Patent Dutus Arteriosus (PDA) Detection with Deep Learning Models of CNN and CRNN

Asshish Bora
University of California, Irvine
boraa1@uci.edu

Chutong Xiao
University of California, Irvine
chutongx@uci.edu

San Zhang
University of California, Irvine
sanz@uci.edu

Tanya Shourya
University of California, Irvine
tshouya@uci.edu

Abstract

This project aims to assist hospitals in the task of detecting patent dutus arteriosus (PDA) in echocardiograms. PDA is a congenital heart disease where an extra blood vessel, the ductus arteriosus, between the aorta and the pulmonary artery does not close as expected after birth. In cases where the blood vessel opening is large, serious symptoms can occur and assisted closure is required. Because caring for infants with PDA can be difficult, early detection of PDA in echocardiograms is critical and can be assisted with using deep learning. Several deep learning models are trained on the clinical data provided by the Children's Hospital of Orange County (CHOC) for the task of binary classification (Positive for PDA vs Negative for PDA) of echocardiograms.

1. Introduction

This project aims to explore different methods and machine learning model architectures to detect patent ductus arteriosus (PDA) using echocardiograms as input. PDA is the second most common congenital heart disease and is more common in preterm infants.[1] The ductus arteriosus serves an important function before birth, but usually closes within the initial weeks after birth as it is no longer needed. In cases where it does not close, the opening pumps extra blood to the lung arteries which congest the lungs and strain the heart. Depending on the size, PDA can have different effects. A small opening may not have an adverse effect, but a large opening can lead to breathing difficulty and can impact growth. Over time, it can cause permanent damage to the lung arteries due to the opening[2]. If the opening does not close, surgery may be required to treat PDA. An echocardiogram, an ultrasound of the heart, can depict detectable PDA and reveal the condition of the heart.

Research performed by the Children's Hospital of Orange County (CHOC) shows positive results towards the approach of using a Convolutional Neural Network (CNN). In order to improve upon existing results, different pre-trained model architectures were used in addition to optimizing classification thresholds. The goal of this project is to improve the detection accuracy of PDA with a focus on sensitivity. Sensitivity is the true positive rate and is calculated using the following equation:

$$Sensitivity = \frac{Number\ of\ True\ Positives}{Number\ of\ True\ Positives + Number\ of\ False\ Negatives}$$

Another metric, specificity, or the true negative rate, is calculated using the below equation:

$$Specificity = \frac{Number\ of\ True\ Negatives}{Number\ of\ True\ Negatives + Number\ of\ False\ Positives}$$

Sensitivity is the more important metric of the two in this project as it is far more critical to correctly classify individuals with PDA rather than being able to correctly classify individuals not affected by PDA.

2. Prior and related work

Previous research performed by CHOC on detecting PDA in echocardiograms using deep learning was done using echocardiograms recorded between 2017-2021 at the Neonatal Intensive care unit in CHOC as mentioned in their paper. This dataset consisting of 461 echocardiograms was used by CHOC for training their deep learning model. Among these, 272 clips were labeled containing PDA and 190 as normal. Each clip contains a set of frames and the number of frames within a clip vary between echocardiograms. The data was split into 316 clips for training, 74 clips for validating and 72 clips for testing the model.

PDA can be clearly identified only during certain frames of a clip, namely the diastolic phase of the cardiac cycle. Frames from echocardiograms were manually selected that clearly show the presence of PDA. This ensures removal of noise and allows for training on only the frames that contain detectable PDA. This filtered data was stored separately in the TRAIN_PDA_FILTERED folder by the CHOC team. This process was not repeated for the TEST and VAL classes to test the model's effectiveness on whole clips.

Data imbalance was further identified and handled by the team by augmenting the class having less clips with additional clips from the same class. The clips were cropped to de-identify patient information and frames were extracted from video clips containing one or more cardiac cycles. Frames were resized to 512x320 pixels as part of pre-processing.

The goal of the project was to explore and analyze the efficacy of convolutional neural networks (CNN) for the classification task. The CNN structure was chosen by the research team at CHOC due to the availability of multiple pre-trained models that have proven effectiveness with image classification tasks. The pre-trained MobileNetV2 was explored due to its lightweight framework for CHOC's long-term goal of eventual clinical deployment onto edge devices. As part of the research, individual frames from clips were treated as inputs to the model. The weights were initialized to the pre-trained weights from ImageNet and were trained on two classes, PDA and normal (non-PDA). The model was trained on 30 epochs with a batch size of 128 frames. After training, the likelihood of the frame containing PDA is returned for each frame. If the likelihood was greater than a threshold of 0.5, it was classified as PDA, else it was identified as normal. The classification at the clip level was performed by aggregating the results of all frames in a clip. A threshold, β, of 50% was taken since in the data used by the CHOC team, approximately 50% of the total frames showed evidence of PDA. Additionally, the β value of 50% returned the best results for classification accuracy of clips in the VAL dataset.

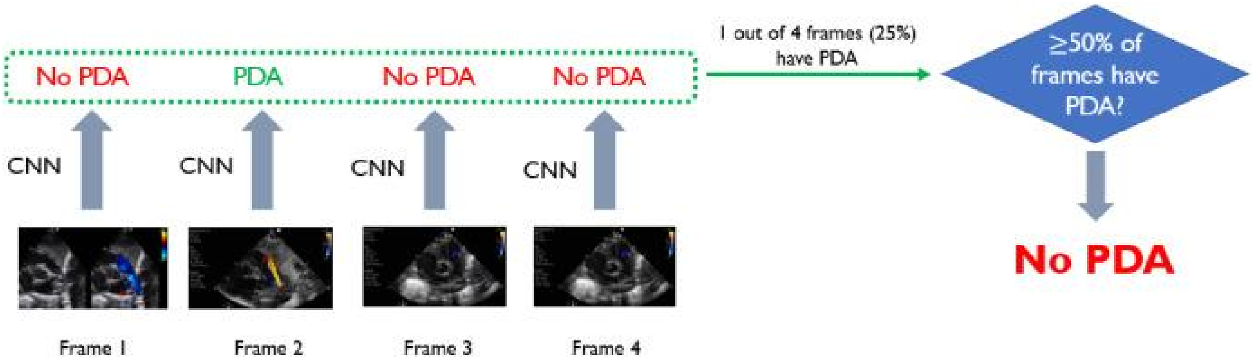
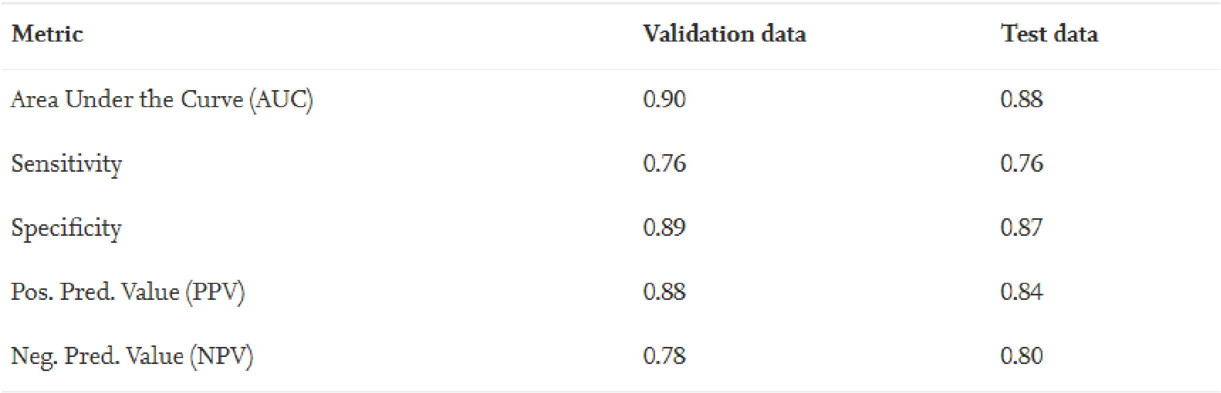


Figure 1. Classifying the clip using a threshold of 0.5. In this case, since 25% of frames have PDA, it is then classified as not containing a PDA [3]

The VAL dataset was used by CHOC to determine number of epochs and batch size while the TEST dataset was considered for the metrics after fine tuning the model on VAL dataset. An AUC score of 0.88 was achieved by MobileNetV2 with a sensitivity of 0.76. The table below shows the results achieved by CHOC as mentioned in their research paper [3].



