

CHEER: Rich Model Helps Poor Model via Knowledge Infusion

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

Trong Nghia Hoang*
MIT-IBM Watson AI Lab
Cambridge, MA, USA
nghiaht@ibm.com

Shenda Hong*
Department of Computer Science and
Engineering, Georgia Institute of
Technology
Atlanta, GA, USA
sdhong1503@gmail.com

Tengfei Ma
IBM Research
Yorktown Heights, NY, USA
Tengfei.Ma1@ibm.com

Jimeng Sun
Department of Computer Science,
University of Illinois at
Urbana-Champaign
Urbana, IL, USA
jimeng@illinois.edu

ABSTRACT

There is a growing interest in applying deep learning (DL) to health-care, driven by the availability of data with multiple feature channels in *rich-data* environments (e.g., intensive care units). However, in many other practical situations, we can only access data with much fewer feature channels in a *poor-data* environments (e.g., at home), which often results in predictive models with poor performance. How can we boost the performance of models learned from such *poor-data* environment by leveraging knowledge extracted from existing models trained using *rich data* in a related environment? To address this question, we develop a knowledge infusion framework named CHEER that can succinctly summarize such *rich model* into transferable representations, which can be incorporated into the *poor model* to improve its performance. The infused model is analyzed theoretically and evaluated empirically on several datasets. Our empirical results showed that CHEER outperformed baselines by 5.60% to 46.80% in terms of the macro-F1 score on multiple physiological datasets.

KEYWORDS

Health Analytics, Representation Learning, Embedding

1 INTRODUCTION

In *rich-data* environments with strong observation capabilities, data often come with rich representations that encompass multiple channels of features. For example, multiple leads of Electrocardiogram (ECG) signals in hospital used for diagnosing heart diseases are measured in intensive care units (ICU), of which each lead is considered a feature channel. The availability of such rich-data environment has thus sparked strong interest in applying deep learning (DL) for predictive health analytics as DL models built on data with multi-channel features have demonstrated promising results in healthcare [40]. However, in many practical scenarios, such rich data are often private and not accessible due to privacy concern. Thus, we often have to develop DL models on lower quality data comprising fewer feature channels, which were collected from *poor-data* environments with limited observation capabilities

(e.g., home monitoring devices which provide only a single channel of feature). Inevitably, the performance of state-of-the-art DL models, which are fueled by the abundance and richness of data, becomes much less impressive in such poor-data environments [33].

To alleviate this issue, we hypothesize that learning patterns consolidated by DL models trained in one environment often encode information that can be transferred to related environments. For example, a heart disease detection model trained on *rich-data* from 12 ECG channels in a hospital will likely carry pertinent information that can help improve a similar model trained on *poor-data* from a single ECG channel collected by a wearable device due to the correlation between their data. Motivated by this intuition, we further postulate that given access to a prior model trained on *rich-data*, the performance of a DL model built on a related *poor-data* can be improved if we can extract transferable information from the *rich* model and infuse them into the *poor* model. This is related to deep transfer learning and knowledge distillation but with a new setup that has not been addressed before, as elaborated in Section 2 below.

In this work, we propose a knowledge infusion framework, named CHEER, to address the aforementioned challenges. In particular, CHEER aims to effectively transfer domain-invariant knowledge consolidated from a *rich* model with high-quality data demand to a *poor* model with low data demand and model complexity, which is more suitable for deployment in *poor-data* settings. We also demonstrate empirically that CHEER helps bridge the performance gap between DL models applied in *rich*- and *poor*-data settings. Specifically, we have made the following key contributions:

1. We develop a transferable representation that summarizes the *rich model* and then infuses the summarized knowledge effectively into the *poor model* (Section 3.2). The representation can be applied to a wide range of existing DL models.
2. We perform theoretical analysis to demonstrate the efficiency of knowledge infusion mechanism of CHEER. Our theoretical results show that under practical learning configurations and mild assumptions, the *poor model*'s prediction will agree with that of the *rich*

*Equal contribution

model with high probability (Section 4).

3. Finally, we also conduct extensive empirical studies to demonstrate the efficiency of CHEER on several healthcare datasets. Our results show that CHEER outperformed the second best approach (knowledge distillation) and the baseline without knowledge infusion by 5.60% and 46.80%, respectively, in terms of macro-F1 score and demonstrated more robust performance (Section 5).

2 RELATED WORKS

Deep Transfer Learning: Most existing deep transfer learning methods transfer knowledge across domains while assuming the target and source models have equivalent modeling and/or data representation capacities. For example, deep domain adaptation have focused mainly on learning domain-invariant representations between very specific domains (e.g., image data) on [4, 6, 9, 11, 16, 22, 32, 42, 46]. Furthermore, this can only be achieved by training both models jointly on source and target domain data.

More recently, another type of deep transfer learning [45] has been developed to transfer only the attention mechanism [3] from complex to shallow neural network to boost its performance. Both source and target models, however, need to be jointly trained on the same dataset. In our setting, since source and target datasets are not available at the same time and that the target model often has to adopt representations with significantly less modeling capacity to be compatible with the *poor-data* domain with weak observation capabilities.

Knowledge Distillation: Knowledge distillation [15] or mimic learning [1] aim to transfer the predictive power from a high-capacity but expensive DL model to a simpler model such as shallow neural networks for ease of deployment [23, 30, 34, 44]. This can usually be achieved via training simple models on soft labels learned from high-capacity models, which, however, assume that both models operate on the same domain and have access to the same data or at least datasets with similar qualities. In our setting, we only have access to low-quality data with *poor* feature representation, and an additional set of limited *paired* data that include both *rich* and *poor* representations (e.g., high-quality ICU data and lower-quality health-monitoring information from personal devices) of the same object.

Domain Adaptation: There also exists another body of non-deep learning transfer paradigms that were often referred to as domain adaption. This however often include methods that not only assume access domain-specific [24, 29, 36–38] and/or model-specific knowledge of the domains being adapted [17, 20, 25, 27, 28, 35, 41, 43], but are also not applicable to deep learning models [10, 39] with arbitrary architecture as addressed in our work.

In particular, our method does not impose any specific assumption on the data domain and the deep learning model of interest. We recognize that our method is only demonstrated on deep model (with arbitrary architecture) in this research but our formulation can be straightforwardly extended to non-deep model as well. We

Table 1: Notations used in CHEER.

Notation	Definition
$\mathcal{H}_r \triangleq \{(\mathbf{x}_i^r, y_i^r)\}_{i=1}^n; \mathcal{T}(y \mathbf{x}^r)$	rich data; rich model
$\mathcal{H}_p \triangleq \{(\mathbf{x}_i^p, y_i^p)\}_{i=1}^m; \mathcal{S}(y \mathbf{x}^p)$	poor data; poor model
$\mathcal{H}_o \triangleq \{(\mathbf{x}_i^r, \mathbf{x}_i^p, y_i)\}_{i=1}^k$	paired data
$\mathbf{Q}_r \triangleq [q_r^1 \dots q_r^{l_r}] \in \mathbb{R}^{d \times l_r}$	domain-specific features
$\mathbf{A}_r(\mathbf{x}^r) \triangleq [a_r^{(1)}(\mathbf{x}^r) \dots a_r^{(d)}(\mathbf{x}^r)]$	feature scoring functions
$\mathbf{O}_r \triangleq \mathbf{O}_r(\mathbf{Q}_r^\top \mathbf{A}_r(\mathbf{x}^r))$	feature aggregation component

omit such detail in the current manuscript to keep the focus on deep models which are of greater interest in healthcare context due to their expressive representation in modeling multi-channel data.

3 THE CHEER METHOD

3.1 Data and Problem Definition

Rich and Poor Datasets. Let $\mathcal{H}_r \triangleq \{(\mathbf{x}_i^r, y_i^r)\}_{i=1}^n$ and $\mathcal{H}_p \triangleq \{(\mathbf{x}_i^p, y_i^p)\}_{i=1}^m$ denote the *rich* and *poor* datasets, respectively. The subscript i indexes the i -th data point (e.g., the i -th patient in healthcare applications), which contains input feature vector \mathbf{x}_i^r or \mathbf{x}_i^p and output target y_i^r or y_i^p of the rich or poor datasets. The *rich* and *poor* input features $\mathbf{x}_i^r \in \mathbb{R}^r$ and $\mathbf{x}_i^p \in \mathbb{R}^p$ are r - and p -dimensional vectors with $p \ll r$, respectively. The output targets, y_i^r and $y_i^p \in \{1 \dots c\}$, are categorical variables. The input features of these datasets (i.e., \mathbf{x}_i^r and \mathbf{x}_i^p) are non-overlapping as they are assumed to be collected from different channels of data (i.e., different **data modalities**). In the remaining of this paper, we will use **data channel** and **data modality** interchangeably.

For example, the rich data can be the physiological data from ICU (e.g., vital signs, continuous blood pressure and electrocardiography) or temporal event sequences such as electronic health records with discrete medical codes, while the poor data are collected from personal wearable devices. The target can be the mortality status of those patients, onset of heart diseases and etc. Note that these raw data are not necessarily plain feature vectors. They can be arbitrary rich features such as time series, images and text data. We will present one detailed implementation using time series data in Section 3.3.

Input Features. We (implicitly) assume that the raw data of interest comprises (says, p or r) multiple sensory channels, each of which can be represented by or embedded¹ into a particular feature signal (i.e., **one feature per channel**). This results in an embedded feature vector of size p or r (per data point), respectively. In a different practice, a single channel may be encoded by multiple latent features and our method will still be applicable. In this paper, however, we will **assume one embedded feature per channel** to remain close to the standard setting of our healthcare scenario, which is detailed below.

¹We embed these channel jointly rather than separately to capture their latent correlation.

Paired Dataset. To leverage both rich and poor datasets, we need a small amount of **paired data** to learn the relationships between them, which is denoted as $\mathcal{H}_o \triangleq \{(\mathbf{x}_i^r, \mathbf{x}_i^p, y_i)\}_{i=1}^k$. Note that the paired dataset contains both *rich* and *poor* input features, i.e. \mathbf{x}_i^r and \mathbf{x}_i^p , of the same subjects (hence, sharing the same target y_i).

