Classification of arrhythmias using ECG

Author: Hermela Shimelis, PhD, MS-DS

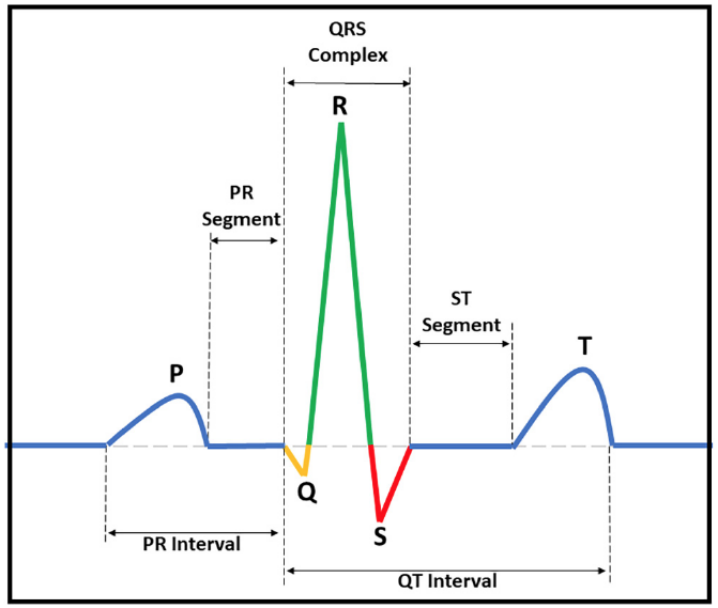
- The aim of this project is to classify arrhythmias into four categories: atrial fibrillation (AFIB), grouped supraventricular tachycardia (GSVT), sinus bradycardia (SB), and sinus rhythm (SR) using publicly available ECG data from 10,646 patients
- The ECG data was identified from the Chapman-Shaoxing ECG database (1)
- 6 deep learning models were investigated
  - 1D Convolutional neural network (1D CNN)
  - 1D CNN + LSTM
  - 1D CNN + GRU
  - ResNet + bidirectional LSTM
  - **InceptionTime**
  - InceptionTime tuned

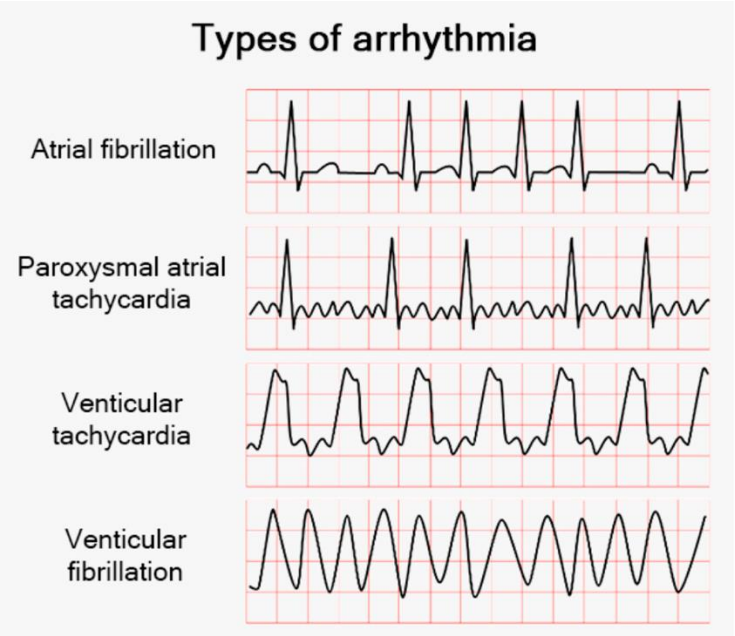
Introduction and brief literature review

- Arrhythmias are one of the most common and serious types of heart conditions, characterized by abnormal heart rhythms, including fast, slow, irregular, or uncoordinated heartbeats.

- Electrocardiogram (ECG) is a non-invasive diagnostic test that captures the heart’s electrical activity, providing information about the heart’s condition.

ECG





- ECG records the heart's electrical activity using 12 leads, which are placed on specific area of the body.
- These leads capture electrical activity from multiple dimensions
- Heart's electrical activity is recorded as waveforms, which consist of P waves, QRS complexes, and T waves
- Physicians analyze these waveforms to make a diagnosis. However, this manual approach has several limitations:
- The accuracy is highly dependent on the physician's experience, it is labor-intensive, and it is susceptible to noise and artifact
- The automation of diagnosis from ECG is beneficial

Arrhythmia classification

- Arrhythmia classification using ECG has been studied extensively
- Deep learning models (DL) have shown several advantages over traditional machine learning models for ECG classification.
  - DL models have automatic feature extraction capabilities
  - They are robust to noise
  - Selected DL models have shown better ability in capturing temporal features that are important for detecting arrhythmias
- ECG apps have in fact been successfully deployed.
- The Apple Watch can classify arrhythmias by monitoring electrical heart activity using digital crown and back crystal to record ECG.

1D CNN

- The 1D convolutional neural network was introduced for ECG classification due to its one-dimensional structure and efficiency in capturing long time series in a fast and accurate manner
- The list of recent deep learning models that have shown superior performance have been extensively reviewed by Ansari et. al.,
- Yildirim et al., employed a deep neural network model combining CNN and LSTM layers to classify arrhythmia using Chapman's ECG data
- The model achieved 96.13% overall accuracy

Comparison of DL models performance on arrhythmia classification from ECG reported in Strodthoff et al., (2)

Model	**Maximum F1 score on arrhythmia classification**
LSTM	0.908
Bidirectional LSTM	0.907
Fully convolutional network	0.899
ReSnet 1D	0.908
InceptionTime	0.917

Authors employed different neural network models for arrythmia classification, providing benchmarks and insights from PTB-XL 12 lead ECG from 18,885 patients.

