JOURNAL OF EMERGING INVESTIGATORS

Diagnosing Hypertrophic Cardiomyopathy Using Machine Learning Models on CMRs and EKGs of the Heart

Surya Kolluri[1], Sriram Hathwar[2]

[1] Dublin Jerome High School, Dublin, Ohio, United States
[2] Inspirit AI, San Francisco, California, United States

Affiliations of "unaffiliated", "mentor", or that only give general industry/position (i.e., tech sector) are NOT allowed. Authors may default to the student author's affiliation if needed.

Student Authors
Surya Kolluri: High School, 12th Grade

SUMMARY

Hypertrophic cardiomyopathy (HCM) is a common inherited heart disorder manifesting as hypertrophy of the left ventricle of the heart. However, it often goes undiagnosed, which we must seek to avoid since the possibility of sudden cardiac death (SCD) as a result of HCM is not insignificant. In this paper, we present a pair of models, one CNN model and one Long Short Term Memory (LSTM) model, that are capable of classifying cardiac magnetic resonance (CMR) and heart electrocardiogram (EKG) scans, respectively. Each of these models classifies their respective scans into HCM and non-HCM categories. The CNN model has an accuracy of 94.71%, a precision of 96.97%, a recall of 91.21%, and an F1 score of 94.85%. The LSTM model has an accuracy of 90.51%, a precision of 60.31%, a recall of 60.08%, and an F1 score of 60.19%. These results show that these machine learning models are viable tools that could assist physicians in the diagnosis of HCM patients.

INTRODUCTION

HCM affects 1 in 2000-2500 people in the American population, through which it could be inferred that around 750,000 Americans have this disease. However, other research suggests that the proportion stated above may only be 10-20% of all cases since the condition remains largely undiagnosed (1). Early diagnosis is a priority for patients because HCM can result in SCD, which is believed to be one of the leading causes of death in younger people (2). The majority of cases of HCM-related SCD occur in undiagnosed individuals, emphasizing the importance of early diagnosis. An Ontario study yielded a sample proportion of 0.31 definite HCM-related SCDs per person-years. Currently, diagnostic processes consist of genetic

testing, personal history (symptoms), physical examination, and family history combined with EKG, echocardiography, and CMR scans (1).

Artificial intelligence (AI) is the training of models to mimic intelligent behavior without significant human intervention. Through AI, drastic advancements in healthcare have been made, including early diagnosis of life-threatening diseases like cancer and healthcare data management (3). The FDA has also created its own separate procedure for handling AI-related medical technologies in order to better handle its specific challenges with training a model and ensuring its safety (4).

The purpose of this study is to develop and validate a pair of models to diagnose HCM based on either CMR images or EKG data to examine their accuracies, precisions, recalls, and F1 scores. The major results of this study show that the Keras CNN model performed with the highest accuracy, precision, recall, and F1 scores for the CMR scans and that the Keras-architecture LSTM model performed the best for the EKG data.

RESULTS

CMR Results

During testing, this study utilized several pre-made scikit-learn library models to classify the CMR scans because of their versatility and simplicity in data classification. Each model was trained on a randomized training set and was tested on a separate testing set. We assessed each of these models on the metrics of accuracy, precision, recall, and F1 score.

We also tested a Keras-architecture convolutional neural network and attempted to find the best layer combination, using techniques like Dropout and MaxPooling2D layers.

During testing, we found that this CNN Keras architecture model performed the best in three of the four metrics assessed (accuracy = 94.71%, precision = 96.97%, recall = 91.21%, F1 score = 94.85%, **Table 1**), with only the Random Forest scikit-learn model outperforming it in one metric with a precision of 98.79%.

EKG Results

At first, we used the same initial scikit-learn models as the CMR files on the EKG data. However, we found that these models were too simple to capture the differences between HCM positive and negative scans. Specifically, we found a low recall, indicating a high amount of false negatives. We switched to the Keras library and attempted to construct more complex models to better capture the positive cases.

