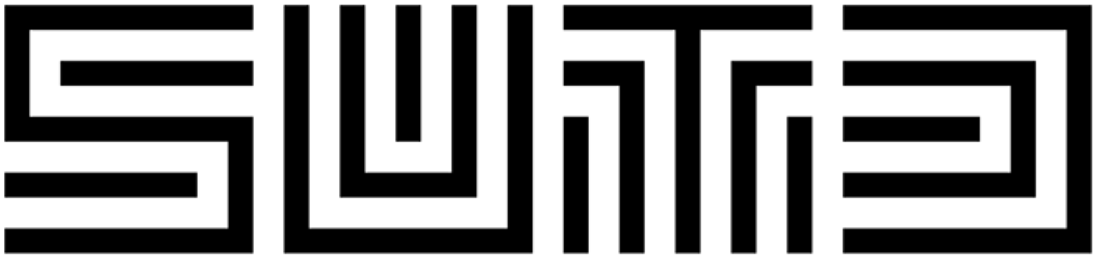


SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

50.039 Theory and Practice of Deep Learning
PROJECT REPORT

Alphonsus Tan (1005534)
Tej Deep Pala (1005282)

Table of contents

Problem statement

Cardiovascular disease is a leading cause of mortality worldwide, and its diagnosis and management require timely and accurate identification of cardiac abnormalities. Electrocardiogram (ECG) signals provide a non-invasive diagnostic tool for the detection of various cardiac conditions, such as myocardial infarction, conduction disturbances, and hypertrophy. However, the manual interpretation of ECG signals can be time-consuming and subject to inter-observer variability, leading to errors and delayed diagnosis.

Moreover, ECG signals alone may not always provide sufficient information for accurate diagnosis. Clinical notes, which contain information about a patient's medical history, symptoms, and physical examination, can provide valuable context and additional features for accurate diagnosis of cardiac abnormalities.

To address these challenges, this project aims to develop and evaluate deep learning models that leverage both ECG signals and clinical notes for the automated classification of cardiac abnormalities using the PTB-XL dataset. The proposed models will incorporate multi-modal inputs and augment the data using random noise and random masking to improve performance and robustness.

The classes for classification in the PTB-XL dataset include NORM (Normal ECG), MI (Myocardial Infarction), STTC (ST/T Change), CD (Conduction Disturbance), and HYP (Hypertrophy). This project will focus on accurately classifying these cardiac conditions using deep learning-based algorithms.

The proposed deep learning models will leverage the temporal and spectral information in ECG signals and the textual information in clinical notes to accurately classify different cardiac abnormalities. The project aims to address the limitations of traditional manual interpretation of ECG signals and clinical notes and enable more efficient and accurate diagnosis of cardiac abnormalities using multi-modal inputs and deep learning-based algorithms.

Dataset

PTB-XL

The PTB-XL dataset is a publicly available dataset of electrocardiogram (ECG) signals and associated clinical notes. The dataset includes 21799 clinical 12-lead ECGs from 18869 patients of 10 second length. The raw waveform data was annotated by up to two cardiologists, who assigned potentially multiple ECG statements to each record. There are in total 71 different ECG statements that conform to the SCP-ECG standard and cover diagnostic, form, and rhythm statements. To ensure comparability of machine learning algorithms trained on the dataset, the authors of this dataset also provided recommended splits into training and test sets.

The use of this dataset makes the objective of this project a multi-label, multi-class classification problem because each ECG recording can have multiple labels corresponding to different cardiac abnormalities, and each label corresponds to a specific class. More specifically, there are five classes in the PTB-XL dataset:

- NORM (normal ECG): This class corresponds to ECG recordings that do not show any signs of cardiac abnormalities.

- MI (Myocardial Infarction): This class corresponds to ECG recordings that show evidence of a myocardial infarction, which is a heart attack caused by the blockage of blood flow to the heart muscle.

- STTC (ST/T Change): This class corresponds to ECG recordings that show changes in the ST or T waves, which can be indicative of various cardiac abnormalities such as myocardial ischemia or infarction.

- CD (conduction disturbance): This class corresponds to ECG recordings that show evidence of conduction disturbances, which can be caused by various conditions such as heart disease or drug toxicity.

- HYP (hypertrophy): This class corresponds to ECG recordings that show evidence of left ventricular hypertrophy, which is an enlargement of the heart muscle that can be caused by various conditions such as high blood pressure or aortic stenosis.

Because each ECG recording can have multiple labels corresponding to different cardiac abnormalities, the classification problem is multi-label. Additionally, because there are multiple classes that each ECG recording can be labeled as, the problem is also multi-class. This makes the classification task complex and challenging, but also more representative of real-world scenarios where patients can have multiple cardiac abnormalities at the same time. The distribution of the classes is as shown in the figure below.

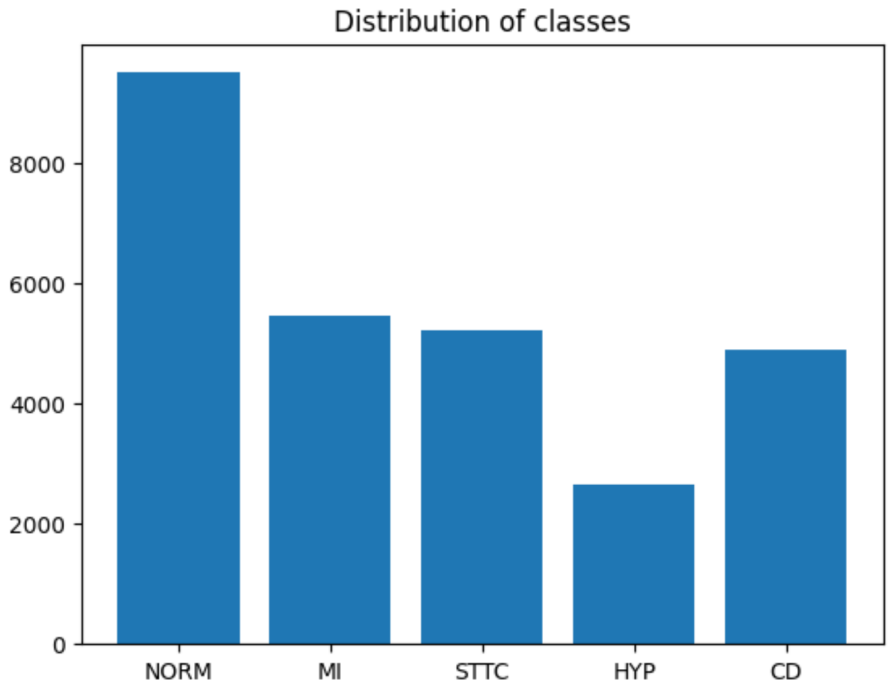


Figure 1: Distribution of labels in PTB-XL

Processing clinical notes

The clinical notes in the PTB-xl dataset are mostly in German (~60%). In order to perform classification on this dataset, we first needed to preprocess the clinical notes by detecting their language and then embedding them using appropriate models.

To detect the language of the clinical notes, we used the langdetect Python library. This library allows us to automatically detect the language of a given text by analyzing its character n-grams and comparing them to a set of pre-defined language profiles. We applied langdetect to each clinical note in the PTB-xl dataset and obtained the language label for each note.

After detecting the language of each clinical note, we then embedded them using two different pre-trained language models: emilyalsentzer/Bio_ClinialBERT for English notes and smanjil/German-MedBERT for non-English notes. Bio_ClinicalBERT is a pre-trained language model specifically designed for clinical text and has been shown to perform well on a range of clinical NLP tasks. German-MedBERT is a pre-trained language model specifically designed for German clinical text and has also been shown to perform well on a range of German NLP tasks.

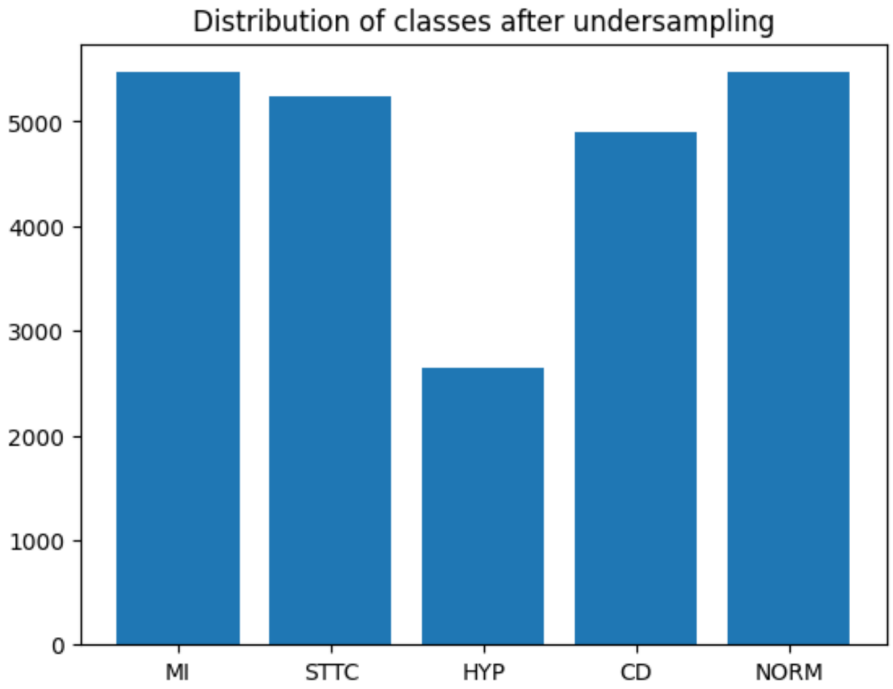
We used the Hugging Face Transformers library to load these pre-trained models and generate embeddings for each clinical note. Specifically, we tokenized each note using the appropriate tokenizer for its language and then passed the resulting tokens through the corresponding language model to obtain a dense embedding for each note. This process hopefully allows us to capture the semantic meaning of the clinical notes in a language-agnostic manner.