Methods

Dataset

- Publicly available12-lead ECG of 10,646 individuals was recorded at a 500 Hz sampling frequency for 10 seconds from Chapman University and Shaoxing People's Hospital
- ECG data has been preprocessed, including denoising, and all extracted features are available as a CSV file.
- 10K CSV files with 5,000 observations and 12 variables, 1 from each lead
- Labels are provided in a separate file

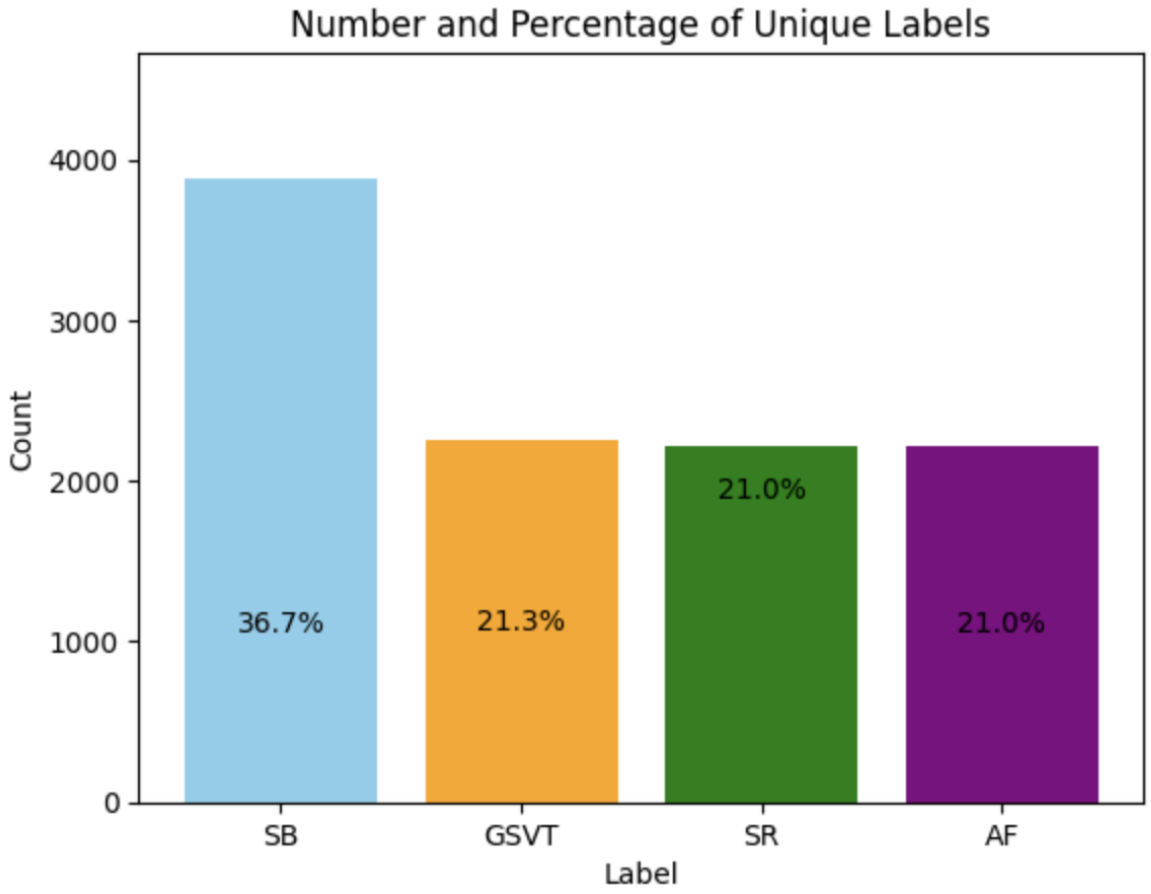
Acronym	Condition name
SB	Sinus Bradycardia
SR	Sinus Rhythm
AFIB	Atrial Fibrillation
ST	Sinus Tachycardia
AF	Atrial Flutter
SI	Sinus Irregularity
SVT	Supraventricular Tachycardia
AT	Atrial Tachycardia
AVNRT	Atrioventricular Node Atrioventricular Node Tachycardia
AVRT	Atrioventricular Reentrant Tachycardia
SAAWR	Sinus Atrium to Atrial Wandering Rhythm

Merged from	4 classes of arrythmias	Count
AF, AFIB	AFIB	2,225
SVT, AT, SAAWR, ST, AVNRT, AVRT	GSVT	2,307
SB	SB	3,889
SR, SI	SR	2,225
All	All	10,646

- Sinus rhythm represent normal ECG in this study

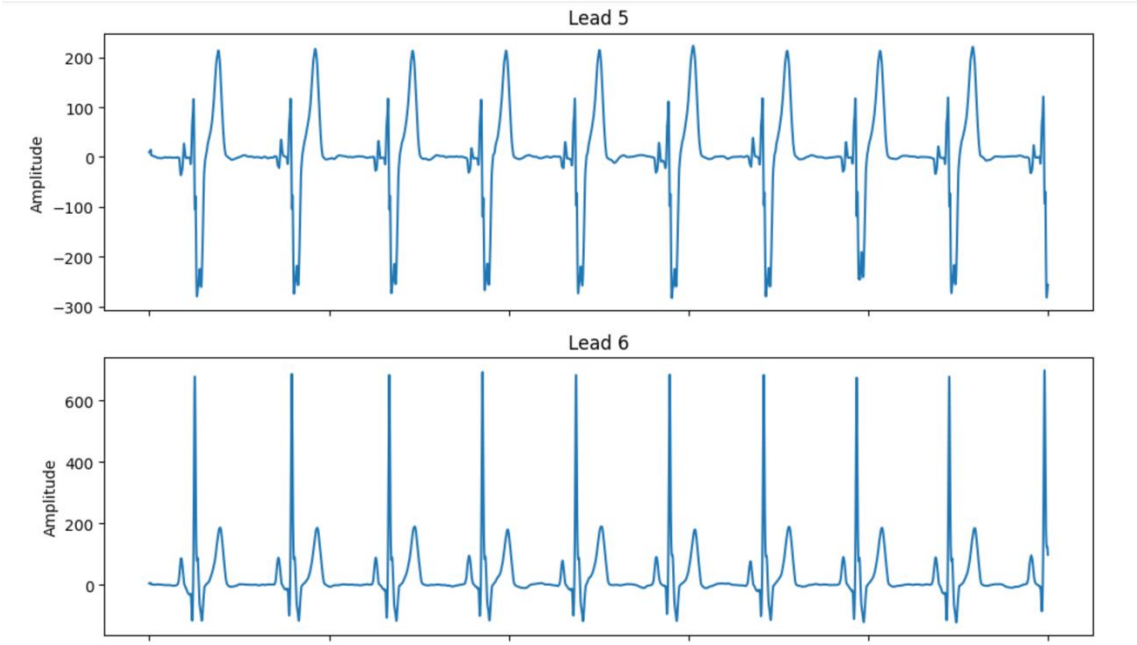
https://figshare.com/collections/ChapmanECG/4560497/2

