

Advanced Deep Learning Techniques for Accurate Prediction of Heart Diseases Using Electrocardiogram Signal Analysis

Dr. S. Senthilkumar¹, Dr. G. Mothilal Nehru^{2*}, Dr. C. Anbarasi³, S. Jayashree⁴, Dr. Vishwa Priya.V⁵, Dr. J. Jebathangam⁶, Dr. M. Yogeshwari⁷, D. Divya Sterlin⁸

¹Associate Professor Department of Computational Sciences, Brainware University, West Bengal, India, Mail Id: youcanwinsure@gmail.com

²Assistant Professor, Department of Computer Application, Faculty of Science and humanities, SRM University, Corresponding Author
Mailid: mgmnehr@gmail.com

³Assistant Professor, Department of Computer Science and Information Technology, Vel's Institute of Science, Technology & Advanced Studies (VISTAS), mail id: anbuchand16feb@gmail.com

⁴Assistant Professor, Department of Applied Computing Emerging Technologies, Vel's Institute of Science, Technology (VISTAS), mail id: shreelogu23@gmail.com

⁵Assistant Professor, Department of Computer Science and Information Technology, Vel's Institute of Science, Technology & Advanced Studies (VISTAS), mail id: vishwapriya13@gmail.com

⁶Assistant Professor, Department of Information And Technology, Vel's Institute of Science, Technology & Advanced Studies (VISTAS), mail id: jthangam.scs@vistas.ac.in

⁷Assistant Professor, Department of Data Analysts, Saveetha liberal arts and Science College Saveetha University
Mail.id: yogisarainteract@gmail.com

⁸Assistant Professor, Department of E-edu Govern, S.D.N.B.Vaishnav College For Womens, Chennai-44, Mail.id: ddsterlin@gmail.com

ARTICLE INFO

ABSTRACT

Received: 19 Dec 2024

Revised: 10 Feb 2025

Accepted: 22 Feb 2025

Early detection of heart disease is critical to the patient's survival. An electrocardiogram (ECG) is a test that analyses heartbeat variations. ECG is a test that checks on how your heartbeats vary. Various cardiac diseases can be detected by the deviation of signals from the typical sinus rhythm as well as from mere anomalies. The ECG signal carries minor amplitude variation, thus can cause errors as it may be difficult to make a diagnosis on the cardiac conditions. The only way to preserve the human lives is by the very accurate recognition method. The ECG signals are utilized in an appropriate and accurate way for classifying and predicting the heart diseases through a proposed study in this study. In the study, Convolutional Neural networks (CNN), Visual Geometry Group (VGG) and Logistic Regression (LR) were employed to predict the heart diseases; the results proved out robust and finally, ensemble approaches were developed based on the combination of CNN, VGG, LR with Bidirectional Recurrent Neural Network (BRNN), Gated Recurrent unit, and Long Short term memory which are used to predict heart diseases and performance of each as discussed.

Keywords: Electrocardiogram, Deep Learning, Heart diseases, Convolutional Neural networks, Visual Geometry Group, Logistic Regression (LR)

1. INTRODUCTION:

Heart failure now causes significant medical problems across the world because more people are reaching senior age. Multiple approaches exist today for finding Cardio Vascular diseases. Examining ECG signals makes up one method of diagnosis. ECG detects heart electrical activity through a basic medical tool. The heartbeats become visible through the graph that ECG produces. The electrical patterns in the ECG display each heartbeat through specific peaks and valleys. ECG data provides two important functions: it helps evaluate heart electrical activity by measuring length and detects lung demands on heart muscle areas. ECG records the heart's electrical signals when operating between 0.05 to 100 Hertz with a voltage variation of 1 to 10 mV.

Deep learning algorithms proved better than prior methods for processing ECG signals accurately. Deep learning works within machine learning to process information like the human brain using neural network algorithms. A neural network builds its neurons from extensive data training sessions. The system takes ready data to create an effective deep learning model after all training steps are completed. (Prusty et al, 2024). People believe the available

machine learning methods for detecting congestive heart failure need better performance. Advanced research and development work is needed to improve these systems' performance standards. Researchers need to develop better algorithms to enhance the entire performance of machine learning systems. ECG diagnosis systems with machine learning face problems creating an entirely steady ECG multi-class diagnosis system(Pachiyamman et al,2024).

1.1 ElectroCardiogram(ECG)

An ECG is a non invasive diagnostic and monitoring method that permits to evaluate the heart's electrocardiography activity through biological items' chest during a definite period of time. Samples them up and records the electrodes placed on the skin of specific biological invention parts and continues to store the relevant Record in a particular structure. A graphic electric circuit that exhibits alterations in voltage between electrodes applied on a patient's torso to illustrate heart operation is known as an electrocardiogram. An ECG signal records electrical activity of the heart and some forms of heart diseases which causes improper heart shape and insufficient blood circulation. The electrical currents in circulation are in relation to the action potential of the contractions within the wall of the heart. Electrodes are attached to the body's surface and they are capable of detecting the several potentials which are produced in the body by the spreading currents. A biological transducer refers to the electrodes which are made up of metals and salts. As applied in practice, the ten electrodes are placed at different parts of the body. This is a conventional procedure for acquiring as well as analyzing ECG data(Peiman et al 2024). An average ECG wave in a healthy male adult is illustrated in the figure below; As shown in figure 1 below, an ECG wave of a healthy male adult is composed of;

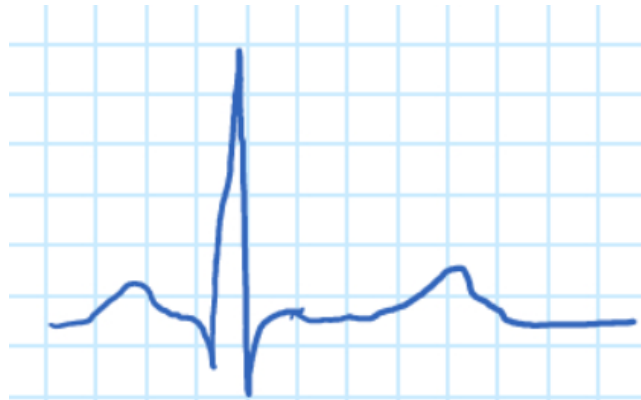


Figure 1. A sample ECG Wave.

The remaining sections are organized as follows. Section 2 summarizes the relevant literature on ML and DL techniques to predicting cardiac disease. Section 3 presents methodology and recommendations, while Section 4 conducts an experimental analysis using performance metrics. Section 5 concludes and discusses future projects.

1.2 Deep Learning and CNN

Deep learning is commonly acknowledged as a branch of machine learning. In essence, deep learning is a strong machine learning technology that works especially well when handling large volumes of data and challenging situations. The performance of an automated classification system that makes use of deep learning algorithms can be greatly affected. Deep learning makes use of multi-layered, intricate neural networks. By allowing the system to recognize various patterns from the input, these networks help the system make better decisions. An algorithm like this simulates how neurons in the human brain communicate and interpret information from the surroundings. One type of network that deep learning algorithms use to predict and classify diverse patterns in a variety of issues is a convolutional neural network (CNN)(Eleyan al,2024).

2. RELATED WORKS

The detection of heart disease through automated Convolutional Neural Networks (CNN) requires the novel application of Scale-Invariant Feature Transform (SIFT) for extracting distinctive ECG signal picture features according to prusty et al. The research team of Zhang et al. developed heart failure NYHA functional classification through deep learning methodology. A deep learning approach with attention mechanism based on CNN-LSTM-SE model performs heart failure classification through 2 to 20-second ECG data segments according to ablation tests.

Initial signal breakdown using Adaptive DWT produces the first feature set for Deep Convolutional Neural Networks (DCNN) to analyze according to [4]. The R-R intervals undergo a first analysis for extraction and then the DCNN performs deep feature extraction to obtain characteristics from the R-R intervals. DCNN extracts the third feature set from QRS waves following their examination in acquired signals. The detection of heart disease happens through the conjunction of all three feature sets while ERBF generates categorized results and accuracy along with F1.Score and AUC evaluate the procedure.

Researchers at Shoughi and his co-team propose a heart condition detection framework that uses the combination of ECG signal analysis through convolutional neural networks together with db2 mother wavelet processing and synthetic minority over-sampling technique (SMOTE) on the MIT-BIH dataset. This study presents two main components: an IoT-based ECG monitoring system which generates data using Node MCU ESP32 along with AD8232 heart rate sensor and develops an intelligent hybrid algorithm to classify the data. The heart disease detection system developed by Golande et al. in 2024 utilizes ensemble methods for feature extraction while deep learning handles the classification process. This framework includes processing ECG data before a hybrid process generates new features which leads to disease categorization. The reliable filtering technique used for signal pre-processing preserved the input signal integrity by reducing artifacts while removing noise. The extraction process for ECG features should integrate both automatic general methods and cardiac cycle particular solutions. Our method employed Convolutional Neural Network (CNN) and Stationary Wavelet Transform (SWT) for implementing hybrid feature engineering.

The Deep Convolutional Neural Network receives training from Chameleon-Sparrow Search Algorithm (CsSA) per Soman et al, 2024 to achieve effective arrhythmia classification through ECG signals. The Daubechies wavelet filter processes the data while both EMD and VMD features are extracted from it. The analysis extracted six vital characteristics from each ECG category through the application of DWT and PCA techniques. A combination of adaptive neuro-fuzzy inference system (ANFIS) classifier was utilized to evaluate the features producing a 99.4% overall classification accuracy with 99.3% average sensitivity and 99.8% average specificity (Abagaro et al, 2024). Nawaz et al. (2024) presented a sophisticated cardiac heartbeat analysis method that unites Support Vector Machine (SVM) with Random Forest (RF), Logistic Regression (LR) and Decision Tree (DT) models for detecting normal or abnormal or COVID-19 heart condition patterns.

Banjarnahor et al, 2024 implemented the decision tree method for decision-making in their machine learning approach according to the author. Decision tree analysis yielded promising outcomes since it detected heart disorders in their early stages effectively 99% of the time. The diagnostic potential of this technique appears strong for medical professionals to detect cardiac abnormalities at an early stage. The CN classifier based on CNN performs Continual Normalization using a higher learning rate for easier standard deviation learning of neuron output. The researchers employed collected features to determine the classification of ECG signals from five significant arrhythmic forms (Begum et al, 2024). CHDdECG represents a deep learning algorithm that extracts pediatric electrocardiogram features automatically alongside wavelet transformation characteristics before integrating them with crucial human-concept features as described by Chen et al., 2024. CHDdECG demonstrated superior diagnostic accuracy against cardiologists for CHD detection while machine-generated electrocardiogram features gained more importance than human-based features so CHDdECG might contain knowledge beyond the comprehension of humans.

Moses et al in 2024 attempted to evaluate how machine learning algorithms handle HRV data as a non-invasive biomarker for heart failure differentiation among healthy patients. The analysis incorporates four classification methods namely support vector machine and k-nearest neighbor (KNN) in combination with naïve Bayes and decision tree (DT). Multiple recent studies from the literature are tabulated in table 1 for utilizing DL approaches

Table 1: Recent Studies in the literature for DL based heart disease prediction

Author Year	Method	Dataset	Score
Prusty et al. , 2024[1]	CNN, SIFT	User collected data with three classes Arrythmia, Congestive Heart Failure and Normal Sinus Rhythm	Accuracy 99.78% F1-Score 99.78%
Pachiyannan et al. ,2024 [2]	CNN, Bi-LSTM and Attention Mechanism (AM)	UCI heart disease dataset	average accuracy rate- 94.28%, precision- 87.54%, recall rate- 96.25%, specificity rate - 91.74%, FPR-8.26%, and FNR- 3.75%.
Zhang et al., 2024 [3]	CNN-LSTM-SE	Care III(MIMIC –III)	accuracy, positive predictive value, sensitivity, and specificity of the NYHA functional classification method were 99.09, 98.9855, 99.033, and 99.649%, respectively.
Dhara et al., 2024 [4]	ADWT, DCNN	User generated dataset	accuracy, F1-score, and AUC.
Shoughi et al., 2024 [5]	SMOTE, CNN, DWT	User generated dataset	Accuracy
Ketu et al., 2024 [6]	Adaboost, Bagging, RF, KNN, SVM	User generated dataset	Accuracy
Golande et al., 2024 [7]	CNN, SWT, LSTM	publicly available multi-disease ECG dataset.	Accuracy
Soman et al., 2024 [8]	CNN, CsSA	MIT-BIH Arrhythmia Database	accuracy, sensitivity, as well as specificity at a rate of 0.947, 0.929, and 0.964
Abagaro et al., 2024 [9]	DWT, PCA, ANFIS	PhysioNet database	accuracy of 99.44%, with average sensitivity and specificity of 99.36% and 99.84%, respectively.
Nawaz et al., 2024 [10]	SVM, RF, LR, DT	Abnormal, normal and COVID -19 affected dataset	Accuracy-77%, 82%, 78%, and 83%.
Zhou et al., 2024 [11]	CNN, FCA	MIT-BIH Arrhythmia Database	Accuracy 99.6 %
Banjarnahor et al.,[12] 2024 [13]	DT	16-lead ECG data	Accuracy -99%
Begum t al., 2024 [13]	CNN	Arrhythmia Database	Accuracy 99.2%
Chen et al., 2024 [14]	DL	ECG dataset	ROC-AUC, Specificity,
Moses et al., 2024[15]	SVM, KNN, DT	PhysioNet database	Precision, Recall, Specificity, F1-Score and Accuracy

3. Methodology

The Study is conducted by combining the various DL approaches as shown below in figure-2. The overview of the each of the models used in this study are explained below.