Undersampling to Balance Class Distribution

The PTB-XL dataset has an uneven distribution of classes, with a large number of ECG recordings labeled as "NORM" compared to the other classes. In order to balance the class distribution and prevent the model from being biased towards the majority class, we decided to use undersampling to reduce the number of "NORM" samples.

To perform undersampling, we used the pandas library in Python. We randomly selected a subset of 5469 samples from the original 9514 "NORM" samples, which is roughly the same number of samples in the other classes.

By performing this undersampling, there is the risk that the information loss from the majority class that is critical or unique to this class, but for the sake of creating a model that generalizes better to new, unseen data, we deemed it necessary to undersample in order to prevent the model from being biased towards this class. The figure below shows the distribution of classes after undersampling has been performed.



Data Augmentation to address class imbalance

In addition to the class imbalance in the "NORM" class, the PTB-XL dataset also has an imbalance in the "HYP" class. This class has significantly fewer samples than the other classes, which can make it difficult for the model to learn to accurately classify this class. To address this

issue, we decided to use data augmentation techniques to artificially increase the number of "HYP" samples in the dataset.

We used two types of data augmentation techniques: random noising and random masking. Random noising involves adding random noise to the ECG signal, which can help the model learn to be more robust to noise and variability in the input data. Random masking involves randomly zeroing out portions of the ECG signal, which can help the model learn to focus on the most informative parts of the signal.

By applying these data augmentation techniques to the "HYP" class, we were able to increase the number of samples from around half of the other classes to a more balanced number of around 5000 samples. The idea is to hopefully allow the model to learn more effectively from the "HYP" class, and ultimately improve the accuracy of the model on this class.

This method is not and should not be taken as a substitute for data collection, which as non-medical professionals we unfortunately are unable to do. However, data augmentation is known to be a useful technique for improving the performance of models when the data is limited or imbalanced.

Random Masking

Random masking is a specific type of data augmentation technique that involves randomly masking a portion of the input data with noise or zeros. In the case of ECG signals, random masking can be used to simulate missing or corrupted data points, which can occur due to various factors such as faulty electrodes, interference, or poor signal quality.

The random masking process involves selecting a random segment of the ECG signal and replacing it with a zeroed signal. A masking factor is used to determine the percentage of the original signal that will be set to zero.

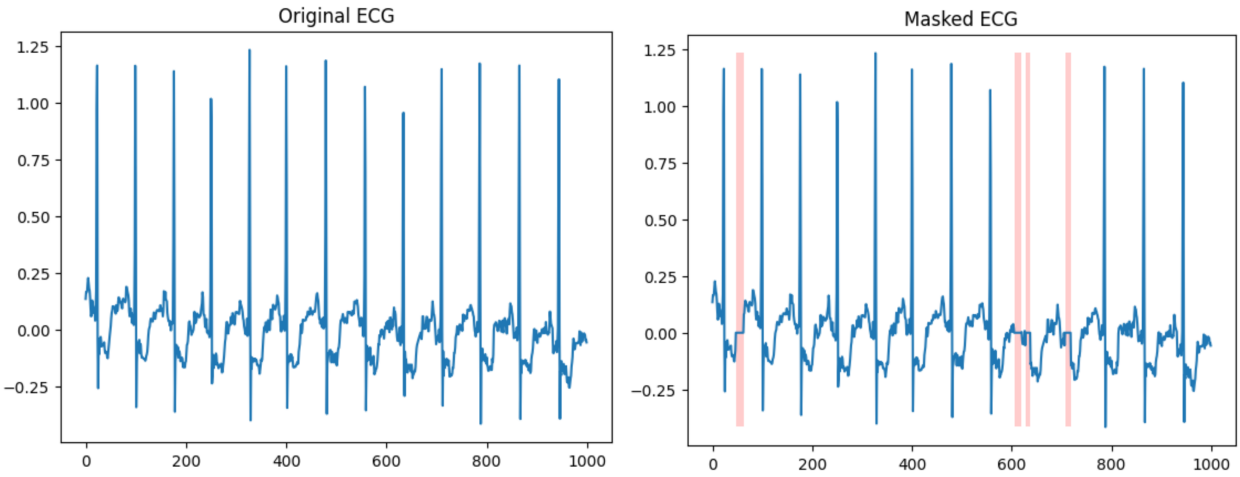
By applying random masking as a data augmentation technique, the model is forced to learn to identify and interpret ECG signals that are missing or corrupted, which can improve its ability to handle real-world scenarios where such issues may occur.

The main reasons for using this technique is to:

- Balance class imbalance
- Better generalizability towards missing or corrupted data points

Since the masked sample is representative of the original sample, the notes from the original sample are used for the masked sample during training. Examples of signals after augmentation are shown below. The masked segments of the signal are highlighted in red.

Examples



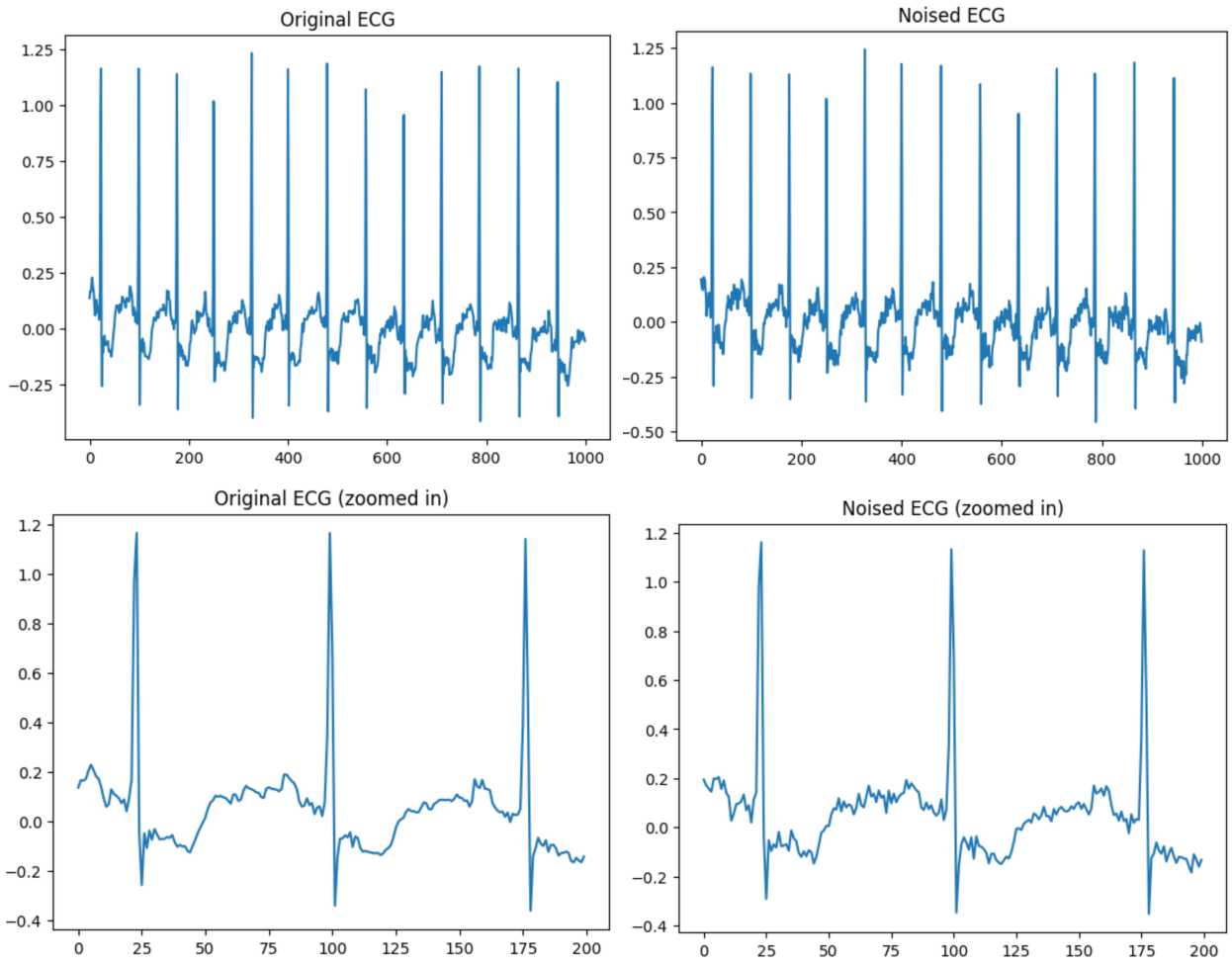
Random Noising

Random noising is a data augmentation technique used to artificially increase the variability in the ECG signal. This can help the model learn to be more robust to noise and variability in the input data. To apply random noising, we added random Gaussian noise to the ECG signal.

The process of adding random Gaussian noise involves generating a random number from a Gaussian distribution with a mean of zero and a standard deviation of the signal's standard deviation multiplied by a signal-to-noise ratio (SNR). We used an SNR of 0.1, meaning the standard deviation of the Gaussian distribution is computed by multiplying the signal's standard deviation by 0.1.

By introducing this additional variability into the data, the idea is to hopefully improve the generalization ability of the model, allowing it to more accurately classify ECG signals and clinical notes that it had not seen before. Examples of the signals after augmentation are shown below.