Class labels



Visualization: recording from 2 leads

A	B	C	D	E	F	G	H	I	J	K	L
947.52	186.11	-767.34	-571.43	862.51	-291.83	-523.92	-942.24	-42.368	2228.3	2513.5	1817.2
865.5	231	-640.37	-552.06	758.38	-205.61	-563.39	-1087.1	-405.48	1633.1	2070.2	1636.3
779.63	274.49	-510.95	-530.22	650.99	-118.9	-605.29	-1232.8	-755.98	1058	1647.5	1459.8
688.51	315.43	-378.83	-504.73	539.4	-32.144	-649.59	-1372.7	-1072.8	529.86	1266.1	1292.4
594.07	352.87	-246.88	-476.1	426.02	52.73	-692.79	-1492.1	-1327.7	81.365	943.85	1139.4
500.13	385.28	-120.47	-445.41	315.49	132.29	-728.23	-1571.4	-1492.9	-258.22	690.88	1003.4
410.73	410.23	-6.054	-413.37	213.17	202.08	-748.26	-1593.2	-1552.3	-473.87	506.4	882.96
329.33	424.9	90.074	-380.17	124.06	257.55	-747.03	-1549.6	-1508	-570.59	380.02	774.08
258.65	427.12	163.04	-345.99	52.015	295.2	-722.49	-1444.5	-1379.6	-571.15	296.72	673.11
200.35	416.22	210.5	-311.29	-0.92	313.53	-676.85	-1293.4	-1197.5	-507.88	241.96	578.76
154.73	393.45	233.42	-276.86	-35.108	313.63	-615.38	-1117.1	-993.82	-413.2	204.76	492.22
124.47	368.62	238.81	-233.14	-50.943	305.79	-556.92	-949.72	-792.38	-314.89	180.81	412.21
100.06	328.51	224.44	-202.15	-56.259	278.65	-480.95	-778.29	-612.16	-221.86	160.31	345.95
82.02	287.36	204.95	-175.79	-54.281	248.37	-407.39	-629.21	-466.5	-151.93	143.47	291.22
68.094	248.39	184.4	-153.2	-49.065	218.64	-341.21	-507.05	-354.53	-104.04	127.56	245.4
56.976	213.54	163.73	-133.62	-44.063	190.73	-284.81	-411.52	-273.15	-72.92	113.71	207.96



Data preprocessing

- 56 samples containing Nan values are removed
- The indices of loaded file name and labels are properly aligned to ensure that the samples correspond accurately. Each ECG file name serves as the sample name.
- Samples were split into train, validation and test with 0.8, 0.1, 0.1 ratio, respectively.
- The data is standardized with standard scaler and labels are encoded with label encoder.

Approach

- Arrhythmia classification from ECG involved analysis of a very long time series, with 5,000 timesteps (500 samples per second taken for 10 sec). The sequential nature of cardiac rhythm made It necessary to model this classification problem as a time series problem.

- 6 deep learning models were investigated
  - 1D Convolutional neural network (1D CNN)
  - 1D CNN + LSTM
  - 1D CNN + GRU
  - ResNet + bidirectional LSTM
  - InceptionTime
  - InceptionTime tuned

Compiling and training models

```
# Define the optimizer with a lower learning rate and gradient clipping
optimizer = tf.keras.optimizers.Adam(learning_rate=0.0001, clipvalue=1.0)

# Compile the model with the modified optimizer
model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])

# Callback to stop training when a monitored quantity has stopped improving
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

# Callback to reduce the learning rate when a metric has stopped improving
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=5, min_lr=0.00001)

# Train the model with the callbacks
history = model.fit(X_train, y_train_encoded, epochs=200, batch_size=64, validation_data=(X_val, y_val_encoded), callbacks=[early_stopping,
#history = model.fit(X_train, y_train_encoded, epochs=100, batch_size=64, validation_split=0.2, callbacks=[early_stopping, reduce_lr])

# Extract the loss and validation loss from the history
train_loss = history.history['loss']
val_loss = history.history['val_loss']

# Plot the training and validation loss
plt.figure(figsize=(10, 5))
plt.plot(train_loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.title('Training and Validation Loss Over Epochs')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.grid()
plt.show()
```

- All models are compiled and trained using the same method to enable model comparison only with changing models.
- All models are compiled with the learning rate = 0.0001, Adam optimizer
- A smaller learning rate allows the model to converge slowly minimizing the risk of missing the global minimum
- clipvalue=1.0 prevents the gradients from becoming too large
- Categorical Crossentropy Loss, Ideal for multi-class classification
- Early stopping setting patience=10, training will stop if no improvement is seen for 10 epochs
- ReduceLROnPlateau adjusts learning rate if no improvement in validation loss.
- Epoch 200 and batch_size of 64 was used

Model evaluation metric

- Identifying ECGs with abnormal rhythms is one of the most crucial aspects of the classification problem, as a missed diagnosis can be detrimental to the patient.
- Therefore, **macro average recall was used for model comparison**.
- The class-wise precision-recall curve will also be employed to assess model performance for each class.
- The area under the precision-recall curve (AUC-PR) is included to obtain a single value that represents how well the model identifies the positive class while minimizing false positives for each label, and this is averaged across all classes.