Metric	Validation data	Test data
Area Under the Curve (AUC)	0.90	0.88
Sensitivity	0.76	0.76
Specificity	0.89	0.87
Pos. Pred. Value (PPV)	0.88	0.84
Neg. Pred. Value (NPV)	0.78	0.80

Table 1. Table showing the metrics achieved by CHOC research as mentioned in their research paper[3]

Another research work [4] in the field of classification of echocardiograms aims to detect prosthetic valves using echocardiograms as inputs. This research compares the performance of 12 pre-trained CNN architectures trained on their dataset. They use a similar approach of extracting frames from clips and pre-processing them to fine-tune their pre-trained models. The researchers used 2046 echocardiograms where 1597 belonged to the Natural Mitral Valve class and 447 belonged to the prosthetic Mitral Valve. The research work mentions that the use of more complex CNNs require more demanding training data. The number of training samples in this work was much larger than the number of training samples provided by CHOC, but the comparison of different CNN architectures provided insight on the best performing CNN architectures for classification of clinical data.

3. Data Description

The dataset from CHOC is made up of folders for echocardiograms, and inside each folder are video frames that have been taken from clips of echocardiograms. The number of frames in a single echocardiogram can be anywhere between 20 and 200.

The dataset provided by CHOC is divided into four folders:

1. TRAIN
2. TEST
3. VAL
4. TRAIN_PDA_FILTERED

Each folder is then further divided into POS and NEG. Each POS folder contains different echocardiograms folders for different patients. These echocardiograms have been classified as containing PDA. Similarly, each NEG folder contains echocardiograms of patients that do not have PDA. The distribution of POS and NEG in the dataset is as shown in Figure 2:

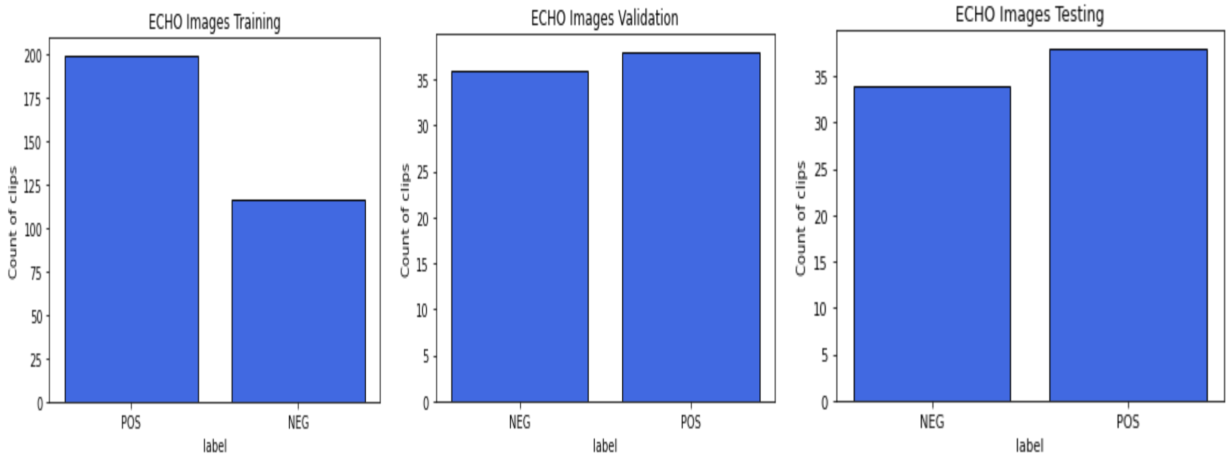


Figure 2: Distribution of images between classes for train, validation and test

As can be seen in Figure 2, the TRAIN folder is imbalanced. Class weights are used in model building to counter this data imbalance. On the other hand, the VAL and TEST data distributions are balanced.

The TRAIN_PDA_FILTERED folder is similar to the TRAIN folder that is used for training. However, the POS section contains frames that have been manually filtered by physicians to have PDA. So instead of having every frame of a POS echocardiogram, it contains only the select frames that have detectable PDA. This manual filtering resulted in a reduction of the number of frames in the POS echocardiograms. The distribution is shown in Figure 3.

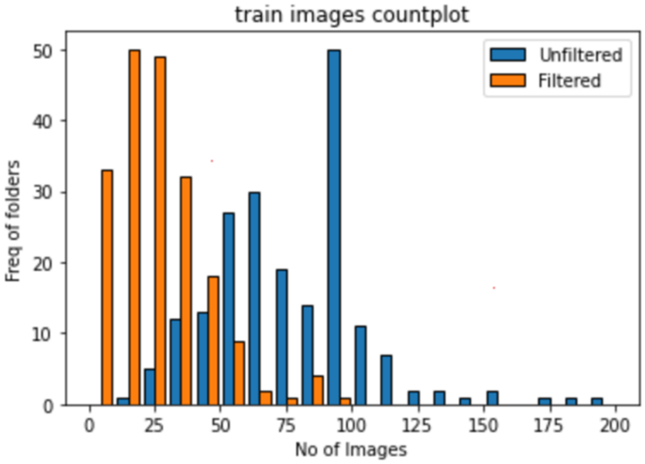


Figure 3: Countplot of images per folder

4. Approach

4.1 Data Preprocessing

4.1.1 Frame-Based Prediction

Echocardiogram images contained textual data that were removed using pre-processing techniques. The text data do not hold importance to the classification problem and different approaches were used to remove them.

Using Custom mask

By keeping the bits that correspond to the black region as 0 and the ones that correspond to the white region as 1, a binary mask is created. The mask's dimensions are kept constant with that of the image. The textual data is then masked using a bitwise and operation from the OpenCV package.

The challenge faced in this approach was inconsistent dimensions of frames. The different types of frames would require different types of custom masks. The data contains some frames with no textual data, some with 2 echocardiograms side by side and some containing only a single echocardiogram. This meant different mask maps needed to be used with the different frames which was not an optimal approach to remove text.

Using image inpainting

Using keras optical character recognition (OCR), text is able to be detected in an image. The box containing the text is detected and the box coordinates are used to inpaint the box containing the text, which is achieved using OpenCV. A mask is created for every image such that the area of the detected box is set to 1 and the rest of the area is set to 0, keeping the dimension of the mask the same as that of the image. The inpaint function from OpenCV is then used with the created mask to remove the text from the image. This approach was successful on a large set of images but the computational cost was expensive for every frame processed.

