Word Count: 2200

Classifying ECG Heartbeat Using the SMOTE-CNN model

Project Title: Design, Development, Analysis and Performance Evaluation of Deep Learning algorithms

NAME: OLAKUNLE KUYORO

Table of Contents

1 Introduction

Heart disease is one of the leading causes of mortality globally each year, according to the World Health Organization (WHO) (Mathews, Kambhamettu and Barner, 2018). Research has focused a lot of attention on how to diagnose cases of cardiac arrest or heart failure. Heartbeat conditions include irregular heartbeats or heartbeats that are too slow, too early, too quick, or both (Li *et al.*, 2020). Symptoms, including shortness of breath, palpitations, dizziness, and fainting, can identify heart disorders. Studies stated many cardiovascular diseases and strokes are prominent with heart failure. Cutting-edge electrocardiogram (ECG) technology has gained widespread attention in studying heart conditions, leading to how heart cases are managed. The ECG depicts the heart's electrical activity variations over time and offers vital physiological data frequently used to evaluate heart function. Electrodes placed on the body's surface record the cardiac electric field. It is measured as a voltage (or potential difference) between two electrodes. A "lead" is a pair of electrodes that create a fictitious line in the body along which electric impulses are monitored (Alarsan and Younes, 2019)

Figure 1 illustrates ECG signals are periodic signals composed of a sequence of periodical waves overtime time. It consists of 6 waves: P, Q, R and S, which form the QRS complex, followed by a T wave and the U wave. A U wave can occasionally be seen, though. The QRS complex is the most recognizable component of an ECG signal. The study of this complex precisely matches the beat-to-beat categorization (Sannino and De Pietro, 2018).

The rapid growth of novel sensing and imaging technologies, such as computer-based diagnostic (CAD) systems, has expanded the efficient capturing and annotation of images like ECG data, coupled with advancements in machine learning (ML) (Aziz, Ahmed and Alouini, 2021).

(Xie *et al.*, 2019) developed a Feature Enrichment-Convolutional Neural Network (FE-CNN) classifier by turning the ECG signals into time-frequency images. Results show that the

MIT-BIH dataset demonstrates that FE-CNN can detect supraventricular ectopic (S) beats with a sensitivity (Sen) of 75.6%, a positive predictive rate (Ppr) of 90.1%, and an F1 score of 0.82. Sen, Ppr, and F1 scores for ventricular ectopic (V) beat identification are 92.8 per cent, 94.5 per cent, and 0.94, respectively.

Romdhane *et al.* (2020) introduced a deep CNN model with a stepwise optimization utilizing a brand-new loss function termed focal loss (FL). The FL seeks to elevate the minority heartbeat classes in prominence. The model's total performance was 98.41 per cent accurate, 98.38 per cent F1-score, 98.37 per cent precise, and 98.41 per cent recall.

In this paper, I offer a SMOTE-Deep CNN for ECG heartbeat classification based on the course's goal. I also provide a random search optimization to choose the appropriate hyperparameter and train the model for the best results.

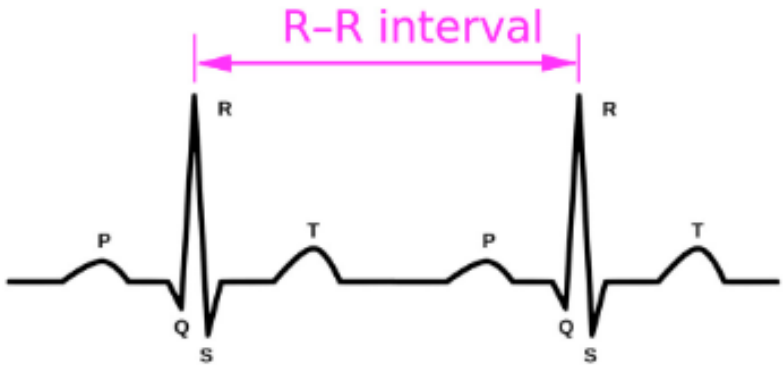


Figure 1 ECG heartbeat waveform (source: Sannino and De Pietro, 2018)

2 Datasets

In this study, I used the MIT-BIH arrhythmia database, which includes many types of arrhythmias. These five classes are defined by AAMI as N, S, V, F and Q, as presented in Table 1. The ECG Heartbeat dataset consisting two parts: MIT-BIH Arrhythmia and PTB Diagnostic.

Table 1. The AAMI EC57 standard notation for the MIT-BIH ECG dataset (source: (Li *et al.*, 2020)

Category	Annotations
N	Normal
S	Supra-ventricular premature
V	Ventricular escape
F	Fusion beats
Q	Unclassifiable proposed

3 Methodology

3.1 Data Pre-processing

I encoded the categories in Table 1 for data encoding on the dataset. The target labels as floating values are converted to an integer.

3.2 Stepwise Procedure of the Proposed Approach

I suggested using SMOTE-CNN to categorize the ECG beats in this study. There are three primary steps to the proposed strategy. The ECG dataset is first balanced using the SMOTE method in the first stage because we are dealing with an unbalanced dataset. The majority class may outperform the minority during training if this data is analyzed without considering its imbalance. The CNN model, the second step, is fed the output from the first stage. The CNN model is a Conv 1D type with a convolutional layer, batch normalization, dense layers, and rectifier linear unit (RELU) as the activation function (AF). Since the result is a multi-class label, a softmax AF is used for the output layer. I introduced a *ReduceLROnPlateau* callback for the model to converge faster on the learning rate.

Meanwhile, before balancing the data using SMOTE, the CNN model is trained with the unbalanced data and performance, including F1-score, precision, recall and area under the

curve (AUC) obtained. After that, the CNN model is re-trained with the balanced dataset and obtains the same evaluation metrics. Having obtained the results and compared, the best model is then subject to random search optimization to determine the best hyperparameter for improving the model, which happens to be the third stage. The overall results are then compared. Figure 2 depicts the proposed methodology used in the paper.

