JOURNAL OF
EMERGING INVESTIGATORS

Article

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
[2] Inspirit AI, San Francisco, California

SUMMARY

Hypertrophic cardiomyopathy (HCM) is a common inherited heart disorder manifesting as hypertrophy of the left ventricle of the heart that often goes undiagnosed. It is imperative that hypertrophic cardiomyopathy is diagnosed early since there is a possibility of sudden cardiac death (SCD) as a result of HCM. In this study, we created a pair of models, one convolutional neural network (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 classified their respective scans into HCM and non-HCM categories. The CNN model had 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 had an accuracy of 90.51%, a precision of 60.31%, a recall of 60.08%, and an F1 score of 60.19%. These results showed that these machine learning models are viable tools that could assist physicians in the diagnosis of HCM patients.

INTRODUCTION

Hypertrophic cardiomyopathy (HCM) affects 1 in 2,000–2,500 people in the American population, translating to around 750,000 individuals (1). However, other research suggests that this proportion may only be 10–20% of all cases since the condition remains largely undiagnosed. Early diagnosis is a priority for patients because HCM can result in sudden cardiac death (SCD), which is one of the leading causes of death in people between 10 and 45 years of age (2). The majority of cases of HCM-related SCD occur in undiagnosed individuals, emphasizing the importance of early diagnosis. A study of the Office of the Chief Coroner of Ontario's database yielded a sample proportion of 0.31 definite HCM-related SCDs per 1,000 person-years (2). Person-years is a medical term that describes a study where the number of participants is multiplied by the number of years the study took place over to calculate person-years. For example, a 10-year-long study involving 100 participants would be 1,000 person-years. For reference, the prevalence of breast cancer has been calculated as 176 cases per 100,000 person-years, or 1.76 cases per 1000 person-years (3). Given that this information only compares the incidence of breast cancer with sudden cardiac deaths from HCM, 0.31 definite HCM-related deaths per 1,000 person-years is still a significantly high number. Currently, diagnostic processes consist of genetic testing, personal history (symptoms), physical examination, and family history combined with electrocardiogram (EKG), echocardiography, and cardiac magnetic resonance (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 earlier diagnosis of life-threatening diseases like cancer. It has also been used for healthcare data management (4). The Food and Drug Administration (FDA) agency has also created its own separate procedure for handling AI-related medical technologies to better handle the specific challenges with training a model and ensuring its safety (5).

One type of AI model often involved in healthcare is the convolutional neural network (CNN). CNNs have been used in models that diagnose heart disease, breast cancer, diabetes, and Parkinson's disease (8). One main benefit of implementing a CNN is its ability to find intricate patterns in images and automatically learn to identify features and structures. CNNs often consist of several layers, including an input layer, convolutional layers, pooling layers, and fully connected (Dense) layers. The input layer simply takes in an image, and then the convolutional layers scan over it, performing calculations called convolutions to identify edges, curves, structures, and other patterns in images. The pooling layers simplify the data, retaining essential information while reducing the amount of space used. Finally, the Dense layers combine all of the detected features to make a prediction on what category to classify the image into.

Another common type of model used for analyzing medical data is a long short-term memory (LSTM) network since it is a type of recurrent neural network (RNN) specifically designed to handle time-series (sequential) data. It is used to predict future outcomes based on previous patterns. Examples of common uses include the stock market and weather forecasting. LSTMs have also been used for electroencephalography (EEG) scans to recognize epilepsy (9). Since EKGs are also sequential data, in which 12 leads are used to measure the electrical activity of the heart over a specific amount of time with multiple beats and waveforms, we decided that an LSTM network would be appropriate. An LSTM network consists of multiple LSTM cells that are arranged into layers. Each layer of the LSTM processes the data in sequential order, generating an output for each individual timestep (the interval at which a signal is recorded). The outputs are then sent down to the next layers. After the LSTM layers, we used a global max pooling layer to simplify the data, similarly to the CNN model, followed by Dense layers used to produce a final classification. Using a bidirectional LSTM simply allows a model to look at both past and future timesteps in order to better understand their relationship.