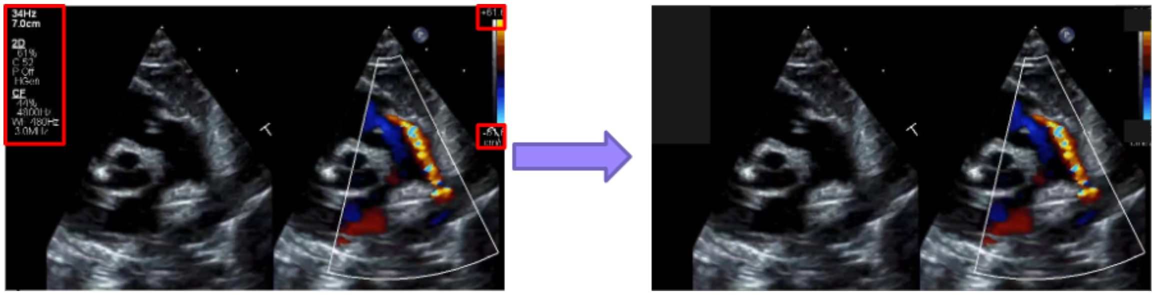


Figure 4. Text detection in echocardiogram on the left and output of image after image inpainting using OpenCV on the right

Using contour detection

Using OpenCV to detect contour in the echocardiogram, the image is cropped to keep only the echocardiogram with the blood flow readings as shown in the Figure 5. The image is first converted to grayscale and then thresholded. The pixels are identified in the threshold and the desired contour is detected and extracted. The bounding box is created to then crop the image to get the desired echocardiogram without the rest of the frame. This approach was successful on a majority of the frames, but using the cropped frames as inputs led to overfitting in the model. The original frames with the text performed on the pretrained CNNs and this could be because each cropped frame has less information to be processed by the CNN, whereas the original frame contains more information, making it more suitable for a complex CNN to process and thereby reducing the chance of overfitting.

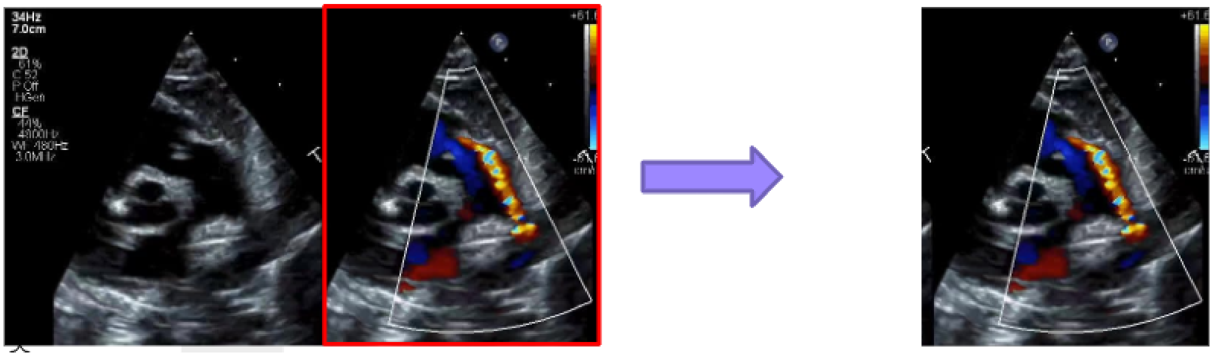


Figure 5. Cropping the frame using contour detection by OpenCV

4.1.2 Clip Based Prediction

Model training is done such that the model is able to predict POS echocardiograms more accurately. To do so, models were initially trained on TRAIN_PDA_FILTERED which contained only frames that had PDA inside them. But due to the difference between TRAIN and TRAIN_PDA_FILTERED as shown in the figure:

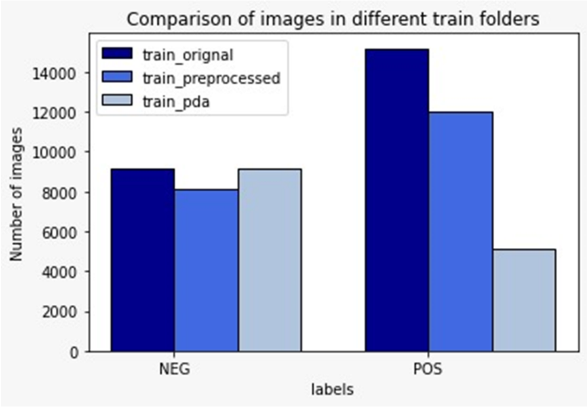


Figure 6. Image distribution between train, train batched and train_pda_filtered

A new strategy is introduced. Instead of working on individual frames, data was split into batches to perform a prediction based on clips rather than frames. As the model requires a fixed number of frames as input, data is split into batches of 16. Each batch is only considered if at least one of the frames inside the batch contains PDA i.e a batch has to have at least 1 frame in TRAIN_PDA_FILTERED. This splitting leads to an increase in size on the number of images from around 5000 to 12000 in the POS section as shown in figure above.

4.2 Modeling

4.2.1 CNN

As mentioned before, the dataset from CHOC consists of echocardiogram folders where each folder contains video frames extracted from the respective echocardiogram. Thus, employing convolutional neural networks (CNN) was an obvious choice for addressing PDA detection due to the numerous pre-trained CNNs available as well as their proven performance in image classification tasks.

Convolutional neural networks are deep learning models that are able to learn characteristics or features from images, making them extremely useful for image classification tasks. The architecture of a CNN is typically a combination of multiple convolutional, pooling, and fully-connected layers. The convolutional neural layer is what takes up the bulk of the computations and is essentially responsible for calculating a feature map given the input. The pooling layer typically follows a convolutional layer and down samples from the feature map calculated from the convolutional layer to reduce the number of features needed to be learned by the model. The fully connected layer is what is typically used at the end of a CNN and is analogous in structure to the neurons in the human brain. The output of the final fully connected layer is essentially a vector of length k (where k is the number of classes) depicting a probability for each class. Another type of layer not always used in CNNs is the dropout layer, which is typically used to combat overfitting. The dropout layer essentially randomly shuts off the contributions of some neurons during the training phase with a probability *p*. This prevents the model from learning possibly insignificant features from an image and potentially overfitting.

Several pre-trained CNNs were selected to be trained for this classification task, namely MobileNet V2, Mobile NASNet, EfficientNet B0 and EfficientNet B3. The MobileNet V2 was selected by CHOC in their research paper due to the model's light framework yet high performance on the ImageNet 1000 class classification. Thus, both MobileNet V2 and Mobile NASNet were selected for this project for that reason. The EfficientNet family of models was also selected due to their higher accuracy and better efficiency in comparison to existing neural networks. Only EfficientNet B0 and EfficientNet B3 were chosen due to hardware limitations as EfficientNets B4-B7 had exceedingly large quantities of trainable parameters in comparison. These CNNs are trained for the PDA detection task using transfer learning by changing the number of output features in the final fully connected layer from the original 1000 to 2. Additionally, inputs are normalized and resized to (224, 224) before being passed into each model.

The aforementioned CNNs are trained using the video frames in the TRAIN_PDA_FILTERED dataset. Rather than using the VAL dataset provided by CHOC, the CNNs are validated on 20% of the TRAIN_PDA_FILTERED dataset and trained on the other 80%. This is due to the 2 datasets having different structures as only the frames containing visible PDA for each echocardiogram are present in the POS class of the TRAIN_PDA_FILTERED dataset while every frame of an echocardiogram, regardless if it contains visible PDA or not, is present in the POS class of the VAL dataset. Each CNN undergoes 30 training epochs on a batch size of 16 images due to memory limitations. The CNNs are validated on video frames to judge how well they can detect PDA in a given video frame before being tested on entire echocardiogram clips.

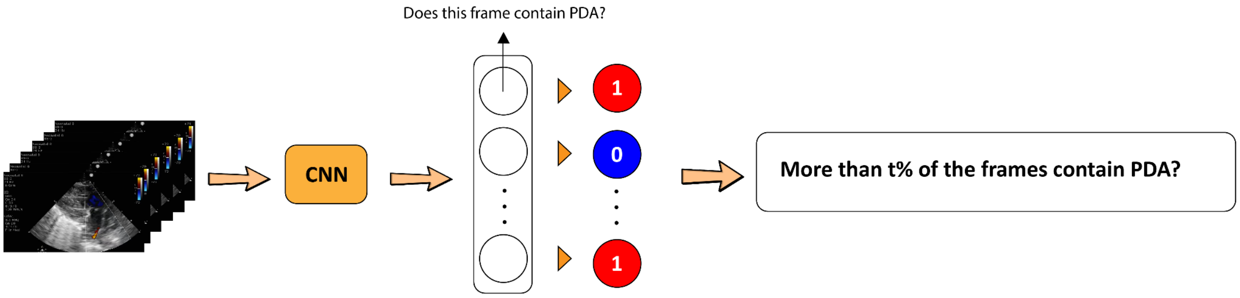

Does this frame contain PDA?
CNN
1
0
1
More than t% of the frames contain PDA?

