As easy as APC: Leveraging self-supervised learning in the context of time series classification with varying levels of sparsity and severe class imbalance

Anonymous Authors[1]

Abstract

High levels of sparsity and strong class imbalance are ubiquitous challenges that are often presented simultaneously in real-world time series data. While most methods tackle each problem separately, our proposed approach handles *both* in conjunction, while imposing fewer assumptions on the data. In this work, we propose leveraging a self-supervised learning method, specifically Autoregressive Predictive Coding (APC), to learn relevant hidden representations of time series data in the context of both missing data and class imbalance. We apply APC using either a GRU or GRU-D encoder on two real-world datasets, and show that applying one-step-ahead prediction with APC improves the classification results in all settings. In fact, by applying GRU-D - APC, we achieve state-of-the-art AUPRC results on the Physionet benchmark.

1. Introduction

Rare event prediction on time series data is known to be a difficult challenge on its own (Branco et al., 2015). The difficulty lies in that when classification is performed on a severely imbalanced data set, models are prone to classify most examples into the majority class. This often results in poor predictive performance, specifically for the minority class of interest (Branco et al., 2015), yielding models which do not perform significantly better than naive baselines.

Missing data are also ubiquitous in real-world time-series, making classification on imbalanced data sets all the more difficult. While missing data does result in a loss of available information, the patterns of missingness themselves can also sometimes be informative (Che et al., 2018).

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Successes have been obtained to deal with these obstacles, however most methods tackle each challenge separately; either dealing with properly learning in the context of missingness (Rubanova et al., 2019; Cao et al., 2018; Che et al., 2018) or correctly identifying the minority class(es) in a class imbalance setting (Barandela et al., 2003; Yen & Lee, 2009; Chawla et al., 2002b; Zhu et al., 2018).

Recently, self-supervised pre-training has gained a lot of interest and momentum, especially within the Natural Language Processing (NLP) community, as it has shown to significantly improve performance on downstream supervised learning tasks (Radford et al., 2018; 2019; Brown et al., 2020). In this work, we propose leveraging a self-supervised autoregressive pre-training framework, Autoregressive Predictive Coding (APC), to improve multi-variate time-series classification in the context of both missingness and class imbalance. Specifically, APC (Chung et al., 2019) is a recently proposed representation learning method that learns from sequential data by performing multiple-step-ahead forward prediction in an autoregressive manner, learning discriminative and meaningful representations from unlabeled data.

Here, we propose applying APC on time series data, learning relevant hidden representations which are in turn used as input to the classifier in a secondary training phase. By doing so, we can take advantage of the potentially informative missingness patterns, without making strong assumptions about the values of these missing data. Furthermore, through learning from unlabeled data, the model is able to learn useful representations which are free from potential class imbalance. We apply our approach on two real-world datasets, Physionet and Clue, both containing varying degrees of missing values and class imbalance, and show that applying APC improves the classification results in both cases. Our results indicate that the representations learned through APC allow the classifier to better identify the minority class(es), even in the face of sparsity. In fact, our approach achieves state-of-the-art AUPRC scores on the Physionet benchmark. Our code is available at `https://github.com/anonICML21/APC`.

2. Related work

2.1. Class imbalance

There are several ways to deal with class imbalances. Some of the most common methods consist of data re-sampling, such as under-sampling the majority class (Barandela et al., 2003; Yen & Lee, 2009) and over-sampling the minority class (Chawla et al., 2002b). However, under-sampling can result in losing a significant amount of data if there is a severe class imbalance, while over-sampling methods, such as SMOTE (Chawla et al., 2002a), artificially increase the data set. Another widely adopted method is class weights (Zhu et al., 2018), which does not involve re-sampling, but provides an incentive to properly classify minority class samples by giving them more weights in the loss function.

One method that does not rely on data re-sampling and leverages unsupervised learning to deal with class imbalance is the dual auto-encoders features (DAF) (Ng et al., 2016). DAF uses two stacked auto-encoders to learn different types of features of the data in an imbalanced setting, and it is shown to outperform traditional data re-sampling methods. This indicates that leveraging unsupervised learning to obtain a set of relevant features is a promising approach to deal with class imbalance. Our approach differs from autoencoders in that a time shifting factor $n \geq 1$ allows the APC model to learn more general/global structures of the data rather than only local ones (Chung et al., 2019) .

2.2. Missing data

Traditional methods for handling missing data often involve first filling in the missing values (data imputation), and then applying predictive models on the imputed data (Che et al., 2018). Choosing a suitable imputation scheme is complex, dataset-specific and relies on a good amount of domain expertise. Furthermore, this results in a two-step process where imputation and prediction models are separated, preventing the prediction model from properly exploring missingness patterns (Che et al., 2018). However, (Che et al., 2018) show that the missing values and patterns may incorporate useful information about the target labels; so called *informative missingness*. When it comes to state-of-the-art time series models, there are only a handful that incorporate these missingness patterns when learning from the data. One example is the Bidirectional Recurrent Imputation for Time Series (BRITS) (Cao et al., 2018), a data imputation method which can simultaneously impute the missing values and perform classification/regression within a joint neural graph. Another similar method is the GRU model with trainable Decays (GRU-D) (Che et al., 2018). Both the BRITS and the GRU-D take advantage of two representations of informative missingness patterns: *masking* and *time interval* (Che et al., 2018). Recently, two models based on ordinary differential equations (ODE), the ODE-RNN and the Latent-ODE, have also shown promising results on irregularly-sampled data (Rubanova et al., 2019). However, the computational complexity of these models is high, which (Horn et al., 2019) concluded led to not finding the optimal hyperparameters. Time Series Cluster Kernel (TCK_{IM}) (Mikalsen et al., 2019) learns from missingness patterns in an unsupervised way by computing the similarities between multivariate time series with Gaussian Mixture Models (GMM) extended with informative priors. However, this method cannot handle time series of variable length.