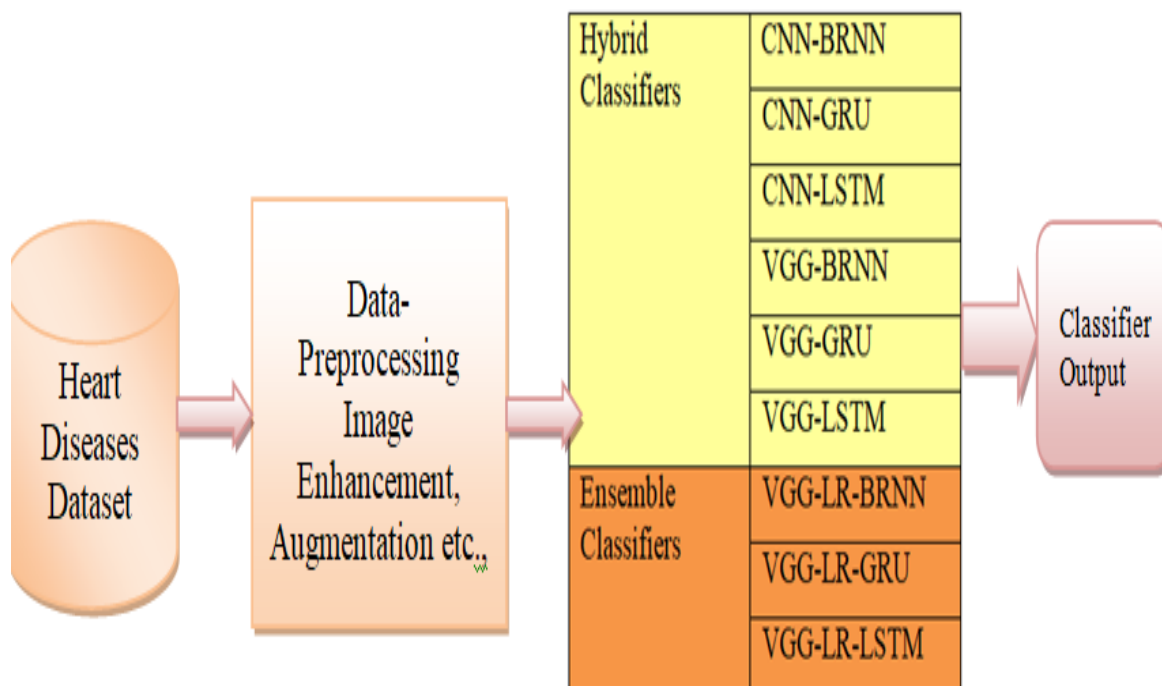


Figure 2 : Workflow of Heart Diseases prediction with DL models

3.1 Convolutional Neural Networks(CNN)

The presented CNN model contains eight total layers that include five CNN layers and one flatten layer followed by two dense layers. The components inside each CNN layer include 1D max pooling and batch normalization alongside 1D CNN. ECG waveforms are transmitted to 500 neurons within the first input signal layer preceding distribution to subsequent layers. A 1D CNN operates to extract relevant information from incoming inputs which makes it the primary layer in the CNN model. After obtaining features in the range of 0 to 1 the batch normalization layer simplifies these features when the features have traveled through. The features are delivered to the max pooling layer as a last step. The maximum values and strongest features are detected by this kernel through its operational method. The procedure decreases data dimensions by filtering out unneeded features. The last four CNN components apply the same stepwise process to their operations. A flatten layer produces one-dimensional column vectors from the features to prepare the data before it reaches the following dense layers consisting of 2000 and 500 fully connected nerves for pre-classification. The output dense layer contains three neurons to represent the three target classes as its final connections according to Eleyan et al, 2024.

3.2 Visual Geometry Group(VGG16)

VGG-16 is an architecture based on VGG, and VGG has the architecture of a CNN network [50]. With 16 connection layers—13 convolution layers and 3 fully connected levels—the VGG- is a DL neural network. The fully connected layers categorize the based on the attributes that the convolution layers have extracted from the input images (Nguyen). The pre-trained VGG16 model can identify a wide range of features because it was trained on a sizable dataset of images. On the other hand, the model's output layer is unique to the dataset used for training (Mahmoud et al, 2023).

3.3 Bi-directional Recurrent Neural Networks (BRNN)

Artificial neural networks (ANN) that analyze input data in both the forward and backward directions are called bi-directional recurrent neural networks, or Bi-RNNs. Natural language processing (NLP) tasks including text classification, named entity recognition, and language translation frequently make use of them. By taking into

account both past and future contexts, they can also capture contextual interdependence in the incoming data. The ultimate output of bi-RNNs is produced by combining the results of two independent RNNs that process the input data in opposite ways. A typical way for merging the outcomes of the forward and reverse RNNs is to combine them. Still, other approaches, such as element-wise addition or multiplication, can also be utilized. The particular task at hand and the intended characteristics of the finished product may influence the combination method selection [Naveeneeth et al.2024,Rahman et al.,2024).

3.4 Gated Recurrent Unit (GRU)

A GRU network has the Gated Recurrent Unit (GRU) cell as the fundamental unit. Their three primary parts are an update gate, a reset gate, and a candidate concealed state. Figure 3 depicts the GRU network with the hidden state and the inputs combined by the cell and passed through the reset and update gates. The output needs to be forecast in the current timestep and this hidden state will be passed through a thick layer with softmax activation, in order to generate output. This acquires a fresh hidden state and advances the next time step. The present GRU cell, an update gate decides as of what the next GRU cell will have an information transfer. It helps to remember only the most important information. The reset gate decides which data needs to not remain in the GRU network, there should be data that needs be removed, and spurious information to discard. In short, it chooses which information to erase at the given time. Unlike RNN networks, GRU networks process time series or spoken language step by step to avoid the hidden state. The vector captures the information of the previous time steps that are relevant in predicting the current time step from the hidden state. The basic idea that gauges the fog for a GRU(Sakla et al, 2023) is allowing the network to decide which of the data from last time step is important to the current time step and which can be omitted.

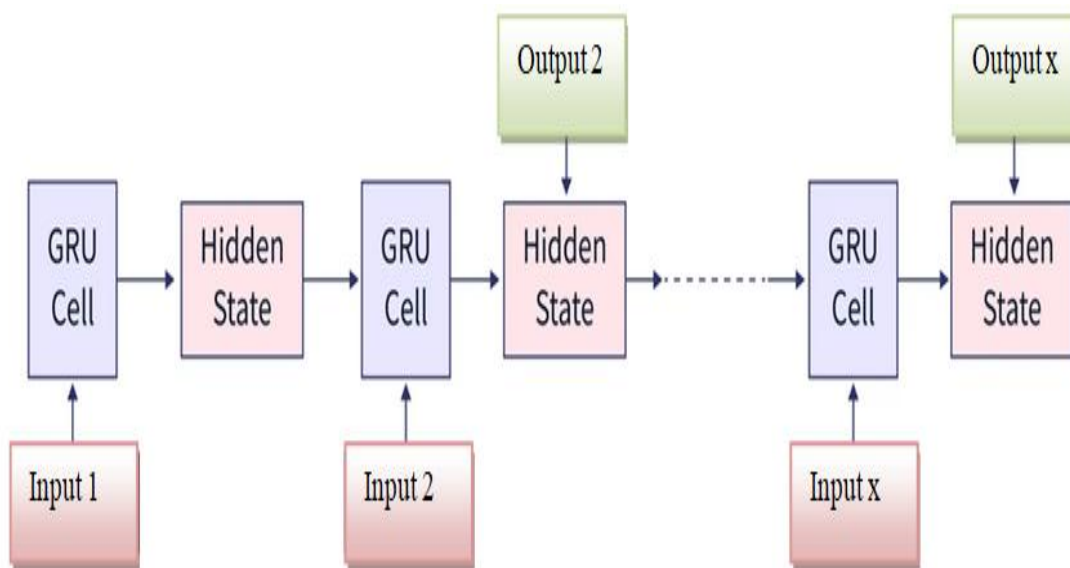



















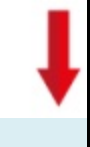






Figure 3 GRU Network










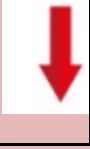


3.5 Long Short Term Memory (LSTM)
















An enhanced RNN called an LSTM employs "gates" to record both short- and long-term memory. These gates aid in preventing the gradient disappearing and explosion problems that arise in conventional RNNs. With gates dubbed "forget gate," "input gate," and "output gate," LSTM boasts a well-built framework. It is made to efficiently absorb and analyze data across a variety of time intervals. A cell state can have information added or removed by an LSTM network. Gates serve as monitoring structures for this process. Gates allow information to flow through them. They are made up of a point-to-point multiplication operation and a sigmoid neural net layer(Dileep et al, 2023, Bhavekar et al.,2024). Table-2 lists the layers used in CNN with BRNN, CNN with GRU and CNN with LSTM.

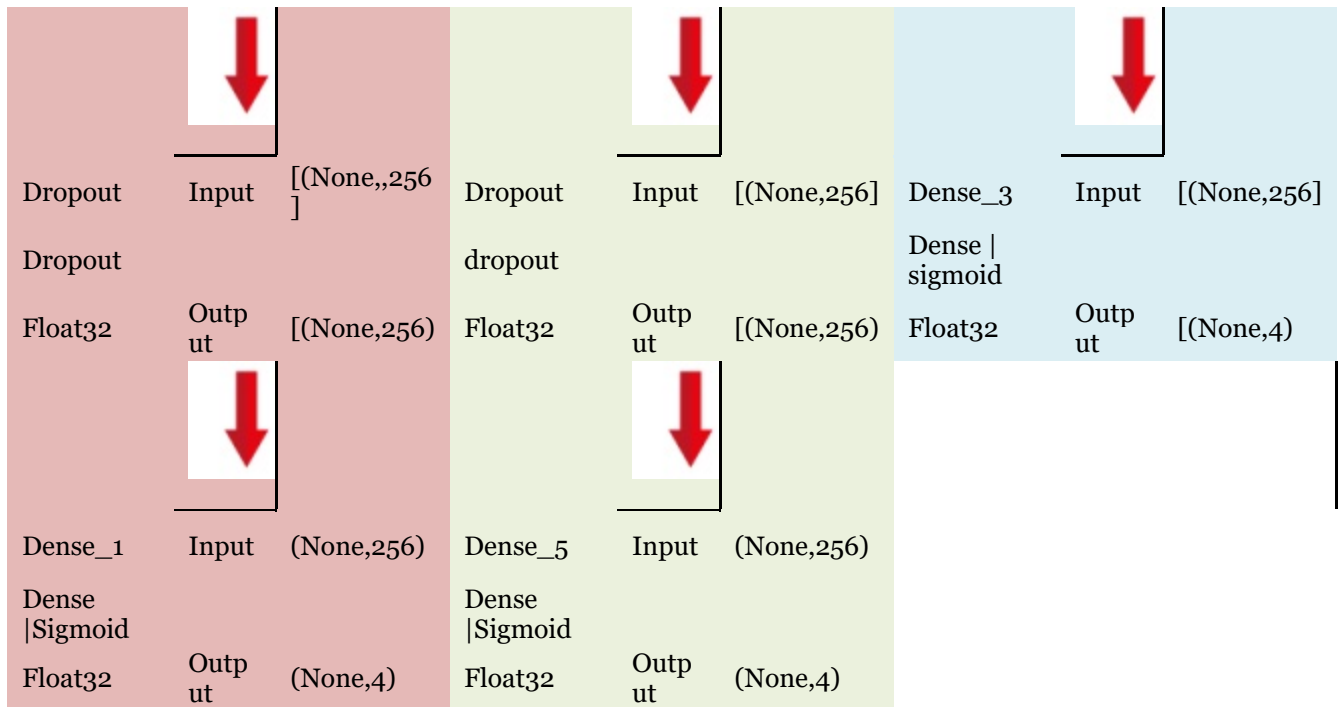
Table 2 Layers in CNN-BRNN, CNN-GRU and CNN-LSTM

