Project 2 Report: Time Series and Representation Learning

Mateo Boglioni, Federica Bruni, Paula Momo Cabrera

May 21, 2024

1 Part 1: Supervised Learning on Time Series

In this part of the project, we investigated various machine learning models for supervised learning on time series data, focusing exclusively on the PTB Diagnostic ECG Database [BKS95] . The primary objective was to classify ECG time series as either healthy or abnormal. The PTB Diagnostic ECG Database includes digitized ECG recordings collected from both healthy individuals and patients with different heart conditions.

1.1 Exploratory Data Analysis

1.1.1 PTB Dataset

Firstly, we began our data exploration by loading the provided training and testing PTB datasets, revealing that the training set comprises 11,640 samples with 188 features each. The dataset was provided already preprocessed to ensure consistency, with all samples cropped, downsampled, and zero-padded to a fixed dimension of 188. The data was provided in two CSV files, representing training and testing subsets. Each row in these files corresponds to an individual ECG sample, with the final element indicating the class label.

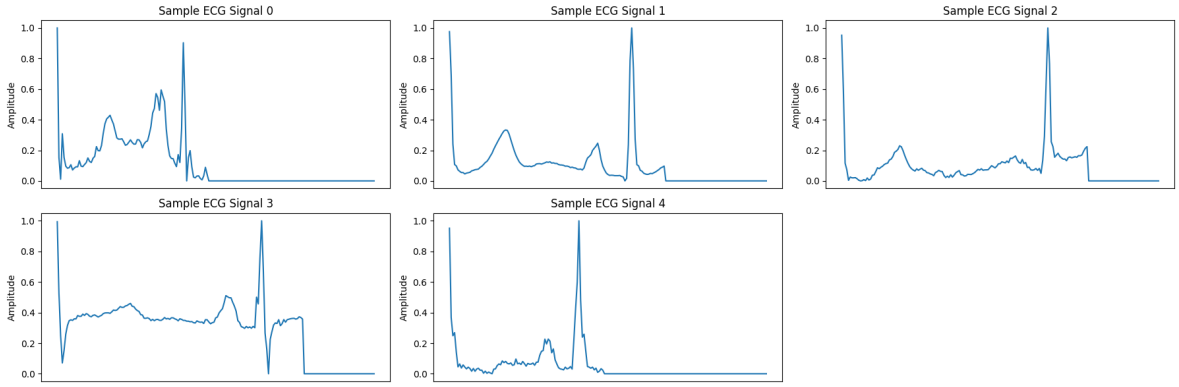

Figure 1: Sample ECG signals over time series

A thorough check for missing values confirmed the training dataset's cleanliness, with no null values detected across the time-series. Also, to gain a visual understanding of the ECG signals, we selected five random samples and plotted their amplitude over time, which provided insights into the nature of the ECG signals in both healthy and abnormal classes, helping us observe the variability and patterns in the ECG signals [BOSB10] (Figure 1).

By examining the class distribution on our training dataset unveiled a class imbalance, with significantly more instances of sick samples compared to healthy ones (Figure 2). This imbalance informs us that the dataset is skewed towards sick samples, which can have significant implications for model training and evaluation. Models trained on imbalanced datasets might become biased towards the majority class, in this case, the sick samples, leading to poor performance in correctly identifying the minority class. Consequently, to prepare the data, we separated the features and labels for both the

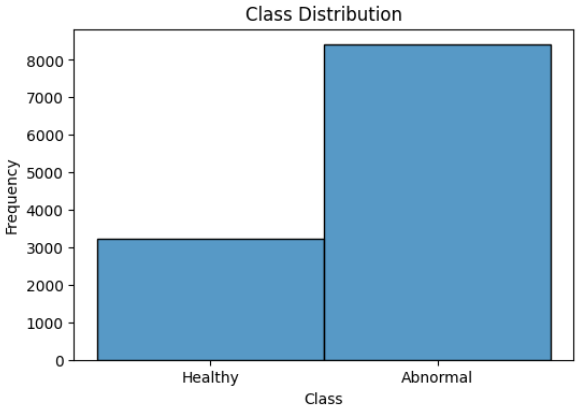

Figure 2: Training dataset class distribution. Myocardial infarction (abnormal) and Healthy controls (healthy)

training and test datasets. Also, to create a validation set, we split the training data into training and validation sets using an 85-15 split, ensuring the data was shuffled to maintain randomness.

To address the above mentioned class imbalance, we utilized the balanced accuracy score from *sklearn.metrics* [BOSB10], which computes the balanced accuracy in binary and multiclass classification problems. The balanced accuracy score is defined as the average of recall obtained on each class, providing a more informative metric for imbalanced datasets compared to standard accuracy. The balanced accuracy ranges from 0 (worst) to 1 (best), with a score adjusted for chance to ensure that random performance scores 0, while perfect performance scores 1. This approach helps to ensure that the model performs well across both classes, rather than being skewed towards the majority class.

Metrics for classification task

For the classification task, we chose balanced accuracy as the primary evaluation metric. While categorical accuracy is commonly used, balanced accuracy was more appropriate for this dataset due to the inherent class imbalance. Balanced accuracy ensures that the model's performance is evaluated fairly across all classes by considering both sensitivity and specificity. This metric provides a more comprehensive evaluation of the model's effectiveness in distinguishing between healthy and abnormal ECG signals. By performing these steps, we ensured that the data was well-understood and appropriately prepared for the supervised learning task, setting a solid foundation for building and evaluating machine learning models for ECG classification.

1.2 Classic Machine Learning Methods

Classic machine learning (ML) methods, such as Logistic Regression and Random Forests, operate on a fixed set of features and are capable of learning linear or non-linear combinations of these features to make predictions. These methods inherently provide insights into feature importance but require careful feature engineering, often leveraging domain-specific knowledge.

