CRICKET SHOT CLASSIFICATION

Deep Learning Project

Name : Anant Sabharwal

Reg No : 200968138

Defining the Problem Statement

Classification of different batting cricket shots based on video input.

Classification of 10 different cricket batting shots using novel dataset CricShot10.

Meta Data of the Dataset

CricShot10 is a novel dataset comprises of uneven lengths offline batting shot videos and unpredictable illumination conditions.

Novel dataset CricShot10 consists of 10 different cricket shots

The dataset contains videos from 10 different cricket batting shots.

The shots belong to the following classes:

- Cover
- Defense
- Flick
- Hook
- Late cut
- Lofted
- Pull
- Square cut
- Straight
- Sweep

Dataset Generation

The videos are from different publicly accessible sources.

The dataset consists of video clips taken from a variety of ICC events (ODI, Test, and T20 matches) as well as the Indian Premier League (IPL), Bangladesh Premier League (BPL) , and Big Bash League (BBL).

The videos consist of sports played around the world and different types of video illumination.

Batting shots from both left- and right-handed batsmen were taken as a part of the dataset.

The dataset consists of variable length videos.

Mean duration time of the videos 2.56 sec

Standard deviation of time = 0.97 sec

Minimal duration = 1.0 sec

Max duration = 7.72 sec

Video resolution = 1280x720px

The distribution of the dataset is as follows

Name	Training Set	Test Set
Cover Drive	153	35
Defensive Shot	157	35
Flick	146	35
Hook	146	35
Late Cut	147	35
Lofted Pull	151	35
Pull	144	35
Square Cut	160	35
Straight Drive	154	35
Sweep	159	35

PROJECT OBJECTIVES

Finding the most effective architecture to properly classify ten cricket shots.

Experiment with various different hyperparameters to find the most effective architecture to perform the classification.

LITERATURE REVIEW

Some of the significant papers reviewed are as follows:

CricShotClassify: An Approach to Classifying Batting Shots from Cricket Videos Using a Convolutional Neural Network and Gated Recurrent Unit

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8072636/

- Need
  - Significant part of context based advertising to users watching cricket.
  - Generate sensory based commendatory systems and coaching assistants for shot form improvement.
  - Commercial aspect
  - Develop an unbiased, equitable and sensory based commentary system
- Proposes a hybrid deep neural network for classification of 10 different shots from offline videos
- Dataset used – CricShot10
  - Uneven length batting shots
  - Unpredictable illumination conditions
- Uses a CNN for feature extraction and GRU for long temporal dependency.
- All models tried in the project
  - CNN
  - Dilated CNN

    - VGG16 + GRU = 86% - freezing all but 4 layers and 8 layers – 93% accuracy.
    - InceptionV3
    - Xception
    - DenseNet169
- Description of the various architectures
    - Custom CNN-GRU
        - 180x180 15 sequences per video
        - Some convolution layer 3x3 filter size
        - ReLU in CNN layer
        - Max pooling 2x2 down sampling
        - Padding done to prevent model from losing spatial information
        - Batch normalization after each CNN
        - CNN outputs fed to a time distributed layer
        - Time distributed layers input shape is a 5D tensor
        - RNN proposed for maintaining temporal dependency, input is a 3D vector
        - LSTM used as it deals with vanishing and exploding gradients
    - Custom DCNN-GRU
        - Useful as sports videos contain objects other than the batsman's action.
        - Dilation rate of 2
        - Valid padding used to incorporate dilation and reduce the spatial dimension
        - Max pooling 2x2
        - ReLu used to introduce non linearity
    - Transfer learning based proposed architectures
        - Imagenet dataset weights considered for all pretrained models and fully connected layers were discarded
        - See architecture link
    - Proposed vgg net architecture
        - Architecture in link
- Hardware config used
    - AMD Ryzen 7 2700X eight-core 3.7 GHz processor.
    - operating system - Ubuntu version 20.04
    - 32 GB RAM
    - Nvidia GeForce RTX 2060 Super 8 GB GPU memory.
    - Keras with a TensorFlow back-end was used.
- Limitations
    - Dataset contained videos that contained several other frames other besides the main shot. Hence due to the high similarity of different batting shots and high similarity of camera view after shot, misclassifications occurred.