Concretely, this means a concatenated input $\mathbf{x}_i^o = [\mathbf{x}_i^r, \mathbf{x}_i^p]$ of the paired dataset has $o = p + r$ features where the first r features are collected from r rich channels (with highly accurate observation capability) while the remaining p features are collected from p poor channels (with significantly more noisy observations). We note that our method and analysis also apply to settings where $\mathbf{x}_i^p \subseteq \mathbf{x}_i^r$. In such cases, $\mathbf{x}_i^o = \mathbf{x}_i^r$ and $o = r$ (though the number of data point i for which \mathbf{x}_i^r is accessible as paired data is much less than the number of those with accessible \mathbf{x}_i^p). To avoid confusion, however, we will proceed with the implicit assumption that there is no feature overlapping between poor and rich datasets in the remaining of this paper.

For example, the paired dataset may comprise of rich data from ICU (\mathbf{x}_i^p) and poor data from wearable sensors (\mathbf{x}_i^r), which are extracted from the same patient i . The paired dataset often contains much fewer data points (i.e., patients) than the rich and poor datasets themselves, and cannot be used alone to train a prediction model with high quality.

Problem Definition. Given (1) a poor dataset \mathcal{H}_p collected from a particular patient cohort of interest, (2) a paired dataset \mathcal{H}_o collected from a limited sample of patients, and (3) a *rich* model $\mathcal{T}(y|\mathbf{x}^r)$ which were pre-trained on private (rich) data of the same patient cohort, we are interested in learning a model $\mathcal{S}(y|\mathbf{x}^p)$ using both $\mathcal{H}_p, \mathcal{H}_o$ and $\mathcal{T}(y|\mathbf{x}^r)$, which can perform better than a vanilla model $\mathcal{D}(y|\mathbf{x}^p)$ generated using only \mathcal{H}_p or \mathcal{H}_o .

Challenges. This requires the ability to transfer the learned knowledge from $\mathcal{T}(y|\mathbf{x}^r)$ to improve the prediction quality of $\mathcal{S}(y|\mathbf{x}^p)$. This is however a highly non-trivial task because (a) $\mathcal{T}(y|\mathbf{x}^r)$ only generates meaningful prediction if we can provide input from rich data channels, (b) its training data is private and cannot be accessed to enable knowledge distillation and/or domain adaptation, and (c) the paired data is limited and cannot be used alone to build an accurate prediction model.

Solution Sketch. Combining these sources of information coherently to generate a useful prediction model on the patient cohort of interest is therefore a challenging task which has not been investigated before. To address this challenge, the idea is to align both rich and poor models using a transferable representation described in Section 3.2. This representation in turn helps infuse knowledge from the rich model into the poor model, thus improving its performance. The overall structure of CHEER is shown in Figure 1. The notations are summarized in Table 1.

3.2 Learning Transferable Rich Model

In our knowledge infusion task, the *rich* model is assumed to be trained in advance using the rich dataset $\mathcal{H}_r \triangleq \{(\mathbf{x}_i^r, y_i^r)\}_{i=1}^n$. The

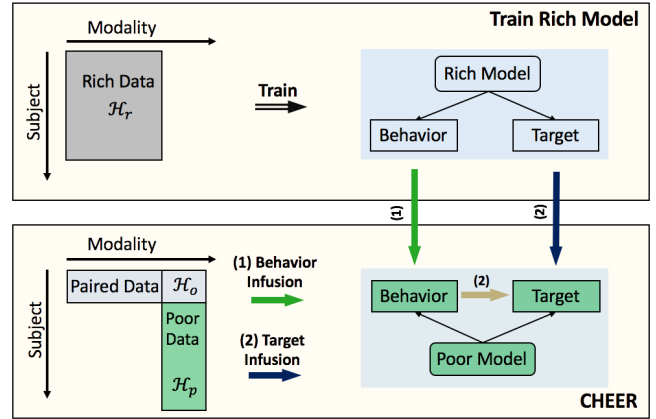


Figure 1: CHEER: (a) a *rich* model was first built using rich multi-modal or multi-channel data; (b) the behaviors of *rich* model are then infused into the *poor* model using paired data (i.e., behavior infusion); and (c) the *poor* model is trained to fit both the *rich* model’s predictions on paired data and its *poor* dataset (i.e., target infusion).

rich dataset is, however, not accessible and we only have access to the *rich* model. The knowledge infusion task aims to consolidate the knowledge acquired by the rich model and infuse it with a simpler model (i.e., the *poor* model).

Transferable Representation. To characterize a DL model, we first describe the building blocks and then discuss how they would interact to generate the final prediction scores. In particular, let $\mathbf{Q}_r(\mathbf{x}^r)$, $\mathbf{A}_r(\mathbf{x}^r)$ and \mathbf{O}_r denote the building blocks, namely *Feature Extraction*, *Feature Scoring* and *Feature Aggregation*, respectively. Intuitively, the *Feature Extraction* first transforms raw input feature \mathbf{x}^r into a vector of high-level features $\mathbf{Q}_r(\mathbf{x}^r)$, whose importance are then scored by the *Feature Scoring* function $\mathbf{A}_r(\mathbf{x}^r)$. The high-level features $\mathbf{Q}_r(\mathbf{x}^r)$ are combined first via a linear transformation $\mathbf{Q}_r(\mathbf{x}^r)^\top \mathbf{A}_r(\mathbf{x}^r)$ that focuses the model’s attention on important features. The results are translated into a vector of final predictive probabilities via the *Feature Aggregation* function $\mathbf{O}_r(\mathbf{Q}_r(\mathbf{x}^r)^\top \mathbf{A}_r(\mathbf{x}^r))$, which implements a non-linear transformation. Mathematically, the above workflow can be succinctly characterized using the following conditional probability distributions:

$$\mathcal{T}(y|\mathbf{x}^r) \triangleq \Pr(y|\mathbf{Q}_r^\top(\mathbf{x}^r)\mathbf{A}_r(\mathbf{x}^r); \mathbf{O}_r). \quad (1)$$

We will describe these building blocks in more details next.

Feature Extraction. Dealing with complex input data such as time series, images and text, it is common to derive more effective features instead of directly using the raw input \mathbf{x}^r . The extracted features are denoted as $\mathbf{Q}_r(\mathbf{x}^r) \triangleq [q_r^1(\mathbf{x}^r) \dots q_r^l(\mathbf{x}^r)] \in \mathbb{R}^{d \times l_r}$ where $q_r^i(\mathbf{x}^r) \in \mathbb{R}^d$ is a d -dimensional feature vector extracted by the i -th feature extractor from the raw input \mathbf{x}^r . Each feature extractor is applied to a separate segment of the time series input (defined later in Section 3.3). To avoid cluttering the notations, we shorten $\mathbf{Q}_r(\mathbf{x}^r)$

as \mathbf{Q}_r .

Feature Scoring. Since the extracted features are of various importance to each subject, they are combined via weights specific to each subject. More formally, the extracted features $\mathbf{Q}_r(\mathbf{x}^r)$ of the *rich model* are combined via $\mathbf{Q}_r^\top(\mathbf{x}^r)\mathbf{A}_r(\mathbf{x}^r)$ using subject-specific weight vector $\mathbf{A}_r(\mathbf{x}^r) \triangleq [a_r^{(1)}(\mathbf{x}^r) \dots a_r^{(d)}(\mathbf{x}^r)] \in \mathbb{R}^d$.

Essentially, each weight component $a_r^{(i)}(\mathbf{x}^r)$ maps from the raw input feature \mathbf{x}^r to the important score of its i^{th} extracted feature. For each dimension i , $a_r^{(i)}(\mathbf{x}^r) \triangleq a_r^{(i)}(\mathbf{x}^r; \boldsymbol{\omega}_r^{(i)})$ parameterized by a set of parameters $\boldsymbol{\omega}_r^{(i)}$, which are learned using the *rich* dataset.

Feature Aggregation. The *feature aggregation* implements a non-linear transformation \mathbf{O}_r (e.g., a feed-forward layer) that maps the combined features into final predictive scores. The input to this component is the linearly combined feature $\mathbf{Q}_r(\mathbf{x}^r)^\top \mathbf{A}_r(\mathbf{x}^r)$ and the output is a vector of logistic inputs,

$$\mathbf{r}(\mathbf{x}^r) \triangleq [r_1 \dots r_c] = \mathbf{O}_r \left(\mathbf{Q}_r(\mathbf{x}^r)^\top \mathbf{A}_r(\mathbf{x}^r) \right), \quad (2)$$

which is subsequently passed through the softmax function to compute the predictive probability for each candidate label,

$$\mathcal{T}(y = j | \mathbf{x}_i^r) \triangleq \exp(r_j) \left/ \left(\sum_{\kappa=1}^c \exp(r_\kappa) \right) \right. . \quad (3)$$

3.3 A DNN Implementation of Rich Model

This section describes an instantiation of the aforementioned abstract building blocks using a popular DNN architecture with self-attention mechanism [21] for modeling multivariate time series [7]:

Raw Features. Raw data from rich data environment often consist of multivariate time series such as physiological signals collected from hospital or temporal event sequences such as electronic health records (EHR) with discrete medical codes. In the following, we consider the raw feature input \mathbf{x}_i^r as continuous monitoring data (e.g., blood pressure measures) for illustration purpose.

Feature Extraction. To handle such continuous time series, we extract a set of domain-specific features using CNN and RNN models. More specifically, we splits the raw time series \mathbf{x}_i^r into l_r non-overlapping segments of equal length.

That is, $\mathbf{x}_i^r \triangleq (\mathbf{s}_{i,m}^r)$ where $m = 1 \dots l_r$ and $\mathbf{s}_{i,m}^r \in \mathbb{R}^{D_r}$ such that $D_r \times l_r = r$ with r denotes the number of features of the rich data. Then, we apply stacked 1-D convolutional neural networks (CNN_r) with mean pooling (\mathbb{P}_r) on each segment, i.e.

$$\mathbf{h}_{i,m}^r \triangleq \mathbb{P}_r \left(\text{CNN}_r \left(\mathbf{s}_{i,m}^r \right) \right) \quad (4)$$

where $\mathbf{h}_{i,m}^r \in \mathbb{R}^{k_r}$, and k_r denotes the number of filters of the CNN components of the rich model. After that, we place a recurrent neural network (RNN_r) across the output segments of the previous CNN and Pooling layers:

$$\mathbf{q}_{i,m}^r \triangleq \text{RNN}_r \left(\mathbf{q}_{i,m-1}^r, \mathbf{h}_{i,m}^r \right) \in \mathbb{R}^d, \quad (5)$$

The output segments of the RNN layer are then concatenated to generate the feature matrix,

$$\mathbf{Q}_r^{(i)} \triangleq \left[\mathbf{q}_{i,1}^r \dots \mathbf{q}_{i,l_r}^r \right] \in \mathbb{R}^{d \times l_r}, \quad (6)$$

which correspond to our domain-specific feature extractors $\mathbf{Q}_r(\mathbf{x}_i^r) \triangleq [q_r^1(\mathbf{x}_i^r) \dots q_r^{l_r}(\mathbf{x}_i^r)]$ where $q_r^t(\mathbf{x}_i^r) = \mathbf{q}_{i,t}^r \in \mathbb{R}^d$, as defined previously in our transferable representation (Section 3.2).

