VARIATIONAL RECURRENT ADVERSARIAL DEEP DOMAIN ADAPTATION

Sanjay Purushotham*, Wilka Carvalho*, Tanachat Nilanon, Yan Liu
Department of Computer Science
University of Southern California
Los Angeles, CA 90089, USA
{spurusho,wcarvalh,nilanon,yanliu.cs}@usc.edu

ABSTRACT

We study the problem of learning domain invariant representations for time series data while transferring the complex temporal latent dependencies between domains. Our model termed as *Variational Recurrent Adversarial Deep Domain Adaptation* (VRADA) is built atop a variational recurrent neural network (VRNN) and trains adversarially to capture complex temporal relationships that are domain-invariant. This is (as far as we know) the first to capture and transfer temporal latent dependencies of multivariate time-series data. Through experiments on real-world multivariate healthcare time-series datasets, we empirically demonstrate that learning temporal dependencies helps our model's ability to create domain-invariant representations, allowing our model to outperform current state-of-the-art deep domain adaptation approaches.

1 INTRODUCTION

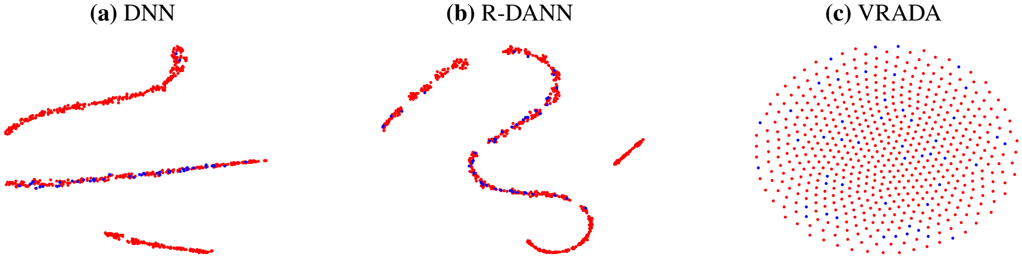
Many real-world applications require effective machine learning algorithms that can learn invariant representations across related time-series datasets. For example, precision medicine for patients of various age groups, mobile application recommendation for users based on locations, and so on. In these examples, while the domains (i.e. age group and location) may vary, there exist common predictive patterns that can aid in inferring knowledge from one domain to another. More often than not, some domains have a significantly larger number of observations than others (e.g., respiratory failure in adults vs. children). Therefore effective domain adaption of time-series data is in great demand.

The general approach to tackling domain adaptation has been explored under many facets which include reducing the domain discrepancy between the source and target domains(Ben-David et al. (2007)), instance re-weighting (Jiang & Zhai (2007)), subspace alignment (Fernando et al. (2013)), and deep learning (Tzeng et al. (2015); Ganin & Lempitsky (2014)). Many of these approaches work very well for non-sequential data but are not suitable for multivariate time-series data as they do not usually capture the temporal dependencies present in the data. For sequential data, earlier work has successfully used dynamic Bayesian Networks(Huang & Yates (2009)) and Recurrent Neural Networks (Socher et al. (2011)) to learn latent feature representations which were domain-invariant. Unfortunately, these works were not flexible enough to model non-linear dynamics or did not explicitly capture and transfer the complex latent dependencies needed to perform domain adaptation of time-series data.

In this paper, we address this problem with a model that learns temporal latent dependencies (i.e. dependencies between the latent variables across timesteps) that can be transferred across domains that experience different distributions in their features. We draw inspiration from the Variational Recurrent Neural Network (Chung et al. (2016)) and use variational methods to produce a latent representation that captures underlying temporal latent dependencies. Motivated by the theory of domain adaptation (Ben-David et al. (2010)), we perform adversarial training on this representation

*: Co-first authors

Figure 1: A Story of Temporal Dependency and Domain Invariance



t-SNE projections for the latent representations of DNN, R-DANN, and our VRADA model. We show adaption from Adult-AHRF to Child-AHRF data. Source data is represented with red circles and target data with blue circles. From left to right, one can see that domain adaptation results in mixing the source and target domain data distributions. We can also see a story of how encoding more temporal dependency into the latent representation induces more domain-invariant representations. As models capture more underlying factors of variation, post domain adaptation representations gradually smoothen and become evenly dispersed, indicating that temporal dependency acts synergestically with domain adaptation.

similarly to the Domain Adversarial Neural Network (DANN) (Ganin et al. (2016)) to make the representations invariant across domains. We call our model the Variational Recurrent Adversarial Deep Domain Adaptation (VRADA) model. As far as we know, this is the first model capable of accomplishing unsupervised domain adaptation while transferring temporal latent dependencies for complex multivariate time-series data. Figure 1 shows an example of the domain invariant representations learned by different deep learning models including our VRADA model. From this figure, we can see that our model (VRADA) shows better mixing of the domain distributions than the competing models indicating that it learns better domain invariant representations.

In order to prove the efficacy of our model, we perform domain adaptation using real-world healthcare time-series data. We choose healthcare data for two primary reasons. (1) Currently, a standard protocol in healthcare is to build, evaluate, and deploy machine learning models for particular datasets that may perform poorly on unseen datasets with different distributions. For example, models built around patient data from particular age groups perform poorly on other age groups because the features used to train the models have different distributions across the groups (Alemayehu & Warner (2004); Lao et al. (2004); Seshamani & Gray (2004)). Knowledge learned from one group is not transferrable to the other group. Domain adaptation seems like a natural solution to this problem as knowledge needs to be transferred across domains which share features that exhibit different distributions. (2) Healthcare data has multiple attributes recorded per patient visit, and it is longitudinal and episodic in nature. Thus, healthcare data is a suitable platform on which to study a model which seeks to capture complex temporal representations and transfer this knowledge across domains.

The rest of the paper is structured as follows. In the following section, we briefly discuss the current state-of-the-art deep domain adaptation approaches. Afterwards, we present our model mathematically, detailing how it simultaneously learns to capture temporal latent dependencies and create domain-invariant representations. In Section 4, we compare and contrast the performance of proposed approach with other approaches on two real-world health care datasets, and provide analysis on our domain-invariant representations.