Deep CNN based Data-driven Recognition of Cricket Batting Shots

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8536277

- Used worked on using pretrained networks trained on kinetic video datasets instead of using image based pre-training networks
- About the dataset
    - Frontal camera view only
    - Eight types of renowned shots

    - 800 video clips duration – 5-6 sec @ 4fps for cleaner solutions and keeping trach of memory demands
- Methodology
    - 2D CNN –
        - Video frame features after passing throught the CNN is passed to the LSTM. We discard all the output from the LSTM except that produced at the last timestep. This output is passed through a full connected layer with a softmax layer that produces the probability of the 8 classes
        - 16 layers – 5 convolution + 3 max pooling + 5 relu layers + 2 batch normalization + 1 fully connected layer with softmax activation function
    - 3D CNN –
        - Same a 2D cnn but takes the depth factor into account and perform the convolution on the contiguous frames in spatial and temporal dimensions of the video to extract representations.
        - Input 16rgb images of size 3x3. Input model is 5D tensor 2x3x16x112x112
        - 29 hidden layers. 9 3D convolution layers+ 8 batch normalization layers + 4 max pooling layers + 8 ReLU layers + fully connected layer + softmax layer

Cricket Shot Classification Using Motion Vector

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7224605

- Motion estimated approach to classifying the shots using 3D MACH for action recognition
- Approach is based on using motion vectors that help to measure the angle of any particular cricket shot.
- Can use google pose api to help trach body movements in this .
- Future Works
    - Reduce the processing time and use effective learning techniques such as SVM.

Shot-Net: A Convolutional Neural Network for Classifying Different Cricket Shots
https://www.researchgate.net/publication/328189966_Shot-Net_A_Convolutional_Neural_Network_for_Classifying_Different_Cricket_Shots#:~:text=%E2%80%9CShot%2DNet%E2%80%9D-,in%20order%20to%20classifying%20six%20categories%20of%20cricket%20shots%2C%20namely,Shot%20and%20Leg%20Glance%20Shot.

- Proposed a 13 layer cnn referred to as shot net inorder to classify six categories of cricket shots (cut shot, core drive, straight drive, pull shot, scoop shot, leg glance shot)
- Methodology
    - In this they used 3600 **images** of cricket shots belonging to 6 different classes. (600 each)
        - Classes = cut shot, cover drive , straight drive, pull shot, leg glance, scoop shot
    - 80 20 split
    - Artificially expanded the dataset to avoid overfitting - data augmentation

A Survey on Video Action Recognition in Sports: Datasets, Methods and Applications
https://arxiv.org/pdf/2206.01038.pdf
- Model generally consist of two modules
    - Video feature extraction
    - Classifier
- Approaches

    - Extracting low/middle level features of each frame using GIST or HOG (Histogram of Oriented gradients) and then averaging the frame features overtime for classification
    - Can apply 2D or 3D HOG's
    - Action Bank employs an template based action detector, which is invariate to appearance changes
    - This is also applied to multiscale and multi views videos . N action detectors and M samples yield NxMx73-D feature space
    - **2D / 3D Convolutions**
    - Skeleton based models – GCN graphical convolution networks

Summary of models proposed in various papers:

- CNN for automatic feature extraction and GRU to deal with long temporal dependencies.
- Deep CNN Based Data-Driven Recognition of Cricket Batting Shots
    - 2D CNN, 3D CNN LSTM and RNN were used
    - Transfer learning from pretrained VGG model was also used.
- Cricket shot classification using motion vector
    - Motion estimated approach
    - Much lower accuracy ~ 60% average
- Activity recognition for quality assessment of batting shots in cricket using a hierarchical representation
    - Five levels of grouping actions were classified using
        - Decision Trees (DT)
        - Support Vector Machines (SVM)
        - K-Nearest Neighbours (KNN)
    - 88.3% class weighted F1 score was attained using the best -performing classifier per level
- Cricket shot detection from videos
    - Pretrained CNN and multiclass SVM
    - 83.098% accuracy for three right handed shots and an accuracy of 65.186% for three left handed shots.
    - Limitation of this experiment was a limited dataset
- Temporal classification of events in cricket videos
    - Initially segmented the cricket video into shots and then used a SVM to classify detected shot into 4 events namely run, four, six, out.
    - Recorded accuracy was around 87.8%
- Structural approach for event resolution in cricket videos
    - Structural approach for identifying various cricket events from full length cricket match.
    - Used techniques to classify the events
        - K-Nearest Neighbors
        - Sequential Minimal Optimization
        - Decision Trees
        - Naïve Bayes Classifiers
    - Author mentioned human interpretation for removing replay frames and highlights between deliveries
- Scene classification for sports video summarization using transfer learning
    - Transfer learning-based method that classified sports video scenes to generate video summarization.
    - AlexNet – CNN based approach attained 99.26% accuracy.

