ProtICU: An Interpretable Predictive Model for Healthcare using Prototype Network

Abstract

Recent developments in predictive modelling has seen impressive predictive performances of Deep Neural Networks (DNN) across a multitude of tasks including healthcare. However, due to the large number of parameters distributed across many variables, the DNN architecture is often difficult to interpret. The issue of interpretability is especially important in domains where the justification to one's decision is almost as important as the decision itself (e.g. healthcare, criminal justice). To address this problem, I developed ProtICU, an interpretable model that uses similarity between patches in a testing example and class-specific prototypes as explanations to its prediction. I implemented this model on a binary classification task of predicting patients' in-hospital mortality given data collected in their first two days of ICU stay. The results show some interpretability of model predictions while having comparable predictive performance with non-interpretable models.

1. Introduction

Patients admitted to the intensive care unit (ICU) often are in critical conditions or have the tendency to develop them at any moment. The ability to predict – timely and accurately – patients' future conditions based on early signs is thus invaluable for informing treatment decision, assuring quality of care, and maximizing use of hospital resources.

The broad effort to collect Electronic Health Record (EHR) in ICUs has catalysed much improvements to predictive modelling of patients' critical conditions. These models varies from 'collapse' models – i.e. models that only use features aggregated along the time axis – as implemented with logistic regression by (Higgins *et al.*, 2007; Moreno *et al.*, 2005; Vincent & Moreno, 2010; Zimmerman *et al.*, 2006), with decision trees by (Kho *et al.*, 2012), as well as machine learning-based models implemented by (He *et al.*, 2014; Jensen *et al.*, 2012; Lasko *et al.*, 2013) to time series models as implemented by (Aczon *et al.,* 2017 ; Ge *et al.*, 2018; Nguyen *et al.,* 2017; Sha & Wang, 2017; Tang *et al.*, 2018) using Deep Neural Networks (DNN).

Among the models listed above, the DNNs, especially those that are aware of the time series nature of the data generally performs better in terms of predictive performance. Which is expected given how deep learning models has been showing impressive predictive performances in other domains (LeCun *et al*., 2015). However, many have argued that predictive performance is not the only defining feature of a successful model. For a model to be truly successful it needs to also be interpretable (Chakraborty *et al*., 2017, Ge *et al.*, 2018). This issue of model interpretability is especially important in domains concerned with high-stakes decision making such as healthcare and criminal justice (Rudin, 2019).

The interpretability problem in the medical domain is most commonly tackled by building models that provide either explanations based on attention-mechanism (Shickel *et al.*, 2019; Zhang *et al.*, 2018; *Choi et al.*, 2016) or explanations based on identification of important features (Maji *et al*., 2020; Popkes *et al*., 2019). The former approach provides interpretability by highlighting the region of time a model pays most attention to when making its prediction. However, it often does not give any explanation to which features the model pays most attention to. In contrast, the later approach only ranks the features that are most activated when making the prediction, hence, losing information of when the event happens.

In this article I propose a model I call *prototypical network implementation for ICU records* (ProtICU) which is a prototype-based model inspired by the architecture of *prototypical part network* (ProtoPNet) in (Chen *et al*., 2019). ProtICU aims to explain its prediction through highlighting the similarity between regions in the data with regions in learned prototypes. This goes further beyond the explanation provided by attention based models as the model not only highlights the time a defining event might have occurred but also shows how the event it pays attention to is similar to a prototypical event it has encountered in training. This explanation is analogous to a doctor explaining their diagnosis by providing examples of prototypical patients they encountered in the past instead of explaining the features of the patient he pays most attention to.

While some recent studies (Mercier *et al.*, 2020; Gee *et al.*, 2019) has used prototypes as explanation of time-series model predictions, both implemented an architecture closer to (Li *et al.*, 2018) which uses autoencoders to decode the latent prototypes. ProtICU is different in how it does not use autoencoders to identify the prototypes. ProtICU is also the first model to apply prototype networks with a one-dimensional CNN as a base.

This method is validated on MIMIC III dataset on the task of predicting in-hospital mortality given data from first 48 hours of ICU stay where I show that ProtICU performs comparably to unconstrained convolutional neural network (CNN) despite it having to also generate explanations.