Feature Scoring. The concatenated features $\mathbf{Q}_r^{(i)}$ is then fed to the self-attention component ATT_r to generate a vector of importance scores for the output components, i.e. $\mathbf{a}_r^{(i)} \triangleq \text{ATT}_r(\mathbf{Q}_r^{(i)}) \in \mathbb{R}^d$. For more details on how to construct this component, see [2, 8, 14] and [21]. The result corresponds to the feature scoring functions² $\mathbf{A}_r(\mathbf{x}_i^r) \triangleq [a_r^{(1)}(\mathbf{x}_i^r) \dots a_r^{(d)}(\mathbf{x}_i^r)]$ where $a_r^{(t)}(\mathbf{x}_i^r) = [\mathbf{a}_r^{(i)}]_t \in \mathbb{R}$.

Feature Aggregation. The extracted features $\mathbf{Q}_r^{(i)}$ are combined using the above feature scoring functions, which yields $\mathbf{Q}_r^{(i)\top} \mathbf{a}_r^{(i)}$. The combined features are subsequently passed through a linear layer with densely connected hidden units (DENSE_r),

$$\mathbf{g}_r^{(i)} \triangleq \text{DENSE}_r \left(\mathbf{Q}_r^{(i)\top} \mathbf{a}_r^{(i)}; \mathbf{w}_r \right), \quad (7)$$

where $\mathbf{g}_r^{(i)} \in \mathbb{R}^c$ with c denotes the number of class labels and \mathbf{w}_r denotes the parametric weights of the dense layers. Then, the output of the dense layer is transformed into a probability distribution over class labels via the following softmax activation functions parameterized with softmax temperatures τ_r :

$$\mathcal{T}(y = j | \mathbf{x}_i^r) \triangleq \exp \left(\left[\mathbf{g}_r^{(i)} \right]_j / \tau_r \right) \left/ \sum_{\kappa=1}^c \exp \left(\left[\mathbf{g}_r^{(i)} \right]_\kappa / \tau_r \right) \right.$$

The entire process corresponds to the feature aggregation function $\mathbf{O}_r(\mathbf{Q}_r(\mathbf{x}^r)^\top \mathbf{A}_r(\mathbf{x}^r))$ parameterized by $\{\mathbf{w}_r, \tau_r\}$.

3.4 Knowledge Infusion for Poor Model

To infuse the above knowledge extracted from the *rich* model to the *poor* model, we adopt the same transferable representation for the *poor* model as follows:

$$\mathcal{S}(y | \mathbf{x}^p) \triangleq \Pr \left(y | \mathbf{Q}_p^\top(\mathbf{x}^p) \mathbf{A}_p(\mathbf{x}^p); \mathbf{O}_p \right)$$

where $\mathbf{Q}_p, \mathbf{A}_p(\mathbf{x}^p) \triangleq [a_p^{(1)}(\mathbf{x}^p; \boldsymbol{\omega}_p^{(1)}) \dots a_p^{(d)}(\mathbf{x}^p; \boldsymbol{\omega}_p^{(d)})] \in \mathbb{R}^d$ and \mathbf{O}_p are the *poor* model's domain-specific feature extractors, feature scoring functions and feature aggregation functions, which are similar in format to those of the *rich* model. Infusing knowledge from the *rich* model to the *poor* model can then be boiled down to matching these components between them. This process can be decomposed into two steps:

Behavior Infusion. As mentioned above, each scoring function $\mathbf{a}_p^{(i)}(\mathbf{x}^p; \boldsymbol{\omega}_p^{(i)})$ is defined by a weight vector $\boldsymbol{\omega}_p^{(i)}$. The collection of these weight vectors thus defines the *poor model's learning behaviors* (i.e., its feature scoring mechanism).

²We use the notation $[a]_t$ to denote the t -th component of vector \mathbf{a} .

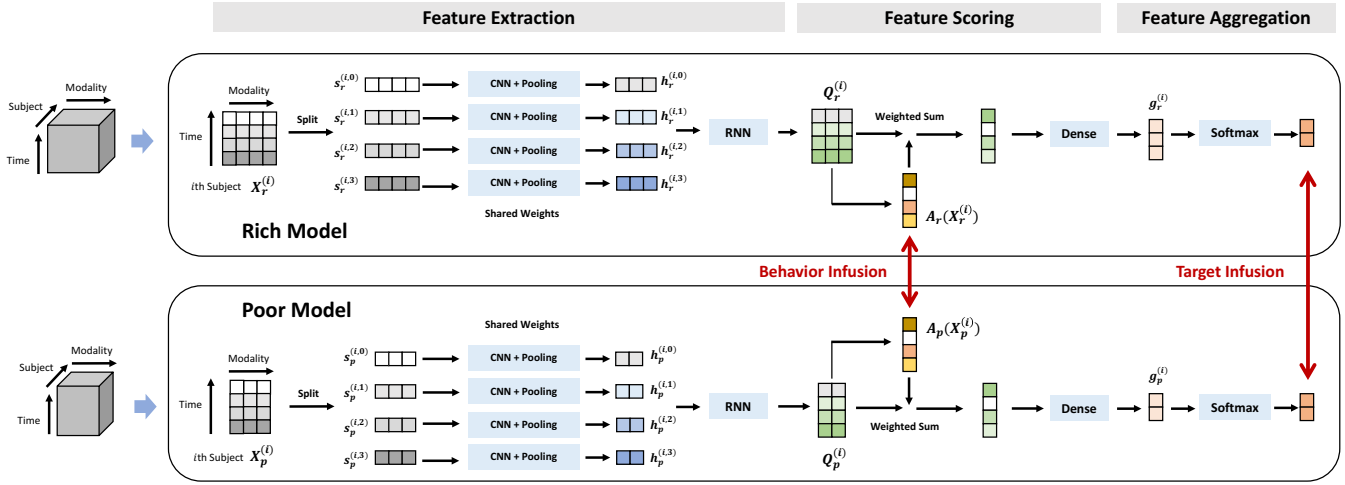


Figure 2: The DNN Implementation of CHEER.

Given the input components $\{(\mathbf{x}_t^p, \mathbf{x}_t^r)\}_{t=1}^k$ of the subjects included in the paired dataset \mathcal{H}_o and the *rich model*'s scoring outputs $\{a_r^{(i)}(\mathbf{x}_t^r)\}_{t=1}^k$ at those subjects, we can construct an auxiliary dataset $\mathcal{B}_i \triangleq \{(\mathbf{x}_t^p, a_r^{(i)}(\mathbf{x}_t^r))\}_{t=1}^k$ to learn the corresponding behavior $\omega_p^{(i)}$ of the *poor model* so that its scoring mechanism is similar to that of the *rich model*. That is, we want to learn a mapping from a *poor* data point \mathbf{x}^p to the important score assigned to its i^{th} latent feature by the *rich model*. Formally, this can be cast as the optimization task given by Eq. 8:

$$\begin{aligned} \underset{\omega_p^{(i)}}{\text{minimize}} \mathcal{L}_i(\omega_p^{(i)}) &\triangleq \frac{1}{2} \sum_{t=1}^k \left(a_p^{(i)}(\mathbf{x}_t^p; \omega_p^{(i)}) - a_r^{(i)}(\mathbf{x}_t^r) \right)^2 \\ &+ \lambda \|\omega_p^{(i)}\|_2^2. \end{aligned} \quad (8)$$

For example, if we parameterize $a_p^{(i)}(\mathbf{x}_t^p; \omega_p^{(i)}) = \omega_p^{(i)\top} \mathbf{x}_t^p$ and choose $\lambda = 0$, then Eq. 8 reduces to a linear regression task, which can be solved analytically. Alternatively, by choosing $\lambda = 1$, Eq. 8 reduces to a maximum a posterior (MAP) inference task with normal prior imposed on $\omega_p^{(i)}$, which is also analytically solvable.

Incorporating more sophisticated, non-linear parameterization for $a_p^{(i)}(\mathbf{x}_t^p; \omega_p^{(i)})$ (e.g., deep neural network with varying structures) is also possible but Eq. 8 can only be optimized approximately via numerical methods (see Section 3.2). Eq. (8) can be solved via standard gradient descent. The complexity of deriving the solution thus depends on the number of iteration τ and the cost of computing the gradient of $\omega_p^{(i)}$ which depends on the parameterization of $a_p^{(i)}$ but is usually $O(w)$ where $w = \max_t |\omega_p^{(i)}|$. As such, the cost of computing the gradient of the objective function with respect to a particular i is $O(kw)$. As there are τ iterations, the cost of solving for the optimal $\omega_p^{(i)}$ is $O(\tau kw)$. Lastly, since we are doing this for d values of i , the total complexity would be $O(\tau kwd)$.

Target Infusion. Given the *poor model*'s learned behaviors $\{\omega_p^{(i)}\}_{i=1}^d$ (which were fitted to those of the *rich model* via solving Eq. 8), we

now want to optimize the *poor model*'s feature aggregation \mathbf{O}_p and feature extraction \mathbf{Q}_p components so that its predictions will (a) fit those of the *rich model* on paired data \mathcal{H}_o ; and also (b) fit the ground truth $\{y_t^p\}_{t=1}^m$ provided by the *poor data* \mathcal{H}_p . Formally, this can be achieved by solving the following optimization task:

$$\begin{aligned} \underset{\mathbf{O}_p, \mathbf{Q}_p}{\text{min}} \mathcal{L}_p &\triangleq \frac{1}{k} \sum_{t=1}^k \sum_{y=1}^c \left(\mathcal{T}(y_t | \mathbf{x}_t^r; \mathbf{O}_r, \mathbf{Q}_r) - \mathcal{S}(y | \mathbf{x}_t^p; \mathbf{O}_p, \mathbf{Q}_p) \right)^2 \\ &+ \frac{1}{m} \sum_{t=1}^m \left(1 - \mathcal{S}(y_t^p | \mathbf{x}_t^p; \mathbf{O}_p, \mathbf{Q}_p) \right)^2 \end{aligned} \quad (9)$$

To understand the above, note that the first term tries to fit *poor model* \mathcal{S} to *rich model* \mathcal{T} in the context of the paired dataset $\mathcal{H}_o \triangleq \{(\mathbf{x}_t^p, \mathbf{x}_t^r, y_t)\}_{t=1}^k$ while the second term tries to adjust the *poor model*'s fitted behavior and target in a local context of its *poor data* $\mathcal{H}_p \triangleq \{(\mathbf{x}_t^p, y_t^p)\}_{t=1}^m$. This allows the second term to act as a filter that downplays distilled patterns which are irrelevant in the *poor data* context. Again, Eq. 9 can be solved depending on how we parameterize the aforementioned components $(\mathbf{O}_p, \mathbf{Q}_p)$.