1.2.1 Initial Training on Raw Time Series Data:

To begin with, we trained two classic ML classifiers—Random Forest and Logistic Regression—using the raw time series data from the PTB dataset. This involved using the raw ECG signal data as features without adding any new features.

- The Random Forest classifier achieved a balanced accuracy score of 0.9499.
- The Logistic Regression classifier obtained a balanced accuracy score of 0.9499.

These initial results indicate that both models performed equally when trained on raw data, although the Random Forest model is typically more robust in handling non-linear relationships and interactions among features.

1.2.2 Enhanced Training with Feature Engineering:

Recognizing the importance of feature design and engineering for classic ML methods, we proceeded to enhance our models by adding engineered features derived from the raw time series data. Using signal processing techniques and leveraging the *tsfresh* library [CBNKL18], we extracted additional features that capture various statistical properties of the ECG signals, such as mean, variance, skewness, kurtosis, and other domain-specific features.

After incorporating these new features, we retrained both the Random Forest and Logistic Regression classifiers and observed improvements in their performance metrics.

- The Random Forest classifier improved its balanced accuracy score to 0.9507.
- The Logistic Regression classifier also improved, achieving a balanced accuracy score of 0.9507.

1.2.3 Evaluation of Classifiers:

Evaluating the pros and cons of the two classifiers used, we observed the following:

Random Forest Classifier:

- Pros: Robust performance with high balanced accuracy, inherently handles non-linearity and feature interactions well, provides insights into feature importance.
- Cons: Computationally intensive, especially with large datasets, can be prone to overfitting if not properly tuned.

Logistic Regression Classifier:

- Pros: Simplicity and ease of implementation, efficient and fast training, interpretable coefficients that provide insights into feature importance.
- Cons: Limited ability to capture non-linear relationships unless features are transformed; performance heavily dependent on proper feature scaling and engineering.

In conclusion, while both classifiers benefited from feature engineering, the slight improvements in balanced accuracy for both models indicate that feature engineering is beneficial but may not drastically change performance in this particular context. The Random Forest model's robustness and ability to handle non-linearity make it a strong candidate, while Logistic Regression remains valuable for its simplicity and interpretability. This highlights the importance of selecting appropriate models and leveraging domain-specific knowledge for feature engineering in classic ML tasks.

1.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are designed to handle sequential data of variable lengths by applying the same network to each time point and preserving a notion of memory through a hidden state vector. However, RNNs suffer from vanishing or exploding gradient issues, which Long Short-Term Memory (LSTM) networks address using gating mechanisms to control the flow of information in and out of the memory cell [HS97]. To demonstrate the application of LSTM networks, we implemented and trained an LSTM model on the PTB dataset.

1.3.1 LSTM Implementation and Training on PTB Dataset

We began by implementing an LSTM network with a single LSTM layer containing 50 units, followed by a dense layer with a sigmoid activation function for binary classification. The model was compiled using the Adam optimizer and binary cross-entropy loss. During training, we applied early stopping and learning rate reduction techniques to prevent overfitting, monitoring the validation loss to ensure optimal performance. After training the model for 60 epochs, the LSTM network achieved a balanced accuracy of 81.13% on the test set (Figure 3). This result indicates that the LSTM model was effective in capturing the temporal dependencies within the ECG data, but there was room for improvement.

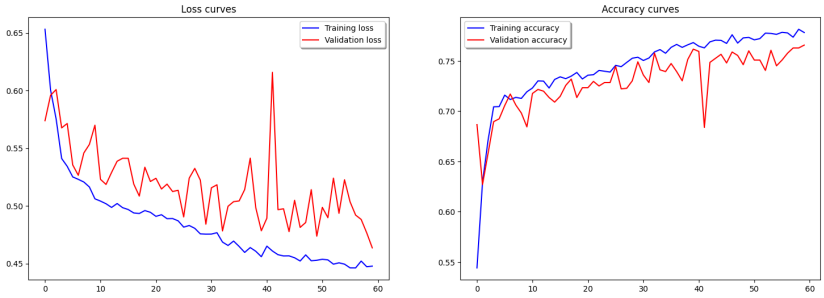

Figure 3: Performance of LSTM network

1.3.2 Utility of Bidirectional Models

Vanilla implementations of RNNs and LSTMs are unidirectional, processing time series data sequentially in one direction. However, bidirectional models can be advantageous in scenarios where future context is as important as past context. In a bidirectional LSTM, two hidden states are maintained: one for processing the sequence from the beginning to the end and another for processing it from the end to the beginning. This approach allows the model to have a more comprehensive understanding of the sequence, capturing dependencies from both directions [SP97].

1.3.3 Implementation and Training of Bidirectional Model

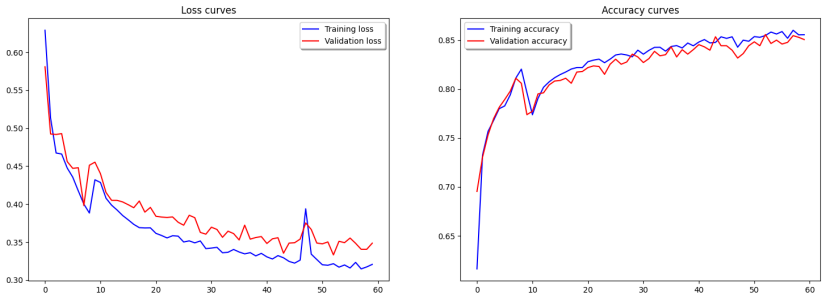

Figure 4: Performance of bidirectional LSTM

To explore the benefits of bidirectional models, we implemented a bidirectional LSTM network. This model processes the input sequence in both directions, enhancing its ability to capture complex patterns in the data. The bidirectional LSTM was configured similarly to the unidirectional LSTM, with early stopping and learning rate reduction applied during training. The training and validation curves indicated improved learning dynamics compared to the unidirectional LSTM.