2. Methods

In this section I propose an interpretable prototype-based deep learning classifier architecture I call ProtICU.

2.1. ProtICU

Similar to the ProtoPNet architecture suggested in (Chen *et al.*, 2019), my architecture is built of three components, the convolutional layers, the prototype layer, and the fully-connected last layer. However, since I am implementing this on a multivariate time series, I use one-dimensional Convolutional Neural Network (1D-CNN) architecture (Oord *et al*, 2016; Alves *et al*, 2018), instead of the more common two dimensional CNN used by ProtoPNet. CNN is used as the base architecture to ProtICU because of invariances to certain transformations that are known in EHR time series (Oh *et al.*, 2018; Bahadori and Lipton, 2019; Bahadori and Price, 2020). These invariance properties make 1D-CNN an appropriate base. The ProtICU architecture is illustrated in Figure 1.

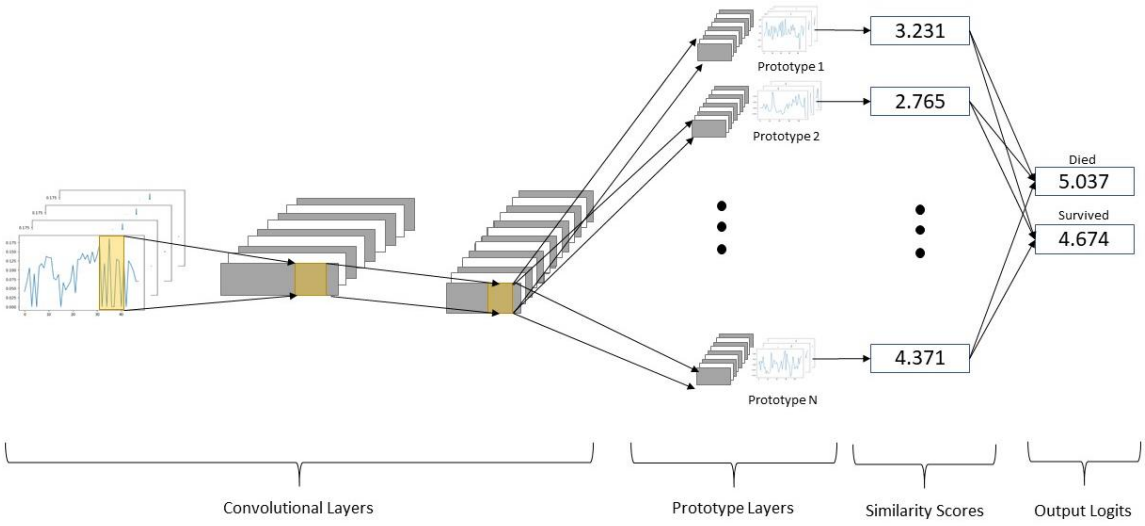

Figure 1. ProtICU Architecture

The convolutional layers consist of multiple one-dimensional convolution and maxpool layer pairs stacked on top of one another where the output of each maxpool operation is activated by the Rectified Linear Unit (ReLU). Atop these convolutional layers, I also added one or two 1 x 1 convolutional layers without maxpooling as suggested by (Chen *et al.l*, 2019). Now, given the input data of dimensions *S x I*, where *S* represents the length of the time series, whilst *I* represents the number of input dimension the convolutional layers accept, the convolutional layer *f* is thus a function that takes an *S x I* real numbered matrix and outputs a real numbered matrix of size *Pa x O*, where *Pa* is the output sequence length (or the number of *patches*) and *O* is the output dimension.

The prototype layer *g* is a function from the space of outputs from the convolutional layers to the space of real vectors with *M* length, where *M* is the number of prototypes. Let *P_1, P_2, … P_M* be *O* dimensional vectors of prototypes. Also, before training, assign every prototype a class such that every class has the same number of prototypes. The function *g*, then, consists of the following operations. Firstly, for every prototype *P_m*, I find out of the *Pa* number of patches in the convolutional outputs, the patch which is closest to the prototype according to their L2 distances. Then, I converted these distances, *d,* into similarities, s, through the equation

$$s = \log((d + 1) / (d + \varepsilon)),$$

where ε is a small number chosen in my case as 0.0001. The resulting similarities *s_1, s_2, … s_M*, are the output activations of this layer. Like other parameters in the network, the weights of the *M x O* prototype matrix are randomly generated and are trained together with other parameters.

