
As easy as APC: overcoming missing data and class imbalance in time series with self-supervised learning

Fiorella Wever

University of Amsterdam, Netherlands
Clue by BioWink GmbH, Germany
fiorella.wever@gmail.com

T. Anderson Keller

University of Amsterdam, Netherlands
t.anderson.keller@gmail.com

Laura Symul

Department of Statistics,
Stanford University, California
lsymul@stanford.edu

Victor Garcia

University of Amsterdam, Netherlands
v.garciasatorras@uva.nl

Abstract

High levels of missing data and strong class imbalance are ubiquitous challenges that are often presented simultaneously in real-world time series data. Existing methods approach these problems separately, frequently making significant assumptions about the underlying data generation process in order to lessen the impact of missing information. In this work, we instead demonstrate how a general self-supervised training method, namely Autoregressive Predictive Coding (APC), can be leveraged to overcome *both* missing data and class imbalance simultaneously without strong assumptions. Specifically, on a synthetic dataset, we show that standard baselines are substantially improved upon through the use of APC, yielding the greatest gains in the combined setting of high missingness and severe class imbalance. We further apply APC on two real-world medical time-series datasets, and show that APC improves the classification performance in all settings, ultimately achieving 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 [2]. The difficulty often arises from the inherent class imbalance associated with rarity, encouraging models to classify most examples into the majority class, frequently resulting in poor performance for the minority class of interest [2]. Compounding this challenge is the ubiquity of missing data in real-world time series, further reducing the available information for classification on imbalanced datasets. Prior work has demonstrated techniques for overcoming these obstacles, however most methods tackle each challenge separately; either dealing with properly learning in the context of missingness [4, 6, 18] or improving identification of the minority class(es) [1, 5, 21, 22].

Recently, self-supervised pre-training has gained popularity, especially within the Natural Language Processing (NLP) community, showing significant performance improvements on downstream supervised learning tasks [3, 16, 17]. In this work, we propose leveraging a self-supervised autoregressive pre-training framework, namely Autoregressive Predictive Coding (APC) [9], to overcome both the challenges of class imbalance and missing data simultaneously in the context of time-series classification. Specifically, we suggest that pre-training recurrent neural network feature extractors to perform multiple-step-ahead forward prediction encourages the learning of discriminative representations which are independent of class labels while simultaneously taking advantage of potentially informative missingness patterns – all without making strong assumptions about the data.

We validate our approach on a synthetic dataset generated with varying levels of missingness and class imbalance, and find that APC substantially improves performance, especially in the severe class imbalance and high sparsity setting. Further, we apply APC on two real-world datasets, Physionet and Clue, both containing missing values and class imbalance, and show that it improves the classification results in both cases, ultimately achieving state-of-the-art AUPRC scores on the Physionet benchmark.

2 Related work

Class Imbalance Some of the most common methods for handling class imbalance consist of data re-sampling, such as under-sampling the majority class [1, 21] and over-sampling the minority class [5]. However, under-sampling can result in losing a significant amount of data, while over-sampling methods, such as SMOTE [5], artificially increase the data set. Another widely adopted method is class weights [22], 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. Most related to this work are methods that approach class imbalance through unsupervised learning. One such method is the dual autoencoders features (DAF) [14], which uses two stacked autoencoders to learn different sets of features of the data in an imbalanced setting. It is shown to outperform traditional data re-sampling methods, and suggests that leveraging unsupervised learning to obtain a set of relevant features could be a promising approach to deal with class imbalance. Our approach differs from those involving autoencoders in that a time shifting factor $n \geq 1$ encourages the model to learn forward predictive features, and thus more global structure of the data, rather than only local structure [9].

Missing Data Traditional methods for handling missing data often involve first filling in the missing values, and then applying predictive models on the imputed data [6]. 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 that prevents the prediction model from properly exploring the missingness patterns [6]. As shown in [6], such *informative missingness* may actually encode useful information about the target labels. 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) [4], which simultaneously imputes the missing values and performs classification/regression within a joint neural graph. Another similar method is the GRU with trainable Decays (GRU-D) [6]. Both these methods take advantage of two representations of informative missingness: *masking* and *time interval* [6]. Recently, two models based on ordinary differential equations, the ODE-RNN and the Latent-ODE, have also shown promising results on irregularly-sampled data [18]. However, the computational complexity of these models is high, which [12] concluded led to not finding the optimal hyperparameters.

