# Discovering Invariances in Healthcare Neural Networks

**Mohammad Taha Bahadori** **Layne C. Price**
Amazon, Seattle, WA
{bahadorm, prilayne}@amazon.com

## Abstract

We study the invariance characteristics of pre-trained predictive models by empirically learning transformations on the input that leave the prediction function approximately unchanged. To learn invariance transformations, we minimize the Wasserstein distance between the predictive distribution conditioned on the data instances and the predictive distribution conditioned on the transformed data instances. To avoid finding degenerate or perturbative transformations, we further regularize by adding a similarity term between the data and its transformed values. Applying the proposed technique to clinical time series data, we discover variables that commonly-used LSTM models do not rely on for their prediction, especially when the LSTM is trained to be adversarially robust.

## 1 Introduction

Understanding the invariant properties of a complex neural network can help analyze its robustness and explain its predictions. Consequently, it can also help us debug the network, detect its unwanted behaviors [9, 7], and gain insights about the model's function that is unique to health data. In this work, we provide a computationally efficient framework for discovering invariances in predictive models that are differentiable with respect to their inputs. We focus on discovering explainable and non-perturbative transformations in the concrete application of intensive care mortality prediction. We show that discovering transformation invariance allows us to find invariance to features and temporal patterns in the data, too.

Our key idea is to learn a transformation that minimizes the distance between a pre-trained model's conditional distribution on the data instances and the conditional distribution on transformed data instances, while simultaneously favoring transformations that are dissimilar from the original data. We choose the Wasserstein distance [18] to simplify the loss function and add a similarity-based regularization term to encourage non-degenerate and non-perturbative solutions. The loss function is differentiable with respect to the parameters of the transformation; thus we can learn it using auto-differentiation and stochastic gradient descent algorithms.

Our framework provides a *global* (dataset-wide) explanation of the model behavior. To quantify the amount of invariance, we measure how different, in terms of correlation coefficient, we can make each feature without significantly changing the predictive distribution. We further show that by choosing a linear transformation in our framework, we can study the impact of varying each feature in the data and assign each feature a normalized score between 0 and 1, assessing the model's sensitivity to that feature. The linear framework allows us to find the invariance of the models to temporal patterns in the data. Finally, we provide an analysis of the linear transformations.

Our experiments on a benchmark in-hospital mortality prediction task [8], defined on the publicly available MIMIC-III dataset [11], highlight the usefulness of the proposed operators. We use our framework to compare the invariances learned by regularly-trained and adversarially-trained LSTM models. Furthermore, we use orthonormal bases to decompose the time series into simple trends and compare the temporal invariance properties of the LSTM and Transformer-based [17] models.

NeurIPS 2019 Healthcare Workshop (ML4H), Vancouver, Canada.

## 2 Invariance Discovery Methodology

Suppose we have a dataset $\mathcal{D} = \{(X_i, y_i)\}$ for $i = 1, \ldots, n$ of feature-label pairs $(X, y) \in \mathbb{X} \times \mathbb{Y}$ and a pre-trained predictive model $p_\theta(y|X) : \mathbb{X} \times \mathbb{Y} \mapsto [0, 1]$, where the real parameters $\theta$ are fixed. For example, the features $X$ might represent a patient's time series of vital signs; the label $y$ can be their status (deceased, alive) at the time of observation; and $p_\theta$ is the predicted probability of the patient's status given their vital signs. Our objective is to find a transformation $T_{\boldsymbol{\phi}} \in \mathbb{T}$, $T_{\boldsymbol{\phi}} : \mathbb{X} \mapsto \mathbb{X}$ such that the prediction of $y$ does not change if it is conditioned on $X$ or $T_{\boldsymbol{\phi}}(X)$. In practice, we are interested in finding a set of transformations parameterized by real values $\boldsymbol{\phi}$. Formally, we find a transformation $T_{\boldsymbol{\phi}}$ by optimizing the parameters $\boldsymbol{\phi}$ such that the following equation is satisfied:

$$p_\theta(y|X) = p_\theta(y|T_{\boldsymbol{\phi}}(X)).$$

To enforce the equality in the above equation, we minimize a distance function $D : [0, 1] \times [0, 1] \mapsto \mathbb{R}_{\geq 0}$ between $p_\theta(y|X)$ and $p_\theta(y|T_{\boldsymbol{\phi}}(X))$. To prevent the trivial solution of $T_{\boldsymbol{\phi}}$ collapsing to the identity transformation, we add a regularization term as follows:

$$\widehat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \; D(p_{\boldsymbol{\theta}}(y|X)) \,,\, p_{\boldsymbol{\theta}}(y|T_{\boldsymbol{\phi}}(X))) + \lambda S(X \,,\, T_{\boldsymbol{\phi}}(X)) \tag{1}$$

where $S : \mathbb{X} \times \mathbb{X} \mapsto \mathbb{R}_{\geq 0}$ is a similarity function and $\lambda > 0$ is the penalization coefficient.

