Journal of Healthcare Informatics Research (2020) 4:112–137
https://doi.org/10.1007/s41666-019-00062-3

RESEARCH ARTICLE

Transfer Learning for Clinical Time Series Analysis Using Deep Neural Networks

Priyanka Gupta[1] · Pankaj Malhotra[1] · Jyoti Narwariya[1] · Lovekesh Vig[1] · Gautam Shroff[1]

Received: 8 December 2018 / Revised: 24 October 2019 / Accepted: 12 November 2019 /
Published online: 13 December 2019

1 Introduction

Electronic health records (EHR) consisting of the medical history of patients are useful in various clinical applications such as diagnosis and recommending medicine [22]. Traditional machine learning approaches often require careful domain-specific feature engineering to achieve good prediction performance. On the other hand, deep learning approaches enable end-to-end learning without the need of hand-crafted and domain-specific features, and have recently produced promising results for various clinical prediction tasks [17, 22, 29]. As a result, there has been a rapid growth in the applications of deep learning to various clinical prediction tasks from electronic health records, e.g., Doctor AI [6] for medical diagnosis, Deep Patient [21] to predict future diseases in patients, and DeepR [23] to predict unplanned readmission after discharge. With various medical parameters being recorded over a period of time in EHR databases, recurrent neural networks (RNNs) can be an effective way to model the sequential aspects of EHR data and, in turn, enable applications in diagnoses [3, 6, 17], mortality prediction, and estimating length of stay [9, 27, 28].

✉ Priyanka Gupta
priyanka.g35@tcs.com

✉ Pankaj Malhotra
malhotra.pankaj@tcs.com

Jyoti Narwariya
jyoti.narwariya@tcs.com

Lovekesh Vig
lovekesh.vig@tcs.com

Gautam Shroff
gautam.shroff@tcs.com

1 TCS Research, New Delhi, India

However, like most deep learning approaches, RNNs are prone to overfitting when labeled training data is scarce, and often require careful and computationally expensive hyper-parameter tuning effort. Transfer learning [2, 24] is known to mitigate this: It enables knowledge transfer from neural networks trained on a *source* task (domain) with sufficient training instances to a related *target* task (domain) with few training instances. For example, training a deep network on a diverse set of images can provide useful features for images from unseen domains [31]. Moreover, fine-tuning a pre-trained network for the target task is often faster and easier than constructing and training a new network from scratch [2, 19].

It has been shown that pre-trained networks can learn to extract a rich set of generic features that can then be applied to a wide range of other similar tasks [19]. Also, it has been argued that transferring weights even from distant tasks can be better than using random initial weights in neural networks [35]. Transfer learning via fine-tuning parameters of pre-trained models for end tasks has been recently considered for medical applications as well [6, 16]. However, fine-tuning a large number of parameters with a small-labeled dataset may still result in overfitting, and requires careful regularization (as we show in Section 9 through empirical evaluation). In this work, we consider two simple yet effective approaches to transfer the knowledge captured in pre-trained deep RNNs for new target tasks in healthcare domain. More specifically, we consider two scenarios: (i) extract features from a pre-trained network and use them to build models for target task and (ii) initialize deep network for target task using parameters of a pre-trained network and then fine-tune using labeled training data for target task.

The key contributions of this work are:

– We propose two approaches for transfer learning for classification tasks such as patient phenotyping and mortality prediction given multivariate time series corresponding to physiological parameters of patients[1]. For a target task in the healthcare domain, we consider (i) a domain adaptation approach using a general-purpose off-the-shelf time series feature extractor based on deep RNN, to remove the effort and resources required to train a deep network from scratch while still leveraging its advantages, and (ii) a task adaptation approach based on a deep RNN trained using labeled data from another set of (source) tasks in healthcare domain to reduce the dependence on labeled training data for the target task and also reduce hyper-parameter tuning effort.
– Our proposed approaches allow extracting robust features from variable length multivariate time series by using pre-trained deep RNNs, thereby reducing dependence on expert domain-driven feature extraction.
– We show that carefully regularized fine-tuning of pre-trained RNNs leads to models that are significantly more robust to training data size in comparison with task-specific classification models trained from scratch.
– We also study the trade-off between domain adaptation and task adaptation with respect to the amount of labeled data from the healthcare domain, and show that

[1]This work contrasts and extends our previous work in [8] and [7].

leveraging pre-trained models from other (seemingly unrelated) domains can be useful in scenarios where task adaptation may be ineffective, e.g., when labeled data for the target task as well as its related tasks within the healthcare domain is scarce.

Through empirical evaluation of patient phenotyping and mortality prediction tasks on MIMIC-III benchmark dataset [13], we demonstrate that our transfer learning approaches yield data- and compute-efficient classification models that require little training or fine-tuning effort while yielding classification performance that is comparable with models with hand-crafted features or carefully trained domain-specific deep networks benchmarked in [9, 10, 32].

The rest of the paper is organized as follows: In Section 2, we present related work, and provide details of an existing off-the-shelf pre-trained deep RNN, namely TimeNet [19], in Section 3. We provide an overview of the proposed domain and task adaptation approaches in Section 4, followed by their details in Sections 5 and 6, respectively. In Section 7, we provide details of cohort selection and datasets. We provide experimental details and observations in Sections 8 and 9, respectively, and comparison of two approaches in Section 10, and finally conclude in Section 11.

2 Related Work

TimeNet-based features have been shown to be useful for various tasks including ECG classification [19]. In this work, we consider application of TimeNet to phenotyping and in-hospital mortality tasks for multivariate clinical time series classification. Deep Patient [21] proposes leveraging features from a pre-trained stacked-autoencoder for EHR data. However, it does not leverage the temporal aspect of the data and uses a non-temporal model based on stacked-autoencoders. Our domain adaptation approach extracts temporal features via TimeNet while incorporating the important sequential nature of EHR data. Doctor AI [6] uses discretized medical codes (e.g., diagnosis, medication, procedure) from longitudinal patient visits via a purely supervised setting while we use real-valued time series. While approaches like Doctor AI require training a deep RNN from scratch, our approach leverages a general-purpose RNN for feature extraction.

Harutyunyan et al. [9] consider training a deep RNN model under multitask learning setting (for multiple prediction tasks including phenotyping and in-hospital mortality) to learn a general-purpose deep RNN for clinical time series. They show that it is possible to train a single network for multiple tasks simultaneously by capturing generic features that work across different tasks. We also consider leveraging generic features for clinical time series but under the transfer learning setting where we pre-train an RNN on different domains or set of different tasks. This pre-training allows our approach to be more data and resource efficient.

Unsupervised pre-training has been shown to be effective in capturing the generic patterns and distribution from EHR data [21]. Further, RNNs for time series classification from EHR data have been successfully explored, e.g., in [3, 17]. However, these approaches do not address the challenge posed by limited labeled data, which is the focus of this work. Transfer learning using deep neural networks has been recently explored for medical applications: A model learned from one hospital could be adapted to another hospital for the same task via RNNs [6]. A deep neural network was used to transfer knowledge from one dataset to another while the source and target tasks (named-entity recognition from medical records) are the same in [16]. However, in both these transfer learning approaches, the source and target tasks are the same while only the dataset changes. In contrast, our task adaptation approach allows to transfer the model trained on several healthcare-specific tasks to a different (although related) classification task using RNNs for clinical time series.

3 Background: TimeNet

Deep (multi-layered) RNNs have been shown to perform hierarchical processing of time series with different layers tackling different time scales [11, 18]. TimeNet [19] is a general-purpose multi-layered RNN trained on large number of diverse univariate time series from UCR Time Series Archive [4] that has been shown to be useful as off-the-shelf feature extractor for time series. TimeNet has been trained on 18 different datasets simultaneously via an RNN autoencoder in an unsupervised manner for reconstruction task. Features extracted from TimeNet have been found to be useful for classification task on 30 datasets from various domains not seen during training of TimeNet, proving its ability to provide meaningful features for unseen datasets.

TimeNet contains three recurrent layers having 60 gated recurrent units (GRUs) [5] each. TimeNet is an RNN trained via an autoencoder consisting of an encoder RNN and a decoder RNN trained simultaneously using the sequence-to-sequence learning framework [1, 33] as shown in Fig. 1. RNN autoencoder is trained to obtain the parameters $\mathbf{W}_E$ of the encoder RNN f_E via reconstruction task such that for input $x_{1...\tau} = x_1, x_2, ..., x_\tau$ $(x_i \in \mathbb{R})$, the target output time series $x_{\tau...1} = x_\tau, x_{\tau-1}, ..., x_1$ is reverse of the input.