For example, \mathbf{O}_p can be set as a linear feed-forward layer with densely connected hidden units, which are activated by a softmax function. Again, Eq. (9) could be solved via standard gradient descent. The cost of computing the gradient would depend linearly on the total no. n_p of neurons in the parameterization of \mathbf{O}_p and \mathbf{Q}_p for the *poor model*. In particular, the gradient computation complexity for one iteration is $O(n_p(mpc + kc^2))$. For τ iteration, the total cost would be $O(\tau n_p(mpc + kc^2))$.

Both steps of **behavior infusion** and **target infusion** are succinctly summarized in Algorithm 1 below.

Algorithm 1 CHEER ($\mathcal{H}_p, \mathcal{T}(y|\mathbf{x}^r), \mathcal{H}_o$)

- 1: Input: *rich model* $\mathcal{T}(y|\mathbf{x}^r)$, *poor data* \mathcal{H}_p and paired data \mathcal{H}_o
 - 2: Infuse *rich model*'s behavior via \mathcal{H}_o
 - 3: $i \leftarrow 1$
 - 4: **while** $i \leq d$ **do**
 - 5: $\omega_p^{(i)} \leftarrow \operatorname{argmin} \mathcal{L}_i(\omega_p^{(i)})$ via (8);
 - 6: $a_p^{(i)}(\mathbf{x}^p) \leftarrow a_p^{(i)}(\mathbf{x}^p; \omega_p^{(i)})$
 - 7: $i \leftarrow i + 1$
 - 8: **end while**
 - 9: $A_p \leftarrow [a_p^{(1)}(\mathbf{x}^p; \omega_p^{(1)}) \dots a_p^{(d)}(\mathbf{x}^p; \omega_p^{(d)})]$
 - 10: Infuse *rich model*'s target via ($\mathcal{H}_o, \mathcal{H}_p$) and A_p
 - 11: $(Q_p, O_p) \leftarrow \operatorname{argmin} \mathcal{L}_p$ via (9)
 - 12: Output: *poor model* $S(y|\mathbf{x}^p) \leftarrow (A_p, Q_p, O_p)$
-

4 THEORETICAL ANALYSIS

In this section, we provide theoretical analysis for CHEER. Our goal is to show that under certain practical assumptions and with respect to a random instance $\mathbf{x} = (\mathbf{x}^p, \mathbf{x}^r) \sim \mathcal{P}(\mathbf{x})$ drawn from an arbitrary data distribution \mathcal{P} , the prediction $y^p \triangleq \operatorname{argmax} S(y|\mathbf{x}^p)$ of the resulting *poor model* will agree with that of the *rich model*, $y^r \triangleq \operatorname{argmax} \mathcal{T}(y|\mathbf{x}^r)$, with high probability, thus demonstrating the accuracy of our knowledge infusion algorithm in Section 3.4.

High-Level Ideas. To achieve this, our strategy is to first bound the *expected* target fitting loss (see Definition 2) on a random instance $\mathbf{x} \triangleq (\mathbf{x}^p, \mathbf{x}^r) \sim \mathcal{P}(\mathbf{x})$ of the *poor model* with respect to its optimized scoring function A_p , feature extraction Q_p and feature aggregation O_p components via solving Eq. 8 and Eq. 9 in Section 3.4 (see Lemma 1).

We can then characterize the sufficient condition on the target fitting loss (see Definition 1) with respect to a particular instance $\mathbf{x} \triangleq (\mathbf{x}^p, \mathbf{x}^r)$ for the *poor model* to agree with the *rich model* on their predictions of \mathbf{x}^p and \mathbf{x}^r , respectively (see Lemma 2). The probability that this sufficient condition happens can then be bounded in terms of the bound on the *expected* target fitting loss in Lemma 1 (see Theorem 1), which in turn characterizes how likely the *poor model* will agree with the *rich model* on the prediction of a random data instance. To proceed, we put forward the following assumptions and definitions:

Definition 1. Let $\theta_p \triangleq \{O_p, Q_p, A_p\}$ denote an arbitrary parameterization of the *poor model*. The *particular* target fitting loss of the *poor model* with respect to a data instance $\mathbf{x} \triangleq (\mathbf{x}^p, \mathbf{x}^r)$ is

$$\begin{aligned} \widehat{\mathcal{L}}_{\mathbf{x}}(\theta_p) &\triangleq \sum_{y=1}^c \left(\mathcal{T}(y|\mathbf{x}^r) - S(y|\mathbf{x}^p) \right)^2 \\ &+ \frac{1}{m} \sum_{t=1}^m \left(1 - S(y_t^p | \mathbf{x}_t^p) \right)^2, \end{aligned} \quad (10)$$

where c denotes the number of classes, $\mathcal{T}(y|\mathbf{x}^r)$ and $S(y|\mathbf{x}^p)$ denotes the probability scores assigned to candidate class y by the *rich* and *poor* models, respectively.

Definition 2. Let θ_p be defined as in Definition 1. The *expected* target fitting loss of the *poor model* with respect to the parameterization θ_p is defined below,

$$\mathcal{L}(\theta_p) \triangleq \mathbb{E}_{\mathbf{x} \sim \mathcal{P}(\mathbf{x})} \left[\widehat{\mathcal{L}}_{\mathbf{x}}(\theta_p) \right], \quad (11)$$

where the expectation is over the unknown data distribution $\mathcal{P}(\mathbf{x})$.

Definition 3. Let $\mathbf{x} = (\mathbf{x}^p, \mathbf{x}^r)$ and $y(\mathbf{x}) \triangleq \operatorname{argmax}_{y=1}^c \mathcal{T}(y|\mathbf{x}^r)$. The robustness constant of the *rich model* is defined below,

$$\phi \triangleq \frac{1}{2} \min_{(\mathbf{x}^p, \mathbf{x}^r)} \left(\mathcal{T}(y(\mathbf{x}) | \mathbf{x}^r) - \max_{y \neq y(\mathbf{x})} \mathcal{T}(y | \mathbf{x}^r) \right), \quad (12)$$

That is, if the probability scores of the model are being perturbed additively within ϕ , its prediction outcome will not change.

Assumption 1. The paired data points $\mathbf{x}_i = (\mathbf{x}_i^p, \mathbf{x}_i^r)$ of \mathcal{H}_o are assumed to be distributed independently and identically from $\mathcal{P}(\mathbf{x})$.

Assumption 2. The hard-label predictions $y^p \triangleq \operatorname{argmax} S(y|\mathbf{x}^p)$ and $y^r \triangleq \operatorname{argmax} \mathcal{T}(y|\mathbf{x}^r)$ of the *poor* and *rich* models are unique.

Given the above, we are now ready to state our first result:

Lemma 1. Let θ_p^* and $\widehat{\theta}_p$ denote the optimal parameterization of the *poor model* that yields the minimum *expected* target fitting loss (see Definition 2) and the optimal solution found by minimizing the objective functions in Eq. 8 and Eq. 9, respectively. Let $\alpha \triangleq \mathcal{L}(\theta_p^*)$, $\delta \in (0, 1)$ and c denote the number of classes in our predictive task. If $k \triangleq |\mathcal{H}_o| \geq ((c+1)^2 / (2\epsilon^2)) \log(2/\delta)$ then,

$$\Pr \left(\mathcal{L}(\widehat{\theta}_p) \leq \alpha + 2\epsilon \right) \geq 1 - \delta. \quad (13)$$

Proof. We first note that by definition in Eq. (10), for all \mathbf{x} , $\widehat{\mathcal{L}}_{\mathbf{x}}(\theta) \leq c + 1$. Then, let us define the *empirical* target fitting loss as

$$\widehat{\mathcal{L}}(\theta) \triangleq \frac{1}{k} \sum_{i=1}^k \widehat{\mathcal{L}}_{\mathbf{x}^{(i)}}(\theta), \quad (14)$$

where $\{\widehat{\mathcal{L}}_{\mathbf{x}^{(i)}}(\theta)\}_{i=1}^k$ can be treated as identically and independently distributed random variables in $(0, c+1)$. Then, by Definition 2, it also follows that $\mathcal{L}(\theta) = \mathbb{E}[\widehat{\mathcal{L}}(\theta)]$. Thus, by Hoeffding inequality:

$$\Pr \left(\left| \mathcal{L}(\theta) - \widehat{\mathcal{L}}(\theta) \right| \leq \epsilon \right) \geq 1 - 2\exp \left(-\frac{2k\epsilon^2}{(c+1)^2} \right). \quad (15)$$

Then, for an arbitrary $\delta \in (0, 1)$, setting $\delta \leq \exp(-2k\epsilon^2 / (c+1)^2)$ and solving for k yields $k \geq ((c+1)^2 / (2\epsilon^2)) \log(2/\delta)$. Thus, for $k \geq ((c+1)^2 / (2\epsilon^2)) \log(2/\delta)$, with probability at least $1 - \delta$, $|\mathcal{L}(\theta) - \widehat{\mathcal{L}}(\theta)| \leq \epsilon$ holds simultaneously for all θ . When that happens with probability at least $1 - \delta$, we have:

$$\begin{aligned} \mathcal{L}(\widehat{\theta}_p) &\leq \widehat{\mathcal{L}}(\widehat{\theta}_p) + \epsilon \\ &\leq \widehat{\mathcal{L}}(\theta_p^*) + \epsilon \leq \mathcal{L}(\theta_p^*) + 2\epsilon = \alpha + 2\epsilon. \end{aligned} \quad (16)$$

That is, $\Pr \left(\widehat{\mathcal{L}}(\widehat{\theta}_p) \leq \alpha + 2\epsilon \right) \geq 1 - \delta$, which completes our proof for Lemma 1. Note that the above 2nd inequality follows from the definition of $\widehat{\theta}_p \triangleq \operatorname{argmin}_{\theta} \widehat{\mathcal{L}}(\theta_p)$, which implies $\widehat{\mathcal{L}}(\widehat{\theta}_p) \leq \widehat{\mathcal{L}}(\theta_p^*)$.