2 Related Work

Domain adaptation is a specific instance of transfer learning in which the feature spaces are shared but their marginal distributions are different. A good survey on the two has been done in several previous works (Pan & Yang (2009); Jiang (2008); Patel et al. (2015)). Domain adaptation has been thoroughly studied in computer vision(Saenko et al. (2010); Gong et al. (2012); Fernando et al. (2013)) and natural language processing (NLP) (Blitzer (2007); Foster et al. (2010)) applications. Recently, the deep learning paradigm has become popular in domain adaptation (Chen et al. (2012); Tzeng et al. (2015); Yang & Eisenstein; Long & Wang (2015)) due to its ability to learn rich, flexible, non-linear domain-invariant representations. Here, we briefly discuss two deep domain adaptation approaches which are closely related to our proposed model. Domain Adversarial Neural Networks (DANN)

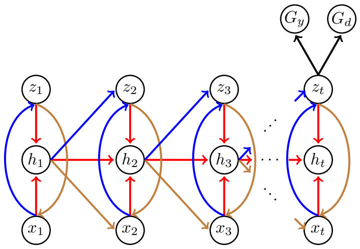


Figure 2: Block diagram of VRADA. Blue lines show the inference process, $q_{\theta_e}(z_t|x_{\leq t}, z_{<t})$. Brown lines show the generation process, $p_{\theta_g}(x_t|z_{\leq t}, x_{<t})$. Red lines show the recurrence process where h_t is informed by h_{t-1}, which is informed by z_{t-1} and x_{t-1}. Black lines indicate classification.

(Ganin et al. (2016)) is a deep domain adaptation model which uses two core components to create domain-invariant representations, a feature extractor that produces the data's latent representation, and an adversarial domain labeler that attempts to classify that data's domain to help the feature extractor produce latent representations which are domain-invariant. In Louizos et al. (2015), the authors propose Variational Fair AutoEncoder, which uses Variational Autoencoding architecture (Kingma & Welling (2013)) to learn latent representations where most of the information about certain known factors of variation are purged from the representation while still retaining as much information about the data as possible. While, these deep learning approaches learn domain-invariant representations, they fail to capture and transfer the underlying complex temporal latent relationships from one domain to another as they use convolutional or feed forward neural networks which we claim are not suitable for multivariate time-series data.

Other works such as Huang & Yates (2009); Xiao & Guo (2013) have used distributed representations for domain adaptation in NLP sequence labeling tasks. However, they either induce hidden states as latent features using dynamic Bayesian networks (DBNs) or learn generalizable distributed representations of words using Recurrent Neural Networks (RNN) (Socher et al. (2011)) to enable domain adaptation. These works either model the highly non-linear dynamics, as one can with RNN, or capture the complex latent dependencies present in sequential data, as one can with DBNs, but not both. To overcome the challenges of DBNs and RNNs, Variational Recurrent Neural Network (VRNN)(Chung et al. (2016)) was proposed recently to capture the complex relationship between the underlying hidden factors of variation and the output variables at different time-steps. The VRNN uses Variational Autoencoders (VAEs)(Kingma & Welling (2013); Goodfellow et al. (2016)) at each time-step to learn a complex relationship between the latent hidden factors across time-steps. Like the VAE, its latent variable is parametric. Combined, these things make it well-suited for multimodal sequential data such as multivariate time-series. In the following section, we discuss our approach, Variational Adversarial Deep Domain Adaptation (VRADA), which uses a VRNN to model and transfer complex domain-invariant temporal latent relationships for unsupervised domain adaptation of multivariate time-series.

3 Variational Recurrent Adversarial Deep Domain Adaptation

In this section, we present our Variational Recurrent Adversarial Deep Domain Adaptation (VRADA) model for the purpose of capturing and transferring temporal latent dependencies across domains via domain-invariant representations. First, we introduce the notations used in this paper and then discuss our VRADA model in detail.

3.1 Notations

Let us denote a multivariate variable-length time series with N data samples as $\{\mathbf{x^i} = (x_t^i)_{t=1}^{T^i}\}_{i=1}^{N}$, where $x_t^i \in \mathbb{R}^D$. (Note: in our experiments, for all data samples $T^i = \tau$, but for generality we maintain T^i). We denote $\{\mathbf{x^i_\mathcal{S}}\}_{i=1}^{n}$ as source domain data and $\{\mathbf{x^i_\mathcal{T}}\}_{i=n+1}^{N}$ as target domain data. We assume that each source domain data sample $\mathbf{x^i_\mathcal{S}}$ comes with L labels $y_i \in \{0,1\}^L$ (for example, these labels may correspond to a clinical outcome such as mortality or ICD9 diagnosis codes), while target domain has no labeled data samples. We assign a domain label $d_i \in \{0, 1\}$ to each data sample to indicate if it comes from the source or target domain. d_i will be used for adversarial training.

3.2 VRADA

The block diagram of our VRADA model is shown in Figure 2. To explicitly model the dependencies between the latent random variable across time steps, the VRADA model utilizes Variational Recurrent Neural Networks (VRNN) (Chung et al. (2016)). The VRNN effectively contains a Variational Auto-Encoders (Kingma & Welling (2013)) at every time step, all of which are conditioned on previous auto-encoders via the hidden state h_{t-1} of an RNN, such as an LSTM (Hochreiter & Schmidhuber (1997)). Therefore, for each time-step of x_t^i, we infer a latent random variable z_t^i via

$$z_t^i | x_t^i \sim \mathcal{N}(\mu_{z,t}, \text{diag}(\sigma_{z,t})), \quad \text{where } [\mu_{z,t}, \sigma_{z,t}] = \varphi_\tau^{enc}(\varphi_\tau^x(x_t^i), h_{t-1})$$

with prior

$$z_t^i \sim \mathcal{N}(\mu_{0,t}, \text{diag}(\sigma_{0,t})), \quad \text{where } [\mu_{0,t}, \sigma_{0,t}] = \varphi_\tau^{prior}(h_{t-1})$$

where $\mu_{*,t}, \sigma_{*,t}$ denote parameters of a generating distribution, and φ_τ^* can be any highly flexible function such as deep neural networks. For each z_t^i, x_t^i is generated via

$$x_t^i | z_t^i \sim \mathcal{N}(\mu_{x,t}, \text{diag}(\sigma_{x,t})), \quad \text{where } [\mu_{x,t}, \sigma_{x,t}] = \varphi_\tau^{dec}(\varphi_\tau^z(z_t^i), h_{t-1})$$

and learned by optimizing the VRNN objective function:

$$\mathcal{L}_r(x_t^i; \theta_e, \theta_g) = E_{q_{\theta_e}(z^i_{\le T^i} | x^i_{\le T^i})}[\sum_{t=1}^{T^i} (-D(q_{\theta_e}(z_t^i | x^i_{\le t}, z^i_{<t}) || p(z_t^i | x^i_{<t}, z^i_{<t})) + \log p_{\theta_g}(x_t^i | z^i_{\le t}, x^i_{<t}))]])$$