The bidirectional LSTM model achieved a balanced accuracy of 86.38% on the test set (Figure 4), demonstrating its superior ability to leverage the full context of the sequential data. This result confirms that bidirectional models are beneficial in capturing the intricate temporal dependencies inherent in ECG signals, providing a more comprehensive understanding of the data compared to unidirectional models. This enhancement highlights the importance of considering bidirectional architectures for tasks involving sequential data with long-range dependencies.

1.4 Convolutional Neural Networks

1.4.1 Advantages of RNNs and CNNs for Time Series Tasks

RNNs and Convolutional Neural Networks (CNNs) offer unique advantages for time series tasks, leveraging different mechanisms to capture dependencies within data. RNNs are specifically designed to handle sequential data, making them adept at capturing temporal dependencies. They process inputs in sequence, maintaining a hidden state that carries information from previous time steps. This characteristic makes RNNs particularly useful for tasks where the order of data points is crucial, such as

language modeling, speech recognition, and any time series forecasting where long-term dependencies are important.

On the other hand, CNNs are traditionally used in image processing due to their ability to capture spatial hierarchies through convolutional layers. When applied to time series tasks, CNNs excel in detecting local patterns and features. They can effectively capture short-term dependencies and perform well in tasks where identifying local motifs is more important than long-term temporal relationships. Additionally, CNNs offer robustness against spatial translations and are computationally more efficient than RNNs, as they allow parallel processing of data through convolutional operations.

1.4.2 Implementation and Training of CNN Models

To demonstrate the relative performance of different CNN architectures on time series data, we implemented and trained two models: a vanilla CNN and a CNN with residual blocks. The vanilla CNN consisted of two convolutional layers followed by max-pooling layers and fully connected layers. The residual CNN incorporated residual blocks, which include shortcut connections allowing gradients to bypass certain layers, thus mitigating the vanishing gradient problem and facilitating the training of deeper networks [HZRS16].

Both models were trained on the PTB dataset using similar training loops, with the Adam optimizer and cross-entropy loss. The training process spanned ten epochs, with performance metrics including loss and balanced accuracy recorded for both training and test datasets.

1.4.3 Test Performance and Findings

The performance metrics for both models are summarized as follows:

- **Vanilla CNN: Test Balanced Accuracy:** 94.90%.
- **Residual CNN: Test Balanced Accuracy:** 97.63%.

The results indicate that both models performed well, with the residual CNN showing superior performance in terms of both loss reduction and metric improvements. The inclusion of residual blocks in the CNN architecture significantly enhanced the model's ability to learn and generalize, likely due to the improved gradient flow and reduced risk of vanishing gradients. Residual connections allowed for more effective training of deeper networks, which in turn facilitated the capture of more complex patterns within the time series data.

Overall, while both RNNs and CNNs have their advantages for time series tasks, the CNN with residual blocks demonstrated enhanced performance, showcasing the benefits of deeper architectures and efficient training mechanisms for capturing intricate data dependencies.

1.5 Attention and Transformers

1.5.1 Implementation and Training of a Transformer Model

The attention mechanism in neural networks helps capture longer-range dependencies in data, which can be particularly useful in time series analysis . To leverage this mechanism, we implemented a simple transformer model and trained it on the PTB dataset. Our transformer model architecture consisted of multiple transformer encoder blocks, each comprising a multi-head self-attention layer and a feed-forward neural network layer. We used layer normalization and dropout for regularization. The final model included global average pooling followed by dense layers with dropout for classification.

The transformer model was compiled using the Adam optimizer and binary cross-entropy loss. We trained the model with early stopping to avoid overfitting. After training, the model achieved a test accuracy of 94.2933%, significantly higher than the accuracy achieved by recurrent neural network models on the same task.

1.5.2 Differences and Advantages of Transformers Over RNNs

Transformers offer several advantages over RNNs. Unlike RNNs, transformers do not require sequential processing of input data, which allows for parallelization during training and inference, leading to faster processing times. Additionally, the self-attention mechanism in transformers can capture dependencies

regardless of their distance in the input sequence, whereas RNNs are limited by their sequential nature and suffer from vanishing or exploding gradients when dealing with long-range dependencies. These characteristics make transformers particularly effective for tasks involving long sequences and complex relationships, such as those found in the PTB dataset.

1.5.3 Visualization of Attention Maps

To gain insights into the model's decision-making process, we visualized the attention maps learned by our transformer model. Specifically, we examined the input layer weights to understand the importance of different data points in distinguishing between normal and abnormal ECG sequences.

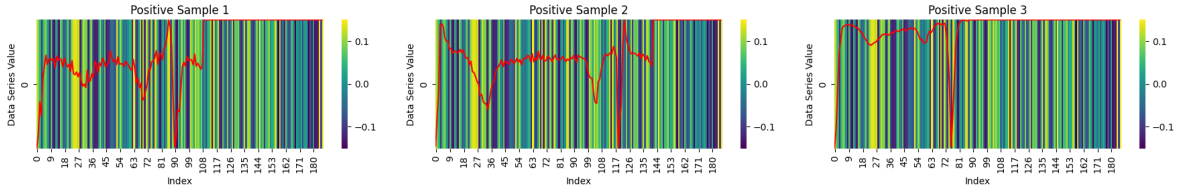

Figure 5: Attention maps for positive (abnormal) samples

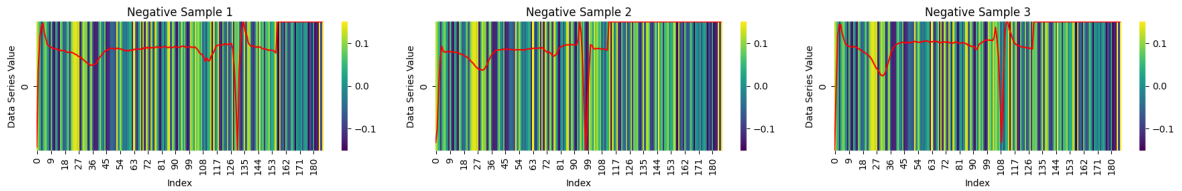

Figure 6: Attention maps for negative (normal) samples

For this analysis, we selected a few positive (abnormal) (Figure 5) and negative (normal) samples (Figure 6) from the training set. We plotted the attention weights alongside the ECG data to visualize which parts of the sequence were deemed most important by the model.