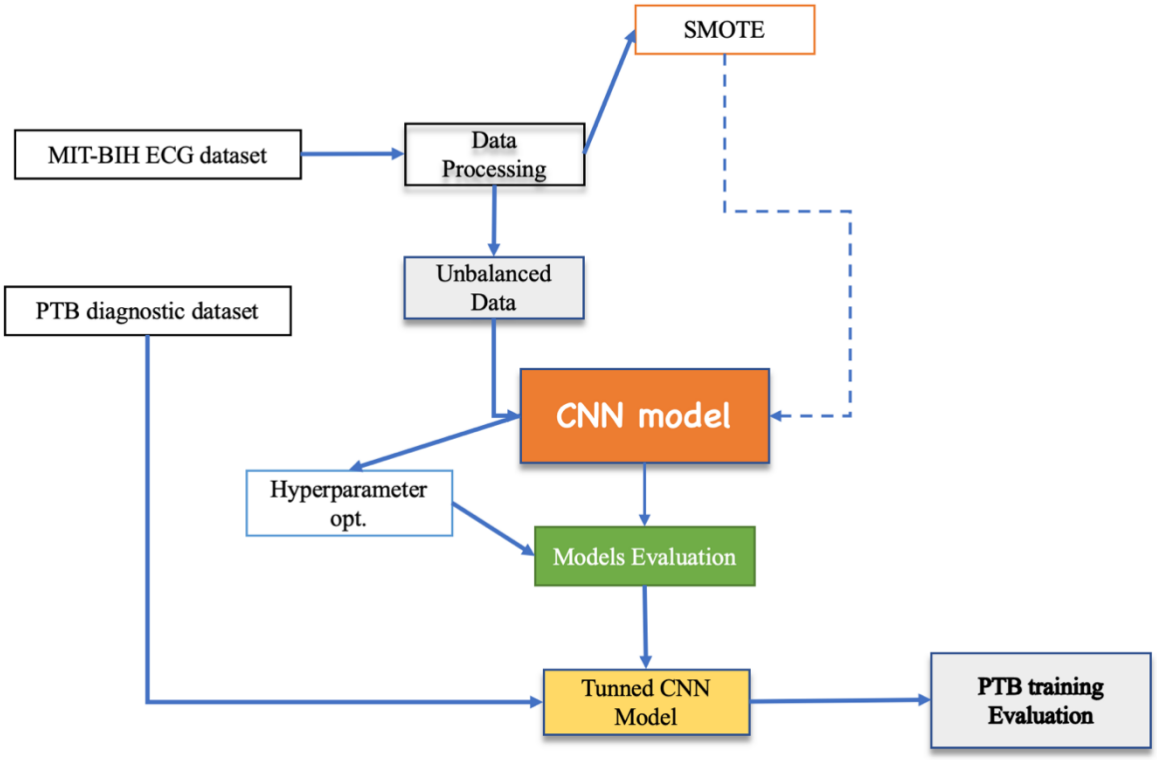


Figure 2 Proposed SMOTE-CNN for ECG heartbeat classification

4 Model Training

I used TensorFlow, Keras, and Python libraries to build the model, including Jupyter Notebooks, for my experiments. The model's starting parameters for training are presented in Table 2.

Table 2. CNN Model setups as pre-trained model

CNN Model	Convl Layers			Global Average Pooling	Dense Layers			Output Layer
Parameters	**1st**	**2nd**	**3rd**		**1st**	**2nd**	**3rd**	
Filters	64	64	64					
Kernel size	9	3	3					
AF	Relu	Relu	Relu		relu	relu	relu	softmax
Input shape	(187,1)	(187,1)	(187,1)					
Padding	Same	same	same					
Pool size								
Strides								
Units					64	32	16	5
Dropout	0.5	0.5	0.5		0.5	0.5	0.5	

5 Results and Analysis

5.1 Loading dataset and pre-processing

The first step is loading the MIT-BIH and PTB datasets into the Notebook environment, as shown in Figures 3 (a) and (b). The PTB dataset contained two different files. One for 'normal' and the other for 'abnormal'. I merged the two row-wise and shuffled them to have the target label miss together.

	0	1	2	3	4	5	6	7	8	9
0	0.977941	0.926471	0.681373	0.245098	0.154412	0.191176	0.151961	0.085784	0.058824	0.049
1	0.960114	0.863248	0.461538	0.196581	0.094017	0.125356	0.099715	0.088319	0.074074	0.082
2	1.000000	0.659459	0.186486	0.070270	0.070270	0.059459	0.056757	0.043243	0.054054	0.045
3	0.925414	0.665746	0.541436	0.276243	0.196133	0.077348	0.071823	0.060773	0.066298	0.058
4	0.967136	1.000000	0.830986	0.586854	0.356808	0.248826	0.145540	0.089202	0.117371	0.156

5 rows × 188 columns Open in new tab

(a) MIT-BIH samples

	0	1	2	3	4	5	6	7	8
931	1.000000	0.542760	0.197290	0.083404	0.092295	0.168501	0.222693	0.216765	
8705	1.000000	0.524785	0.259749	0.161269	0.111038	0.058163	0.044944	0.029742	
680	0.952135	0.981242	0.460543	0.188875	0.000000	0.091850	0.253558	0.278137	
7327	1.000000	0.878049	0.652820	0.438262	0.298399	0.153963	0.089177	0.065549	
2048	0.991001	1.000000	0.744839	0.397035	0.154579	0.166226	0.133404	0.113817	

(b) PTB samples

Figure 3 ECG Datasets

I checked for missing values and converted the target label value to an integer value, as shown in Figure 4. There are no missing values in both datasets.

