher feed-forward network conditioned on PI only. Specifically, we use a weighted sum of the PI embeddings for each $x_i^* = \{w_1, ..., w_k\}$:

$$\phi(x_i^*) = \frac{1}{k}\sum_{j=1}^{k} w_j^T \theta_{g*}$$

Table 2: Comparison of performance across baseline models. Micro-averaged AUC and Macro-averaged AUC are denoted as Mi-AUC and Ma-AUC, respectively. Micro-averaged F1-score and AUC of precision-recall curve are denoted as Mi-F1 and AUPRC, respectively.

Model	Ma-AUC	Mi-AUC	Mi-F1	AUPRC
RNN Student	0.735	0.783	0.299	0.260
MLP Student	0.715	0.756	0.235	0.211
MLP Teacher*	0.824	0.868	0.446	0.432
Oracle Teacher*	**0.845**	**0.882**	**0.497**	**0.510**

(*) *denotes Teacher models using PI.*

  which maps each PI vector x_i^* into a lower-dimensional representation space, and θ_{g*} denotes the look-up matrix of embeddings obtained in the first step of our LUPI algorithm.
- **Oracle Teacher** denotes the Teacher model which uses both EF and PI for prediction.

$$h(x_i) = RNN(x_i)$$
$$f_c^*(x_i, x_i^*) = \sigma(W_{hc}h(x_i) + W_{gc}g(x_i^*) + b_c)$$

  EF inputs are encoded into fixed-length vectors by a set of RNN layers and the PI features are embedded into lower-dimensional space by $g(x^*)$ described previously. Since there are C tasks (i.e., C outputs), a classifier layer is used to predict the $0-1$ label for each ICD-9 code.

Here, we note that the embedding matrix θ_{g*} for learning the lower dimensional representation of PI is obtained by the embedding mechanism highlighted in the PI Embedding equations. Taking the set of UMLS concept codes as the PI vocabulary, we leverage the corpus available in STRIDE and Documents datasets to learn the θ_{g*}, conditioned on the UMLS codes. For example, given a medical document consisting of a set of n relevant UMLS codes $\vec{v} = \{w_1, w_2, ...w_n\}$, we can train the θ_{g*} for the UMLS codes by Eqn. 2, with the modified scoring function:

$$Score(w_1, w_2) = \begin{cases} 1 & \text{if } w_1, w_2 \subset \vec{v} \\ 0 & \text{otherwise} \end{cases} \qquad \text{(Co-occurence)}$$

Note that each UMLS concept is represented by $w_i \in \{0, 1\}^d$, where $w_{ij} = 1$ for the index corresponding to the UMLS code. Thus, $\theta_{g*}[:, i]$ gives the the *distributed representation* of i^{th} UMLS concept. We train the embedding matrix θ_{g*} over STRIDE and Documents before applying θ_{g*} on the MIMIC-III dataset for LUPI.

We see from table 2 that the PI provides strong signals for ICD-9 prediction. Large differences exist between the Student baselines and the Teacher models across all performance metrics, suggesting that the PI provides more information about the label space compared to the original time-series features. Again, we emphasize here that discharge notes (PI) are generated only *after* the diagnostic predictions have been made by clinicians, and thus the Teacher models are actually not available at inference time. Interestingly, we also see that the Oracle teacher with combined features provided additional performance boost compared to using PI exclusively as features. This suggests that the temporal features provide some complementary information not contained in the PI.

Transfer Learning: Next, we benchmark performance for several existing transfer learning paradigms for incorporating PI with the Student model: transductive learning, inductive learning, and model distillation. Under the transductive framework, we treat the PI as *auxiliary targets*, much like *target replication* in [18]. We train a joint hypothesis model $h : X \rightarrow X^* \times Y$ to map from the original EF space to the joint PI and label space. By contrast, we incorporate X^* as an auxiliary input for the inductive framework. We use *data fusion* to learn a joint representation $g : X \times X^* \rightarrow Z$ before learning a hypothesis function to predict $h : Z \rightarrow Y$. For model distillation, a Teacher network predicts a set of soft-labels over the PI information, which the Student model uses as auxiliary input for the final prediction model $h : X \times X^* \rightarrow Y$. Details of the setup is explained below:

a) Multi-task learning. MTL is the representative transductive learning technique. There is only one source domain $\mathcal{D} = \{X, P(X)\}$ and two target tasks: $\mathcal{T}_S = \{X^*, G\}$ and $\mathcal{T}_T = \{Y, \mathcal{H}\}$. The MTL model learns a joint model:

$$g_k(x_i) = MLP(RNN(x_i; W_{rep}); W_k) \quad \text{(Shared Rep.)}$$

$$h_c(x_i) = MLP(RNN(x_i; W_{rep}); W_c) \quad \text{(Individual Hyp.)}$$

$$\mathcal{L}_{MTL} = \frac{1}{mC} \sum_{i=1}^{m} \sum_{c=1}^{C} L_Y(h_c(x_i), y_{ic}) + \frac{\lambda}{dm} \sum_{i=1}^{m} \sum_{j=1}^{d} L_{X^*}(g_j(x_i), x^*_{ij}) \quad \text{(MTL Obj.)}$$

where W_{rep} is the shared weights for the representation model $RNN(x_i; W_{rep})$, W_k and W_c are task-specific weights for target hypothesis models. The MTL loss is composed of two parts: (1) a loss component over the joint label space $\mathbb{E}[L_Y(h(X), Y)]$, and (2) a loss term over the joint PI space $\mathbb{E}[L_{X^*}(g(X), X^*)]$. λ is a hyperparameter which controls the trade-off between the multiple objectives during learning. $L(.)$ denotes some evaluation criterion to approximate the $0-1$ loss, for example the binary cross-entropy (BCE) or mean squared error (MSE). We used BCE in the proceeding experiments.

b) Data Fusion. For inductive learning, we used a variant of the Siamese Network [14] to achieve *data fusion* between EF and PI. We use two parallel networks, $g : X \rightarrow Z$ and $g^* : X^* \rightarrow Z$ and minimize the distance between $g(x)$ and $g^*(x^*)$ using the BCE loss. We then learn a hypothesis function $h : Z \rightarrow Y$.

$$g(x_i) = RNN(x_i; W_x) \quad \text{(EF Embedding)}$$

$$g^*(x^*_i) = MLP(x^*_i; W_{x^*}) \quad \text{(PI Embedding)}$$

$$h(x_i, x^*_i) = \sigma(W_g g(x_i) + W_{g^*} g^*(x^*_i) + b_g) \quad \text{(Joint Hyp.)}$$

$$\mathcal{L}_Z(W_x, W_{x^*}) = \sum_{i=1}^{m} BCE(g(x_i), g^*(x^*_i)) \quad \text{(Fusion Loss)}$$

$$\mathcal{L}_Y(W_g, W_{g^*}) = \sum_{i=1}^{m} BCE(h(x_i, x^*_i), y_i) \quad \text{(Task Loss)}$$

$\mathcal{L}_Z$ and $\mathcal{L}_Y$ are trained iteratively using alternating stochastic gradient descent (SGD). At test time, a masking vector $x^*_{test} = \{0\}^d$ is used to represent PI, as it is unavailable for inference. Thus, $h(x_i, x^*_{test}) = y_i$ is used for evaluation.

c) Distillation. Our distillation also contains two parts: a Teacher network trained to generate soft-labels for PI, and Student network conditioned on the EF and PI soft-labels to predict the ICD-9 targets.

$$g(x_i) = RNN(x_i; W_T) \quad \text{(Teacher)}$$

$$h_c(x_i) = MLP([g(x_i); RNN(x_i; W_{RNN})]; W_c) \quad \text{(Student)}$$

$$\mathcal{L}_{TS} = \sum_{i=1}^{m} \sum_{c=1}^{C} BCE(h_c(x_i), y_i) \quad \text{(Distillation Loss)}$$

Here, $[g(x_i); RNN(x_i; W_{RNN})]$ denotes concatenation of the PI soft-labels and the last hidden state of the $RNN(x_i)$. The joint T-S loss connects the Teacher and Student loss together, allowing the two networks to be trained end-to-end.

Table 3: Comparison of performance across transfer learning models. Micro-averaged AUC and Macro-averaged AUC are denoted as Mi-AUC and Ma-AUC, respectively. Micro-averaged F1-score and AUC of precision-recall curve are denoted as Mi-F1 and AUPRC, respectively.