Figure 7. Overview of the CNN method

During the testing phase on echocardiogram clips, each frame of a clip in the TEST dataset is classified by the CNN as POS or NEG for containing PDA. After all frames are put through the model and classified, the ratio of POS frames in each clip is calculated and compared against a threshold value t. If the ratio of POS frames is greater than the threshold value t, then the echocardiogram is classified as containing PDA. Figure 7 above depicts an overview of the process for which echocardiograms are classified. An initial value of 0.5 was used for t, meaning over 50% of the frames in any echocardiogram must be classified as POS for the clip to be classified as containing PDA. Additional threshold values tested included the β value mentioned in the CHOC research paper as well as the maximum geometric mean value. The β value is based off the percentage of all video frames in the training set that contain PDA. Specifically, out of the 15,182 total video frames in the training set, 5,082 (approximately 33.47%) contained PDA, resulting in a β value of 0.3. The geometric mean value is a metric calculated from the Receiver Operating Characteristic (ROC) curve at a given threshold using the equation:

$$Geometric\ Mean = \sqrt{Sensitivity * Specificity}$$

Selecting the maximum geometric mean value returns a threshold that balances sensitivity and specificity. Because each model has different ROC curves, different maximum geometric mean values are calculated for each and used as threshold values.

Feature Visualization

A deep learning model is like a black box. Due to the intricate math involved in each model, it is extremely challenging to understand what is happening inside of it. Feature visualization is used to display how models interact with an image throughout the layers. A vague understanding is developed by processing the image through different layers of the model and used to estimate the number of layers that works best.

A new model is created by using the weights of the previously trained model. The input of the model is defined by an input layer of shape (224,224,3). The output of every layer after the input layer is saved and used to visualize the features.

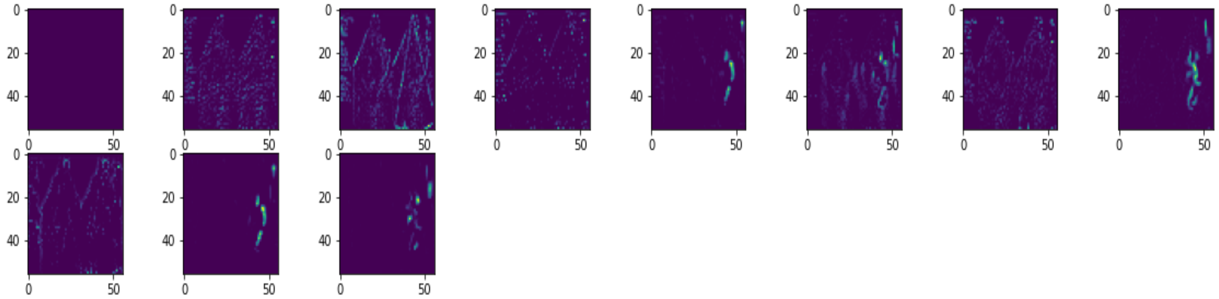


Figure 8: Features extracted from the 15th layer

Figure 8 shows the 15th layer's feature visualization with dimensions (1,56,56,11). CNN filters with various setups extract different features from the frame to make this work. Following feature extraction from one layer, the output is used as input for further CNN filters, and so on, until the neural network is able to interpret those values and return an outcome.

Starting layers are able to extract high-level features and as the input progresses through the training process, increasingly complicated features essential to the neural network, some of which may not make sense to the human eye, are extracted. What a model does during feature extraction is shown below.

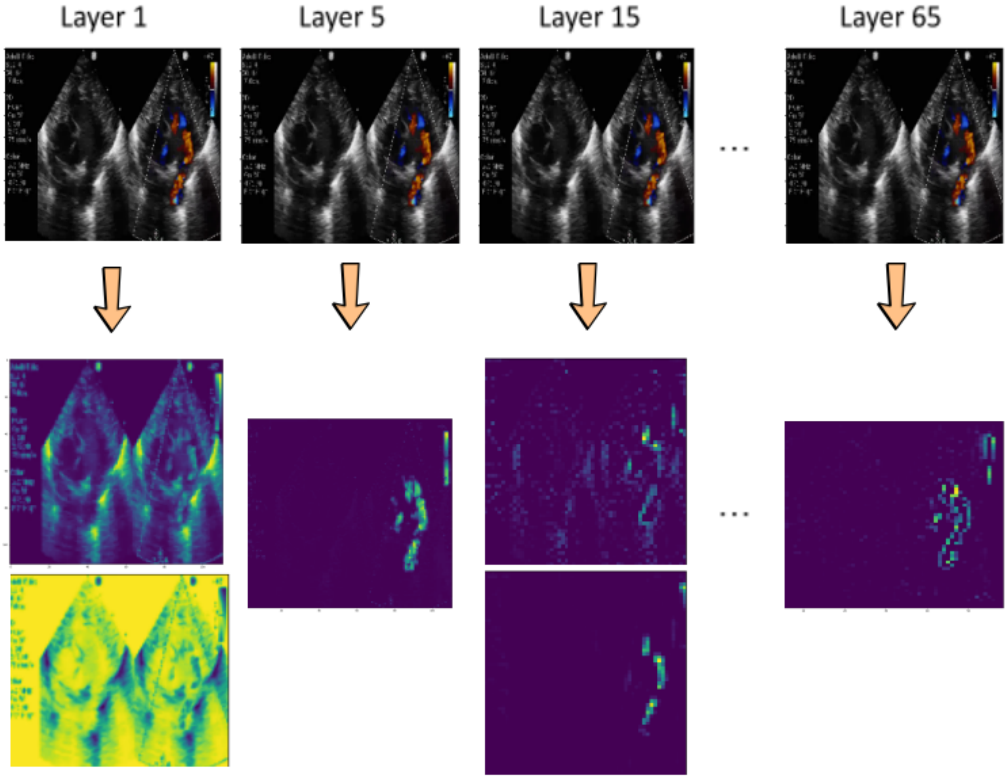


Figure 9: Samples of Feature Extraction applied on layer 1, 5, 15 and 65

Figure 9 shows high level extraction operations like grayscaling or picture inversion carried out in layer 1. The model begins to extract features from the colored areas of the image by layer 5. By layer 15, the model is differentiating different hues, such as the blue and red portions, and may be able to recognize different blood flows and distinctions between PDA and non-PDA images.

The model begins to outline the blood flow area by layer 65 as shown in. Over 400 layers make up the model that is being used in these figures for feature visualization. The results of visualizing through the initial layers suggested that such complicated models may not be necessary. Given that PDA appears to be a high level trait rather than a low level, having a complicated model may contribute to overfitting. Thus, models with lighter frameworks and smaller numbers of trainable parameters were chosen for this project, combating the problem of overfitting.

4.2.2 CRNN

Known as a sequential learning method, Recurrent Neural Networks (RNN) are widely used in speech recognition, time series prediction, and machine translation. While Convolutional Neural Networks (CNN) excel at processing spatial data, Recurrent Neural Networks (RNN) prefer temporal data. The echocardiograms from CHOC consist of 20 ~ 200 video frames in chronological order, some of which contain PDA. This combination of spatial and temporal data provoked the exploration of how the correlation between frames contributes to the detection of PDA using a hybrid CNN-RNN Deep Learning framework.