Using Keras-architecture CNN and LSTM models, we found that the LSTM model performed the best (Table 2), because it was the only model with sufficient precision and recall. The others failed to have precision and recall reach over 60%.

DISCUSSION

The Keras CNN model classified CMR scans into HCM and non-HCM with a high accuracy, precision, recall, and F1 score. It outperformed all of the scikit-learn models in all categories except the Random Forest model's precision. This indicates that it could likely be used as a potential tool to assist physicians in diagnosing HCM, which could lead to earlier diagnosis and disease tracking.

The LSTM classified EKG scans into HCM and non-HCM with a high accuracy and precision, but lower recall and F1 score. This indicates that the model has a tendency to under-diagnose, and thus must be improved on predictions of positive values before it could be used clinically as an assistive tool.

Using the same scikit-learn models as the CMR scans for the EKG models did not yield similar results. The models all overgeneralized and ended up drastically underdiagnosing HCM, with recalls all below 40%. We switched to using Keras architecture LSTM and CNN models to try to improve recall, which was the lowest value because of the large amount of false negative predictions. The LSTM resulted in the best metrics.

The novelty of this study is the pairing of these two models. In a typical diagnostic process, a doctor uses several scans and risk factors to determine whether or not a patient has a risk of HCM. This pair of models utilize multiple modalities in order to better assist a doctor in diagnosis.

These models are limited by the fact that they were only trained on one dataset each. The Hypertrophic Cardiomyopathy Dataset was collected in Iran, and thus the CNN might be influenced in decisions because of commonalities between people of the same region. This could result in potential incorrect classifications on scans from outside Iran. In addition, the PTB-XL dataset LSTM model is subject to the same possible issues on scans of people from outside of Germany.

In addition, the CMR CNN has an additional potential handicap. If it receives CMR images from angles it has not trained on, it might also misclassify the image. In the future, this research could be expanded on by increasing the size of the dataset, diversifying the group of patients scanned, utilizing scans from different angles, and using higher frequency EKG to capture smaller patterns.

The CMR CNN holds somewhat more promise than the EKG model because of its generally higher performance in all assessed performance metrics.

MATERIALS AND METHODS

This study utilized Kaggle, an online data science community with access to a variety of machine learning datasets, to collect both the CMR and EKG data for these models. The CMR data was from the Hypertrophic Cardiomyopathy Dataset, collected at Omid Hospital in Tehran, Iran (Figure 1). It consisted of two directories, Sick and Healthy, which held several subdirectories with the CMR .jpg files. In total, there were 37241 healthy and 21846 HCM scans. The PTB-XL ECG dataset was collected at the Physikalisch-Technische Bundesanstalt (PTB), a research institute in Germany. The dataset consists of directories of .dat and .hea files holding 10-second-long EKG records of 18885 patients sampled at a frequency of 100 Hz, and additionally, versions sampled at 500 Hz (Figure 2). Both of these datasets were imported into a Python-language notebook in Google Colaboratory, or Google Colab, for short.

Google Colab and Python Libraries:

Google Colab is a free cloud-based interactive programming notebook software similar to a Jupyter Notebook. It allows for the separation of code cells, allowing for running separate sections of code out of order, when normally, Python code would just run in order. This also gives the option to correct typos or small errors without rerunning the whole code at once. It also has several useful Python libraries preinstalled for machine learning, such as os, pandas, NumPy, matplotlib, scikit-learn, and joblib. In addition, it allows for the installation of other libraries needed for certain data and models, like cv2, wfdb, keras, and Streamlit. It also comes with limited free access to GPU and TPU hardware accelerators.

The os library allows for interaction with the computer's operating system. This allows the user to change directories and manage folders and files. It allows the user to learn more about the software environment as well (7). The main application of the pandas library includes useful structures like DataFrames to organize table data, which is useful when handling any data stored in CSV files. Pandas DataFrames can also be directly trained on by models (8).

We also utilized NumPy, a fundamental Python library used for its unique array data type and mathematical functions for processing said arrays (9), and matplotlib, a data visualization library that can create data plots. The latter is useful for generating graphics of data and analyzing model performance for hyperparameter tuning (10).