Self-Supervised Pre-Training A special case of semi-supervised learning, called self-supervised pre-training, aims to learn a good initialization point for the supervised setting instead of changing the supervised learning objective [16]. Most models used for this purpose are based on autoencoders [19], while some of the most recent and promising methods are based on the idea of predictive coding, such as Contrastive Predictive Coding (CPC) [15] and Autoregressive Predictive Coding (APC) [9]. Similar to CPC, APC learns from sequential data by trying to predict $n \geq 1$ steps ahead of the current step [9]. However, while CPC learns from forward prediction in a contrastive way, APC does so in an autoregressive manner, and has shown to significantly outperform CPC on speech recognition and phonetic classification tasks [8, 9]. Although APC pre-training has been previously proposed to improve supervised tasks, as far as we are concerned this is the first exploration of its application to handle both missing values and class imbalance.

3 Methods

APC with MaskedMSE 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 4). 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:

$$\mathbf{h}_t = \text{Enc}(\mathbf{h}_{t-1}, \mathbf{x}_t) \quad \mathbf{y}_t = \mathbf{W}\mathbf{h}_t \quad (1)$$

In prior work [9], 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 observe this loss encourages over-emphasis of missing predictions (analogous to the challenges encountered with class imbalance). We thus propose a MaskedMSE loss to minimize reconstruction error over *observed time steps* only. Formally:

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

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.

Encoder Models & Missingness Handling 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 [7], and 2) the state-of-the-art GRU-D [6] built to handle missingness. Since the GRU architecture does not inherently handle missing values, we add a binary “missingness” flag next to each measurement x_t^d , showing 1 if the value is missing at that time-step and 0 otherwise. GRU-D builds on the GRU, but with trainable Decays 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, and is used to exponentially *decay* missing input variables and hidden states from their last value toward their empirical mean. Representations 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, trained with a cross-entropy loss.

4 Experiments

Datasets We first evaluate our method on a synthetic dataset for which we control the level of missing values, as well as the level of class imbalance. This provides insights on the potential gains in performance provided by APC in handling class imbalance and data missingness both independently and in combination. The synthetic dataset consists of time-series of two variables, one continuous (noisy sine curve) and one binary (random Bernoulli). The output is a non-linear function of the sine curve frequency and of the Bernoulli variable probability (See Appendix for details). We also experiment on two real-world datasets: a benchmark dataset from the Physionet Challenge 2012 [11], and time series data from the menstrual cycle tracking app Clue [10]. Clue samples contain 70.29% missing values on average with 92% majority class, while Physionet contains 79.75% missing with 86% majority class. Our code is available at <https://github.com/anonSSLneurips/APC>.

Performance Evaluation For binary classification tasks (such as our synthetic dataset and Physionet), we choose to report AUPRC score as it more accurately shows how well the model is handling the minority class. For multi-class classification in the class imbalance setting (as with Clue), the weighted F1 score is the most common metric [20]. We also introduce the weighted F1 minority score (defined in appendix) to highlight the performances on the minority classes. Similar to prior work on Physionet [12], we use 60 % of the dataset as our training data, 20 % as validation and 20 % as the test set, preserving the class distribution in each set. Evaluation on the test set was performed using the model with the best performance on the validation set. Results are reported as the mean \pm std. of the performances on the test set over 3 *independent* runs.

Baselines For Physionet, we compare our results to the state-of-the art results for this benchmark dataset, achieved with GRU-D [12]. We note our reimplementation does not reach the same performance as the published model, and address the main differences between the implementations in the appendix. Despite these differences, our model still surpasses the best published AUPRC score for Physionet, as can be seen in Table 1. For Clue, given the proprietary nature of the dataset, we compare our results to a naive majority class baseline as well as ablations of our own model, observing APC augmented models again perform best. We additionally compare the performance of APC time shift factor $n \geq 1$ to an autoencoder ($n = 0$), observing optimal performance with $n = 1$, validating the use of APC compared to standard auto-encoding. As additional baselines, we compare our APC implementations to several different types of imputation methods (GRU-Mean, GRU-Forward, GRU-Simple) [12] in combination with class imbalance methods such as class weights.