The last layer, *h*, is a linear layer with no bias and no non-linear activation. This layer takes the similarities from the previous layers and returns the real logit vector of size C, the number of possible classes.

The rationale behind having the prototype layer and the last layer is to ensure that every prediction is made solely based on the similarities of one data point's latent representation and the latent representations of the same *M* prototypes. This ensures that the parameters that provide explanations (in the form of prototypes) are the exact parameters that are responsible to the predictions.

2.2. Training Regime

Training of the ProtICU model is done using stochastic gradient descent on the entire network before projection of prototypes to one of the training examples. The loss function and the projection operation are described below.

2.2.1. Loss Function

The loss function implemented consists of three parts, the cross-entropy loss, the cluster cost and the separation cost. These three costs represent the three property of the network that ProtICU pays attention to. Firstly, the cross-entropy loss penalises wrong predictions, hence, minimising this loss improves accuracy of the model. However, ProtICU is not only concerned about accuracy, vut also the activation of the correct prototypes by the correct training example. For instance, an example of class *c* should have a latent representation as similar as possible to prototypes of class *c,* but be as different as possible to prototypes that are not of class *c*. The first effect is regulated by the cluster cost *Clust*, while the later by the separation cost *Sep*.

This loss function can be represented concisely as

$$Loss = \frac{1}{n}\sum_{i=1}^{n} CrossEntropy(h \circ g \circ f(x_i), y_i) + \lambda_1 Clust + \lambda_2 Sep$$

$$Clust = \frac{1}{n}\sum_{i=1}^{n} \min_{j:p_j \in P^{(y_i)}} \min_{z \in patches(f(x_i))} \left\|z - p_j\right\|_2^2;$$

$$Sep = -\frac{1}{n}\sum_{i=1}^{n} \min_{j:p_j \notin P^{(y_i)}} \min_{z \in patches(f(x_i))} \left\|z - p_j\right\|_2^2$$

where P(y_i) refers to the set of prototypes assigned to class y_i and the lambdas are constants defining the relative importance between the three terms.

2.2.2. Projection of Prototypes

At the end of training, all training data are passed through the learned convolutional layers *f* to find the *M* different patches (out of the *Pa* x *N_train* patches) that are closest to each of the *M* prototypes given that both the patch and the prototype belong to the same class. Once these patches are found the *M x O* representation are "pushed" to become the new prototypes, while the training data and the location in the sequence corresponding to those *M* new prototypes are saved. These training data are regarded as stereotypical representation of certain features most pertinent to the classification, which is used as explanation to the model's prediction. In the bird species classification example given by (Chen *et al*, 2019), these stereotypical representation could be the beak shape or a feather pattern a certain bird species has as illustrated in (Chen *et al.*, 2019 Figure 2).

It has also been proven that this projection of prototypes operation reduces classification accuracy in a bounded manner given that the model is trained with *Clust* loss (Chen *et al.*, 2019 Supplementary S1).

2.3. Visualisation and Interpretation

In the case of multivariate time series, for every prediction made, this model returns with its prediction the *k* most activated prototypes and the patch of the input data that activates them. Since activation in this case depends on latent space similarities between the prototype and a particular patch, we can interpret that the prediction is made based on how the patch 'looks' similar to the prototype. Further illustration of this idea can be found in the next section.

3. Dataset and Experiments

In this section I present the steps I take to obtain and preprocess the medical time series data as well as the model specification used to illustrate the performance and interpretability of ProtICU.

3.1. Data source and Preprocessing

To illustrate the utility of ProtICU, I consider a sequential binary classification task which is predicting in-hospital mortality of a patient given the electronic medical records of their first 48 hours of ICU stay. I obtain the data from MIMIC III (Johnson *et al.*, 2016).

I used the pipeline developed by (Harutyunyan *et al.*, 2019) to clean and extract features from the MIMIC III databases. This procedure includes removing patients without labels and patients who died or were discharged before 48 hours. This procedure reduced the number of patients to 21,139 out of which 2,797 died in the hospital.