2.3. Self-supervised pre-training

Self-supervised pre-training is a special case of semi-supervised learning where the aim is to learn a good initialization point for the supervised setting instead of changing the supervised learning objective (Radford et al., 2018). Most models used for this purpose are based on autoencoders (Sagheer & Kotb, 2019), but some of the most recent and promising methods are based on the idea of predictive coding, such as Contrastive Predictive Coding (CPC) (Oord et al., 2018) and Autoregressive Predictive Coding (APC) (Chung et al., 2019). While CPC is able to do multi-step forward prediction in a contrastive way, APC does so in an autoregressive manner (Chung et al., 2019).

3. Autoregressive Predictive Coding

Autoregressive Predictive Coding (APC) learns from sequential data by trying to predict $n \geq 1$ steps ahead of the current one (Chung et al., 2019). APC was proposed as a method for speech representation learning and is able to learn transferable representations from large quantities of unlabeled data (Chung & Glass, 2020). Previous work on APC shows that the learned representations significantly improve phonetic classification and other speech recognition benchmarks, outperforming CPC as well. Consequently, we propose leveraging APC to improve our classification tasks on multi-variate time-series data, in the face of both class imbalance and missing values.

Given a length N sequence $\{\mathbf{x}_t\}_{t=1}^{N}$ of observation vectors $\mathbf{x}_t \in \mathbb{R}^m$, the APC framework trains an autoregressive encoder Enc to sequentially reconstruct the observation at $n \geq 1$ time steps ahead of the current step, for $t = 0$ to $N - n$ (see Figure 1). Denoting $\mathbf{h}_t$ the hidden state of the encoder at time t, with $\mathbf{h}_0 = \mathbf{0}$, the output of the network $\mathbf{y}_t \in \mathbb{R}^m$ is given as:

$$\begin{aligned} \mathbf{h}_t &= \text{Enc}(\mathbf{h}_{t-1}, \mathbf{x}_t) \\ \mathbf{y}_t &= \mathbf{W}\mathbf{h}_t \end{aligned} \tag{1}$$

In prior work (Chung et al., 2019), the model was then trained to minimize the L1 distance between all predictions and ground truth (i.e $\mathcal{L} = \sum_{t=1}^{N-n} ||\mathbf{x}_{t+n} - \mathbf{y}_t||_1$). However, if applied directly in the highly-missing data setting, we

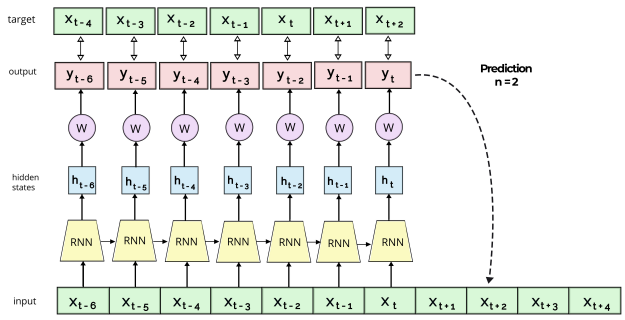

Figure 1. **Autoregressive Predictive Coding (APC)**: input sequence $\{\mathbf{x}_t\}_{t=1}^N$ is encoded (e.g. with an RNN) at each time step to a hidden state $\mathbf{h}_t$. A matrix $\mathbf{W}$ then transforms the hidden states to an output sequence $\{\mathbf{y}\}_{t=1}^N$ of the same dimension as $\mathbf{x}_t$.

observe this loss encourages over-emphasis of missing predictions (analogous to the challenges encountered with class imbalance). In the following section we provide an extension to the APC framework which overcomes this challenge through the use of a masked MSE loss.

4. Method

In the following experiments involving APC, each encoder is optimized to minimize the reconstruction loss *of observed time steps*. Specifically we denote this the masked mean squared error (MSE):

$$\textbf{MaskedMSE}(\{\mathbf{x_t}, \mathbf{y}_t\}_{t=1}^N) = \frac{\sum_{t=1}^{\text{N-n}} (\mathbf{x}_{t+n} - \mathbf{y}_t)^2 \cdot \mathbf{m}_{t+n}}{\sum_{t=1}^{\text{N}-n} \mathbf{m}_{t+n}} \tag{2}$$

where the masking vector $\mathbf{m}_{t+n} \in \{0, 1\}^m$ indicates which variables are missing at time step $t + n$, and ensures that the encoder does not become biased towards missing values.

4.1. Encoder models

To evaluate the impact of specific RNN encoder architectures within APC and how they handle missing values, we compare two different encoders: 1) a baseline GRU (Cho et al., 2014) encoder, and 2) a more complex, state-of-the-art GRU-D (Che et al., 2018) encoder that uses trainable decays to take advantage of the missingness patterns from the data.