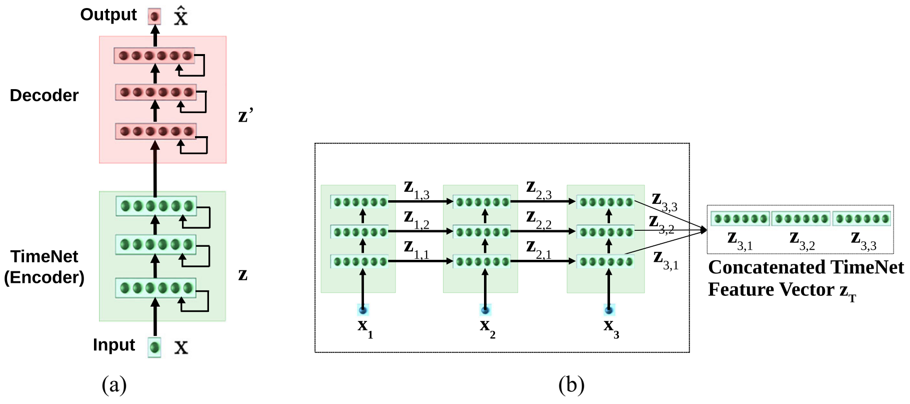


Fig. 1 **a** TimeNet trained via RNN Encoder-Decoder with three hidden GRU layers. **b** TimeNet-based feature extraction. TimeNet is shown unrolled for L = 3

The RNN encoder f_E provides a non-linear mapping of the univariate input time series to a fixed-dimensional vector representation $\mathbf{z}_\tau : \mathbf{z}_\tau = f_E(x_{1\dots\tau}; \mathbf{W}_E)$, followed by an RNN decoder f_D based non-linear mapping of $\mathbf{z}_\tau$ to univariate time series: $\hat{x}_{\tau\dots1} = f_D(\mathbf{z}_\tau; \mathbf{W}_D)$; where $\mathbf{W}_E$ and $\mathbf{W}_D$ are the parameters of the encoder and decoder, respectively. The model is trained to minimize the average squared reconstruction error. Training on 18 diverse datasets simultaneously results in robust time series features getting captured in $\mathbf{z}_\tau$: the decoder relies on $\mathbf{z}_\tau$ as the only input to reconstruct the time series, forcing the encoder to capture all the relevant information in the time series into the fixed-dimensional vector $\mathbf{z}_\tau$. This vector $\mathbf{z}_\tau$ is used as the feature vector for input $x_{1\dots\tau}$. This feature vector is then used to train a simpler classifier (e.g., SVM, as used in [19]) for the end task. TimeNet maps a univariate input time series to 180-dimensional feature vector, where each dimension corresponds to final output of one of the 60 GRUs in the 3 recurrent layers.

4 Approach Overview

Consider sets $\mathcal{D}_S$ and $\mathcal{D}_T$ of time series instances corresponding to a source (S) and a target (T) dataset, respectively. $\mathcal{D}_S = \{(\mathbf{x}_S^{(i)}, \mathbf{y}_S^{(i)})\}_{i=1}^{N_S}$, where N_S is the number of time series instances in the source dataset. Denoting time series $\mathbf{x}_S^{(i)}$ by $\mathbf{x}$ and the corresponding target label $\mathbf{y}_S^{(i)}$ by $\mathbf{y}$ for simplicity of notation, we have $\mathbf{x} = \mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_\tau$ denote a time series of length τ, where $\mathbf{x}_t \in \mathbb{R}^n$ $(t = 1 \dots \tau)$ is an n-dimensional vector corresponding to n parameters. Further, $\mathbf{y} = [y_1, \dots, y_K] \in \{0, 1\}^K$, where K is the number of binary classification tasks. Similarly, $\mathcal{D}_T = \{(\mathbf{x}_T^{(i)}, y_T^{(i)})\}_{i=1}^{N_T}$ such that $N_T \ll N_S$, and $y_T^{(i)} \in \{0, 1\}$ such that the target task is a binary classification task. We consider $\mathcal{D}_S$ and $\mathcal{D}_T$ to be from same (different) *domain* if the n parameters in $\mathcal{D}_S$ and $\mathcal{D}_T$ are the same (different). Further, we consider the *tasks* for $\mathcal{D}_S$ and $\mathcal{D}_T$ to be the same if number of target classes in $\mathbf{y}_S$ and $\mathbf{y}_T$ is same and the corresponding classes are semantically same, for example, both $\mathbf{y}_S$ and $\mathbf{y}_T$ contain two classes {patient survives, patient dies}.

We consider two scenarios for transfer learning using RNNs:

i) *Domain adaptation* where $\mathcal{D}_S$ contains time series from various domains such as electric devices, motion capture, spectrographs, sensor readings, ECGs, and simulated time series, taken from publicly available UCR Time Series Classification Archive [4], and $\mathcal{D}_T$ contains clinical time series from EHR database (i.e., sets $\mathcal{D}_S$ and $\mathcal{D}_T$ are from different domains). We consider pre-training an RNN using $\mathcal{D}_S$ via unsupervised learning and study its ability to compute useful features for time series from an unseen domain (healthcare in our case). We adapt pre-trained model using $\mathcal{D}_T$ via supervised learning (Note: As we adapt model trained via unsupervised learning for supervised task, set $\mathcal{D}_T$ and $\mathcal{D}_S$ are from different task);
ii) *Task adaptation* where both $\mathcal{D}_S$ and $\mathcal{D}_T$ contain time series from the healthcare domain such that the parameters in $\mathbf{x}_S$ and $\mathbf{x}_T$ correspond to the same n physiological parameters such as heart rate, pulse rate, and oxygen saturation. Further,

$\mathbf{y}_S$ corresponds to various tasks, such as presence/absence of phenotypes, e.g., acute cerebrovascular disease, diabetes mellitus with complications, and gastrointestinal hemorrhage, and $\mathbf{y}_T$ corresponds to a related but different task, e.g., present/absence of new phenotypes that are not present in $\mathbf{y}_S$. We consider pre-training an RNN model using $\mathcal{D}_S$ via supervised learning on the diverse set of tasks in $\mathbf{y}_S$ such that the model learns to capture and extract a rich set of generic features from clinical time series that can be useful for other tasks in the same domain. We adapt the pre-trained model using $\mathcal{D}_T$ via supervised learning.

5 Domain Adaptation: Adapting Universal Time Series Feature Extractors to Healthcare Domain

General-purpose time series feature extractors such as TimeNet [19] and Universal Encoder [30] usually constrain the input time series to be univariate as it is difficult to cater to multivariate time series with varying dimensionality in a single neural network. In this scenario, we consider adapting TimeNet to healthcare domain with two key considerations[2]: (i) extend TimeNet for multivariate clinical time series classification tasks that requires simultaneous consideration of various physiological parameters and (ii) adapt the features from TimeNet for specific tasks from healthcare such as patient phenotyping and mortality prediction tasks. We show how TimeNet can be adapted to these classification tasks by training computationally efficient traditional linear classifiers on top of features extracted for each parameter using TimeNet, as depicted in Fig. 2. Further, we propose a simple mechanism to leverage the weights of the trained linear classifier to provide insights into the relevance of each raw input feature (physiological parameter) for a given phenotype (described in Section 5.3).

Consider $\mathcal{D}_T$ is set of labeled time series instances from an EHR database: $\mathcal{D}_T = \{(\mathbf{x}_T^{(i)}, y_T^{(i)})\}_{i=1}^{N_T}$, where $\mathbf{x}_T^{(i)}$ is a multivariate time series, $y_T^{(i)} \in \{0, 1\}$ such that the target task is a binary classification task, N_T is the number of time series instances corresponding to patients. We consider the presence or absence of a phenotype as a binary classification task and learn an independent model for each phenotype (unlike [9] which consider phenotyping as a multi-label classification problem). This allows us to build simple and compute-efficient linear binary classification models as described next. In practice, the outputs of these binary classifiers can then be considered together to estimate the set of phenotypes present in a patient. Similarly, mortality prediction is considered to be a binary classification task where the goal is to classify whether the patient will survive (after admission to ICU) or not.

5.1 Feature Extraction for Multivariate Clinical Time Series

For a multivariate time series $\mathbf{x} = \mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_\tau$, where $\mathbf{x}_t \in \mathbb{R}^n$, we consider time series for each of the n raw input features (physiological parameters, e.g.,

[2]It is to be noted that we take TimeNet as an example to illustrate our proposed domain-adaptation approach, but the proposed approach is generic and can be used to adapt any universal time series feature extractor to healthcare domain.

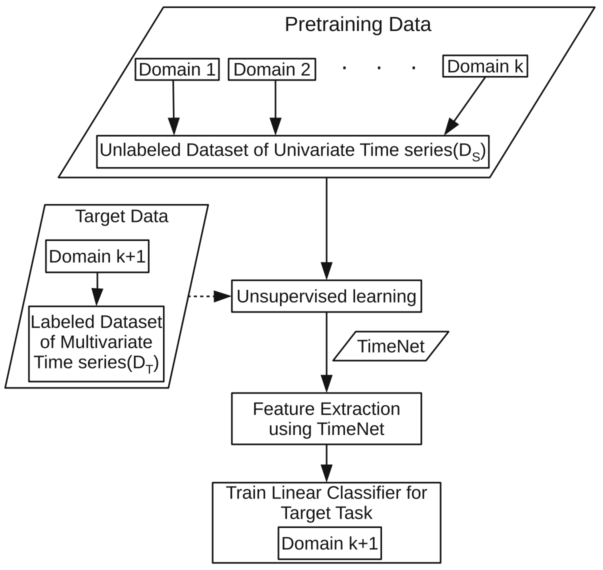

Pretraining Data
Domain 1
Domain 2
Domain k
Unlabeled Dataset of Univariate Time series(D_S)
Target Data
Domain k+1
Labeled Dataset of Multivariate Time series(D_T)
Unsupervised learning
TimeNet
Feature Extraction using TimeNet
Train Linear Classifier for Target Task
Domain k+1

Fig. 2 Domain adaptation scenario. TimeNet is pre-trained in an unsupervised manner on time series from k diverse domains, and then used to extract features from time series in $(k + 1)$-th domain for subsequent target classification task

glucose level and heart rate) independently, to obtain univariate time series $x_j = x_{j1}, x_{j2}, \ldots x_{j\tau}$, where $j = 1 \ldots n$. (Note: We use $\mathbf{x}$ instead of $\mathbf{x}^{(i)}$ and omit superscript (i) for ease of notation). We obtain the vector representation $\mathbf{z}_{j\tau} = f_E(x_j; \mathbf{W}_E)$ for x_j using TimeNet, where $\mathbf{z}_{j\tau} \in \mathbb{R}^c$ with $c = 180$ (as described in Section 3). We concatenate the TimeNet-features $\mathbf{z}_{j\tau}$ for each raw input feature j to get the final feature vector $\mathbf{z}_\tau = [\mathbf{z}_{1\tau}, \mathbf{z}_{2\tau}, \ldots, \mathbf{z}_{n\tau}]$ for time series $\mathbf{x}$, where $\mathbf{z}_\tau \in \mathbb{R}^m$, $m = n \times c$ as illustrated in Fig. 3.