where $q_{\theta_e}(z_t^i | x^i_{\le t}, z^i_{<t})$ is the inference model, $p(z_t^i | x^i_{<t}, z^i_{<t})$ is the prior, $p_{\theta_g}(x_t^i | z^i_{\le t}, x^i_{<t})$ is the generative model, θ_e is the parameters of the VRNN's encoder, θ_g the parameters of the VRNN's decoder, and $D(\cdot||\cdot)$ refers to KL-Divergence. Note: $z_{\le T}$ refers to the set of all z_t such that $t \le T$, likewise for $z_{<T}$. For each $\mathbf{x^i}$, we use $\tilde{z}^i \sim q_{\theta_e}(z^i_{T^i} | x^i_{\le T^i}, z^i_{<T^i})$ as our feature representation for source domain classification task since it captures temporal latent dependencies across the time-steps. Training the VRNN for the source domain classification involves solving the following optimization:

$$\min_{\theta_e, \theta_g, \theta_y} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{T^i} \mathcal{L}_r(\mathbf{x^i}; \theta_e, \theta_g) + \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_y(\mathbf{x^i}; \theta_y, \theta_e) + \lambda \mathcal{R}(\theta_e) \tag{1}$$

where $\mathcal{R}(\theta_e)$ is a regularizer for the parameters of VRNN encoder (which is also the feature extractor of VRADA) with a tuning hyperparameter λ.

As we are interested in achieving domain adaptation via the latent representation $\tilde{z}^i$ (i.e. to make $\tilde{z}^i$ domain-invariant), we can adversarially train the above objective function (equation 1) by employing the domain adaptation idea proposed in Ganin et al. (2016). Let $G_y(\tilde{z}^i; \theta_y)$ and $G_d(\tilde{z}^i; \theta_d)$ represent the source label classifier (to predict source labels y_i) and domain label classifier (to predict domain labels d_i) respectively with parameters θ_y and θ_d for a given input $\tilde{z}^i$. Here, $G_y(.)$ and $G_d(.)$ can be deep neural networks. Let us denote their loss functions respectively as

$$\mathcal{L}_y(\mathbf{x^i}; \theta_y, \theta_e) = \mathcal{L}_B(G_y(V_e(\mathbf{x^i}; \theta_e); \theta_y), y_i); \quad \mathcal{L}_d(\mathbf{x^i}; \theta_d, \theta_e) = \mathcal{L}_B(G_d(V_e(\mathbf{x^i}; \theta_e); \theta_d), d_i)$$

where $\mathcal{L}_B$ is the classification loss such as a binary or categorical cross-entropy loss function and $V_e(\mathbf{x^i}; \theta_e)$ is the VRNN encoder that maps input $\mathbf{x^i}$ to $\tilde{z}^i$.

Now, for adversarial training, we consider the following domain adaptation term as the regularizer of equation 1.

$$\mathcal{R}(\theta_e) = \max_{\theta_d} \Big[-\frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_d(\mathbf{x^i}; \theta_d, \theta_e) - \frac{1}{n'} \sum_{i=n+1}^{N} \mathcal{L}_d(\mathbf{x^i}; \theta_d, \theta_e) \Big] \tag{2}$$

where n' is the number of target domain samples. As shown in Ganin et al. (2016), $\mathcal{R}$ is the domain regularizer and it is derived from the empirical $\mathcal{H}-$divergence between the source domain and target domain samples(Ben-David et al. (2010)).

Combining the joint optimization problem of equations 1 and 2 leads to our VRADA model, where we minimize the source classification risk and at the same time achieve domain adaptation. Mathematically, we optimize the following complete objective function:

$$E(\theta_e, \theta_g, \theta_y, \theta_d) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{T^i}\mathcal{L}_r(\mathbf{x^i}; \theta_e, \theta_g) + \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_y(\mathbf{x^i}; \theta_y) - \lambda(\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_d(\mathbf{x^i}; \theta_d) + \frac{1}{n'}\sum_{i=n+1}^{N}\mathcal{L}_d(\mathbf{x^i}; \theta_d))) \tag{3}$$

where λ is a *trade-off* between optimizing on making domain-invariant representations and optimizing source classification accuracy. Our optimization involves minimization with respect to some parameters, and maximization with respect to the others, i.e., we iteratively solve the following:

$$(\hat{\theta}_g, \hat{\theta}_y, \hat{\theta}_e) = \arg\min_{\theta_g, \theta_y, \theta_e} E(\theta_e, \theta_g, \theta_y, \hat{\theta}_d)$$

$$\hat{\theta}_d = \arg\max_{\theta_d} E(\hat{\theta}_e, \hat{\theta}_g, \hat{\theta}_y, \theta_d)$$

with the gradient updates calculated as:

$$\theta_e \leftarrow \theta_e - \eta(\tfrac{\partial \mathcal{L}_r}{\partial \theta_e} + \tfrac{\partial \mathcal{L}_y}{\partial \theta_y} - \lambda\tfrac{\partial \mathcal{L}_d}{\partial \theta_d}) \tag{4}$$

$$\theta_g \leftarrow \theta_g - \eta\tfrac{\partial \mathcal{L}_r}{\partial \theta_g} \tag{5}$$

$$\theta_d \leftarrow \theta_d - \eta\tfrac{\partial \mathcal{L}_d}{\partial \theta_d} \tag{6}$$

$$\theta_y \leftarrow \theta_y - \eta\lambda\tfrac{\partial \mathcal{L}_y}{\partial \theta_y} \tag{7}$$

where η is the learning rate. We can use stochastic gradient descent (SGD) to solve the equations (5-7). To solve equation (4), we can use SGD and the gradient reversal layer (GRL)(Ganin et al. (2016)). The role of GRL is to reverse the gradient sign while performing backpropagation. This ensures that the domain classification loss is maximized which makes the feature representations domain-invariant.

Thus, VRADA results in learning feature representations which are domain-invariant (due to domain regressor $\mathcal{R}$) and which capture the temporal latent dependencies (due to optimizing VRNN objective function $\mathcal{L}_r$). These things combine to allow the VRADAs' discriminative power on the source domain to transfer to the target domain.

4 Experiments

We conduct experiments on two real-world health care datasets to answer the following questions: (a) How does our VRADA model perform when compared to the state-of-the-art domain adaptation and non-adaptation approaches? (b) How different are the domain-invariant representations learned by various domain adaptation methods? (c) How do we show that the temporal latent dependencies are transferred between domains? In the remainder of this section, we will describe the datasets, methods, empirical results, and show visualizations to answer the above questions.

4.1 Dataset Description

We conduct experiments on two health care datasets, including the MIMIC-III dataset and a Pediatric ICU (PICU) dataset from Children's Hospital Los Angeles.