Examples

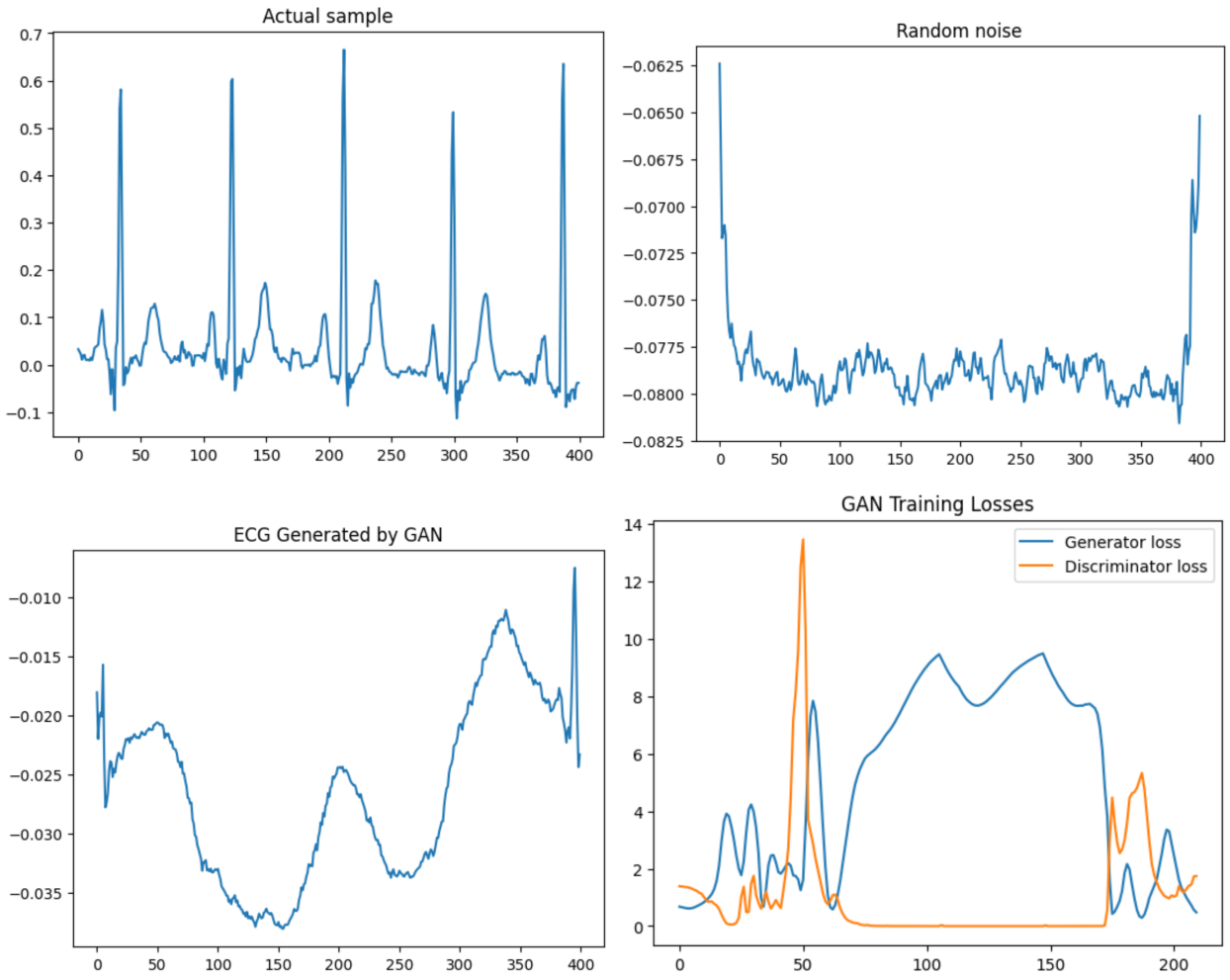


Generating ECG signals using GANs (Failed Attempt)

GANs have been successfully applied in various applications, including image and speech synthesis. In recent years, researchers have explored the use of GANs for generating synthetic ECG signals to augment training data for classification models. One such approach is to use a variational autoencoder (VAE) for extraction of features to synthesize new ECG signals (Kuznetsov et. al., 2020). There have also been a range of GAN architectures developed to generate synthetic ECGs (Delaney et. al., 2019).

In this project, we experimented with a generator that has a Bidirectional LSTM layer with 128 units, LeakyReLU activation on hidden convolutional layers and Tanh activation on output convolutional layer, and a discriminator that has LeakyReLU activation on hidden convolutional layers and a dense layer as output.

However, despite extensive training and hyperparameter tuning, the generated ECG signals failed to capture the complexity and variability of real ECG signals. The erratic training curves, although exhibit a slimmer of interleaved training, ultimately produced signals that looked unrealistic (see below) and were not suitable for augmenting the dataset. The figure below shows one of the training curves obtained in the experiments.



GANs have been notoriously difficult to work with and require a lot of data and tuning, which is why the training of such an architecture in this project has failed. Additionally, ECG signals are highly complex and dynamic, which makes it difficult for our GAN architecture to capture the variability in a particular class of cardiovascular irregularity with just a few thousand samples.

Therefore, we decided to fall back on traditional ECG augmentation techniques like random noise and masking instead of further attempting to generate synthetic signals using GANs.

Models used for classification

Training Approach

The task of classifying ECG signals into five different classes is complicated by the fact that patients may exhibit multiple conditions simultaneously. To account for this, the model should be able to predict multiple conditions accurately. The probability of each condition should be independent of the others, if a patient exhibits multiple conditions. To ensure a logical prediction, the output of every neural network must pass through a sigmoid function, which follows the rule of independent probability rather than softmax. The binary cross-entropy loss function with logits (BCEWithLogitsLoss) is compatible with multi-label classification. The labels are represented as vectors of the five classes, where multi-hot encoding represents multiple conditions. The binary cross-entropy loss function "rewards" the model for correctly producing either high probabilities for existing conditions or low probabilities for non-existing conditions.

In this project, accuracy is defined as the agreement between the model's highest probability output and the corresponding true label for each individual condition. If the model's highest probability output matches the true label for a given condition, then the prediction is considered accurate. This approach to evaluating accuracy is chosen for several reasons. Although the dataset has been under-sampled to ensure a balanced class distribution, there are still significantly more instances of singular conditions than of multiple conditions. Therefore, selecting the singular condition with the highest probability for accuracy is a sensible approach. Furthermore, if the model's strongest prediction is any element of the multi-class labels, then the model can be considered technically accurate. Lastly, the loss function used in this project is designed to direct the model towards predicting all corresponding classes correctly. In summary, accuracy and loss are two different representations, with accuracy being evaluated based on one-hot encoding and loss being evaluated based on multi-hot encoding.

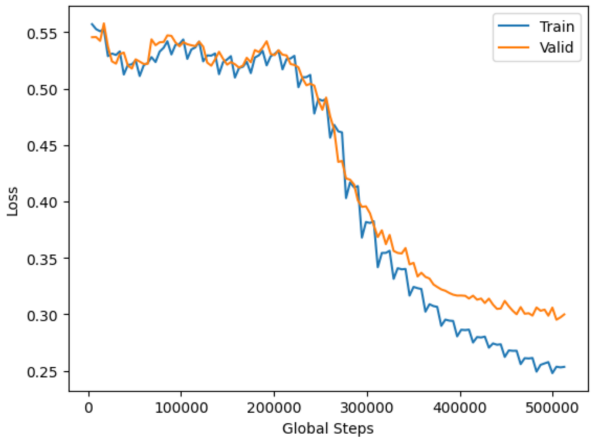
After training our models, we saved their checkpoints using torch.save(), along with the corresponding metrics. This allowed us to easily load and evaluate them at a later stage without having to retrain them. For each model, we saved the checkpoint with the best validation accuracy achieved during training. We also saved the training and validation losses for each epoch.

RNN-based Models

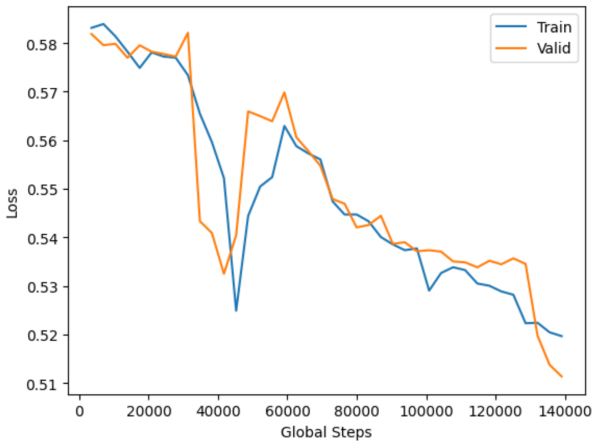
Model 1: Vanilla

Recurrent neural networks (RNN) have been widely used in time-series data analysis tasks, including electrocardiogram (ECG) signal classification. In this project, we experiment with an RNN-based model for classification with the PTB-XL dataset. Specifically, we use one long short-term memory (LSTM) layer, followed by a few linear layers with rectified linear unit (ReLU) as activation. The model parameters are initialized by Xavier normal. The results from training on the various datasets and training curves are as shown in the figures below.