There are 17 features considered in this approach as listed in Table 1. Mean normalisation and masking of missing data is applied as has been suggested by the works of (Lipton *et al.,* 2016; Che *et al.*, 2018). Then, categorical variables are transformed into their respective one-hot vector encoding. This results in a time series input tensor of sequence length *S=48* and input dimensions *I=46* each with a corresponding binary output label indicating in-hospital mortality.

Continuous Variable	**Fraction Missing**
Capillary refill rate	.9965
Diastolic blood pressure	.0964
Fraction inspired oxygen	.9380
Glascow coma scale total	.8164
Glucose	.7404
Heart Rate	.0760
Height	.9960
Mean blood pressure	.1003
Oxygen saturation	.1075
Respiratory rate	.0900
Systolic blood pressure	.0962
Temperature	.6738
Weight	.9689
pH	.8688
Glascow coma scale eye opening	.6907
Glascow coma scale motor opening	.6920
Glascow coma scale verbal opening	.6917

Table 2. Raw predictors to the binary classification problem.

The dataset representing 21,139 patients is then shuffled and split on an 80:10:10 ratio, which generates the data of 14,797 patients for testing, 2,114 for validation, and 2,114 for testing. Four test statistics are collected to assess performance which are area under the receiver operating curve (AUROC), area under precision recall curve (AUPRC), accuracy and F-1 score.

3.2. Experiments

In these experiments, I consider Multi Layers Perceptron (MLP) and 1D-CNN architectures as performance baselines. The MLP model has 4 layers of sizes 1024, 512, 256 and 128. The 1D-CNN is implemented with three convolutional layers of sizes 128, 256, and 512 with kernel sizes 9, 5, and 3, respectively. The maxpool layer of the CNN architecture is of size 2 and the convolutional layers are then connected to a fully connected layer of size 128.

The ProtICU architecture is implemented with the same convolutional layer architecture as the 1D-CNN above with the addition of one 1x1 convolution of size 256. The fully connected layer is replaced by a prototype layer with 5 prototypes for each class.

All the experiments are run with 30 epochs with batch size 256, using learning rate of 0.001, dropout 0.2 and early stopping with patience 2 when validation loss decreases no more than 0.0001 compared to the existing minimum. The losses for the two baselines are the normal cross-entropy and all architectures are activated with ReLU and trained with the Adam optimizer. To get a variance bound around the prediction, I repeat each experiments 10 times.

The codes of the models and visualisation tools used to conduct this experiment are available in https://github.com/kristoforusbryant/ProtICU.

4. Results

From the experiments outlined above I found that ProtICU performs comparably to the 1D-CNN model in all four statistics as shown in Table 2. This shows that the addition of the prototype layer does not reduce predictive performance in a significant manner. Also shown in Table 2 is that both ProtICU and 1D-CNN both outperforms the MLP baseline.

Model Architecture	AUROC	AUPRC	ACCURACY	F1 SCORE
ProtICU	.8116 (.0115)	.4367 (.0274)	**.8271 (.0302)**	.4321 (.2138)
1D-CNN	**.8285 (.0171)**	**.4441 (.0274)**	.8020 (.0325)	**.4520 (.0262)**
MLP	.8080 (.0069)	.4239 (.0117)	.7791 (.0281)	.4671 (.0100)

Table 2. Test Statistics of the different Neural Network Architectures

Atop of having similar performance to 1D-CNN, ProtICU generates explanations to its prediction in the form of prototype comparisons. For instance, Figure 2 visualises an example data of a patient that died in-hospital alongside the (three most activated) prototypical example. The highlighted ranges of the example data and the prototypes shows the pair of regions are similar to each other in latent space.