Adding an RNN to model architecture is further motivated by the importance of timing in PDA detection. PDA, or the opening between vessels where blood flows swiftly when the heart pumps, can be marked by timing. In the CHOC study, the marker was specifically described as "flow timing during diastole"[3], which implies that physicians assess the presence of PDA in an echocardiogram based on their temporal memory or understanding of the timing of the cardiac cycle. RNN simulates the biological process in the human brain when processing information flow, giving reason to speculate that it may be able to detect the transition from systole to diastole.

Long Short-Term Memory

In order to examine the impact of RNN, CRNN models that included pre-trained ResNet18 and MobileNetV2 as base models, along with a Long short-term memory (LSTM) block were deployed. LSTM was first introduced by Hochreiter and Schmidhuber in 1997 [5], in which they introduced the concept of memory cells and gate units as a means of storing and processing information. A new gate unit, the forget gate, was added to the LSTM model by Gers, Schmidhuber, and Cummins in 2000, thereby preventing the model from collecting useless information over time [6]. The echocardiograms do not have a clear starting or ending point of a complete cardiac cycle, which has proven to be a difficult challenge for the original LSTM model to learn. Therefore, the forget gate is a crucial component of the model architecture. Below are mathematical expressions that illustrate how an LSTM with a forget gate updates its memory during the learning process.

$$f_t = \sigma(W_{fh}h_{t-1} + W_{fx}x_t + b_f)$$

$$i_t = \sigma(W_{ih}h_{t-1} + W_{ix}x_t + b_i)$$

$$\tilde{c}_t = tanh(W_{\tilde{c}h}h_{t-1} + W_{\tilde{c}x}x_t + b_{\tilde{c}})$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t$$

$$o_t = \sigma(W_{oh}h_{t-1} + W_{ox}x_t + b_o)$$

$$h_t = o_t \cdot tanh(c_t)$$

Variable	Definition
f_t	The Forget Gate, which determines whether to forget parts of the long term memory given the input and the previous hidden state.
i_t	The Input Gate, which determines whether to add new information to the long term memory given the input and the previous hidden state.
$\tilde{c}_t$	The new memory network that outputs vectors of the new memory given the input and the previous hidden state.
c_t	The updated cell state, generated by combining the pointwise multiplied vectors given by the Forget Gate and the Input Gate.
o_t	The Output Gate, which determines the next hidden state given the input, the previous hidden state and the updated cell state.
h_t	The Hidden State, encoded memories.
σ	Sigmoid activation function which restricts the function value to the interval between 0 and 1.
$tanh$	Tanh activation function which restricts the function value to the interval between -1 and 1.

Table 2. Explanation of the mathematical symbols in LSTM equations

In summary, the gate units, all of which come with sigmoid activation functions, are structures for LSTMs to make YES-OR-NO decisions during memory formation. As their names suggest, these units act as gates that allow information to pass when output is near 1 and block it when output is near 0. The memory cell in an LSTM resembles a room with three gates. Its internal furnishings are progressively added to or removed as the learning process unfolds.

Model Architecture

As mentioned in Section 4.1, TRAIN video folders are divided into multiple subfolders with 16 video frames each. To ensure that each subfolder contained at least one PDA in the POS, the video frames in each subfolder had been compared with the TRAIN_PDA_FILTERED folder. Because there are only 1262 subfolders for training, data augmentation including random flipping and affine transformations was performed to increase the size of the dataset. CNN base models, serving as encoders, transformed spatial data into feature vectors.

The two base models are summarized as follows:

1) A ResNet18 network with pretrained weights learned from UCF101. UCF101 includes 101 categories of action videos collected from YouTube.
2) A MobileNetV2 network with pretrained weights learned from Section 4.2.1, as MobileNetV2 is the best-performing model amongst all pre-trained CNN models.

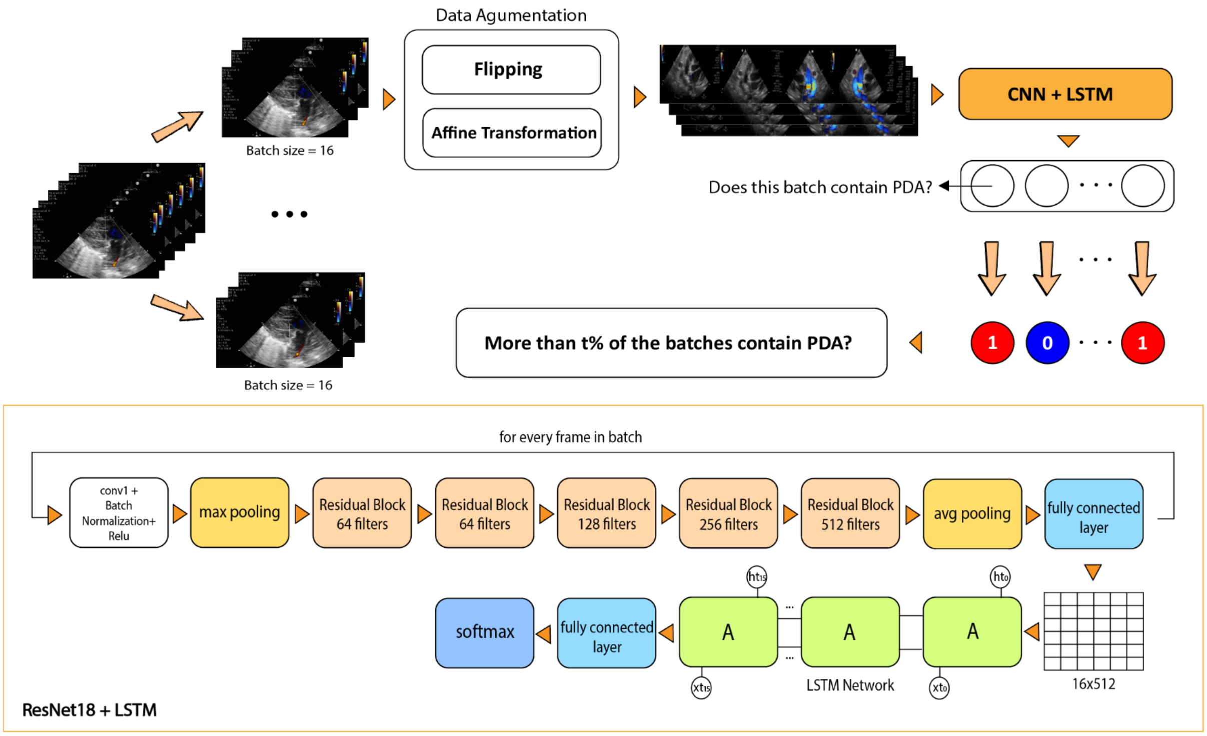


Figure 10. Overview of the CRNN method and the structure of the ResNet18+LSTM network.

Figure 10 is an overview of the entire CRNN model process and the ResNet18+LSTM model structure. For each data batch, CNN produces 16 feature vectors, which LSTM treats as input for one information flow. The LSTM decodes the encoded vector sequence and outputs the last hidden state. A softmax activation function was applied to estimate the probability of an echocardiogram segment containing PDA. The average predictions for segments from the same echocardiogram determines whether the video clip as a whole has PDA.