CNN-BRNN			CNN-GRU			CNN-LSTM		
CONV2D_12 Input Layer Input	[(None,224,224,3)]		CONV2D_12_input Input Layer Input	[(None,224,224,3)]		CONV2D_6_input Input Layer Input	[(None,224,224,3)]	
Float32 Output	[(None,224,224,3)]		Float32 Output	[(None,224,224,3)]		Float32 Output	[(None,224,224,3)]	
CONV2D_12 Conv 2D Relu Input	[(None,224,224,3)]		CONV2D_12 Conv 2D Relu Input	[(None,224,224,3)]		CONV2D_6 Conv 2D Relu Input	[(None,224,224,3)]	
Float32 Output	[(None,222,222,32)]		Float32 Output	[(None,222,222,32)]		Float32 Output	[(None,222,222,32)]	
Max_pooling2d_12 MaxPooling2D Input	[(None,222,222,32)]		Max_pooling2d_12 MaxPooling2D Input	[(None,222,222,32)]		Max_pooling2d_6 MaxPooling2D Input	[(None,222,222,32)]	
Float32 Output	[(None,111,111,32)]		Float32 Output	[(None,111,111,32)]		Float32 Output	[(None,111,111,32)]	
CONV2D_13 Conv2D ReLu Input	[(None,111,111,32)]		CONV2D_13 Conv2D ReLu Input	[(None,111,111,32)]		CONV2D_7 Conv2D ReLu Input	[(None,111,111,32)]	
Float32 Output	[(None,109,109,32)]		Float32 Output	[(None,109,109,32)]		Float32 Output	[(None,109,109,32)]	

Max_pooling2d_13	Input	[(None,109,109,32)]	Max_pooling2d_13	Input	[(None,109,109,32)]	Max_pooling2d_7	Input	[(None,109,109,32)]
MaxPooling2D			MaxPooling2D			MaxPooling2D		
Float32	Output	[(None,54,54,32)]	Float32	Output	[(None,54,54,32)]	Float32	Output	[(None,54,54,32)]
								
CONV2D_14	Input	[(None,54,54,32)]	CONV2D_14	Input	[(None,54,54,32)]	CONV2D_8	Input	[(None,54,54,32)]
Conv2D ReLu			Conv2D ReLu			Conv2D ReLu		
Float32	Output	[(None,54,54,64)]	Float32	Output	[(None,54,54,64)]	Float32	Output	[(None,52,52,64)]
								
Max_pooling2d_14	Input	[(None,54,54,64)]	Max_pooling2d_14	Input	[(None,54,54,64)]	Max_pooling2d_8	Input	[(None,26,26,64)]
MaxPooling2D			MaxPooling2D			MaxPooling2D		
Float32	Output	[(None,26,26,64)]	Float32	Output	[(None,26,26,64)]	Float32	Output	[(None,26,26,64)]
								
CONV2D_15	Input	[(None,26,26,64)]	CONV2D_15	Input	[(None,26,26,64)]	CONV2D_9	Input	[(None,24,24,64)]
Conv2D ReLu			Conv2D ReLu			Conv2D ReLu		
Float32	Output	[(None,24,24,64)]	Float32	Output	[(None,24,24,64)]	Float32	Output	[(None,24,24,64)]
								
Max_pooling2d_15	Input	[(None,24,24,64)]	Max_pooling2d_15	Input	[(None,24,24,64)]	Max_pooling2d_9	Input	[(None,24,24,64)]

MacxPooli ng2D	Output	[(None, 12,12,,64]	MacxPooli ng2D	Output	[(None, 12,12,,64]	MacxPooli ng2D	Output	[(None, 12,12,,64]
Float32			Float32			Float32		
								
CONV2D_ 16	Input	[(None,12,1 2,64)]	CONV2D_ 16	Input	[(None,12,1 2,64)]	CONV2D_ 10	Input	[(None,12,1 2,64)]
Conv2D ReLu			Conv2D ReLu			Conv2D ReLu		
Float32	Output	[None, 10,10,12]	Float32	Output	[None, 10,10,12]	Float32	Output	[None,10,10 ,128]
								
Max_pooli ng2d_16	Input	[(None, 10,10,128)]	Max_pooli ng2d_16	Input	[(None, 10,10,128)]	Max_pooli ng2d_10	Input	[(None, 10,10,128)]
MacxPooli ng2D	Output	[(None, 5,5,128]	MacxPooli ng2D	Output	[(None, 5,5,128]	MacxPooli ng2D	Output	[(None, 5,5,128]
Float32			Float32			Float32		
								
CONV2D_ 17	Input	[(None, 5,4,128]	CONV2D_ 17	Input	[(None, 5,4,128]	CONV2D_ 11	Input	[(None, 5,5,128]
Conv2D ReLu			Conv2D ReLu			Conv2D ReLu		
Float32	Output	[(None, 3,3,128]	Float32	Output	[(None, 3,3,128]	Float32	Output	[(None, 3,3,128]
								
Max_pooli ng2d_17	Input	[(None, 3,3,128]	Max_pooli ng2d_17	Input	[(None, 3,3,128]	Max_pooli ng2d_11	Input	[(None, 3,3,128]
MacxPooli ng2D	Output	[(None, 1,1,128]	MacxPooli ng2D	Output	[(None, 1,1,128]	MacxPooli ng2D	Output	[(None, 1,1,128]
Float32			Float32			Float32		

								
Flatten_2	Input	[(None, 1,1,128)]	Flatten_2	Input	[(None, 1,1,128)]	Flatten_1	Input	[(None, 1,1,128)]
Flatten			Flatten			Flatten		
Float32	Output	[(None, 128)]	Float32	Output	[(None, 128)]	Float32	Output	[(None, 128)]
								
Repeat_vector	Input	[(None, 128)]	Repeat_vector_2	Input	[(None, 128)]	Repeat_vector_1	Input	[(None, 128)]
RepeatVector			RepeatVector			RepeatVector		
Float32	Output	[(None, 10,128)]	Float32	Output	[(None, 1,128)]	Float32	Output	[(None, 1,128)]
								
Reshape	Input	[(None, 10,128)]	GRU	Input	[(None, 1,128)]	Lstm_1	Input	[(None, 1,128)]
Reshape			GRU tanh			LSTM tanh		
Float32	Output	[(None,1,128)]	Float32	Output	[(None,128)]	Float32	Output	[(None,128)]
								
Bidirectional (LSTM)	Input	[(None,1,128)]	Dense_4	Input	[(None,1,128)]	Dense_2	Input	[(None,128)]
Bidirectional (LSTM)			Dense ReLu			Dense ReLu		
Float32	Output	[(None,,256)]	Float32	Output	[(None,256)]	Float32	Output	[(None,256)]
								
Dense	Input	[(None,256)]	Dense	Input	[(None,,256)]	Dropout_1	Input	[(None,,256)]
Dense ReLu			Dense ReLu			Dropout		
Float32	Output	[(None,,256)]	Float32	Output	[(None,256)]	Float32	Output	[(None,256)]



3.5 Visual Geometry Group(VGG16)

ConvNets, a type of ANN are sometimes referred to as CNN. An input layer, an output layer, and several hidden layers make up a CNN. CNNs, like the VGG16 model are thought to be among the most effective computer vision algorithms available today. The model's developers assessed the networks and used architecture with minuscule (3 × 3) convolution filters to increase the depth, demonstrating a notable advance above previous state-of-the-art setups. They increased the depth to roughly 138 trainable parameters by pushing it to 16–19 weight layers. The sixteen in VGG16 stands for sixteen weighted layers. Although VGG16 contains twenty-one layers total—sixteen convolutional layers, five Max Pooling layers, and three Dense layers—it only has sixteen weight layers, or learnable parameters layers. The most distinctive feature of VGG16 is that its convolution layers of a 3x3 filter with stride 1 are its main focus, rather than having a lot of hyper-parameters. The padding and maxpool layer of a 2x2 filter with stride 2 are also always employed. The arrangement of the convolution and max pool layers is constant throughout the architecture. There are 64 filters in Conv-1 Layer, 128 filters in Conv-2, 256 filters in Conv-3, and 512 filters in Conv-4 and Conv-5. After a stack of convolutional layers, three Fully-Connected (FC) layers are placed: the first two have 4096 channels apiece, while the third conducts 1000-way ILSVRC classification, resulting in 1000 channels (one for each class). Soft-max layer is the last layer(Ainiwaer et al., 2024). The table 3 lists the layer in VGG –BRNN, VGG-GRU and VGG-LSTM methods employed for heart disease prediction.







Table 3 Layers in VGG with BRNN, VGG and LSTM

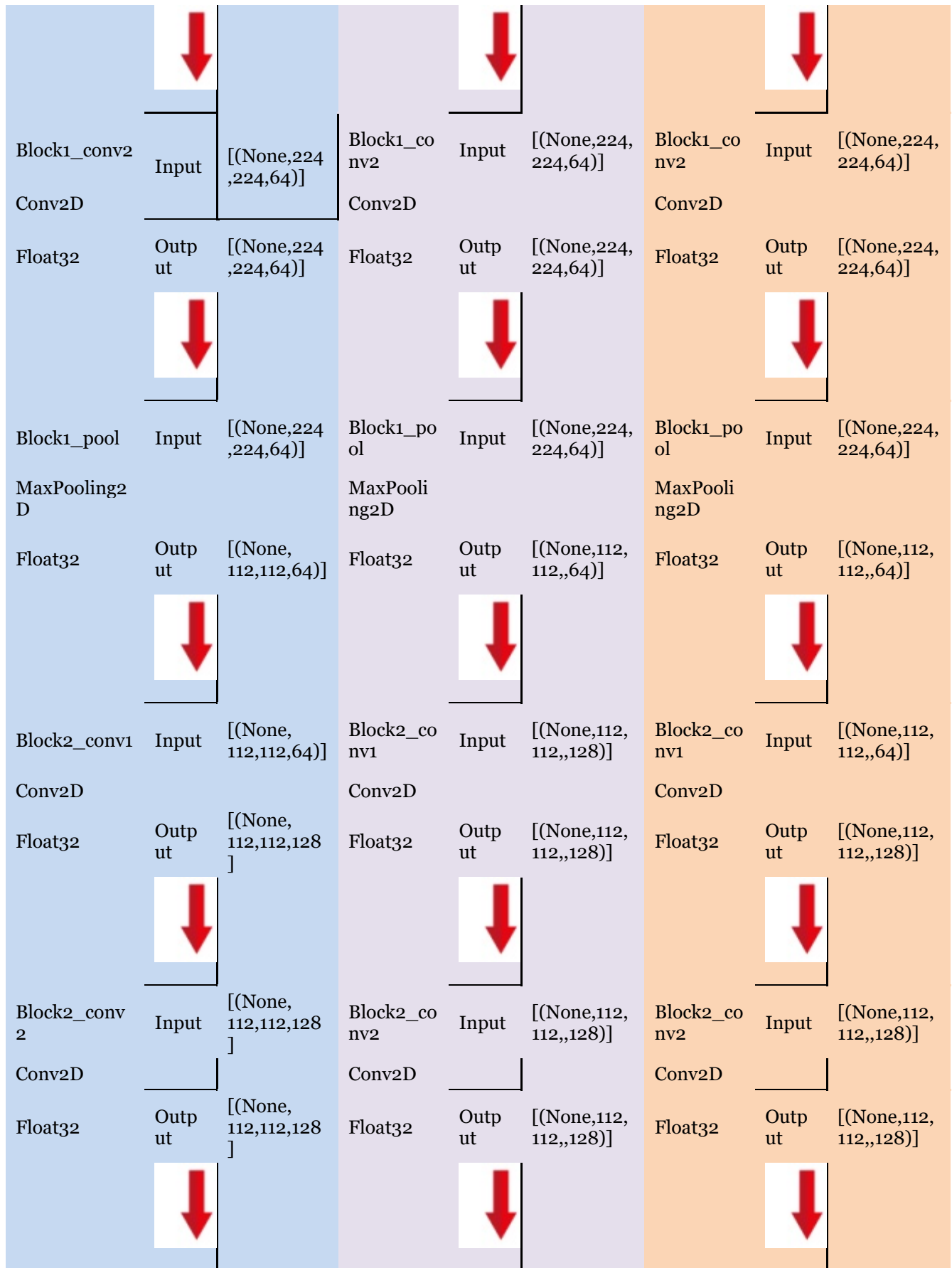
3.6 VGG 16- Cyclical Learning Rate(CLR)













Training time and model effectiveness are influenced by learning rate. The loss function landscape, which is influenced by the dataset and model architecture, determines the learning rate. An ideal learning rate is needed for the model to converge more quickly. The ideal learning rate is one which leads to a sharp decline in loss severity. The learning rate is best when it decreases more steeply.