Results

Model 1: 1D CNN model

```
model = Sequential([
    Conv1D(filters=64, kernel_size=21, strides=11, input_shape=(time_steps, 12)),
    MaxPooling1D(pool_size=2),
    BatchNormalization(),
    LeakyReLU(alpha=0.1),
    Dropout(rate=0.3),
    Conv1D(filters=64, kernel_size=7, strides=1),
    MaxPooling1D(pool_size=2),
    BatchNormalization(),
    Conv1D(filters=128, kernel_size=5, strides=1),
    MaxPooling1D(pool_size=2),
    Conv1D(filters=256, kernel_size=13, strides=1),
    Conv1D(filters=512, kernel_size=7, strides=1),
    Dropout(rate=0.3),
    Conv1D(filters=256, kernel_size=9, strides=1),
    MaxPooling1D(pool_size=2),
    GlobalAveragePooling1D(),
    Dense(units=64, activation='relu'),
    Dense(units=4, activation='softmax')
])
```

- The model begins with a series of 1D CNN layers to capture various features and patterns within the input data.
- The first layer has 64 filters, a large kernel size of 21, and a stride of 11, which helps reduce the input size early on.
- It includes six 1D convolutional layers, MaxPooling layers to reduced dimensionality, and batch normalization layers to stabilize and speed up training.
- Subsequent convolutional layers progressively refine the features, increasing the number of filters up to 512 in the network's deeper layers.
- The output from convolution layers is passed to fully connected dense layer with relu activation and softmax classifier with number of neurons equal to the number of classes.
- This model structure is similar to one reported in Yildirim et al., 2020, but it does not include the LSTM layer. This was done to compare the effect of adding LSTM layer.

Model 1: 1D CNN model performance

	precision	recall	f1-score	support
AF	0.96	0.80	0.87	222
GSVT	0.89	0.96	0.92	226
SB	0.96	0.98	0.97	389
SR	0.91	0.95	0.93	222
accuracy			0.93	1059
macro avg	0.93	0.92	0.92	1059
weighted avg	0.93	0.93	0.93	1059

- The model achieved good performance with macro average recall of 0.92
- This model showed lower performance on classifying AF, as some samples were being classified as GSVT.
- The following two models will include LSTM and GRU layers to test whether adding these layers on top of the deep CNN models improves performance

Model 2: CNN + LSTM model

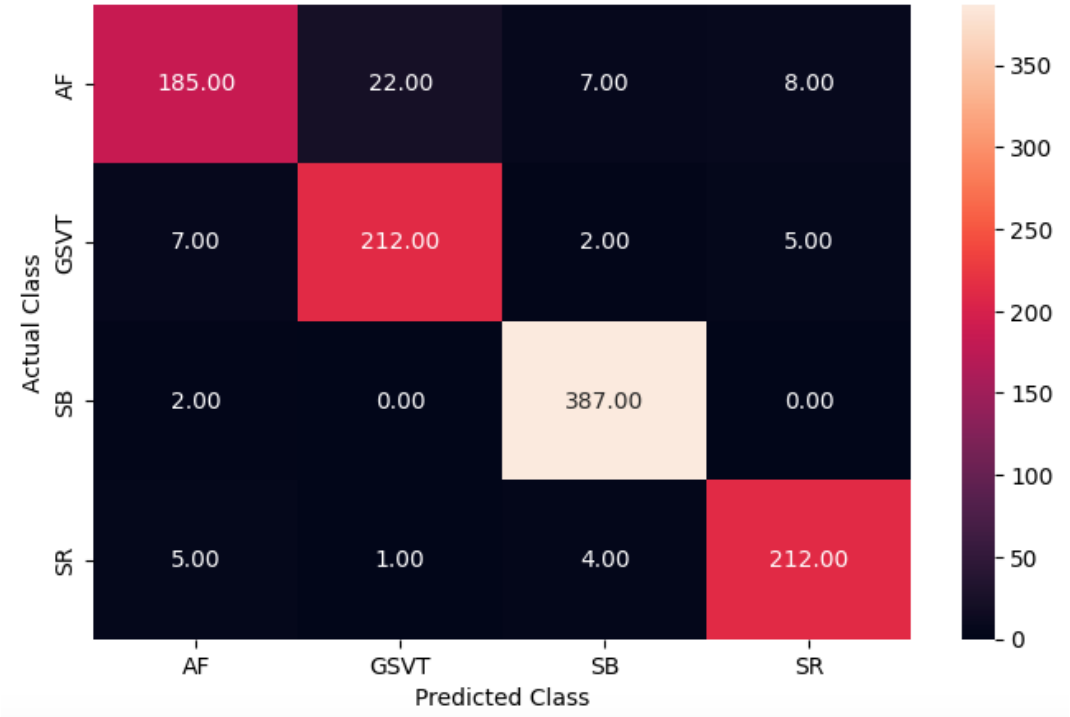
```
# CNN + LSTM model
# number of features = 12

model = Sequential([
    Conv1D(filters=64, kernel_size=21, strides=11, input_shape=(time_steps, 12)),
    MaxPooling1D(pool_size=2),
    BatchNormalization(),
    LeakyReLU(alpha=0.1),
    Dropout(rate=0.3),
    Conv1D(filters=64, kernel_size=7, strides=1),
    MaxPooling1D(pool_size=2),
    BatchNormalization(),
    Conv1D(filters=128, kernel_size=5, strides=1),
    MaxPooling1D(pool_size=2),
    Conv1D(filters=256, kernel_size=13, strides=1),
    Conv1D(filters=512, kernel_size=7, strides=1),
    Dropout(rate=0.3),
    Conv1D(filters=256, kernel_size=9, strides=1),
    MaxPooling1D(pool_size=2),
    LSTM(units=128, return_sequences=False),
    #Bidirectional(LSTM(128, return_sequences=False)), # LSTM showed better performance than biLSTM
    Dense(units=64, activation='relu'),
    Dense(units=4, activation='softmax')
])
```

- This model structure is similar to one reported in Yildirim et al., 2020, include the LSTM layer.

- LSTM is known for its ability to capture longer-range dependencies in data due to its internal mechanisms composed of forget, input, and output gates.

Model 2: CNN + LSTM model performance

	precision	recall	f1-score	support
AF	0.93	0.83	0.88	222
GSVT	0.90	0.94	0.92	226
SB	0.97	0.99	0.98	389
SR	0.94	0.95	0.95	222
accuracy			0.94	1059
macro avg	0.94	0.93	0.93	1059
weighted avg	0.94	0.94	0.94	1059



- The model achieved good performance with macro average recall of 0.93
- Including the LSTM layer improved the overall model performance, and on classifying AF.
- Using bidirectional LSTM instead of LSTM layer showed slight reduction in performance.