Cv2, or OpenCV, is a computer vision (CV) library used to filter, segment, edit, and recolor images that are being trained on for computer vision models. This allows the user to

eliminate any distractions for the model so that it only focuses on the specific images that the user wishes it to. It can also resize images so that they all fit into the proper dimensions for a model to analyze them properly (11).

WFDB is a library used to interpret .dat and .hea files that hold the EKG information. By applying certain functions, it can be used to view the actual EKG leads and waves, which helps for a better understanding of the data (12).

Keras is a Tensorflow-based library used for creating machine learning models. It is capable of creating custom CNNs, recurrent neural networks (RNNs), LSTMs, U-Net models, and more. It has various features and hyperparameters that can be edited to improve model performances based on the specific aspects of certain datasets (13).

Joblib is a library used for model storage. Using it, one can store models and all of their assigned weights (14).

CMR Dataset Analysis

Upon examining each of the images with matplotlib and the in-built .shape function, we realized that the images were a variety of pixel sizes. To ensure that the model could work efficiently, we grouped all images of the same size together in Python lists. Lists that had similar numbers of images with the same size for both Diseased and Healthy patients were the only images chosen for training. One image size was chosen, and all of the images in the selected lists were converted to the same size.

Each of the lists was converted to a NumPy array, and had a corresponding NumPy array column of labels created (0: healthy, 1: diseased). All image data arrays were added together using the np.vstack function to make an X (training images) array, and all label arrays were added together to make a y (label) array. Once the arrays were complete, the train_test_split function from scikit-learn was used to split the data into a randomized 80%-20% split of training and testing data in the variables X_train, X_test, y_train, and y_test.
Each individual testing model was fit to the X_train and y_train variables, which contained all training data. Each model then predicted the labels of X_test based on their training. Each model was assessed on the accuracy, precision, recall, and F1 score (Table 1). Precision is the percentage of scans correctly classified out of all scans classified as HCM positive by the models. Recall is the percentage of scans correctly classified out of all actual HCM positive scans. F1 score is a combination of both.

The exact architecture of the model (Figure 3) was a two iteration for loop that contained three Conv2D layers with 64, 32, and 16 units, all with ReLu activations and kernel sizes of

(3,3), followed by a MaxPooling2D layer with a pool_size of (2,2) within the loop. After the loop were three Dense layers with 64, 16, and 4 units each, with ReLu activations and l1(0.0001) kernel regularizers. Each Dense layer had a layer of Dropout following it, with values of 0.2, 0.2, and 0.1 respectively. Following the last Dropout layer was a final Dense layer with 1 unit, a kernel regularizer of l1(0.0001), and a sigmoid activation function to produce a value between 0 and 1. The CNN was compiled with binary_crossentropy as the loss, adam as the optimizer, and accuracy as the metrics. It was fitted to X_train, y_train, a batch size of 200, set shuffle = true, and was validated using X_test and y_test. It was also tested on the four previously mentioned metrics (Table 1), with a full confusion matrix in Table 3.

EKG Dataset Analysis

The creators of the PTB-XL dataset created a recommended data split for training and testing data. It divided the data into a 90% training and 10% testing split. This recommended split was used for the model. In addition, the label set was modified. Originally, each EKG had a list of specific diagnoses as the label. This was simplified to an 'HYP' label for the EKGs with an HCM diagnosis and non-HYP for the EKGs without.

Originally, the same scikit-learn models that were used for the CMR scans were used, but with poor results. Each model tended to under-diagnose HCM, resulting in an excess of false negatives and, consequently, lower accuracy and recall. In addition, these models also had precision below 50%. Because of the lack of results, we switched to Keras CNN, RNN, and LSTM models. We found that LSTMs could generally classify the best out of the three, so we prioritized its improvement.