This result implies the *expected* target fitting loss $\mathcal{L}(\widehat{\theta}_p)$ incurred by our knowledge infusion algorithm in Section 3.4 can be made arbitrarily close (with high confidence) to the optimal *expected* target fitting loss $\alpha \triangleq \mathcal{L}(\theta_p^*)$ with a sufficiently large paired dataset \mathcal{H}_o .

Lemma 2. Let $\mathbf{x} = (\mathbf{x}^p, \mathbf{x}^r)$ and $\widehat{\theta}_p$ as defined in Lemma 1. If the corresponding *particular* target fitting loss (see Definition 1) $\widehat{\mathcal{L}}_{\mathbf{x}}(\widehat{\theta}_p) \leq \phi^2$, then both *poor* and *rich* models agree on their predictions for \mathbf{x}^p and \mathbf{x}^r , respectively. That is, $y^p \triangleq \max_y \mathcal{S}(y|\mathbf{x}^p)$ and $y^r \triangleq \max_y \mathcal{T}(y|\mathbf{x}^r)$ are the same.

Proof. Let y^p and y^r be defined as in the statement of Lemma 2. We have:

$$\begin{aligned} \mathcal{S}(y^p | \mathbf{x}^p) &\geq \mathcal{S}(y^r | \mathbf{x}^p) \geq \mathcal{T}(y^r | \mathbf{x}^r) - \phi \\ &\geq \mathcal{T}(y^p | \mathbf{x}^r) + 2\phi - \phi \\ &\geq \mathcal{S}(y^p | \mathbf{x}^p) + 2\phi - 2\phi = \mathcal{S}(y^p | \mathbf{x}^p). \end{aligned} \quad (17)$$

To understand Eq. (17), note that the first inequality follows from the definition of y^p . The second inequality follows from the fact that $\widehat{\mathcal{L}}_{\mathbf{x}}(\widehat{\theta}_p) \leq \phi^2$, which implies $\forall y (\mathcal{S}(y|\mathbf{x}^p) - \mathcal{T}(y|\mathbf{x}^r))^2 \leq \phi^2$ and hence, $|\mathcal{S}(y^r|\mathbf{x}^p) - \mathcal{T}(y^r|\mathbf{x}^r)| \leq \phi$ or $\mathcal{S}(y^r|\mathbf{x}^p) \geq \mathcal{T}(y^r|\mathbf{x}^r) - \phi$. The third inequality follows from the definitions of ϕ (see Definition 3) and y^r . Finally, the last inequality follows from the definition of y^p and that $\widehat{\mathcal{L}}_{\mathbf{x}}(\widehat{\theta}_p) \leq \phi^2$, which also implies $\mathcal{T}(y^p|\mathbf{x}^r) \geq \mathcal{S}(y^p|\mathbf{x}^p) - \phi$.

Eq. (17) thus implies $\mathcal{S}(y^p|\mathbf{x}^p) \geq \mathcal{S}(y^r|\mathbf{x}^p) \geq \mathcal{S}(y^p|\mathbf{x}^p)$ and hence, $\mathcal{S}(y^p|\mathbf{x}^p) = \mathcal{S}(y^r|\mathbf{x}^p)$. Since the hard-label prediction is unique (see Assumption 3), this means $y^r = y^p$ and hence, by definitions of y^r and y^p , the *poor* and *rich* models yield the same prediction. This completes our proof for Lemma 2.

Intuitively, Lemma 2 specifies the sufficient condition under which the *poor model* will yield the same hard-label prediction on a particular data instance \mathbf{x} as the *rich model*. Thus, if we know how likely this sufficient condition will happen, we will also know how likely the *poor model* will imitate the *rich model* successfully on a random data instance. This intuition is the key result of our theoretical analysis and is formalized below:

Theorem 1. Let $\delta \in (0, 1)$ and $\mathbf{x} = (\mathbf{x}^p, \mathbf{x}^r)$ denote a random instance drawn from $\mathcal{P}(\mathbf{x})$. Let $k \triangleq |\mathcal{H}_o|$ denote the size of the paired dataset \mathcal{H}_o , which were used to fit the learning behaviors of the *poor model* to that of the *rich model*, and \mathcal{E} denotes the event that both models agree on their predictions of \mathbf{x} . If $k \geq ((c+1)^2/(2\epsilon^2)) \log(2/\delta)$, then with probability at least $1 - \delta$,

$$\Pr(\mathcal{E}) \geq 1 - \frac{1}{\phi^2} (\alpha + 2\epsilon). \quad (18)$$

Proof. Since $\widehat{\mathcal{L}}_{\mathbf{x}}(\widehat{\theta}_p) \leq \phi^2$ implies \mathcal{E} , it follows that

$$\Pr(\mathcal{E}) \geq \Pr\left(\widehat{\mathcal{L}}_{\mathbf{x}}(\widehat{\theta}_p) \leq \phi^2\right). \quad (19)$$

Then, by Markov inequality, we have

$$\Pr\left(\widehat{\mathcal{L}}_{\mathbf{x}}(\widehat{\theta}_p) > \phi^2\right) \leq \phi^{-2} \mathbb{E}\left[\widehat{\mathcal{L}}_{\mathbf{x}}(\widehat{\theta}_p)\right] = \phi^{-2} \mathcal{L}(\widehat{\theta}_p). \quad (20)$$

Subtracting both sides of Eq. (20) from a unit probability yields

$$\Pr\left(\widehat{\mathcal{L}}_{\mathbf{x}}(\widehat{\theta}_p) \leq \phi^2\right) \geq 1 - \phi^{-2} \mathcal{L}(\widehat{\theta}_p), \quad (21)$$

where the last equality follows because $\mathbb{E}[\widehat{\mathcal{L}}_{\mathbf{x}}(\widehat{\theta}_p)] = \mathcal{L}(\widehat{\theta}_p)$, which follows immediately from Definitions 1-2 and Assumption 1. Thus, plugging Eq. (21) into Eq. (19) yields

$$\Pr(\mathcal{E}) \geq 1 - \phi^{-2} \mathcal{L}(\widehat{\theta}_p). \quad (22)$$

Applying Lemma 2, we know that with probability $1 - \delta$, $\mathcal{L}(\widehat{\theta}_p) \leq \alpha + 2\epsilon$. Thus, plugging this into Eq. (22) yields

$$\Pr(\mathcal{E}) \geq 1 - \phi^{-2} (\alpha + 2\epsilon). \quad (23)$$

That is, by union bound, with probability at least $1 - \delta - \phi^{-2}(\alpha + 2\epsilon)$, the *poor model* yields the same prediction as that of the *rich model*. This completes our proof for Theorem 1.

This immediately implies \mathcal{E} will happen with probability at least $1 - \delta - (1/\phi^2)(\alpha + 2\epsilon)$. The chance for the *poor model* to yield the same prediction as the *rich model* on an arbitrary instance (i.e., knowledge infusion succeeds) is therefore at least $1 - \delta - (1/\phi^2)(\alpha + 2\epsilon)$.

5 EXPERIMENTS

5.1 Experimental Settings

Datasets. We use the following datasets in our evaluation.

A. MIMIC-III Critical Care Database (MIMIC-III)³ is collected from more than 58,000 ICU patients at the Beth Israel Deaconess Medical Center (BIDMC) from June 2001 to October 2012 [18]. We collect a subset of 9,488 patients who has one of the following (most frequent) diseases in their main diagnosis: (1) acute myocardial infarction, (2) chronic ischemic heart disease, (3) heart failure, (4) intracerebral hemorrhage, (5) specified procedures complications, (6) lung diseases, (7) endocardium diseases, and (8) septicemia. The task is disease diagnosis classification (i.e., predicting which of 8 diseases the patient has) based on features collected from 6 data channels: vital sign time series including Heart Rate (HR), Respiratory Rate (RR), Blood Pressure mean (Bpm), Blood Pressure systolic (BPs), Blood Pressure diastolic (BPd) and Blood Oxygen Saturation (SpO2). We randomly divided the data into training (80%), validation (10%) and testing (10%) sets.

B. PTB Diagnostic ECG Database (PTBDB)⁴ is a 15-channel 1000 Hz ECG time series including 12 conventional leads and 3 Frank leads [5, 12] collected from both healthy controls and cases of heart diseases, which amounts to a total number of 549 records. The given task is to classify ECG to one of the following categories: (1) myocardial infarction, (2) healthy control, (3) heart failure, (4) bundle branch block, (5) dysrhythmia, and (6) hypertrophy. We down-sampled the data to 200 Hz and pre-processed it following the "frame-by-frame" method [31] with sliding windows of 10-second duration and 5-second stepping between adjacent windows.

C. NEDC TUH EEG Artifact Corpus (EEG)⁵ is a 22-channel 500 Hz sensor time series collected from over 30,000 EEGs spanning the years from 2002 to present [26]. The task is to classify 5 types of EEG events including (1) eye movements (EYEM), (2) chewing (CHEW), (3) shivering (SHIV), (4) electrode pop, electrode static,

³<https://mimic.physionet.org/>

⁴<https://physionet.org/physiobank/database/ptbdb/>

⁵https://www.isip.piconepress.com/projects/tuh_eeeg/html/overview.shtml

and lead artifacts (ELPP), and (5) muscle artifacts (MUSC). We randomly divided the data into training (80%), validation (10%) and testing (10%) sets by records.

The statistics of the above datasets, as well as the architectures of the *rich* and *poor* models on each dataset are summarized in the tables below.

Table 2: Data statistics.

	MIMIC-III	PTBDB	EEG
# subjects	9,488	549	213
# classes	8	6	5
# features	6	15	22
Average length	48	108,596	13,007
Sample frequency	1 per hour	1,000 Hz	500 Hz

Table 3: The architecture of *rich* model in MIMIC-III, which includes a total of 51.6k parameters.

Layer	Type	Hyper-parameters	Activation
1	Split	n_seg=6	
2	Convolution1D	n_filter=64, kernel_size=4, stride=1	ReLU
3	Convolution1D	n_filter=64, kernel_size=4, stride=1	ReLU
4	AveragePooling1D		
5	LSTM	hidden_units=64	ReLU
6	PositionAttention		
7	Dense	hidden_units=n_classes	Linear
8	Softmax		

Table 4: The architecture of the infused, *poor* model used by CHEER, Direct, KD and AT for knowledge infusion in MIMIC-III, which includes a total of 3.5k parameters.