In addition, each dataset was tested with standard scikit-

https://doi.org/10.59720/23-209

learn classification models for comparison, including random forest, logistic regression, decision tree, ridge classification, and support vector machines **(Table 1)**.

The objective of this study was to compare the ability of AI models trained on CMR and EKG data to diagnose HCM. We evaluated several scikit-learn classification models, a CNN, and a LSTM for accuracy, precision, recall, and F1 score (a machine learning metric to measure a model's accuracy, which is a combination of a model's precision and recall scores). We hypothesized that the CNN would perform best with the CMR data, and the LSTM would perform best with the EKG data. The Keras CNN model performed best across all metrics with the highest accuracy, precision, recall, and F1 scores for the CMR scans. We also report that the Keras-architecture Bidirectional Long Short-Term Memory (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 (80% of the dataset) and was tested on a separate testing set (the remaining 20%). The CMR data was from the Hypertrophic Cardiomyopathy Dataset, collected at Omid Hospital in Tehran, Iran (6) **(Figure 1)**. The PTB-XL ECG (ECG and EKG are interchangeable and use differs based on country) dataset was collected at the Physikalisch-Technische Bundesanstalt (PTB), a research institute in Germany (7) **(Figure 2)**.

We assessed each of these models on the metrics of accuracy, precision, recall, and F1 score.

We also tested a Keras-architecture CNN and attempted to find the best layer combination.

During testing, we assessed the model on four metrics: accuracy, precision, recall, and F1 score. 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. We determined that the 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 [a machine learning metric to measure a model's accuracy, which is a combination of a model's precision and recall scores] = 94.85%), with only the Random Forest scikit-learn model outperforming it in one metric with a precision of 98.79% (**Table 2**). We also generated a confusion matrix

RF: The Random Forest algorithm, developed by Leo Breiman and Adele Cutler, is widely employed in machine learning. It merges the outcomes of numerous decision trees to produce a unified outcome. Its widespread acceptance stems from its user-friendliness and adaptability, making it proficient in tackling classification as well as regression tasks.
Logreg: Logistic regression predicts the likelihood of an event happening using a dataset of independent variables. Commonly known as a logit model, this statistical approach finds utility in classification and predictive analytics tasks. Due to its nature, where outcomes are probabilities, the dependent variable is constrained within the range of 0 to 1. Logistic regression employs a logit transformation to handle odds, representing the ratio of success probability to failure probability.
Ridgeclass: This classifier transforms the target values into {-1, 1}, subsequently approaching the problem as a regression task (in the multiclass scenario, it's termed multi-output regression).
Decision Tree: Decision Trees (DTs) are a versatile method used in supervised learning for classification and regression. They aim to predict the target variable by learning straightforward decision rules from the data features. Essentially, a tree acts as a way to break down the data into simpler, consistent segments.
SVM: Support Vector Machines (SVMs) are versatile tools for tasks like classification, regression, and outlier detection. They work well in high-dimensional spaces and are memory-efficient. However, choosing the right parameters is important to prevent overfitting when there are more features than samples. SVMs don't directly give probability estimates, so calculating them can be computationally expensive.
Wavelet Transformed LSTM: The wavelet transform is a valuable computational tool widely applied in various signal and image processing tasks. It is notably useful for compressing digital images, which helps save memory space and facilitates faster, more reliable image transmission. Various organizations, including the FBI and NASA, have employed wavelet transforms for compressing images for different purposes. Additionally, wavelet-based algorithms are utilized in software for deep-space communication requirements. Furthermore, wavelet transforms aid in refining signals and images by reducing noise and blurring effects. Many algorithms used in processing astronomical images are based on wavelet or similar transformations. By adding this transformation to an LSTM model, we attempted to see if we could improve in any performance metrics.