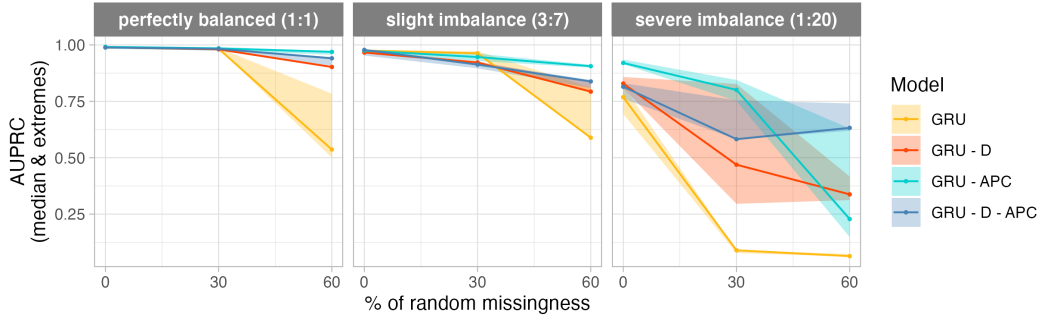


Figure 1: Classification results on synthetic dataset. GRU and GRU-D are implemented using class weights, while the APC implementations do not use class weights. The lines in the figure show the median, and the borders of the shaded region indicate the min and max (extremes) of the results.

Results: Synthetic Data The results on our simulated dataset (Figure 1) show that applying APC improves the classification performance independently in the class imbalance setting and in the sparsity setting, as well as when dealing with both in conjunction. Figure 1 highlights the median and min and max (extremes) of the results. We see that APC improves performance substantially and significantly, especially in the high sparsity and severe class imbalance setting, improving the mean AUPRC score of the baseline GRU from 6.61% to 66.42% (appendix Table 6) when GRU-D and APC are applied in combination. Interestingly, we see that adding APC to a baseline GRU alone appears sufficient to achieve optimal performance, improving over GRU-D-APC in the majority of settings (see Table 6). As shown next, these results are further supported by the performance of APC models on real-world data, ultimately lending strength to the idea that self-supervised learning is a feasible solution to both class imbalance and data missingness *on its own* without the need for data-dependent assumptions.

Results: Real-World Data Applied to real-world data, we find that APC with time shift factor $n = 1$ performs best on both Physionet (Table 1) and Clue (appendix Tables 8 & 9), outperforming standard autoencoding ($n = 0$), indicating that the features learned through forward prediction are more relevant than those learned by reconstructing the input directly. In Table 1, we show that the GRU-APC, which tackles both class imbalance and missing data simultaneously, outperforms baselines that use a combination of independent methods. We include four literature baselines to showcase the variability in existing results, and observe that our models leveraging APC outperform all of them, achieving state-of-the-art AUPRC scores and competitive AUROC scores without needing class weights. Further, in Table 2, we show a continuation of this trend, observing that GRU-APC also improves the results on Clue compared to baselines, although in this case oversampling + class weights are also needed likely due to the more severe imbalance. Interestingly, we see a different trend in performance of the imputation methods on both datasets, indicating that these methods impose assumptions that are dataset-specific. One major advantage of APC is that it does not impose such strong assumptions on what the missing values should be, yet consistently achieves competitive performance.

5 Discussion

In this work we demonstrate self-supervised APC pre-training as a promising method to deal with both severe class imbalance and high levels of missing data in the context of time-series classification. Although APC pre-training is not a drastically novel method, the proposed application to handle missing values and class imbalance has not been significantly explored before, making our contribution a grounded exploration of this idea. We show empirically on synthetic and real world data that representations learned by APC pre-training are superior to standard auto-encoding baselines, and additionally perform on-par or better than existing methods built explicitly to handle the challenges of imbalance and missingness. We note that this work is inherently preliminary and limited in scope, however we believe it provides a grounded foundation for further exploration of the benefits of self-supervised learning in the context of missing data and class imbalance.

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

Model	Class imbalance method	AUROC	AUPRC
GRU-D [6]	OS	84.24 ± 1.2	-
GRU-D [12]	OS	86.3 ± 0.3	53.7 ± 0.9
GRU-D [4]	none	83.4 ± 0.2	-
BRITS [4]	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 ($n = 0$)	CW	84.2 ± 0.3	50.4 ± 0.3
GRU-D - APC ($n = 0$)	CW	85.0 ± 0.3	53.3 ± 0.3
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.