Layer	Type	Hyper-parameters	Activation
1	Split	n_seg=6	
2	Convolution1D	n_filter=16, kernel_size=4, stride=1	ReLU
3	Convolution1D	n_filter=16, kernel_size=4, stride=1	ReLU
4	AveragePooling1D		
5	LSTM	hidden_units=16	ReLU
6	PositionAttention		
7	Dense	hidden_units=n_classes	Linear
8	Softmax		

Baselines. We compare CHEER against the following baselines:

Direct: In all experiments, we train a neural network model parameterized with CHEER directly on the poor dataset without knowledge infusion from the *rich model*. The resulting model can be used to produce a lower bound of predictive performance on each dataset.

Knowledge Distilling (KD) [15]: KD transfers predictive power from teacher to student models via soft labels produced by the teacher model. In our experiments, all KD models have similar complexity as the infused model generated by CHEER. The degree of

Table 5: The architecture of *rich* model in PTBDB, which includes a total of 688.8k parameters.

Layer	Type	Hyper-parameters	Activation
1	Split	n_seg=10	
2	Convolution1D	n_filter=128, kernel_size=16, stride=2	ReLU
3	Convolution1D	n_filter=128, kernel_size=16, stride=2	ReLU
4	Convolution1D	n_filter=128, kernel_size=16, stride=2	ReLU
5	AveragePooling1D		
6	LSTM	hidden_units=128	ReLU
7	PositionAttention		
8	Dense	hidden_units=n_classes	Linear
9	Softmax		

Table 6: The architecture of the infused, *poor* model used by CHEER, Direct, KD and AT for knowledge infusion in PTBDB, which includes a total 45.0k parameters.

Layer	Type	Hyper-parameters	Activation
1	Split	n_seg=10	
2	Convolution1D	n_filter=32, kernel_size=16, stride=2	ReLU
3	Convolution1D	n_filter=32, kernel_size=16, stride=2	ReLU
4	Convolution1D	n_filter=32, kernel_size=16, stride=2	ReLU
5	AveragePooling1D		
6	LSTM	hidden_units=32	ReLU
7	PositionAttention		
8	Dense	hidden_units=n_classes	Linear
9	Softmax		

label softness (i.e., the temperature parameter of soft-max activation function) in KD is set to 5.

Table 7: The architecture of *rich* model in EEG, which includes a total of 417.4k parameters.

Layer	Type	Hyper-parameters	Activation
1	Split	n_seg=5	
2	Convolution1D	n_filter=128, kernel_size=8, stride=2	ReLU
3	Convolution1D	n_filter=128, kernel_size=8, stride=2	ReLU
4	Convolution1D	n_filter=128, kernel_size=8, stride=2	ReLU
5	AveragePooling1D		
6	LSTM	hidden_units=128	ReLU
7	PositionAttention		
8	Dense	hidden_units=n_classes	Linear
9	Softmax		

Table 8: The architecture of the infused, *poor* model used by CHEER, Direct, KD and AT for knowledge infusion in EEG, which includes a total of 51.6k parameters.

Layer	Type	Hyper-parameters	Activation
1	Split	n_seg=5	
2	Convolution1D	n_filter=32, kernel_size=8, stride=2	ReLU
3	Convolution1D	n_filter=32, kernel_size=8, stride=2	ReLU
4	Convolution1D	n_filter=32, kernel_size=8, stride=2	ReLU
5	AveragePooling1D		
6	LSTM	hidden_units=32	ReLU
7	PositionAttention		
8	Dense	hidden_units=n_classes	Linear
9	Softmax		

Attention Transfer (AT) [45]: AT enhances shallow neural networks by leveraging attention mechanism [3] to learn a similar attention behavior of a full-fledged deep neural network (DNN). In our experiments, we first train a DNN with attention component, which can be parameterized by CHEER. The trained attention component of DNN is then transferred to that of a shallow neural networks in poor-data environment via activation-based attention transfer with L_2 -normalization.

Heterogeneous Domain Adaptation (HDA) [43]: Maximize Mean Discrepancy (MMD) loss [13] has been successfully used in domain adaptation such as [22]. However, one drawback is that these works only consider homogeneous settings where the source and target domains have the same feature space, or use the same architecture of neural network. To mitigate this limitation, HDA [43] proposed modification of soft MMD loss to handle with heterogeneity between source domain and target domain.

Performance Metrics. The tested methods’ prediction performance was compared based on their corresponding areas under the Precision-Recall (PR-AUC) and Receiver Operating Characteristic curves (ROC-AUC) as well as the accuracy and F1 score, which are often used in multi-class classification to evaluate the tested method’s prediction quality. In particular, accuracy is measured by the ratio between the number of correctly classified instances and the total number of test instances. F1 score is the harmonic average of precision (the proportion of true positive cases among the predicted positive cases) and recall (the proportion of positive cases whose are correctly identified), with threshold 0.5 to determine whether a predictive probability for being positive is large enough (larger than threshold) to actually assign a positive label to the case being considered or not.

Then, we use the average of F1 scores evaluated for each label (i.e., macro-F1 score) to summarize the averaged predictive performance of all tested methods across all classes. The ROC-AUC and PR-AUC scores are computed based on predicted probabilities and ground-truths directly. For ROC-AUC, it is the area under the curve produced by points of true positive rate (TPR) and false positive rate (FPR) at various threshold settings. Likewise, the PR-AUC score is the area under the curve produced by points of (precision, recall) at various threshold settings. In our experiments, we report the average PR-AUC and ROC-AUC since all three tasks are multi-class classification.

Training Details For each method, the reported results (mean performance and its empirical standard deviation) are averaged over 20 independent runs. For each run, we randomly split the entire dataset into training (80%), validation (10%) and test sets (10%). All models are built using the training and validation sets and then, evaluated using test set. We use Adam optimizer [19] to train each model, with the default learning rate set to 0.001. The number of training epoches for each model is set as 200 and an early stopping criterion is invoked if the performance does not improve in 20 epoches. All models are implemented in Keras with Tensorflow backend and tested on a system equipped with 64GB RAM, 12 Intel Core i7-6850K 3.60GHz CPUs and Nvidia GeForce GTX 1080. For fair comparison,

we use the same model architecture and hyper-parameter setting for Direct, KD, AT, HDA and CHEER. For rich dataset, we use the entire amount of dataset with the entire set of data features. For poor dataset, we vary the size of paired dataset and the number of features to analyze the effect of knowledge infusion in different data settings as shown in Section 4.3. The default maximum amount of paired data is set to 50% of entire dataset, and the default number of data features used in the poor dataset is set to be half of the entire set of data features. In Section 4.2, to compare the tested methods’ knowledge infusion performance under different data settings, we use the default settings for all models (including CHEER and other baselines).

5.2 Performance Comparison

Results on MIMIC-III, PTBDB and EEG datasets are reported in Table 9, Table 10 and Table 11, respectively. In this experiment, we set the size of paired dataset to 50% of the size of the *rich data*, and set the number of features used in poor-data environment to 3, 7, 11 for MIMIC-III, PTBDB and EEG, respectively. In all datasets, it can be observed that the infused model generated by CHEER consistently achieves the best predictive performance among those of the tested methods, which demonstrates the advantage of our knowledge infusion framework over existing transfer methods such as KD and AT.

Notably, in terms of the macro-F1 scores, CHEER improves over KD, AT, HDA and Direct by 5.60%, 23.95%, 31.84% and 46.80%, respectively, on MIMIC-III dataset. The infused model generated by CHEER also achieves 81.69% performance of the *rich* model on PTBDB in terms of the macro-F1 score (i.e., 0.299/0.366, see Table 10) while adopting an architecture that is 15.14 times smaller than the *rich* model’s (see Tables 5 and 6). We have also performed a significance test to validate the significance of our reported improvement of CHEER over the baselines in Table 12.

Furthermore, it can also be observed that the performance variance of the infused model generated by CHEER (as reflected in the reported standard deviation) is the lowest among all tested methods’, which suggests that CHEER’s knowledge infusion is more robust. Our investigation in Section 5.3 further shows that this is the result of CHEER being able to perform both target and behavior infusion. This helps the infused model generated by CHEER achieved better and more stable performance than those of KD, HDA and AT, which either match the prediction target or reasoning behavior of the *rich* and *poor* models (but not both). This consequently leads to their less robust performance with wide fluctuation in different data settings, as demonstrated next in Section 5.3.

Table 9: Performance comparison on MIMIC-III dataset.

	ROC-AUC	PR-AUC	Accuracy	Macro-F1
Direct	0.622±0.062	0.208±0.044	0.821±0.012	0.141±0.057
KD	0.686±0.043	0.257±0.029	0.833±0.012	0.196±0.049
AT	0.645±0.064	0.225±0.044	0.826±0.013	0.167±0.057
HDA	0.655±0.034	0.225±0.029	0.824±0.011	0.157±0.038
CHEER	0.697±0.024	0.266±0.023	0.835±0.010	0.207±0.030
Rich Model	0.759±0.014	0.341±0.024	0.852±0.007	0.295±0.027

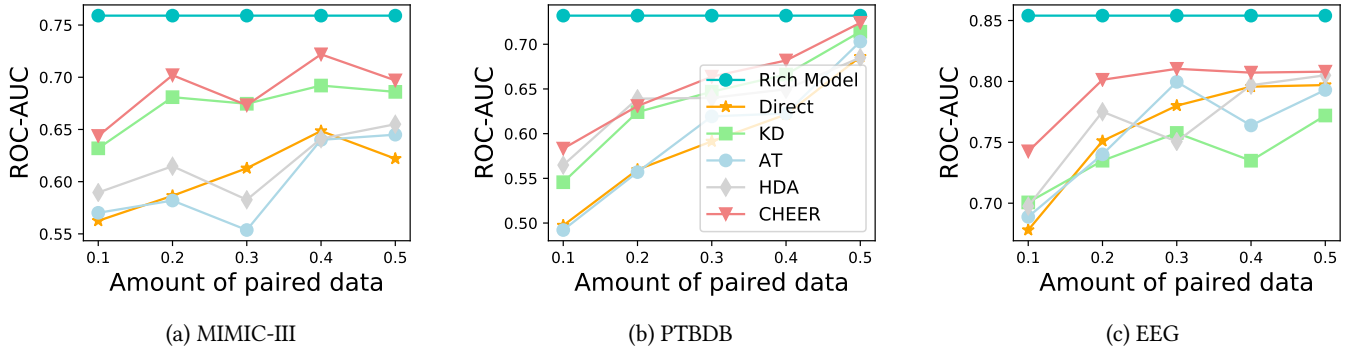


Figure 3: Graphs of achieved ROC-AUC scores on (a) MIMIC-III, (b) PTBDB and (c) EEG of the infused models generated by Direct, KD, AT, HDA and CHEER with different sizes of the paired datasets. The X-axis shows the ratio between the size of the paired dataset and that of the *rich* dataset.