The full structure of the LSTM model (Figure 4) was five Bidirectional LSTM layers with 64, 32, 16, 32, and 64 units each, with a layer of Dropout in between each. Each Dropout layer had 0.2 except for the last, with 0.15. Following the LSTM layers was a GlobalMaxPooling1D layer, followed by three Dense layers with 32, 16, and 4 units each. Each Dense layer also had ReLu activations and a l2 kernel regularizer value of 0.0001. We then put a Batch Normalization layer and a final 1 unit Dense layer with sigmoid activation. We then rounded each prediction to either 0 or 1 to assess accuracy.

Each of the scikit-learn and Keras models were assessed on the accuracy, precision, recall, and F1 score (Table 2), with a full confusion matrix of the top performing model in Table 4.

ACKNOWLEDGMENTS (Optional)

REFERENCES

1. Maron, Barry J., et al. “Diagnosis and Evaluation of Hypertrophic Cardiomyopathy: JACC State-of-the-Art Review.” Journal of the American College of Cardiology, 24 Jan. 2022, www.sciencedirect.com/science/article/pii/S0735109721082735#abs0010.

2. Weissler-Snir, Adaya, et al. *Hypertrophic Cardiomyopathy–Related Sudden Cardiac Death in Young ...*, 21 Oct. 2019, www.ahajournals.org/doi/10.1161/CIRCULATIONAHA.119.040271.

3. Hamet, Pavel, and Johanne Tremblay. “Artificial Intelligence in Medicine.” *Metabolism*, 11 Jan. 2017, www.sciencedirect.com/science/article/abs/pii/S002604951730015X.

4. Center for Devices and Radiological Health. “Artificial Intelligence and Machine Learning in Software.” *U.S. Food and Drug Administration*, www.fda.gov/medical-devices/software-medical-device-samd/artificial-intelligence-and-machine-learning-software-medical-device. Accessed 29 July 2023.

5. Sharifrazi, Danial. “Hypertrophic Cardiomyopathy Dataset.” *Kaggle*, 4 Oct. 2021, www.kaggle.com/datasets/danialsharifrazi/hypertrophic-cardiomyopathy-dataset.

6. “PTB-XL ECG Dataset.” *Kaggle*, www.kaggle.com/datasets/khyeh0719/ptb-xl-dataset. Accessed 29 July 2023.

7. “OS - Miscellaneous Operating System Interfaces.” *Python Documentation*, docs.python.org/3/library/os.html. Accessed 29 July 2023.

8. “User Guide.” *User Guide - Pandas 2.0.3 Documentation*, pandas.pydata.org/docs/user_guide/index.html. Accessed 29 July 2023.

9. “Numpy User Guide.” *NumPy User Guide - NumPy v1.25 Manual*, numpy.org/doc/1.25/user/index.html#user. Accessed 29 July 2023.

10. “Users Guide.” *Users Guide - Matplotlib 3.7.2 Documentation*, matplotlib.org/stable/users/index.html. Accessed 29 July 2023.

11. “Introduction.” *OpenCV*, docs.opencv.org/3.4/d1/dfb/intro.html. Accessed 29 July 2023.

12. “WFDB.” *Wfdb*, wfdb.readthedocs.io/en/latest/. Accessed 29 July 2023.

13. Team, Keras. “Keras Documentation: Developer Guides.” *Keras*, keras.io/guides/. Accessed 29 July 2023.

14. “Running Python Functions as Pipeline Jobs.” *Joblib*, joblib.readthedocs.io/en/stable/. Accessed 29 July 2023.