MIMIC-III(Johnson et al. (2016)) is a public dataset with deidentified clinical care data collected at Beth Israel Deaconess Medical Center from 2001 to 2012. It contains over 58,000 hospital admission records of 38,645 adults and 7,875 neonates. For our experiments, we extracted the following two datasets:

- **Adult-AHRF dataset**: To study domain adaptation for adult patients with acute hypoxemic respiratory failure (AHRF), we extracted 20 time series features (such as Base excess, blood pH value, Mean Air Pressure, PaO2, etc.) from 5527 admission records based on Khemani et al. (2009). We grouped the patients into 4 groups/cohorts based on their age[1] - Group 2: working-age adult (20 to 45 yrs, 508 patients); Group 3: old working-age adult (46 to 65 yrs, 1888 patients); Group 4: elderly (66 to 85 yrs, 2394 patients); Group 5: old elderly (85 yrs and up, 437 patients). We treated each group as a separate domain with which we could perform domain adaptation. For each patient, we used the first 4 day after admission (with each day serving as a single time-step) as time series data for training and testing our models.

- **ICD9 dataset**: For this dataset we extracted 99 time series features from 19714 admission records from 4 modalities including input-events (fluids into patient, e.g., insulin), output-events (fluids out of the patient, e.g., urine), lab-events (lab test results, e.g., blood pH values, platelet count, etc.) and prescription-events (drugs prescribed by doctors, e.g., aspirin, potassium chloride, etc.). These modalities are known to be extremely useful for monitoring ICU patients. All the time series are of more than 48 hours of duration, and only the first 24 hours (after admission) 2-hourly sampled time series data is used for training and testing our models. We use this dataset to predict the ICD9 Diagnosis code categories for each patient's admission record.

Child-AHRF dataset: This is a PICU dataset which contains health records of 398 children patient with acute hypoxemic respiratory failure in the intensive care unit at Children's Hospital Los Angeles (CHLA)(Khemani et al. (2009)). Similar to Adult-AHRF, this dataset has 20 time series features collected for 4 days after ICU admission. This dataset is considered as one group (Group 1: children, age 0 to 19 yrs) and represents one domain.

4.1.1 Prediction and Domain Adaptation tasks

Mortality Prediction: For Adult-AHRF and Child-AHRF datasets, we are interested in predicting mortality, i.e. whether a patient dies from AHRF during their hospital stay. 20.10% of all the patients in Child-AHRF and 13.84% of all patients in Adult-AHRF have a positive mortality label (i.e. the patients who die in hospital).

ICD9 Code Prediction: Each admission record in MIMIC-III dataset has multiple ICD-9 diagnosis codes. We group all the occurrences of the ICD-9 codes into 20 diagnosis groups[2]. For the ICD9 dataset, we are interested in predicting these 20 ICD-9 Diagnosis Categories for each admission record. We treat this as a multi-task prediction problem.

Domain Adaptation Tasks: We study unsupervised domain adaptation (i.e. target domain labels are unavailable during training and validation) task with-in age groups of Adult-AHRF dataset, ICD9 dataset and across Adult and Child-AHRF datasets. For Adult-AHRF and ICD9 datasets, we created 12 source-target domain pairs using the age groups, pairing up each domain D_i with another domain $D_{j \neq i}$, for example, the source-target pair 2-5 was used for adapting from group 2 (working-age adult) to group 5 (old elderly). We also created 4 source-target pairs for performing domain adaptation from 4 adult age-groups to 1 child age-group.

4.2 Methods and Implementation Details

We categorize the methods used in our main experiments into the following groups:

- *Non-adaptive baseline methods:* Logistic Regression (LR), Adaboost with decision regressors (Adaboost), and feed forward deep neural networks (DNN)
- *Deep Domain adaptation methods:* Domain Adversarial Neural Networks (DANN) (Ganin et al. (2016)); DANN with a RNN (LSTM) as feature extractor (R-DANN); Variational Fair Autocoder (VFAE)(Louizos et al. (2015))
- *Our method:* Variational Recurrent Adversarial Deep Domain Adaptation (VRADA)[3].

[1]:https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/NationalHealthExpendData/

[2]: http://tdrdata.com/ipd/ipd_SearchForICD9CodesAndDescriptions.aspx. "Conditions Originating in the Perinatal Period" is not present in the preprocessed dataset.

[3]: Codes will be publicly released soon

In all our experiments, we conducted unsupervised domain adaptation where target domain labels are unavailable during training and validation. For R-DANN, we used LSTM(Hochreiter & Schmidhuber (1997)) as the feature extractor network instead of the feed-forward neural networks used in DANN. For VFAE, DANN and all the non-domain adaptive approaches we flattened the time series along time axis and treat it as the input to the model. For fairness, the classifier and feature extractors of the VRADA and R-DANN were equivalent in depth and both had the same model capacity. We also ensure that the size of latent feature representation $\tilde{z}^i$ are similar for VRADA and DANN models. The model capacity of VFAE was chosen to be similar to VRADA. All the deep domain adaptation models including ours had depth of size 8 (including output classifier layers). We used the Adam optimizer (Kingma & Ba (2014)) and ran all models for 500 epochs with a learning rate of $3e{-}4$. We set an early stopping criteria that the model does not experience a decrease in the validation loss for 20 epochs. Source domain data was split into train/validation subsets with a 70/30 ratio and target domain data into train/validation/test subsets with a 70/15/15 ratio. In order to compare all the methods, we report AUC scores on the entire target domain set, and the test subset for each target domain data of a source-target pair.

4.3 Quantitative Results

In Table 1, we compare non domain adaptation and domain adaptation models' performance on the target domain test subset for the AHRF mortality prediction task. It is immediately clear that domain adaptation methods consistently outperform non domain adaptation methods. We see that generally the VRADA outperforms both variants of the DANN with it consistently seeing scores $\sim 4\%$ higher. While the standard deviation for the VRADA was about 1%, it was about 2% for the R-DANN, further showing our models efficacy as it converges to more stable local optima. Our model VRADA beats state-of-the-art DANN(Ganin et al. (2016)) and VFAE(Louizos et al. (2015)) on all the source-pair domain adaptation tasks for Adult-AHRF dataset. For the domain adaptation from Adult-AHRF to Child-AHRF dataset, we observe that VRADA mostly outperforms all the competing models. This shows that our model can perform well even for smaller target domain datasets.