### 2.1 Computationally Efficient Solutions

We make the following choices to simplify the computation of Eq. (1):

**The distance metric $D$:** We use the Wasserstein-1 distance as a robust metric for measuring the distance between two distributions [18, 5, 1]. While there are efficient ways for computing the Wasserstein-1 distance [4, 6], we can use simple estimation schemes in this paper because the mortality prediction task we focus on has a low-dimensional label-space that follows a few simple distributions. Namely, the Wasserstein-1 distance reduces to

$$W_1(p_\theta, p_{\theta'}) = |f_\theta - f_{\theta'}|$$

in the following circumstances: (1) binary classification tasks when we model the output with a Bernoulli distribution $f_\theta(X) \equiv p_\theta(y = 1|X)$; (2) point-mass distributions with $p_{\boldsymbol{\theta}}(y|X) = \delta(y - f_{\boldsymbol{\theta}}(X))$; and (3) univariate regression when we model the output with a Gaussian distribution with a constant, but possibly unknown, variance, $p_{\boldsymbol{\theta}}(y|X) = \mathcal{N}(y|f_{\boldsymbol{\theta}}(X), \sigma_0^2)$. Note that Wasserstein-1 distance does not depend on the observed label $y$ for any of these cases and that the definition of $f_\theta$ is circumstance-dependent.

**The similarity function $S$:** The similarity term encourages the discovery of an invariance transformation that is significantly different from the the identity transformation, assuming one exists. In this work, we choose the cosine similarity between $X$ and $T_\phi(X)$ as it is a normalized measure and allows us to define a convenient threshold for similarity. Moreover, it does not depend on the magnitude of the transformations. For example, penalization with cosine similarity will not shrink the magnitude of the parameters of a linear transformation. For numerical stability purposes, we define the cosine similarity as $\cos(X, T_{\boldsymbol{\phi}}(X)) = (X^\top T_{\boldsymbol{\phi}}(X))/(\|X\|_2(\|T_{\boldsymbol{\phi}}(X)\|_2 + \epsilon))$, where $\epsilon$ is a small number.

Given the choices of $D$ and $S$, the transformation-agnostic loss function is obtained as the empirical expectation of Eq. (1):

$$\widehat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \big[|f_{\boldsymbol{\theta}}(X_i)) - f_{\boldsymbol{\theta}}(T_{\boldsymbol{\phi}}(X_i))| + \lambda \cos\left(X_i, T_{\boldsymbol{\phi}}(X_i)\right)\big]. \tag{2}$$

**The residual transformation function $T_{\boldsymbol{\phi}}$:** In clinical time series analysis, the features can be represented as $X = \{\mathbf{x}_t \in \mathbb{R}^d | t = 1, \ldots, T\}$. Thus, we use two residual network blocks with one-dimensional convolutional networks to model $T_{\boldsymbol{\phi}}$. We will call this transformation "*Residual Transformation*" and provide the details of its architecture in Appendix C. To interpret the learned transformation, we can compute the correlation coefficient between the original and transformed input data for each feature. If a feature can be transformed to a less correlated version with its original version without a significant change in $p_\theta(y|X)$, we conclude that the model is insensitive to the feature.

**The gating transformation function $T_{\boldsymbol{g}}$:** We also study simple linear transformations of the data, not only because we can obtain theoretical insights about the solution, but also use it to understand the sensitivities of the learning algorithms. The linear transformation, called "*Gating Transformation*", is described as $T_{\boldsymbol{g},\boldsymbol{b}}(\mathbf{x}_t) = \mathbf{x}_t \odot \boldsymbol{g} + \boldsymbol{b}$, where $\boldsymbol{g} \in [0,1]^d, \boldsymbol{b} \in \mathbb{R}^d$, and $\odot$ denotes the element-wise product. This transformation performs a soft joint-variable selection on the input features. Each element $g_j$ for $j = 1, \ldots, d$ denotes the sensitivity of the model to the $j^{\text{th}}$ feature in the time series. Since the elements of $\boldsymbol{g}$ are between 0 and 1, the interpretation of these coefficients is easier. We call $g_j$ the *sensitivity* of the corresponding feature $x_j$.