Table 1: Description of alternative models utilized in study. All models were from the Python scikit-learn library, except for the wavelet-transformed LSTM, in which the pywt wavelet transformation library was used on a Keras LSTM.

https://doi.org/10.59720/23-209

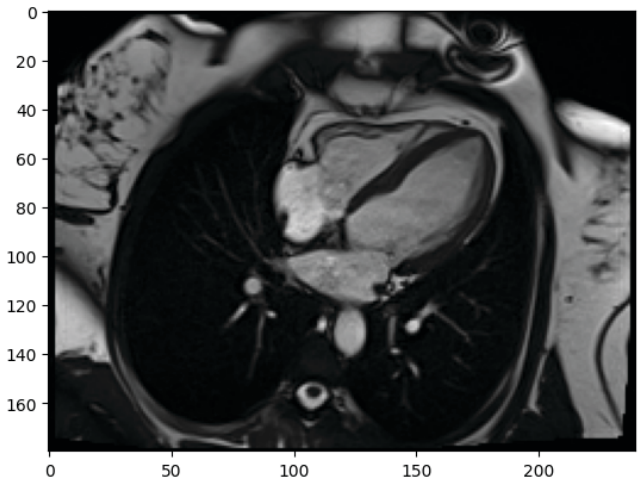

Figure 1: CMR scan from the Omid Hospital Hypertrophic Cardiomyopathy Dataset (5). Example of one of the many JPEG (.jpg) files of CMR scans visualized with the Cv2 and matplotlib libraries. The CNN model trained on these by scanning over segments of it to identify patterns in heart structure. The model was trained on 80% of these images and tested on the remaining 20%.

for the model, which is a table that shows the quantity of true positive (TP), false positive (FP), false negative (FN), and true negative (TN), values that the model assigned **(Table 3)**. This table is another way of assessing the performance of the model. Since recall = TP/(TP + FN), a greater number of FN values will decrease the recall metric. Since precision = TP/(TP + FP), a greater number of FP values will decrease the precision metric.

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 number 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 Bidirectional LSTM model performed the best because it was the only model with sufficient precision and recall **(Table 4)**. The others failed to have precision and recall reach over 60%. Our goal was to use a model that had both values reach over at least 60% to limit false positives and false