Table 1: AUC Comparison for AHRF Mortality Prediction task with and without Domain Adaptation

Source-Target	LR	Adaboost	DNN	DANN	VFAE	R-DANN	VRADA
3- 2	0.555	**0.562**	0.569	0.572	0.615	0.603	**0.654**
4- 2	0.624	**0.645**	0.569	0.589	0.635	0.584	**0.656**
5- 2	0.527	**0.554**	0.551	0.540	0.588	0.611	**0.616**
2- 3	**0.627**	0.621	0.550	0.563	0.585	0.708	**0.724**
4- 3	**0.681**	0.636	0.542	0.527	0.722	**0.821**	0.770
5- 3	0.655	**0.706**	0.503	0.518	0.608	0.769	**0.782**
2- 4	0.585	**0.591**	0.530	0.560	0.582	0.716	**0.777**
3- 4	**0.652**	0.629	0.531	0.527	0.697	**0.769**	0.764
5- 4	0.689	**0.699**	0.538	0.532	0.614	0.728	**0.738**
2- 5	**0.565**	0.543	0.549	0.526	0.555	0.659	**0.719**
3- 5	0.576	**0.587**	0.510	0.526	0.533	0.630	**0.721**
4- 5	**0.682**	0.587	0.575	0.548	0.712	0.747	**0.775**
5- 1	0.502	**0.573**	0.557	0.563	0.618	0.563	**0.639**
4- 1	**0.565**	0.533	0.572	0.542	**0.668**	0.577	0.636
3- 1	0.500	0.500	0.542	0.535	0.570	0.591	**0.631**
2- 1	**0.520**	0.500	0.534	0.559	0.578	0.630	**0.637**

In the above table, we test classification without adaptation using Logistic Regression (LR), Adaboost with decision tree classifiers and Feed forward Deep Neural Networks (DNN); and with adaptation using Deep Domain Adversarial Neural Networks (DANN), a DANN with an LSTM in its feature extractor (R-DANN), Variational Fair Autoencoder (VFAE) and our Variational Adversarial Domain Adaptation Model (VRADA). All results are reported on the target domain test subset dataset.

As the AHRF mortality prediction task made it clear that domain adaptation is necessary for inter-group adaptation, for the ICD9 multi-task prediction task that involved data with time-steps of length 12, we focused strictly on domain adaptive models (i.e. the DANN, R-DANN, and VRADA). Table 2 shows the aggregated AUC scores on the entire target domain dataset and test data of the target domain for the 20 tasks of the ICD9 Code Prediction task. Here, we clearly see that VRADA and

Table 2: AUC Comparison for ICD9 Diagnosis Code Prediction task

Model		23	24	25	32	34	35	42	43	45	52	53	54
DANN	entire target	0.513	0.508	0.509	0.511	0.508	0.514	0.511	0.507	0.512	0.505	0.508	0.506
	target test	0.509	0.513	0.531	0.527	0.515	0.531	0.515	0.521	0.521	0.518	0.514	0.519
R-DANN	entire target	0.608	0.581	0.562	0.618	0.610	0.586	0.604	0.607	0.575	0.573	0.558	0.566
	target test	0.605	0.579	0.570	0.628	0.609	0.589	0.614	0.616	0.586	0.573	0.563	0.564
VRADA	entire target	0.620	0.564	0.557	0.611	0.617	0.580	0.598	0.615	0.588	0.571	0.582	0.576
	target test	0.609	0.563	0.560	0.620	0.617	0.580	0.606	0.623	0.594	0.576	0.581	0.576

Here, we compare results for the ICD9 Diagnosis Code Prediction task on the ICD9 dataset. For each model, the top row corresponds to the performance on the entire target domain dataset and the bottom row corresponds to performance on the test subset (15%) of the target domain dataset.

R-DANN models outperform DANN Ganin et al. (2016) by significant margins. We also observe that VRADA outperforms R-DANN by $1.5 \sim 2\%$ when averaged over all the source-target domain pairs.

4.4 Discussion

Figure 3 shows the temporal latent dependencies captured by our VRADA as compared to the R-DANN for *3-4* source-target pair. While both models learn temporal latent dependencies fairly well, the VRADA outperforms the R-DANN in two ways. First, the VRADA's neurons learned stronger predictions of whether features are relevant towards modeling the data. If we look at the VRADA row, for both AHRF and ICD9 we see that the neural activation patterns are more consistent across time-steps than for R-DANN. Figure 4 shows the unrolled memory cell states (in the form Examples $\times$ (Time $*$ Neurons)) for all the source and target domain data points. We see a consistent activation firing patterns across all these data points for VRADA but not for R-DANN. Together with the stronger performance on *3-4* for AHRF and *2-5* for ICD9, this potentially indicates that VRADA is better learning the temporal dependencies.

Second, nuanced values are consistent across time-steps for the VRADA, exhibiting a gradual transition towards stronger activation with time, whereas the temporal activation pattern of the R-DANN seems somewhat sporadic. While activation gradients across time are consistent for both the R-DANN and VRADA, more consistent inhibitory and excitatory neuron firing patterns indicate that the VRADA better transfers knowledge. Another indication of domain adaptation was shown in Figure 1c. Looking at the t-SNE projections of feature representations of DNN, R-DANN, and VRADA we can see that the addition of temporal latent dependencies might help in better mixing of the domain distributions since we observe that the data is more evenly spread out. Figure 1c and Figure 3 together indicate that the VRADA's temporal latent dependency capturing power and ability to create domain-invariant representations act synergistically. For plots of activation patterns without domain adaptation, please see appendix section 6.2.3.

5 Summary

Because of its diverse range of patients and its episodic and longitudal nature, healthcare data provides a good platform to test domain adaptation techniques for temporal data. With it as our example, we showcase the Variational Recurrent Adversarial Domain Adaptation (VRADA) model's ability to learn temporal latent representations that are domain-invariant. By comparing our model's latent representations to others', we show its ability to use variational methods to capture hidden factors of variation and produce more robust domain-invariant representations. We hope this work serves as a bedrock for future work capturing and adapting temporal latent representations across domains.

Acknowledgments

This material is based upon work supported by the NSF research grants IIS-1134990, IIS-1254206, Samsung GRO Grant and the NSF Graduate Research Fellowship Program under Grant No. DGE-1418060. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies. We also acknowledge Thailand's Development and Promotion of Science and Technology Talents Project for financial support. We thank Dr. Robinder Khemani for sharing the Child-AHRF dataset.