**Analysis of temporal patterns in the data:** We use the gating transformation function to gain more insights into the invariance of the model to different temporal patterns in the time series. To this end, we select a set of $K$ orthonormal temporal basis functions and decompose each dimension of the original time series $\boldsymbol{x}_j = \{x_{jt}\}_{t=1}^T$ in terms of $K$ bases. In this case, the transformation $T_{\boldsymbol{g},\boldsymbol{b}}$ for each feature will map from the $K$-dimensional space of the decomposition back to the 1-dimensional input space, i.e., the transformation $T_{\boldsymbol{g},\boldsymbol{b}}$ is $d$ linear maps each with a $K \times 1$ dimensional weight vector.

For analysis of the non-stationary temporal patterns in clinical time series, we use two sets of orthonormal basis functions: (1) Chebyshev polynomials of the first kind for analysis of a model's invariance to the mean, linear, quadratic, and the residual trends, and (2) pulse waves to analyze the impact of time of the events in the model. Figure 4 in the appendix visualizes these basis functions.

### 2.2 Discovering Variable Invariance – Simple Analysis

Our first observation is about the correctness of the algorithm.

**Proposition 1.** *If $p_\theta$ is invariant to the $j^{\text{th}}$ feature $X_j$, then the solution of Eq. (2) satisfies $g_j^* = 0$.*

*Proof.* According to the definition of invariance, if $p_\theta$ is invariant to the $j^{\text{th}}$ feature, then $\partial p_\theta(y|X)/\partial X_j = 0$. Using the chain rule, we can show that $\partial |f_{\boldsymbol{\theta}}(X)) - f_{\boldsymbol{\theta}}(\boldsymbol{g} \odot X)|/\partial g_j = 0$. Thus the gradient of the loss function in Eq. (2) only depends on the cosine similarity term. Given the constraint $g_j \in [0,1]$, the minimum of the cosine similarity term occurs at $g_j = 0$. □

To gain insights into the structure of the solution, we study the solution of the following simplified problem, which is designed with the goal of obtaining a closed-form solution, while being similar to the original problem. Given the homogeneous linear regression $y = \boldsymbol{\beta}^\top \mathbf{x}$, we look to find $\boldsymbol{g}$ in terms of $\boldsymbol{\beta}$ and $\{\mathbf{x}_i\}_{i=1}^n$. Instead of the Wasserstein-1 loss considered in Eq. (2), we further simplify the problem by solving the squared loss and inner product as follows:

$$\boldsymbol{g}^* = \min_{\boldsymbol{g}} \frac{1}{n} \sum_{1}^{n} \left\{ (\boldsymbol{\beta}^\top (\mathbf{x}_i - \boldsymbol{g} \odot \mathbf{x}_i))^2 + \lambda (\boldsymbol{g} \odot \mathbf{x}_i)^\top \mathbf{x}_i \right\} \tag{3}$$

**Theorem 1.** *Suppose the features $x_j$ have zero mean with empirical correlation matrix $\boldsymbol{C}_n = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^\top$. The solution of Eq. (3) for $\boldsymbol{g}$ is given as:*

$$\boldsymbol{g}^* = \left[ \mathbf{1} - (\lambda/2) \left( \boldsymbol{B}\boldsymbol{C}_n\boldsymbol{B} \right)^{-1} \text{diag}(\boldsymbol{C}_n) \right]_0^1,$$

*where the diagonal matrix $\boldsymbol{B}$ is defined as $\boldsymbol{B} = \text{diag}(\boldsymbol{\beta})$ and the clamp operator $[\cdot]_0^1$ is an element-wise projection to the interval $[0,1]$.*

The proof is based on finding the global solution of the quadratic function and projecting it to the feasible interval. To find the global optimum, we take the derivative with respect to $\boldsymbol{g}$ and set it to zero. The detailed proof is provided in Appendix B.