5. Results

5.1 CNN Pretrained Models

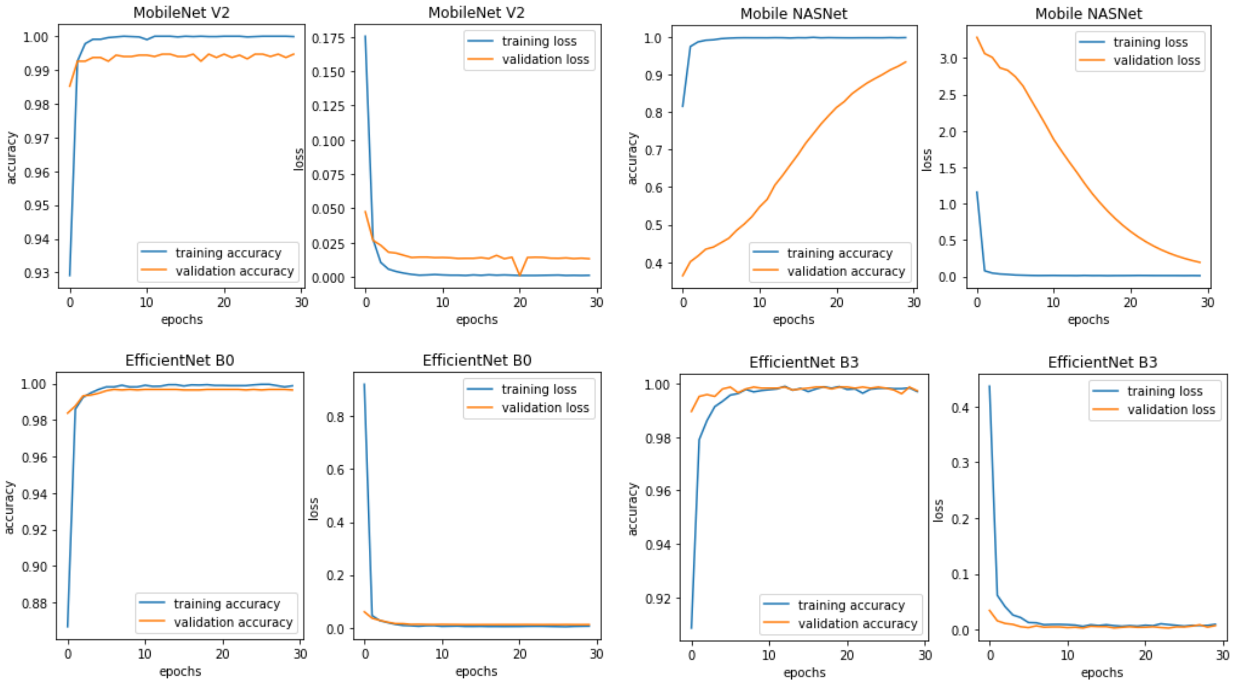


Figure 11. The training and validation accuracy/loss graphs for CNNs trained on individual frames

Figure 11 shows the training and validation accuracy and loss graphs for the CNNs trained on individual frames. Each CNN was trained on 30 epochs with a batch size of 16 on an NVIDIA RTX 2060 GPU. As can be seen, every CNN was able to converge within 30 epochs other than the Mobile NASNet. Due to memory limitations, Mobile NASNet was not able to be trained to convergence and is expected to have the worst performance out of the 4 models. Table 3 below depicts the training and prediction times of each model.

Model	MobileNet V2	Mobile NASNet	EfficientNet B0	EfficientNet B3
Training time	29 min 35s	30 min 15s	37 min 0s	76 min 1s

Prediction time	0.56 sec/clip	0.49 sec/clip	0.67 sec/clip	1.16 sec/clip

Table 3. Training and prediction times for CNNs trained on individual frames

Model	AUC		*t* = 0.5		*t* = 0.3		Max Geom Mean	
	VAL	TEST	VAL	TEST	VAL	TEST	VAL	TEST
MobileNetV2	0.987	0.971					*t* = 0.2	
Sensitivity			0.737	0.676	0.868	0.852	0.921	0.912
Specificity			1.000	0.974	1.000	0.974	0.944	0.974
PPV (Pos. Pred. Value)			1.000	0.958	1.000	0.967	0.946	0.969
NPV (Neg. Pred. Value)			0.783	0.711	0.878	0.881	0.919	0.925
EfficientNet B0	0.991	0.964					*t* = 0.2	
Sensitivity			0.816	0.735	0.895	0.853	0.921	0.912
Specificity			1.000	1.000	1.000	0.947	1.000	0.947
PPV (Pos. Pred. Value)			1.000	1.000	1.000	0.935	1.000	0.939
NPV (Neg. Pred. Value)			0.837	0.809	0.900	0.878	0.923	0.923
EfficientNet B3	0.993	0.944					*t* = 0.4	
Sensitivity			0.711	0.676	0.895	0.824	0.789	0.824
Specificity			1.000	0.974	1.000	0.947	1.000	0.974
PPV (Pos. Pred. Value)			1.000	0.958	1.000	0.933	1.000	0.966
NPV (Neg. Pred. Value)			0.766	0.771	0.900	0.857	0.818	0.860
Mobile NASNet	0.916	0.877					*t* = 0.6	
Sensitivity			0.921	0.971	0.974	1.000	0.895	0.912
Specificity			0.722	0.684	0.556	0.526	0.778	0.763
PPV (Pos. Pred. Value)			0.778	0.733	0.698	0.654	0.810	0.775
NPV (Neg. Pred. Value)			0.897	0.963	0.952	1.000	0.875	0.906

Table 4. The evaluation metrics of CNNs trained on individual frames using different thresholds

Table 4 shows the sensitivity, specificity, positive predictive value, and negative predictive values of the pre-trained CNNs using the 3 different thresholds tested. As expected from the training and validation loss and accuracy graphs, Mobile NASNet had the worst performance, which is further supported by the ROC curves in the below Figure 12. MobileNet V2 and EfficientNet B0 had the best performances while having comparable prediction times. In terms of efficiency and performance on TEST data, MobileNet V2 performs just marginally better than EfficientNet B0.

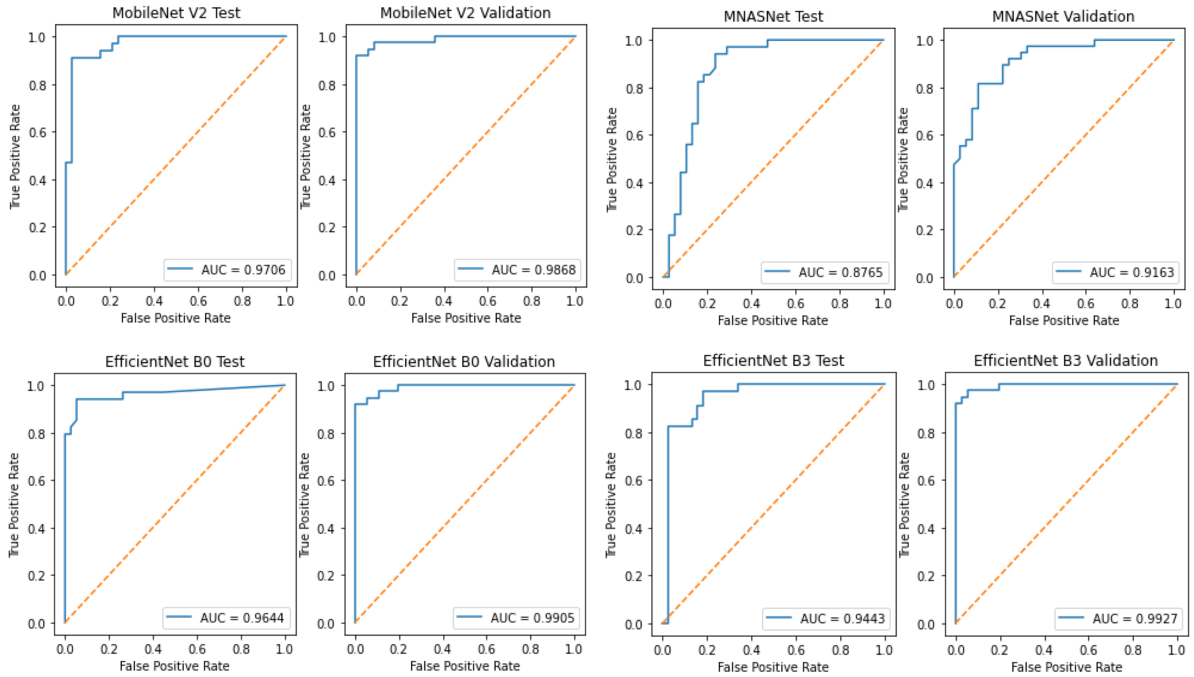
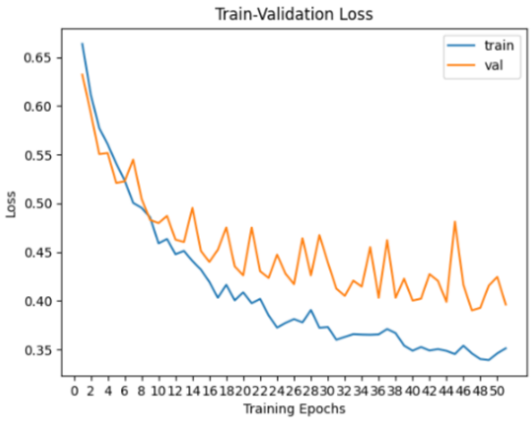