The attention maps revealed that certain segments of the ECGs (eg. areas close to index 27) consistently received higher attention scores for both positive and negative samples, indicating their significance in the classification task. Given that inputs weights are expected to be the same for both sample classes, these visualizations still provide valuable insights into how the transformer model identifies potential features of interest in medical time series data.

2 Part 1: General Questions

2.0.1 Q1: Competitiveness of Classic Methods with Deep Learning Architectures

In this project, we observed that classic machine learning methods, such as Logistic Regression and Random Forests, can still be competitive with deep learning architectures. For instance, both the Random Forest and Logistic Regression classifiers achieved balanced accuracy scores of 0.9499 on the raw time series data from the PTB dataset. This indicates that classic methods can perform on par with deep learning models in certain scenarios. The competitiveness of classic methods can be attributed to several factors. Classic methods often require less computational power and are easier to interpret, making them suitable for scenarios with limited resources or where model interpretability is crucial. Moreover, with appropriate feature engineering, classic methods can capture essential patterns in the data, thereby reducing the performance gap with deep learning models. However, deep learning architectures, particularly those with attention mechanisms, demonstrated superior performance in capturing complex dependencies within the data, highlighting their potential in more intricate tasks.

2.0.2 Q2: Use of causal/unidirectional architectures in time series modeling

When modeling time series data using deep learning architectures, a causal or unidirectional architecture is often preferred in scenarios where the prediction at each time step depends only on past and present information, not future data points. An example of such a task is real-time financial forecasting, where the model needs to predict stock prices based on historical data up to the current moment without access to future prices. Using a unidirectional model, such as a causal RNN or LSTM, ensures that the predictions are made in a sequential manner, respecting the temporal order of the data and mimicking the real-world constraints of the task.

2.0.3 Q3: Attention-related bottlenecks in very long time series

One significant bottleneck of attention mechanisms in handling very long time series is their computational complexity. As highlighted by Vaswani et al. (2017) [VSP+17], the self-attention mechanism scales quadratically with the length of the input sequence, rendering it computationally expensive and memory-intensive for extremely long sequences. To mitigate this challenge, methods such as hierarchical attention networks have been proposed. These networks aim to hierarchically aggregate information from different levels of granularity, thereby providing an efficient solution for processing long documents or sequences [YYD+16].

2.0.4 Q4: Challenges in self-supervised representation learning

Self-supervised representation learning presents several challenges. One primary difficulty observed in this approach is the need for effective pretext tasks that can generate meaningful and diverse representations from the data without requiring labeled examples. In our approach, the selection of appropriate pretext tasks significantly influenced the quality of the learned representations. Another challenge is the potential for negative transfer, where the representations learned from the pretext task do not generalize well to the downstream task [VSP+17]. Additionally, the training process for self-supervised learning can be computationally intensive, requiring extensive resources. Other difficulties include balancing the trade-off between representation learning and task-specific fine-tuning, and dealing with noise and variability in the data, which can impact the robustness of the learned representations.

3 Part 2: Transfer and Representation Learning

In this section, we explore leveraging the larger MIT-BIH dataset [MM01] to enhance learning on the smaller PTB dataset. Transfer learning utilizes models trained on related tasks to improve performance, while representation learning uses unsupervised methods to learn data representations.

3.1 Supervised Model for Transfer

In this part of the project, we utilized a deep learning model, specifically a Transformer-based architecture, trained on the MIT-BIH arrhythmia dataset to explore transfer learning. The aim was to leverage this model for improving performance on a related task, such as working with the smaller PTB dataset.

3.1.1 Model Implementation and Training

The chosen model architecture is the Transformer encoder, which is well-suited for time-series data due to its ability to capture temporal dependencies effectively. The model included two transformer blocks, each consisting of multi-head attention layers and feed-forward networks, supplemented with layer normalization and dropout for regularization. This architecture helps in learning intricate patterns from the ECG signals.

The input shape for the model was set to (187, 1), matching the length and depth of the ECG signals. The global average pooling layer was employed to reduce the dimensionality of the data while retaining critical information. Following this, a multilayer perceptron (MLP) head with dense layers and dropout was used to output class probabilities for five different classes, reflecting the classification task at hand.

The model was compiled using categorical cross-entropy as the loss function and the Adam optimizer, which is known for its efficiency in handling such tasks. Early stopping was implemented to prevent overfitting by monitoring the validation loss and restoring the best weights if no improvement was observed over 15 epochs.

3.1.2 Test performance

Upon training, the model was evaluated on the MIT-BIH test set, achieving a balanced accuracy of 87.04%. Balanced accuracy was chosen as the evaluation metric because the dataset is imbalanced, and this metric ensures that the performance across all classes is fairly considered. This high balanced accuracy indicates that the model performs well across different types of arrhythmias, making it a robust choice for transfer learning.

3.1.3 Motivation for Model Choice and Metrics

The selection of a Transformer-based model was motivated by its proven capability to handle sequential data, like ECG signals, effectively. Transformers excel at learning long-range dependencies [VSP+17], which are crucial for accurately classifying arrhythmias. The use of multi-head attention allows the model to focus on various parts of the input sequence simultaneously, enhancing its ability to learn complex patterns.