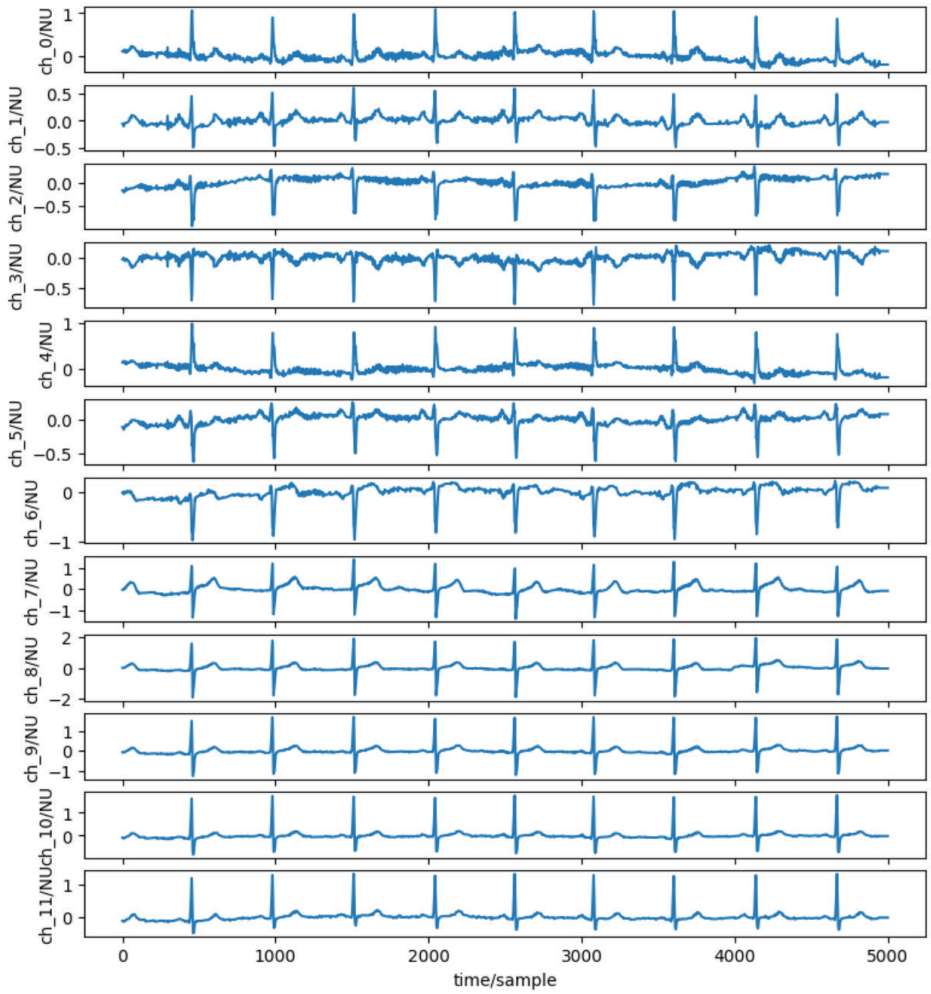

Figure 2: Example EKG scan from the PTB-XL Dataset obtained through WFDB library (6). Leads are depicted as waves (n = 12), each with 1000 samples taken within a 10 second interval (*i.e.,* frequency of 100 Hz). Each row corresponds to a lead, where ch_0/NU is the first lead, and ch_11/NU is the twelfth and final lead. The model once again trained on 80% of the data and tested on 20% of the data.

https://doi.org/10.59720/23-209

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

CNN Model Confusion Matrix		LSTM Model Confusion Matrix	
TP = 576	FP = 18	TP = 158	FP = 104
FN = 17	TN = 560	FN = 105	TN = 1836

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

negatives. Similarly to the CNN model, we also generated a confusion matrix to assess quantities of TP, FP, FN, and TN values.

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 in precision. The CNN having a slightly lower precision means that it can produce slightly more FP values than the random forest model, but its substantially higher recall indicates a lower propensity for FN values. This indicates that it could be used as a tool to assist physicians in diagnosing HCM, which could lead to earlier diagnosis and disease tracking.

The Bidirectional 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 tends to under-diagnose, and thus must be improved on predictions of positive values before it could be used clinically as an assistive tool. Potential methods for improvement include different layer structures, further experimentation with dropout, and different pooling strategies.

Using the same scikit-learn models as the CMR scans for the EKG models did not yield similar results. The models all overgeneralized and drastically underdiagnosed HCM, with recalls all below 40%. We hypothesized that the models overgeneralized because they were not able to capture precise enough patterns to diagnose HCM, and thus tended

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%
Bidirectional LSTM	90.51%	60.31%	60.08%	60.19%
Wavelet Transformed LSTM	91.10%	74.10%	39.16%	51.24%
CNN	90.24%	87.50%	21.29%	34.25%

Table 4: EKG accuracy metrics for different scikit-learn models. The listed models are random forest (RF), logistic regression (Logreg), ridge classification (Ridgeclass), decision tree, support vector machine (SVM), bidirectional long, short term memory network (LSTM), Wavelet Transformed LSTM, and convolutional neural network (CNN).

https://doi.org/10.59720/23-209

to not diagnose HCM when left up to chance. 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 Bidirectional 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 analyzes several scans and risk factors manually to determine if a patient has a risk of HCM. This pair of models utilizes 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 Bidirectional LSTM model is subject to the same possible issues on scans of people from outside of Germany. The CMR CNN has another potential weakness. 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. Both models show promise for eventual use in a diagnostic process for HCM, which would help save many lives through early diagnosis and decrease the cost of the diagnostic process.