Model	weighted F1	weighted F1 minority
naive classifier	87.47	-
GRU	90.3 ± 0.7	22.3 ± 7.2
GRU-Mean	88.4 ± 0.5	22.1 ± 1.8
GRU-Forward	87.5 ± 0.2	22.1 ± 0.5
GRU-Simple	87.8 ± 0.3	22.2 ± 0.4
GRU-D	87.5 ± 0.3	22.5 ± 0.7
GRU-APC ($n = 1$)	90.7 ± 0.1	25.7 ± 1.1
GRU-D - APC ($n = 1$)	90.3 ± 0.0	27.3 ± 0.5

References

- [1] Barandela, R., Sánchez, J. S., García, V., & Rangel, E. (2003). Strategies for learning in class imbalance problems. *Pattern Recognition*, 36(3), 849-851.
- [2] Branco, P., Torgo, L., & Ribeiro, R. (2015). A survey of predictive modelling under imbalanced distributions. *arXiv preprint arXiv:1505.01658*.
- [3] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.
- [4] Cao, W., Wang, D., Li, J., Zhou, H., Li, L., & Li, Y. (2018). Brits: Bidirectional recurrent imputation for time series. *arXiv preprint arXiv:1805.10572*.
- [5] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16, 321-357.
- [6] 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.
- [7] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.
- [8] Chung, Y. A., & Glass, J. (2020, May). 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.
- [9] Chung, Y. A., Hsu, W. N., Tang, H., & Glass, J. (2019). An unsupervised autoregressive model for speech representation learning. *arXiv preprint arXiv:1904.03240*.
- [10] Clue by BioWink GmbH (2020). URL <https://helloclue.com/>. Adalbertstraße 7-8, 10999 Berlin, Germany.
- [11] Goldberger, A. L., Amaral, L. A., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., ... & Stanley, H. E. (2000). PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. *circulation*, 101(23), e215-e220.
- [12] Horn, M., Moor, M., Bock, C., Rieck, B., & Borgwardt, K. (2020, November). Set functions for time series. In *International Conference on Machine Learning* (pp. 4353-4363). PMLR.
- [13] Mikalsen, K. Ø., Soguero-Ruiz, C., Bianchi, F. M., Revhaug, A., & Jenssen, R. (2021). Time series cluster kernels to exploit informative missingness and incomplete label information. *Pattern Recognition*, 115, 107896.
- [14] Ng, W. W., Zeng, G., Zhang, J., Yeung, D. S., & Pedrycz, W. (2016). Dual autoencoders features for imbalance classification problem. *Pattern Recognition*, 60, 875-889.
- [15] Oord, A. V. D., Li, Y., & Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.
- [16] Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.

- [17] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.
- [18] Rubanova, Y., Chen, R. T., & Duvenaud, D. (2019). Latent odes for irregularly-sampled time series. arXiv preprint arXiv:1907.03907.
- [19] Sagheer, A., & Kotb, M. (2019). Unsupervised pre-training of a deep LSTM-based stacked autoencoder for multivariate time series forecasting problems. Scientific reports, 9(1), 1-16.
- [20] Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. Information processing & management, 45(4), 427-437.
- [21] Yen, S. J., & Lee, Y. S. (2009). Cluster-based under-sampling approaches for imbalanced data distributions. Expert Systems with Applications, 36(3), 5718-5727.
- [22] Zhu, M., Xia, J., Jin, X., Yan, M., Cai, G., Yan, J., & Ning, G. (2018). Class weights random forest algorithm for processing class imbalanced medical data. IEEE Access, 6, 4641-4652.

A Appendix

A.1 Datasets

Synthetic dataset

The synthetic dataset is generated as N two-variable time series of length T . Here $N = 2000$ and $T = 100$. The first variable, x , is a continuous variable simulated as

$$x_n(t) = 1 + o_n + \cos\left(\frac{2\pi t}{P_n}\right) + \epsilon$$

with $\epsilon \sim \mathcal{N}(0, \sigma)$ and where o_n , a random offset, is uniformly sampled from $[0, 1]$ and P_n , the cosine period, is uniformly sampled from $[P_{min}, P_{max}]$. Here, we chose $P_{min} = 5$ and $P_{max} = 20$. The second variable, b , is simulated as a Bernoulli variable with a probability p_n , which is randomly sampled in $[0, 1]$ for each time series n .