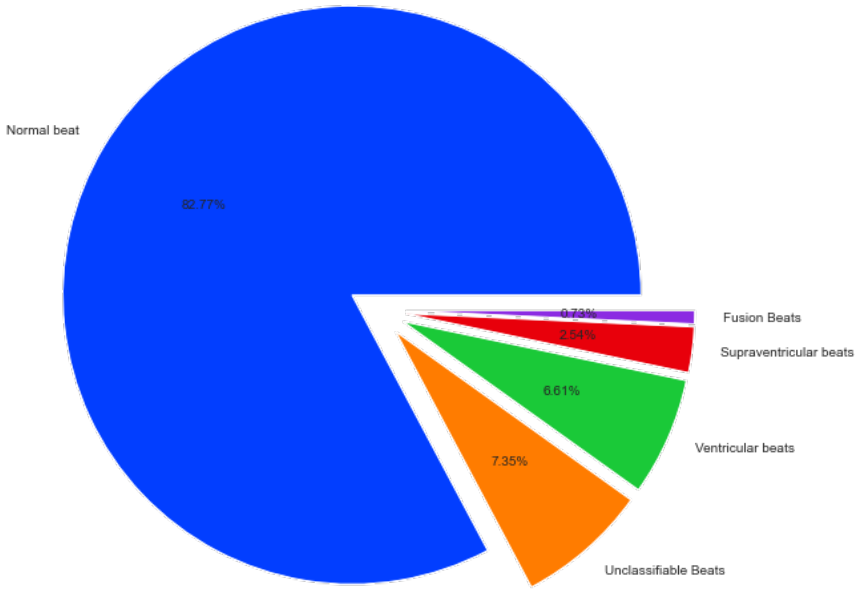
Figure 4 Checking for missing values

5.2 Exploratory Data Analysis

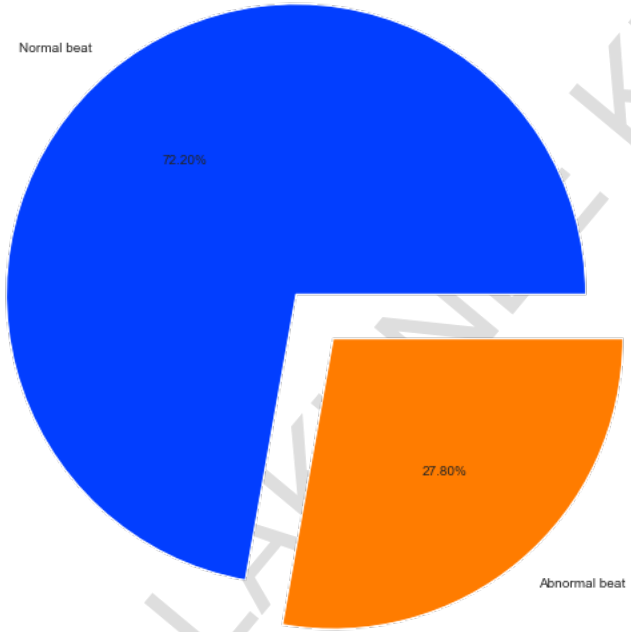
I performed a descriptive analysis and visualization technique to understand the datasets. Frist, I explore the distribution of the target label since we are looking for a classification problem.

Figure 5 MIT-BIH and PTB instances and features

Figure 6 shows the distribution of the two datasets. The datasets MIT-BIH and PTB are not uniformly distributed. The 'normal heartbeat':0 has the largest value, 10x more than the rest of the heartbeat types, while in the case of PTB, the normal is twice the abnormal. The distributions and percentage counts of each label are represented in Table 3.



(a) Pie Chart for MIT-BIH class distribution



(a) Pie chart for PTB class distribution

Figure 6 MIT-BIH and PTB classes distribution

Table 3 Distribution of MIT-BIH class labels

Category	Category Abbreviation	Counts	% Count
0	N	72471	82.77
1	S	2223	2.54
2	V	5788	6.61
3	F	641	0.73
4	Q	6431	7.35
		87554	

From the above, it shows that the datasets are imbalanced. This scenario is called an imbalanced data problem, and the need for the imbalanced-based technique is proposed with the CNN model.

I used a plot function to understand a regular, 'normal', and 'abnormal' heartbeats. Figure 7 shows a typical heartbeat with an oscillating waveform and the 'normal' and 'abnormal' heartbeat shown in Figure 8.

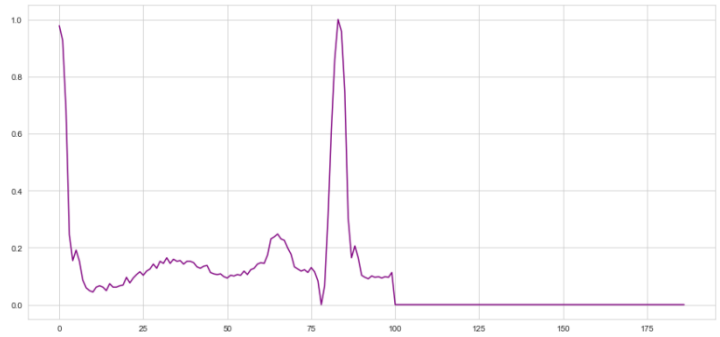
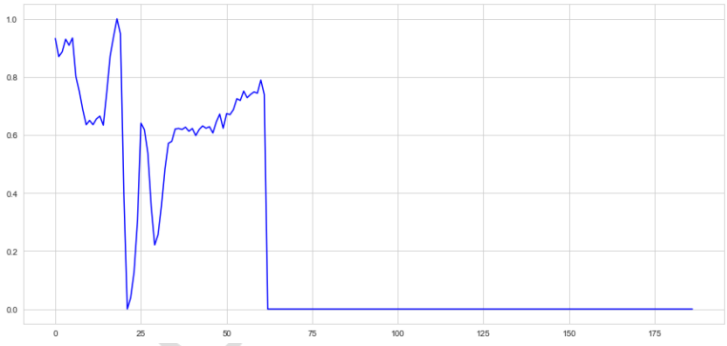
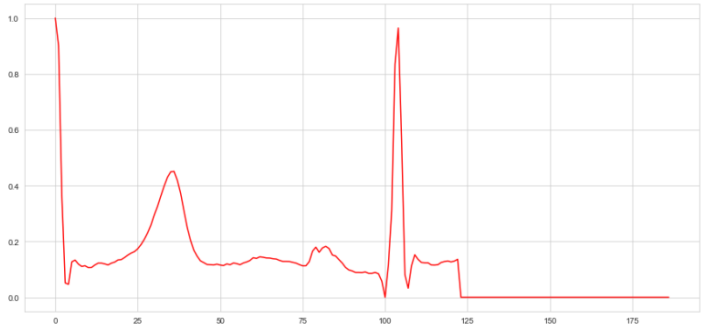


Figure 7 A heartbeat wave



(a) Normal



(b) Abnormal

Figure 8 Normal and abnormal heartbeat wave

I performed a comparative plot to distinguish the 'normal' from other heartbeats, as shown in Figure 9.