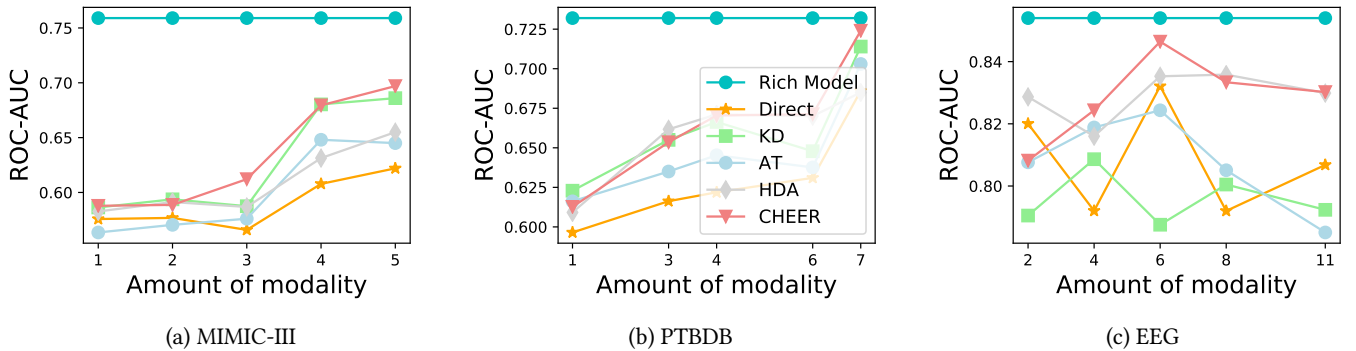


Figure 4: Graphs of achieved ROC-AUC scores on (a) MIMIC-III, (b) PTBDB and (c) EEG of the infused models generated by Direct, KD, AT, HDA and CHEER with different number of data channels (i.e., features) included in the poor dataset. Notice that all method use the same set of selected features for each run.

Table 10: Performance comparison on PTBDB dataset.

	ROC-AUC	PR-AUC	Accuracy	Macro-F1
Direct	0.686±0.114	0.404±0.088	0.920±0.015	0.275±0.057
KD	0.714±0.096	0.439±0.093	0.925±0.016	0.295±0.043
AT	0.703±0.117	0.402±0.078	0.921±0.016	0.283±0.056
HDA	0.685±0.113	0.430±0.080	0.924±0.011	0.299±0.051
CHEER	0.724±0.103	0.441±0.080	0.927±0.017	0.299±0.052
Rich Model	0.732±0.110	0.483±0.101	0.930±0.017	0.366±0.071

Table 11: Performance comparison on EEG dataset.

	ROC-AUC	PR-AUC	Accuracy	Macro-F1
Direct	0.797±0.064	0.506±0.083	0.888±0.015	0.425±0.078
KD	0.772±0.083	0.512±0.082	0.888±0.021	0.445±0.097
AT	0.793±0.071	0.502±0.082	0.884±0.012	0.417±0.062
HDA	0.805±0.050	0.523±0.073	0.884±0.019	0.455±0.073
CHEER	0.808±0.066	0.535±0.061	0.895±0.016	0.460±0.076
Rich Model	0.854±0.069	0.657±0.077	0.922±0.014	0.595±0.070

Table 12: The p -values of corresponding t -tests (on one-tail) for every two samples of ROC-AUC scores of CHEER and a tested benchmark (i.e., Direct, KD, AT and HDA) on MIMIC-III, PTBDB and EEG datasets, respectively. The corresponding significance percentage (s) is provided in the parentheses next to each reported p -value.

	MIMIC-III	PTBDB	EEG
with Direct	0.0000 ($s = 01\%$)	0.0154 ($s = 05\%$)	0.1720 ($s = 20\%$)
with KD	0.0450 ($s = 05\%$)	0.1874 ($s = 20\%$)	0.0124 ($s = 05\%$)
with AT	0.0007 ($s = 01\%$)	0.0421 ($s = 05\%$)	0.1821 ($s = 20\%$)
with HDA	0.0000 ($s = 01\%$)	0.0741 ($s = 10\%$)	0.4823 ($s = 20\%$)

5.3 Analyzing Knowledge Infusion Effect in Different Data Settings

To further analyze the advantages of CHEER’s knowledge infusion over those of the existing works (e.g., KD and AT), we perform additional experiments to examine how the variations in (1) sizes of the paired dataset and (2) the number of features of the *poor* dataset will affect the infused model’s performance. The results are shown

in Fig. 3 and Fig. 4, respectively. In particular, Fig. 3 shows how the ROC-AUC of the infused model generated by each tested method varies when we increase the ratio between the size of the paired dataset and that of the *rich* data. Fig. 4, on the other hand, shows how the infused model’s ROC-AUC varies when we increase the number of features of the *poor* dataset. In both settings, the reported performance of all methods is averaged over 10 independent runs.

Varying Paired Data. Fig. 3 shows that (a) CHEER outperforms all baselines with varying sizes of the paired data and (b) direct learning on *poor data* yields significantly worse performance across all settings. Both of which are consistent with our observations earlier on the superior knowledge infusion performance of CHEER. The infused models generated by KD, HDA and AT both perform consistently worse than that of CHEER by a substantial margin across all datasets. Their performance also fluctuates over a much wider range (especially on EEG data) than that of CHEER when we vary the size of the paired datasets. This shows that CHEER’s knowledge infusion is more data efficient and robust under different data settings.

On another note, we also notice that when the amount of paired data increases from 20% to 30% of the *rich* data, there is a performance drop that happens to all tested methods with attention transfer (i.e., CHEER and AT) on MIMIC-III but not on PTBDB and EEG. This is, however, not surprising since unlike PTBDB and EEG, MIMIC-III comprises of more heterogeneous types of signals and its data distribution is also more unbalanced, which affects the attention learning, and causes similar performance drop patterns between methods with attention transfer such as CHEER and AT.

Varying The Number of Features. Fig. 4 shows how the prediction performance of the infused models generated by tested methods changes as we vary the number of features in *poor data*. In particular, it can be observed that the performance of CHEER’s infused model on all datasets increases steadily as we increase the number of input features observed by the *poor model*, which is expected.

On the other hand, it is perhaps surprising that as the number of features increases, the performance of KD, HDA, AT and Direct fluctuates more widely on PTBDB and EEG datasets, which is in contrast to our observation of CHEER. This is, however, not unexpected since the informativeness of different features are different and hence, to utilize and combine them effectively, we need an accurate feature weighting/scoring mechanism. This is not possible in the cases of Direct, KD, HDA and AT because (a) Direct completely lacks knowledge infusion from the *rich model*, (b) KD and HDA only performs target transfer from the *rich to poor* model, and ignores the weighting/scoring mechanism, and (c) AT only transfers the scoring mechanism to the *poor* model (i.e., attention transfer) but not the feature aggregation mechanism, which is also necessary to combine the weighted features correctly. In contrast, CHEER transfers both the weighting/scoring (via behavior infusion) and feature aggregation (via target infusion) mechanisms, thus performs more robustly and is able to produce steady gain (without radical fluctuations) in term of performance when the number of features increases. This supports our observations earlier regarding

Table 13: CHEER’s performance on MIMIC-III, PTBDB and EEG with (left-column) K features with highest mutual information (MI) to the class label as features of the poor dataset; and (right-column) K features with lowest mutual information to the class label as features of the poor dataset. K is set to 2 for MIMIC-III, 5 for PTBDB and 7 for EEG.

Dataset	Highest MI Features	Lowest MI Features
MIMIC-III	0.672 ± 0.044	0.657 ± 0.012
PTBDB	0.646 ± 0.133	0.639 ± 0.115
EEG	0.815 ± 0.064	0.807 ± 0.042

the lowest performance variance achieved by the infused model of CHEER, which also suggests that CHEER’s knowledge infusion scheme is more robust than those of KD, HDA and AT.

Finally, to demonstrate how the performance of CHEER varies with different choices of feature sets for poor data, we computed the mutual information between each feature and the class label, and then ranked them in decreasing order. The performance of CHEER on all datasets is then reported in two cases, which include (a) K features with highest mutual information, and (b) K features with lowest mutual information. In particular, the reported results (see Table 13) show that a feature set with low mutual information to the class label will induce worse transfer performance and conversely, a feature set (with the same number of features) with high mutual information will likely improve the transfer performance.

To further inspect the effects of used modalities in CHEER, we also computed the averaged entropy of each modality across all classes, and ranked them in decreasing order for each dataset. Then, we selected a small number of top-ranked, middle-ranked and bottom-ranked features from the entire set of modalities. These are marked as Top, Middle and Bottom respectively in Table 14.

The number of selected features for each rank is 2, 4 and 5 for MIMIC-III, PTBDB and EEG, respectively. Finally, we report the ROC-AUC scores achieved by the corresponding infused models generated by CHEER for each of those feature settings in Table 14. It can be observed from this table that the ROC-AUC of the infused model degrades consistently across all datasets when we change the features of *poor* data from those in Top to Middle and then to Bottom. This verifies our statement earlier that the informativeness of different data features are different.

Table 14: CHEER’s performance using different sets of data features with different information quality (as measured by their entropy scores).

	MIMIC-III	PTBDB	EEG
Top	0.688 ± 0.010	0.710 ± 0.131	0.839 ± 0.044
Middle	0.676 ± 0.014	0.682 ± 0.132	0.788 ± 0.065
Bottom	0.664 ± 0.012	0.633 ± 0.130	0.758 ± 0.066
Rich	0.759 ± 0.014	0.732 ± 0.110	0.854 ± 0.069

6 CONCLUSION

This paper develops a knowledge infusion framework (named CHEER) that helps infuse knowledge acquired by a *rich model* trained on feature-rich data with a *poor model* which only has access to feature-poor data. The developed framework leverages a new model representation to re-parameterize the *rich model* and consequently, consolidate its learning behaviors into succinct summaries that can be infused efficiently with the *poor model* to improve its performance. To demonstrate the efficiency of CHEER, we evaluated CHEER on multiple real-world datasets, which show very promising results. We also develop a formal theoretical analysis to guarantee the performance of CHEER under practical assumptions. Future extensions of CHEER includes the following potential settings: incorporating meta/contextual information as part of the features and/or learning from data with missing labels.