The output y_n is built as a combination of x 's period (P_n) and b 's probability (p_n). Specifically, $y_n = (P_n \leq 0.5)(p_n \geq 0.5)$.

Physionet

For the benchmark dataset from the Physionet Challenge 2012 [11], the classification task is to predict whether an ICU patient will survive or die during their stay at the hospital. Physionet samples contain 79.75 % missing values on average [62.61, 99.94]. Physionet is a *binary* classification benchmark, where the majority class makes up 86% of all the samples.

Clue

For the Clue dataset, we seek to predict the discontinuation of birth control methods over time. Clue samples contain 70.29 % missing values on average [6.06, 81.45]. Since we're dealing with 4 different output classes (see Table 4), Clue requires *multi class* classification, where the majority class makes up 92 % of all the samples.

A.2 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

A.3 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 %.

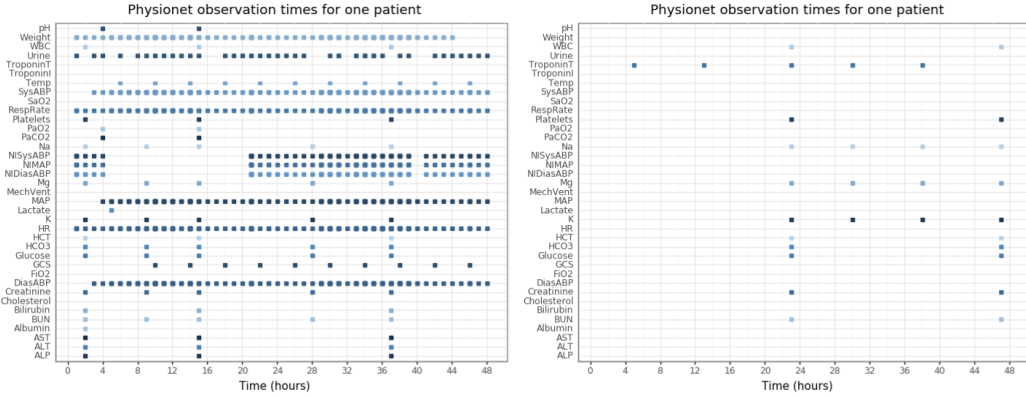


Figure 2: Physionet: Examples of time series for two different ICU patients with different frequencies of observations. On the left, we have a time series with a high frequency of observations. On the right is an example of a patient who is barely observed.

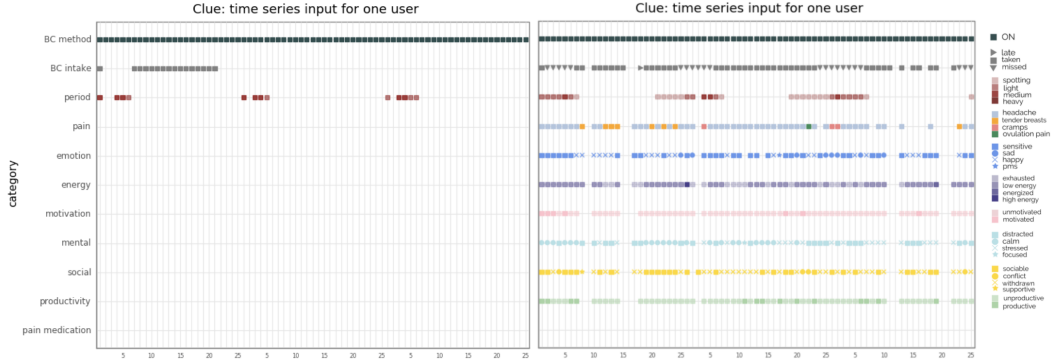


Figure 3: Clue: Example of the time series of the 3 input cycles for a user that tracks with low frequency (left), and a user that tracks with high frequency (right).

A.4 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.

A.5 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 [6]. 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 [6]. This allows the model to learn with more detail the time interval since a variable was last observed.