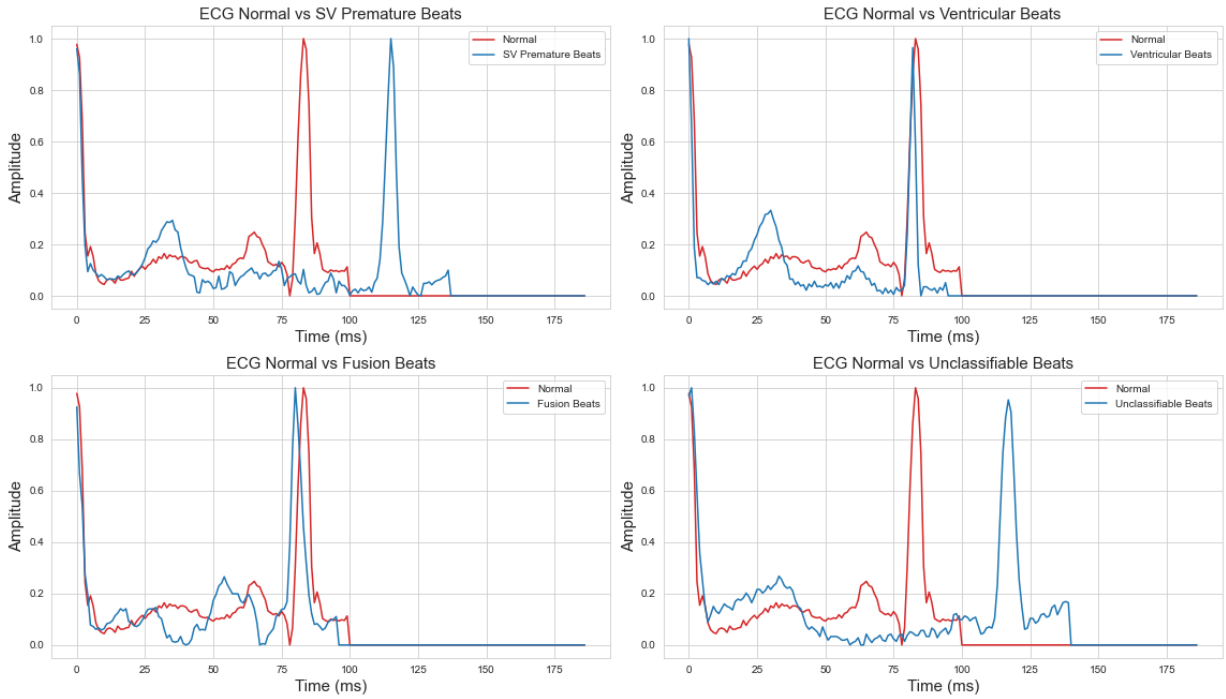


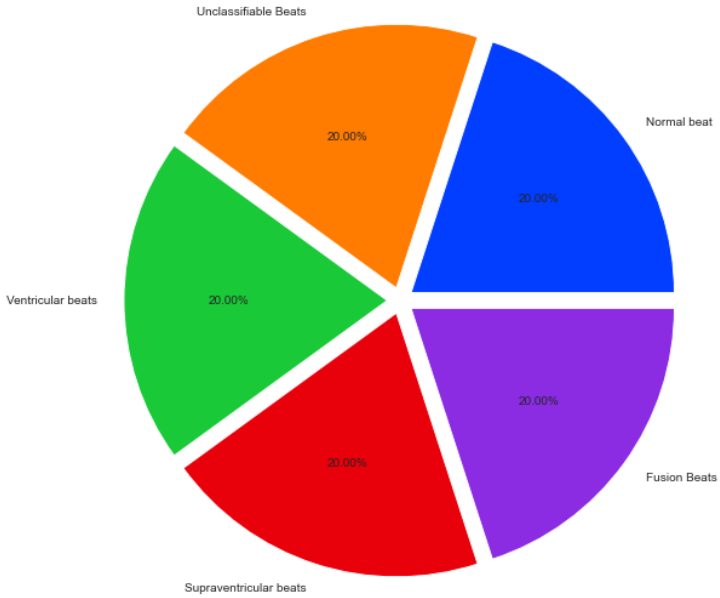
Figure 9 Comparison of Heartbeat classes

5.3 Imbalance Data Technique Used

Based on the evidence that the heartbeat datasets are imbalanced, analyzing such data, especially with CNN, has every tendency for the model to overfit. To overcome the overfitting challenge due to the imbalanced data, I used an oversampling technique called Synthetic Minority Oversampling Technique (SMOTE) (Chawla *et al.*, 2002). The SMOTE resampling result is shown in Figure 10.

```
Original train dataset shape:  Counter({0: 72471, 4: 6431, 2: 5788, 1: 2223, 3: 641})
New train dataset shape:  Counter({0: 72471, 1: 72471, 2: 72471, 3: 72471, 4: 72471})
Original test dataset shape:  Counter({0: 18118, 4: 1608, 2: 1448, 1: 556, 3: 162})
New test dataset shape:  Counter({0: 18118, 1: 18118, 2: 18118, 3: 18118, 4: 18118})
```