The learning rate fluctuates cyclically, always reverting to its starting value, much like adaptive learning rate. A lower learning rate can result in the model convergent extremely slowly or diverging from the local minima, whereas a very high learning rate leads the model to vary more. By maintaining a high and low learning rate, cyclical learning rate (CLR) prevents the model from diverging as it jumps from local minima. The learning rate in CLR varies between the base and maximum learning rates. The learning rate can oscillate according to three different functions: the Hann window (sinusoidal), the Welch window (parabolic), or the triangle (linear). A more straightforward method of adjusting the learning rate is the triangle window(Raja et al, 204). The table 4 lists the VGG models with the CLR in BRNN, GRU and LSTM













Table 4 VGG-CLR with BRNN, GRU and LSTM

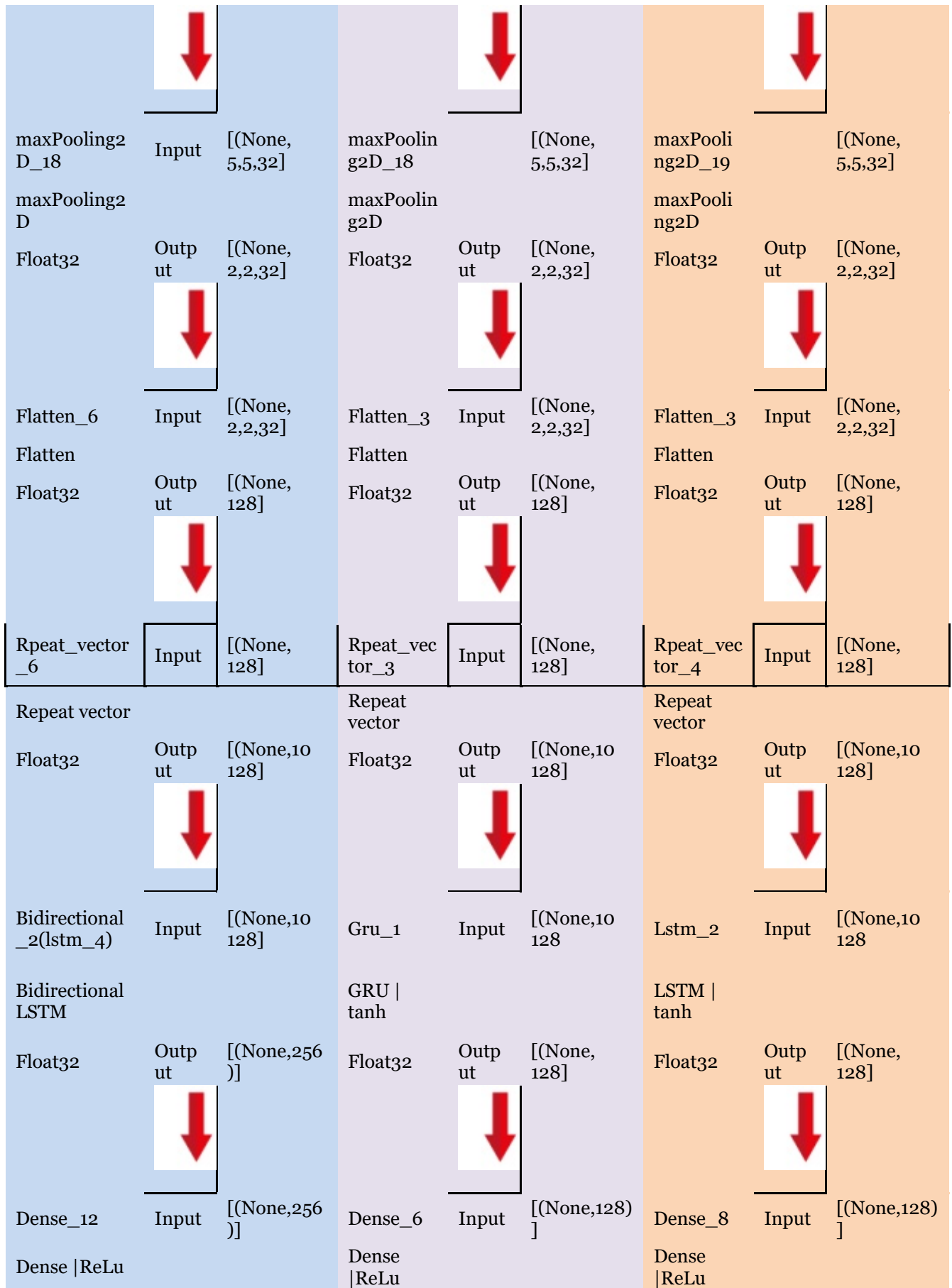
BRNN-VGG			GRU -VGG			LSTM-VGG		
Input_8	Input	[(None,224,224,3)]	Input_2	Input	[(None,224,224,3)]	Input_3	Input	[(None,224,224,3)]
Input Layer			Input Layer			Input Layer		
Float32	Output	[(None,224,224,3)]	Float32	Output	[(None,224,224,3)]	Float32	Output	[(None,224,224,3)]
								
Input_7	Input	[(None,224,224,3)]	Input_1	Input	[(None,224,224,3)]	Input_3	Input	[(None,224,224,3)]
Input Layer			Input Layer			Input Layer		
Float32	Output	[(None,224,224,3)]	Float32	Output	[(None,224,224,32)]	Float32	Output	[(None,224,224,3)]
								
Block1_conv1	Input	[(None,224,224,3)]	Block1_conv1	Input	[(None,224,224,3)]	Block1_conv1	Input	[(None,224,224,3)]
Conv2D			Conv2D			Conv2D		
Float32	Output	[(None,224,224,64)]	Float32	Output	[(None,224,224,64)]	Float32	Output	[(None,224,224,64)]


















Block2_pool	Input	[(None, 112, 112, 128)]	Block2_pool	Input	[(None, 112, 112, 128)]	Block2_pool	Input	[(None, 112, 112, 128)]
Maxpooling2D			Maxpooling2D			Maxpooling2D		
Float32	Output	[(None, 56, 56, 128)]	Float32	Output	[(None, 56, 56, 128)]	Float32	Output	[(None, 56, 56, 128)]
								
Block3_conv1	Input	[(None, 56, 56, 128)]	Block3_conv1	Input	[(None, 56, 56, 128)]	Block3_conv1	Input	[(None, 56, 56, 128)]
Conv2D			Conv2D			Conv2D		
Float32	Output	[(None, 56, 56, 256)]	Float32	Output	[(None, 56, 56, 256)]	Float32	Output	[(None, 56, 56, 256)]
								
Block3_conv2	Input	[(None, 56, 56, 256)]	Block3_conv2	Input	[(None, 56, 56, 256)]	Block3_conv2	Input	[(None, 56, 56, 256)]
Conv2D			Conv2D			Conv2D		
Float32	Output	[(None, 56, 56, 256)]	Float32	Output	[(None, 56, 56, 256)]	Float32	Output	[(None, 56, 56, 256)]
								
Block3_conv3	Input	[(None, 56, 56, 256)]	Block3_conv3	Input	[(None, 56, 56, 256)]	Block3_conv3	Input	[(None, 56, 56, 256)]
Conv2D			Conv2D			Conv2D		
Float32	Output	[(None, 56, 56, 256)]	Float32	Output	[(None, 56, 56, 256)]	Float32	Output	[(None, 56, 56, 256)]
								
Block3_pool	Input	[(None, 56, 56, 256)]	Block3_pool	Input	[(None, 56, 56, 256)]	Block3_pool	Input	[(None, 56, 56, 256)]
MaxPooling2D			MaxPooling2D			MaxPooling2D		

Float32	Output	[(None, 28, 28, 256)]	Float32	Output	[(None, 28, 28, 256)]	Float32	Output	[(None, 28, 28, 256)]
								
Block4_conv1	Input	[(None, 28, 28, 256)]	Block4_conv1	Input	[(None, 28, 28, 256)]	Block4_conv1	Input	[(None, 28, 28, 256)]
Conv2D			Conv2D			Conv2D		
Float32	Output	[(None, 28, 28, 512)]	Float32	Output	[(None, 28, 28, 512)]	Float32	Output	[(None, 28, 28, 512)]
								
Block4_conv2	Input	[(None, 28, 28, 512)]	Block4_conv2	Input	[(None, 28, 28, 512)]	Block4_conv2	Input	[(None, 28, 28, 512)]
Conv2D			Conv2D			Conv2D		
Float32	Output	[(None, 28, 28, 512)]	Float32	Output	[(None, 28, 28, 512)]	Float32	Output	[(None, 28, 28, 512)]
								
Block4_conv3	Input	[(None, 28, 28, 512)]	Block4_conv3	Input	[(None, 28, 28, 512)]	Block4_conv3	Input	[(None, 28, 28, 512)]
Conv2D			Conv2D			Conv2D		
Float32	Output	[(None, 28, 28, 512)]	Float32	Output	[(None, 28, 28, 512)]	Float32	Output	[(None, 28, 28, 512)]
								
Block4_pool	Input	[(None, 28, 28, 512)]	Block4_pool	Input	[(None, 28, 28, 512)]	Block4_pool	Input	[(None, 28, 28, 512)]
MaxPooling2D			MaxPooling2D			MaxPooling2D		
Float32	Output	[(None, 14, 14, 512)]	Float32	Output	[(None, 14, 14, 512)]	Float32	Output	[(None, 14, 14, 512)]












								
Block5_conv1	Input	[(None, 14, 14, 512)]	Block5_conv1	Input	[(None, 14, 14, 512)]	Block5_conv1	Input	[(None, 14, 14, 512)]
Conv2D	Input		Conv2D	Input		Conv2D	Input	
Float32	Output	[(None, 14, 14, 512)]	Float32	Output	[(None, 14, 14, 512)]	Float32	Output	[(None, 14, 14, 512)]
								
Block5_conv2	Input	[(None, 14, 14, 512)]	Block5_conv2	Input	[(None, 14, 14, 512)]	Block5_conv2	Input	[(None, 14, 14, 512)]
Conv2D	Input		Conv2D	Input		Conv2D	Input	
Float32	Output	[(None, 14, 14, 512)]	Float32	Output	[(None, 14, 14, 512)]	Float32	Output	[(None, 14, 14, 512)]
								
Block5_conv3	Input	[(None, 14, 14, 512)]	Block5_conv3	Input	[(None, 14, 14, 512)]	Block5_conv3	Input	[(None, 14, 14, 512)]
Conv2D	Input		Conv2D	Input		Conv2D	Input	
Float32	Output	[(None, 14, 14, 512)]	Float32	Output	[(None, 14, 14, 512)]	Float32	Output	[(None, 14, 14, 512)]
Block5_pool	Input	[(None, 14, 14, 512)]	Block5_pool	Input	[(None, 14, 14, 512)]	Block5_pool	Input	[(None, 14, 14, 512)]
MaxPooling2D	Input		MaxPooling2D	Input		MaxPooling2D	Input	
Float32	Output	[(None, 7, 7, 512)]	Float32	Output	[(None, 7, 7, 512)]	Float32	Output	[(None, 7, 7, 512)]
								