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. The CMR dataset was the Hypertrophic Cardiomyopathy Dataset from the Omid Hospital in Tehran, Iran (6) **(Figure 1)**. The EKG dataset was the PTB-XL ECG Dataset from the Physikalisch Technische Bundesanstalt in Braunschweig, Germany (7) **(Figure 2)**. The CMR dataset consisted of two directories, Sick and Healthy, which held several subdirectories with the CMR .jpg files. In total, there were 37,241 healthy and 21,846 HCM scans. The EKG dataset consisted of directories of .dat and .hea files holding 10-second-long EKG records of 18,885 patients sampled at frequencies of 100 Hz and 500 Hz **(Figure 2)**. These were obtained using the WFDB library. Both of these datasets were imported into a Python-language notebook in Google Colaboratory (Colab) and analyzed using the os, NumPy, Cv2, WFDB, and Keras libraries.

Python Libraries

During analysis, we used the os library to navigate the directory format of the dataset, which allowed us to access individual files from the dataset for viewing and further modification (10). We also made use of the pandas library in order to read organizational CSVs and construct a DataFrame to organize all of the EKG data (11).

NumPy, a mathematics and array creation library, was utilized in analyzing the data (12). Matplotlib, a data visualization library was used to generate graphics of data and analyze model performance for hyperparameter tuning (13).

Cv2, or OpenCV was a computer vision library used to resize and recolor CMR images that the model was trained on. (14). WFDB was used to interpret .dat and .hea files that hold the EKG information. By applying certain functions, the library can be used to view the actual EKG leads and waves, which helps for a better understanding of the data (15).

Keras is a Tensorflow-based library used for creating machine learning models, and it was used to construct both the CNN and LSTM models (16).

CMR Dataset Analysis

Upon examining each of the images with using the Cv2 library and matplotlib libraries and the in-built .shape function, it was observed that the images were a variety of pixel sizes. To ensure that the model could work efficiently, all images of the same size were grouped together in Python lists. Only the lists that had similar numbers of images with the same size for both Diseased and Healthy patients were chosen for training. Once our lists were chosen, they were converted to a single standard size of (132, 192, 3), which was chosen both to maintain the shape of as many images as possible and to prevent the images from becoming too complex (or rather have too many pixels for the model to efficiently run through).

Each of the lists was converted to a NumPy array and NumPy array column of labels were 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 2)**.

The exact architecture of the Keras CNN model 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 **(Figure 3)**. 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. The model 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, and a full confusion matrix was evaluated **(Tables 2, 3)**.

The additional models (scikit-learn random forest, logistic regression, ridge classifier, decision tree, and support vector machine models) were also trained on the data with the same training split. Confusion matrices and other performance metrics were also generated for each of the scikit-learn models after they were trained and tested on the CMR data. The study compared each of the models based on overall accuracy, precision, and recall. When determining the best-