(a) Train and test instances after oversampling using SMOTE

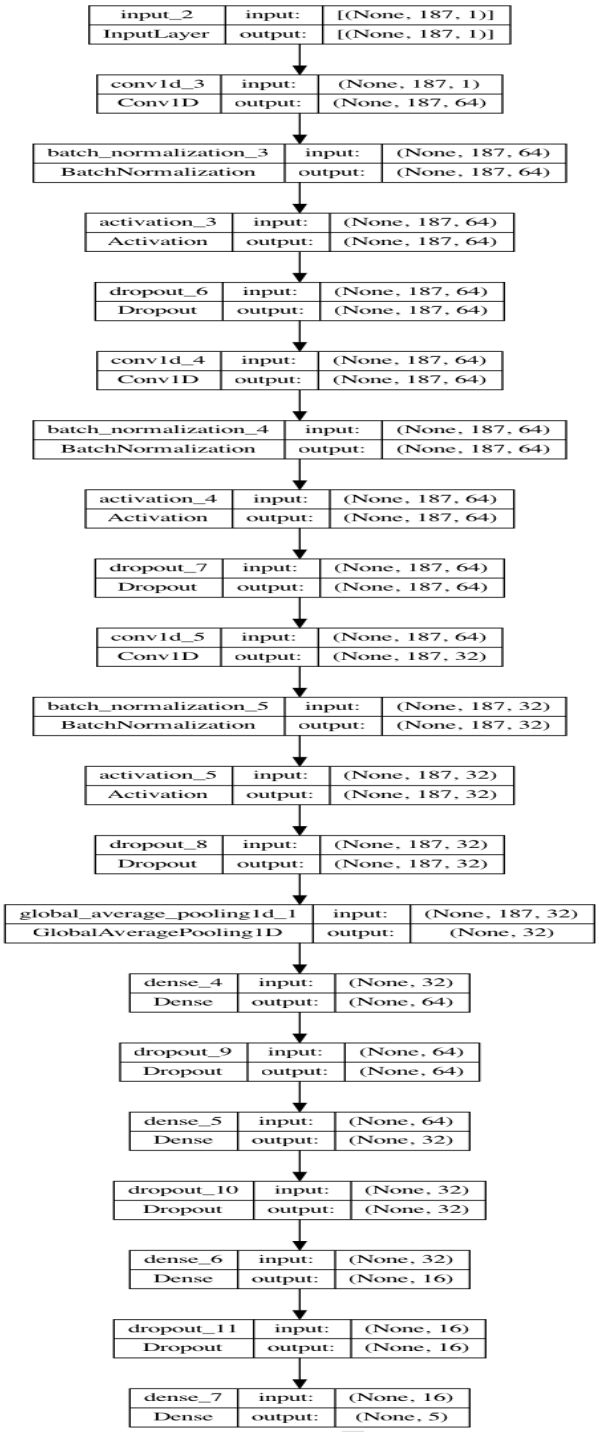


(b) SMOTE generates an equal fraction of the class label instances.

Figure 10 SMOTE resampling

5.4 Model Training

Based on the fact that a test set is already provided, I did not perform a splitting process of the dataset. I used to_categorical() method in *keras.utils.np utils* to extract the class labels from each newly acquired sample and convert them to categorical data. The train and test features are separated from each newly produced sample with the model diagram, summary and training, as shown in Figure_11.



(a) The CNN Model graph

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 187, 64)           640

 batch_normalization (BatchN  (None, 187, 64)          256
 ormalization)

 activation (Activation)     (None, 187, 64)           0

 dropout (Dropout)           (None, 187, 64)           0
```

(b) Part of the model summary