The metrics chosen for evaluation are also crucial. While categorical accuracy is a standard metric, balanced accuracy was specifically highlighted due to the class imbalance inherent in arrhythmia datasets [BCV14]. Balanced accuracy ensures that the model's performance is not biased towards the majority class and provides a more holistic view of its effectiveness across all classes. the Transformer-based model trained on the MIT-BIH dataset not only demonstrated robust performance but also serves as a valuable pre-trained encoder for further transfer learning applications. This approach underscores the potential of using large, labeled datasets to improve outcomes on smaller, related tasks.

3.2 Representation Learning Model

3.2.1 Pretraining the Encoder

In this task, we pre-trained an encoder using a contrastive learning approach on the MIT-BIH dataset. The objective was to learn useful representations from the data without relying on labels, making it applicable for future tasks where labeled data might be scarce.

We adopted a contrastive learning method, where pairs of similar and dissimilar ECG signal segments were created. The idea is to train the model to minimize the distance between similar pairs and maximize the distance between dissimilar pairs. Specifically, we used the information noise-contrastive estimation (InfoNCE) loss [FH24], which is effective for such tasks.

The architecture of our encoder was designed to be similar to the one used in section 3.1 for a fair comparison. It consisted of transformer blocks to capture the temporal dependencies in the ECG signals. The output of the transformer blocks was aggregated using global average pooling to obtain a low-dimensional representation of the input signals.

3.2.2 Monitoring Pre-training

The pre-training process was monitored by evaluating the accuracy of the Siamese network [noa] on a validation set of paired signals. The network's performance, as indicated by the loss and accuracy metrics, provided insights into the quality of the learned representations.

3.3 Evaluation of Pre-training

To evaluate the quality of the learned representations, we used a Random Forest classifier trained on the representations extracted from the MIT-BIH training set. The classifier was then evaluated on the test set using balanced accuracy, the same metric employed in section 3.1.

3.3.1 Results

The table below summarizes the test performance of the models:

Model	Balanced Accuracy(%)
Supervised Transformer Model	87.04
Contrastive Learning Encoder with Random Forest	80.42

Table 1: Performance comparison of cupervised and celf-cupervised models on MIT-BIH test set

The supervised Transformer model achieved a balanced accuracy of 87.04%, while the contrastive learning approach followed by a Random Forest classifier yielded a balanced accuracy of 80.42%. This demonstrates that the representations learned via contrastive learning are effective but still fall short of the supervised approach. However, they provide a viable alternative when labeled data is limited.

In conclusion, the contrastive learning approach for representation learning successfully created meaningful representations that could be used for downstream tasks, albeit with slightly lower performance compared to the supervised model. This highlights the potential of self-supervised methods in scenarios where labeled data is difficult to obtain.

3.4 Visualising Learned Representations

In this section we visualized the representations learned by the encoders pre-trained in section 3.1 (supervised Transformer model) and in section 3.2 (contrastive learning encoder) using UMAP [MHM20], a dimensionality reduction technique. This visualization helps in understanding the distribution of data points with different labels and comparing the distributions of the MIT-BIH and PTB datasets.

3.4.1 First encoder (supervised transformer) on PTB dataset

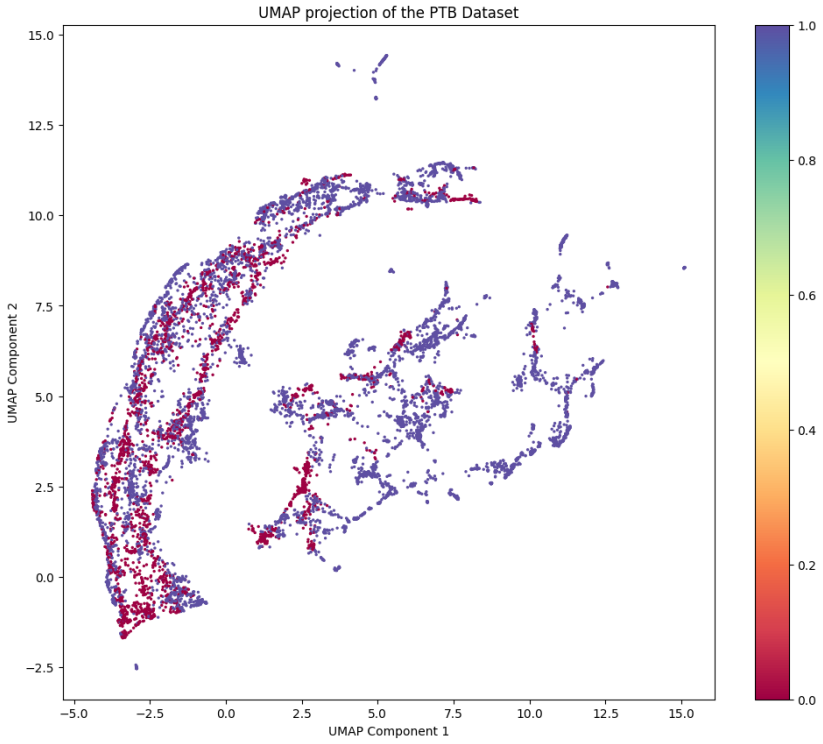

Figure 7: UMAP projection of the PTB Dataset using the supervised Transformer encoder trained on the MIT-BIH dataset

We obtained the representations of the PTB dataset by feeding the data through the supervised Transformer encoder trained on the MIT-BIH dataset. The UMAP visualization revealed distinct

clusters corresponding to different labels in the PTB dataset (positive =1 and negative= 0) (Figure 7), indicating that the supervised model was effective in capturing relevant features that separate different classes. This suggests that the representations learned by the supervised model generalize well to the PTB dataset.