Conv2d_21	Input	[(None, 7, 7, 512)]	Conv2d_18	Input	[(None, 7, 7, 512)]	Conv2d_19	Input	[(None, 7, 7, 512)]
Conv2D ReLU	Input		Conv2D ReLU	Input		Conv2D ReLU	Input	
Float32	Output	[(None, 5, 5, 32)]	Float32	Output	[(None, 5, 5, 32)]	Float32	Output	[(None, 5, 5, 32)]

































Float32	Output	[(None,256)]	Float32	Output	[(None,256)]	Float32	Output	[(None,256)]
								
Dropout_6 dropout	Input	[(None,256)]	Dropout_3 dropout	Input	[(None,256)]	Dropout_4 dropout	Input	[(None,256)]
Float32	Output	[(None,256)]	Float32	Output	[(None,256)]	Float32	Output	[(None,256)]
								
Dense_13 Dense softmax	Input	[(None,256)]	Dense_7 Dense ReLU	Input	[(None,128)]	Dense_9 Dense ReLU	Input	[(None,128)]
Float32	Output	[(None,4)]	Float32	Output	[(None,4)]	Float32	Output	[(None,4)]










BRNN-VGG			GRU –VGG			LSTM-VGG		
Input_12	Input	[(None,2 24,224,3)]	Input_10	Input	[(None,2 24,224,3)]	Input_14	Input	[(None,2 24,224,3)]
Input Layer			Input Layer			Input Layer		
Float32	Output	[(None,2 24,224,3)]	Float32	Output	[(None,2 24,224,3)]	Float32	Output	[(None,2 24,224,3)]
								
Input_11	Input	[(None,2 24,224,3)]	Input_9	Input	[(None,2 24,224,3)]	Input_13	Input	[(None,2 24,224,3)]
Input Layer			Input Layer			Input Layer		
Float32	Output	[(None,2 24,224,3)]	Float32	Output	[(None,2 24,224,3 2)]	Float32	Output	[(None,2 24,224,3)]
								
Block1_ conv1	Input	[(None,2 24,224,3)]	Block1_ conv1	Input	[(None,2 24,224,3)]	Block1_ conv1	Input	[(None,2 24,224,3)]
Conv2D			Conv2D			Conv2D		
Float32	Output	[(None,2 24,224,6 4)]	Float32	Output	[(None,2 24,224,6 4)]	Float32	Output	[(None,2 24,224,6 4)]
								
Block1_ conv2	Input	[(None,2 24,224,6 4)]	Block1_ conv2	Input	[(None,2 24,224,6 4)]	Block1_ conv2	Input	[(None,2 24,224,6 4)]
Conv2D			Conv2D			Conv2D		
Float32	Output	[(None,2 24,224,6 4)]	Float32	Output	[(None,2 24,224,6 4)]	Float32	Output	[(None,2 24,224,6 4)]
Block1_ pool	Input	[(None,2 24,224,6 4)]	Block1_ pool	Input	[(None,2 24,224,6 4)]	Block1_ pool	Input	[(None,2 24,224,6 4)]
MaxPool ing2D			MaxPool ing2D			MaxPool ing2D		

Float32	Output	[(None, 112,112, 64)]	Float32	Output	[(None,1 12,112,,6 4)]	Float32	Output	[(None,1 12,112,,6 4)]
								
Block2_ conv1	Input	[(None, 112,112,6 4)]	Block2_ conv1	Input	[(None,1 12,112,,1 28)]	Block2_ conv1	Input	[(None,1 12,112,,6 4)]
Conv2D			Conv2D			Conv2D		
Float32	Output	[(None, 112,112,1 28)]	Float32	Output	[(None,1 12,112,,1 28)]	Float32	Output	[(None,1 12,112,,1 28)]
								
Block2_ conv2	Input	[(None, 112,112,1 28)]	Block2_ conv2	Input	[(None,1 12,112,,1 28)]	Block2_ conv2	Input	[(None,1 12,112,,1 28)]
Conv2D			Conv2D			Conv2D		
Float32	Output	[(None, 112,112,1 28)]	Float32	Output	[(None,1 12,112,,1 28)]	Float32	Output	[(None,1 12,112,,1 28)]
								
Block2_ pool	Input	[(None, 112,112,1 28)]	Block2_ pool	Input	[(None,1 12,112,,1 28)]	Block2_ pool	Input	[(None,1 12,112,,1 28)]
Maxpooli ng2D			Maxpooli ng2D			Maxpooli ng2D		
Float32	Output	[(None, 56 56,128)]	Float32	Output	[(None,5 6,56,128)]	Float32	Output	[(None,5 6,56,128)]
								
Block3_ conv1	Input	[(None, 56 56,128)]	Block3_ conv1	Input	[(None,5 6,56,128)]	Block3_ conv1	Input	[(None,5 6,56,128)]
Conv2D			Conv2D			Conv2D		
Float32	Output	[(None, 56 56,256)]	Float32	Output	[(None,5 6,56,256)]	Float32	Output	[(None,5 6,56,256)]
								
Block3_ conv2	Input	[(None, 56 56,256)]	Block3_ conv2	Input	[(None, 56 56,256)]	Block3_ conv2	Input	[(None, 56 56,256)]

Conv2D			Conv2D			Conv2D		
Float32	Output	[(None, 56, 56,256)]	Float32	Output	[(None, 56, 56,256)]	Float32	Output	[(None, 56, 56,256)]
Block3_conv3	Input	[(None, 56, 56,256)]	Block3_conv3	Input	[(None, 56, 56,256)]	Block3_conv3	Input	[(None, 56, 56,256)]
Conv2D			Conv2D			Conv2D		
Float32	Output	[(None, 56, 56,256)]	Float32	Output	[(None, 56, 56,256)]	Float32	Output	[(None, 56, 56,256)]
								
Block3_pool	Input	[(None, 56, 56,256)]	Block3_pool	Input	[(None, 56, 56,256)]	Block3_pool	Input	[(None, 56, 56,256)]
MaxPooling2D			MaxPooling2D			MaxPooling2D		
Float32	Output	[(None, 28, 28,256)]	Float32	Output	[(None, 28, 28,256)]	Float32	Output	[(None, 28, 28,256)]
								
Block4_conv1	Input	[(None, 28, 28,256)]	Block4_conv1	Input	[(None, 28, 28,256)]	Block4_conv1	Input	[(None, 28, 28,256)]
Conv2D			Conv2D			Conv2D		
Float32	Output	[(None, 28, 28,512)]	Float32	Output	[(None, 28, 28,512)]	Float32	Output	[(None, 28, 28,512)]
								
Block4_conv2	Input	[(None, 28, 28,512)]	Block4_conv2	Input	[(None, 28, 28,512)]	Block4_conv2	Input	[(None, 28, 28,512)]
Conv2D			Conv2D			Conv2D		
Float32	Output	[(None, 28, 28,512)]	Float32	Output	[(None, 28, 28,512)]	Float32	Output	[(None, 28, 28,512)]
								
Block4_conv3	Input	[(None, 28, 28,512)]	Block4_conv3	Input	[(None, 28, 28,512)]	Block4_conv3	Input	[(None, 28, 28,512)]

Conv2D			Conv2D			Conv2D		
Float32	Output	[(None, 28, 28, 512)]	Float32	Output	[(None, 28, 28, 512)]	Float32	Output	[(None, 28, 28, 512)]
								
Block4_pool	Input	[(None, 28, 28, 512)]	Block4_pool	Input	[(None, 28, 28, 512)]	Block4_pool	Input	[(None, 28, 28, 512)]
MaxPooling2D			MaxPooling2D			MaxPooling2D		
Float32	Output	[(None, 14, 14, 512)]	Float32	Output	[(None, 14, 14, 512)]	Float32	Output	[(None, 14, 14, 512)]
								
Block5_conv1	Input	[(None, 14, 14, 512)]	Block5_conv1	Input	[(None, 14, 14, 512)]	Block5_conv1	Input	[(None, 14, 14, 512)]
Conv2D	Input		Conv2D			Conv2D		
Float32	Output	[(None, 14, 14, 512)]	Float32	Output			Output	[(None, 14, 14, 512)]
								
Block5_conv2	Input	[(None, 14, 14, 512)]	Block5_conv2	Input	[(None, 14, 14, 512)]	Block5_conv2	Input	[(None, 14, 14, 512)]
Conv2D			Conv2D			Conv2D		
Float32	Output	[(None, 14, 14, 512)]	Float32	Output	[(None, 14, 14, 512)]	Float32	Output	[(None, 14, 14, 512)]
								
Block5_conv3	Input	[(None, 14, 14, 512)]	Block5_conv3	Input	[(None, 14, 14, 512)]	Block5_conv3	Input	[(None, 14, 14, 512)]
Conv2D			Conv2D			Conv2D		
Float32	Output	[(None, 14, 14, 512)]	Float32	Output	[(None, 14, 14, 512)]	Float32	Output	[(None, 14, 14, 512)]
								









Block5_pool	Input	[(None, 14, 14, 512)]	Block5_pool	Input	[(None, 14, 14, 512)]	Block5_pool	[(None, 14, 14, 512)]	Float32
MaxPooling2D			MaxPooling2D			MaxPooling2D		
Float32	Output	[(None, 7, 7, 512)]	Float32	Output	[(None, 7, 7, 512)]	Float32	Output	[(None, 7, 7, 512)]
								
Conv2d_23	Input	[(None, 7, 7, 512)]	Conv2d_22		[(None, 7, 7, 512)]	Conv2d_24		[(None, 7, 7, 512)]
Conv2D ReLu			Conv2D ReLu			Conv2D ReLu		
Float32	Output	[(None, 5, 5, 32)]	Float32		[(None, 5, 5, 32)]	Float32		[(None, 5, 5, 32)]
								
maxPooling2D_23	Input	[(None, 5, 5, 32)]	maxPooling2D_22		[(None, 5, 5, 32)]	maxPooling2D_24		[(None, 5, 5, 32)]
maxPooling2D			maxPooling2D			maxPooling2D		
Float32	Output	[(None, 2, 2, 32)]	Float32	Output	[(None, 2, 2, 32)]	Float32	Output	[(None, 2, 2, 32)]
								
Flatten_8	Input	[(None, 2, 2, 32)]	Flatten_7	Input	[(None, 2, 2, 32)]	Flatten_9	Input	[(None, 2, 2, 32)]
Flatten			Flatten			Flatten		
Float32	Output	[(None, 128)]	Float32	Output	[(None, 128)]	Float32	Output	[(None, 128)]
								
Rpeat_vector_8	Input	[(None, 128)]	Rpeat_vector_7	Input	[(None, 128)]	Rpeat_vector_9	Input	[(None, 128)]
Repeat vector			Repeat vector			Repeat vector		
Float32	Output	[(None, 10, 128)]	Float32	Output	[(None, 10, 128)]	Float32	Output	[(None, 10, 128)]
								
Bidirectional_3(lstm_5)	Input	[(None, 10, 128)]	Gru_2	Input	[(None, 10, 128)]	Lstm_6	Input	[(None, 10, 128)]













Bidirectional LSTM			GRU tanh			LSTM tanh		
Float32	Output	[(None,256)]	Float32	Output	[(None,128)]	Float32	Output	[(None,128)]
								
Dense_16	Input	[(None,256)]	Dense_14	Input	[(None,128)]	Dense_18	Input	[(None,128)]
Dense ReLu			Dense ReLu			Dense ReLu		
Float32	Output	[(None,256)]	Float32	Output	[(None,256)]	Float32	Output	[(None,256)]
								
Dropout_8	Input	[(None,256)]	Dropout_7	Input	[(None,256)]	Dropout_9	Input	[(None,256)]
Dropout			dropout			dropout		
Float32	Output	[(None,256)]	Float32	Output	[(None,256)]	Float32	Output	[(None,256)]
								
Dense_17	Input	[(None,256)]	Dense_15	Input	[(None,256)]	Dense_19	Input	[(None,128)]
Dense softmax			Dense ReLu			Dense ReLu		
Float32	Output	[(None,4)]	Float32	Output	[(None,4)]	Float32	Output	[(None,4)]