Details for shortlisted Models

- **Custom CNN-GRU Architecture**
    - This was a custom architecture
    - Considered frames of 180 x 180 px and 15 sequence per video
    - Dominant spatial features are extracted using CNN.
    - Four layers namely convolution, pooling , flattening, and fully connected layer are required to perform feature extraction from images
    - 3x3 filter size used for CNN
    - ReLu activation function is used
    - Max Pooling layer of 2x2 is used for down sampling the inputs.
    - Padding was done to they have the same dimensions after each convolution operation to prevent the model from loosing spatial information
    - Batch Normalization was used after each convolution block to reduce the number of training steps.

Layer(input)	Output shape	Parameters
Input layer	(none,180,180,3)	-
Conv2D	(none,180,180,64)	1792
Batch Normalization	(none,180,180,64)	256
Max Pooling 2D	(none,90,90,64)	0
Conv2D	(none,90,90,128)	73856
Conv2D	(none,90,90,128)	147584
Batch Normalization	(none,90,90,128)	512
Max Pooling 2D	(none,45,45,128)	0
Conv2D	(none,45,45,256)	295168
Conv2D	(none,45,45,256)	590080
Batch Normalization	(none,45,45,256)	1024
Max Pooling 2D	(none,22,22,256)	0
Conv2D	(none,22,22,384)	885120
Conv2D	(none,22,22,384)	1327488
Batch Normalization	(none,22,22,384)	1536
Max Pooling 2D	(none,11,11,384)	0
Conv2D	(none,11,11,480)	1659360
Conv2D	(none,11,11,480)	2074080
Batch Normalization	(none,11,11,480)	1920
Max Pooling 2D	(none,5,5,480)	0

    - CNN output then fed through a time-distributed wrapper
    - Weights were updated for all frames if frames from a video were individually passed to their adjacent layers .Therefore difference between video frames was meaningless
    - Time distributed layer was introduced that provided the option to share layers across video frames
    - Time distributed layer's input was in the form of a 5D tensor.
    - Following that the time distributed flatten operation generated an output shape that could be passed to an RNN.
    - GRU as it is a simplified version of the LSTM. It solves problem of time dependency , it demonstrated unrivalled success in long term feature dependency using fewer gates than LSTM.

Layer(Type)	Output Shape	Parameters
Custom CNN model	(none,15,5,5,480)	7059776
Flatten()	(None,15,12000)	0

GRU	(none,180)	6577740
Dense	(none,512)	92672
Dense	(none,128)	65664
Dense	(none,64)	8256
Dense	(none,10)	650

  - SGD with lr = 0.0001 and 0.9 momentum was used
- **Custom DCNN-GRU Architecture**
  - For the convolution operation to be termed dilated, the gap factor must be greater than 2.
  - Why Dialated – sports videos of different batting shots contain objects other than only a batsman's action. Thus it is used to have a large overview of the scene content.
  - Dilation rate of 2 was used
  - Valid Padding was added to incorporate dilation and reduce the spatial dimensions
  - Max pooling operation of 2x2 was added after every layer
  - Batch normalization was utilized after each convolution block
  - ReLU activation function needed to introduce non linearity.

Layer(type)	Output Shape	Parameters
Input layer	(none,180,180,3)	-
Conv2D	(none,176,176,96)	2688
Batch Normalization	(none,176,176,96)	384
MaxPooling2D	(none,88,88,96)	0
Conv2D	(none,84,84,128)	110720
Conv2D	(none,80,80,128)	147584
Batch Normalization	(none,80,80,128)	512
MaxPooling2D	(none,40,40,128)	0
Conv2D	(none,36,36,480)	553440
Conv2D	(none,32,32,480)	2074080
Batch Normalization	(none,32,32,480)	1920
MaxPooling2D	(none,16,16,480)	0
Conv2D	(none,12,12,768)	3318528
Conv2D	(none,8,8,768)	5309184
Batch Normalization	(none,8,8,768)	3072
MaxPooling2D	(none,4,4,768)	0

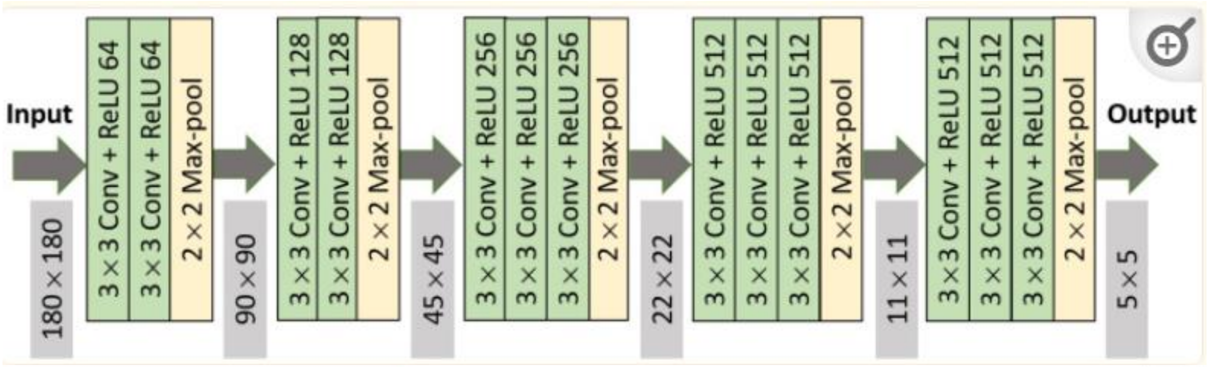
Output then transferred to a time distributed wrapper