Figures and Figure Captions

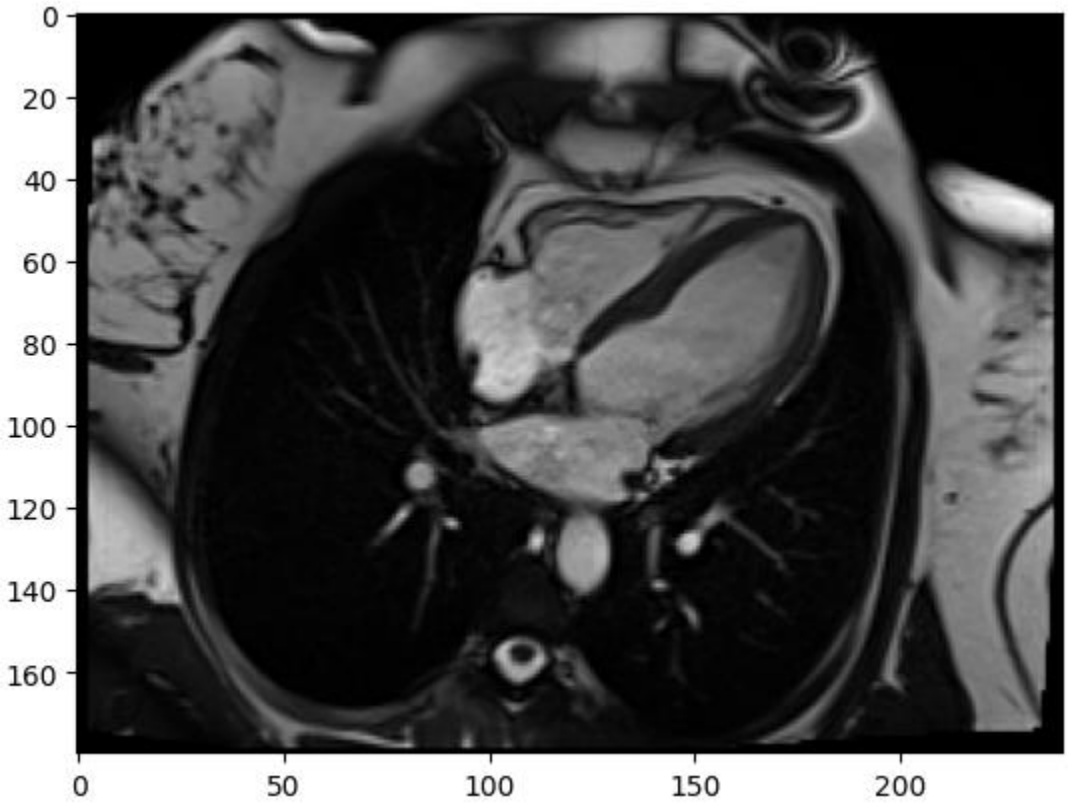

Figure 1. CMR scan from the Omid Hospital Hypertrophic Cardiomyopathy Dataset. .jpg files visualized with the cv2 and matplotlib libraries.