Figure 12. The ROC curves of CNNs trained on individual frames

5.2 CRNN Models



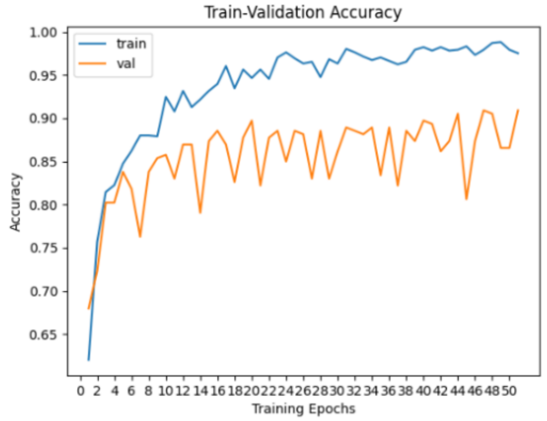


Figure 13. The training and validation accuracy/loss graphs for ResNet18+LSTM trained on video segments

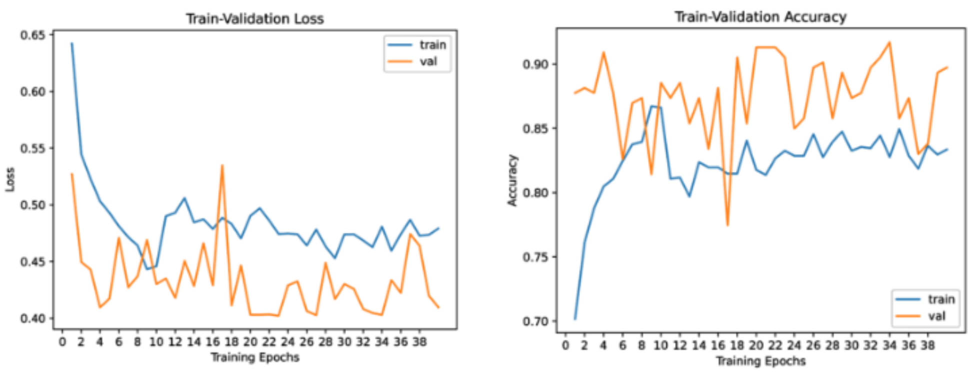


Figure 14. The training and validation accuracy/loss graphs for MobileNetV2+LSTM trained on video segments

As shown in Figure 13 and 14, each CRNN model was trained on 40 epochs with batch size = 8 on a NVIDIA GeForce RTX GPU. The batch size was kept small due to memory limitations, and the learning curves of loss and accuracy indicated that the model failed to converge. The validation accuracy of ResNet18+LSTM and MobileNetV2+LSTM fluctuated between 0.85 and 0.90. Although the validation accuracy of MobileNetV2 + LSTM achieved above 0.85 at the beginning of training, it did not show a tendency to improve throughout.

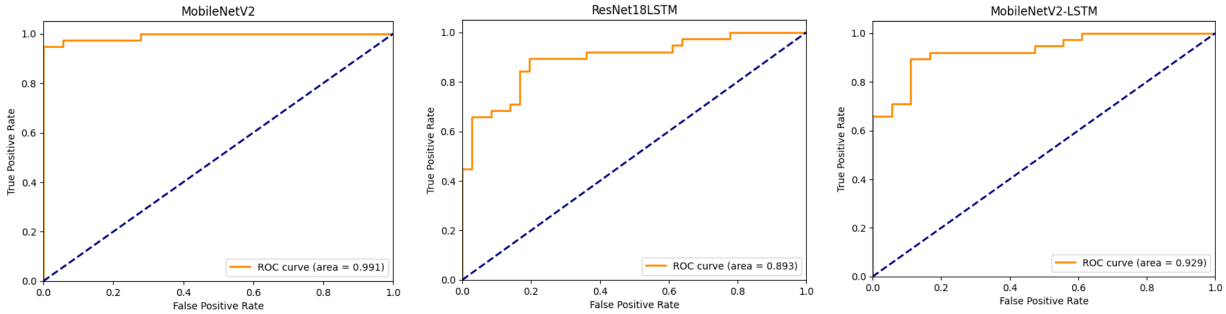


Figure 15. The ROC curves of MobileNetV2, RestNet18+LSTM and MobileNetV2-LSTM showing the performance of the models on VALIDATION.

Figure 15 and 16 show the ROC curves of MobileNetV2, RestNet18+LSTM and MobileNetV2+LSTM running on VALIDATION and TEST. As part of a fair comparison, the MobileNetV2 model started with predicting segments of 16 frames. It then used an average of POS probabilities to make its final predictions, which was the same as how CRNN models perform their testing.

As can be seen, all models have AUCs above 0.85, while all MobileNetV2 involved models reach AUCs above 0.9. Of the three models the MobileNetV2 trained with only TRAIN_PDA_FILTERED appears to be the most competitive.

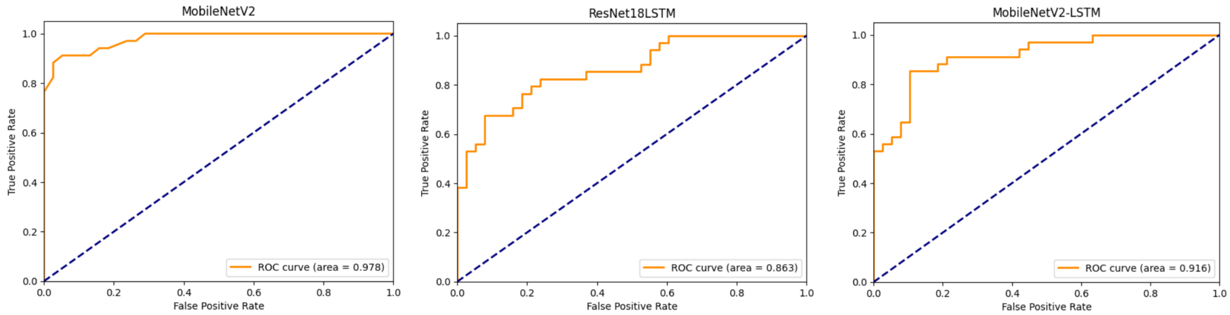


Figure 16. The ROC curves of MobileNetV2, RestNet18+LSTM and MobileNetV2-LSTM showing the performance of the models on TEST.