A.6 APC model architecture

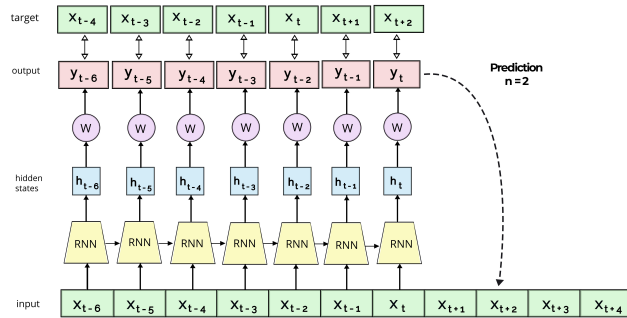


Figure 4: **Autoregressive Predictive Coding (APC)**: input sequence $\{x_t\}_{t=1}^N$ is encoded (e.g. with an RNN) at each time step to a hidden state h_t . A matrix W then transforms the hidden states to an output sequence $\{y\}_{t=1}^N$ of the same dimension as x_t .

A.7 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 encoder 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.

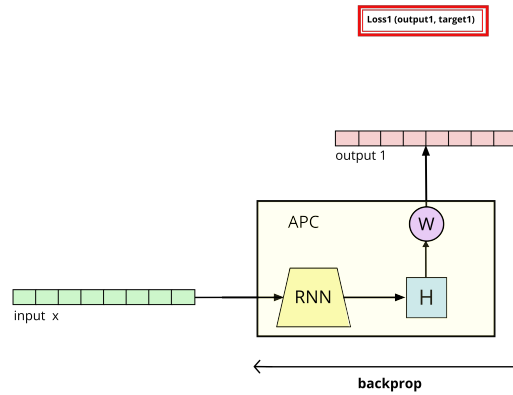


Figure 5: **Step 1:** The APC is pre-trained by optimizing the autoregressive loss

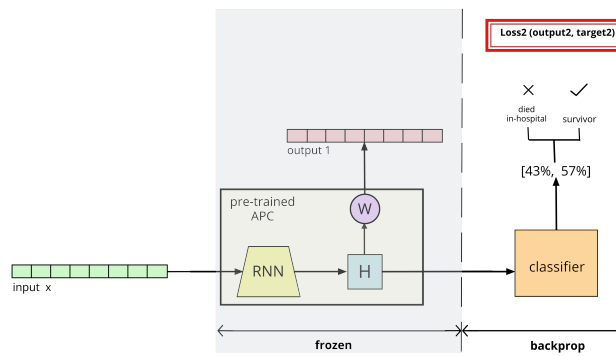


Figure 6: **Step 2:** The APC encoder weights are frozen & the classifier is trained by optimizing the cross-entropy loss.

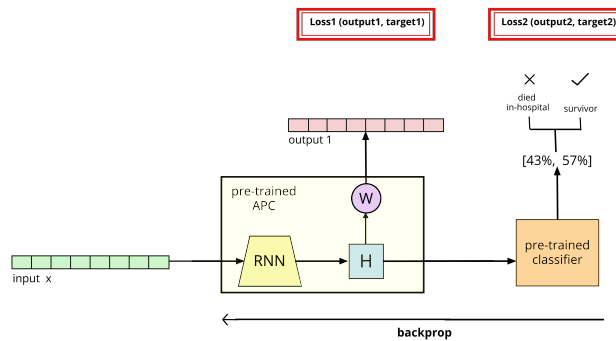


Figure 7: **Step 3:** Both the encoder & the classifier are trained end-to-end, using the pre-trained encoder & pre-trained classifier weights.

A.8 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 [12].

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 [12] 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, [12] 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.