**Remarks.**

- If the features are uncorrelated, i.e., $\boldsymbol{C}_n$ is diagonal, then $g_j = [1 - \lambda \beta_j^{-2}/2]_0^1$.
- If the features are uncorrelated and $\beta_j \leq \sqrt{\lambda/2}$, then $g_j^* = 0$. This shows how to control the sparsity of $\boldsymbol{g}^*$ by increasing $\lambda$.
- The theorem implies that smaller (larger) values of the regression coefficient, $\boldsymbol{\beta}$, correspond to smaller (larger) values of sensitivity, $\boldsymbol{g}^*$ . This observation helps us in interpreting and comparing the non-zero values of $\boldsymbol{g}^*$.

- The dependence of $\boldsymbol{g}^*$ on the empirical covariance matrix indicates that features that are correlated will share the sensitivity with each other.

# 3 Experiments

We evaluate the accuracy of the transformations in finding the LSTM's invariances on the benchmark mortality task. We also compare the temporal invariance properties of LSTMs and Transformers.

## 3.1 Data and Training Details

**Dataset and tasks:** We evaluate the proposed algorithm on the in-hospital mortality benchmark task [8], using the publicly available MIMIC-III dataset [11]. The objective of this task is to predict mortality outcome based on a time series of multivariate observational data from patients in the intensive care unit (ICU). This is one of the most commonly studied tasks in healthcare analytics and is widely used in estimating patients' clinical risk and managing costs in hospitals [13].

We extract the patient sequences from the MIMIC-III database and partition the data into training and testing sets, following Ref. [8] for cohort construction. To make the interpretation of the results easier, we treat the ordinal values as real values and represent each ordinal variable with a single real variable. We remove outliers and normalize data as described by Ref. [2].

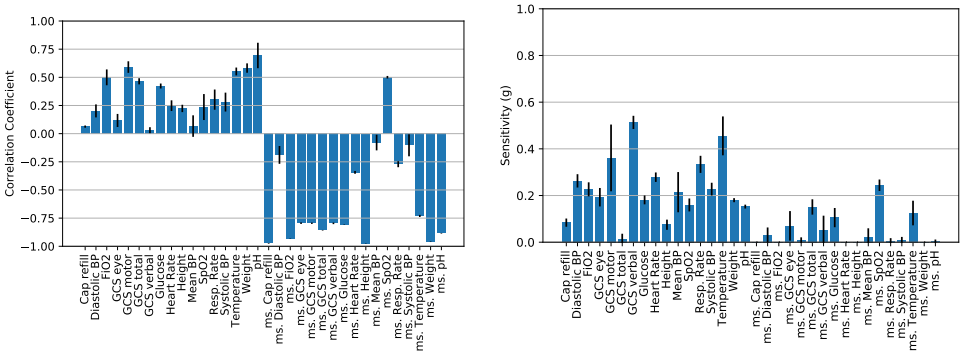
After preprocessing, the features are in the form of multivariate time series of length 60 timestamps. We have 17 features (listed in Table 1 in the appendix) and we add 17 more binary variables that indicate a missing measurement for each of the features. Following Ref. [18], we substitute the missing values with zeros. Thus, the input features are $34 \times 60$ time series and the labels are binary.

We hold out $15\%$ of the training data as a validation set for tuning the hidden layer sizes and hyperparameters. We report the test results based on the best validation performance. For optimization, we use Adam [12] with the AMSGrad modification [16] with batch size of $100$. We halve the learning rate after plateauing for $10$ epochs (determined on validation data) and stop training after the learning rate drops below $5 \times 10^{-6}$.

**Regular and adversarial training of the base predictor $f_{\boldsymbol{\theta}}$:** We choose to train an LSTM network because this is a common benchmark for analysis of clinical time series [13]. We use a two-layered LSTM with 200 hidden neurons in each layer. Given the relationship between adversarial training and invariances [9], we train the model both in a regular (non-adversarial) and adversarial settings. For adversarial training we use the projected gradient descent algorithm [14]. Our goal of using adversarial training is to avoid the possible rediscovery of easy-to-avoid adversarial examples in the transformation $T_\phi$ found by our algorithm. The regular and adversarially trained models achieve test AUC of 0.8600 and 0.8554, respectively, which are comparable to the results reported in Ref. [2].