https://doi.org/10.59720/23-209

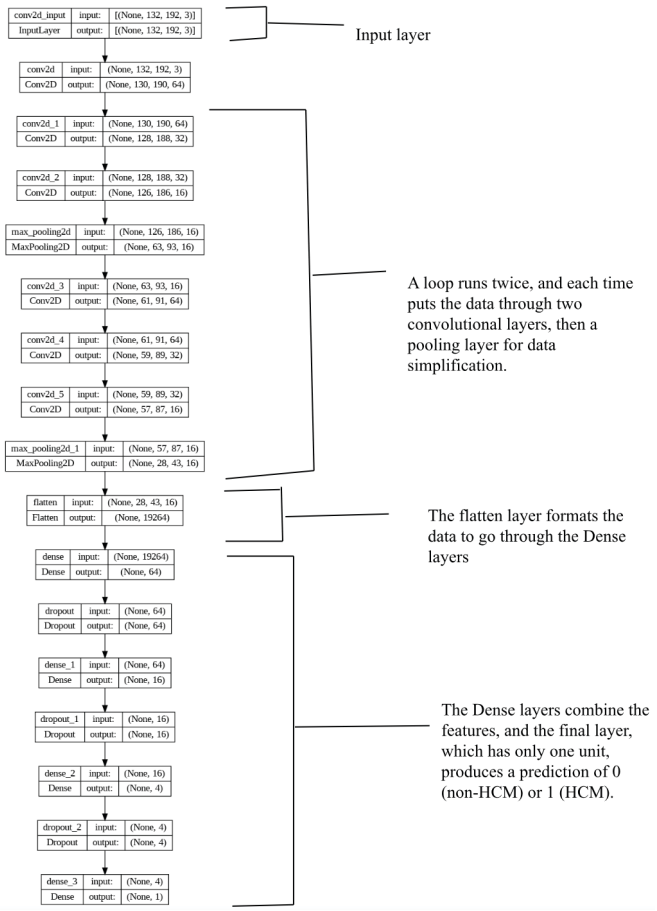

Figure 3: Structure of CMR CNN Model. Diagram describing layer type and input and output sizes of each layer, with three Conv2D layers, followed by a MaxPooling2D layer, followed by three more Conv2D layers, another MaxPooling layer, a Flatten layer, and then four Dense layers separated by Dropout layers. The model trained on 80% of the CMR models and was tested on the remaining 20%.

performing model, we wanted the model that had the best overall metrics between them. For example, the random forest model had a slightly higher overall precision than the CNN (about 2% more), but the recall was significantly lower than in the CNN (6%) **(Table 2)**.

EKG Dataset Analysis

The creators of the PTB-XL dataset created a recommended data split for training and testing data, dividing the data into a 90% training and 10% testing split. This recommended split was used for our model. Originally, each EKG had a list of specific diagnoses as the label. We modified this labeling system by simplifying to a "HYP" label for the EKGs scans diagnosed with HCM, and "non-HYP" for the EKGs without HCM.

Originally, the same scikit-learn models used for the CMR scans were used for EKG analysis, except for the SVM. The SVM was left out of the second experiment because of its longer runtime, tendency to crash the Colab notebook, and relative inefficiency compared to the more successful models. 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 poor performance of these models, Keras CNN, RNN, and LSTM models were used instead. We found that LSTMs could generally classify the best out of the three, so we prioritized its improvement over the other methodologies. We attempted to use different types of LSTMS, including a Bidirectional LSTM, and a wavelet transformation using the pywt library (17). We applied a discrete wavelet transformation to each data sample to produce an approximation coefficient matrix, which the model was trained on. The wavelet transformation was used to try to eliminate noise from the LSTM, but still underperformed compared to the Bidirectional LSTM.

The full structure of the Bidirectional LSTM model was five Bidirectional LSTM layers with 64, 32, 16, 32, and 64 units each, with a layer of Dropout in between each **(Figure 4)**. 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. Predictions were rounded 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 4).** Additionally, a full confusion matrix of the top-performing model was generated **(Table 3)**.