In general, time series length τ varies across instances, i.e., depends on i (e.g., based on length of stay in the hospital). We assume equal length for each time series

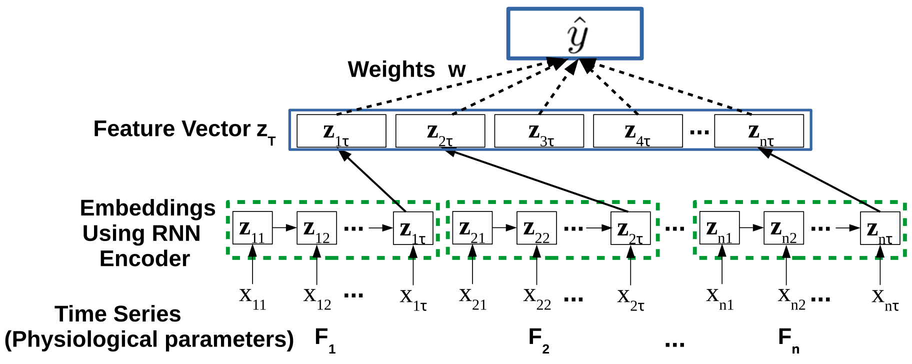

ŷ
Weights w
Feature Vector z_τ
z_1τ
z_2τ
z_3τ
z_4τ
z_nτ
Embeddings Using RNN Encoder
z_11 z_12 z_1τ z_21 z_22 z_2τ z_n1 z_n2 z_nτ
x_11 x_12 x_1τ x_21 x_22 x_2τ x_n1 x_n2 x_nτ
Time Series (Physiological parameters)
F_1
F_2
F_n

Fig. 3 Domain adaptation. TimeNet-based feature extraction and classification

for sake of clarity without loss of generality. In practice, we convert each time series to have equal length τ by suitable pre/post-padding with 0s.

5.2 Using TimeNet-Based Features for Classification

The final concatenated feature vector $\mathbf{z}_\tau$ is used as input to the phenotyping and mortality prediction classification models. We consider a linear mapping from input TimeNet features $\mathbf{z}_\tau$ to the target label y s.t. the estimate $\hat{y} = \mathbf{w} \cdot \mathbf{z}_\tau$, where $\mathbf{w} \in \mathbb{R}^m$. We note that since $c = 180$ is large, $\mathbf{z}_\tau$ has large number of features $m \geq 180$. We, therefore, constrain the linear model with weights $\mathbf{w}$ to use only a few of these large number of features. The weights are obtained using LASSO-regularized loss function [34]:

$$\arg\min_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^{N} (y^{(i)} - \mathbf{w} \cdot \mathbf{z}_\tau^{(i)})^2 + \alpha ||\mathbf{w}||_1 \tag{1}$$

where $y^{(i)} \in \{0, 1\}$, $||\mathbf{w}||_1 = \sum_{j=1}^{n} \sum_{k=1}^{c} |w_{jk}|$ is the L_1-norm, w_{jk} represents the weight assigned to the k-th TimeNet feature for the j-th raw feature, and α controls the extent of sparsity with higher α implying more sparsity, i.e., fewer TimeNet features are implicitly used to arrive at the final classification decision.

5.3 Obtaining Relevance Scores for Raw Features

Determining relevance of the n raw input features for a given phenotype is potentially useful to obtain insights into the obtained classification model. The learned weights $\mathbf{w}$ are easy to interpret and can give interesting insights into relevant features for a classification task (e.g., as used in [20]). We obtain the relevance r_j of the j-th raw input feature as the sum of the absolute values of the weights w_{jk} assigned to the corresponding TimeNet features $\mathbf{z}_{j\tau}$ as shown in Fig. 4, s.t.

$$r_j = \sum_{k=1}^{c} |w_{jk}|, \ \ j = 1 \ldots n. \tag{2}$$

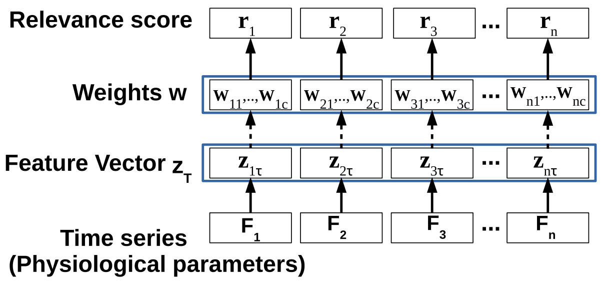


Fig. 4 Obtaining relevance scores for raw input features. Here, τ time series length, n number of raw input features

Further, r_j is normalized using min-max normalization to obtain $r'_j = \frac{r_j - r_{\min}}{r_{\max} - r_{min}} \in [0, 1]$; where $r_{\min} = \min(r_1, \ldots, r_n)$ and $r_{\max} = \max(r_1, \ldots, r_n)$. In practice, these normalized relevance scores for the raw features help to interpret and validate the overall model. For example, one would expect blood glucose level raw input feature to have a high relevance score in the classification model learned to detect diabetes mellitus phenotype (we provide such insights later in Section 8).

6 Task Adaptation: Adapting Healthcare-Specific Pre-trained Models to a New Task

As depicted in Fig. 5, the goal of task adaptation is to transfer the learning from a set of source tasks to a related target task for clinical time series by means of an RNN. Considering phenotype detection from time series of physiological parameters as a binary classification task, we train HealthNet as an RNN classifier on a diverse set of such binary classification tasks simultaneously (one task per phenotype) using a large labeled dataset. We consider the following approaches to adapt it to an unseen target task:

1. Initialize parameters of the target task-specific RNN using the parameters of HealthNet previously trained on a large number of source tasks; so that Health-Net provides good initialization of parameters of task-specific RNN and train the model (described in Section 6.2).

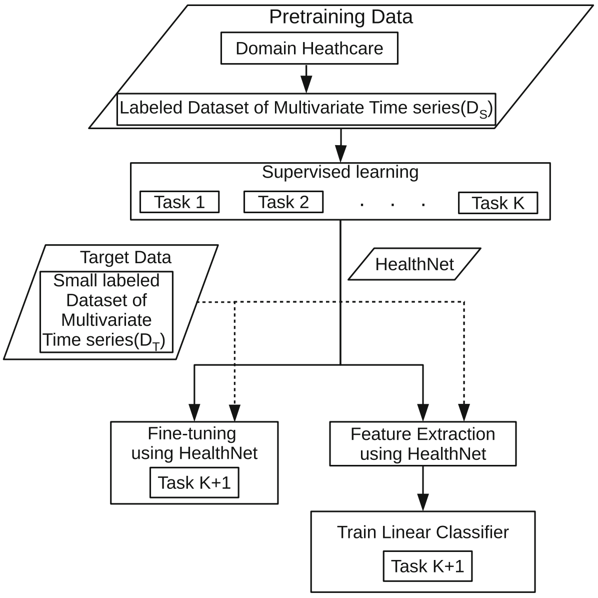

Pretraining Data
Domain Heathcare
Labeled Dataset of Multivariate Time series(D$_S$)
Supervised learning
Task 1
Task 2
. . .
Task K
Target Data
Small labeled Dataset of Multivariate Time series(D$_T$)
HealthNet
Fine-tuning using HealthNet
Task K+1
Feature Extraction using HealthNet
Train Linear Classifier
Task K+1

Fig. 5 Task adaptation scenario. HealthNet is pre-trained for K classification tasks from healthcare domain simultaneously via supervised training. Then, it is adapted for target task either via fine-tuning or feature extraction (potentially using only a small amount of labeled training data for target task)

2. Extract features using HealthNet and then learn an easily trainable non-temporal linear classification model such as a logistic regression model [12] for target tasks with few labeled instances (described in Section 6.3).

More specifically, consider $\mathcal{D}_S$ and $\mathcal{D}_T$ are sets of labeled time series instances from an EHR database (i.e., same *domain*): $\mathcal{D}_S = \{(\mathbf{x}_S^{(i)}, \mathbf{y}_S^{(i)})\}_{i=1}^{N_S}$, where $\mathbf{x}_S^{(i)}$ is a multivariate time series, $\mathbf{y}_S^{(i)} = \{y_1, \ldots, y_K\} \in \{0, 1\}^K$, K is the number of binary classification tasks, and N_S is the number of time series instances corresponding to patients. Similarly, $\mathcal{D}_T = \{(\mathbf{x}_T^{(i)}, y_T^{(i)})\}_{i=1}^{N_T}$ such that $N_T \ll N_S$, and $y_T^{(i)} \in \{0, 1\}$ such that the target task is a binary classification task. As depicted in Fig. 6a, we first train HealthNet on K source tasks using $\mathcal{D}_S$ (refer to Section 6.1 for details), and then consider the following two scenarios for adapting to the pre-trained model to target tasks using $\mathcal{D}_T$:

- Fine-tune the HealthNet with suitable regularization (refer Section 6.2 for details), as shown in Fig. 6b. This allows us to train a model that does not require hyper-parameter tuning efforts.
- Train the simpler logistic regression (LR) classifier for target task and the features obtained via HealthNet (refer Section 6.3 for details), as shown in Fig. 7, which is compute-efficient.

We next provide details of training and fine-tuning the HealthNet and training the LR models.