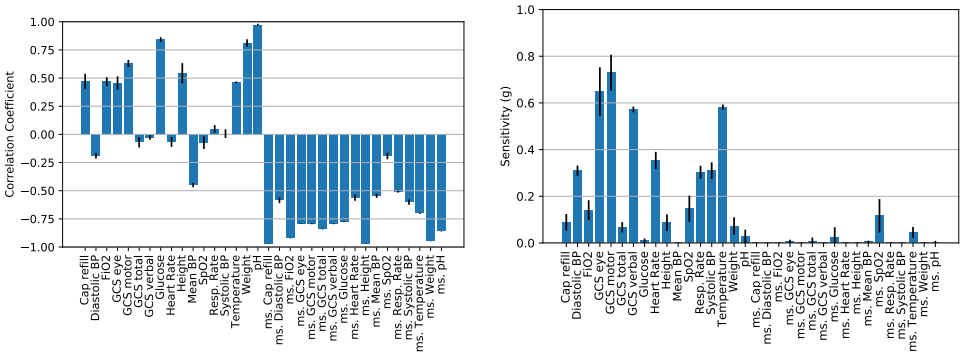
For comparison, we also train a network based on the transformer encoder [17] and convolutional layers (details provided in Appendix C). The regular and adversarially trained transformer-based models achieve test AUC of 0.8539 and 0.8533, respectively. Given the superior performance of LSTM, we present the main experiments only with LSTM and use the transformers only in the temporal patterns analysis for comparison.

**Training of the transformation $T_{\boldsymbol{\phi}}$:** Fixing the parameters of the trained LSTM predictor $f_{\boldsymbol{\theta}}$ as above, we train both the gating and residual transformations, defined above, using Eq. (2). Further implementation details are in the appendix. For the gating transformation, we initialize the transformation matrix as an identity matrix whose elements are perturbed by independent Gaussian noise $\mathcal{N}(0, 0.02)$. We select a value for $\lambda$ and the weight decay to find solutions that satisfy two upper limits on maximum $W_1$ distance (0.05) and cosine similarity (0.5). To capture the training variations because of the random initialization, we repeat the training 24 times each with different random initialization. Then, we pick the top five best trained models based on the validation loss and report the mean and standard deviation.




(a) Residual Transformation (b) Gating Transformation

Figure 1: Discovering the invariance of the *regularly-trained* LSTM model to the features using two types of transformation functions. The correlation coefficient is taken with respect to the data and its transformed values. The missing value indicators are identified with "ms." at the beginning.




(a) Residual Transformation (b) Gating Transformation

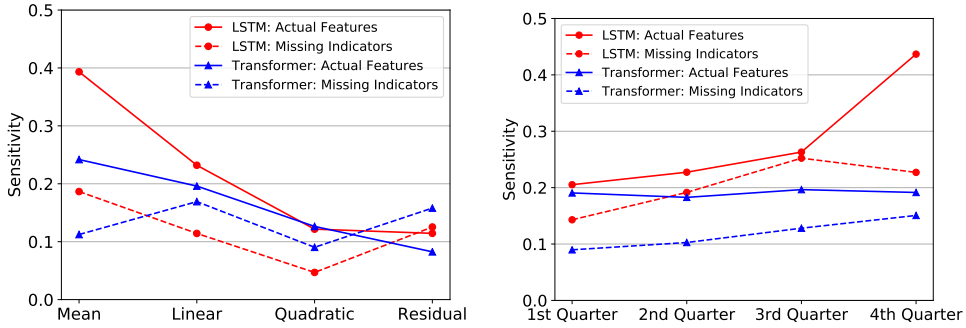
Figure 2: Discovering the invariance of the *adversarially-trained* LSTM model to the features using two types of transformation functions. The adversarially trained model is invariant to more features, including many missing value indicators (identified with "ms." at the beginning.).

### 3.2 Results and Analysis

**Sensitivities of the LSTM.** Figure 1 shows the correlation coefficients and sensitivity numbers obtained by the residual and gating transformations on the regularly trained LSTM model. The full name of the features are provided in Table 1 in Appendix C. Using the residual transformation function, the average correlation coefficients for the actual features and the missing value indicators are $0.3281$ and $-0.5877$, respectively. Similarly, the average sensitivity scores for the actual features and the missing value indicators are $0.2301$ and $0.0485$, respectively. While this shows the significance of the main features, it also underlines the contributions of the missing value indicators.

Looking at the main features we observe that the one of the coma scores has the smallest score, because the total GCS score can be explained by the other GCS scores, as shown in the correlation matrix in Figure 5 in Appendix C. The coefficient for the "height" feature is also small, indicating the minor role of patients' height in the mortality in the intensive care units. The "capillary refill rate" also receives small scores with both methods. Examining the data shows that capillary refill rate is available only for 0.28% of the timestamps; and the learned transformation shows that the feature is not reliable in prediction of the outcome for a large number of patients, as might be expected from its low incidence rate. It is important to note that we have not explicitly trained the LSTM with any regularization criteria that enforce sparsity or variable selection.