Model 3: CNN + GRU model

```
model = Sequential([
    Conv1D(filters=64, kernel_size=21, strides=11, input_shape=(time_steps, 12)),
    MaxPooling1D(pool_size=2),
    BatchNormalization(),
    LeakyReLU(alpha=0.1),
    Dropout(rate=0.3),
    Conv1D(filters=64, kernel_size=7, strides=1),
    MaxPooling1D(pool_size=2),
    BatchNormalization(),
    Conv1D(filters=128, kernel_size=5, strides=1),
    MaxPooling1D(pool_size=2),
    Conv1D(filters=256, kernel_size=13, strides=1),
    Conv1D(filters=512, kernel_size=7, strides=1),
    Dropout(rate=0.3),
    Conv1D(filters=256, kernel_size=9, strides=1),
    MaxPooling1D(pool_size=2),
    GRU(units=128, return_sequences=False),
    Dense(units=64, activation='relu'),
    Dense(units=4, activation='softmax')
])
```

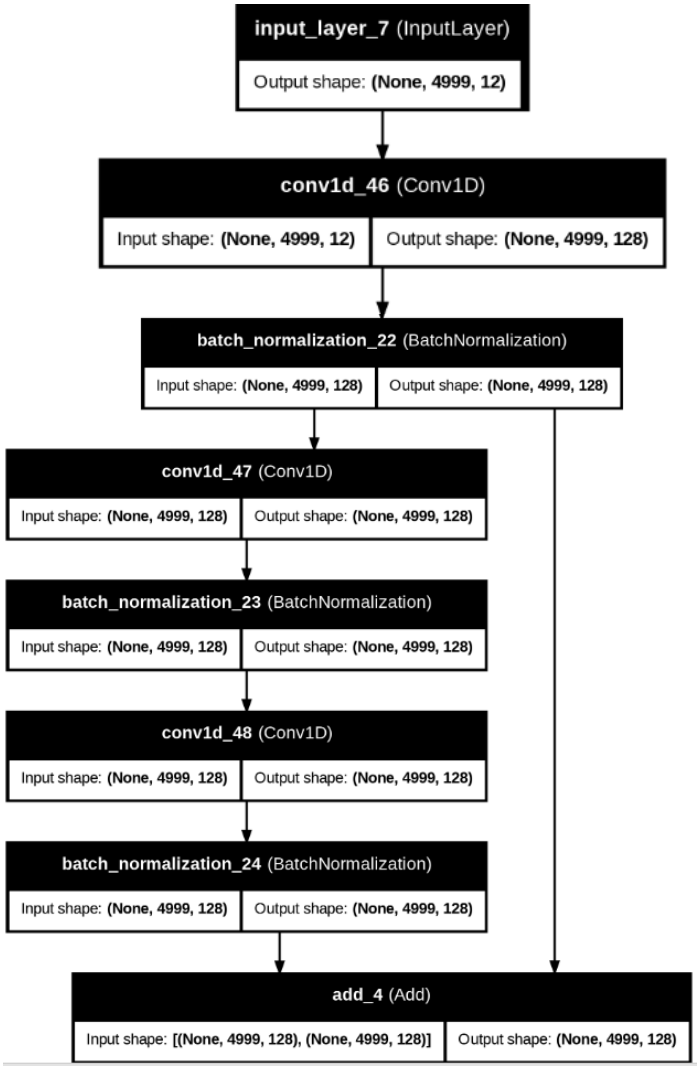
- This model structure is similar to one reported in Yildirim et al., 2020, but instead of LSTM, it uses GRU layer
- Both LSTM and GRU are known for their ability to capture longer-range dependencies, but they might not be as effective as LSTMs when dealing with very complex sequence relationships.
- The LSTM model has more trainable parameters (model 2 total trainable parameters = over 8 million) with three gates as opposed to two in GRUs (model 3 total trainable parameters = 1.4 million)

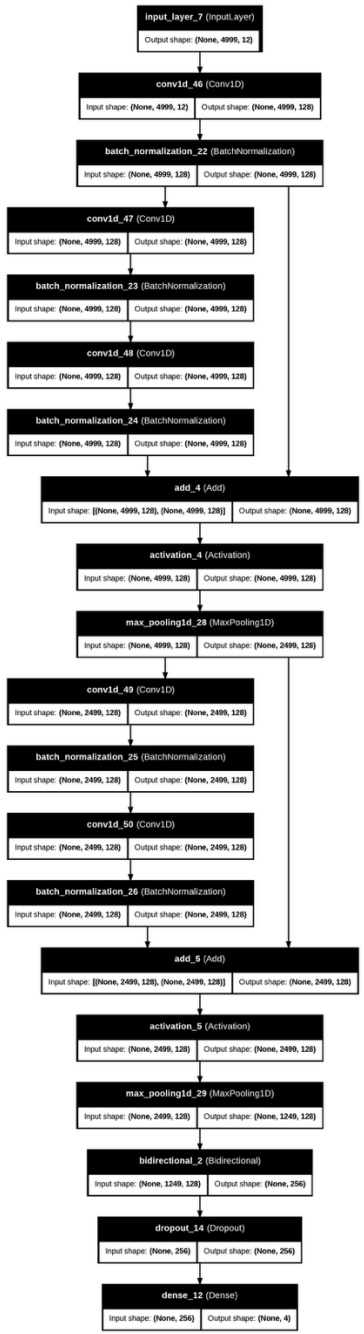
Model 3: CNN + GRU model performance

	precision	recall	f1-score	support
AF	0.90	0.80	0.85	222
GSVT	0.89	0.92	0.90	226
SB	0.96	0.98	0.97	389
SR	0.92	0.94	0.93	222
accuracy			0.92	1059
macro avg	0.92	0.91	0.91	1059
weighted avg	0.92	0.92	0.92	1059

- Model 3 did not show improvement in performance when compared to the CNN only model.
- Compared to the CNN + LSTM model, using GRU did not improve performance. The macro average recall decreased from 0.92 to 0.91.
- The CNN + GRU did not show improvement in performance when compared to the CNN only model
- This illustrates that the LSTM layer is capturing additional features not captured by the CNN layers, while GRU doesn't capture additional features that were not captured by the CNN layers.