GRU (Cho et al., 2014) is a type of RNN that uses two gates (update and reset gate) to learn dependencies from the data over time. Since the GRU model architecture does not inherently handle missing values, we add a binary "missingness" flag next to each measurement x_t^d, containing 1 if the value is missing at that time-step and 0 otherwise. GRU-D builds on the **GRU**, but with trainable **D**ecays and takes advantage of two representations of informative missingness patterns, *masking* and *time interval*, which replace the "missingness" flags of the GRU. The *time interval* $\delta_t^d \in \mathbb{R}$ indicates how long it has been for each variable d since its last observation. GRU-D uses a *decay* mechanism for the input variables and the hidden states, which exponentially decays the missing variables over time from their last value toward its empirical mean.

Representations that were learned in the self-supervised setting with APC are leveraged by using $\mathbf{h}_t$, the output of the last layer of the APC encoder (GRU/GRU-D), as input for the classifier. The classifier is trained by minimizing the Cross-Entropy Loss, which could be binary or categorical.

5. Experiments

5.1. Datasets and performance evaluation

We evaluate our methods on two real-world datasets: a benchmark dataset called Physionet Challenge 2012 (Goldberger et al., 2000), and time series data from the menstrual cycle tracking app Clue (Clu). For Physionet, the classification task is to predict whether an ICU patient will survive or die during their stay at the hospital. For Clue, we seek to predict the discontinuation of birth control methods over time. Both datasets contain severe class imbalances as well as varying levels of missing values. Clue samples contain 70.29 % missing values on average [range : 6.06, 81.45], while this is 79.75 % for Physionet [range : 62.61, 99.94]. Clue also contains a more severe class imbalance compared to Physionet (92% majority class vs 86%) (appendix Tables 3 & 4) . While Physionet is a *binary* classification benchmark, Clue requires *multi class* classification.

For the binary classification task, we chose the AUPRC evaluation metric as it shows how well the model is handling the minority class. For the multi-class classification in the class imbalance setting, the weighted F1 score (weighted harmonic mean of the precision and recall) is the most common metric (Sokolova & Lapalme, 2009). We also introduce the weighted F1 minority score (defined in appendix) to highlight the performances on the minority classes (Clue).

Similar to prior work on Physionet (Horn et al., 2019), we split the datasets into three subsets: 60 % of the dataset is used as our training data, 20 % as validation and 20 % as the test set. Class distribution is preserved in each set. Evaluation on the test set was performed after the model was restored to the state with the best performance on the validation set. Results are reported as the mean ± std. of the performances on the test set over 3 *independent* runs.

Since Physionet has been used as a benchmark dataset, we compare our results to state-of-the art results obtained on this dataset, achieved with GRU-D (Horn et al., 2019). For full disclosure, we address the main differences between our implementations in the appendix. For Clue, since this classification task has never been implemented before, we

compare our results to those obtained with a naive classifier, always predicting the majority class. In order to compare the APC performances to those of an autoencoder, we implement our approach setting the time shift factor $n = 0$. We then experiment with different values for $n \geq 1$, and compare the effects of using either a GRU or GRU-D encoder. Finally, we compare our APC implementations to several baselines that use different types of imputation methods (GRU-Mean, GRU-Forward, GRU-Simple) (Horn et al., 2019) in combination with class imbalance methods.

5.2. Results

We find that APC with $n = 1$ outperforms the autoencoder on Physionet (Figure 2) and Clue (appendix Tables 7 & 8), indicating that the features learned through predicting future time steps are more relevant compared to those learned by encoding the input directly. Comparing different values of the time shift factor, we see that APC with $n = 1$ performs best in all settings on both datasets. Table 1 shows that for Physionet the GRU-APC, which tackles both class imbalance and missing data simultaneously, outperforms baselines that use a combination of methods that tackle each of these separately. Table 2 shows that GRU-APC also improves the results on Clue compared to these baselines, although in this case oversampling + class weights are also needed, given the more severe imbalance. Further, we see a different trend in performance of the imputation methods on both datasets, indicating that these methods impose assumptions that are very dataset-specific. One major advantage of APC is that it does not impose such strong assumptions on what the missing values should be. When comparing the APC encoder models, we find that GRU-D outperforms GRU on both datasets, indicating that the GRU-D is taking advantage of some potentially *informative missingness*. Table 2 shows that our Clue classification results are considerably better than random, and that APC improves these results, better identifying the minority classes. For Physionet, we include four literature baselines to showcase the variability in existing results (Table 1). We see that applying APC allows us to surpass state-of-the-art AUPRC scores.

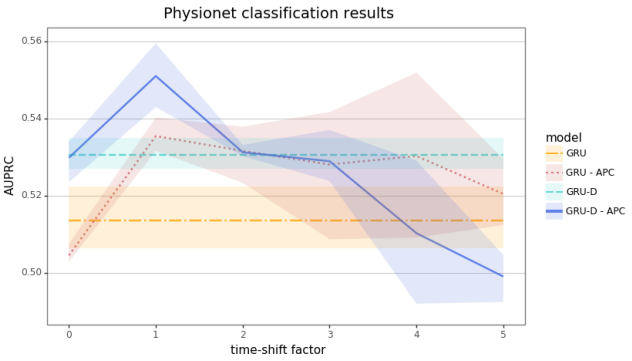

Figure 2. Physionet classification results for different values of the time shift factor n. All models are implemented with class weights.

Table 1. Comparisons on Physionet 2012 mortality task. OS = oversampling, CW = class weights.