We see from the results that the missing value indicator for oxygen saturation $\mathrm{SpO}_2$ is exceptionally important. Clinically, the absence of $\mathrm{SpO}_2$ may not inform providers if the patient is receiving enough oxygen to prevent hypoxemia and tissue hypoxia. We further observe that the mean arterial




(a) Chebyshev basis

(b) Pulse basis

Figure 3: Identifying the influential temporal trends in the *regularly-trained* networks' decisions. The numbers are the average over all features. (a) We decompose the time series to mean, linear, quadratic, and residual trends. The plot shows that the mean and linear trends in the data play a bigger role in the networks' decisions. (b) Using the pulse basis (Figure 4b), we evaluate the invariance of the model to the information in four time periods of the time series. Note that overall more recent information, especially in the actual features, play a bigger role in the LSTM's decisions compared to the Transformer-based network.

blood pressure (MAP) is less important compared to the systolic (SBP) and diastolic (DBP) pressures. While the systolic and diastolic pressure measurements are ubiquitous, a direct measurement of the MAP is invasive; and it can also typically be estimated as $\mathrm{MAP} = (2\mathrm{DBP} - \mathrm{SBP})/3$, which is a common clinical rule-of-thumb [3].

**The impact of adversarial training.** Figure 2 shows the correlation coefficients and sensitivity results by the residual and gating transformations on the adversarially trained LSTM. To have a fair comparison between the two models, we ensure that the Wasserstein loss term for both the regular and adversarially trained models are equal to $0.05$. In the adversarially trained model, the average correlation coefficients for the actual features and missing value indicators are $0.2845$ and $-0.7270$, respectively. Similarly, the average sensitivity scores for the actual features and missing value indicators are $0.2615$ and $0.0120$, respectively. The results show that, by using adversarial training, the model no longer depends on many of the missing value indicators. Notice that the sensitivity scores for features such as "GCS total" have further decreased in comparison to Fig. 1. This experiment is in line with the finding of Ref. [10], which argued that the perturbation-based robustness can create new invariances in the model.

**The impact of trends.** To analyze the contribution of non-stationary trends in time, we decompose each individual time series using the first three Chebyshev polynomials (Fig. 4a), combined with the residual value between the time series and these three basis elements. Figure 3a shows the sensitivity to each component, averaged over all features. It shows that the lower order trend of the time series play the biggest role in the LSTM's decisions. Similar trend can be seen in the Transformer-based model.

**The impact of temporality.** Using the pulse basis in Fig. 4b, we evaluate the invariance of the model to the information in four distinct time periods of the time series. Again, we average the sensitivity scores over all features. Figure 3b shows that overall more recent information, especially in the actual features, play a bigger role in the LSTM's decisions. This can be due to either physiological reasons or an LSTM's structural sensitivity to more recent inputs. In contrast to the LSTM, the Transformer-based model pays equal attention to all time periods on the actual features.

## 4 Discussion

Discovering invariances is closely related to discovering adversarial examples [9]. To discover perturbative adversarial examples, many common algorithms maximize the change in the conditional probability with minimum perturbations in the input. In contrast, in this work we focus on invariance-based examples by keeping the change in the conditional probability minimal, while maximizing

the allowed transformations of the input that have the same predictive power. Unlike [9], we do not use a new model $p_{\boldsymbol{\theta}}$, rather, we find the invariances in a given pretrained $p_{\boldsymbol{\theta}}$. Finding large allowed transformations in the data allows us to better understand the contributions of the individual features in clinical time series data.

## Acknowledgment

The authors thank Daniel Navarro, RN, for discussions about our results from a clinical prospective.

## References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. *arXiv:1701.07875*, 2017.

[2] M. T. Bahadori and Z. C. Lipton. Temporal-clustering invariance in irregular healthcare time series. *arXiv:1904.12206*, 2019.

[3] W. Brzezinski. Blood pressure. In *Clinical Methods: The History, Physical, and Laboratory Examinations. 3rd edition.*, 1990.

[4] M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *NIPS*, 2013.