6.1 Obtaining HealthNet Using Supervised Pre-training of RNN

Training an RNN on K binary classification tasks simultaneously can be considered as a multi-label classification problem. We train a multi-layered RNN with L recurrent layers having GRUs to map $\mathbf{x}^{(i)} \in \mathcal{D}_S$ to $\mathbf{y}^{(i)}$. Let $\mathbf{z}_{t,l} \in \mathbb{R}^h$ denote the output of recurrent units in l-th hidden layer at time t, and $\mathbf{z}_t = [\mathbf{z}_{t,1}, \ldots, \mathbf{z}_{t,L}] \in \mathbb{R}^m$ denote

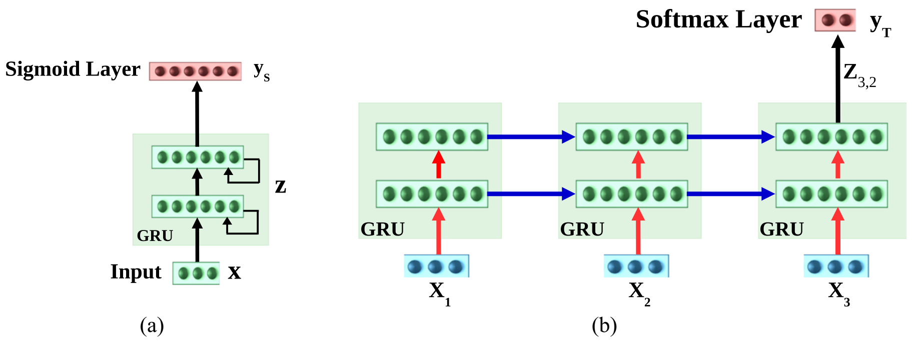


Fig. 6 **a** HealthNet trained via supervised learning for multiple source tasks simultaneously using final activation as sigmoid. **b** Fine-tuning HealthNet for a new target task using final activation as softmax. Here, blue and red arrows correspond to recurrent and feed forward weights of the recurrent layers, respectively. Only feed forward (red) weights are regularized while fine-tuning the HealthNet. RNN with L = 2 hidden layers is shown unrolled over $\tau = 3$ time steps

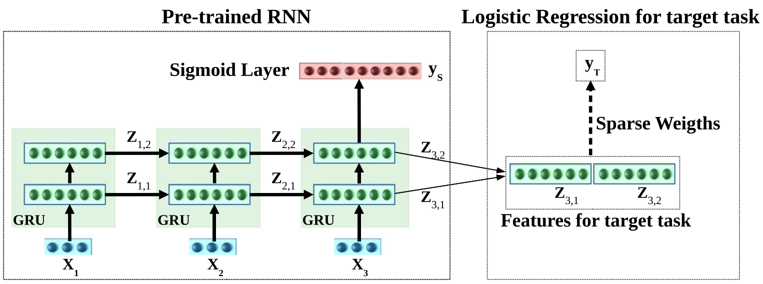


Fig. 7 Inference in task adaptation. Using features extracted from HealthNet. RNN with $L = 2$ hidden layers is shown unrolled over $\tau = 3$ time steps

the hidden state at time t obtained as concatenation of hidden states of all layers, where h is the number of GRU units in a hidden layer and $m = h \times L$. The parameters of the network are obtained by minimizing the cross-entropy loss $\mathcal{L}$ via stochastic gradient descent:

$$
\begin{aligned}
\mathbf{z}_\tau^{(i)} &= f_E(\mathbf{x}^{(i)}; \mathbf{W}'_E),\ \hat{\mathbf{y}}^{(i)} = \sigma(\mathbf{W}_C\, \mathbf{z}_{\tau,L}^{(i)} + \mathbf{b}_C) \\
C(y_k^{(i)}, \hat{y}_k^{(i)}) &= y_k^{(i)} \cdot \log(\hat{y}_k^{(i)}) + (1 - y_k^{(i)}) \cdot \log((1 - \hat{y}_k^{(i)})) \\
\mathcal{L} &= -\frac{1}{N_S \times K} \sum_{i=1}^{N_S} \sum_{k=1}^{K} C(y_k^{(i)}, \hat{y}_k^{(i)}).
\end{aligned}
\tag{3}
$$

Here $\sigma(x) = (1 + e^{-x})^{-1}$ is the sigmoid activation function, $\hat{\mathbf{y}}^{(i)}$ is the estimate for target $\mathbf{y}^{(i)}$, $\mathbf{W}'_E$ are parameters of recurrent layers, and $\mathbf{W}_C$ and $\mathbf{b}_C$ are parameters of the classification layer.

6.2 Fine-Tuning of HealthNet

We initialized the target task-specific RNN parameters by the pre-trained RNN parameters of recurrent layers ($\mathbf{W}'_E$) and a new binary classification layer parameters ($\mathbf{W}'_C$ and $\mathbf{b}'_C$). We obtain probabilities of two classes for the binary classification task as $\hat{\mathbf{y}}^{(i)}$ = softmax$(\mathbf{W}'_C\, \mathbf{z}_{\tau,L}^{\prime(i)} + \mathbf{b}'_C)$, where $\mathbf{z}_{\tau,L}^{\prime(i)}$ is the output of recurrent units in last layer (L) at last timestamp (τ). Let $\mathbf{W}'_{EF}$ and $\mathbf{W}'_{ER}$ are feed forward and recurrent weights of the recurrent layers. All parameters are trained together by minimizing cross-entropy loss with regularizer. We consider two regularizer techniques to obtain two different fine-tuned models with loss given by $\mathcal{L}_1$ and $\mathcal{L}_2$ via stochastic gradient descent:

$$
\mathcal{L}_1 = -\frac{1}{N_T} \sum_{i=1}^{N_T} C(y^{(i)}, \hat{y}^{(i)}) + \lambda \|\mathbf{W}'_{EF}\|_1 \tag{4}
$$

$$
\mathcal{L}_2 = -\frac{1}{N_T} \sum_{i=1}^{N_T} C(y^{(i)}, \hat{y}^{(i)}) + \lambda \|\mathbf{W}'_{EF}\|_2 \tag{5}
$$

where $\hat{y}^{(i)}$ is the probability of positive class, $||\mathbf{W}'_{EF}||_1 = \sum_{j=1}^{m} |W_j|$ is the L_1 norm with λ controlling the extent of sparsity, and $||\mathbf{W}'_{EF}||_2 = \sum_{j=1}^{m} {W_j}^2$ is the L_2 norm. As [25] suggests that using an L_1 or L_2 penalty on the recurrent weights compromises the ability of the network to learn and retain information through time, therefore, we apply L1 or L2 regularizer only to the feed forward connections across recurrent layers and not the weights of the recurrent connections.

6.3 LR Models Using Features Extracted from HealthNet

For input $\mathbf{x}^{(i)} \in \mathcal{D}_T$, the hidden state $\mathbf{z}_\tau^{(i)}$ at last time step τ is used as input feature vector for training the LR model. We obtain probability of the positive class for the binary classification task as $\hat{y}^{(i)} = \sigma(\mathbf{w}'_C\, \mathbf{z}_\tau^{(i)} + b'_C)$, where $\mathbf{w}'_C$, b'_C are parameters of LR. The parameters are obtained by minimizing the negative log-likelihood loss $\mathcal{L}'$:

$$\mathcal{L}' = -\frac{1}{N_T}\sum_{i=1}^{N_T} C(y^{(i)}, \hat{y}^{(i)}) + \lambda\|\mathbf{w}'_C\|_1 \tag{6}$$

where $||\mathbf{w}'_C||_1 = \sum_{j=1}^{m} |w_j|$ is the L$_1$ regularizer with λ controlling the extent of sparsity—with higher λ implying more sparsity, i.e., fewer features from the representation vector are selected for the final classifier. It is to be noted that this way of training the LR model on pre-trained RNN features is equivalent to freezing the parameters of all the hidden layers of the pre-trained RNN while tuning the parameters of a new final classification layer which has been used successfully in, e.g., [14]. The sparsity constraint ensures that only a small number of parameters are to be tuned which is useful to avoid overfitting when labeled data is small.

7 Dataset Description

We use MIMIC-III (v1.4) clinical database [13] which consists of over 60,000 ICU stays across 40,000 critical care patients. We use the same experimental setup as in [9], with same splits and features for train, validation, and test datasets[3] based on 17 physiological parameters with 12 real-valued (e.g., blood glucose level and systolic blood pressure) and 5 categorical time series (e.g., Glascow coma scale motor response and Glascow coma scale verbal), sampled at 1-h intervals. The categorical variables are converted to one-hot vectors such that the final multivariate time series has $n = 76$ raw input features (59 actual features and 17 masking features to denote missing values). Refer to Table 4 in the Appendix for names of raw features used. In all our experiments, we restrict training time series data up to first 48 h in ICU stay, such that $\tau = 48$ with one reading every 1 h while training all models to imitate practical scenario where early predictions are important, unlike [9, 32] which use entire time series for training the classifier for phenotyping task. The benchmark dataset contains label information for presence/absence of 25 phenotypes common in adult

[3]https://github.com/yerevann/mimic3-benchmarks

ICUs (e.g., acute cerebrovascular disease, diabetes mellitus with complications, and gastrointestinal hemorrhage), and in-hospital mortality, whether the patient survived or not after ICU admission (class 1: patient dies, class 0: patient survives).

The benchmark dataset contains label information for presence/absence of 25 phenotypes common in adult ICUs including 12 critical (and sometimes life-threatening) conditions, such as respiratory failure and sepsis; 8 chronic conditions that are common comorbidities and risk factors in critical care, such as diabetes and metabolic disorders; and 5 conditions considered "mixed" because they are recurring or chronic with periodic acute episodes. We also consider the task of predicting in-hospital mortality, i.e., whether the patient survived or not after ICU admission (class 1: patient dies, class 0: patient survives).