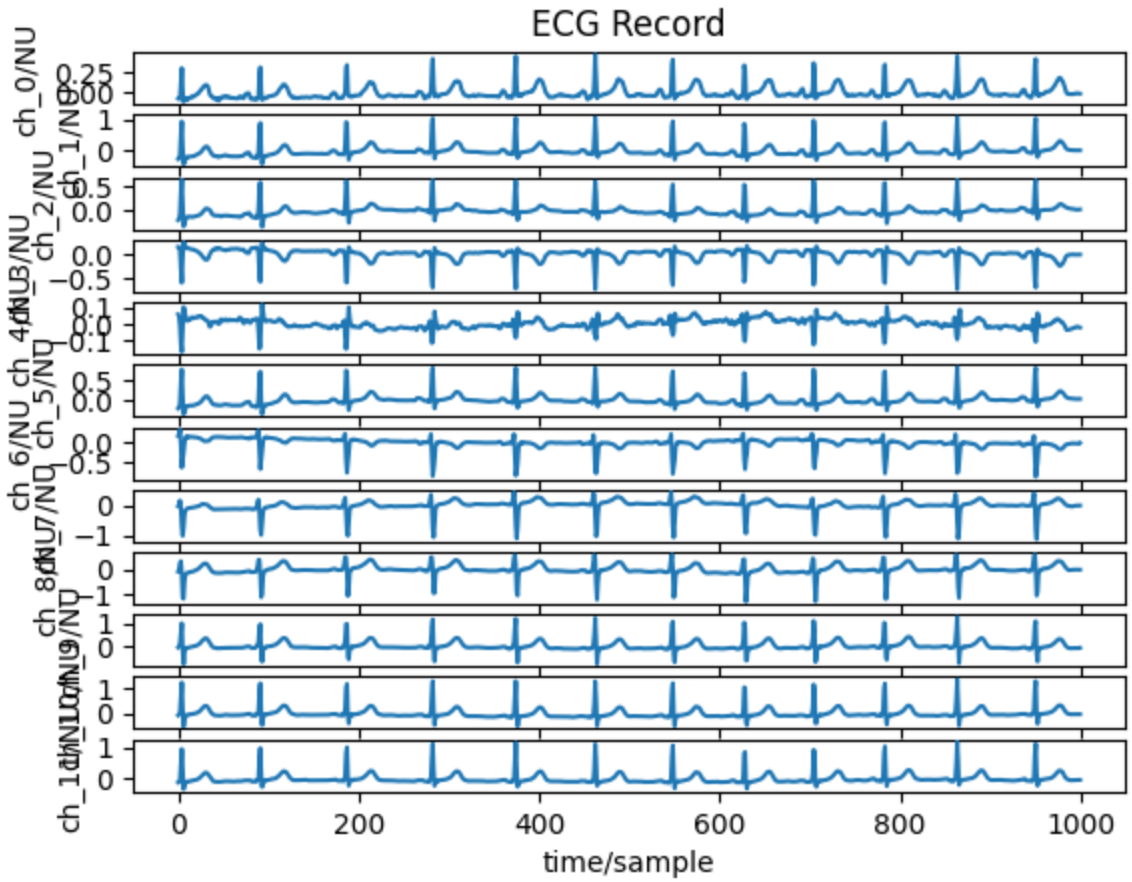

Figure 2. Example EKG scan obtained through WFDB library. 12 leads are depicted as waves, each with 1000 samples taken within a 10 second interval, or at a frequency of 100 Hz.

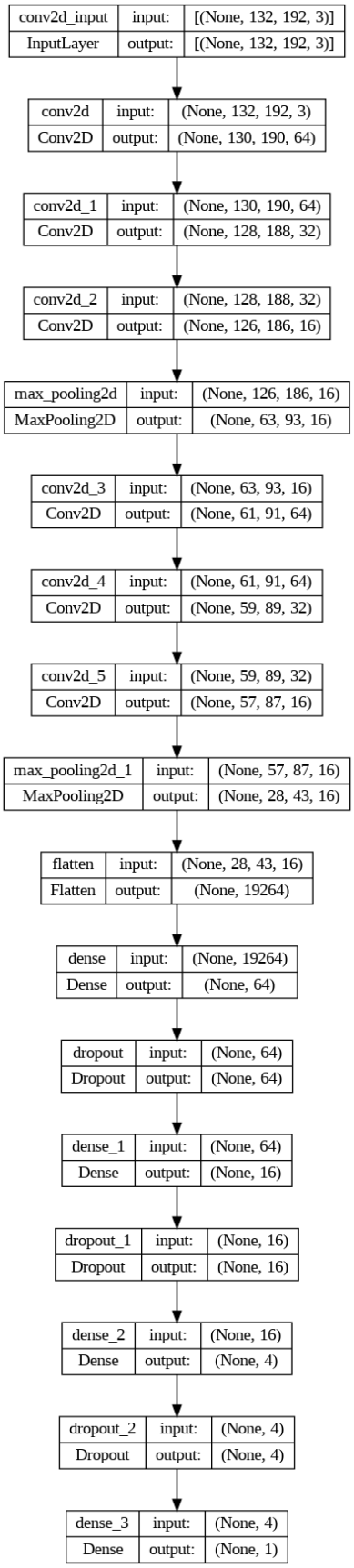
conv2d_input | input: | [(None, 132, 192, 3)]
InputLayer | output: | [(None, 132, 192, 3)]
conv2d | input: | (None, 132, 192, 3)
Conv2D | output: | (None, 130, 190, 64)
conv2d_1 | input: | (None, 130, 190, 64)
Conv2D | output: | (None, 128, 188, 32)
conv2d_2 | input: | (None, 128, 188, 32)
Conv2D | output: | (None, 126, 186, 16)
max_pooling2d | input: | (None, 126, 186, 16)
MaxPooling2D | output: | (None, 63, 93, 16)
conv2d_3 | input: | (None, 63, 93, 16)
Conv2D | output: | (None, 61, 91, 64)
conv2d_4 | input: | (None, 61, 91, 64)
Conv2D | output: | (None, 59, 89, 32)
conv2d_5 | input: | (None, 59, 89, 32)
Conv2D | output: | (None, 57, 87, 16)
max_pooling2d_1 | input: | (None, 57, 87, 16)
MaxPooling2D | output: | (None, 28, 43, 16)
flatten | input: | (None, 28, 43, 16)
Flatten | output: | (None, 19264)
dense | input: | (None, 19264)
Dense | output: | (None, 64)
dropout | input: | (None, 64)
Dropout | output: | (None, 64)
dense_1 | input: | (None, 64)
Dense | output: | (None, 16)
dropout_1 | input: | (None, 16)
Dropout | output: | (None, 16)
dense_2 | input: | (None, 16)
Dense | output: | (None, 4)
dropout_2 | input: | (None, 4)
Dropout | output: | (None, 4)
dense_3 | input: | (None, 4)
Dense | output: | (None, 1)