[5] M. Cuturi and A. Doucet. Fast computation of wasserstein barycenters. In *ICML*, pages 685–693, 2014.

[6] C. Frogner, C. Zhang, H. Mobahi, M. Araya, and T. A. Poggio. Learning with a wasserstein loss. In *NIPS*, pages 2053–2061, 2015.

[7] D. Gopinath, A. Taly, H. Converse, and C. S. Pasareanu. Finding invariants in deep neural networks. *arXiv:1904.13215*, 2019.

[8] H. Harutyunyan, H. Khachatrian, D. C. Kale, G. Ver Steeg, and A. Galstyan. Multitask learning and benchmarking with clinical time series data. *Scientific Data*, 6(1):96, 2019.

[9] J.-H. Jacobsen, J. Behrmann, R. Zemel, and M. Bethge. Excessive invariance causes adversarial vulnerability. In *ICLR*, 2019.

[10] J.-H. Jacobsen, J. Behrmannn, N. Carlini, F. Tramer, and N. Papernot. Exploiting excessive invariance caused by norm-bounded adversarial robustness. *arXiv:1903.10484*, 2019.

[11] A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark. MIMIC-III, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[13] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzel. Learning to diagnose with LSTM recurrent neural networks. In *ICLR*, 2016.

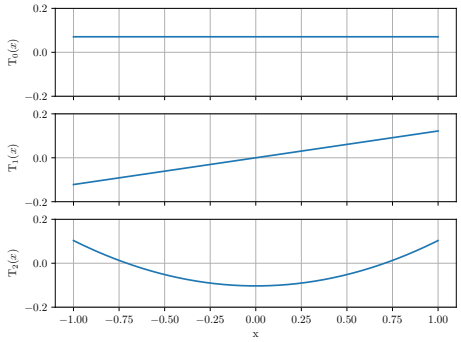
[14] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2019.

[15] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In *NIPS-W*, 2017.

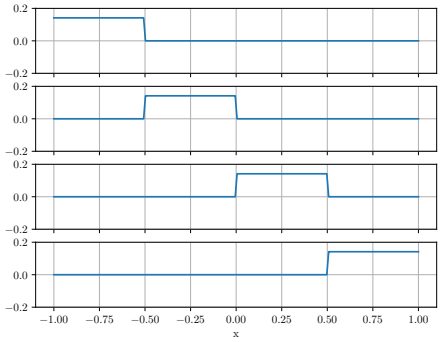
[16] S. J. Reddi, S. Kale, and S. Kumar. On the convergence of adam and beyond. In *ICLR*, 2018.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[18] C. Villani. *Optimal transport: old and new*, volume 338. Springer, 2008.



(a) Chebyshev bases of first kind



(b) Pulse basis functions

Figure 4: (a) The first three Chebyshev polynomials of first kind, capturing the mean, linear, and quadratic trends in the data. We also use the residual of the time series as the fourth dimension. (b) The pulse basis functions.)

# A Appendix

# B Proof of Theorem 1

*Proof.* We note that the unconstrained objective function is a quadratic function in terms of $\boldsymbol{g}$. Thus, we can find the solution to the constrained by finding the global solution of the quadratic function and projecting it to the feasible interval. To find the global optimum, we take the derivative with respect to $\boldsymbol{g}$ and set it to zero.

$$\nabla_{\boldsymbol{g}} \frac{1}{n} \sum_{1}^{n} \left\{ (\boldsymbol{\beta}^\top (\mathbf{x}_i - \boldsymbol{g} \odot \mathbf{x}_i))^2 + \lambda (\boldsymbol{g} \odot \mathbf{x}_i)^\top \mathbf{x}_i \right\} = \mathbf{0},$$

$$\frac{1}{n} \sum_{1}^{n} -2(\boldsymbol{\beta}^\top (\boldsymbol{q} \odot \mathbf{x}_i))(\boldsymbol{\beta} \odot \mathbf{x}_i) + \lambda \mathbf{x}_i^2 = \mathbf{0},$$

where $\mathbf{x}_i^2$ denotes a vector whose elements are $x_{ij}^2$ for $j = 1, \ldots, d$. Given that $\mathbf{x}$ is zero mean with covariance matrix $\boldsymbol{C}_n$, defining $\boldsymbol{q} = \mathbf{1} - \boldsymbol{g}$ and simple algebraic reordering results in

$$\frac{1}{n} \sum_{1}^{n} (\boldsymbol{q}^\top (\boldsymbol{\beta} \odot \mathbf{x}_i))(\boldsymbol{\beta} \odot \mathbf{x}_i) = (\lambda/2)\, \mathrm{diag}(\boldsymbol{C}_n)$$