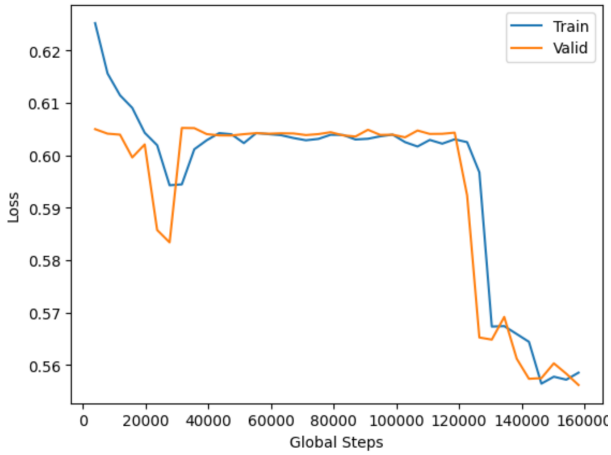
Results and loss visualizations



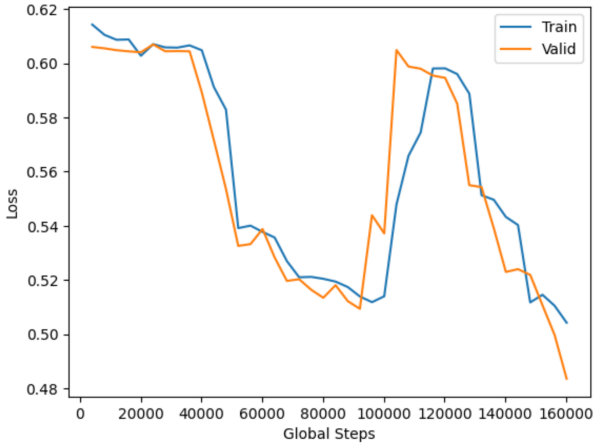
Loss from training on full dataset



Loss from training on undersampled



Losses from random masking augmentation



Losses from random noising augmentation

Accuracies for Vanilla RNN			
Without Undersampling	With undersampling	With augmentation	
		Random Masking	Random Noising
79.4%	48.2%	43.6%	56.4%

Discussion of results

The results above seem to imply that this vanilla RNN may not be sufficient to capture the complexities in the input signals. The accuracy of the model without undersampling was still decent at 79.4%, indicating that the model was able to learn some of the distinguishing features between the different heart conditions. However, there is a significant drop in accuracy to 48.2% when the data is undersampled to achieve class balance, suggesting that the model was struggling to generalize to the undersampled data.

This drop in accuracy is somewhat expected because the model is no longer able to see as many samples of each class, which leads to a decrease in its ability to distinguish between the classes, especially since the patterns behind each class is complicated by nature, as in the case of ECGs. This is an important consideration when working with imbalanced datasets because it highlights the need for techniques such as advanced data augmentation instead of undersampling.

The results from adding randomly masking samples as augmentation to the undersampled dataset show a slight further decrease in accuracy, but adding randomly noised samples actually improved the undersampling accuracy to 56.4%. Although the accuracies are no longer as high as the dataset without undersampling, which can be attributed to the smaller number of samples that the model has access to, it is promising to see that augmenting the dataset with randomly noised samples potentially helps the model learn the underlying patterns behind the data more acutely.

Overall, these results highlight the challenges of working with imbalanced ECG datasets and the importance of careful preprocessing and model selection. While the vanilla RNN was able to achieve reasonable accuracy on the full dataset, it struggled when faced with the imbalanced data. This suggests that more advanced techniques such as multimodal learning, as discussed later in the report, may be necessary to achieve better performance on this type of data. Additionally, more sophisticated data augmentation techniques may be needed to help the model learn more robust features and improve its ability to generalize to new samples.

Model 2: Multi-Modal RNN

In order to leverage both ECG signals and clinical notes for improved classification accuracy, we employed a multimodal RNN architecture. The model takes in two inputs, the ECG signals and the clinical notes, and processes them separately before combining the results for final classification.

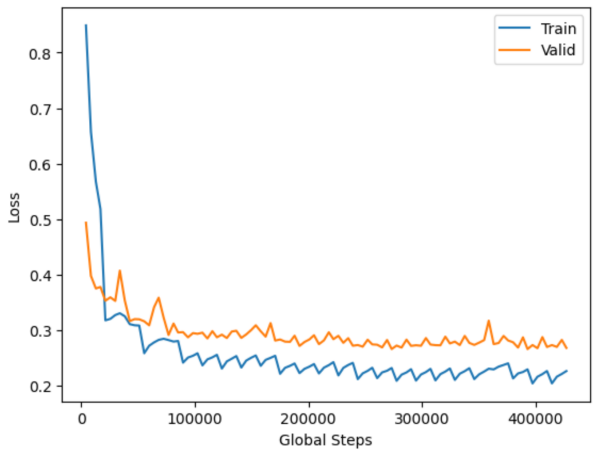
The ECG signals are passed through an LSTM layer with a hidden size of 64, followed by a linear layer with in_features=64 and out_features=768. The output of this linear layer is then passed through a LayerNorm layer to normalize the output.

The clinical notes are first passed through a Layernorm layer to normalize the input, and the outputs of the two pathways are then concatenated and passed through a final linear layer with five output units, corresponding to the five classes of the dataset.

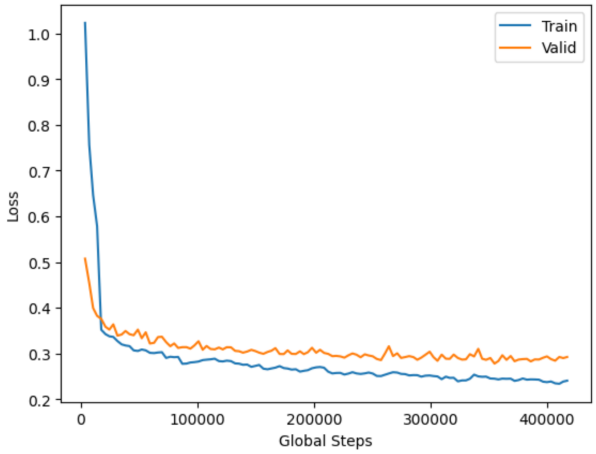
During training, we used binary cross entropy with logits as the loss function, and experiments were done with the AdamW and Adam optimizer with learning rates between 0.1 and 0.001. These models were trained between 20 to 30 epochs with batch sizes between 64 and 256. For the sake of brevity, the results displayed are the best out of all the experiments.

By incorporating both ECG signals and clinical notes, the multimodal RNN was able to achieve higher accuracy than the ECG-only RNN. This suggests that the clinical notes contain useful information for predicting cardiac abnormalities, and can complement the information provided by the ECG signals.

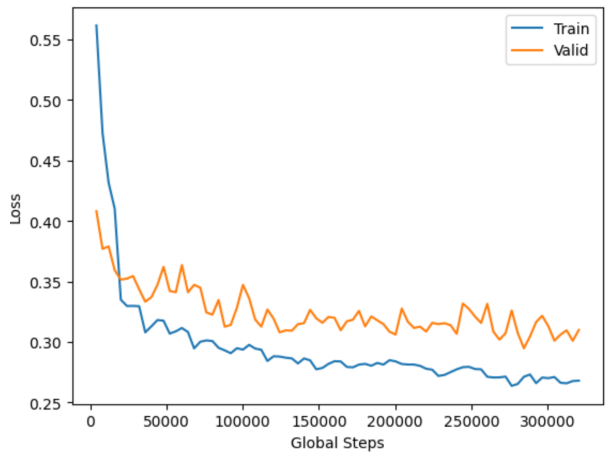
Results and loss visualizations



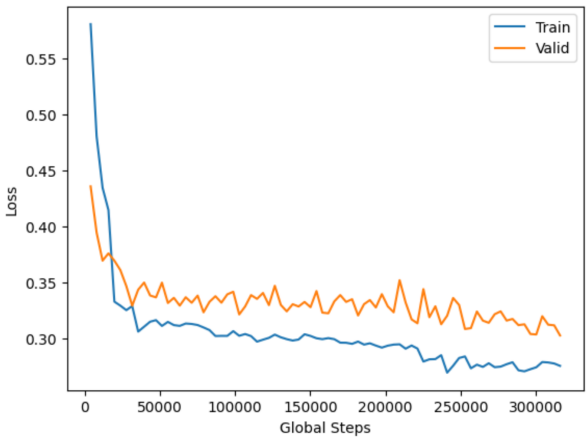
Loss from training on full dataset



Loss from training on undersampled dataset



Training with random noising augmented



Training with random masking augmented

Results for Multi-Modal RNN			
Without Undersampling	With undersampling	With augmentation	
		Random Masking	Random Noising
83.5%	83.1%	83.9%	84.2%

Discussion of results

The results obtained from the multimodal RNN with both ECG signals and clinical notes as inputs are encouraging. The model was able to achieve an accuracy of 83.5% on the full dataset, which is higher than the accuracy obtained from the vanilla RNN without multimodal input. This suggests that the addition of clinical notes as input features can provide complementary information to the ECG signals, improving the model's ability to classify the different cardiac conditions.

The drop in accuracy after undersampling has been applied can be attributed to the fact that when we undersample the "NORM" class, we remove a large portion of the data that the model could use to easily predict "NORM" as the output label. This means that the model has to learn to recognize and distinguish the other four classes more accurately in order to achieve high accuracy. The undersampled data presents a more challenging classification problem for the model, and it cannot rely on the abundance of "NORM" samples to obtain high accuracy, which is potentially why the accuracy of the model dropped.

However, it is interesting to see that the multimodal model's performance is not affected as much by the undersampling technique, as the accuracy only decreased slightly to 83.1%. This is perhaps a better representative of how the model performs than the higher accuracy it obtained by potentially overfitting to the data. In other words, the model has become less biased towards the "NORM" class and can now better predict the other four classes, which is the ultimate goal of this task.