MODEL	CLASS IMBALANCE METHOD	AUROC	AUPRC
GRU-D (CHE ET AL., 2018)	OS	84.24 ± 1.2	-
GRU-D (HORN ET AL., 2019)	OS	**86.3 ± 0.3**	53.7 ± 0.9
GRU-D (CAO ET AL., 2018)	NONE	83.4 ± 0.2	-
BRITS (CAO ET AL., 2018)	NONE	85.0 ± 0.2	-
GRU	CW	85.4 ± 0.4	51.4 ± 0.9
GRU-MEAN	CW	84.6 ± 0.2	50.3 ± 0.7
GRU-FORWARD	CW	84.3 ± 1.0	52.0 ± 1.4
GRU-SIMPLE	CW	85.5 ± 0.1	53.8 ± 0.2
GRU-D	CW	85.5 ± 0.3	53.1 ± 0.4
GRU-APC ($n = 1$)	NONE	**86.0 ± 0.5**	**54.1 ± 1.0**
GRU-D - APC ($n = 1$)	NONE	85.2 ± 0.9	54.1 ± 2.3
GRU-APC ($n = 1$)	CW	85.9 ± 0.3	53.5 ± 0.5
GRU-D - APC ($n = 1$)	CW	85.3 ± 0.1	**55.1 ± 0.9**

Table 2. Comparisons on Clue classification task. OS = oversampling, CW = class weights.

MODEL	CLASS IMBALANCE METHOD	WEIGHTED F1	WEIGHTED F1 MINORITY
NAIVE CLASSIFIER		87.47	-
GRU	OS + CW	90.3 ± 0.7	22.3 ± 7.2
GRU-MEAN	OS + CW	88.4 ± 0.5	22.1 ± 1.8
GRU-FORWARD	OS + CW	87.5 ± 0.2	22.1 ± 0.5
GRU-SIMPLE	OS + CW	87.8 ± 0.3	22.2 ± 0.4
GRU-D	OS + CW	87.5 ± 0.3	22.5 ± 0.7
GRU-APC ($n = 1$)	NONE	90.0 ± 0.1	21.2 ± 0.7
GRU-APC ($n = 1$)	OS + CW	**90.7 ± 0.1**	**25.7 ± 1.1**
GRU-D - APC ($n = 1$)	OS + CW	90.3 ± 0.0	**27.3 ± 0.5**

6. Discussion

We demonstrate that APC shows promising results beyond NLP, since it learns improved features from complex multivariate time series data with severe class imbalance and many missing values. APC allows the classifier to better identify the minority class(es), even in the face of sparsity, while making fewer assumptions about the data compared to data imputation methods. Clue is a more limited and less controlled real-world dataset than Physionet, since it relies on self-reported features by app users which could contain many artifacts. This could account for the more pronounced improvements on Physionet compared to Clue. In future work we intend to focus on model interpretability, either by analyzing the internal representations obtained from APC, performing sensitivity analysis, or applying an attention layer to the model. We would also like to train APC on a larger quantity of unlabeled data, explore different encoder models (e.g. Transformers), and add more encoder layers, since these methods have previously shown to improve performance. Finally, the exploration of APC's generative ability for multi-variate time-series presents an interesting future area of work.

References

Barandela, R., Sánchez, J. S., Garca, V., and Rangel, E. Strategies for learning in class imbalance problems. *Pattern Recognition*, 36(3):849–851, 2003.

Branco, P., Torgo, L., and Ribeiro, R. P. A survey of predictive modelling under imbalanced distributions. *CoRR*, abs/1505.01658, 2015. URL http://arxiv.org/abs/1505.01658.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners, 2020.

Cao, W., Wang, D., Li, J., Zhou, H., Li, L., and Li, Y. Brits: Bidirectional recurrent imputation for time series. In *Advances in Neural Information Processing Systems*, pp. 6775–6785, 2018.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. Smote: Synthetic minority over-sampling technique. *J. Artif. Int. Res.*, 16(1):321–357, June 2002a. ISSN 1076-9757.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002b.

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

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. URL https://www.aclweb.org/anthology/D14-1179.

Chung, Y.-A. and Glass, J. Generative pre-training for speech with autoregressive predictive coding. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3497–3501. IEEE, 2020.

Chung, Y.-A., Hsu, W.-N., Tang, H., and Glass, J. An unsupervised autoregressive model for speech representation learning. *arXiv preprint arXiv:1904.03240*, 2019.

Clue by BioWink GmbH, Adalbertstraße 7-8, 10999 Berlin, Germany. https://helloclue.com/ (2020).

Goldberger, A. L., Amaral, L. A. N., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C.-K., and Stanley, H. E. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000. Circulation Electronic Pages: http://circ.ahajournals.org/content/101/23/e215.full PMID:1085218; doi: 10.1161/01.CIR.101.23.e215.

Horn, M., Moor, M., Bock, C., Rieck, B., and Borgwardt, K. Set functions for time series. *arXiv preprint arXiv:1909.12064*, 2019.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Mikalsen, K. Ø., Soguero-Ruíz, C., Bianchi, F. M., Revhaug, A., and Jenssen, R. Time series cluster kernels to exploit informative missingness and incomplete label information. *CoRR*, abs/1907.05251, 2019. URL http://arxiv.org/abs/1907.05251.

Ng, W. W., Zeng, G., Zhang, J., Yeung, D. S., and Pedrycz, W. Dual autoencoders features for imbalance classification problem. *Pattern Recognition*, 60:875–889, 2016.