Model 4. ResNet + bidirectional LSTM





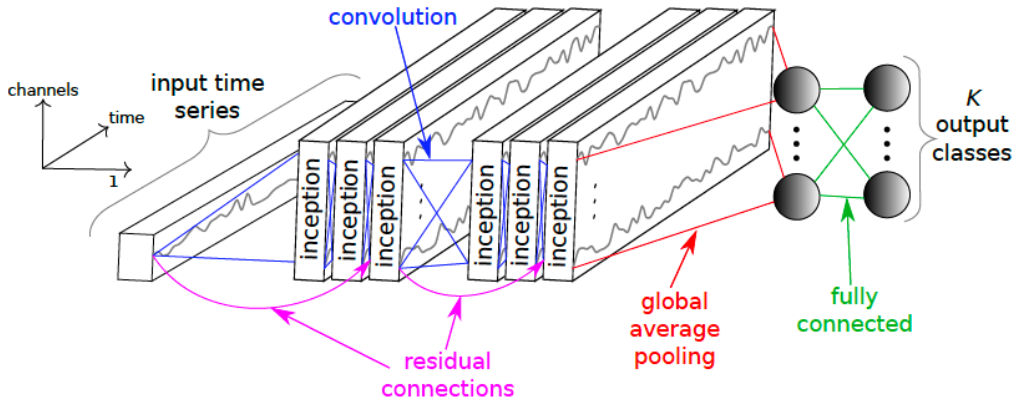
- The ResNet model includes 1D convolutional layers, Residual blocks and Bidirectional LSTM layer.
- This model includes only two residual blocks, instead of 3 described in Wang et al., 2017
- It starts with a convolutional layer with ReLU activation and batch normalization, followed by another convolutional layer.
- The input (shortcut) is added back to the output of these convolutions to form a residual connection
- A ReLU activation is applied after adding the shortcut.
- Two of these residual blocks are used in sequential layers.
- After each residual block, Maxpooling is used to reduce dimensionality.
- Bidirectional LSTM layer is added to capture additional sequence related features. Return sequence is set to 'False' to ensure that the layer output is a single vector instead of sequence, since this is the last layer before the output layer.

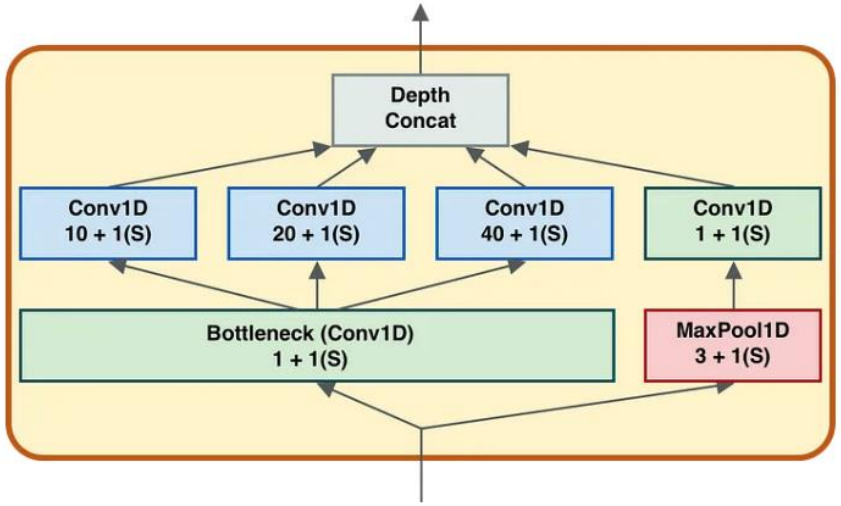
Model 4. ResNet + bidirectional LSTM model performance

	precision	recall	f1-score	support
AF	0.85	0.82	0.83	222
GSVT	0.81	0.81	0.81	226
SB	0.91	0.96	0.93	389
SR	0.83	0.79	0.81	222
accuracy			0.86	1059
macro avg	0.85	0.84	0.85	1059
weighted avg	0.86	0.86	0.86	1059

- The ResNet + bidirectional LSTM did not perform better than the deep CNN model. The macro average recall of 0.84 is much lower than the CNN + LSTM or CNN + GRU models.
- ResNet with GlobalAveragePooling1D layer, as described previously, instead of LSTM was also tested, but model performance was similar to one with bidirectional LSTM.
- Adding additional residual blocks might improve performance.
- The train and validation loss plot shows instability in learning rate during the first 10 ephocs, suggesting potential model instability