The use of data augmentation techniques such as random masking and random noising did not significantly improve the model's performance, with accuracies of 83.9% and 84.2%, respectively. This may be because the model was already able to capture the important features in the ECG signals and clinical notes, even in the underrepresented class, without the need for additional data augmentation.

Overall, the results suggest that a multimodal approach is effective in improving the accuracy of classifying these ECG signals. Perhaps combining this approach with more sophisticated models will yield more promising results.

Reproducibility

The Jupyter Notebooks corresponding to the various experiments described above can be found on our Github repository, with the following names (*.ipynb):

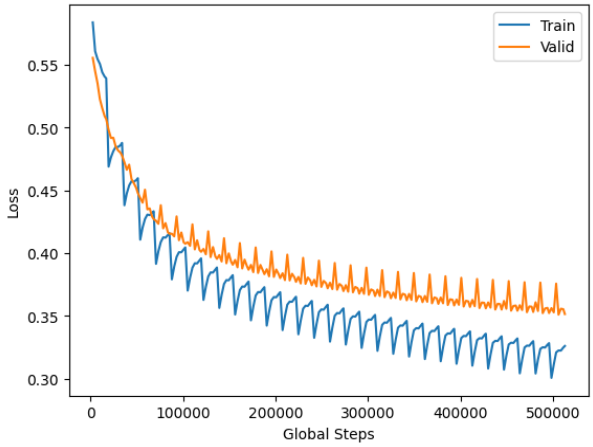
Model Name	Notebook names			
	Without Undersampling	With undersampling	With augmentation	
			Random Masking	Random Noising
MMRNN	MMLSTM	MMLSTM_undersample	MMLSTM_undersample_augmented	
Vanilla RNN	RNN_Base	RNN_Base_undersample	RNN_Base_undersample_augmented	

CNN-based Models

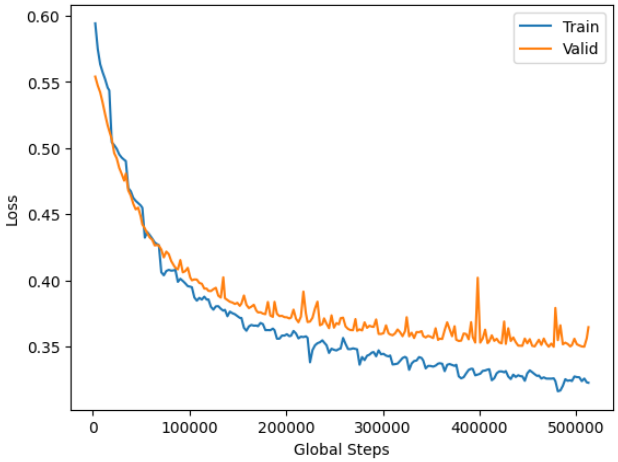
Model 1: Vanilla CNN

1D Convolution layers are a type of neural network layer that is commonly used for processing one-dimensional sequences of data, such as time series data. The main idea behind using 1D Convolution layers is that they can learn to extract local features from the input sequence, which can help in identifying patterns and trends that are relevant to the task at hand. 1D Convolution layers can be used to learn from time series data by extracting local features and learning complex relationships between the input and output. Since our ECG Signals are also a time series, we can use 1D Convolution Layers to extract features from the ECG Signals for classifying Cardiac abnormalities.

To create the Vanilla CNN Architecture used, we start with 3 Convolution blocks. Each block includes a 1D Convolution Layer, an Activation function (ReLU) and a Max pooling layer with kernel_size of 2 and stride of 2. The output of the 3 Convolution blocks are then flattened to form a 1D tensor of length 5856. Following this, a Fully Connected Linear Layer is added as a Classifier with 3 Linear Layers having output sizes of 512, 128 and 5 respectively. ReLU activation function is applied to each of the linear layers. The final output tensor is of length 5, corresponding to the number of classes. We can then apply a sigmoid function after the forward operation to get the predictions of whether the patient has shown signs of each of the 5 conditions.



Loss from training on unshuffled dataloader

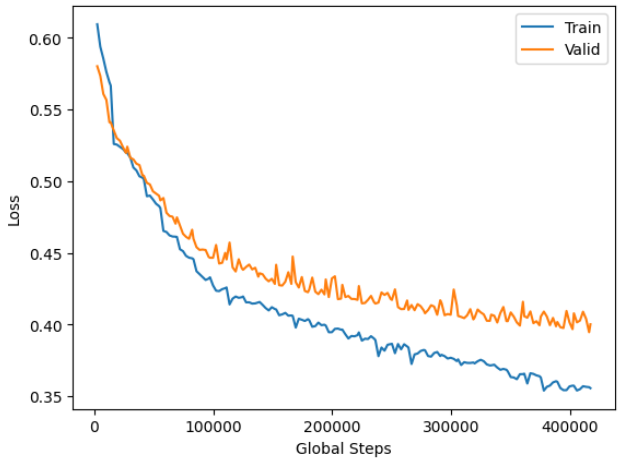


Loss from training on shuffled dataloader

The 2 graphs above show the training and validation loss for the Vanilla CNN architecture when trained on a shuffled and unshuffled dataloader. From the Loss curve of the model trained using an unshuffled dataloader, we can see that the loss value tends to have a periodic pattern which might be due to the model picking up on oscillations in the data when training. We found that this pattern was the most obvious in the Vanilla CNN model, which might be caused by the simplicity of the model. Although the pattern is also present in other complex models, the

models learn to ignore these oscillations over time and generalize well on the validation set with less periodic trend in the validation loss.

The vanilla CNN model managed to get a test accuracy of 71.27% on the unshuffled data and 71.46% on the shuffled data. These two graphs can be generated by running *CNN_Base.ipynb* and *CNN_Base_shuffled.ipynb* in the repository.



Loss from training on undersampled dataset (*CNN_Base_Shuffle.ipynb*)

Test accuracy for VanillaCNN	
Without Undersampling	With undersampling
71.46%	69.32%

From the graph and table above, we can see that the accuracy is worse on the undersampled dataset. This might be because the Vanilla CNN model might have learnt to abuse the class imbalance in the original dataset to achieve higher accuracy without actually learning about the differences between the classes. Although the undersampled accuracy is worse, there is less output class imbalance in the undersampled dataset. Thus, it is better able to recognise classes other than “NORM” so it would be more useful in correctly detecting cardiac abnormalities.

Model 2: Multi-Modal CNN

Similar to the Multi-Modal RNN, we use the Clinical notes along with the ECG Signals to improve classification accuracy by leveraging on additional information provided by the Clinical Notes. We utilized a few ways to combine the features extracted from the clinical notes and the ECG Signals to classify the sample. The 3 main methods we used are Summing up features, Concatenating features and using cross attention.

ECG Signal Feature Extraction

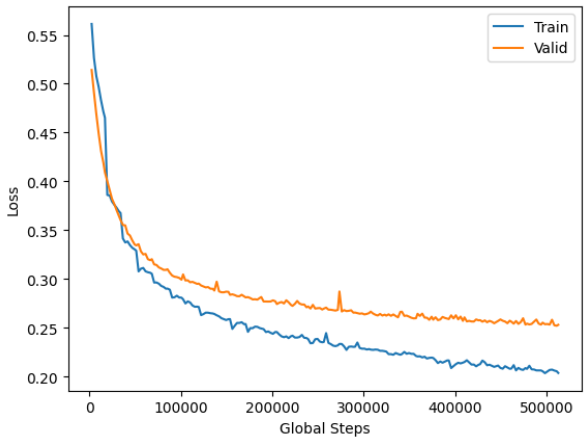
We use 3 convolution blocks similar to the Vanilla CNN Model to extract features from the ECG Signals. Then, we flatten the output of the final 1D CNN layer and apply 2 linear layers with output sizes 512 and 128 to get a final output of a 1D tensor of length 128 to represent features extracted from the ECG Signals.

Clinical Notes Feature Extraction

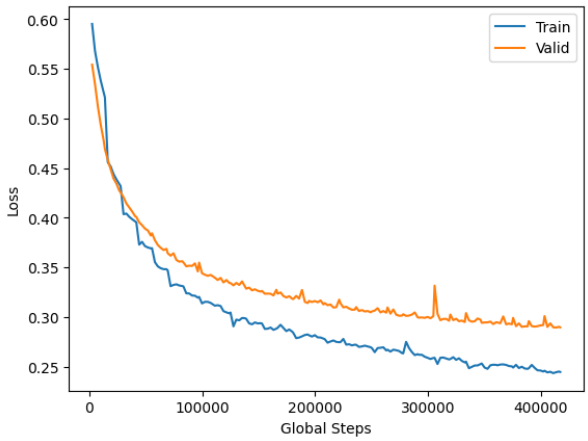
We first use the Clinical Bert model to generate embedding of size 768 to represent the clinical notes for each sample. Then, we use a single linear layer with output size 128 to get an output of 1D tensor of size 128 to represent features extracted from the clinical notes

Model 2.1: MMCNN_CAT

For the Concatenation Approach, we simply concatenate the features from ECG Signals and Clinical notes to generate a 1D tensor of size 256 and use a linear layer of output size 5 to generate the final outputs of the model. The results from training on the various datasets and training curves are as shown in the figures below.