7 ACKNOWLEDGMENTS

This work is part supported by National Science Foundation award IIS-1418511, CCF-1533768 and IIS-1838042, the National Institute of Health award NIH R01 1R01NS107291-01 and R56HL138415.

REFERENCES

[1] Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep?. In *Advances in neural information processing systems*. 2654–2662.

[2] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. 2014. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755* (2014).

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR* (2015).

[4] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain Separation Networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS’16)*. Curran Associates Inc., USA, 343–351. <http://dl.acm.org/citation.cfm?id=3157096>. 3157135

[5] R Bousselot, D Kreiseler, and A Schnabel. 1995. Nutzung der EKG-Signaldatenbank CARDIODAT der PTB über das Internet. *Biomedizinische Technik/Biomedical Engineering* 40, s1 (1995), 317–318.

[6] Minmin Chen, Zhixiang Xu, Kilian Q. Weinberger, and Fei Sha. 2012. Marginalized Denoising Autoencoders for Domain Adaptation. In *Proceedings of the 29th International Conference on Machine Learning (ICML’12)*. Omnipress, USA, 1627–1634. <http://dl.acm.org/citation.cfm?id=3042573.3042781>

[7] Keunwoo Choi, George Fazekas, Mark Sandler, and Kyunghyun Cho. 2016. Convolutional recurrent neural networks for music classification. *arXiv preprint arXiv:1609.04243* (2016).

[8] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Advances in neural information processing systems*. 577–585.

[9] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial Training of Neural Networks. *J. Mach. Learn. Res.* 17, 1 (Jan. 2016), 2096–2030. <http://dl.acm.org/citation.cfm?id=2946645.2946704>

[10] Muhammad Ghifary, David Balduzzi, W. Bastiaan Kleijn, and Mengjie Zhang. 2017. Scatter Component Analysis: A Unified Framework for Domain Adaptation and Domain Generalization. *IEEE Trans. Pattern Anal. Mach. Intell.* 39, 7 (2017), 1414–1430. <https://doi.org/10.1109/TPAMI.2016.2599532>

[11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain Adaptation for Large-scale Sentiment Classification: A Deep Learning Approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML’11)*. Omnipress, USA, 513–520. <http://dl.acm.org/citation.cfm?id=3104482.3104547>

[12] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. 2000. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. *Circulation* 101, 23 (2000), e215–e220.

[13] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander J. Smola. 2006. A Kernel Method for the Two-Sample-Problem. In *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British*

Columbia, Canada, December 4-7, 2006. 513–520. <http://papers.nips.cc/paper/3110-a-kernel-method-for-the-two-sample-problem>

[14] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. 1693–1701.

[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[16] Sheng-Huang Jia, Jia-Wei Zhao, and Zhao-Yang Liu. 2018. Cost-Effective Training of Deep CNNs with Active Model Adaptation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*. 1580–1588. <https://doi.org/10.1145/3219819.3220026>

[17] Wenhao Jiang, Hongchang Gao, Wei Lu, Wei Liu, Fu-Lai Chung, and Heng Huang. 2019. Stacked Robust Adaptively Regularized Auto-Regressions for Domain Adaptation. *IEEE Trans. Knowl. Data Eng.* 31, 3 (2019), 561–574. <https://doi.org/10.1109/TKDE.2018.2837085>

[18] 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).

[19] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[20] Wen Li, Zheng Xu, Dong Xu, Dengxin Dai, and Luc Van Gool. 2018. Domain Generalization and Adaptation Using Low Rank Exemplar SVMs. *IEEE Trans. Pattern Anal. Mach. Intell.* 40, 5 (2018), 1114–1127. <https://doi.org/10.1109/TPAMI.2017.2704624>

[21] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *ICLR* (2017).

[22] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. 2015. Learning Transferable Features with Deep Adaptation Networks. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. 97–105. <http://proceedings.mlr.press/v37/long15.html>

[23] David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. 2015. Unifying distillation and privileged information. *ICLR* (2015).

[24] Chen Luo, Zhengzhang Chen, Lu An Tang, Anshumali Shrivastava, Zhichun Li, Haifeng Chen, and Jieping Ye. 2018. TINET: Learning Invariant Networks via Knowledge Transfer. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*. 1890–1899. <https://doi.org/10.1145/3219819.3220003>

[25] Yong Luo, Yonggang Wen, Tongliang Liu, and Dacheng Tao. 2019. Transferring Knowledge Fragments for Learning Distance Metric from a Heterogeneous Domain. *IEEE Trans. Pattern Anal. Mach. Intell.* 41, 4 (2019), 1013–1026. <https://doi.org/10.1109/TPAMI.2018.2824309>

[26] Iyad Obeid and Joseph Picone. 2016. The temple university hospital eeg data corpus. *Frontiers in neuroscience* 10 (2016), 196.

[27] Sinno Jialin Pan, Ivor W. Tsang, James T. Kwok, and Qiang Yang. 2009. Domain Adaptation via Transfer Component Analysis. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*. 1187–1192. <http://ijcai.org/Proceedings/09/Papers/200.pdf>

[28] Sinno Jialin Pan, Ivor W. Tsang, James T. Kwok, and Qiang Yang. 2011. Domain Adaptation via Transfer Component Analysis. *IEEE Trans. Neural Networks* 22, 2 (2011), 199–210. <https://doi.org/10.1109/TNN.2010.2091281>

[29] Peixi Peng, Yonghong Tian, Tao Xiang, Yaowei Wang, Massimiliano Pontil, and Tiejun Huang. 2018. Joint Semantic and Latent Attribute Modelling for Cross-Class Transfer Learning. *IEEE Trans. Pattern Anal. Mach. Intell.* 40, 7 (2018), 1625–1638. <https://doi.org/10.1109/TPAMI.2017.2723882>

[30] Ilija Radosavovic, Piotr Dollár, Ross Girshick, Georgia Gkioxari, and Kaiming He. 2017. Data distillation: Towards omni-supervised learning. *arXiv preprint arXiv:1712.04440* (2017).

[31] Attila Reiss and Didier Stricker. 2012. Creating and benchmarking a new dataset for physical activity monitoring. In *Proceedings of the 5th International Conference on Pervasive Technologies Related to Assistive Environments*. ACM, 40.

[32] Artem Rozantsev, Mathieu Salzmann, and Pascal Fua. 2019. Beyond Sharing Weights for Deep Domain Adaptation. *IEEE Trans. Pattern Anal. Mach. Intell.* 41, 4 (2019), 801–814. <https://doi.org/10.1109/TPAMI.2018.2814042>

[33] Hojjat Salehinejad, Joseph Barfett, Shahrokh Valaei, and Timothy Dowdell. 2018. Training Neural Networks with Very Little Data - A Draft. (2018).

[34] Bharat Bhusan Sau and Vineeth N Balasubramanian. 2016. Deep model compression: Distilling knowledge from noisy teachers. *arXiv preprint arXiv:1610.09650* (2016).

[35] Noam Segev, Maayan Harel, Shie Mannor, Koby Crammer, and Ran El-Yaniv. 2017. Learn on Source, Refine on Target: A Model Transfer Learning Framework with Random Forests. *IEEE Trans. Pattern Anal. Mach. Intell.* 39, 9 (2017), 1811–1824. <https://doi.org/10.1109/TPAMI.2016.2618118>

[36] Yuxing Tang, Josiah Wang, Xiaofang Wang, Boyang Gao, Emmanuel Dellandrea, Robert J. Gaizauskas, and Liming Chen. 2018. Visual and Semantic Knowledge Transfer for Large Scale Semi-Supervised Object Detection. *IEEE Trans. Pattern*

- Anal. Mach. Intell.* 40, 12 (2018), 3045–3058. <https://doi.org/10.1109/TPAMI.2017.2771779>
- [37] Bo Wang, Minghui Qiu, Xisen Wang, Yaliang Li, Yu Gong, Xiaoyi Zeng, Jun Huang, Bo Zheng, Deng Cai, and Jingren Zhou. 2019. A Minimax Game for Instance based Selective Transfer Learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4–8, 2019*. 34–43. <https://doi.org/10.1145/3292500.3330841>
- [38] Pengfei Wei, Yiping Ke, and Chi Keong Goh. 2019. A General Domain Specific Feature Transfer Framework for Hybrid Domain Adaptation. *IEEE Trans. Knowl. Data Eng.* 31, 8 (2019), 1440–1451. <https://doi.org/10.1109/TKDE.2018.2864732>
- [39] Qingyao Wu, Hanrui Wu, Xiaoming Zhou, Mingkui Tan, Yonghui Xu, Yuguang Yan, and Tianyong Hao. 2017. Online Transfer Learning with Multiple Homogeneous or Heterogeneous Sources. *IEEE Trans. Knowl. Data Eng.* 29, 7 (2017), 1494–1507. <https://doi.org/10.1109/TKDE.2017.2685597>
- [40] Cao Xiao, Edward Choi, and Jimeng Sun. 2018. Opportunities and challenges in developing deep learning models using electronic health records data: a systematic review. *Journal of the American Medical Informatics Association* (2018).
- [41] Yonghui Xu, Sinno Jialin Pan, Hui Xiong, Qingyao Wu, Ronghua Luo, Huaqing Min, and Hengjie Song. 2017. A Unified Framework for Metric Transfer Learning. *IEEE Trans. Knowl. Data Eng.* 29, 6 (2017), 1158–1171. <https://doi.org/10.1109/TKDE.2017.2669193>
- [42] Zhe Xu, Shaoli Huang, Ya Zhang, and Dacheng Tao. 2018. Webly-Supervised Fine-Grained Visual Categorization via Deep Domain Adaptation. *IEEE Trans. Pattern Anal. Mach. Intell.* 40, 5 (2018), 1100–1113. <https://doi.org/10.1109/TPAMI.2016.2637331>
- [43] Yuan Yao, Yu Zhang, Xutao Li, and Yunming Ye. 2019. Heterogeneous Domain Adaptation via Soft Transfer Network. In *ACM Multimedia*.
- [44] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. 2017. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2.
- [45] Sergey Zagoruyko and Nikos Komodakis. 2017. Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer. In *ICLR*. <https://arxiv.org/abs/1612.03928>
- [46] Guangyou Zhou, Zhiwen Xie, Xiangji Huang, and Tingting He. 2016. Bi-Transferring Deep Neural Networks for Domain Adaptation. In *ACL*.