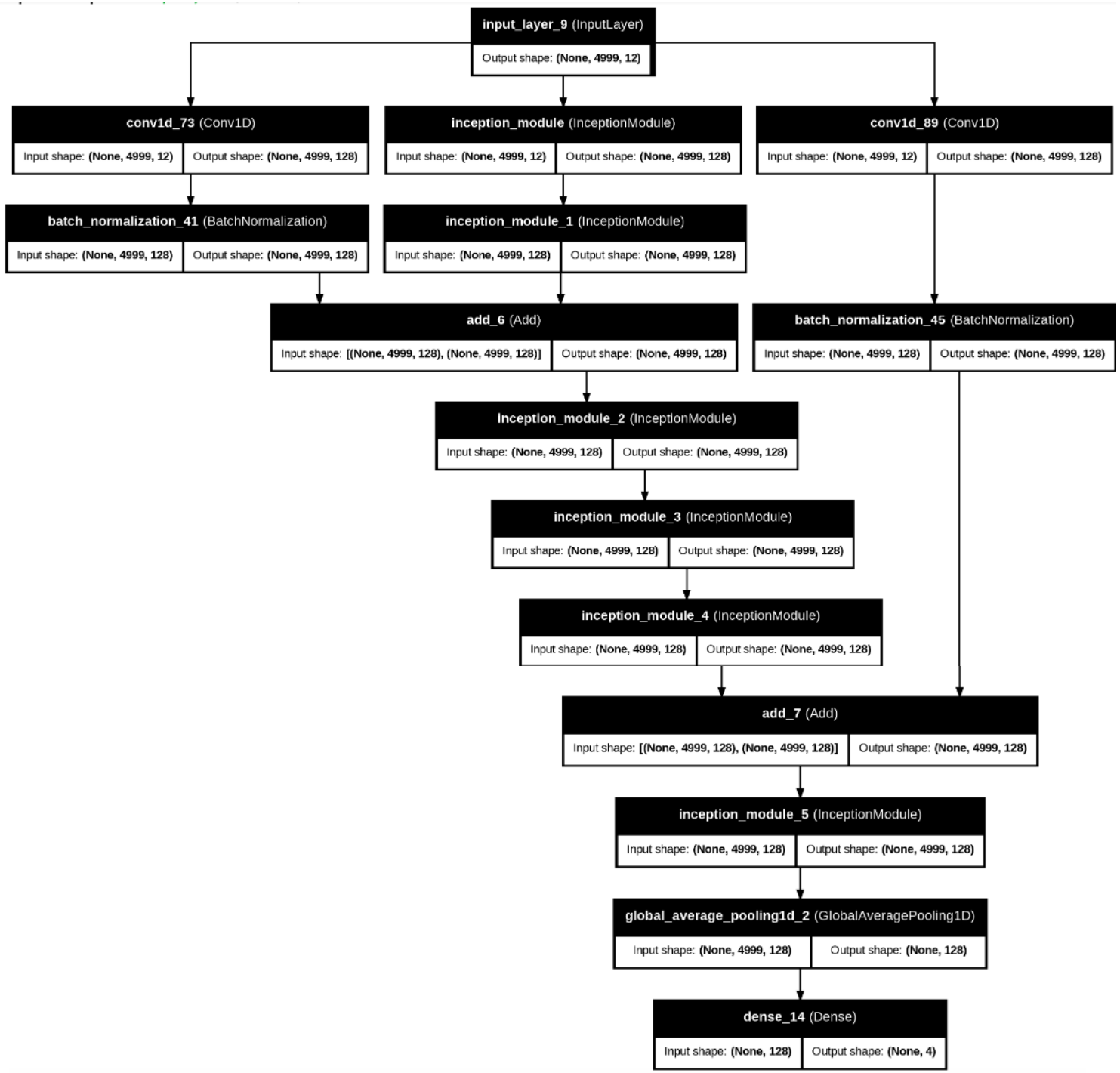
InceptionTime model





Source: https://medium.com/towards-data-science/deep-learning-for-time-series-classification-inceptiontime-245703f422db

- The original model architecture described in Ismail Fawaz et al., 2020 is used.
- The network includes 6 Inception modules and 2 residual connections
- Inception Modules: Implemented three parallel branches with convolutions of different kernel sizes and an average pooling branch. These branches are concatenated.
- Inception module:
  - Number of filters = 32
  - Relu activation is applied after each convolution
  - conv1: A 1D convolutional layer with a kernel size of 1 and padding set to same, acting as a bottleneck to reduce dimensions.
  - maxpool: A 1D max pooling layer with a pool size of 3
  - conv2, conv3, conv4: Three 1D convolutional layers with increasing kernel sizes (16, 32, and 64, respectively), all using 'relu' activation.
  - conv5: Another 1D convolutional layer with a 1x1 convolution to process the output from the max pooling layer.
  - concat: A concatenate layer to merge the outputs from different convolution paths.
- Shortcut layer: Implements a shortcut (or skip) connection for a residual network. Adds the input tensor and the output of the Inception module.



Model 5: InceptionTime model

- Inception module
- Shortcut layer: Implements a shortcut (or skip) connection for a residual network.
- Adds the input tensor and the output of the Inception module.
-

Model 5: InceptionTime model performance

	precision	recall	f1-score	support
AF	0.94	0.94	0.94	222
GSVT	0.92	0.92	0.92	226
SB	0.96	0.99	0.98	389
SR	0.97	0.91	0.93	222
accuracy			0.95	1059
macro avg	0.95	0.94	0.94	1059
weighted avg	0.95	0.95	0.95	1059

- The InceptionTime model performed the best of all the models above, with macro average recall of 0.94
- This model showed the least confusion between AF and GSVT class.
- Although the model has over 2 million trainable parameters, it had a lot less trainable parameter compared to the CNN + LSTM model with over 8M parameters.
- It ran faster than the CNN + LSTM model.
- Hyperparameter tuning was performed for the InceptionTime model below

Model 6: Tuned InceptionTime model

```python
# Hyperparameter ranges for tuning
num_filters = hp.Int('num_filters', min_value=16, max_value=64, step=16)
kernel_size_2 = hp.Int('kernel_size_2', min_value=8, max_value=32, step=8)
kernel_size_3 = hp.Int('kernel_size_3', min_value=16, max_value=64, step=16)
kernel_size_4 = hp.Int('kernel_size_4', min_value=32, max_value=128, step=32)

class InceptionModule(tf.keras.layers.Layer):
    def __init__(self, activation='relu', **kwargs):
        super(InceptionModule, self).__init__(**kwargs)
        self.activation = activations.get(activation)

        # Create layers with hyperparameters
        self.conv1 = Conv1D(filters=num_filters, kernel_size=1, strides=1, activation='relu', use_bias=False, padding='same')
        self.maxpool = MaxPool1D(pool_size=3, strides=1, padding='same')
        self.conv2 = Conv1D(filters=num_filters, kernel_size=kernel_size_2, strides=1, activation='relu', use_bias=False, padding='same')
        self.conv3 = Conv1D(filters=num_filters, kernel_size=kernel_size_3, strides=1, activation='relu', use_bias=False, padding='same')
        self.conv4 = Conv1D(filters=num_filters, kernel_size=kernel_size_4, strides=1, activation='relu', use_bias=False, padding='same')
        self.conv5 = Conv1D(filters=num_filters, kernel_size=1, strides=1, activation='relu', use_bias=False, padding='same')
        self.concat = Concatenate()
        self.batchnorm = BatchNormalization()
```

```
Best number of filters: 32
Best kernel size for conv2: 16
Best kernel size for conv3: 32
Best kernel size for conv4: 96
```

- As suggested by Ismael et al., 2020, most important hyperparameters are the number of filters and kernel sizes in the three convolution layers within the inception module.