3.4.2 Second encoder (contrastive learning encoder) on PTB dataset

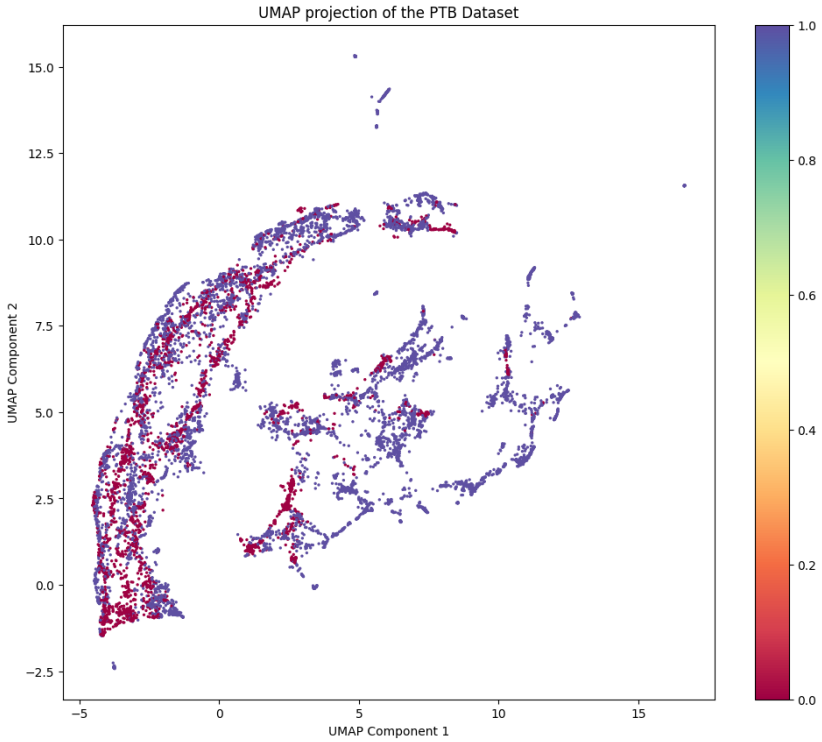

Figure 8: UMAP projection of the PTB Dataset using the contrastive learning encoder trained on the MIT-BIH dataset

Similarly, we obtained the representations of the PTB dataset using the encoder trained with contrastive learning. The UMAP visualization showed that the data points with different labels formed identifiable clusters (Figure 8), although the separation between some classes was less distinct compared to the supervised model. This implies that while the contrastive learning approach was able to capture useful representations, it might not be as discriminative as the supervised model for this particular task.

3.4.3 First encoder (supervised transformer) on MIT-BIH dataset

For the MIT-BIH dataset, the UMAP visualization of the representations obtained from the supervised Transformer encoder showed well-separated clusters for different arrhythmia classes (Figure 9). This clear separation highlights the model's ability to learn strong discriminative features from the MIT-BIH training data, which translates into high performance during classification.

3.4.4 Second encoder (contrastive learning encoder) on MIT-BIH dataset

The UMAP visualization of the MIT-BIH dataset using the contrastive learning encoder also displayed distinct clusters for various arrhythmia classes, comparable to that of the supervised model (Figure 10).

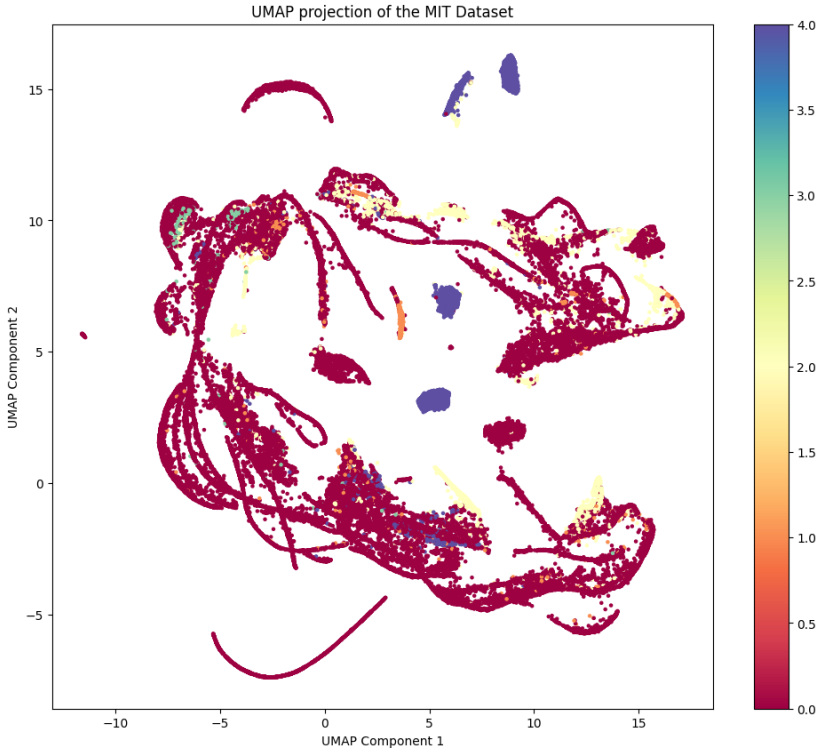

Figure 9: UMAP projection of the MIT-BIH dataset using the supervised Transformer encoder trained on the MIT-BIH dataset