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. Improving language understanding by generative pre-training, 2018.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019.

Rubanova, Y., Chen, R. T., and Duvenaud, D. Latent odes for irregularly-sampled time series. *arXiv preprint arXiv:1907.03907*, 2019.

Sagheer, A. and Kotb, M. M. W. Unsupervised pre-training of a deep lstm-based stacked autoencoder for multivariate time series forecasting problems. *Scientific Reports*, 9, 2019.

Sokolova, M. and Lapalme, G. A systematic analysis of performance measures for classification tasks. *Information processing & management*, 45(4):427–437, 2009.

Yen, S.-J. and Lee, Y.-S. Cluster-based under-sampling approaches for imbalanced data distributions. *Expert Systems with Applications*, 36(3):5718–5727, 2009.

Zhu, M., Xia, J., Jin, X., Yan, M., Cai, G., Yan, J., and Ning, G. Class weights random forest algorithm for processing class imbalanced medical data. *IEEE Access*, 6:4641–4652, 2018.

Appendix

A. Class imbalance

Table 3. Physionet: Class distribution.

OUTPUT LABEL	PERCENTAGE % OF TOTAL
SURVIVOR	85.76
DIED IN-HOSPITAL	14.24

Table 4. Clue: Class distribution

OUTPUT LABEL	PERCENTAGE % OF TOTAL
ON	91.52
OFF	4.88
OTHER-HORMONAL	2.22
OTHER-NON-HORMONAL	1.38

B. Missing values

Clue contains 70.29 % missing values on average ranging from 6.06 % to 81.45 %, while Physionet contains 79.75 % missing values on average ranging from 62.61 % to 99.94 %.

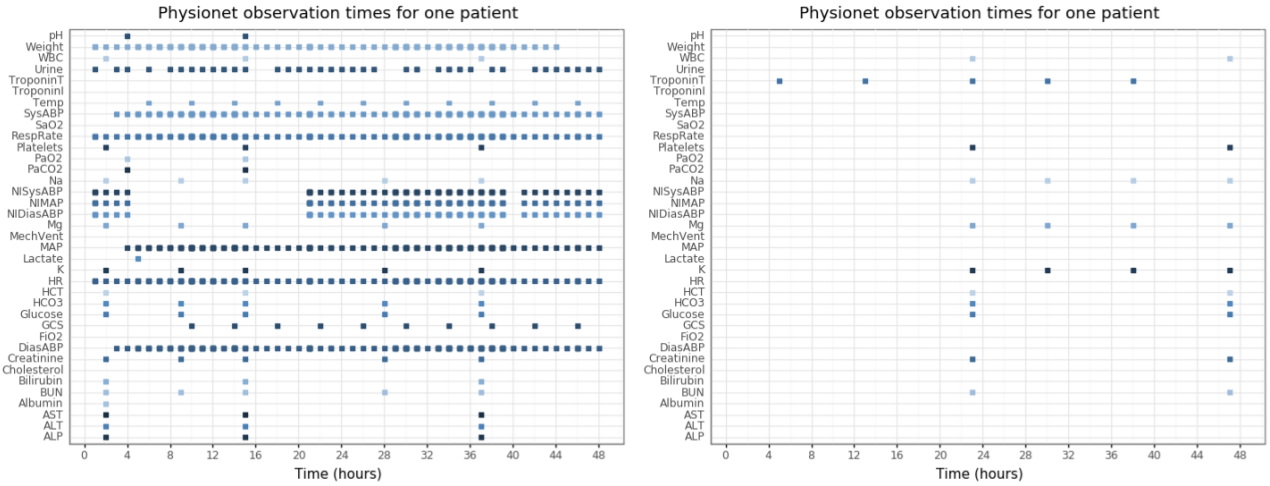

Figure 3. Physionet: Examples of time series for two different ICU patients with different frequencies of observations. On the left, we have a time series with a high frequency of observations. On the right is an example of a patient who is barely observed.

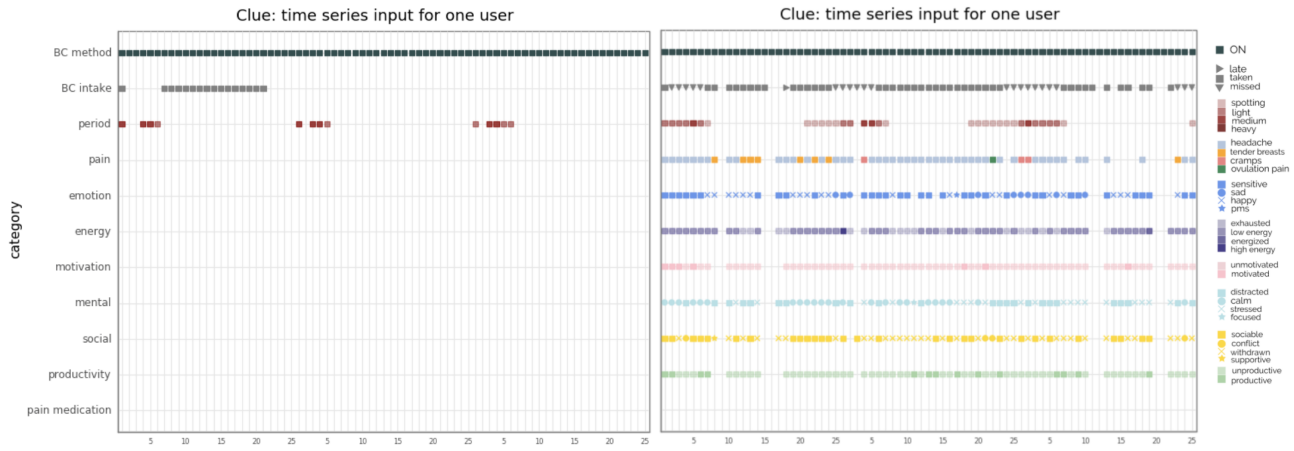

Figure 4. Clue: Example of the time series of the 3 input cycles for a user that tracks with low frequency (left), and a user that tracks with high frequency (right).