Model 6: Tuned InceptionTime model performance

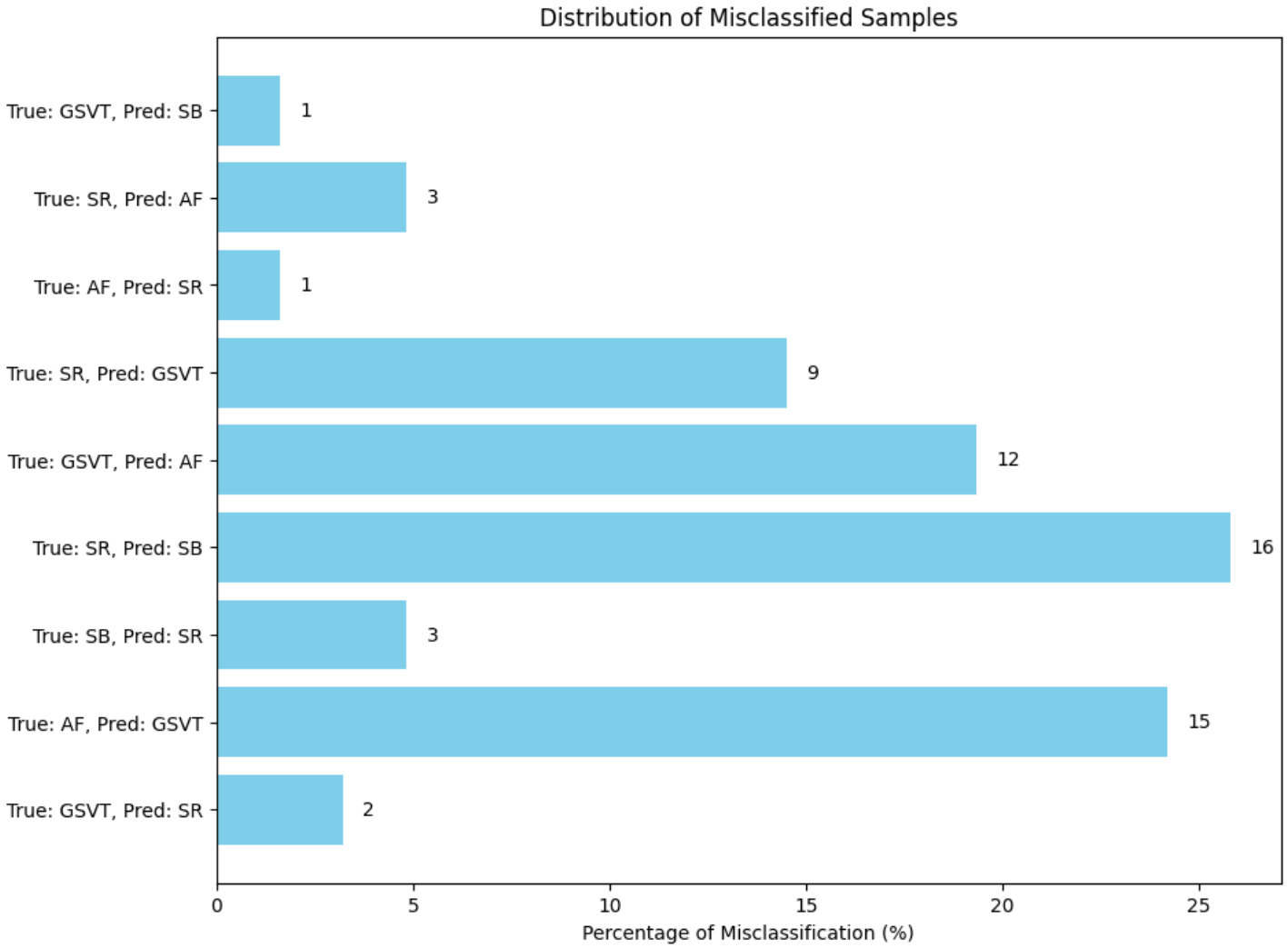
	precision	recall	f1-score	support
AF	0.93	0.93	0.93	222
GSVT	0.90	0.93	0.92	226
SB	0.96	0.99	0.97	389
SR	0.97	0.87	0.92	222
accuracy			0.94	1059
macro avg	0.94	0.93	0.93	1059
ighted avg	0.94	0.94	0.94	1059

- The macro average recall was 0.93, which was slightly less than model 5 score of 0.94.
- The tuned model showed reduced performance on SR class, compared to model 5

Model comparisons

Labels	CNN	CNN+LSTM	CNN+GRU	ResNet + LSTM	InceptionTIme	InceptionTIme tuned
AF	0.80	0.83	0.80	0.82	0.94	0.93
GSVT	0.96	0.94	0.92	0.81	0.92	0.93
SB	0.98	0.99	0.98	0.96	0.99	0.99
SR	0.95	0.95	0.94	0.79	0.91	0.87
Macro average recall	0.92	0.93	0.91	0.84	**0.94**	0.93

Evaluation of misclassified samples



Discussion and Conclusions

- InceptionTime, showed the best performance with macro average recall of 0.94, despite having less trainable parameters compared to the CNN + LSTM model
- The deep 1D CNN + LSTM model showed an accuracy of 96% on arrhythmia previously. Consistently, in this report, the model showed an average accuracy of 94%, which is very close to the published results.
- Previous study using ECGs from another database reported F1 score of 0.92 on arrhythmia classification. Similarly, the the analysis in this study found similar performance, with F1 score of 0.94.
- InceptionTime model showed a good balance between correct classification of AF and GSVT groups, while all the deep 1D CNN models performed the worst in classifying these two groups.

Discussion and Conclusions cont.

- The AF and GSVT confusion could be investigated by sequentially removing each of the minority arrhythmia subtypes and observing whether the confusion is reduced could help determine which ECG might be causing the misclassification.

- In addition, including age and sex in the prediction could further improve the InceptionTime model performance as they are clinically important.

- In fact, a study have shown that ECG can predict sex and age with a 90% accuracy, strongly indicating that inclusion of these variables in arrythmia classification is important.

References

1. J. Zheng, J. Zhang, S. Danioko, H. Yao, H. Guo, C. Rakovski, A 12-lead electrocardiogram database for arrhythmia research covering more than 10,000 patients. *Sci. Data* **7**, 48 (2020)
2. N. Strodthoff, P. Wagner, T. Schaeffter, W. Samek, Deep learning for ECG analysis: Benchmarks and insights from PTB-XL. *IEEE J. Biomed. Health Inform.* **25**, 1519–1528 (2021).

END