Loss for Training on Original Dataset



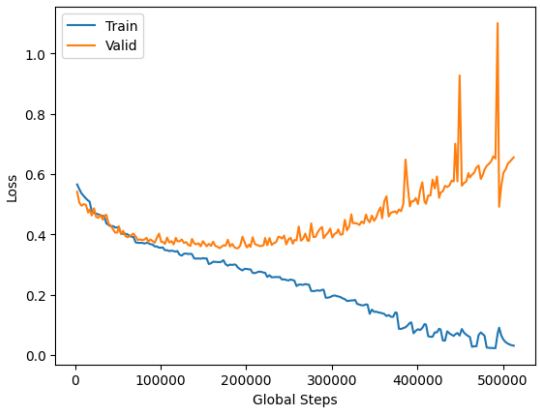
Loss for training on Undersampled Dataset

Test accuracy for MMCNN_CAT	
Without Undersampling	With undersampling
84.29%	81.94%

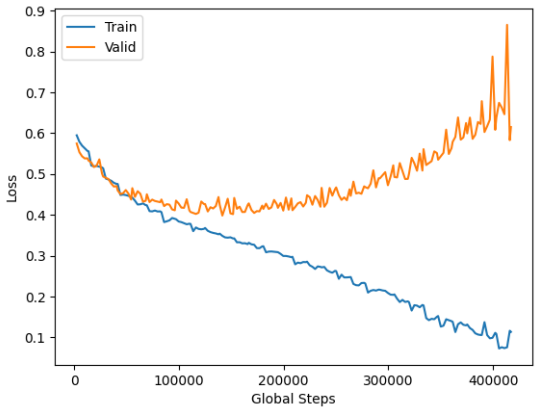
The above graphs and figures can be generated by running the notebook MMCNN_CAT.ipynb.

Model 2.2: MMCNN_ATT

For the Cross Attention Approach, we first applied a layernorm to the features from the ECG signals and the clinical notes. We then used a multi-head attention layer with 8 heads and passed the clinical Notes feature as the query input and the ECG Signal input as the key and value inputs. The resulting output is a 1D tensor of size 128 which we pass to a linear layer with output size 5 to classify the sample into the 5 classes. The results from training on the various datasets and training curves are as shown in the figures below.



Loss for Training on Original Dataset

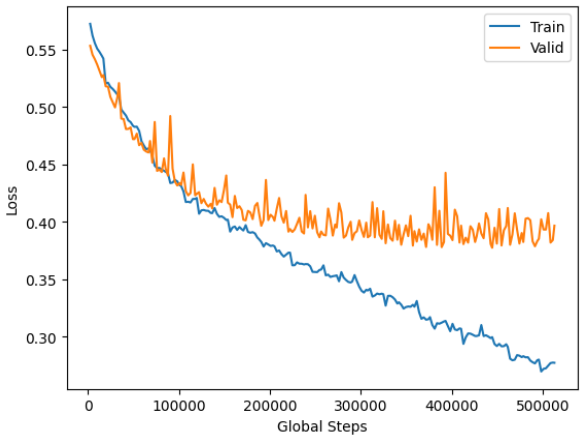


Loss for training on Undersampled Dataset

Test accuracy for MMCNN_ATT	
Without Undersampling	With undersampling
72.15%	67.74%

The above graphs and figures can be generated by running the notebook MMCNN_ATT.ipynb. We observe that the test accuracy for the undersampling dataset has dropped compared to the Vanilla CNN model. This might be because there is limited data for the attention layer to be helpful or it might be due to a high learning rate that led to overfitting.

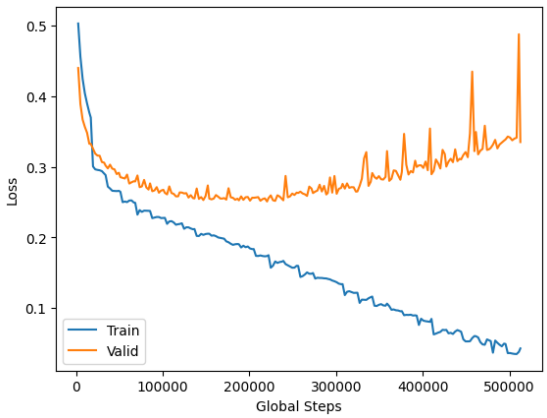
From the Graphs of training and Validation losses, we can observe that the model is overfitting greatly. Thus, we did hyperparameter tuning and set a smaller Learning Rate to try to prevent overfitting. Although the overfitting was reduced, the final accuracy on the test set dropped to 70.44% on the original dataset. The training and validation loss for the training on a lower learning rate is shown below.



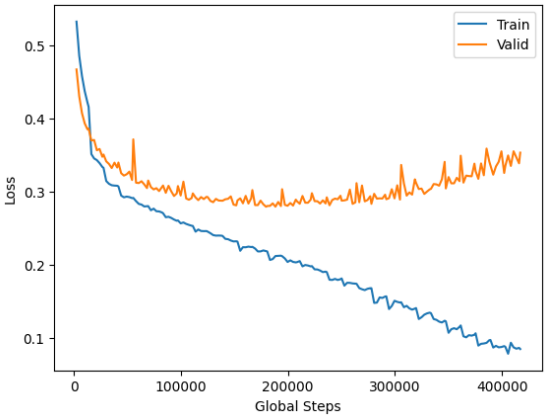
Loss for Training on Original Dataset with lower learning Rate

Model 2.3: MMCNN_SUM

For the Sum Approach, we simply add the 2 feature tensors to generate a 1D tensor of size 128. We then apply a Layer Normalisation to the vector and a Linear layer with output size 5 to classify the sample into the 5 classes. The results from training on the various datasets and training curves are as shown in the figures below.



Loss for Training on Original Dataset



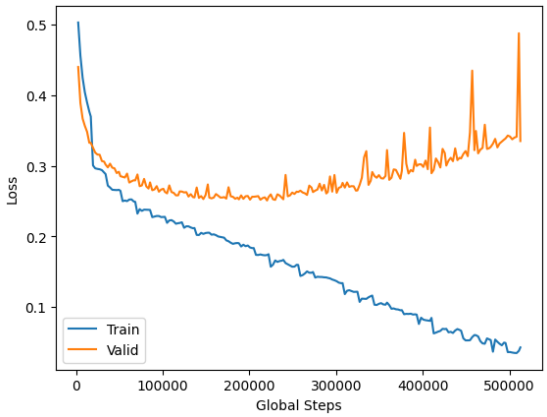
Loss for training on Undersampled Dataset

Test accuracy for MMCNN_SUM	
Without Undersampling	With undersampling
84.94%	83.75%

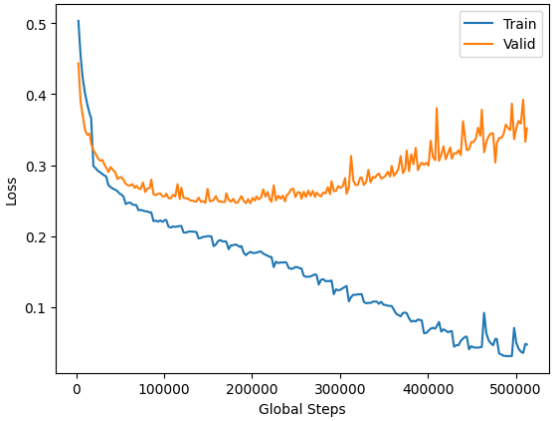
The above graphs and figures can be generated by running the notebook MMCNN_SUM.ipynb. As we can see from the graphs of MMCNN_SUM and MMCNN_ATT, the models tend to overfit to the training data and generalize worse on the validation and test set. Thus, we decided to add in some Dropout and BatchNorm layers to try to prevent overfitting and improve the classification accuracy.

Model 2.3.1: MMCNN_SUM_Dropout

For this modification, we added an additional dropout layer after each 1D convolution layer in the ECG Signal feature extractor. This dropout is used as a regularization technique. During training, a certain percentage of neurons are randomly selected and ignored, forcing the network to learn more robust features.The dropout layer prevents the network from relying too heavily on specific features and helps to prevent overfitting to the training data. This can also be helpful in improving the performance of the model on unseen data. The results from training on the Original dataset and training curves are as shown in the figure below.



Loss for Training without dropout



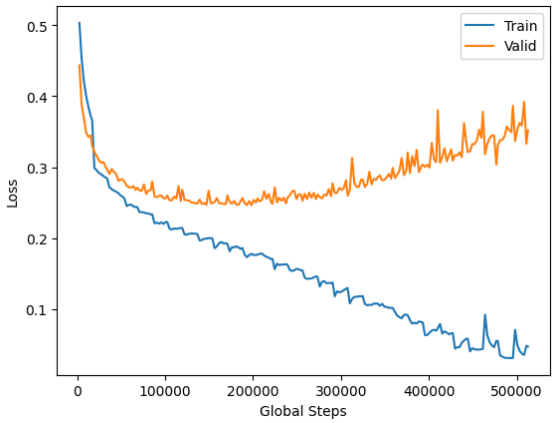
Loss for training with dropout

Test accuracy for MMCNN_SUM	
Without Dropout	With Dropout
84.94%	85.26%

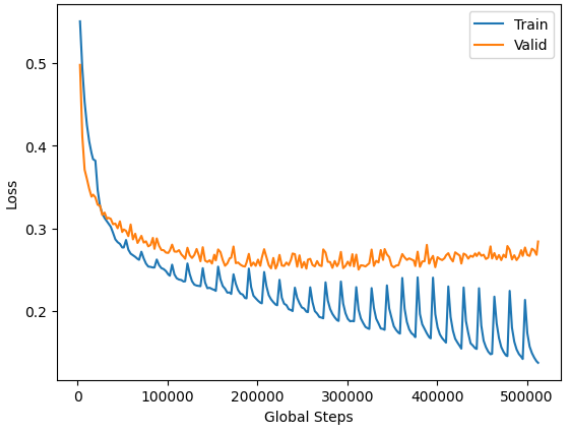
From the 2 training graphs, we can observe that the model with dropout tends to overfit less. However, there is still overfitting but there is slight improvement in the test accuracy of the model on the original dataset

Model 2.3.2: MMCNN_SUM_Dropout_Batchnorm

For this modification, we add a batchnorm layer between the dropout and 1d Convolution layer in the ECG signal feature extractor. Batch normalization is a technique used to improve the training of deep neural networks by reducing the internal covariate shift, leading to faster training and better performance on validation data. It also acts as a form of regularization.The results from training on the Original dataset and training curves are as shown in the figures below.