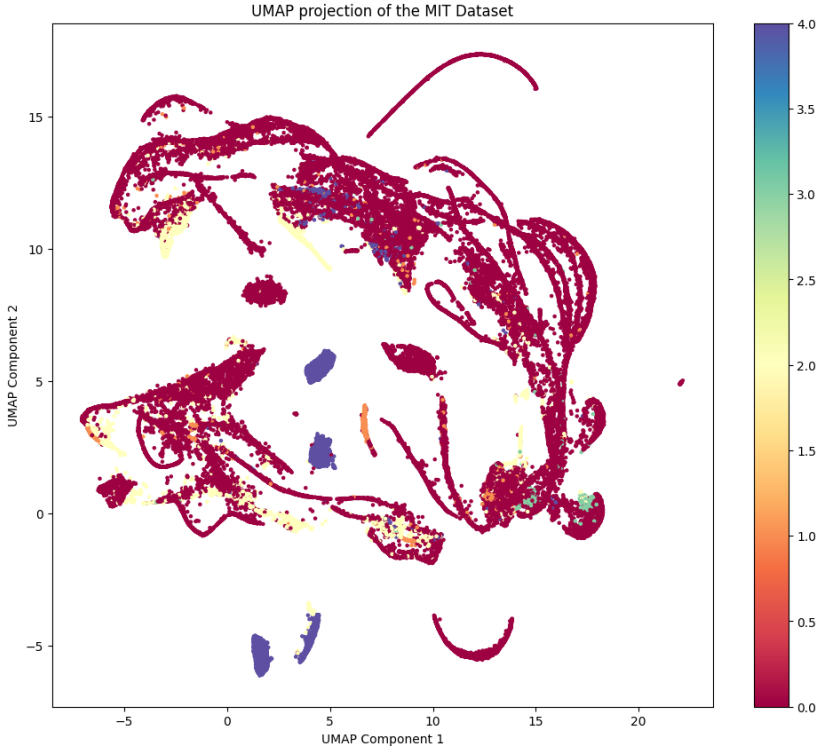

Figure 10: UMAP projection of the on MIT-BIH dataset using the contrastive learning encoder trained on the MIT-BIH dataset

3.4.5 Comparison of datasets distribution

To determine whether the MIT-BIH and PTB datasets are distributed identically, we conducted a quantitative assessment using a classification approach. We trained a Random Forest classifier to distinguish between the representations of the two datasets obtained from both the supervised Transformer encoder and the contrastive learning encoder.

First, we prepared the data by combining the UMAP embeddings of the PTB and MIT-BIH datasets. Labels were assigned as 0 for PTB and 1 for MIT-BIH. The combined data was then split into training and testing sets with a 70-30 split. Class weights were computed to handle any class imbalance in the training set.

We trained a Random Forest classifier using the training set and evaluated its balanced accuracy on the test set. For the supervised Transformer encoder, the balanced accuracy was 0.8761. This high balanced accuracy indicates that the classifier could effectively distinguish between the two datasets. Consequently, we reject the null hypothesis and conclude that the two datasets do not have the same distribution. Similarly, for the contrastive learning encoder, the balanced accuracy was also 0.8761. Again, this high balanced accuracy suggests that the classifier could differentiate between the two datasets. Therefore, we reject the null hypothesis and conclude that the two datasets do not have the same distribution.

By these balanced accuracy scores we could conclude that both the MIT-BIH and PTB datasets do not have the same distribution.

3.5 Finetuning Strategies

In this section, we explored various fine-tuning strategies to leverage our pre-trained encoder models for prediction on the PTB dataset. The goal was to evaluate the performance of different strategies and determine the best approach for this task.

3.5.1 Classic ML method

Similar to section 3.4, we obtained representations for the PTB dataset by feeding the dataset through the pre-trained encoder. We then used a classic ML method (Random Forest) to train and test for the PTB task using these representations.

Supervised Transformer Encoder:

- Test balanced accuracy: 95.2022

Contrastive Learning Encoder:

- Test balanced accuracy: 95.2022

3.5.2 MLP output layers

We added output layer(s) for the PTB binary classification task to our encoder model and implemented the following fine-tuning strategies:

3.5.3 Train the Output Layer(s) Only (Freezing the Encoder)

Supervised Transformer Encoder:

- Test balanced accuracy: 76.8016%

Contrastive Learning Encoder:

- Test balanced accuracy: 77.6152%

3.5.4 Train the Entire Model (Encoder + Output Layers)

Supervised Transformer Encoder:

- Test balanced accuracy: 76.5158%

Contrastive Learning Encoder:

- Test balanced accuracy: 77.6011%

3.5.5 First Train the Output Layers, Then Unfreeze and Train the Entire Model

Supervised Transformer Encoder:

- Test balanced accuracy: 77.4875%

Contrastive Learning Encoder:

- Test balanced accuracy: 77.6152%

3.5.6 Results summary

Strategy	Balanced Accuracy (%)
Supervised Transformer Encoder	
Classic ML (Random Forest)	95.2022
Train Output Layer Only	76.8016
Train Entire Model	76.5158
Pretrain Output Layers, then Train Entire Model	77.4875
Contrastive Learning Encoder	
Classic ML (Random Forest)	95.2022
Train Output Layer Only	77.6152
Train Entire Model	77.6011
Pretrain Output Layers, then Train Entire Model	77.6152

Table 2: Performance Comparison of Fine-tuning Strategies on PTB Dataset

As seen in Table 2, the highest balanced accuracy was achieved using the Classic ML method (Random Forest) with representations from both encoders, suggesting that this method leverages the pre-trained features most effectively without further modification. Among the fine-tuning strategies, training the output layers only, while freezing the encoder, provided the best results for the contrastive learning encoder, closely followed by the strategy of pretraining the output layers and then training the entire model. This outperformance of Random Forest was expected, given that the use of a dense layer trained on the PTB dataset may require further optimization. Dense layers typically necessitate extensive hyperparameter tuning and sufficient training data to generalize well, whereas Random Forests are less sensitive to such factors due to their ensemble nature, which mitigates overfitting and leverages the inherent structure of the data more effectively. Additionally, Random Forests can handle variations in the pre-trained feature representations more robustly without the need for additional modifications. The supervised Transformer encoder showed similar trends but with slightly lower performance compared to the contrastive learning encoder in these fine-tuning approaches.