We use the same evaluation metrics and protocol as in [9, 10] unless mentioned otherwise, including macro- and micro-averaged AUROC for phenotyping task, and AUROC and AUPRC for in-hospital mortality prediction task, where AUROC (area under the receiver operator characteristic curve) is the commonly used metric in classification tasks, micro AUROC is calculated by single AUROC computed on flattened model prediction and ground truth matrices, micro AUROC calculated by averaging performance metric per-class/label. AUPRC (area under the precision-recall curve) better suits to problems with imbalanced classes which happens to be the case here for both phenotyping and in-hospital mortality tasks, e.g., 4493 (10.63%) out of 42,276 patients died in-hospital making in-hospital mortality prediction task to have high class imbalance.

8 Experimental Evaluation for Domain Adaptation

We evaluate TimeNet-based transfer learning approach on binary classification tasks (i) presence/absence of 25 phenotypes and (ii) in-hospital mortality task.

8.1 Experimental Setup

We have $n = 76$ raw input features resulting in $m = 13,680$-dimensional ($m = 76 \times 180$) TimeNet feature vector for each admission. We use $\alpha = 0.0001$ for phenotype classifiers and use $\alpha = 0.0003$ for in-hospital mortality classifier (α is chosen based on hold-out validation set). Table 1 summarizes the results of phenotyping and in-hospital mortality prediction task, and provides comparison with existing benchmarks. Refer to Table 3 in the Appendix for detailed phenotype-wise results. We report the values of the performance metrics for all models along with 95% confidence intervals obtained by resampling the test set 10,000 times with replacement.

We consider two variants of classifier models for phenotyping task: (i) *TimeNet-x* using data from current episode and (ii) *TimeNet-x-Eps* using data from previous episode of a patient as well (whenever available) via an additional input feature related to presence or absence of the phenotype in previous episode. Each classifier is trained using up to first 48 h of data after ICU admission. However, we consider two classifier variants depending upon hours of data x used to estimate the target class at

Table 1 Classification performance comparison of TimeNet-based models with LR (logistic regression model over manually designed statistical features) and LSTM (LSTM classifier trained from scratch) for the phenotyping and in-hospital mortality prediction tasks

Phenotyping task		
approach	Micro AUROC	Macro AUROC
LR	0.799 (0.796, 0.803)	0.739 (0.734, 0.743)
LSTM	0.821 (0.818, 0.825)	0.770 (0.766, 0.775)
TimeNet-48	0.812 (0.808, 0.815)	0.761 (0.757, 0.765)
TimeNet-All	0.813 (0.810, 0.817)	0.764 (0.759, 0.768)
TimeNet-48-Eps	0.820 (0.817, 0.824)	0.772 (0.768, 0.777)
TimeNet-All-Eps	0.822 (0.819, 0.825)	0.775 (0.771,0.779)
In-hospital mortality prediction task		
approach	AUROC	AUPRC
LR	0.848 (0.828, 0.868)	0.474 (0.419, 0.529)
LSTM	0.855 (0.835, 0.873)	0.485 (0.431, 0.537)
TimeNet-48	0.852 (0.831, 0.872)	0.519 (0.467, 0.571)

Numbers in round brackets denote 95% confidence intervals. Here, results of LR and LSTM are taken from [10] (Note: For phenotyping, we compare TimeNet-48-Eps with existing benchmarks over TimeNet-All-Eps as it is more applicable in practical scenarios. Only TimeNet-48 variant is applicable for in-hospital mortality task.)

test time. For $x = 48$, data up to first 48 h after admission is used for determining the phenotype. For $x = All$, the learned classifier is applied to all 48-h windows (overlapping with shift of 24 h) over the entire ICU stay period of a patient, and the average phenotype probability across windows is used as the final estimate of the target class. In *TimeNet-x-Eps*, the additional feature is related to the presence (1) or absence (0) of the phenotype during the previous episode. We use the ground-truth value for this feature during training time, and the probability of presence of phenotype during previous episode (as given via LASSO-based classifier) at test time.

For evaluating the efficacy of the features extracted via TimeNet, we compare our approach with a logistic regression model (LR) trained using manually designed statistical features from raw time series as used in [9]: For each input physiological parameter, it uses minimum, maximum, mean, standard deviation, skew, and number of measurements for each of the seven different subsequences or chunks of a given time series. The seven subsequences correspond to the full time series, and the data over the first 10%, 25%, 50%, last 50%, last 25%, and last 10% of time. Effectively, this results in $17 \times 7 \times 6 = 714$ features per time series (where 17 is the number of original raw physiological parameters).

8.2 Results and Observations

8.2.1 Classification Tasks

For the phenotyping task, we make the following observations from Table 1:

1. *TimeNet-48 versus LR*: *TimeNet-based features perform significantly better than hand-crafted features* as used in LR ([9, 10], while using only the first 48 h of data unlike the LR approach that uses the entire episode's data. This proves the effectiveness of TimeNet features for MIMIC-III data. Further, it only requires *tuning a single hyper-parameter* α for LASSO, unlike other approaches like LSTM [9, 10] that would involve finding the suitable number of hidden units, layers, learning rate, etc. and training the deep networks from scratch.
2. *TimeNet-x versus TimeNet-x-Eps*: Leveraging previous episode's time series data for a patient significantly improves the classification performance.
3. *TimeNet-48-Eps* performs at par existing benchmarks, while still being *practically more feasible* as it looks at only up to 48 h of the current episode of a patient rather than the entire current episode. For in-hospital mortality task, we observe comparable performance to existing benchmarks.

Training linear models is significantly fast and it took around 10 min for obtaining any of the binary classifiers while tuning for $\alpha \in [10^{-5} - 10^{-3}]$ (five equally spaced values) on a 32GB RAM machine with Quad Core i7 2.7GHz processor. We observe that LASSO leads to 96.2 ± 0.8 % sparsity (i.e., percentage of weights $w_{jk} \approx 0$) for all classifiers leading to around 550 useful features (out of 13,680) for each phenotype classification.

8.2.2 Relevance Scores for Raw Input Features

We observe intuitive interpretation for relevance of raw input features using the weights assigned to various TimeNet features (refer (2)): For example, as shown in Fig. 8, we obtain highest relevance scores for glucose level (feature 1) and systolic blood pressure (feature 20) for diabetes mellitus with complications (Fig. 8a), and essential hypertension (Fig. 8b), respectively. Refer to the Appendix Fig. 13 for more details. We conclude that even though TimeNet was never trained on MIMIC-III data, it still provides meaningful general-purpose features from time series of raw input features, and LASSO helps to select the most relevant ones for end task by using labeled data. Further, extracting features using a deep recurrent neural network

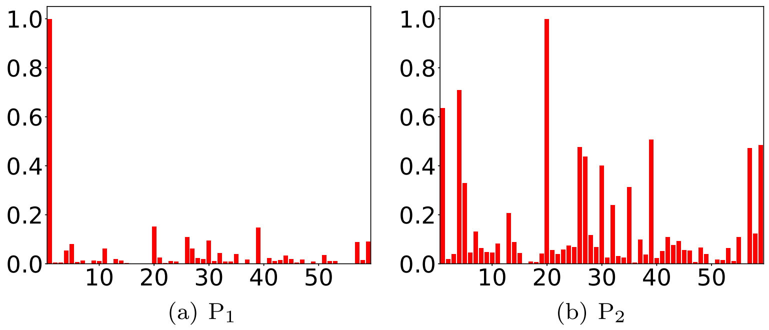


Fig. 8 **a**, **b** Relevance scores for raw input features or parameters. *x*-axis: feature number, *y*-axis: relevance score. Here, P_1: diabetes mellitus with complications, P_2: essential hypertension

model for time series of each raw input feature independently—rather than considering a multivariate time series—eventually allows to easily assign relevance scores to raw features in the input domain, allowing a high-level basic model validation by domain-experts.

9 Experimental Evaluation for Task Adaptation

9.1 Experimental Setup

We evaluate the HealthNet-based transfer learning approach on the same tasks as used in Section 8 with the following evaluation protocol as depicted in Fig. 9:

Out of 25 phenotypes, we consider $K = 20$ phenotypes to obtain the pre-trained RNN which we refer to as HealthNet (HN) and test the transferability of the features from HN to remaining 5 phenotype classification tasks with varying labeled data sizes. Since more than one phenotype may be present in a patient at a time, we remove all patients with any of the 5 test phenotypes from the original train and validate sets (despite them having one of the 20 train phenotypes also) to avoid any information leakage. Except for this, note that the train, validate, and test splits remain the same as in [9]. We report average results in terms of weighted AUROC (as in [9]) on two

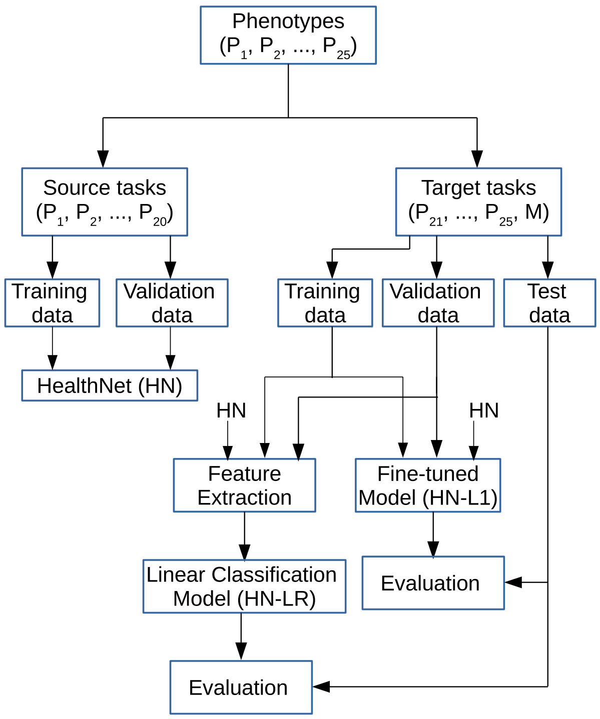


Fig. 9 Evaluation protocol for task adaptation

random splits of 20 train (and validate) phenotypes and 5 test phenotypes, such that we have 10 test phenotypes (tested one-at-a-time). We also test the transferability of HN features to in-hospital mortality prediction task.