Loss for Training without BatchNorm



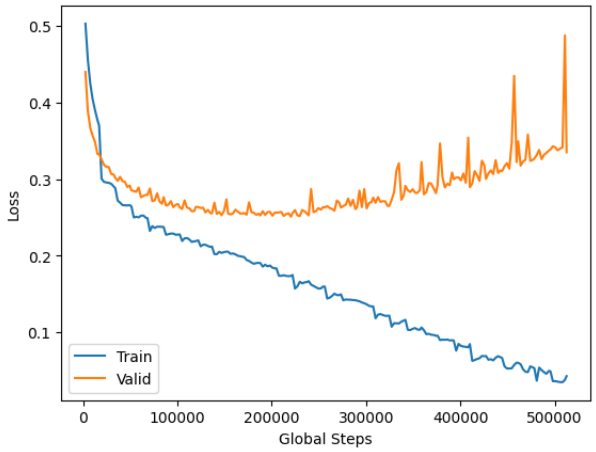
Loss for training with BatchNorm

Test accuracy for MMCNN_SUM	
Without BatchNorm	With BatchNorm
85.26%	84.85%

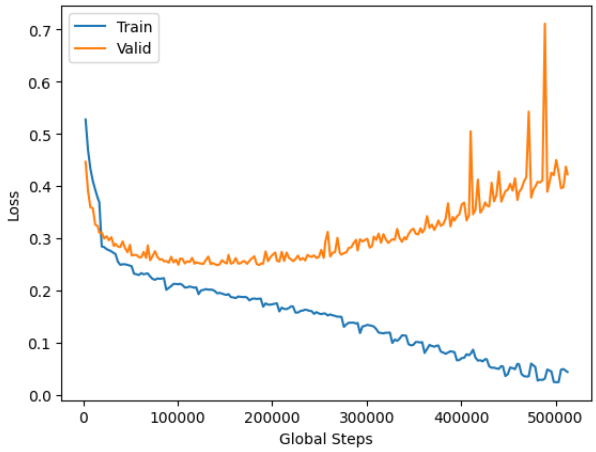
As we can see from the curves, the use of Batchnorm has reduced fluctuations in the amplitude fluctuations in the Validation loss and there is also less overfitting compared to using dropout only. The graphs from MMCNN_SUM_Dropout_Batchnorm and MMCNN_SUM_Dropout can be seen in MMCNN_SUM_dropout_batchnorm.ipynb and MMCNN_SUM_dropout.ipynb. However, before running the notebook, you need to adjust the Conv1D_layer class in models/CNN.py and comment out some layers depending on whether you want Batchnorm, Dropout or both for the experiment.

Model 2.3.2: MMCNN_SUM_ATT

Considering that the Sum approach worked the best, we wanted to test if adding a self attention layer after the Sum operation would be helpful in improving the performance of the model. TO implement this, we simply add a Multihead attention layer and pass the sum of the ECG and Clinical notes features as the Query, Key and value inputs into the attention layer and pass the output of the attention layer into a Linear classifier to get an output os size 5 to classify the sample into the 5 classes. The results from training on the Original dataset and training curves are as shown in the figure below



Loss for Training without Attention



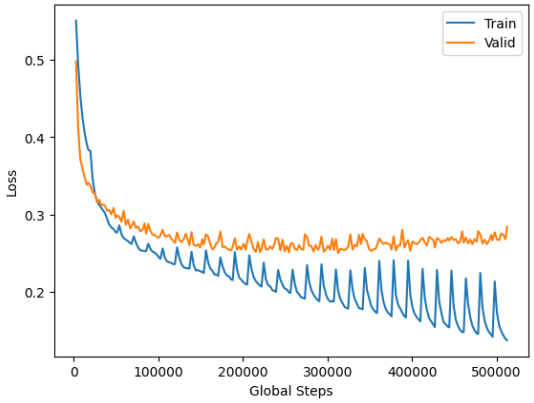
Loss for Training with Attention

Test accuracy for MMCNN_SUM	
Without Attention	With Attention
84.94%	83.87%

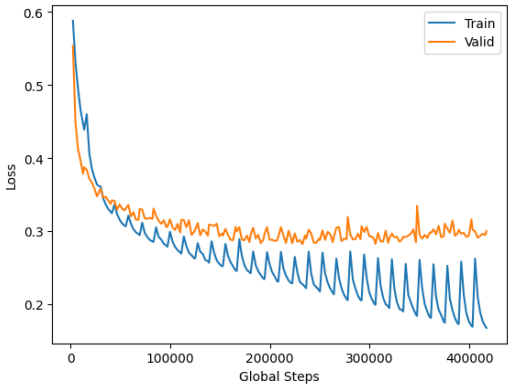
The above graphs and figures can be generated by running the notebook MMCNN_SUM_ATT.ipynb. From the graphs, we can see that this model with self attention suffers greatly from overfitting which may have led to the model generalizing poorly on the test set leading to worse test accuracy on the original dataset

Data Augmentation with CNN

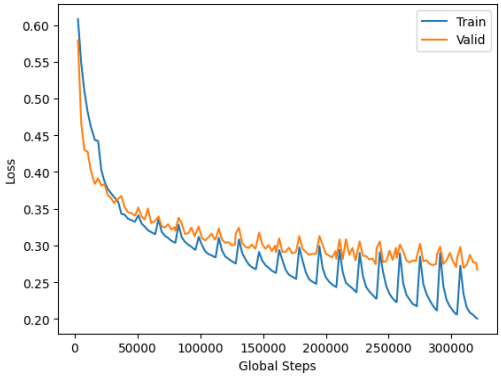
As mentioned earlier, we have come up with 2 ways of applying Data Augmentation, Random Masking and Random Noising, to artificially increase the number of “HYP” class samples to reduce the class imbalance. We have also tested the above CNN and Multimodal CNN models on the Augmented datasets. The use of Augmented datasets have led to slight improvements of test accuracies of most of the CNN models. The results from training on the augmented datasets can be found in the results discussion section below. For example, we will compare the performance of the MMCNN_SUM_Dropout_Batchnorm on the 4 different datasets.



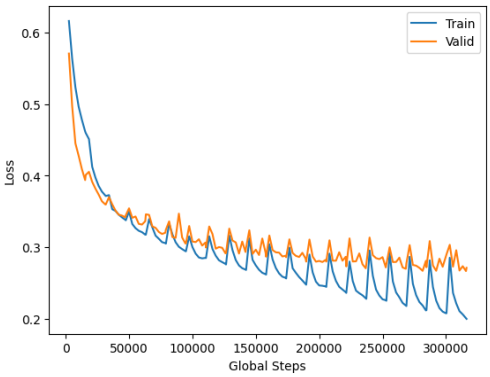
Loss for Training on Original Dataset



Loss for training on Undersampled Dataset



Loss for Training on Random Noise Dataset



Loss for training on Random Masking Dataset

As we can see from the graphs, the use of Augmented datasets has slightly reduced overfitting. The presence of more “HYP” class samples enables the model to better learn its characteristics which allows the model to better distinguish the 5 cardiac abnormality classes leading to better accuracy of classification.

Results for MMCNN_SUM_Dropout_Batchnorm			
Without Undersampling	With undersampling	Random Masking	Random Noising
84.85%	82.93%	85.06%	85.22%

CNN Reproducibility

The following table shows the notebook you can refer to in the repository to reproduce our results for a specific CNN model.

Model Name	Notebook name
Vanilla CNN	CNN_Base_Shuffle.ipynb
MMCNN_CAT	MMCNN_CAT.ipynb
MMCNN_ATT	MMCNN_ATT.ipynb
MMCNN_SUM	MMCNN_SUM.ipynb
MMCNN_SUM_Dropout	MMCNN_SUM_dropout.ipynb (modify Conv1d_layer class in models/CNN.py to set dropout)
MMCNN_SUM_Dropout_BatchNorm	MMCNN_SUM_dropout_batchnorm.ipynb (modify Conv1d_layer class in models/CNN.py to set dropout and batchnorm)
MMCNN_SUM_ATT	MMCNN_SUM_ATT.ipynb

For the performance of these models on the random mask and random noise Augmented datasets, refer to the notebooks CNN_Aug_random_mask.ipynb and CNN_Aug_random_noise.ipynb

Transformer-based Models

We attempted to train a transformer model based on the Paper “MVMTNET: A MULTI-VARIATE MULTI-MODAL TRANSFORMER FOR MULTI-CLASS CLASSIFICATION OF CARDIAC IRREGULARITIES USING ECG WAVEFORMS AND CLINICAL NOTES” which makes use of a transformer with the architecture shown below.