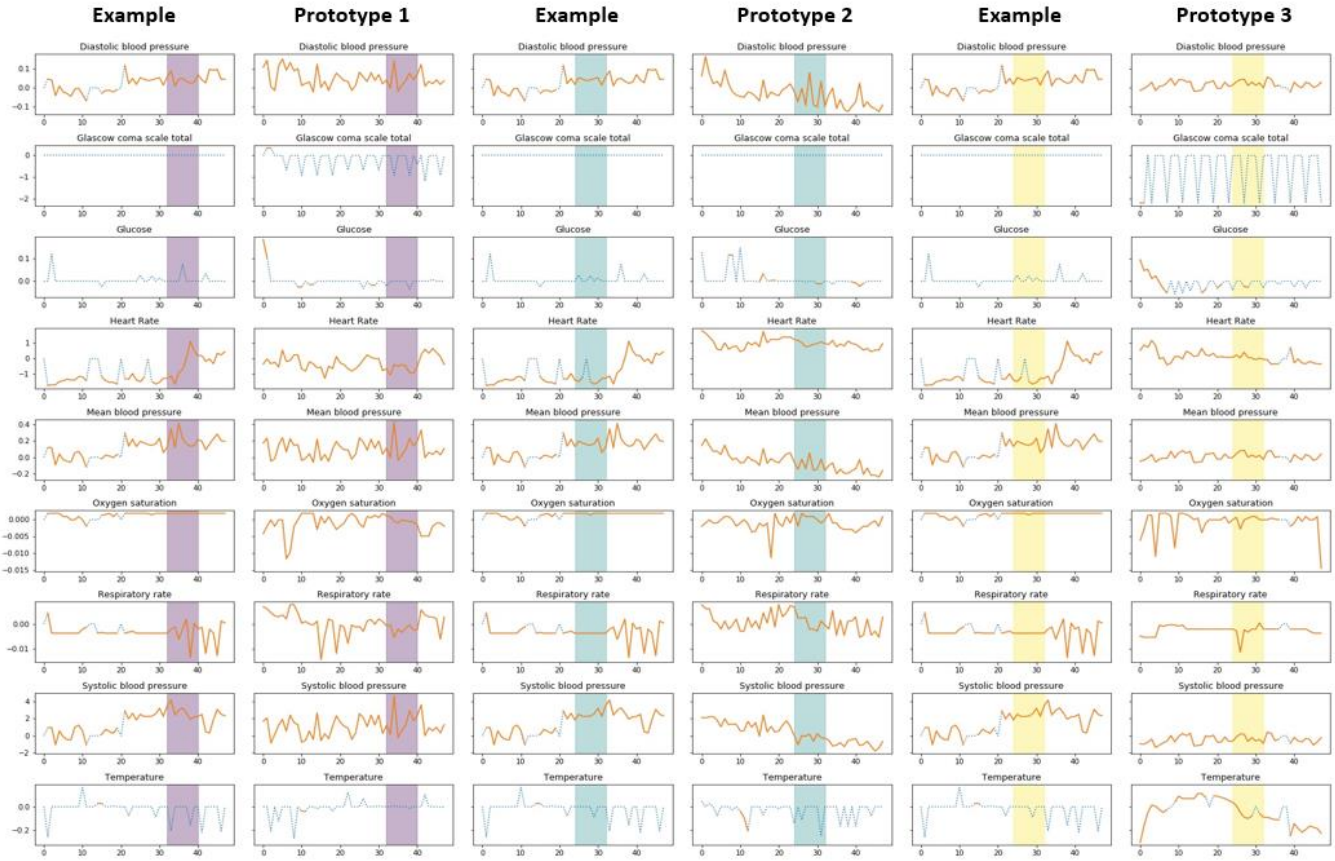

Figure 2. Visualisation examples of a patient who died in-hospital with the 3 best prototypes to explain the example. Solid orange lines represents measured data, while dotted blue lines represents imputed data.

With such visualisation, one can use the similar patches between example and prototypes to interpret why the model predicts how it predicts. For example, in Figure 2, one might interpret the purple regions as a medical event that might help explain the eventual death of the patient. Considering that around this period, the systolic blood pressure spikes, while the diastolic blood pressure remains approximately normal might indicate that the patient is experiencing isolated systolic hypertension around that time, which may contribute to the risk of heart attack and stroke (Petrovitch *et al.*, 1995; Qureshi *et al.*, 2002; Paultre and Mosca, 2005). Admittedly, my interpretation might be shallow and limited as I am not a medical professional, hence the assessment of this form of explanation must be verified as I will explain in the next section.

5. Discussion and Conclusion

In this article, I explained my method of making deep learning model in the field of healthcare more interpretable by showing how patches of the example being predicted is similar to prototypical patches from the training set. I also show how this can be achieved with a small decrease in predictive performance.