Model	Ma-AUC	Mi-AUC	Mi-F1	AUPRC
MTL	0.783	0.836	0.384	0.336
Distillation	0.738	0.793	0.289	0.245
Data Fusion	0.779	0.811	0.374	0.328
Ours	**0.838**	**0.845**	**0.397**	**0.344**

Table 3 summarizes the AUC and retrieval scores for our LUPI model against transfer learning baselines. We see that our LUPI formulation outperformed other transfer learning baselines in all major performance metrics. Vanilla *Data Fusion* and *MTL* networks produced comparable performances, and *Distillation* did not have significant improvement over the baseline Student models that did not use PI. For *Data Fusion*, it is likely that since the PI contains a lot more information than the original EF, the decision function of the hypothesis model $h(x, x^*)$ relied heavily on access to PI.

Since PI is masked during testing due to unavailability, $h(x, \{0\}^d)$ likely resulted in poor generalization. Comparable results can be seen in MTL. The drop in performance is most likely due to *negative transfer* [21] due to the wide range of tasks (ICD-9 codes) that contribute uniformly to the multi-objective learning process. Unlike our LUPI, the PI is not used to inform similarity between training samples from different tasks, which do not share the same support (i.e. different diagnosis may come from very different underlying distributions). Thus, in both *Data Fusion* and *MTL* cases, the PI is incorporated in a less efficient way than our proposed model.

Interestingly, we see that the *MLP Teacher* and *Oracle Teacher* models in Table 2) still provided better AUC, F1 and AUPRC performances over the all transfer learning models, including our LUPI model. This result suggests that the PI is more informative for diagnostic tasks compared to the original EF, which is what enables the LUPI method to be effective. One possible explanation of the predictive power of the PI is that the embeddings of the UMLS terms, which comprise the PI vocabulary, are learned based on their co-occurrence with disease codes in public literature.

Performance with Sparse Examples: In addition to broad coverage of tasks, we evaluate the *sample efficiency* of our proposed model against transfer learning baselines by considering the more rare diseases with very few training samples. This is actually quite typical in the EHR setting, where diagnosis labels often have very long tail distributions. We restrict our predictions to diagnostic

Table 4: Performance of various models for 30 ICD-9 codes appearing less than 100 times in the dataset.

Model	Ma-AUC	Mi-AUC	Mi-F1	AUPRC
RNN Student	0.628	0.639	0.104	0.096
MLP Teacher	0.821	0.833	0.212	0.196
Oracle Teacher	0.801	0.805	0.256	0.146
MTL	0.717	0.724	0.150	0.122
Distillation	0.729	0.738	0.158	0.149
Data Fusion	0.821	0.826	0.307	0.237
Ours	**0.834**	**0.835**	**0.381**	**0.330**

codes appearing less than 100 times in the training and test sets and examine the generalization of various modeling schemes.

In table 4, we see that performance decreased drastically for non-transfer learning models such as the RNN Student and the Teacher models (both MLP and Oracle). Transfer learning schemes such as *MTL* and *Distillation* also decreased greatly in F1-score and AUPRC. Interestingly, *Data Fusion* method was able to outperform other transfer learning baselines in F1-score and AUPRC, suggesting that learning a domain-invariant representation between the original features and the PI provided a key improvement for sample efficiency. However, our LUPI model achieved the best performance among all the models for long-tail tasks. In fact, its performance across stayed relatively consistent among this subset of tasks compared to its performance on the original set of common diagnoses.

7 DISCUSSION AND CONCLUSION

In this work, we presented a novel LUPI framework for retaining PI in the multi-task setting to improve sample complexity over a wide range of related tasks. The key idea was to learn a joint representation of the original feature space and the PI by leveraging their co-occurrence information in the data. Decomposing the PI into distributed representations of basis features was vital for the realization of this mechanism. Experiments show that our proposed LUPI method can out-perform baseline models and other transfer learning methods in multi-task learning scenarios, particularly in situations where training samples are very rare ($<$ 100 samples per task). In addition to improved performance, we also provided sample complexity analysis that outline scenarios under which our LUPI method can provide similar benefits over traditional transfer learning approaches.