Model	AUC		t=0.2		t=0.3		t=0.4		t=0.5	
	VAL	TEST	VAL	TEST	VAL	TEST	VAL	TEST	VAL	TEST
Frame-based										
MobileNetV2	0.991	0.978								
Sensitivity			1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Specificity			0.868	0.765	0.816	0.765	0.737	0.706	0.684	0.647
PPV (Pos. Pred. Value)			0.878	0.826	0.837	0.826	0.783	0.792	0.750	0.760
NPV (Neg. Pred. Value)			1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Clip-based										
ResNet18-LSTM	0.893	0.863								
Sensitivity			0.722	0.658	0.806	0.711	0.806	0.737	0.806	0.816
Specificity			0.921	0.853	0.895	0.824	0.895	0.824	0.868	0.794
PPV (Pos. Pred. Value)			0.897	0.833	0.879	0.818	0.879	0.824	0.853	0.81
NPV (Neg. Pred. Value)			0.778	0.690	0.829	0.718	0.829	0.737	0.825	0.794
MobileNetV2-LSTM	0.929	0.916								
Sensitivity			0.778	0.684	0.806	0.763	0.889	0.842	0.889	0.921
Specificity			0.921	0.882	0.921	0.882	0.842	0.824	0.763	0.735
PPV (Pos. Pred. Value)			0.903	0.867	0.906	0.879	0.842	0.842	0.780	0.795

NPV (Neg. Pred. Value)	0.814	0.714	0.833	0.769	0.889	0.824	0.879	0.893

Table 5. The evaluation metrics of MobileNetV2, RestNet18+LSTM and MobileNetV2-LSTM.

Table 5 shows a comprehensive look at the comparison of three models with thresholds ranging from 0.2 to 0.5. It is noticeable how MobileNetV2 is robust at detecting PDAs. While MobileNetV2 displays high performance on sensitivity and PPV, CRNN models have higher performance on specificity and NPV on all thresholds. CRNNs appear to be more cautious when predicting PDA, which is reasonable as CRNNs consider 16 times the information compared to CNN(MobileNetV2) per prediction.

5.2.1 Comparison of the Efficiencies

Model	Time consumed per prediction (second)
MobileNetV2	0.101
ResNet18-LSTM	0.474
MobileNetV2-LSTM	0.543

Table 6. The efficiency of MobileNetV2, RestNet18+LSTM and MobileNetV2-LSTM.

Incorporating LSTM into model architecture has the drawback of having a long evaluation time. A comparison of MobileNetV2 and CRNN models is shown in Table 6. A CRNN model takes four to five times longer to make a prediction than a CNN model. Iteration on the sequential inputs is the primary reason for LSTM's low computational efficiency.

6. Discussions

6.1 Limitations of the Current Research

Data Size

As shown in the Results section, the CRNN models do not outperform MobileNetV2. Insufficient data when training a CRNN model was considered one of the main reasons. A CNN can make a prediction based on only one image, whereas an LSTM uses a sequence of 16 encoded images per prediction. RNN has around 1250 video segments while CNN has around 14,000 video frames. Both CNN and CRNN update their weights based on the validation loss. In particular, when CRNN models update their weights based on the average validation loss, their decisions are more likely to be influenced by single prediction results. This can cause CRNN to become unstable.

Adaptability of LSTM

There has been evidence that LSTM is capable of video classification on popular datasets such as UCF-101 Human Actions and Columbia Consumer Videos (CCV). However, it should be noted that "detecting a key signal appears in a video" and "recognizing continuous movement" can be two completely different tasks. Since PDAs appear only about 33% of the time in the echocardiogram, it is perhaps more important to remember their physical positions and features at the time of appearance than to pay attention to chamber activity before and after. As evidence, MobileNetV2 trained with TRAIN_PDA_FILTERED reached a high level of sensitivity, demonstrating its ability to recall the characteristics of the PDA. There is no evidence to support the claim that RNNs will improve the detection of PDAs at this time.

6.2 Conclusion

The prevalence of AI in the medical field increases each year with models other than CNNs. The use of automated diagnostics has two distinct benefits for society. First, efficiency of healthcare institutions can be improved by screening suspicious cases with deep learning networks before further review by physicians. To achieve this, neural networks used for diagnosis must be sensitive enough to detect meaningful signals in images. Physicians can manually review filtered data to further reduce the risk of misdiagnosis. Second, in the New Media Age, patients can use their own medical data to conduct an initial test for PDA through mobile apps. With this automated and mobile diagnostic approach, patients can gain a basic understanding of their condition and have smoother communication with their physicians. Since limiting users and data uploads to a small number is not sustainable, the model should be lightweight and capable of instant responsiveness. In light of these considerations, a simple MobileNetV2 network may be the best choice for problems like PDA detection.

References

[1] Retrieved from https://kidshealth.org/en/parents/patent-ductus-arteriosus.html

[2] Retrieved from https://www.heart.org/en/health-topics/congenital-heart-defects/about-congenital-heart-defects/patent-ductus-arteriosus-pda

[3] Lei, Howard & Ashrafi, Amir & Chang, Peter & Chang, Anthony & Lai, Wyman. (2022). Patent ductus arteriosus (PDA) detection in echocardiograms using deep learning. Intelligence-Based Medicine. 6. 100054. 10.1016/j.ibmed.2022.100054.

[4] Vafaeezadeh, Majid & Behnam, Hamid & Hosseinsabet, Ali & Gifani, Parisa. (2021). A deep learning approach for the automatic recognition of prosthetic mitral valve in echocardiographic images. Computers in Biology and Medicine. 133. 104388. 10.1016/j.compbiomed.2021.104388

[5] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

[6] Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. Neural computation, 12(10), 2451-2471.

Report Contribution

Members: Tanya Shourya, Aashish Bora, San Zhang, Chutong Xiao

- ☐ Abstract
- ☐ Introduction
    - ☐ Define the problem
- ☐ Prior and Related Work
- ☐ Data Description
    - ☐ Fig: Sample distribution of TRAIN, VAL and TEST
    - ☐ Fig: Distribution of frames for video clips in TRAIN, VAL and TEST
- ☐ Data Preprocessing
    - ☐ Text Removal / Contour Detection (frame-based)
    - ☐ Splitting and reorganization of the video clips (clip-based)
- ☐ Methodology
    - ☐ CNN
        - ☐ CNN structure
        - ☐ Pre-trained models overview
        - ☐ Thresholds
        - ☐ Training
        - ☐ Feature visualization
    - ☐ CRNN
        - ☐ RNN overview
        - ☐ LSTM
        - ☐ Model Architecture
- ☐ Results
    - ☐ CNN
        - ☐ Metrics table, AUC plots, train-val plots, prediction times
    - ☐ CRNN
        - ☐ Metrics table, AUC plots, train-val plots, prediction times
- ☐ Conclusion
    - ☐ Limitations of the Current Research
    - ☐ Conclusion
- ☐ References

Project Contribution

Contributor(s)	File Name
Aashish Bora, Tanya Shourya	Batching and Data Visualization.ipynb
Aashish Bora	Feature_visualization.ipynb
	Cropping_Scans.ipynb
San Zhang	CHOC PDA EfficientNet_B0.ipynb
San Zhang	CHOC PDA EfficientNet_B0.ipynb
San Zhang	CHOC PDA MobileNet-V2.ipynb
San Zhang	MNASNet.ipynb
San Zhang	Threshold Experiments.ipynb
San Zhang	Predictions CHOC.ipynb
Tanya Shourya	PDA Detection NASNet Mobile(preprocessing+modelling+evaluation).ipynb
Tanya Shourya, Aashish Bora	Time_Series.ipynb
Aashish Bora	rresnet50(tensorflow)(modelling + evaluation).ipynb
Aashish Bora	text_removed.ipynb
Chutong Xiao	ResNet18-LSTM.ipynb
Chutong Xiao	Evaluation-ResNet18+LSTM.ipynb
Chutong Xiao	MobileNetV2-LSTM.ipynb
Chutong Xiao	Evaluation-MobileNetV2+LSTM.ipynb
Chutong Xiao	MobileNetV2_for_CRNN_fair_comparison.ipynb
Tanya Shourya	Text extraction1.ipynb
Tanya Shourya, Aashish Bora	Text Extraction.ipynb

myutil.py was retrieved from PyTorch Computer Vision Cookbook Chapter 10.