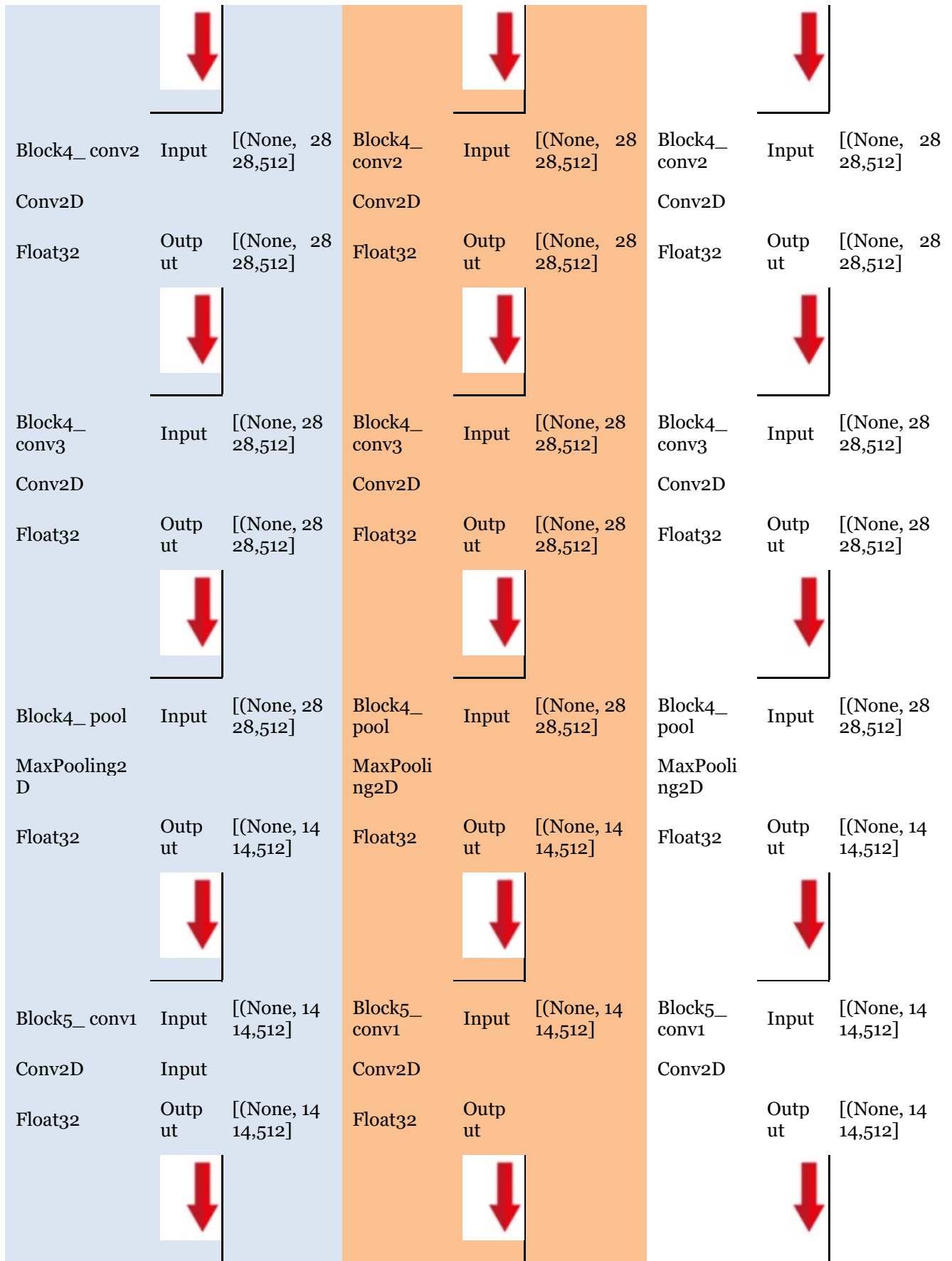
3.7 VGG-Attention Mechanism(AM)













In order to enhance the encoder-decoder model's machine translation performance, the attention mechanism was included. By combining all of the encoded input vectors into a weighted combination and assigning the highest weights to the most relevant vectors, the attention mechanism was designed to allow the decoder to make flexible use of the most relevant segments of the input sequence(Xu et al, 2024) . The table 5 lists the VGG with attention mechanism (AM) in BRNN, GRU and LSTM.

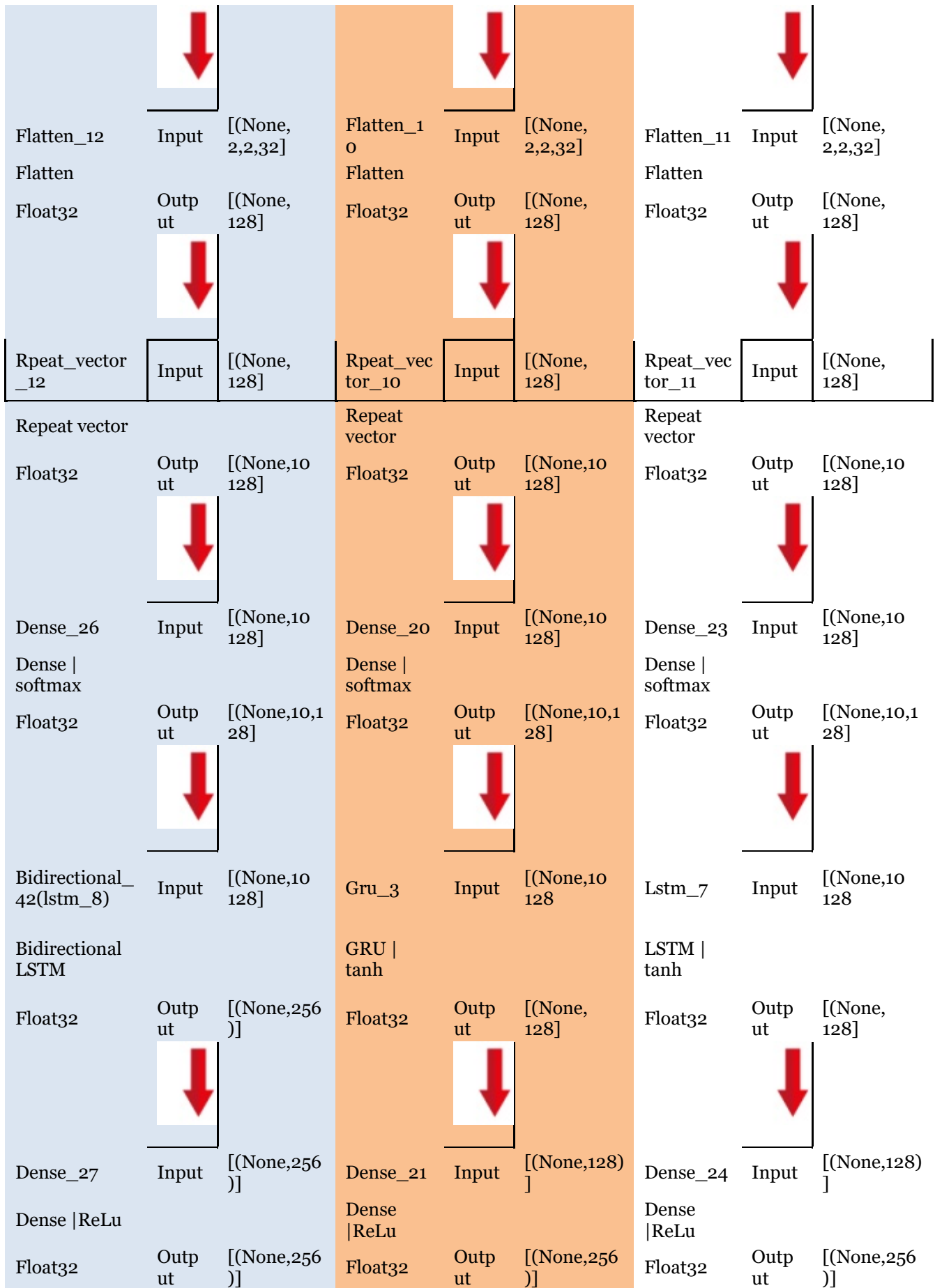
Table 5 VGG-AM with BRNN, GRU and LSTM

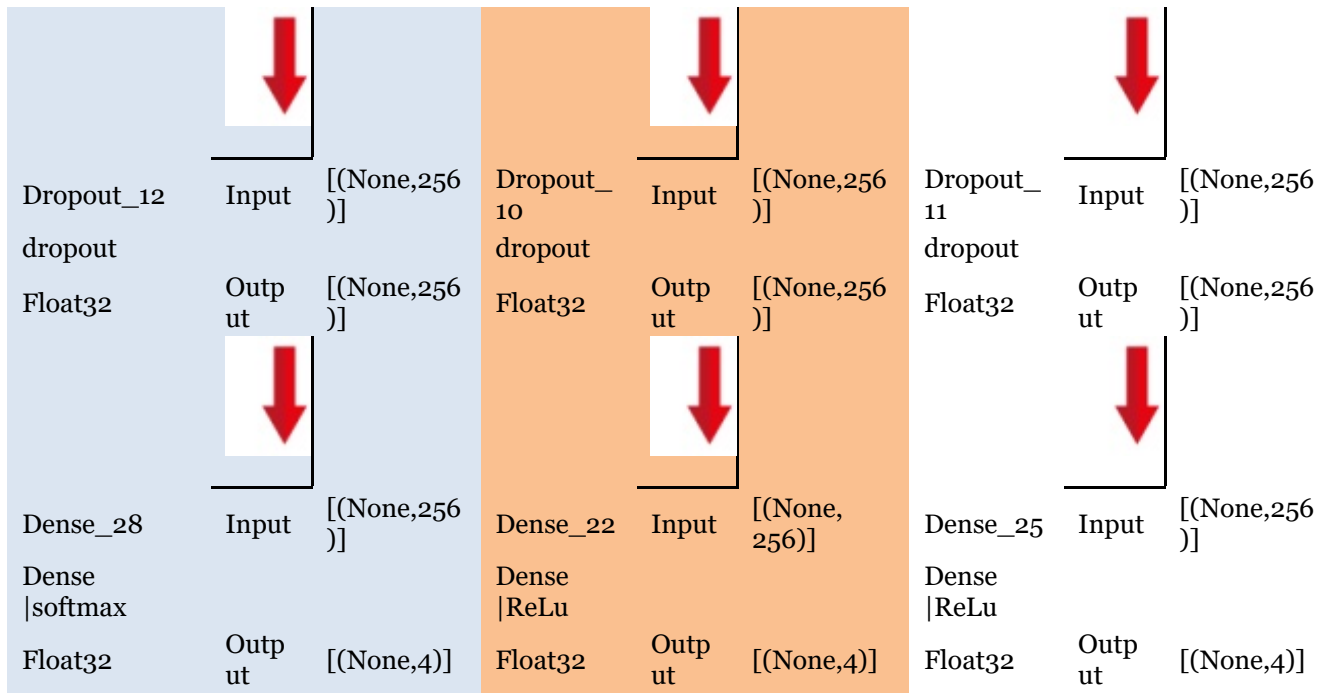
VGG-BRNN-AM	VGG-GRU-AM	VGG-LSTM-AM						
Input_20 Input Layer	Input [(None,224,224,3)]	Input_16 Input Layer	Input	[(None,224,224,3)]	Input_18 Input Layer	Input	[(None,224,224,3)]	
Float32	Output [(None,224,224,3)] 	Float32	Output	[(None,224,224,3)]	Float32	Output	[(None,224,224,3)]	
Input_19 Input Layer	Input [(None,224,224,3)]	Input_15 Input Layer	Input	[(None,224,224,3)]	Input_17 Input Layer	Input	[(None,224,224,3)]	
Float32	Output [(None,224,224,3)] 	Float32	Output	[(None,224,224,32)]	Float32	Output	[(None,224,224,3)]	
Block1_conv1 Conv2D	Input [(None,224,224,3)]	Block1_conv1 Conv2D	Input	[(None,224,224,3)]	Block1_conv1 Conv2D	Input	[(None,224,224,3)]	
Float32	Output [(None,224,224,64)] 	Float32	Output	[(None,224,224,64)]	Float32	Output	[(None,224,224,64)]	
Block1_conv2 Conv2D	Input [(None,224,224,64)]	Block1_conv2 Conv2D	Input	[(None,224,224,64)]	Block1_conv2 Conv2D	Input	[(None,224,224,64)]	
Float32	Output [(None,224,224,64)] 	Float32	Output	[(None,224,224,64)]	Float32	Output	[(None,224,224,64)]	

Block1_pool	Input	[(None,224,224,64)]	Block1_pool	Input	[(None,224,224,64)]	Block1_pool	Input	[(None,224,224,64)]
MaxPooling2D			MaxPooling2D			MaxPooling2D		
Float32	Output	[(None,112,112,64)]	Float32	Output	[(None,112,112,64)]	Float32	Output	[(None,112,112,64)]
								