C. Clue's Empirical Mean

While Physionet has one mean per variable, we implement a multi-dimensional mean for the Clue dataset, with each variable having a different mean for each day of the cycle.

D. Time scales

In the Physionet dataset, the measurements are observed at irregular time steps at a minute-by-minute resolution. For the GRU model, we aggregate the measurements to a 1 hour resolution, similar to what had been done in previous work on this dataset (Che et al., 2018). However, for the GRU-D model we aggregate the measurements to a 1-decimal time resolution (e.g. 2.6 hours) as in the original GRU-D paper (Che et al., 2018). This allows the model to learn with more detail the time interval since a variable was last observed.

E. Frozen vs fine-tuned APC weights

To leverage the representations that were learned in the self-supervised setting with APC, we take the output of the last layer of the GRU/GRU-D in the APC as the extracted representations, i.e. $\text{Enc}(\mathbf{x}) = \mathbf{h}$, then implement the classifier on top of them. In the semi-supervised setting, we used two scenarios, consecutively:

1. **frozen**: the weights of the pre-trained encoder are kept frozen and only the classification model is optimized.
2. **fine-tuned**: APC's encoder and the classifier are trained end-to-end so that the learned representations are better suited to the final classification task.

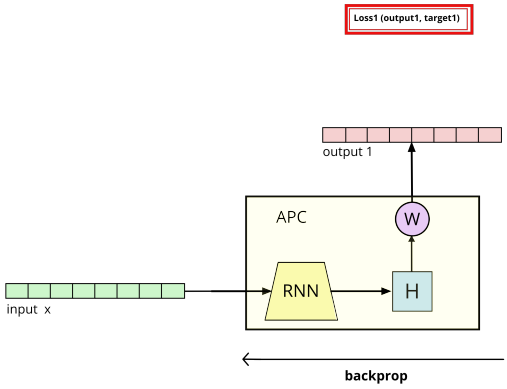

Figure 5. **Step 1**: The APC is pre-trained by optimizing the autoregressive loss

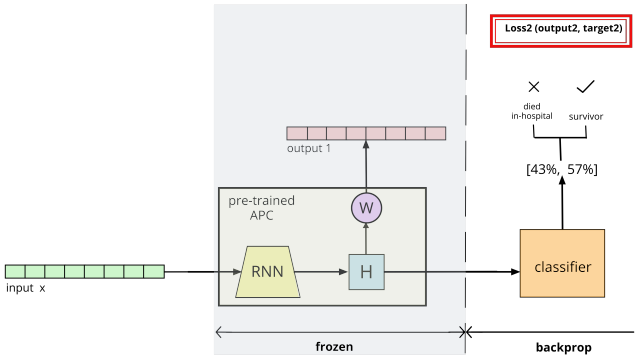

Figure 6. **Step 2**: The APC encoder weights are frozen & the classifier is trained by optimizing the cross-entropy loss.

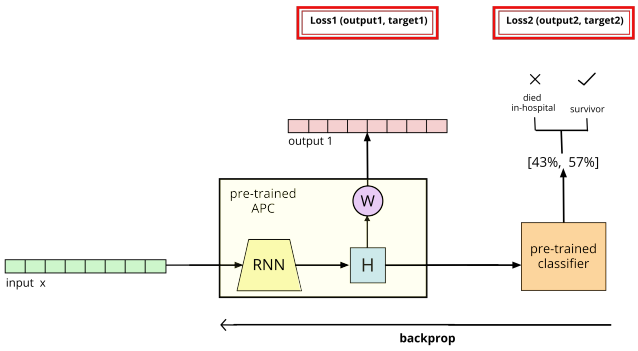

Figure 7. **Step 3**: Both the encoder & the classifier are trained end-to-end, using the pre-trained encoder & pre-trained classifier weights.

F. Comparison to Referenced Publications

To our knowledge, the most competitive and state-of-the-art results that have been achieved to date on the Physionet in-hospital mortality task are by Horn et al. (2019), with their GRU-D and Transformer implementations attaining AUPRC scores of 53.7 ± 0.9, and 52.8 ± 2.2, respectively (Horn et al., 2019).

Consequently, we set out to implement the GRU-D similar to their implementation, so that our methods are directly comparable. However, we noticed that some architectural decisions taken in (Horn et al., 2019) were leading to lower performance scores overall compared to our initial implementation. Therefore, we decided to keep to our initial implementation. For full disclosure, the main differences between the two implementations are:

- When processing the static variables, (Horn et al., 2019) compute the initial state of the RNN based on these static variables. Contrary to their implementation, our RNN does not handle the static variables. Instead, the static variables are concatenated with the final hidden states learned by the RNN which are then input directly into the classifier.
- To handle the class imbalance, they oversample the minority class, while we implement the GRU-D using class weights.
- Our dataset partition (train, validation, and test splits) might be different from theirs, which could be another source of potential variations in results.