For each target task (i.e., 10 phenotypes and in-hospital mortality prediction task), we test the robustness of HN to labeled training data size. We use random stratified sub-sampling of the training and validation data to obtain reduced labeled training and validation sets. We consider the number of hidden layers $L = 2$, batch size of 128, regularization using dropout factor [26] of 0.3, and Adam optimizer [15] with initial learning rate 10^{-4} for training RNNs. The number of hidden units h with minimum $\mathcal{L}$ (3) on the validation set is chosen from $\{100, 200, 300, 400\}$. Best HN model was obtained for $h = 300$ such that total number of features is $m = 600$. For fine-tuning of HN, we use the same parameters as used in training HN with regularization factor $\lambda = 0.01$ (4 and 5). For the linear classification model, the parameter λ is tuned on $\{0.1, 1.0, \ldots, 10^4\}$ (on a logarithmic scale) to minimize $\mathcal{L}'$ (6) on the validation set.

9.2 Results and Observations

We refer to the fine-tuned model using L_1 regularizer as **HN-L1**, using L_2 regularizer as **HN-L2**, without regularizer as **HN-Tune**, and LR model learned using HN features as **HN-LR**, and consider two baselines for comparison: (i) logistic regression (**LR**) using manually designed statistical features described in Section 8.1 and (ii) RNN classifier (**RNN-C**) with gated units as in LSTMs (or GRUs) obtained using training and validation data for the target task. To test the robustness of the models for small-labeled training sets, we consider subsets of training and validation datasets as explained in Section 9.1, while the test set remains the same. Further, we also evaluate the relevance of layer-wise features $\mathbf{z}_{\tau,l}$ from the $L = 2$ hidden layers of HealthNet. **HN-LR-1** and **HN-LR-2** refer to models trained using $\mathbf{z}_{\tau,2}$ (the topmost hidden layer only) and $\mathbf{z}_{\tau} = [\mathbf{z}_{\tau,1}, \mathbf{z}_{\tau,2}]$ (from both hidden layers), respectively.

Comparing Variants of HealthNet-Based Transfer Learning Techniques The results for phenotyping tasks in Fig. 10a suggest that (i) all the variants of HN-based transfer learning performs equally well on 100% training data; (ii) regularized fine-tuned models (HN-L1 and HN-L2) consistently outperform non-regularized fine-tuned model (HN-Tune) as training dataset is reduced. Importantly, as the size of labeled training set is reduced, non-regularized fine-tuned model degrades quickly as it is prone to overfitting due to a large number of trainable parameters. (iii) For medium-sized datasets (20% and 60% of original training data), the fine-tuned models HN-L1 and HN-L2 perform better than feature extraction based HN-LR models as well as the models without any regularization. However, for extremely large or small data sizes (5% and 100% cases), the difference is insignificant, thus indicating the expected trade-off between fine-tuning and feature extraction-based methods w.r.t. training data size: when labeled training data is small, fine-tuning all the parameters of a deep network may still be prone to some overfitting even after careful regularization.

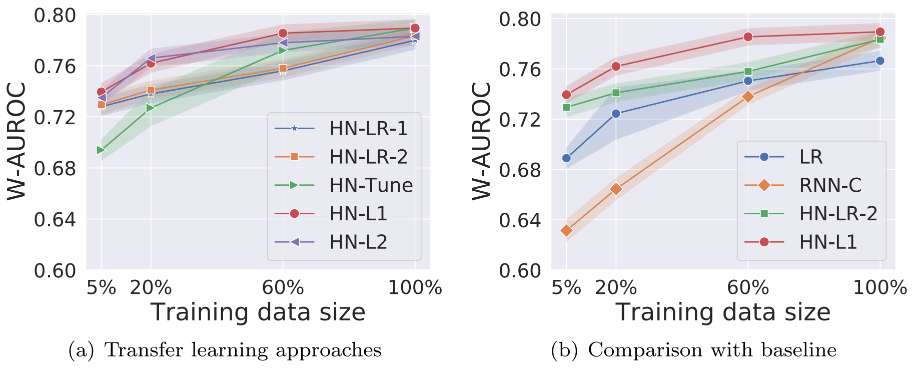


Fig. 10 **a**, **b** Classification performance comparison along with 95% confidence intervals for phenotyping task. Here, baseline models are LR and RNN-C, W-AUROC is weighted-AUROC over 10 phenotypes

Robustness of Transfer Learning Models Versus Models Trained From Scratch to Training Data Size We compare the best fine-tuning based and feature extraction based variants, i.e., HN-L1 and HN-LR-2, with the LR and RNN-C models trained from scratch. As shown in Fig. 10b, we observe that:

(i) HN-L1, HN-LR-2, and RNN-C perform equally well when using 100% training data, and are better than LR. This implies that the transfer learning–based models are as effective as models trained specifically for the target task on large labeled datasets while reducing training efforts and automating feature extraction.

(ii) HN-L1 consistently outperforms RNN-C and LR models as training dataset is reduced. As the size of labeled training dataset reduces, though the performance of the model trained from scratch (RNN-C) as well as transfer learning models (HN-L1 and HN-LR-2) degrades, we observe that transfer learning based models degrade much more gracefully and perform significantly better than RNN-C for reduced data sizes (here, the 5% labeled data scenario). The performance gains from transfer learning are greater when the training set of the target task is small. Therefore, with transfer learning, fewer labeled instances are needed to achieve the same level of performance as model trained on target data alone.

(iii) As labeled training set is reduced, LR performs better than RNN-C confirming that deep networks are prone to overfitting on small datasets.

We also evaluate the transfer learning approach on in-hospital mortality task, and found transfer learning to be useful there as well. However, since the task is potentially unrelated to the phenotyping task, we do not observe significant improvements over the models trained from scratch. Refer to Fig. 12a in the Appendix for more details. From Fig. 12b, we interestingly observe that HN-LR-2 results are at least as good as RNN-C and LR on the seemingly unrelated task of mortality prediction, suggesting that the features learned are generic enough and transfer well.

Number of Relevant Features for a Task We observe that only a small number of features are actually relevant for a target classification task out of large number of input features to LR models (714 for LR, 300 for HN-LR-1, and 600 for HN-LR-2). As shown in Table 2, $> 95\%$ of features have weight ≈ 0 (absolute value < 0.001) for HN-LR models corresponding to phenotyping tasks due to sparsity constraint (6), i.e., most features do not contribute to the classification decision. The weights of features that are non-zero for at least one of target tasks for HN-LR-1 are shown in Supplementary Material Fig. 14. We observe that, for example, for HN-LR-1 model, only 130 features (out of 300) are relevant across the 10 phenotype classification tasks and the mortality prediction task. This suggests that HN provides several generic features while LR learns to select the most relevant ones given a small-labeled dataset. Table 2 (and Fig. 14 in the Appendix) also suggests that HN-LR models use a larger number of features for mortality prediction task, possibly because concise features for mortality prediction are not available in the learned set of features as HN was pre-trained for phenotype identification tasks.

10 Domain Adaptation Versus Task Adaptation

We study in what scenarios can domain adaptation be valuable compared with task adaptation and vice versa. To this effect, we consider a scenario where none or a very small number of source tasks from the healthcare domain are available for pre-training. When there is no source task from the healthcare domain for pre-training, the user has to resort to either training from scratch for the target task or use domain adaptation via TimeNet. However, when a small number (say, 2–3) of source tasks are available from the healthcare domain, it may be worth consider pre-training a healthcare domain-specific model rather than solely relying on TimeNet.

We evaluate on 5 binary classification target tasks, i.e., presence/absence of 5 phenotypes and report average results in Fig. 11. We use the same setup as in Sections 8 and 9 for domain and task adaptation, respectively. We vary the number of randomly chosen source tasks for task adaptation with $K = \{2, 5, 20\}$ phenotypes out of the 20 remaining phenotypes (out of 25 tasks, 5 are used for testing) to obtain the pre-trained RNN, i.e., HealthNet (HN), and test the transferability of HN-L1 to 5 phenotypes with 20% of training data. We report the results using split K randomly selected train phenotypes out of 20 and 5 fixed test phenotypes, and report the average of three-run. For domain adaptation, we rely only on domain adaptation without using any of the K source tasks for pre-training.

Table 2 Fraction of features with weight ≈ 0. Here, the average and standard deviation over 10 phenotypes is reported for phenotyping task

Series task	Series LR	Series HN-LR-1	Series HN-LR-2
Phenotyping	0.902 ± 0.023	0.955 ± 0.020	0.974 ± 0.011
In-hospital mortality	0.917	0.787	0.867

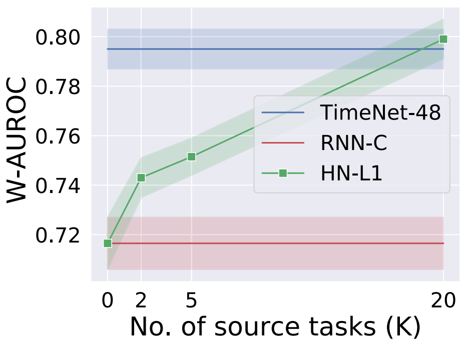


Fig. 11 Classification performance comparison along with 95% confidence intervals. HealthNet-based transfer learning (HN-L1) versus TimeNet-based transfer learning (TimeNet-48). Here, W-AUROC is weighted-AUROC over 5 phenotype target tasks

10.1 Results and Observations

From Fig. 11, we observe that the performance of task adaptation approach HN-L1 degrades in comparison with domain adaptation approach (TimeNet-48) when the number of source tasks K available for training reduces. Further, TimeNet-48 and HN-L1 perform comparably when there is enough labeled data from a diverse set of tasks within the healthcare domain to allow for pre-training and leverage within-domain transfer. The advantage of TimeNet-like off-the-shelf generic models is evident as such models can be trained on publicly available time series data across domains in an unsupervised manner, and therefore, the performance of domain adaptation models is not very sensitive to the data in the healthcare domain. In other words, domain adaptation–based approaches are useful when there is not enough labeled data for the target task nor sufficient tasks in the target domain to pre-train a domain-specific model to leverage the benefits of transfer learning.