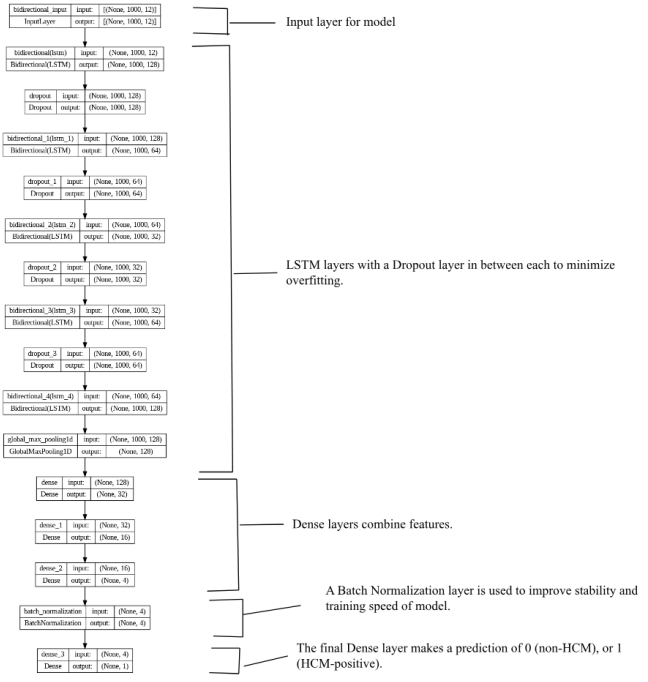

Figure 4: EKG LSTM model structure. Shows structure and input and output sizes for each layer. Consists of 5 Bidirectional LSTM layers with Dropout in between each, followed by a GlobalMaxPooling1D layer, followed by three Dense layers, a Batch Normalization layer, and a final Dense layer with one unit. This model structure was implemented to classify the EKG scans into HYP-positive and HYP-negative values.

https://doi.org/10.59720/23-209

Received: August 4, 2023
Accepted: November 30, 2023
Published: July 29, 2024

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,* vol. 79, no. 4, Feb. 2022, https://doi.org/10.1016/j.jacc.2021.12.002.
2. Weissler-Snir, Adaya, *et al.* "Hypertrophic Cardiomyopathy–Related Sudden Cardiac Death in Young People in Ontario." Circulation, vol. 140, no. 21, Oct. 2019, https://doi.org/10.1161/CIRCULATIONAHA.119.040271.
3. Bazzi, Talal, *et al.* "Trends in Breast Cancer Incidence and Mortality in the United States From 2004-2018: A Surveillance, Epidemiology, and End Results (SEER)-Based Study." *Cureus* vol. 15,4 e37982. 22 Apr. 2023, https://doi.org/10.7759/cureus.37982.
4. Hamet, Pavel, and Johanne Tremblay. "Artificial Intelligence in Medicine." *Metabolism,* 11 Jan. 2017, https://doi.org/10.1016/j.metabol.2017.01.011.
5. 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.
6. Sharifrazi, Danial. "Hypertrophic Cardiomyopathy Dataset." *Kaggle,* 4 Oct. 2021, www.kaggle.com/datasets/danialsharifrazi/hypertrophic-cardiomyopathy-dataset.
7. "PTB-XL ECG Dataset." *Kaggle,* www.kaggle.com/datasets/khyeh0719/ptb-xl-dataset. Accessed 29 July 2023.
8. Ahsan, Md Manjurul, *et al.* "Machine-Learning-Based Disease Diagnosis: A Comprehensive Review." *MDPI,* Multidisciplinary Digital Publishing Institute, 15 Mar. 2022, https://doi.org/10.3390/healthcare10030541.
9. Huang, Zhentao, *et al.* "DSCNN-LSTMs: A Lightweight and Efficient Model for Epilepsy Recognition." *Brain Sciences,* vol. 12,12 1672. 5 Dec. 2022, https://doi.org/10.3390/brainsci12121672.
10. "OS - Miscellaneous Operating System Interfaces." *Python Documentation,* docs.python.org/3/library/os.html. Accessed 29 July 2023.
11. "User Guide." U*ser Guide - Pandas 2.0.3 Documentation,* pandas.pydata.org/docs/user_guide/index.html. Accessed 29 July 2023.
12. "Numpy User Guide." *NumPy User Guide - NumPy v1.25 Manual,* numpy.org/doc/1.25/user/index.html#user. Accessed 29 July 2023.
13. "Users Guide." *Users Guide - Matplotlib 3.7.2 Documentation,* matplotlib.org/stable/users/index.html. Accessed 29 July 2023.
14. "Introduction." *OpenCV,* docs.opencv.org/3.4/d1/dfb/intro.html. Accessed 29 July 2023.
15. "WFDB." *Wfdb,* wfdb.readthedocs.io/en/latest/. Accessed 29 July 2023.
16. Team, Keras. "Keras Documentation: Developer Guides." *Keras,* keras.io/guides/. Accessed 29 July 2023.
17. "Wavelet Transforms in Python." *PyWavelets,* pywavelets.readthedocs.io/en/latest/. Accessed 18 Mar. 2024.