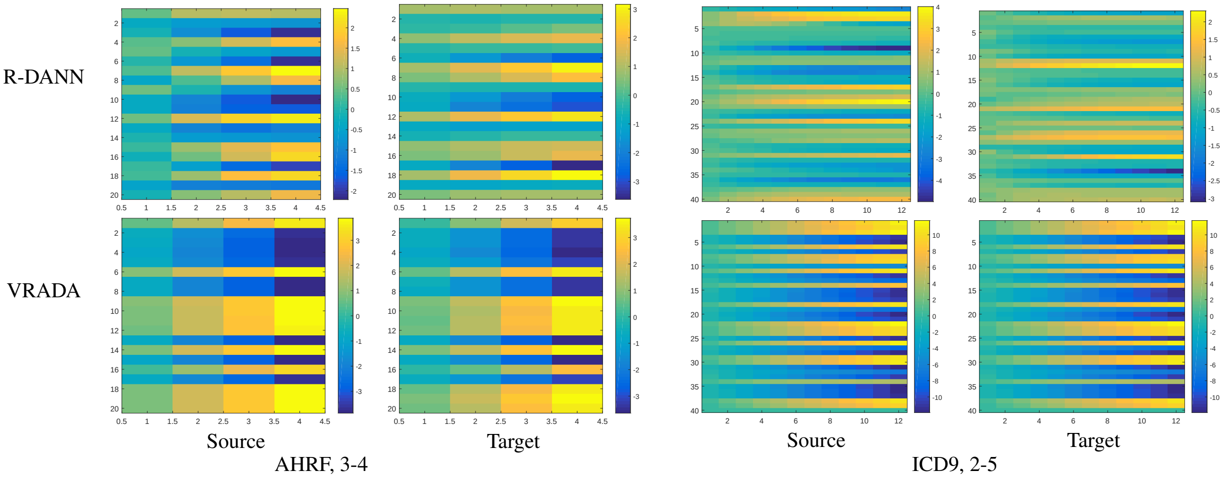


Figure 3: Cell states of memory cell for R-DANN and VRADA showing temporal latent dependencies captured by neurons of the R-DANN and VRADA for the source domain and transferred to the target domain. Each step along the y-axis refers to the activation of a single neuron with blue for strong inhibition and yellow for strong excitation. Step along the x-axis refers to activation per time-step. The left shows a single example in adapting 3-4 and the right for adapting 2-5.

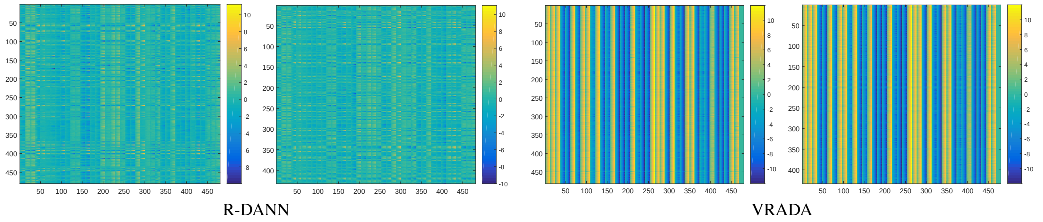


Figure 4: Cell states of memory cell for R-DANN and VRADA showing activation for all ICD9 2-5 adaptation examples. Here, we show temporal dependencies learned across time, feature pairs for examples in a domain. The y-axis values refer to values per data point and the x-axis shows activation at time, feature pairs with the time and feature dimensions being flattened.

References

Berhanu Alemayehu and Kenneth E Warner. The lifetime distribution of health care costs. *Health services research*, 39(3):627–642, 2004.

S Ben-David, J Blitzer, and K Crammer. Analysis of representations for domain adaptation. *Advances in Neural . . .* , pp. 137–144, 2007.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

John Blitzer. *Domain adaptation of natural language processing systems*. PhD thesis, University of Pennsylvania, 2007.

Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation. *arXiv preprint arXiv:1206.4683*, 2012.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron Courville, and Yoshua Bengio. A Recurrent Latent Variable Model for Sequential Data. *arXiv.org*, May 2016.

Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2960–2967, 2013.

George Foster, Cyril Goutte, and Roland Kuhn. Discriminative instance weighting for domain adaptation in statistical machine translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pp. 451–459. Association for Computational Linguistics, 2010.

Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*, 2014.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1), 2016.

Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 2066–2073. IEEE, 2012.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Mit Press, December 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Fei Huang and Alexander Yates. Distributional representations for handling sparsity in supervised sequence-labeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pp. 495–503. Association for Computational Linguistics, 2009.

Jing Jiang. A literature survey on domain adaptation of statistical classifiers. *URL: http://sifaka. cs. uiuc. edu/jiang4/domainadaptation/survey*, 2008.

Jing Jiang and ChengXiang Zhai. Instance weighting for domain adaptation in nlp. In *ACL*, volume 7, pp. 264–271, 2007.

AEW Johnson, TJ Pollard, L Shen, L Lehman, M Feng, M Ghassemi, B Moody, P Szolovits, LA Celi, and RG Mark. Mimic-iii, a freely accessible critical care database. *Scientific Data*, 2016.

Robinder G Khemani, David Conti, Todd A Alonzo, Robert D Bart III, and Christopher JL Newth. Effect of tidal volume in children with acute hypoxemic respiratory failure. *Intensive care medicine*, 35(8):1428–1437, 2009.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv.org*, December 2013.

Zhiqiang Lao, Dinggang Shen, Zhong Xue, Bilge Karacali, Susan M Resnick, and Christos Davatzikos. Morphological classification of brains via high-dimensional shape transformations and machine learning methods. *Neuroimage*, 21(1):46–57, 2004.

Mingsheng Long and Jianmin Wang. Learning transferable features with deep adaptation networks. *CoRR, abs/1502.02791*, 1:2, 2015.

Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard Zemel. The variational fair auto encoder. *arXiv preprint arXiv:1511.00830*, 2015.

Sinno Jialin Pan and Qiang Yang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2009.

Vishal M Patel, Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Visual domain adaptation: A survey of recent advances. *IEEE signal processing magazine*, 32(3):53–69, 2015.

Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pp. 213–226. Springer, 2010.

Meena Seshamani and Alastair M Gray. A longitudinal study of the effects of age and time to death on hospital costs. *Journal of health economics*, 23(2):217–235, 2004.

Table 3: AUC Comparison for AHRF Mortality Prediction task for different types of VRADA training

Training	23	24	25	32	34	35	42	43	45	52	53	54
I	0.704	0.777	0.682	0.540	0.764	0.721	0.603	0.727	0.710	0.616	0.782	0.738
II	0.724	0.656	0.719	0.627	0.748	0.683	0.656	0.770	0.755	0.595	0.736	0.732
III	0.721	0.688	0.656	0.654	0.757	0.691	0.609	0.766	0.775	0.602	0.709	0.714

Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 129–136, 2011.

Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4068–4076, 2015.

Min Xiao and Yuhong Guo. Domain adaptation for sequence labeling tasks with a probabilistic language adaptation model. In *ICML (1)*, pp. 293–301, 2013.

Yi Yang and Jacob Eisenstein. Unsupervised multi-domain adaptation with feature embeddings.

6 Appendix

6.1 Training Variations

We tested 3 variations of training VRADA: (a) training VRADA regularly as discussed in Section 3 (denoted by **I**), (b) loading a pretrained VRNN encoder and optimizing strictly off the classification errors, i.e.

$$E(\theta_e, \theta_y, \theta_d) = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_y(\mathbf{x^i};\theta_y) - \lambda(\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_d(\mathbf{x^i};\theta_d) + \frac{1}{n'}\sum_{i=n+1}^{N}\mathcal{L}_d(\mathbf{x^i};\theta_d))) \tag{8}$$

and (c) loading a pretrained VRNN encoder and using the objective as presented in equation 3 (denoted by **III**). Key to note is that in method **II**, we do not apply variational methods towards learning the shared latent representation. This was done to test whether they were helpful or harmful towards the learned latent representation used for classification. In method **III**, we train VRADA as normal but load a pretrained encoder. We pretrain the encoder by training the VRNN on all source and target domain samples for a desired source-target adaptation pair. In order to choose how many samples would be used for training, we looked at which domain had more examples and chose the larger of the two. For example, if the source domain was group 2 with 508 patients and the target domain was group 5 with 437 patients, the VRNN would see 508 samples of each domain, with group 5 being sampled with replacement after seeing all its samples. As the encoder was used for learning latent representations, we thought it worth investigating whether if pretrained it better captured the latent representations that were being used by the domain classifier for adversarial training. We thought beginning domain classification at a better initialization point might help VRADA avoid local minima. For each method, we fed one source domain sample to G_y and either a source or target domain sample to G_d. (For this training and all training samples, order was randomized.) We only calculated the loss $\mathcal{L}_r$ once for the G_d samples so as to not bias the optimization of the VRNN.

Table 3 shows the results of AHRF Mortality Prediction task for different types of VRADA training. From these experiments, we found that jointly training VRADA (i.e method **I**) usually performed better than the other pretrained training approaches.

6.2 Model Variations

6.2.1 Adversarial training at every time-step

A natural question is whether adversarial training at every time-step is more effective than adversarial training at the last time-step of a latent representation. If done at every time-step, the network learns to create domain-invariant representations of subsets of your input $x_{\leq T}$. Do these domain-invariant representations help the network find more optimal domain-invariant representations of x? We empirically tested this scenario (Table 4) and found the results to be sub-optimal when compared to only performing adversarial training at the last time-step (Table 1). Below are results for the R-DANN and VRADA models for adversarial training at every time-step.

Table 4: AUC Comparison for AHRF Mortality Prediction task with adversarial training done at every time-step

Model	23	24	25	32	34	35	42	43	45	52	53	54
R-DANN	.651	.599	.598	.557	.679	.534	.563	.768	.588	.528	.696	.669
VRADA	.681	.691	.643	.594	.733	.641	.733	.794	.675	.583	.755	.726

6.2.2 Effect of Reconstruction Loss

Table 5 shows the effect of reconstruction loss for our VRADA model. We observe that reconstructing the original data (i.e. using the decoder for reconstructing the data) helps in the overall performance improvement of our VRADA model.

Table 5: AUC Comparison of VRADA model for AHRF Mortality Prediction task with and without reconstruction loss

Model	23	24	25	32	34	35	42	43	45	52	53	54
Without reconstruction	0.703	0.623	0.570	0.647	0.622	0.564	0.577	0.608	0.552	0.599	0.640	0.676
With reconstruction	0.724	0.777	0.719	0.654	0.764	0.721	0.656	0.770	0.775	0.616	0.782	0.738

6.2.3 Impact of Adversarial training

In figures 5 and 6 we show the cell state activations for the VRADA and R-DANN without domain adaptation (i.e. no adversarial training). From these figures, we see that the dependencies between source and target domains are not transferred correctly since we do not perform adversarial training. On the otherhand, as discussed in section 4.4, figure 3 shows that adversarial training helps in transferring the dependencies between source and target domains efficiently.

6.3 R-DANN model information

Here we provide more details on the network architectures of the R-DANN and DANN. Please refer to Figure 7 for a diagram of the R-DANN model showing the dimensions of each layer and the connections between layers. The R-DANN and DANN were essentially identical except that, for the DANN, the first layer used a fully-connected layer instead of an RNN and took input flattened over the time-dimension. Thus the input dimensions corresponded to f and $t \times f$ for the R-DANN and DANN, respectively, where f is the number of features and t is the length of the time-dimension.