Reformulating the above equation as a linear system of equations in terms of $\boldsymbol{q}$, we obtain the following solution for the unconstrained problem:

$$\boldsymbol{g} = \mathbf{1} - (\lambda/2)\, (\boldsymbol{B}\boldsymbol{C}_n\boldsymbol{B})^{-1} \mathrm{diag}(\boldsymbol{C}_n).$$

The final result is the projection of the above solution to the valid solutions interval of $[0, 1]$. □

# C Details of the Models

**Implementation of the gating transformation** We use PyTorch [15] for implementing our algorithm. The transformation function is implemented as follows:

```
Conv1d(in_channels=34*e, 34, kernel_size=1, groups=34, stride=1, padding=0,
       dilation=1, bias=True, padding_mode='zeros')
```

where $e = 4$ if we use the basis functions and $e = 1$ otherwise.

We use the proximal algorithm to enforce the condition on $\boldsymbol{g}$ variables by clamping the coefficients back into $[0, 1]$ after every optimization step.

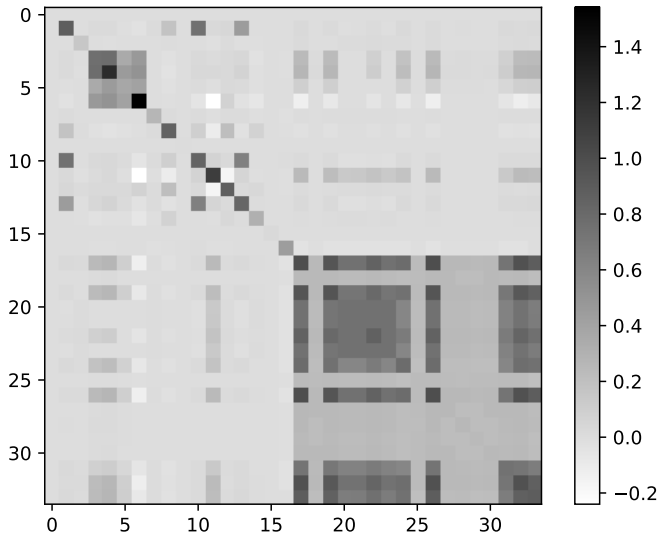



Figure 5: The empirical correlation matrix of the training data.

Table 1: Names of the variables used in the MIMIC-III benchmark [8]

| **Variable Index** | **Description** | **Abbreviation in Figs 1 & 2** |
|---|---|---|
| 1 | Capillary refill rate | Cap Refill |
| 2 | Diastolic blood pressure | Diastolic BP |
| 3 | Fraction inspired oxygen | $\mathrm{FiO}_2$ |
| 4 | Glascow coma scale eye opening | GCS eye |
| 5 | Glascow coma scale motor response | GCS motor |
| 6 | Glascow coma scale total | GCS total |
| 7 | Glascow coma scale verbal response | GCS verbal |
| 8 | Glucose | Glucose |
| 9 | Heart Rate | Heart Rate |
| 10 | Height | Height |
| 11 | Mean blood pressure | Mean BP |
| 12 | Oxygen saturation | $\mathrm{SpO}_2$ |
| 13 | Respiratory rate | Resp. Rate |
| 14 | Systolic blood pressure | Systolic BP |
| 15 | Temperature | Temperature |
| 16 | Weight | Weight |
| 17 | pH | PH |

**Implementation of the residual transformation function** The residual transformation function consists of a cascade of two residual blocks in the form of `y = x + conv2(relu(bnorm(conv1(x))))`, where `conv1 = Conv1d(50, 150, kernel_size=5, padding=2, bias=False)` and `conv2 = Conv1d(150, 50, kernel_size=5, padding=2)`.

**Implementation of the Transformer-based model** We first use `nn.Conv1d(34, 50, kernel_size=5)` to embed the time series into a 50 dimensional space. We add positional encoding transformed by a linear function to the embedded input. The result is processed by `TransformerEncoderLayer(d_model=50, nhead=5, dim_feedforward=50, dropout=0.1)` with layer normalization after each layer. Finally, we use a `nn.Conv1d(50, 1, kernel_size=1)` followed by global average pooling to estimate the logit.