```
Epoch 47/50
11324/11324 [==============================] - 451s 40ms/step - loss: 0.3912 -
accuracy: 0.8635 - val_loss: 0.2914 - val_accuracy: 0.9017 - lr: 1.0000e-04
Epoch 48/50
11324/11324 [==============================] - 467s 41ms/step - loss: 0.3950 -
accuracy: 0.8630 - val_loss: 0.2936 - val_accuracy: 0.9032 - lr: 1.0000e-04
Epoch 49/50
11324/11324 [==============================] - 380s 34ms/step - loss: 0.3912 -
accuracy: 0.8643 - val_loss: 0.2937 - val_accuracy: 0.9035 - lr: 1.0000e-04
Epoch 50/50
11324/11324 [==============================] - 445s 39ms/step - loss: 0.3894 -
accuracy: 0.8647 - val_loss: 0.2940 - val_accuracy: 0.9034 - lr: 1.0000e-04
```

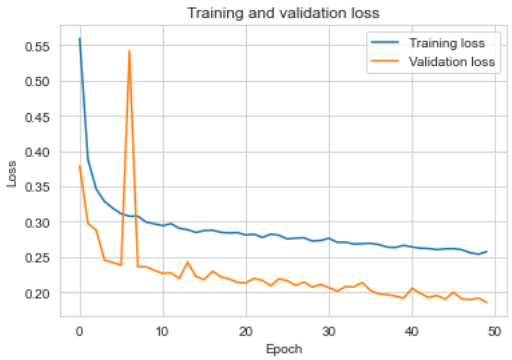
(c) Model training process

Figure 11 The CNN model

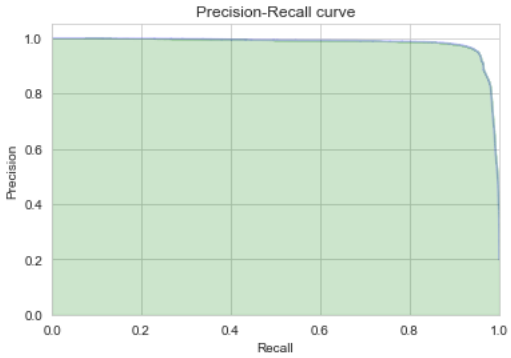
5.5 No Sampling-CNN model

The CNN model is first trained with ECG data without resampling. The training and validation accuracy, training and validation loss, precision, recall, and AUC were obtained, as shown in Figure 12. Subfigure 12(a) shows that the model overfits because validation accuracy peaks higher than the training accuracy. The same trend is observed for training and validation loss in Subfigure 12(b). Though the AUC obtains 0.95% as a relatively good performance, Table 4 correctly indicates that the model is biased towards the majority classes (0, 2, and 4) while the categories (1 and 3) as minority class scores zero for all metrics. Such a result like this is undesirable. The overall model result is presented in Table 5.

(a) Training and validation accuracy



(b) Training and validation loss



(c) The AUC for the No-sampling CNN model

Figure 12 No sampling-CNN Model performance visualization

Table 4 No-sampling-CNN model performance for each class

Class	Recall	Precision	F1-score	Sample Support
0	1.00	0.95	0.98	18,118
1	**0.00**	**0.00**	**0.00**	**556**
2	0.87	0.89	0.88	1,448
3	**0.00**	**0.00**	**0.00**	**162**
4	0.92	0.98	0.95	1,608

Table 5 No-sampling-CNN model overall performance

Model	Validation Loss	Validation Accuracy	F1-score	Precision	Recall	Area Under Curve
No-Sampling-CNN	0.19	0.95	0.94	0.92	0.95	0.95

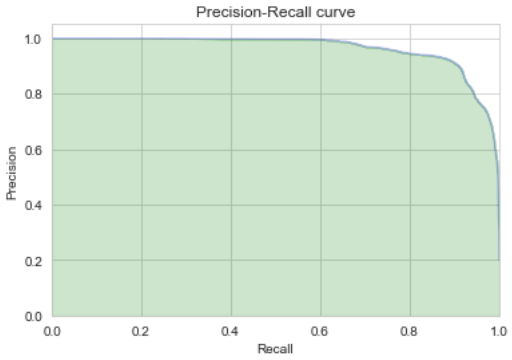
5.6 SMOTE-CNN model

The CNN model behaves differently with SMOTE. However, the validation accuracy and loss project are each shown in subfigure (a) and (b) of Figure 13. Meanwhile, Table 6 shows that the challenge of class overfitting is solved. The model performance cut across the classes and especially for the minority classes (1 and 3) with F1-scores of 0.81 and 0.93, respectively. The overall performance of the SMOTE-CNN model in terms of F1-score 0.90 is lower than the initial model with data resampling of 0.95; however, the AUC of SMOTE-CNN with 0.98 shows that the model generalizes than the initial model with AUC of 0.95. From those, as mentioned earlier, I discussed in the next section the hyperparameters optimization of the

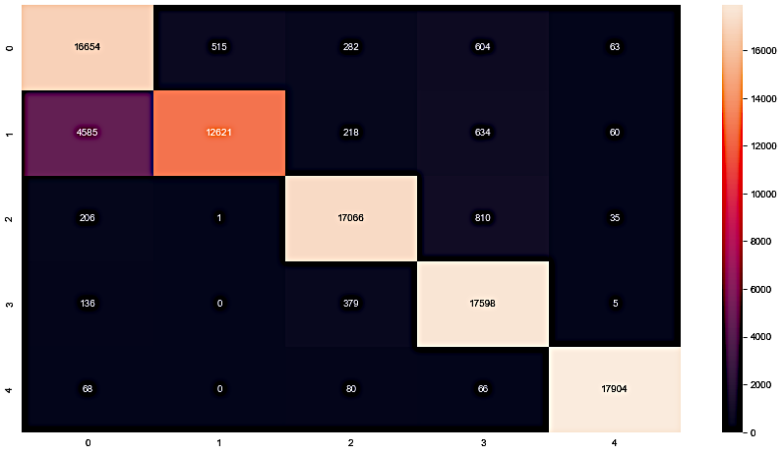
SMOTE-CNN using the Random Search algorithm. The overall SMOTE-CNN is presented in Table 7.

(a) Training and validation accuracy

(b) Training and validation loss



(c) The AUC for the SMOTE-CNN model



(d) Confusion matrix of the SMOTE-CNN model

Figure 13 SMOTE-CNN Model performance visualization

Table 6 SMOTE-CNN model performance for each class

Class	Recall	Precision	F1-score	Sample Support
0	0.92	0.77	0.84	18,118
1	**0.70**	**0.96**	**0.81**	18,118
2	0.94	0.95	0.94	18,118
3	**0.97**	**0.89**	**0.93**	18,118
4	0.99	0.99	0.99	18,118

Table 7 SMOTE-CNN model performance

Model	Validation Loss	Validation Accuracy	F1-score	Precision	Recall	Area Under Curve
SMOTE-CNN	0.29	0.90	0.90	0.91	0.90	0.98

5.7 Random Search Hyperparameter

To investigate further improvement on the SMOTE-CNN model, I performed a Random Search hyperparameter. I also included an early stopping criterion for the initial callbacks to obtain the best model. The hyperparameter setting and the early stopping criterion are defined as follows:

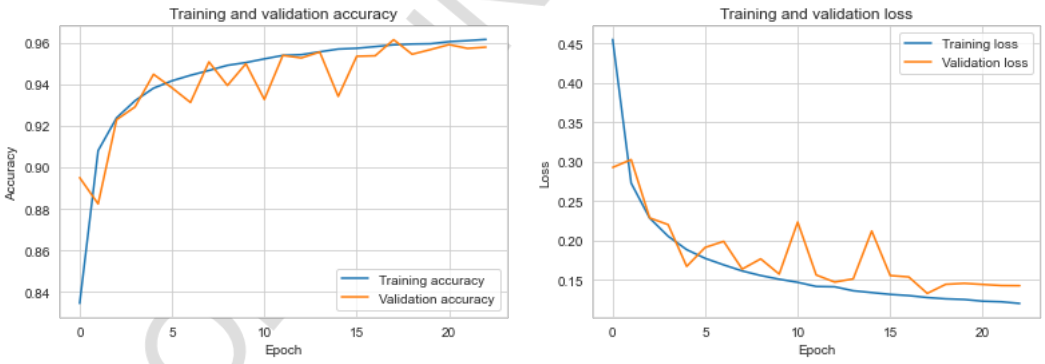
```
optimizer=['adam', 'Adadelta', 'Nadam']
filters= [64, 128]
kernel_size=[3, 6, 9]
dense_units=[32, 64, 128]
batch_size=[32, 64]
dropout=[0.2, 0.3, 0.5]

reduce_lr =
keras.callbacks.ReduceLROnPlateau(monitor='val_loss',
factor=0.2, patience=5, min_lr=0.0001)
early = [EarlyStopping(monitor='val_loss',
patience=5),ModelCheckpoint(filepath='smote_cnn.h5',
monitor='val_loss', save_best_only=True)]
```

The best model hyperparameters obtained are `optimizer='Nadam', filters=64, kernel_size=9, dense_units=64, batch_size=64, dropout=0.2 with 50 epochs.`

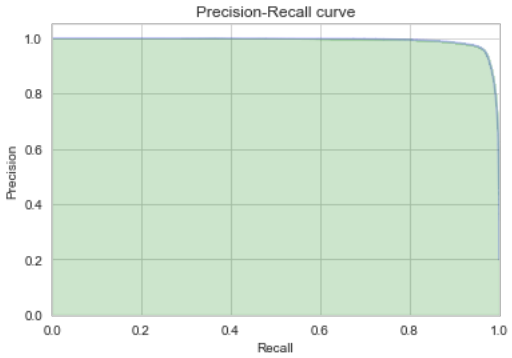
`Obtained from the model below:`