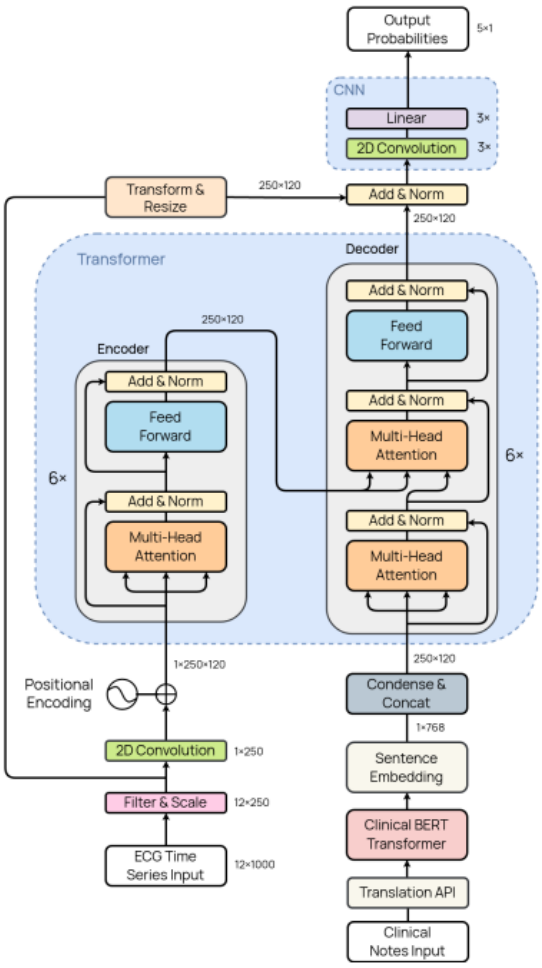


Figure 2: MVMTnet Transformer architecture.

Transformer Architecture based on Paper

Due to Computational limitations, we were not able to train the model for a sufficient number of epochs, as we could only use a batch size of 1 which caused the transformer model to be slow to train. We also could not find any available pretrained checkpoints for the model, so we were only able to train the model for very few epochs and the results obtained were not better than the CNN or RNN models. This might be caused by the small size of dataset available and the limited time we were able to train the transformer.

Result Discussion

Paper Results

Model	TS-CNN	TS-RNN	TST	M-CNN	M-RNN	MVMTnet
Test Accuracies (%)	67.34	62.77	59.26	70.39	67.81	76.03
Train Accuracies (%)	75.60	83.32	88.27	73.32	74.47	82.17
Validation Accuracies (%)	69.22	63.87	58.92	71.11	64.41	76.16
Validation Losses (%)	0.455	0.607	0.559	0.430	0.526	0.416

The above table shows the accuracies and losses for the various Vanilla and Multimodel CNN, RNN and Transformer models according to the paper. We have used the same accuracy calculation as the paper but found that our values of loss and accuracy tend to be much better compared to the results reported by the paper. This might be because of the different preprocessing approaches taken by us.

Firstly, we use the entire sequence of the ECG signal while the paper only made use of the first quarter of the signal sequence. The paper claimed that " Since the waveforms are highly periodic, the first quarter of the waveform can be used without significant loss of new information" (Samanta et al., 2023).This might have provided our models with greater information leading to better accuracy.

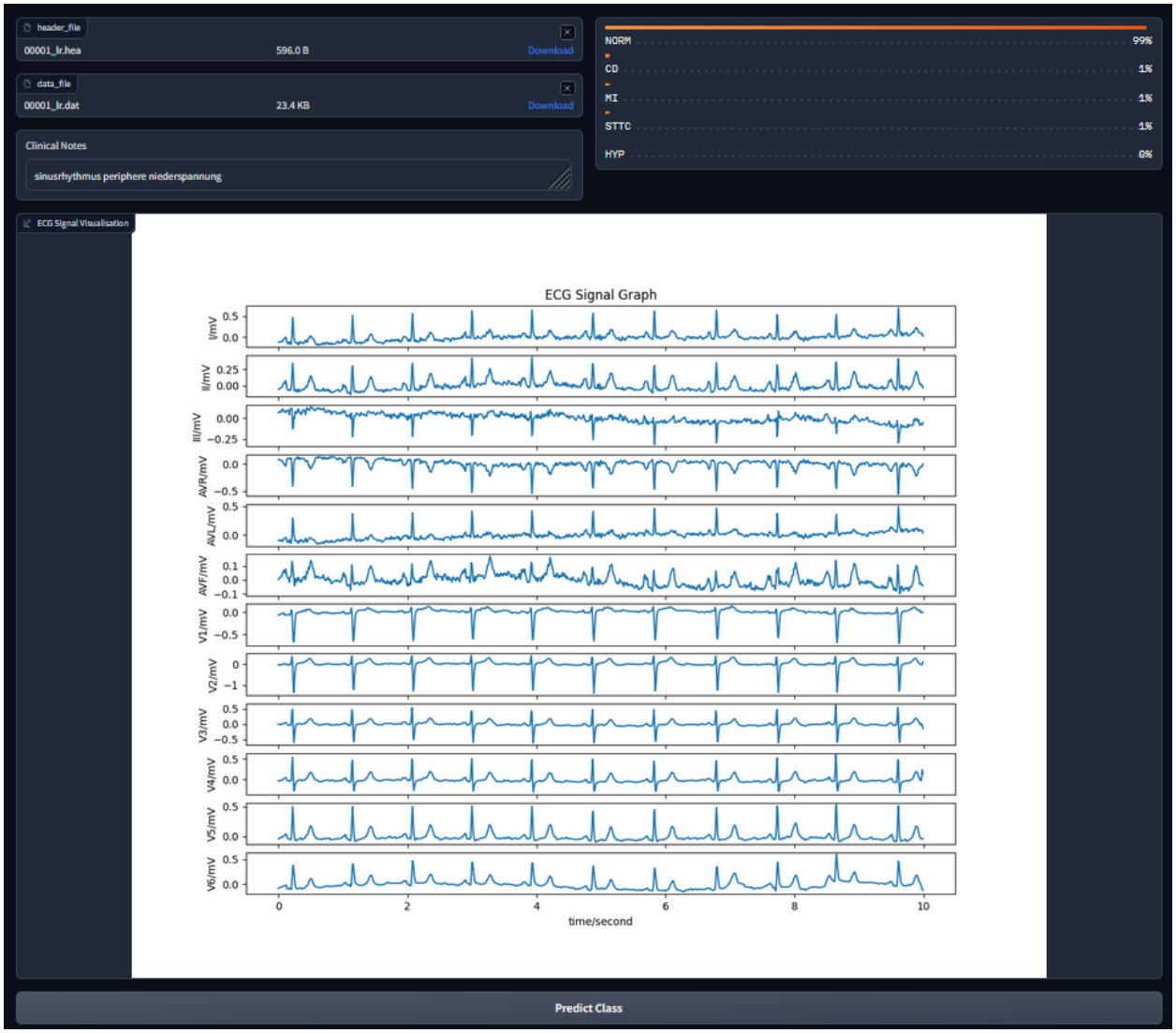
Secondly, we generated embeddings for the clinical notes differently compared to the paper. The paper used a google translate API to convert the notes to English before generating embeddings using a Clinical BERT, while in our approach we used a multilingual Clinical BERT model. Our approach may have generated embeddings that can be better classified.

Model Name	Without Undersampling	With undersampling	With augmentation	
			Random Masking	Random Noising
MMRNN	83.5%	83.1%	83.6%	82.7%
Vanilla RNN	79.4%	48.2%	43.6%	56.4%
Vanilla CNN	71.46%	69.32%	67.92%	68.20%
MMCNN_CAT	84.29%	81.94%	81.01%	81.42%
MMCNN_ATT	72.15%	67.74%	72.23%	73.67%
MMCNN_SUM	84.94%	83.75%	-	-
MMCNN_SUM_Dropout	85.26%	83.87%	-	-

Model Name	Without Undersampling	With undersampling	With augmentation	
			Random Masking	Random Noising
MMCNN_SUM_Dropout_BatchNorm	84.85%	82.93%	85.06%	85.22%
MMCNN_SUM_ATT	84.85%	83.87%	85.11%	84.91%

Demo

We used the gradio package to create a GUI to allow for users to generate inferences on ECG Signal data and Clinical notes. The figure below shows an overview of the interface.



On the top left of the page, we have 3 UI components to take in the Inputs. The inputs are the .hea and .dat file which contain the ECG signal data and a text box to input the clinical data. The outputs produced will be the class probabilities (Top right of the webpage) and a Visualization of the ECG Signal data(Bottom of the webpage).

Contributions

All group members contributed equally to the development of this project.

Github Repository and Google Drive Link

Github repository link: Here
Google drive containing model and metrics saves: Here

References

Kuznetsov, V. V., Moskalenko, V. A., & Zolotykh, N. Y. (2020). Electrocardiogram Generation and Feature Extraction Using a Variational Autoencoder. ArXiv [Eess.SP]. Retrieved from http://arxiv.org/abs/2002.00254

Delaney, A. M., Brophy, E., & Ward, T. E. (2019). Synthesis of Realistic ECG using Generative Adversarial Networks. ArXiv [Eess.SP]. Retrieved from http://arxiv.org/abs/1909.09150

Wagner, P., Strodthoff, N., Bousseljot, R., Samek, W., & Schaeffter, T. (2022). PTB-XL, a large publicly available electrocardiography dataset (version 1.0.3). *PhysioNet*. https://doi.org/10.13026/kfzx-aw45.

Samanta, A., Karlov, M., Ravikumar, M., Clarke, C. M., Rajadas, J., & Hassani, K. (2023). MVMTnet: A Multi-variate Multi-modal Transformer for Multi-class Classification of Cardiac Irregularities Using ECG Waveforms and Clinical Notes. ArXiv [Cs.LG]. Retrieved from http://arxiv.org/abs/2302.11021