That said, several limitations still remain. Firstly, as suggested by (Doshi-Velez and Kim, 2017), I agree that interpretability should be assessed rigorously by conducting validation experiments involving domain experts. This is done to ensure that explanations provided by the models are in line with the current understanding in the field and thus useful to the domain. (Kim *et al*., 2015a) and (Kim *et al*., 2015b) outlines two examples of how to conduct this type of validation experiments. With respect to ProtICU, an example of a validation experiment is to ask several healthcare professional to guess the causes of death of patients with and without the ProtICU explanation. ProtICU method of explaining its prediction will thus be validated if the giving the professional ProtICU explanations leads to significant improvement in guessing performance. Unfortunately, for this dataset causes of death are not provided. Hence, a natural extension is to consider other tasks such as those suggested in (Harutyunyan *et al.*, 2019) and (Wang *et al*., 2020). It will also be interesting to explore the efficacy of this method to multi-class classifier such as the task of predicting ICD diagnosis codes.

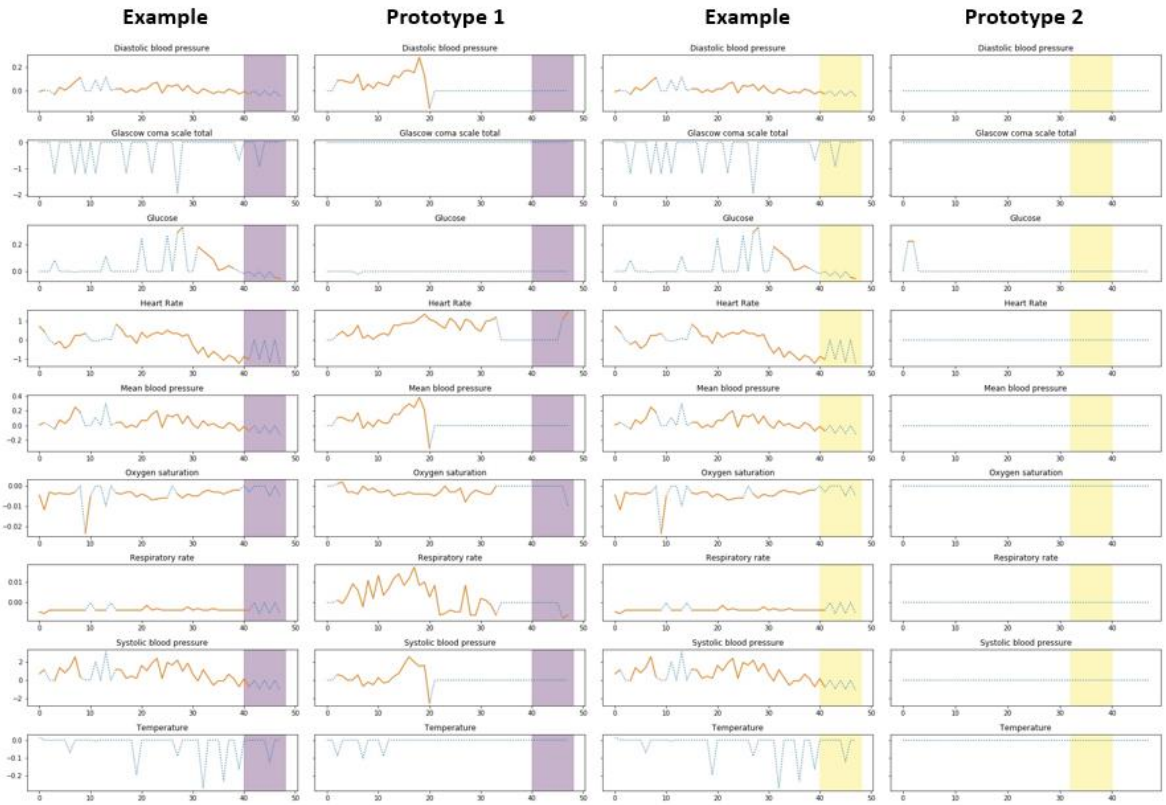

Figure 3. It is unclear from the explanation given here whether the network is explaining the imputed value or the missingness of the data.