A.9 Evaluation Metrics

Binary Classification

The AUROC score is given by the following formula:

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

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

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

$$\text{false positive rate} = \frac{\text{false positive}}{\text{false positive} + \text{true negative}} \quad (5)$$

$$\text{true negative rate} = \frac{\text{true negative}}{\text{true negative} + \text{false positive}} \quad (6)$$

However, when dealing with imbalanced classification, ROC curves may give an extremely optimistic view of the performance [2], 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 thresholds. The recall, also known as the true positive rate or sensitivity, is displayed in Equation 4, and the precision is defined as:

$$\text{precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}} \quad (7)$$

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-Recall 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** [20], which is the harmonic mean of the precision and recall:

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

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**:

$$\frac{\#\text{Class 1}}{\#\text{total}} \mathbf{x F1}_{\text{Class 1}} + \frac{\#\text{Class 2}}{\#\text{total}} \mathbf{x F1}_{\text{Class 2}} + \frac{\#\text{Class 3}}{\#\text{total}} \mathbf{x F1}_{\text{Class 3}} + \frac{\#\text{Class 4}}{\#\text{total}} \mathbf{x F1}_{\text{Class 4}} \quad (9)$$

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

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

A.10 Data imputation baselines

GRU-Mean:

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

GRU-Forward:

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

GRU-Simple:

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

A.11 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 [5] and using class weights. In this case, we oversample the minority class to 12% and then use class weights on the oversampled class frequency.

A.12 Hyperparameters

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

Model	Hyperparameters	Synthetic	Physionet	Clue
GRU	learning rate	1e-03	1e-03	1e-04
	batch size	32	32	100
	hidden units	64	64	200
	dropout	0.0	0.0	0.4
	recurrent dropout	0.0	0.0	0.1
	epochs	150	50	200
GRU-D	learning rate	1e-03	1e-03	1e-03
	batch size	32	32	100
	hidden units	32	32	200
	dropout	0.0	0.1	0.1
	recurrent dropout	0.0	0.0	0.0
	epochs	100	50	200
GRU - APC	learning rate - step 1	1e-03	1e-03	1e-04
	learning rate - step 2 & 3	1e-04	1e-04	1e-04
	batch size	32	32	100
	hidden units	120	64	200
	dropout	1.0	0.0,	0.4
	recurrent dropout	0.0	0.0	0.1
GRU-D - APC	epochs	100	100	50
	learning rate - step 1	1e-03	1e-03	1e-03
	learning rate - step 2 & 3	1e-04	1e-04	1e-04
	batch size	32	100	100
	hidden units	250	256	250
	dropout	0.0	0.1	0.1
	recurrent dropout	0.0	0.0	0.0
	epochs - step 1	100	100	50
epochs - step 2 & 3	100	50	50	

A.13 Results

Table 6: Classification results on the synthetic dataset.

CLASS (IM)BALANCE	MISSING DATA	AUPRC (MEAN \pm SD)			
		GRU	GRU-D	GRU-APC	GRU-D-APC
PERFECTLY BALANCED (1:1)	0 %	98.97 \pm 0.1	99.01 \pm 0.24	98.87 \pm 0.39	98.85 \pm 0.2
	30 %	98.4 \pm 0.16	98.1 \pm 0.22	98.24 \pm 0.53	98.07 \pm 0.44
	60 %	60.68 \pm 12.6	90.33 \pm 0.22	96.67 \pm 0.76	93.14 \pm 1.67
SLIGHT IMBALANCE (3:7)	0 %	97.32 \pm 0.45	96.88 \pm 0.61	97.26 \pm 0.4	96.98 \pm 1.21
	30 %	96.45 \pm 0.58	92.22 \pm 0.43	94.55 \pm 1.33	91.03 \pm 1.05
	60 %	66.19 \pm 10.54	80.55 \pm 1.94	90.57 \pm 0.52	83.06 \pm 1.52
SEVERE IMBALANCE (1:20)	0 %	76.05 \pm 5.11	83.16 \pm 2.03	92.31 \pm 0.88	80.22 \pm 2.95
	30 %	8.56 \pm 0.58	53.05 \pm 22.12	79.9 \pm 3.8	63.94 \pm 8.14
	60 %	6.61 \pm 0.5	35.54 \pm 4.44	33.56 \pm 21.06	66.42 \pm 5.42