```
def create_model (optimizer='adam', filters=32,kernel_size=9,
dense_units=32, dropout=0.2, batch_size=32, epochs=20):
    model = keras.Sequential([
        layers.Input(shape=(187,1)),
        layers.Conv1D(filters=filters, kernel_size=kernel_size,
padding='same'),
        layers.BatchNormalization(),
        layers.Activation('relu'),
        layers.Dropout(dropout),
        layers.Conv1D(filters=filters, kernel_size=(kernel_size-3),
padding='same'),
        layers.BatchNormalization(),
        layers.Activation('relu'),
        layers.Dropout(dropout),
        layers.Conv1D(filters=int(filters//2),
kernel_size=int(kernel_size//3), padding='same'),
        layers.BatchNormalization(),
        layers.Activation('relu'),
        layers.Dropout(dropout),
        layers.GlobalAveragePooling1D(),
        layers.Dense(dense_units, activation='relu'),
        layers.Dropout(dropout),
        layers.Dense(int(dense_units//2), activation='relu'),
        layers.Dropout(dropout),
        layers.Dense(int(dense_units//4), activation='relu'),
        layers.Dropout(dropout),
        layers.Dense(y_train.shape[1], activation='softmax')
    ])
    model.compile(optimizer=optimizer, loss='categorical_crossentropy',
metrics=['accuracy'])
    return model
```

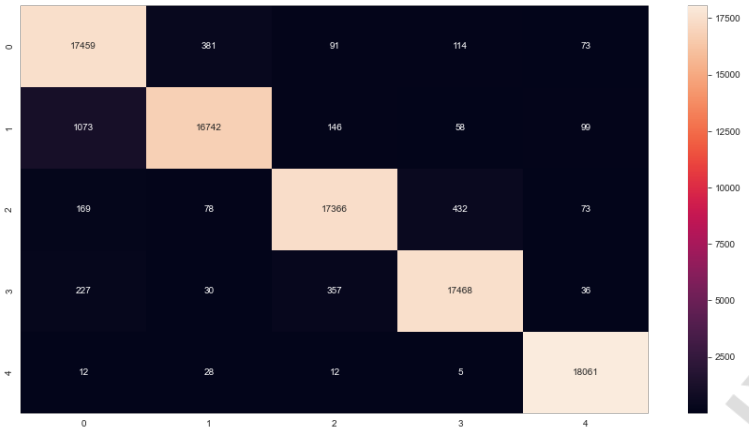
The results show that the tunned model performed considerably well, as shown in Figure 14. The model performance for each class significantly improved with coverage for the minority class maintained, as presented in Table. The tunned SMOTE-CNN obtains an F1-score of 0.96, outperforming the SMOTE-CNN and CNN without data resampling models.



(a) Training and validation accuracy (b) Training and validation loss



(c) The AUC for the Tunned-SMOTE-CNN model



(d) Confusion matrix of the Tunned-SMOTE-CNN model

Figure 14 Tuned-SMOTE-CNN Model performance visualization

Table 8 Tunned-SMOTE-CNN model performance for each class

Class	Recall	Precision	F1-score	Sample Support
0	0.96	0.92	0.94	18,118
1	0.92	0.97	0.95	18,118
2	0.96	0.97	0.96	18,118
3	0.96	0.87	0.97	18,118
4	1.00	0.98	0.99	18,118

Table 9 Tunned-SMOTE-CNN model performance

Model	Validation Loss	Validation Accuracy	F1-score	Precision	Recall	Area Under Curve
Tunned-SMOTE-CNN	0.13	0.96	0.96	0.96	0.96	0.99

6 Pre-trained model transfer on PTB ECG Heartbeat

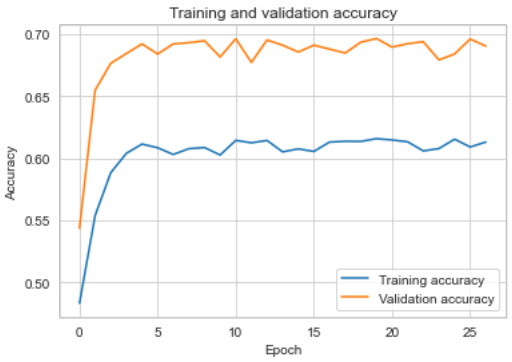
Consider that the MIH-BIT and PTB ECG are benchmarked for heartbeat analysis. The MIH-BIT classifier model weights are used for the final heartbeat classification on the PTB dataset. The former consists of five classes, while the latter consists of two categories:' normal' and 'abnormal'. I split the dataset first into two: train and test sets with a ratio of 80:20. Then split the train again into train and validation sets with a balance of 70:30. I used both the train and validation sets for training and validating the pre-trained SMOTE-CNN model. The test is used to evaluate the SMOTE-CNN model. To adapt the pre-trained SMOTE-CNN model to the PTB analysis, I load the weights of the pre-trained model and fit on the data with the following parameter settings: loss=binary_crossentropy, batch_size=32, AF='sigmoid', epoch=50, optimizer='adam', and the output layer=2 (1 or 0). The same callbacks setting is also used to obtain the best model for evaluation, as shown below:

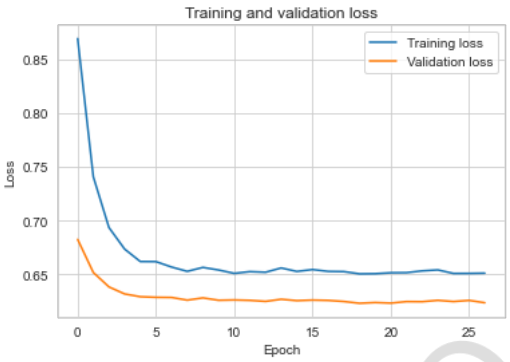
```
reduce_lr =
keras.callbacks.ReduceLROnPlateau(monitor='val_loss',
factor=0.2, patience=5, min_lr=0.0001)