Another limitation of this approach stems from the problem of missingness that plague medical records. While the prediction performance is improved by the addition of masks as suggested by (Lipton *et al.,* 2016; Che *et al.*, 2018), this presents a problem when implemented with ProtICU. As shown in Figure 3, one cannot be sure that the region highlighted is deemed similar due to the similarity between the masks or the similarity between the imputed values.

Lastly, it would be of interest to explore ways to extend the prototypes to not only highlight similarity on the time axis, but also on the feature axis. While, in this context, 17 features are still arguably small enough for an expert to interpret, when dealing with larger datasets, explanations may need to be more detailed with respect to which features are similar between the prototypes and the example.

6. Acknowledgements

I would like to thank Feng Xie and Prof Bibhas Chakraborty for helpful discussions and inputs. I would also like to thank The Yale-NUS College Summer Research Programme for their support.

7. References

Aczon, M., Ledbetter, D., Ho, L., Gunny, A., Flynn, A., Williams, J., & Wetzel, R. (2017). Dynamic mortality risk predictions in pediatric critical care using recurrent neural networks. *arXiv preprint arXiv:1701.06675*.

Bahadori, M. T., & Lipton, Z. C. (2019). Temporal-clustering invariance in irregular healthcare time series. *arXiv preprint arXiv:1904.12206*.

Bahadori, M. T., & Price, L. C. (2019). Discovering Invariances in Healthcare Neural Networks. *arXiv preprint arXiv:1911.03295*.

Che, Z., Purushotham, S., Cho, K., Sontag, D., & Liu, Y. (2018). Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, *8*(1), 1-12.

Choi, E., Bahadori, M. T., Sun, J., Kulas, J., Schuetz, A., & Stewart, W. (2016). Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In *Advances in Neural Information Processing Systems* (pp. 3504-3512).

Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Ge, W., Huh, J. W., Park, Y. R., Lee, J. H., Kim, Y. H., & Turchin, A. (2018). An Interpretable ICU Mortality Prediction Model Based on Logistic Regression and Recurrent Neural Networks with LSTM units. In *AMIA Annual Symposium Proceedings* (Vol. 2018, p. 460). American Medical Informatics Association.

Gee, A. H., Garcia-Olano, D., Ghosh, J., & Paydarfar, D. (2019). Explaining deep classification of time-series data with learned prototypes. *arXiv preprint arXiv:1904.08935*.

He, D., Mathews, S. C., Kalloo, A. N., & Hutfless, S. (2014). Mining high-dimensional administrative claims data to predict early hospital readmissions. *Journal of the American Medical Informatics Association*, *21*(2), 272-279.

Higgins, T. L., Teres, D., Copes, W. S., Nathanson, B. H., Stark, M., & Kramer, A. A. (2007). Assessing contemporary intensive care unit outcome: an updated Mortality Probability Admission Model (MPM0-III). *Critical care medicine*, *35*(3), 827-835.

Johnson, A. E., Pollard, T. J., Shen, L., Li-Wei, H. L., Feng, M., Ghassemi, M., ... & Mark, R. G. (2016). MIMIC-III, a freely accessible critical care database. *Scientific data*, *3*(1), 1-9.

Jensen, P. B., Jensen, L. J., & Brunak, S. (2012). Mining electronic health records: towards better research applications and clinical care. *Nature Reviews Genetics*, *13*(6), 395-405.

Kho, A. N., Hayes, M. G., Rasmussen-Torvik, L., Pacheco, J. A., Thompson, W. K., Armstrong, L. L., ... & Bielinski, S. J. (2012). Use of diverse electronic medical record systems to identify genetic risk for type 2 diabetes within a genome-wide association study. *Journal of the American Medical Informatics Association*, *19*(2), 212-218.

Kim, B., Shah, J. A., & Doshi-Velez, F. (2015a). Mind the gap: A generative approach to interpretable feature selection and extraction. In *Advances in Neural Information Processing Systems* (pp. 2260-2268).

Kim, B., Glassman, E., Johnson, B., & Shah, J. (2015b). iBCM: Interactive Bayesian case model empowering humans via intuitive interaction.