Block2_conv1	Input	[(None,112,112,64)]	Block2_conv1	Input	[(None,112,112,128)]	Block2_conv1	Input	[(None,112,112,64)]
Conv2D			Conv2D			Conv2D		
Float32	Output	[(None,112,112,128)]	Float32	Output	[(None,112,112,128)]	Float32	Output	[(None,112,112,128)]
								
Block2_conv2	Input	[(None,112,112,128)]	Block2_conv2	Input	[(None,112,112,128)]	Block2_conv2	Input	[(None,112,112,128)]
Conv2D			Conv2D			Conv2D		
Float32	Output	[(None,112,112,128)]	Float32	Output	[(None,112,112,128)]	Float32	Output	[(None,112,112,128)]
								
Block2_pool	Input	[(None,112,112,128)]	Block2_pool	Input	[(None,112,112,128)]	Block2_pool	Input	[(None,112,112,128)]
Maxpooling2D			Maxpooling2D			Maxpooling2D		
Float32	Output	[(None,56,56,128)]	Float32	Output	[(None,56,56,128)]	Float32	Output	[(None,56,56,128)]
								
Block3_conv1	Input	[(None,56,56,128)]	Block3_conv1	Input	[(None,56,56,128)]	Block3_conv1	Input	[(None,56,56,128)]
Conv2D			Conv2D			Conv2D		



Block5_conv2	Input	[(None, 14, 14, 512)]	Block5_conv2	Input	[(None, 14, 14, 512)]	Block5_conv2	Input	[(None, 14, 14, 512)]
Conv2D			Conv2D			Conv2D		
Float32	Output	[(None, 14, 14, 512)]	Float32	Output	[(None, 14, 14, 512)]	Float32	Output	[(None, 14, 14, 512)]
								
Block5_conv3	Input	[(None, 14, 14, 512)]	Block5_conv3	Input	[(None, 14, 14, 512)]	Block5_conv3	Input	[(None, 14, 14, 512)]
Conv2D			Conv2D			Conv2D		
Float32	Output	[(None, 14, 14, 512)]	Float32	Output	[(None, 14, 14, 512)]	Float32	Output	[(None, 14, 14, 512)]
								
Block5_pool	Input	[(None, 14, 14, 512)]	Block5_pool	Input	[(None, 14, 14, 512)]	Block5_pool	[(None, 14, 14, 512)]	Float32
MaxPooling2D			MaxPooling2D			MaxPooling2D		
Float32	Output	[(None, 7, 7, 512)]	Float32	Output	[(None, 7, 7, 512)]	Float32	Output	[(None, 7, 7, 512)]
								
Conv2d_27	Input	[(None, 7, 7, 512)]	Conv2d_25	[(None, 7, 7, 512)]	Conv2d_19	[(None, 7, 7, 512)]		
Conv2D ReLU			Conv2D ReLU		Conv2D ReLU			
Float32	Output	[(None, 5, 5, 32)]	Float32	[(None, 5, 5, 32)]	Float32	[(None, 5, 5, 32)]		
								
maxPooling2D_27	Input	[(None, 5, 5, 32)]	maxPooling2D_25	[(None, 5, 5, 32)]	maxPooling2D_19	[(None, 5, 5, 32)]		
maxPooling2D			maxPooling2D		maxPooling2D			
Float32	Output	[(None, 2, 2, 32)]	Float32	Output	[(None, 2, 2, 32)]	Float32	Output	[(None, 2, 2, 32)]





4. Findings and Discussions

4.1 Dataset Description

The study uses a “cardiovascular ECG Images” dataset, which is freely downloadable from Kaggle. The dataset consists of the ECG images of patients with Myocardial Infarction, abnormal heartbeat and normal heart beat. The sample ECG images from the different category are shown in figure 4



Figure 4 Sample ECG Images (a)Myocardial Infarction (b)Abnormal Heart Beat (c) Normal heart beat

4.2 Performance Metrics

Accuracy, precision, recall, F1-Score, Mean Absolute Error (MAE), Mean Squared Error (MAE), and Area Under Curve-Receiver Operating Characteristics (AUC-ROC) are among the performance indicators used to evaluate the efficacy of the suggested model.

(i) Accuracy

One typical performance indicator used in classification tasks is accuracy. Accuracy is defined as the proportion of correctly identified cases to all items in the dataset. The Metric is calculated by dividing the total number of predictions the model made by the total number of accurate forecasts.

$$Accuracy = \frac{TN+TP}{TN+FP+TP+FN} \tag{1}$$

(ii) Precision

Precision, also referred to as positive predictive value, is the ratio of correctly predicted positive observations to the total number of expected positives.

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

(iii) Recall (Sensitivity, True Positive Rate)

Recall is defined as the proportion of correctly anticipated positive observations to all observations made in the actual class.

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

(iv) F1 Score

The F1 score is the harmonic mean of recall and precision. The F1-score is a statistic that takes precision and recall into account (Balaji et al., 2024). It has the following definition:

$$F1 \text{ Score} = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (4)$$

(v) Mean Absolute Error(MAE)

Within the field of ML/DL, absolute error denotes the extent of discrepancy between an observation's predicted value and its actual value. The size of mistakes for the entire group is determined by the MAE, which takes the average of absolute errors for a set of observations and forecasts. MAE is also known as the L1 loss function.

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (5)$$

Where n is the number of observation, y_i is the actual value of the i th observation, and \hat{y}_i is the predicted value of the i th observation.

(vi) Mean Squared Error(MSE)

In machine learning, especially in regression analysis MSE is a primary statistic. It is an essential instrument for assessing the efficacy and precision of predictive models. The regressive loss metric mean square error (MSE) is used. The difference between the model's predictions and the ground truth, squared and averaged over the dataset, is the MSE. It is employed to determine the degree to which the expected and actual values agree. Similar to RMSE, a smaller value denotes a better fit and penalizes significant errors or outliers severely.

$$MSE = \frac{\sum(y_i - p_i)^2}{n} \quad (6)$$

where y_i is the i th observed value, p_i is the corresponding predicted value for y_i , and n is the number of observations. The symbol κ denotes the execution of a summation across all values of i (Kamnath et al., 2024)

(vii) The AUC-ROC Curve

It is employed to evaluate the performance of the classification issues at different threshold values. AUC is a metric or degree of separability as opposed to ROC, which is a probability curve. It illustrates the degree to which the model can discriminate based on class. AUC evaluates the quality of a model. The bigger the AUC, the more adept the model is in distinguishing between areas with and without damage. TPR is shown on the y-axis and FPR is shown on the x-axis to create the ROC curve (Rimal et al, 2024). The True Positive Rate (TPR) and False Positive Rate (FPR) are as follows.

$$TPR \text{ or Recall or Sensitivity} = \frac{TP}{TP+FN} \tag{6}$$

FPR =1-Specificity

$$\text{Where Specificity} = \frac{TN}{TN+FP} \tag{7}$$

$$FRP = \frac{FP}{TN+FP} \tag{8}$$

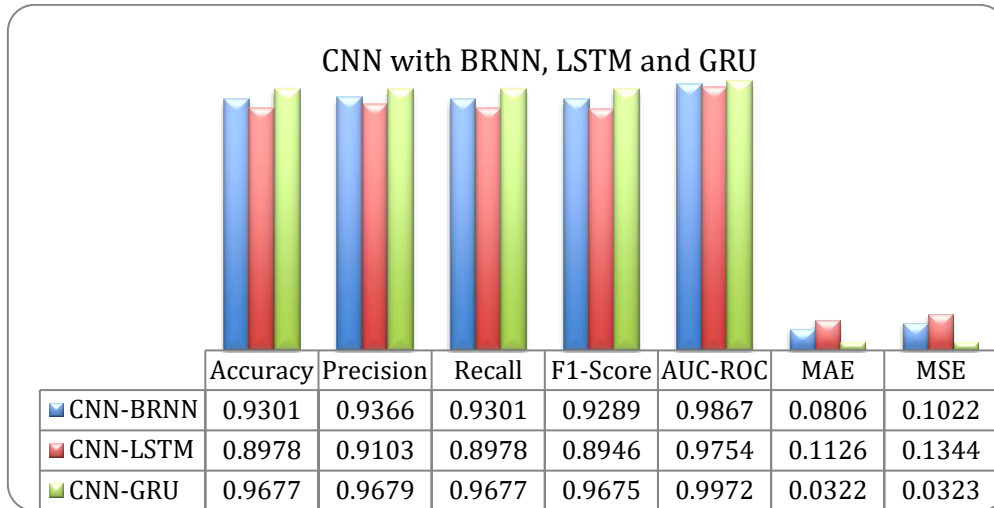


Figure 5 : Performance Indicators (CNN with BRNN, LSTM and GRU)

Figure 5 shows the performances of CNN-BRNN, CNN-GRU and CNN LSTM with regard to accuracy, precision, recall, F1-Score, AUC-ROC, MAE and MSE. It is found that the CNN with GRU outperforms the other combination of methods. The error values are found to be less in the case of CNN-GRU.

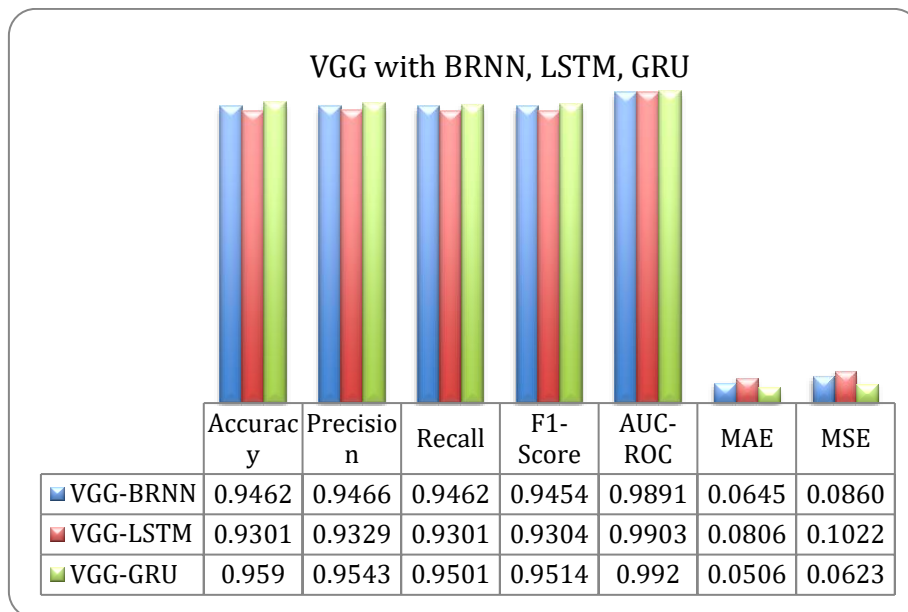


Figure 6 Performance Indicators (VGG with BRNN, LSTM and GRU)

From the figure 6 , it is clear that the VGG with GRU outperforms the other two methods in predicting the heart diseases. The loss value is found to be less in this case.