We also contrast the time taken to fine-tune TimeNet-based models and HealthNet-based models. Training and fine-tuning times are recorded on a 32GB RAM machine with Quad Core i7 2.7GHz processor. Training linear models (e.g., over TimeNet features) is fast, and it took around 10 min to obtain any of the target task-specific binary classifiers while tuning for $\alpha \in [10^{-5} - 10^{-3}]$ (five equally spaced values). However, fine-tuning a deep learning model (e.g., HN) took around 45 min which is slower in comparison with TimeNet-based models but still faster than training a deep learning model (i.e., RNN-C) from scratch that took around 3 h for obtaining a final model while tuning for number of hidden units from {100, 200, 300, 400}.

11 Conclusion

Deep neural networks require heavy computational resources for training and are prone to overfitting. Scarce labeled training data, significant hyper-parameter tuning efforts, and scarce computational resources are often a bottleneck in adopting deep learning–based solutions to healthcare applications. In this work, we have proposed effective approaches for transfer learning in the healthcare domain by using deep recurrent neural networks (RNN). We considered two scenarios for transfer learning: (i) adapting a deep RNN-based universal time series feature extractor (TimeNet) to healthcare tasks and applications and (ii) adapting a deep RNN (HealthNet) pre-trained on healthcare tasks to a new related task. Our approach brings the advantage of deep learning such as automated feature extraction and ability to easily deal with variable length time series while still being simple to adapt to the target tasks. We have demonstrated that our transfer learning approaches can lead to significant gains in classification performance compared with traditional models using carefully designed statistical features or task-specific deep models in scarcely labeled training data scenarios. Further, leveraging pre-trained models ensures very little tuning effort, and therefore, fast adaptation. We also found that raw feature-wise handling of time series via TimeNet and subsequent linear classifier training can provide insights into the importance and relevance of a raw feature (physiological parameter) for a given task while still modeling the temporal aspect. This raw feature relevance scoring can help domain-experts gain at least a high-level insight into the working of otherwise opaque deep RNNs.

In future, evaluating a domain-specific TimeNet-like model for clinical time series (e.g., trained only on MIMIC-III database) will be interesting. Also, transferability and generalization capability of RNNs trained simultaneously on diverse tasks (such as length of stay, mortality prediction, and phenotyping [9, 32]) to new tasks is an interesting future direction.

Acknowledgments We would like to thank the anonymous reviewers for their valuable comments that have helped to significantly enhance this paper.

Compliance with Ethical Standards

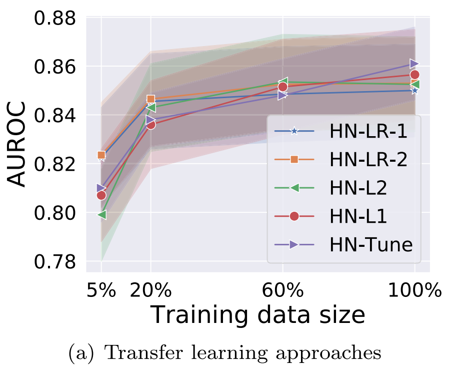
Conflict of Interest The authors declare that they have no conflict of interest.

Appendix: Multi-layered RNN with Gated Recurrent Units

A gated recurrent unit (GRU) [5] consists of an *update gate* and a *reset gate* that control the flow of information by manipulating the *hidden state* of the unit as in Equation (7).

In an RNN with L hidden layers, the reset gate is used to compute a proposed value $\tilde{\mathbf{z}}_t^l$ for the hidden state $\mathbf{z}_t^l$ at time t for the l-th hidden layer by using the hidden state $\mathbf{z}_{t-1}^l$ and the hidden state $\mathbf{z}_t^{l-1}$ of the units in the lower hidden layer at time t. The update gate decides as to what fractions of previous hidden state and proposed hidden state to use to obtain the updated hidden state at time t. In turn, the values of the reset gate and update gate themselves depend on the $\mathbf{z}_{t-1}^l$ and $\mathbf{z}_t^{l-1}$

We use dropout variant for RNNs as proposed in [26] for regularization such that dropout is applied only to the non-recurrent connections, ensuring information flow across time steps.



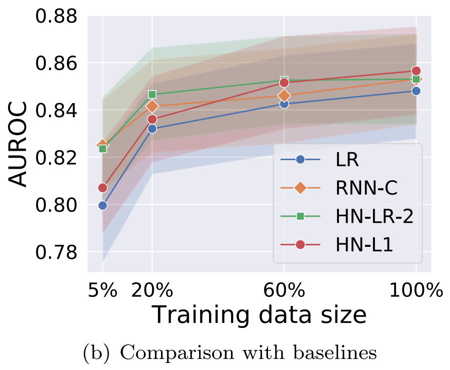


Fig. 12 **a**, **b** Classification performance comparison along with 95% confidence intervals for In-hospital mortality task. Here, baseline models are LR and RNN-C

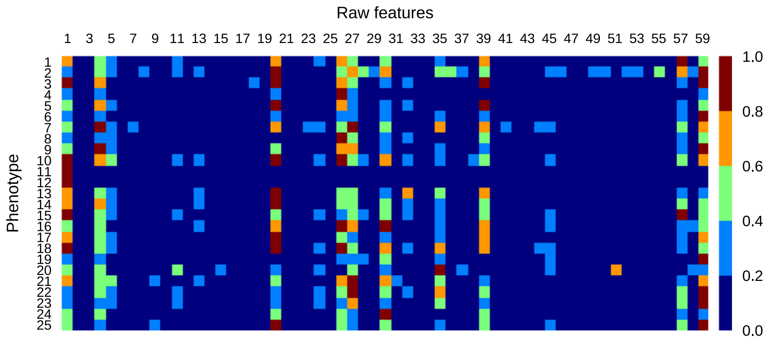


Fig. 13 Feature relevance scores for 25 phenotypes using TimeNet-based transfer learning. Refer to Table 3 for the names of phenotypes and Table 4 for the names of raw features

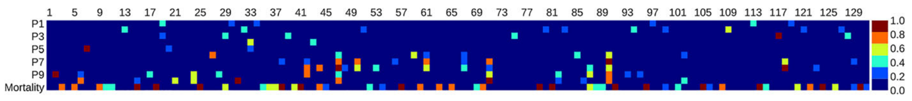


Fig. 14 Feature weights (absolute) for HN-LR-1. Here P$_i$ ($i = 1, \ldots, 10$) denotes i-th phenotype identification task. x-axis: feature number, y-axis: task

Table 3 Phenotype-wise classification performance in terms of AUROC

S.No.	Phenotype	LSTM-Multi	TimeNet-48	TimeNet-All	TimeNet-48-Eps	TimeNet-All-Eps
1	Acute and unspecified renal failure	0.8035	0.7861	0.7887	0.7912	0.7941
2	Acute cerebrovascular disease	0.9089	0.8989	0.9031	0.8986	0.9033
3	Acute myocardial infarction	0.7695	0.7501	0.7478	0.7533	0.7509
4	Cardiac dysrhythmias	0.684	0.6853	0.7005	0.7096	0.7239
5	Chronic kidney disease	0.7771	0.7764	0.7888	0.7960	0.8061
6	Chronic obstructive pulmonary disease and bronchiectasis	0.6786	0.7096	0.7236	0.7460	0.7605
7	Complications of surgical procedures or medical care	0.7176	0.7061	0.6998	0.7092	0.7029
8	Conduction disorders	0.726	0.7070	0.7111	0.7286	0.7324
9	Congestive heart failure; nonhypertensive	0.7608	0.7464	0.7541	0.7747	0.7805
10	Coronary atherosclerosis and other heart disease	0.7922	0.7764	0.7760	0.8007	0.8016
11	Diabetes mellitus with complications	0.8738	0.8748	0.8800	0.8856	0.8887
12	Diabetes mellitus without complication	0.7897	0.7749	0.7853	0.7904	0.8000
13	Disorders of lipid metabolism	0.7213	0.7055	0.7119	0.7217	0.7280
14	Essential hypertension	0.6779	0.6591	0.6650	0.6757	0.6825
15	Fluid and electrolyte disorders	0.7405	0.7351	0.7301	0.7377	0.7328
16	Gastrointestinal hemorrhage	0.7413	0.7364	0.7309	0.7386	0.7343
17	Hypertension with complications and secondary hypertension	0.76	0.7606	0.7700	0.7792	0.7871
18	Other liver diseases	0.7659	0.7358	0.7332	0.7573	0.7530
19	Other lower respiratory disease	0.688	0.6847	0.6897	0.6896	0.6922
20	Other upper respiratory disease	0.7599	0.7515	0.7565	0.7595	0.7530
21	Pleurisy; pneumothorax; pulmonary collapse	0.7027	0.6900	0.6882	0.6909	0.6997
22	Pneumonia	0.8082	0.7857	0.7916	0.7890	0.7943
23	Respiratory failure; insufficiency; arrest (adult)	0.9015	0.8815	0.8856	0.8834	0.8876
24	Septicemia (except in labor)	0.8426	0.8276	0.8140	0.8296	0.8165
25	Shock	0.876	0.8764	0.8564	0.8763	0.8562