early = [EarlyStopping(monitor='val_loss',
patience=5),ModelCheckpoint(filepath='smote_cnn.h5',
monitor='val_loss', save_best_only=True)]
```

The results show the pre-trained SMOTE-CNN model performs on the PTB data with an AUC score of 76%, as presented in Table. Figures 17 (a) and (b) illustrate the training and validation's accuracy and loss. The training and validation accuracy increases as the epochs increase, while the training and validation loss decreases as the epochs increase.

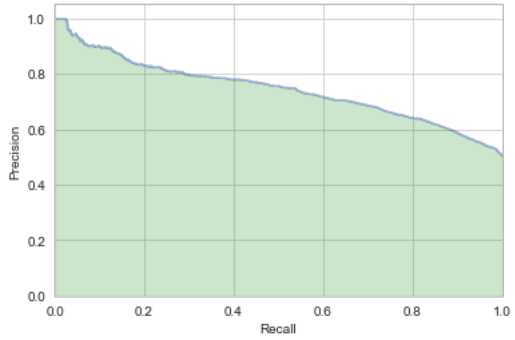
Table 10 Tunned-SMOTE-CNN classifier on PTB datasets

Model	Validation Loss	Validation Accuracy	F1-score	Precision	Recall	Area Under Curve
Tunned-SMOTE-CNN	0.62	0.69	0.69	0.71	0.69	0.76

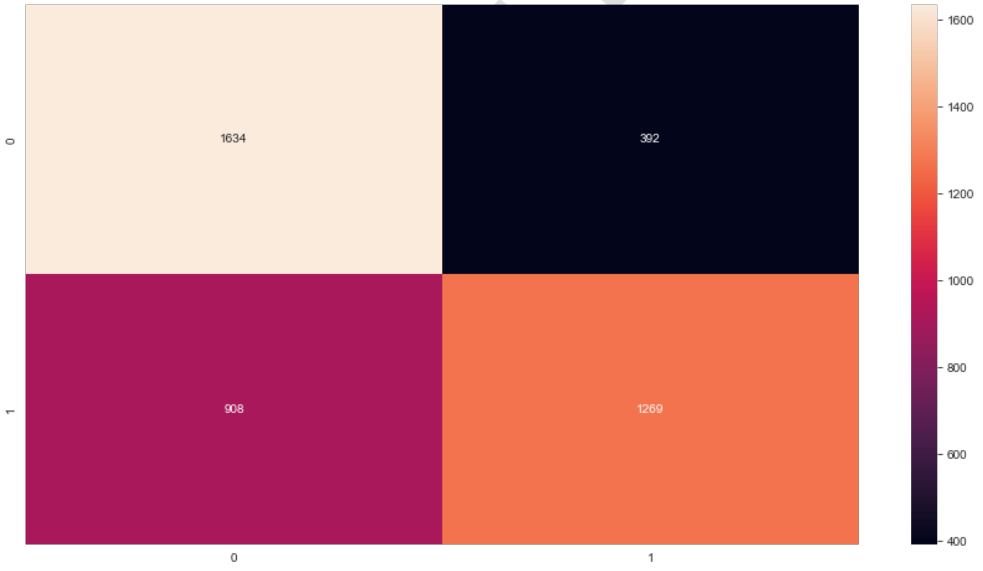




(a) Training and validation accuracy (b) Training and validation loss



(c) The AUC of the Tunned-SMOTE-CNN for the PTB dataset



(d) Confusion matrix of the tunned-SMOTE-CNN model for the PTB dataset

7 Conclusion

I have demonstrated using a deep learning model for ECG heartbeat classification in this paper. I implemented a Conv 1D of the CNN model keeping in mind the nature of the datasets. I have also discussed using SMOTE to balance the datasets to solve the challenge of model overfitting. The random search hyperparameter improves the SMOTE-CNN model with an F1-score of 96%. In addition, I discussed since the PTB is conditional on MIH-BIT, the tunned-SMOTE-CNN model is used to classify the PTB. The model yielded an AUC of 76%. In future work, one may need to train a classifier for PTB separately or employ other optimization techniques to improve the result.

8 References

Alarsan, F.I. and Younes, M. (2019) 'Analysis and classification of heart diseases using heartbeat features and machine learning algorithms', *Journal of Big Data*, 6(1). Available at: https://doi.org/10.1186/s40537-019-0244-x.

Aziz, S., Ahmed, S. and Alouini, M.S. (2021) 'ECG-based machine-learning algorithms for heartbeat classification', *Scientific Reports*, 11(1), pp. 1–14. Available at: https://doi.org/10.1038/s41598-021-97118-5.

Chawla, N.C. *et al.* (2002) 'SMOTE: Synthetic Minority Over-sampling Technique Nitesh', *Journal of Artificial Intelligence Research*, 16, pp. 321–357.

Li, Z. *et al.* (2020) 'Heartbeat classification using deep residual convolutional neural network from 2-lead electrocardiogram', *Journal of electrocardiology*, 58, pp. 105–112. Available at: https://doi.org/10.1016/j.jelectrocard.2019.11.046.

Mathews, S.M., Kambhamettu, C. and Barner, K.E. (2018) 'A novel application of deep learning for single-lead ECG classification', *Computers in Biology and Medicine*, 99(May 2017), pp. 53–62. Available at: https://doi.org/10.1016/j.compbiomed.2018.05.013.

Romdhane, T.F. *et al.* (2020) 'Electrocardiogram heartbeat classification based on a deep convolutional neural network and focal loss', *Computers in Biology and Medicine*, 123(July), p. 103866. Available at: https://doi.org/10.1016/j.compbiomed.2020.103866.

Sannino, G. and De Pietro, G. (2018) 'A deep learning approach for ECG-based heartbeat classification for arrhythmia detection', *Future Generation Computer Systems*, 86, pp. 446–455. Available at: https://doi.org/10.1016/j.future.2018.03.057.

Xie, Q. *et al.* (2019) 'Feature enrichment based convolutional neural network for heartbeat classification from electrocardiogram', *IEEE Access*, 7, pp. 153751–153760. Available at: https://doi.org/10.1109/ACCESS.2019.2948857.