G. Evaluation Metrics

Binary Classification

The AUROC score is given by the following formula:

$$\text{AUROC} = \frac{1 + \text{TP}_{\text{rate}} - \text{FP}_{\text{rate}}}{2} = \frac{\text{TP}_{\text{rate}} + \text{TN}_{\text{rate}}}{2} \tag{3}$$

where the TP_{rate}, FP_{rate}, and the TN_{rate} are defined as:

$$\text{true positive rate (recall)} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}} \tag{4}$$

$$\text{false positive rate} = \frac{\text{false positive}}{\text{false positive} + \text{true negative}} \tag{5}$$

$$\text{true negative rate} = \frac{\text{true negative}}{\text{true negative} + \text{false positive}} \tag{6}$$

However, when dealing with imbalanced classification, ROC curves may give an extremely optimistic view of the performance (Branco et al., 2015), and in these cases the precision-recall curves (PR curves) are recommended as the more appropriate metric. The PR curve shows a plot of the recall vs. precision for different probability tresholds. The recall, also known as the true positive rate or sensitivity, is displayed in Equation 4, and the precision is defined as:

$$\text{precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}} \tag{7}$$

Both the precision and recall are mostly focused on the positive class (minority class) and are not concerned about the true negatives (majority class). The Area Under this Precision-Cecall Curve score (AUPRC) is therefore mostly a representation of how well the model is handling the minority class. Since the AUPRC is arguably the most appropriate metric in the case of imbalanced classification, we focus on this performance metric for the binary classification task.

MULTICLASS CLASSIFICATION

We evaluate the performance of the multi-class classifier with the **F1-score** (Sokolova & Lapalme, 2009), which is the harmonic mean of the precision and recall:

$$\mathbf{F}_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{8}$$

Since our test set is class imbalanced, we do not want to give equal weights to each class, but instead we want to calculate a **weighted average F1-score**:

$$\frac{\#\text{Class 1}}{\#\text{total}} \textbf{ x F1}_{\text{Class 1}} + \frac{\#\text{Class 2}}{\#\text{total}} \textbf{ x F1}_{\text{Class 2}} + \frac{\#\text{Class 3}}{\#\text{total}} \textbf{ x F1}_{\text{Class 3}} + \frac{\#\text{Class 4}}{\#\text{total}} \textbf{ x F1}_{\text{Class 4}} \tag{9}$$

Furthermore, to highlight the performances on the minority classes, we define the **weighted F1 minority**:

$$\frac{\#\text{Class 2}}{\#\text{total minority}} \textbf{ x F1}_{\text{Class 2}} + \frac{\#\text{Class 3}}{\#\text{total minority}} \textbf{ x F1}_{\text{Class 3}} + \frac{\#\text{Class 4}}{\#\text{total minority}} \textbf{ x F1}_{\text{Class 4}}$$

H. Data imputation baselines

GRU-Mean:

$$x_t^d \leftarrow m_t^d x_t^d + (1 - m_t^d)\tilde{x}^d \tag{10}$$

GRU-Forward:

$$x_t^d \leftarrow m_t^d x_t^d + (1 - m_t^d)x_{t'}^d \tag{11}$$

GRU-Simple:

$$x_t^{(n)} \leftarrow [x_t^{(n)}; m_t^{(n)}; \delta_t^{(n)}] \tag{12}$$

I. Oversampling & class weights

Because the Clue dataset contains a more severe class imbalance compared to Physionet, we propose another method specifically for Clue, which combines both oversampling (Chawla et al., 2002b) and using class weights. In this case, we oversample the minority class to 12% and then use class weights on the oversampled class frequency.

J. Hyperparameters

Table 5. Final hyperparameters used for each model on each dataset, after performing hyperparameter optimization.

Model	Physionet	Clue
GRU	learning rate = 1e-03, batch size = 32, hidden units = 64, dropout = 0.0, recurrent dropout = 0.0, epochs = 50	learning rate = 1e-04, batch size = 100, hidden units = 200, dropout = 0.4, recurrent dropout = 0.1, epochs = 200
GRU-D	learning rate = 1e-03, batch size = 32, hidden units = 32, dropout = 0.1, recurrent dropout = 0.0, epochs = 50	learning rate = 1e-03, batch size = 100, hidden units = 200, dropout = 0.1, recurrent dropout = 0.0, epochs = 200
GRU - APC	learning rate - step 1 = 1e-03, learning rate - step 2 & 3 = 1e-04, batch size = 32, hidden units = 64, dropout = 0.0, recurrent dropout = 0.0, epochs = 100	learning rate = 1e-04, batch size = 100, hidden units = 200, dropout = 0.4, recurrent dropout = 0.1, epochs = 50
GRU-D - APC	learning rate - step 1 = 1e-03, learning rate - step 2 & 3 = 1e-04, batch size = 100, hidden units = 256, dropout = 0.1, recurrent dropout = 0.0 epochs - step 1 = 100 epochs - step 2 & 3 = 50	learning rate step 1 = 1e-03, learning rate step 2 & 3 = 1e-04, batch size = 100, hidden units = 250, dropout = 0.1, recurrent dropout = 0.0 epochs = 50

K. Results

Table 6. Comparison of using frozen versus fine-tuned APC weights on the Physionet classification results. These results also show the impact of using class weights as a class imbalance method. All APC methods are implemented with time shift factor n = 1.