Table 7: 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 \pm 0.3	48.4 \pm 1.3
GRU	-	OVERSAMPLING	84.6 \pm 0.5	51.4 \pm 1.7
GRU	-	CLASS WEIGHTS	85.4 \pm 0.4	51.4 \pm 0.9
GRU - APC	FROZEN	NONE	85.1 \pm 0.2	51.8 \pm 0.2
GRU - APC	FROZEN	CLASS WEIGHTS	85.0 \pm 0.2	51.3 \pm 0.1
GRU - APC	FINE-TUNED	NONE	86.0 \pm 0.5	54.1 \pm 1.0
GRU - APC	FINE-TUNED	CLASS WEIGHTS	85.9 \pm 0.3	53.5 \pm 0.5
GRU-D - APC	FROZEN	NONE	85.1 \pm 0.9	54.0 \pm 2.3
GRU-D - APC	FROZEN	CLASS WEIGHTS	85.3 \pm 0.1	55.1 \pm 0.9
GRU-D - APC	FINE-TUNED	NONE	85.2 \pm 0.9	54.1 \pm 2.3
GRU-D - APC	FINE-TUNED	CLASS WEIGHTS	85.3 \pm 0.2	55.0 \pm 0.9

Table 8: 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 \pm 0.0	19.7 \pm 4.6	28.2 \pm 1.7	17.1 \pm 1.4	90.4 \pm 0.2	21.5 \pm 2.1
FINE-TUNED	0	96.7 \pm 0.1	22.0 \pm 12.3	29.8 \pm 2.4	14.6 \pm 0.4	90.4 \pm 0.7	22.8 \pm 6.6
FROZEN	1	96.7 \pm 0.0	24.8 \pm 6.4	24.1 \pm 3.2	16.6 \pm 2.9	90.5 \pm 0.3	23.2 \pm 3.0
FINE-TUNED	1	96.7 \pm 0.0	28.8 \pm 2.2	25.7 \pm 2.4	14.8 \pm 2.5	90.7 \pm 0.1	25.7 \pm 1.1
FROZEN	2	96.7 \pm 0.1	27.3 \pm 4.0	22.9 \pm 3.3	16.2 \pm 3.5	90.6 \pm 0.3	24.3 \pm 2.5
FINE-TUNED	2	96.6 \pm 0.3	23.3 \pm 14.0	24.0 \pm 2.4	14.6 \pm 1.5	90.3 \pm 0.9	22.1 \pm 7.3
FROZEN	5	96.5 \pm 0.2	24.7 \pm 5.9	17.4 \pm 9.7	17.4 \pm 2.2	90.2 \pm 0.5	21.6 \pm 4.5
FINE-TUNED	5	96.0 \pm 0.7	16.6 \pm 8.9	14.8 \pm 13.0	15.0 \pm 0.6	89.2 \pm 1.3	15.9 \pm 8.2

Table 9: 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	90.3 ± 0.0	27.3 ± 0.5
FINE-TUNED	1	96.0 ± 0.5	30.3 ± 0.2	21.7 ± 4.0	13.7 ± 2.2	90.0 ± 0.5	25.3 ± 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

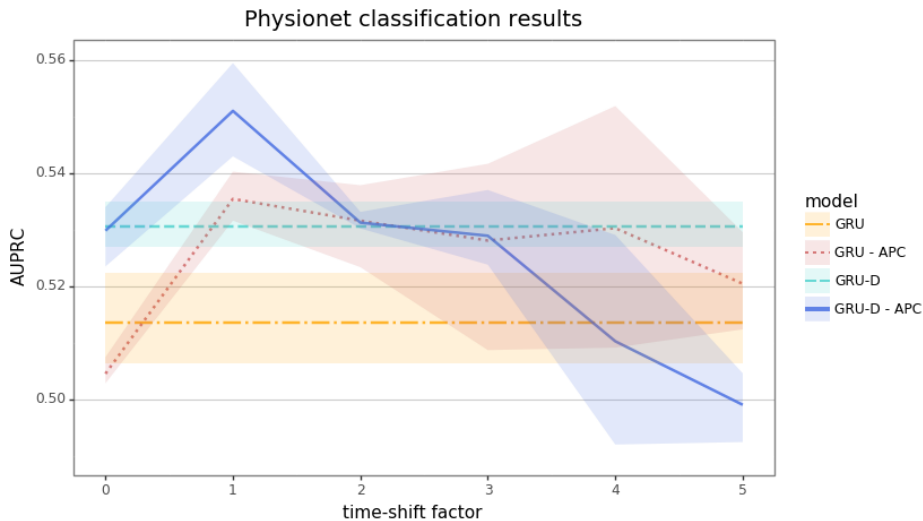


Figure 8: Physionet classification results for different values of the time shift factor n . All models are implemented with class weights.