8 ACKNOWLEDGMENTS

This research was supported by a grant from the MIT-IBM Watson AI Lab. Work was done while FT was at MIT. JZ is supported by the National Science Foundation (NSF) IIS-1749940, IIS-1615597, IIS-1565596, and the Office of Naval Research (ONR) N00014-17-1-2265. FW is supported by NSF IIS-1750326, IIS-1716432 and ONR N00014-18-1-2585. LL is in part supported by NIH Grant R01GM104987

REFERENCES

[1] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. 2018. Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2018).
[2] Olivier Bodenreider. 2004. The unified medical language system (UMLS): integrating biomedical terminology. *Nucleic acids research* 32, suppl_1 (2004), D267–D270.
[3] Miguel A Carreira-Perpinan and Geoffrey E Hinton. 2005. On contrastive divergence learning.. In *Aistats*, Vol. 10. Citeseer, 33–40.
[4] Rich Caruana. 1997. Multitask learning. *Machine learning* 28, 1 (1997), 41–75.
[5] Edward Choi, Mohammad Taha Bahadori, Elizabeth Searles, Catherine Coffey, Michael Thompson, James Bost, Javier Tejedor-Sojo, and Jimeng Sun. 2016. Multi-layer representation learning for medical concepts. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1495–1504.
[6] 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*. ACM, 787–795.
[7] Youngduck Choi, Chill Yi-I Chiu, and David Sontag. 2016. Learning low-dimensional representations of medical concepts. *AMIA Summits on Translational Science Proceedings* 2016 (2016), 41.
[8] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. 2017. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. *arXiv preprint* 1711 (2017).
[9] Samuel G Finlayson, Paea LePendu, and Nigam H Shah. 2014. Building the graph of medicine from millions of clinical narratives. *Scientific data* 1 (2014), 140032.
[10] Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *null*. IEEE, 1735–1742.
[11] Hrayr Harutyunyan, Hrant Khachatrian, David C Kale, and Aram Galstyan. 2017. Multitask learning and benchmarking with clinical time series data. *arXiv preprint arXiv:1703.07771* (2017).
[12] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).
[13] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific data* 3 (2016).
[14] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, Vol. 2.
[15] John Lambert, Ozan Sener, and Silvio Savarese. 2018. Deep Learning under Privileged Information Using Heteroscedastic Dropout. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8886–8895.
[16] Xue Li, Bo Du, Chang Xu, Yipeng Zhang, Lefei Zhang, and Dacheng Tao. 2018. R-SVM+: Robust Learning with Privileged Information.. In *IJCAI*. 2411–2417.
[17] Tsungnan Lin, Bill G Horne, and C Lee Giles. 1998. How embedded memory in recurrent neural network architectures helps learning long-term temporal dependencies. *Neural Networks* 11, 5 (1998), 861–868.
[18] Zachary C Lipton, David C Kale, Charles Elkan, and Randall Wetzel. 2015. Learning to diagnose with LSTM recurrent neural networks. *arXiv preprint arXiv:1511.03677* (2015).
[19] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.
[20] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. 2011. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*. 689–696.
[21] Sinno Jialin Pan, Qiang Yang, et al. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2010), 1345–1359.
[22] Dhanesh Ramachandram and Graham W Taylor. 2017. Deep multimodal learning: A survey on recent advances and trends. *IEEE Signal Processing Magazine* 34, 6 (2017), 96–108.
[23] Shai Shalev-Shwartz and Shai Ben-David. 2014. *Understanding machine learning: From theory to algorithms*. Cambridge university press.
[24] Fengyi Tang, Cao Xiao, Fei Wang, and Jiayu Zhou. 2018. Predictive modeling in urgent care: a comparative study of machine learning approaches. *JAMIA Open* 1, 1 (2018), 87–98.
[25] Alexander B Tsybakov et al. 2004. Optimal aggregation of classifiers in statistical learning. *The Annals of Statistics* 32, 1 (2004), 135–166.
[26] Vladimir Vapnik and Rauf Izmailov. 2015. Learning using privileged information: similarity control and knowledge transfer. *Journal of machine learning research* 16, 2023-2049 (2015), 2.
[27] Vladimir Vapnik and Akshay Vashist. 2009. A new learning paradigm: Learning using privileged information. *Neural networks* 22, 5-6 (2009), 544–557.
[28] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks?. In *Advances in neural information processing systems*. 3320–3328.
[29] Jing Zhao, Xijiong Xie, Xin Xu, and Shiliang Sun. 2017. Multi-view learning overview: Recent progress and new challenges. *Information Fusion* 38 (2017), 43–54.
[30] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint* (2017).