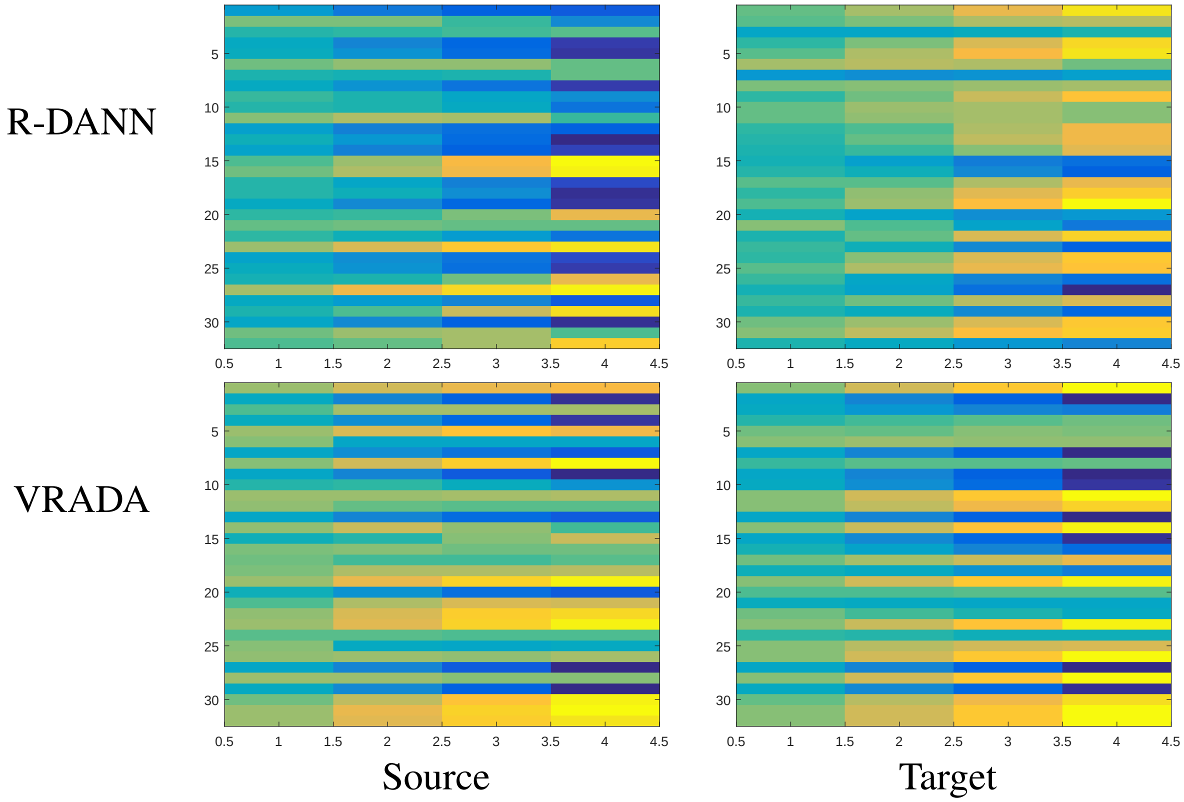


Figure 5: Cell states of memory cell for R-DANN and VRADA showing temporal latent dependencies captured by neurons of the R-DANN and VRADA for the source domain and the target domain. Each step along the y-axis refers to the activation of a single neuron with blue for strong inhibition and yellow for strong excitation. Step along the x-axis refers to activation per time-step. The figure shows a single example in adapting 3-4 for AHRF dataset.

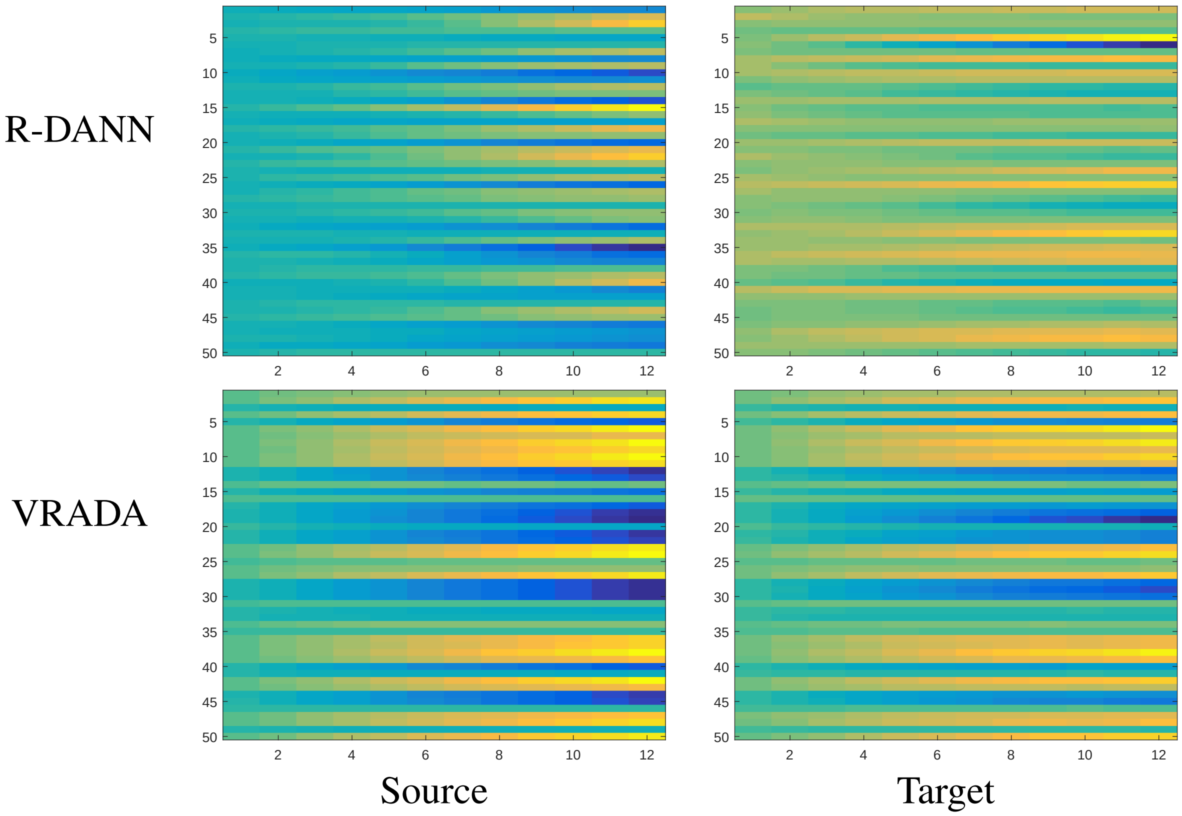


Figure 6: Cell states of memory cell for R-DANN and VRADA showing temporal latent dependencies captured by neurons of the R-DANN and VRADA for the source domain and the target domain. Each step along the y-axis refers to the activation of a single neuron with blue for strong inhibition and yellow for strong excitation. Step along the x-axis refers to activation per time-step. The figure shows a single example in adapting 2-5 for ICD9 dataset.

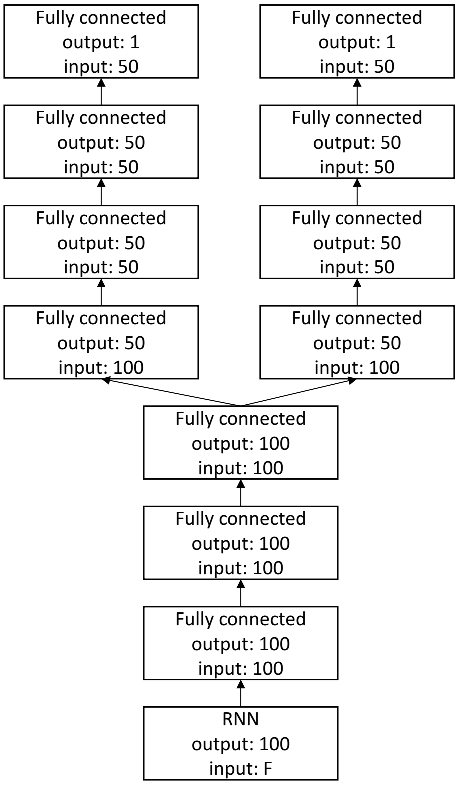


Figure 7: Block diagram of the R-DANN showing the number of neurons used in each layer and how the layers were connected. This model had a capacity of about $46,000$ parameters.