Figure 3. Structure of CMR CNN Model. Shows layer type and input and output sizes of each layer, with three Conv2D layers, followed by a MaxPooling2D layer, followed by three more Conv2D layers, another MaxPooling layer, a Flatten layer, and then four Dense layers separated by Dropout layers.

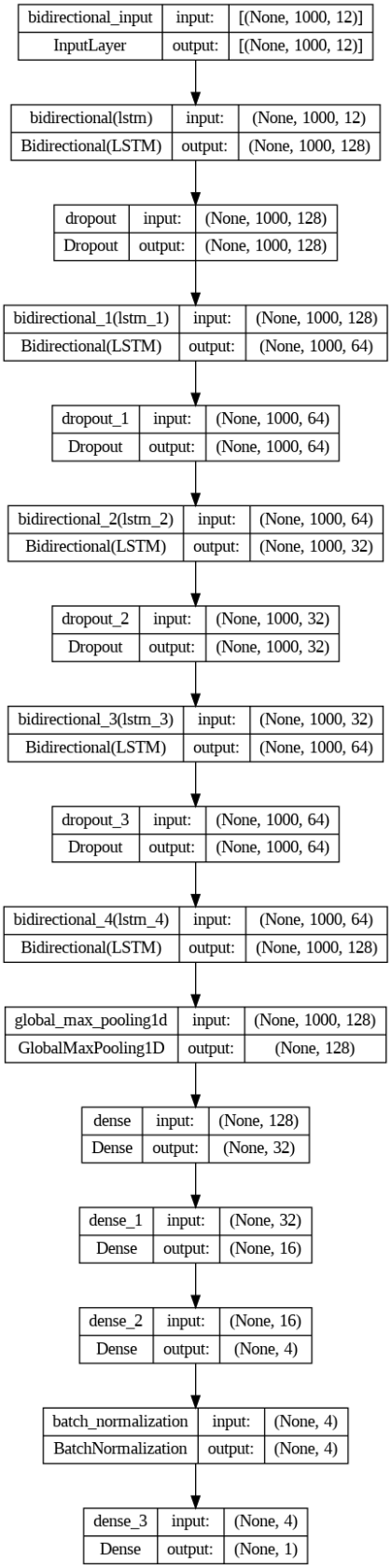
bidirectional_input | input: | [(None, 1000, 12)]
InputLayer | output: | [(None, 1000, 12)]
bidirectional(lstm) | input: | (None, 1000, 12)
Bidirectional(LSTM) | output: | (None, 1000, 128)
dropout | input: | (None, 1000, 128)
Dropout | output: | (None, 1000, 128)
bidirectional_1(lstm_1) | input: | (None, 1000, 128)
Bidirectional(LSTM) | output: | (None, 1000, 64)
dropout_1 | input: | (None, 1000, 64)
Dropout | output: | (None, 1000, 64)
bidirectional_2(lstm_2) | input: | (None, 1000, 64)
Bidirectional(LSTM) | output: | (None, 1000, 32)
dropout_2 | input: | (None, 1000, 32)
Dropout | output: | (None, 1000, 32)
bidirectional_3(lstm_3) | input: | (None, 1000, 32)
Bidirectional(LSTM) | output: | (None, 1000, 64)
dropout_3 | input: | (None, 1000, 64)
Dropout | output: | (None, 1000, 64)
bidirectional_4(lstm_4) | input: | (None, 1000, 64)
Bidirectional(LSTM) | output: | (None, 1000, 128)
global_max_pooling1d | input: | (None, 1000, 128)
GlobalMaxPooling1D | output: | (None, 128)
dense | input: | (None, 128)
Dense | output: | (None, 32)
dense_1 | input: | (None, 32)
Dense | output: | (None, 16)
dense_2 | input: | (None, 16)
Dense | output: | (None, 4)
batch_normalization | input: | (None, 4)
BatchNormalization | output: | (None, 4)
dense_3 | input: | (None, 4)
Dense | output: | (None, 1)