Lasko, T. A., Denny, J. C., & Levy, M. A. (2013). Computational phenotype discovery using unsupervised feature learning over noisy, sparse, and irregular clinical data. *PloS one*, *8*(6).

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436-444.

Li, O., Liu, H., Chen, C., & Rudin, C. (2018, April). Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *Thirty-second AAAI conference on artificial intelligence*.

Lipton, Z. C., Kale, D. C., & Wetzel, R. (2016). Modeling missing data in clinical time series with rnns. *arXiv preprint arXiv:1606.04130*.

Maji, S., Bali, R., Ankem, S. H., & Ayyadevara, K. V. (2020). A Simple and Interpretable Predictive Model for Healthcare. *arXiv preprint arXiv:2007.13351*.

Mercier, D., Dengel, A., & Ahmed, S. (2020). P2ExNet: Patch-based Prototype Explanation Network. *arXiv preprint arXiv:2005.02006*.

Moreno, R. P., Metnitz, P. G., Almeida, E., Jordan, B., Bauer, P., Campos, R. A., ... & SAPS 3 Investigators. (2005). SAPS 3—From evaluation of the patient to evaluation of the intensive care unit. Part 2: Development of a prognostic model for hospital mortality at ICU admission. *Intensive care medicine*, *31*(10), 1345-1355.

Nguyen, P., Tran, T., & Venkatesh, S. (2017). Deep learning to attend to risk in ICU. *arXiv preprint arXiv:1707.05010*.

Oh, J., Wang, J., & Wiens, J. (2018). Learning to exploit invariances in clinical time-series data using sequence transformer networks. *arXiv preprint arXiv:1808.06725*.

Paultre, F., & Mosca, L. (2005). Association of blood pressure indices and stroke mortality in isolated systolic hypertension. *Stroke*, *36*(6), 1288-1290.

Petrovitch, H., Curb, J. D., & Bloom-Marcus, E. (1995). Isolated systolic hypertension and risk of stroke in Japanese-American men. *Stroke*, *26*(1), 25-29.

Popkes, A. L., Overweg, H., Ercole, A., Li, Y., Hernández-Lobato, J. M., Zaykov, Y., & Zhang, C. (2019). Interpretable outcome prediction with sparse Bayesian neural networks in intensive care. *arXiv preprint arXiv:1905.02599*.

Qureshi, A. I., Suri, M. F. K., Mohammad, Y., Guterman, L. R., & Hopkins, L. N. (2002). Isolated and borderline isolated systolic hypertension relative to long-term risk and type of stroke: a 20-year follow-up of the national health and nutrition survey. *Stroke*, *33*(12), 2781-2788.

Sha, Y., & Wang, M. D. (2017, August). Interpretable predictions of clinical outcomes with an attention-based recurrent neural network. In *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics* (pp. 233-240).

Shickel, B., Loftus, T. J., Adhikari, L., Ozrazgat-Baslanti, T., Bihorac, A., & Rashidi, P. (2019). DeepSOFA: a continuous acuity score for critically ill patients using clinically interpretable deep learning. *Scientific reports*, *9*(1), 1-12.

Tang, F., Xiao, C., Wang, F., & Zhou, J. (2018). Predictive modeling in urgent care: a comparative study of machine learning approaches. *JAMIA Open*, *1*(1), 87-98.

Vincent, J. L., & Moreno, R. (2010). Clinical review: scoring systems in the critically ill. *Critical care*, *14*(2), 207.

Wang, S., McDermott, M. B., Chauhan, G., Ghassemi, M., Hughes, M. C., & Naumann, T. (2020, April). Mimic-extract: A data extraction, preprocessing, and representation pipeline for mimic-iii. In *Proceedings of the ACM Conference on Health, Inference, and Learning* (pp. 222-235).

Zhang, J., Kowsari, K., Harrison, J. H., Lobo, J. M., & Barnes, L. E. (2018). Patient2vec: A personalized interpretable deep representation of the longitudinal electronic health record. *IEEE Access*, *6*, 65333-65346.

Zimmerman, J. E., Kramer, A. A., McNair, D. S., & Malila, F. M. (2006). Acute Physiology and Chronic Health Evaluation (APACHE) IV: hospital mortality assessment for today's critically ill patients. *Critical care medicine*, *34*(5), 1297-1310.