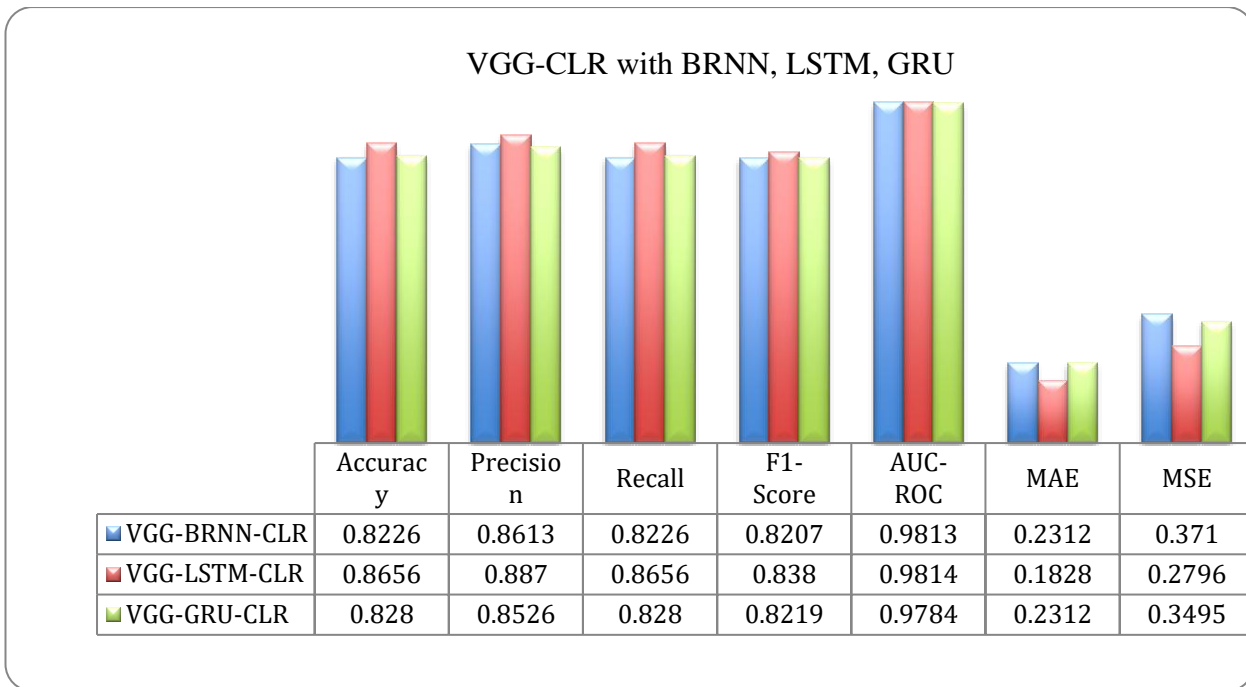


Figure 7 Performance Indicators (VGG-CLR with BRNN, LSTM and GRU)

From the figure 7, it is evident that the VGG with the cyclical learning rate with LSTM performs better than the other two methods.

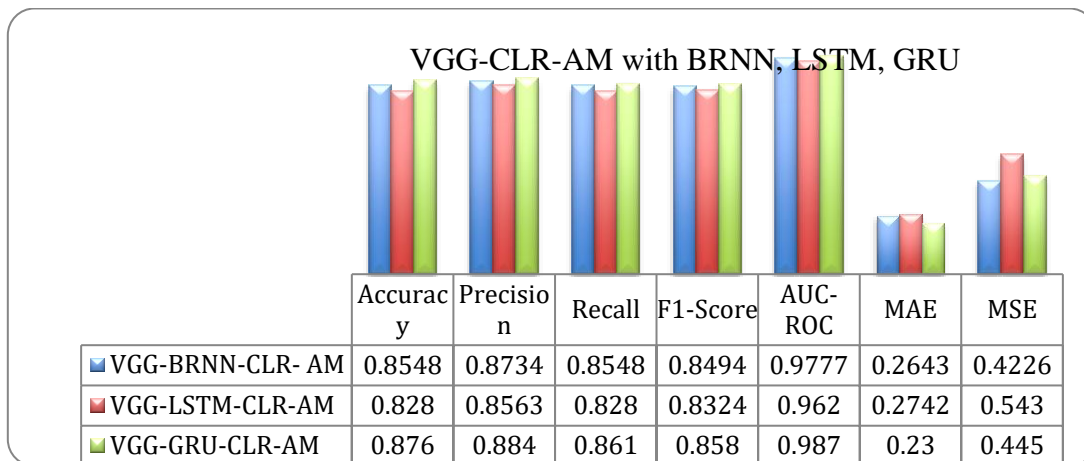


Figure 8: Performance Indicators (VGG-CLR-AM with BRNN, LSTM and GRU)

From the figure 8, it is found that VGG- GRU improved with the learning rate and attention mechanism outperforms the VGG-LSTM and VGG BRNN with a learning rate and attention mechanism. The MAE and MSE also found to be less compared to the other two methods. GRU is faster and requires fewer memory resources than LSTM; however on datasets with longer sequences, LSTM performs better. GRU is significantly more efficient than LSTM. The notion that a human pays more attention to specific areas of its environment while identifying something in it gave rise to the AM in DL approaches. Natural language processing makes extensive use of this model structure in a variety of applications. Nonetheless, a small number of research works have combined with AM and GRU to predict heart diseases. Moreover, the AM executes prediction processes by focusing on meaningful information rather than considering all elements equally(RAo et al,2024). The overall pattern demonstrated that, even if the differences between classifiers were substantial, the use of CLR and AM enhance DL models' discriminative capacity for heart disease prediction.

V. CONCLUSION AND FUTURE WORK

This research combines various DL algorithms with the purpose of improving their operational outcomes. This research integrates CNN and VGG models together with LSTM and CNN-BRNN along with CNN-GRU. The research results show CNN-GRU and VGG-GRU alone deliver better performance than other investigated approaches. The CLR and AM serve as components in a study to boost the classifier performance capabilities. The VGG hybrid classifiers generate better results than CNN classifiers when CLR and AM are applied to them. Multiple ways exist to boost the performance of hybrid classifiers by implementing CLR alongside AM algorithms along with optimized features. The research can continue by examining how the hybrid classifiers perform with optimizer techniques and feature extraction methods.

REFERENCES

1. Prusty, Manas Ranjan, et al. "Scalar invariant transform based deep learning framework for detecting heart failures using ECG signals." *Scientific Reports* 14.1 (2024): 2633.
2. Pachiyannan, Prabu, et al. "A Novel Machine Learning-Based Prediction Method for Early Detection and Diagnosis of Congenital Heart Disease Using ECG Signal Processing." *Technologies* 12.1 (2024): 4.
3. Zhang, Chang-Jiang, et al. "Heart failure classification using deep learning to extract spatiotemporal features from ECG." *BMC Medical Informatics and Decision Making* 24.1 (2024): 17.
4. Dhara, Sanjib Kumar, Nilankar Bhanja, and Prabodh Khampariya. "An adaptive heart disease diagnosis via ECG signal analysis with deep feature extraction and enhanced radial basis function." *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization* 11.7 (2024): 2245927.
5. Shoughi, Armin, et al. "Automatic ECG classification using discrete wavelet transform and one-dimensional convolutional neural network." *Computing* 106.4 (2024): 1227-1248.
6. Ketu, Shwet, and Pramod Kumar Mishra. "An intelligent hybrid classification model for heart disease detection using imbalanced electrocardiogram signals." *The Journal of Supercomputing* 80.3 (2024): 4286-4308.
7. Golande, Avinash L., and T. Pavankumar. "Electrocardiogram-based heart disease prediction using hybrid deep feature engineering with sequential deep classifier." *Multimedia Tools and Applications* (2024): 1-33.
8. Soman, Anila, and R. Sarath. "Optimization-enabled deep convolutional neural network with multiple features for cardiac arrhythmia classification using ECG signals." *Biomedical Signal Processing and Control* 92 (2024): 105964.
9. Abagaro, Ahmed 3, et al. "Automated ECG Signals Analysis for Cardiac Abnormality Detection and Classification." *Journal of Electrical Engineering & Technology* (2024): 1-17.
10. Nawaz, Syed Ali, et al. "ECG Based Heart Disease Diagnosis Using Machine Learning Approaches." *Journal of Computing & Biomedical Informatics* (2024).
11. Zhou, Feiyan, and Duanshu Fang. "Multimodal ECG heartbeat classification method based on a convolutional neural network embedded with FCA." *Scientific Reports* 14.1 (2024): 8804.
12. Banjarnahor, Jepri, et al. "Application of Decision Tree Method in ECG Signal Classification For Heart Disorder Detection." *Sinkron: jurnal dan penelitian teknik informatika* 8.2 (2024): 1064-1072.
13. Begum, Ghousia S., and Vipula Singh. "Automated Detection of Arrhythmia in ECG Signals using CNN." *International Journal of Intelligent Systems and Applications in Engineering* 12.16s (2024): 491-502.
14. Chen, Jintai, et al. "Congenital heart disease detection by pediatric electrocardiogram based deep learning integrated with human concepts." *Nature Communications* 15.1 (2024): 976.
15. Moses, Jeban Chandir, et al. "Time-domain heart rate variability features for automatic congestive heart failure prediction." *ESC Heart Failure* 11.1 (2024): 378-389.
16. Peiman Shahbeigi-Roodposhti and Sina Shahbazmohamadi, Acquisition and Analysis of an ECG (electrocardiography) Signal. <https://www.jove.com/v/10473/acquisition-and-analysis-of-an-ecg-electrocardiography-signal>

17. Eleyan, Alaa, Ebrahim AlBoghbaish, Abdulwahab AlShatti, Ahmad AlSultan, and Darbi AlDarbi. 2024. "RHYTHMI: A Deep Learning-Based Mobile ECG Device for Heart Disease Prediction" *Applied System Innovation* 7, no. 5: 77. <https://doi.org/10.3390/asi7050077>
18. Nguyen, Thanh-Hai, Thanh-Nghia Nguyen, and Ba-Viet Ngo. 2022. "A VGG-19 Model with Transfer Learning and Image Segmentation for Classification of Tomato Leaf Disease" *AgriEngineering* 4, no. 4: 871-887. <https://doi.org/10.3390/agriengineering4040056>
19. Navaneeth Malingan(2024), Understanding Bidirectional RNN, <https://www.scaler.com/topics/deep-learning/bidirectional-rnn/>
20. Shukla, Prashant Kumar, Shalini Stalin, Shubham Joshi, Piyush Kumar Shukla, and Piyush Kumar Pareek. "Optimization assisted bidirectional gated recurrent unit for healthcare monitoring system in big-data." *Applied Soft Computing* 138 (2023): 110178.
21. Rahman, Atta Ur, Yousef Alsenani, Adeel Zafar, Kalim Ullah, Khaled Rabie, and Thokozani Shongwe. "Enhancing heart disease prediction using a self-attention-based transformer model." *Scientific Reports* 14, no. 1 (2024): 514.
22. Mahmoud, Shaimaa, Mohamed Gaber, Gamal Farouk, and Arabi Keshk. "Prediction of Heart Disease Using New Proposed CNN Model Architecture." In 2023 3rd International Conference on Electronic Engineering (ICEEM), pp. 1-8. IEEE, 2023.
23. Dileep, P., Kunjam Nageswara Rao, Prajna Bodapati, Sitaratnam Gokuruboyina, Revathy Peddi, Amit Grover, and Anu Sheetal. "An automatic heart disease prediction using cluster-based bi-directional LSTM (C-BiLSTM) algorithm." *Neural Computing and Applications* 35, no. 10 (2023): 7253-7266.
24. Bhavekar, Girish Shrikrushnarao, Agam Das Goswami, Chafle Pratiksha Vasantrao, Amit K. Gaikwad, Amol V. Zade, and Harsha Vyawahare. "Heart disease prediction using machine learning, deep Learning and optimization techniques-A semantic review." *Multimedia Tools and Applications* (2024): 1-28.
25. Raja E, Soni B, Lalrempui C, Borgohain SK. An adaptive cyclical learning rate based hybrid model for Dravidian fake news detection. *Expert Systems with Applications*. 2024 May 1;241:122768.
26. Xu, HW., Qin, W., Sun, YN. et al. Attention mechanism-based deep learning for heat load prediction in blast furnace ironmaking process. *J Intell Manuf* 35, 1207–1220 (2024). <https://doi.org/10.1007/s10845-023-02106-3>
27. B. Kannan, M. Sakthivanitha, S. Jayashree and R. Maruthi, "Prediction of Cyber Attacks Utilizing Deep Learning Model using Network/Web Traffic Data," *2024 3rd International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, Salem, India, 2024, pp. 363-367, doi: 10.1109/ICAAIC60222.2024.10575032.
28. Kamath MV, Prashanth S, Kumar M, Tantri A. Machine-Learning-Algorithm to predict the High-Performance concrete compressive strength using multiple data. *Journal of Engineering, Design and Technology*. 2024 Mar 4;22(2):532-60.
29. Ainiwaer, Aikeliyaer, Wen Qing Hou, Quan Qi, Kaisaierjiang Kadier, Lian Qin, Rena Rehemuding, Ming Mei et al. "Deep learning of heart-sound signals for efficient prediction of obstructive coronary artery disease." *Heliyon* 10, no. 1 (2024).
30. Rimal Y, Sharma N. Hyperparameter optimization: a comparative machine learning model analysis for enhanced heart disease prediction accuracy. *Multimedia Tools and Applications*. 2024 May;83(18):55091-107.
31. Rao, G.M., Ramesh, D., Sharma, V. et al. AttGRU-HMSI: enhancing heart disease diagnosis using hybrid deep learning approach. *Sci Rep* 14, 7833 (2024). <https://doi.org/10.1038/s41598-024-56931-4>