Figure 4. EKG LSTM model structure. Shows structure and input and output sizes for each layer. Consists of 5 Bidirectional LSTM layers with Dropout in between each, followed by a GlobalMaxPooling1D layer, followed by three Dense layers, a Batch Normalization layer, and a final Dense layer with one unit.

Tables with Captions

Model	Accuracy	Precision	Recall	F1 Score
RF	94.71%	98.79%	91.21%	94.85%
Logreg	88.13%	89.27%	87.02%	88.13%
Ridgeclass	88.56%	90.66%	86.75%	88.66%
Decision Tree	90.09%	91.35%	88.89%	90.10%
SVM	91.03%	96.54%	86.78%	91.40%
CNN	97.01%	96.97%	97.13%	97.05%

Table 1. CMR accuracy metrics for different scikit-learn models. The listed models are random forest (RF), logistic regression (Logreg), ridge classification (Ridgeclass), support vector model (SVM), and convolutional neural network (CNN).

Model	Accuracy	Precision	Recall	F1 Score
RF	87.97%	46.88%	5.70%	10.17%
Logreg	86.70%	27.27%	6.84%	10.94%
Ridgeclass	84.88%	23.48%	11.79%	15.70%
Decision Tree	82.61%	27.61%	28.14%	27.87%
LSTM	90.51%	60.31%	60.08%	60.19%
Wavelet Transformed LSTM	91.10%	74.10%	39.16%	51.24%

Model	Accuracy	Precision	Recall	F1 Score
RF	87.97%	46.88%	5.70%	10.17%
Logreg	86.70%	27.27%	6.84%	10.94%
Ridgeclass	84.88%	23.48%	11.79%	15.70%
Decision Tree	82.61%	27.61%	28.14%	27.87%
LSTM	90.51%	60.31%	60.08%	60.19%
CNN	90.24%	87.50%	21.29%	34.25%

Table 2. EKG accuracy metrics for different scikit-learn models. The listed models are random forest (RF), logistic regression (Logreg), ridge classification (Ridgeclass), decision tree, and long short term memory network (LSTM).

TP = 576	FP = 18
FN = 17	TN = 560

Table 3. Confusion matrix for CMR CNN model. TP = true positives, FP = false positives, FN = false negatives, and TN = true negatives.

TP = 158	FP = 104
FN = 105	TN = 1836

Table 4. Confusion matrix for EKG LSTM model. TP = true positives, FP = false positives, FN = false negatives, and TN = true negatives.

Appendix (If applicable)

https://github.com/suryakolluri6/HCM_diagnostic_models/tree/main