3.5.7 Best performing strategy

The Classic ML method using Random Forest provided the best performance, with a balanced accuracy of 95.2022% for both encoders. This indicates that the pre-trained features are already highly discriminative, and adding more complexity through fine-tuning does not necessarily improve the performance.

3.5.8 Comparison of Pre-training strategies

The performance of the supervised Transformer encoder and the contrastive learning encoder was comparable when using the Classic ML method. However, in fine-tuning scenarios, the contrastive learning encoder generally performed better, suggesting that representations learned through self-supervised methods can be more robust and adaptable when labeled data is scarce. This highlights the potential of contrastive learning in scenarios where labeled data is limited, providing a flexible and effective alternative to supervised pre-training.

4 Part 2: General Questions

4.0.1 Q1: Competitiveness of Classic Methods with Deep Learning Architectures

In this project, we observed that classic machine learning methods, specifically the Random Forest classifier, performed competitively with deep learning architectures. This was particularly evident when evaluating the representations obtained from the pre-trained encoders. The Random Forest classifier achieved a balanced accuracy of 95.2022% for both the supervised Transformer encoder and the contrastive learning encoder representations on the PTB dataset. This competitive performance can be attributed to several factors. First, the pre-trained encoders generated high-quality representations that captured essential features of the data. These representations were highly discriminative, allowing classic methods like Random Forest to perform well without additional complexity. Second, classic methods are often simpler and more interpretable than deep learning models. They require less computational power and are easier to tune, which can be advantageous when working with smaller datasets or when interpretability is crucial. Lastly, classic methods like Random Forest are less prone to overfitting, especially when the dataset is small or imbalanced. This robustness makes them a reliable choice in various machine learning settings. Thus, the competitive performance of classic methods in this project highlights the importance of choosing the right tool for the task at hand, considering both the quality of representations and the specific characteristics of the data.

4.0.2 Q2: Use of causal/unidirectional architectures in time series modeling

As already explained in section 2.0.2, a causal or unidirectional architecture is desired when the prediction at any time step should only depend on past information and not on future information. Another example is predicting patient health outcomes in an intensive care unit (ICU). Here, the model must base its predictions on previously recorded vital signs and medical history without future data, enabling timely and accurate interventions.

4.0.3 Q3: Attention-related bottlenecks in very long time series

An attention-related bottleneck in dealing with very long time series is the quadratic complexity of the attention mechanism. The computational and memory requirements of self-attention scale quadratically with the length of the input sequence, making it challenging to handle very long sequences efficiently. This bottleneck can lead to significant resource constraints and slower training times. For such long time series, methods that reduce the complexity of the attention mechanism are more suitable. One such method is the use of Transformer models with sparse attention mechanisms, which selectively focus on relevant parts of the sequence rather than the entire sequence. Additionally, models like the Longformer [BPC20] or the Reformer [KKL20], which are designed to handle longer sequences by incorporating sparse attention patterns or more efficient memory management, are conceptually more suitable for very long time series.

4.0.4 Q4: Challenges in self-supervised representation learning

Using self-supervised representation learning presents several challenges. While self-supervised methods can learn useful representations without labeled data, the quality of these representations may not always match those obtained from supervised learning. In this project, the contrastive learning encoder showed slightly lower performance compared to the supervised Transformer encoder, indicating room for improvement in the learned representations. Self-supervised learning methods often require careful tuning of hyperparameters, such as the choice of augmentations, the structure of the loss function, and the architecture of the model. This tuning process can be complex and time-consuming. Additionally, the learned representations may not always generalize well to downstream tasks. Fine-tuning may be required to adapt the representations to specific tasks, which can be challenging if the labeled data for the target task is scarce. Self-supervised learning methods, especially those involving contrastive learning, can also be computationally intensive due to the need for generating and processing large numbers of augmented data pairs. In addition to these observed difficulties, other challenges include ensuring robustness to noise and variations in the data, dealing with domain shifts between pre-training and target tasks, and developing methods to automatically evaluate and improve the quality of learned representations without extensive labeled data.

References

[BCV14] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation Learning: A Review and New Perspectives, April 2014. arXiv:1206.5538 [cs].

[BKS95] R. Bousseljot, D. Kreiseler, and A. Schnabel. Nutzung der EKG-Signaldatenbank CARDIODAT der PTB über das Internet. 40(s1):317–318, January 1995. Publisher: De Gruyter Section: Biomedical Engineering / Biomedizinische Technik.

[BOSB10] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M. Buhmann. The Balanced Accuracy and Its Posterior Distribution. In *2010 20th International Conference on Pattern Recognition*, pages 3121–3124, August 2010. ISSN: 1051-4651.

[BPC20] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The Long-Document Transformer, December 2020. arXiv:2004.05150 [cs].

[CBNKL18] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W. Kempa-Liehr. Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package). *Neurocomputing*, 307:72–77, September 2018.

[FH24] Patrick Feeney and Michael C. Hughes. SINCERE: Supervised Information Noise-Contrastive Estimation REvisited, February 2024. arXiv:2309.14277 [cs].

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997. Conference Name: Neural Computation.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. pages 770–778, 2016.

[KKL20] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The Efficient Transformer, February 2020. arXiv:2001.04451 [cs, stat].

[MHM20] Leland McInnes, John Healy, and James Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction, September 2020. arXiv:1802.03426 [cs, stat].

[MM01] G.B. Moody and R.G. Mark. The impact of the MIT-BIH Arrhythmia Database. *IEEE Engineering in Medicine and Biology Magazine*, 20(3):45–50, May 2001. Conference Name: IEEE Engineering in Medicine and Biology Magazine.

[noa] Siamese Neural Network - an overview | ScienceDirect Topics.

[SP97] M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, November 1997.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[YYD+16] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical Attention Networks for Document Classification. In Kevin Knight, Ani Nenkova, and Owen Rambow, editors, *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California, June 2016. Association for Computational Linguistics.