MODEL	APC WEIGHTS METHOD	CLASS IMBALANCE METHOD	AUROC	AUPRC
GRU	-	UNDERSAMPLING	84.2 ± 0.3	48.4 ± 1.3
GRU	-	OVERSAMPLING	84.6 ± 0.5	51.4 ± 1.7
GRU	-	CLASS WEIGHTS	85.4 ± 0.4	51.4 ± 0.9
GRU - APC	FROZEN	NONE	85.1 ± 0.2	51.8 ± 0.2
GRU - APC	FROZEN	CLASS WEIGHTS	85.0 ± 0.2	51.3 ± 0.1
GRU - APC	FINE-TUNED	NONE	$\mathbf{86.0 \pm 0.5}$	$\mathbf{54.1 \pm 1.0}$
GRU - APC	FINE-TUNED	CLASS WEIGHTS	85.9 ± 0.3	53.5 ± 0.5
GRU-D - APC	FROZEN	NONE	85.1 ± 0.9	54.0 ± 2.3
GRU-D - APC	FROZEN	CLASS WEIGHTS	85.3 ± 0.1	$\mathbf{55.1 \pm 0.9}$
GRU-D - APC	FINE-TUNED	NONE	85.2 ± 0.9	54.1 ± 2.3
GRU-D - APC	FINE-TUNED	CLASS WEIGHTS	85.3 ± 0.2	55.0 ± 0.9

Table 7. Classification results on the Clue dataset using **GRU** - **APC** for different values of the time shift factor.

APC WEIGHTS METHOD	TIME SHIFT	F1 ON	F1 OFF	F1 OTHER H	F1 OTHER NH	WEIGHTED F1	WEIGHTED F1 MINORITY
FROZEN	0	96.7 ± 0.0	19.7 ± 4.6	28.2 ± 1.7	17.1 ± 1.4	90.4 ± 0.2	21.5 ± 2.1
FINE-TUNED	0	96.7 ± 0.1	22.0 ± 12.3	29.8 ± 2.4	14.6 ± 0.4	90.4 ± 0.7	22.8 ± 6.6
FROZEN	1	96.7 ± 0.0	24.8 ± 6.4	24.1 ± 3.2	16.6 ± 2.9	$\mathbf{90.5 \pm 0.3}$	$\mathbf{23.2 \pm 3.0}$
FINE-TUNED	1	96.7 ± 0.0	28.8 ± 2.2	25.7 ± 2.4	14.8 ± 2.5	$\mathbf{90.7 \pm 0.1}$	$\mathbf{25.7 \pm 1.1}$
FROZEN	2	96.7 ± 0.1	27.3 ± 4.0	22.9 ± 3.3	16.2 ± 3.5	90.6 ± 0.3	24.3 ± 2.5
FINE-TUNED	2	96.6 ± 0.3	23.3 ± 14.0	24.0 ± 2.4	14.6 ± 1.5	90.3 ± 0.9	22.1 ± 7.3
FROZEN	5	96.5 ± 0.2	24.7 ± 5.9	17.4 ± 9.7	17.4 ± 2.2	90.2 ± 0.5	21.6 ± 4.5
FINE-TUNED	5	96.0 ± 0.7	16.6 ± 8.9	14.8 ± 13.0	15.0 ± 0.6	89.2 ± 1.3	15.9 ± 8.2

Table 8. Classification results on the Clue dataset using **GRU-D** - **APC** for different values of the time shift factor.

APC WEIGHTS METHOD	TIME SHIFT	F1 ON	F1 OFF	F1 OTHER H	F1 OTHER NH	WEIGHTED F1	WEIGHTED F1 MINORITY
FROZEN	0	95.6 ± 0.6	28.1 ± 2.3	20.1 ± 2.6	11.6 ± 2.4	89.5 ± 0.7	23.3 ± 2.4
FINE-TUNED	0	96.2 ± 0.1	32.4 ± 0.8	23.1 ± 2.1	14.4 ± 2.6	90.3 ± 0.1	27.0 ± 0.5
FROZEN	1	96.2 ± 0.0	32.0 ± 0.0	25.4 ± 0.2	13.9 ± 2.8	$\mathbf{90.3 \pm 0.0}$	$\mathbf{27.3 \pm 0.5}$
FINE-TUNED	1	96.0 ± 0.5	30.3 ± 0.2	21.7 ± 4.0	13.7 ± 2.2	$\mathbf{90.0 \pm 0.5}$	$\mathbf{25.3 \pm 0.8}$
FROZEN	2	96.1 ± 0.1	31.4 ± 0.1	24.8 ± 0.5	10.7 ± 0.8	90.2 ± 0.0	26.3 ± 0.2
FINE-TUNED	2	95.7 ± 0.2	31.2 ± 0.6	19.2 ± 1.0	11.4 ± 2.6	89.7 ± 0.3	24.9 ± 1.1
FROZEN	5	95.7 ± 0.4	31.6 ± 1.2	18.6 ± 4.8	11.2 ± 3.7	89.7 ± 0.5	24.8 ± 1.1
FINE-TUNED	5	95.8 ± 0.1	32.6 ± 0.2	17.0 ± 1.0	10.8 ± 1.5	89.8 ± 0.2	25.0 ± 0.6