Table 4 List of raw input features

1	Glucose	31	Glascow coma scale eye opening → 3 to speech
2	Glascow coma scale total → 7	32	Height
3	Glascow coma scale verbal response → incomprehensible sounds	33	Glascow coma scale motor response → 5 localizes pain
4	Diastolic blood pressure	34	Glascow coma scale total → 14
5	Weight	35	Fraction inspired oxygen
6	Glascow coma scale total → 8	36	Glascow coma scale total → 12
7	Glascow coma scale motor response → obeys commands	37	Glascow coma scale verbal response → confused
8	Glascow coma scale eye opening → none	38	Glascow coma scale motor response → 1 no response
9	Glascow coma scale eye opening → to pain	39	Mean blood pressure
10	Glascow coma scale total → 6	40	Glascow coma scale total → 4
11	Glascow coma scale verbal response → 1.0 ET/Trach	41	Glascow coma scale eye opening → to speech
12	Glascow coma scale total → 5	42	Glascow coma scale total → 15
13	Glascow coma scale verbal response → 5 oriented	43	Glascow coma scale motor response → 4 flex-withdraws
14	Glascow coma scale total → 3	44	Glascow coma scale motor response → no response
15	Glascow coma scale verbal response → no response	45	Glascow coma scale eye opening → spontaneously
16	Glascow coma scale motor response → 3 abnorm flexion	46	Glascow coma scale verbal response → 4 confused
17	Glascow coma scale verbal response → 3 inapprop words	47	Capillary refill rate → 0.0
18	Capillary refill rate → 1.0	48	Glascow coma scale total → 13
19	Glascow coma scale verbal response → inappropriate words	49	Glascow coma scale eye opening → 1 no response
20	Systolic blood pressure	50	Glascow coma scale motor response → abnormal extension
21	Glascow coma scale motor response → flex-withdraws	51	Glascow coma scale total → 11
22	Glascow coma scale total → 10	52	Glascow coma scale verbal response → 2 incomp sounds
23	Glascow coma scale motor response → obeys commands	53	Glascow coma scale total → 9
24	Glascow coma scale verbal response → no response-ETT	54	Glascow coma scale motor response → abnormal flexion
25	Glascow coma scale eye opening → 2 to pain	55	Glascow coma scale verbal response → 1 no response
26	Heart rate	56	Glascow coma scale motor response → 2 abnorm extensn
27	Respiratory rate	57	pH

Table 4 (continued)

28	Glascow coma scale verbal response → oriented	58	Glascow coma scale eye opening → 4 spontaneously
29	Glascow coma scale motor response → localizes pain	59	Oxygen saturation
30	Temperature		

The time series goes through the following transformations iteratively for $t = 1$ through T, where T is length of the time series:

$$\begin{aligned}
\text{reset gate} : \mathbf{r}_t^l &= \sigma(\mathbf{W}_r^l \cdot \mathbf{D}(\mathbf{z}_t^{l-1}), \mathbf{z}_{t-1}^l]) \\
\text{update gate} : \mathbf{u}_t^l &= \sigma(\mathbf{W}_u^l \cdot [\mathbf{D}(\mathbf{z}_t^{l-1}), \mathbf{z}_{t-1}^l]) \\
\text{proposed state} : \tilde{\mathbf{z}}_t^l &= \tanh(\mathbf{W}_p^l \cdot [\mathbf{D}(\mathbf{z}_t^{l-1}), \mathbf{r}_t \odot \mathbf{z}_{t-1}^l]) \\
\text{hidden state} : \mathbf{z}_t^l &= (1 - \mathbf{u}_t^l) \odot \mathbf{z}_{t-1}^l + \mathbf{u}_t^l \odot \tilde{\mathbf{z}}_t^l \qquad (7)
\end{aligned}$$

where $\odot$ is Hadamard product, $[\mathbf{a}, \mathbf{b}]$ is concatenation of vectors $\mathbf{a}$ and $\mathbf{b}$, $\mathbf{D}(\cdot)$ is dropout operator that randomly sets the dimensions of its argument to zero with probability equal to dropout rate, and $\mathbf{z}_t^0$ equals the input at time t. $\mathbf{W}_r$, $\mathbf{W}_u$, and $\mathbf{W}_p$ are weight matrices of appropriate dimensions s.t. $\mathbf{r}_t^l, \mathbf{u}_t^l, \tilde{\mathbf{z}}_t^l$, and $\mathbf{z}_t^l$ are vectors in $\mathbf{R}^{c^l}$, where c^l is the number of units in layer l. The sigmoid (σ) and $tanh$ activation functions are applied element-wise.

References

1. Bahdanau D, Cho K, Bengio Y (2014) Neural machine translation by jointly learning to align and translate. arXiv:14090473
2. Bengio Y (2012) Deep learning of representations for unsupervised and transfer learning. In: Proceedings of ICML workshop on unsupervised and transfer learning, pp 17–36
3. Che Z, Purushotham S, Cho K, Sontag D, Liu Y (2016) Recurrent neural networks for multivariate time series with missing values. arXiv:160601865
4. Chen Y, Keogh E, Hu B, Begum N, et al. (2015) The ucr time series classification archive. www.cs.ucr.edu/eamonn/time_series_data/
5. Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv:14061078
6. Choi E, Bahadori MT, Schuetz A, Stewart WF, Sun J (2016) Doctor ai: predicting clinical events via recurrent neural networks. In: Machine Learning for Healthcare Conference, pp 301–318
7. Gupta P, Malhotra P, Vig L, Shroff G (2018) Transfer learning for clinical time series analysis using recurrent neural networks
8. Gupta P, Malhotra P, Vig L, Shroff G (2018) Using features from pre-trained timenet for clinical predictions
9. Harutyunyan H, Khachatrian H, Kale DC, Galstyan A (2017) Multitask learning and benchmarking with clinical time series data. arXiv:170307771
10. Harutyunyan H, Khachatrian H, Kale DC, Ver Steeg G, Galstyan A (2019) Multitask learning and benchmarking with clinical time series data. Scientific Data 6(1):96
11. Hermans M, Schrauwen B (2013) Training and analysing deep recurrent neural networks. In: Advances in neural information processing systems, pp 190–198

12. Hosmer DW Jr, Lemeshow S, Sturdivant RX (2013) Applied logistic regression, vol 398. Wiley, Hoboken
13. Johnson AE, Pollard TJ et al (2016) Mimic-iii, a freely accessible critical care database. Scientific Data 3:160035
14. Kashiparekh K, Narwariya J, Malhotra P, Vig L, Shroff G (2019) Convtimenet: a pre-trained deep convolutional neural network for time series classification. In: 2019 International joint conference on neural networks (IJCNN). IEEE
15. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv:14126980
16. Lee JY, Dernoncourt F, Szolovits P (2017) Transfer learning for named-entity recognition with neural networks. arXiv:170506273
17. Lipton ZC, Kale DC, Elkan C, Wetzel R (2015) Learning to diagnose with lstm recurrent neural networks. arXiv:151103677
18. Malhotra P, Vig L, Shroff G, Agarwal P (2015) Long short term memory networks for anomaly detection in time series. In: ESANN, 23rd European symposium on artificial neural networks, computational intelligence and machine learning, pp 89–94
19. Malhotra P, TV V, Vig L, Agarwal P, Shroff G (2017) TimeNet: pre-trained deep recurrent neural network for time series classification. In: 25th European symposium on artificial neural networks, computational intelligence and machine learning, pp 607–612
20. Micenková B, Dang XH, Assent I, Ng RT (2013) Explaining outliers by subspace separability. In: 2013 IEEE 13th international conference on data mining (ICDM). IEEE, pp 518–527
21. Miotto R, Li L, Kidd BA, Dudley JT (2016) Deep patient: an unsupervised representation to predict the future of patients from the electronic health records. Scientific reports 6:26094
22. Miotto R, Wang F, Wang S, Jiang X, Dudley JT (2017) Deep learning for healthcare: review, opportunities and challenges. Briefings in bioinformatics
23. Nguyen P, Tran T, Wickramasinghe N, Venkatesh S (2017) Deepr: a convolutional net for medical records. IEEE J Biomed Health Info 21(1):22–30
24. Pan SJ, Yang Q (2010) A survey on transfer learning. IEEE Trans Knowl Data Eng 22(10):1345–1359
25. Pascanu R, Mikolov T, Bengio Y (2013) On the difficulty of training recurrent neural networks. arXiv:12115063
26. Pham V, Bluche T, Kermorvant C, Louradour J (2014) Dropout improves recurrent neural networks for handwriting recognition. In: Frontiers in handwriting recognition (ICFHR). IEEE, pp 285-?290
27. Purushotham S, Meng C, Che Z, Liu Y (2017) Benchmark of deep learning models on large healthcare mimic datasets. arXiv:171008531
28. Rajkomar A, Oren E, Chen K, Dai AM, Hajaj N, Liu PJ, Liu X, Sun M, Sundberg P, Yee H, et al. (2018) Scalable and accurate deep learning for electronic health records. arXiv:180107860
29. Ravì D, Wong C, Deligianni F, Berthelot M, Andreu-Perez J, Lo B, Yang GZ (2017) Deep learning for health informatics. IEEE J Biomed Health Infor 21(1):4–21
30. Serra J, Pascual S, Karatzoglou A (2018) Towards a universal neural network encoder for time series
31. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv:14091556
32. Song H, Rajan D, Thiagarajan JJ, Spanias A (2017) Attend and diagnose: clinical time series analysis using attention models. arXiv:171103905
33. Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems, pp 3104–3112
34. Tibshirani R (1996) Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society Series B (Methodological): 267–288
35. Yosinski J, Clune J, Bengio Y, Lipson H (2014) How transferable are features in deep neural networks? In: Advances in neural information processing systems, pp 3320–3328