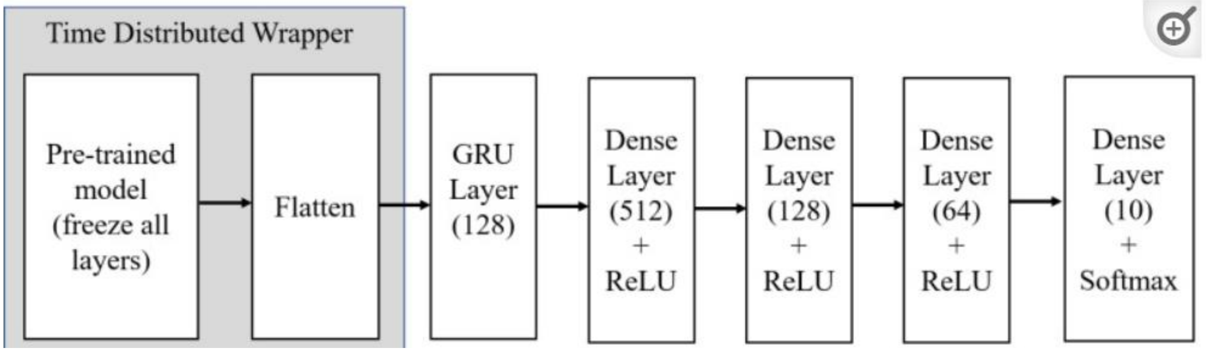
Layer(Type)	Output Shape	Parameters
Custom DCNN model	(none,15,4,4,768)	11522112
Flatten()	(None,15,12288)	0
GRU	(none,180)	6733260
Dense	(none,512)	92672
Dense	(none,128)	65664
Dense	(none,64)	8256
Dense	(none,10)	650

- **Transfer Learning Based Proposed Architectures**
  - most pretrained models use very large datasets that form the basis for further analysis
  - layer freezing is applied to disable the weight updating of the layers of the models.

- ImageNet dataset weights were considered for all pretrained models, and all fully connected layers were discarded.
- Flattening layer was introduce after loading a pretrained model, which converted the input shape into a 1D array.
- Time distributed wrapper was then initialized to wrap the pretrained model layers and the flattening layer , which allowed for sharing the layer weights for all sampled video frames
- GRU layer was appended to trace temporal changes across video frames
- 3 dense layers with 512,128 and 64 hidden units were also used.
- ReLU used as the activation function

- **Proposed VGG-Based Architecture**
  - In this we take a 180x180 px input and the fully connected layer was discarded.
  - Thus the output from the final layer became (5,5,512)
  - CNN's initial features extract low-level features such as colors and lines.
  - In this wro approaches were tried
    - Final 4 layer of the VGG16 were made trainable
    - Final 8 layers of the VGG16 were made trainable





- **VGG16Net followed by LSTM**
  - We use LSTM as it is a type of RNN and capable of capturing long-term dependencies in sequential data
  - Model contains 16 hidden layers for convolution network while the LSTM contains 256 hidden layers.
  - Gradients are first backpropagated through LSTM and then through the fully connected layers of the CNN.
- **3D CNN**
  - Similar to 2D CNN but it also takes depth factor into account and performs the convolution on the contiguous frames in spatial and temporal dimensions of video to extract representations
  - Input contained 16rgb images of size 112x112.
  - The kernel size is 3x3.

- Input model is a 5D tensor 2x3x16x112x112
- 29 hidden layers
    - 9 3D Convolution layers
    - 8 batch normalization’
    - 4 max pooling layers
    - 8 relu layers
    - At the end we have 4096 features which are passed through a fully connected layers
- Implementation
    - Time taken
        - 3D – 9hrs
        - 2D – 13hrs
        - Learning rate = 0.001
        - Dropout 0.6
        - Stochastic gradient descent optimizer
        - Frames resized to dimensions 224x224x3

PROJECT PIPELINE

The project follows the following pipeline.

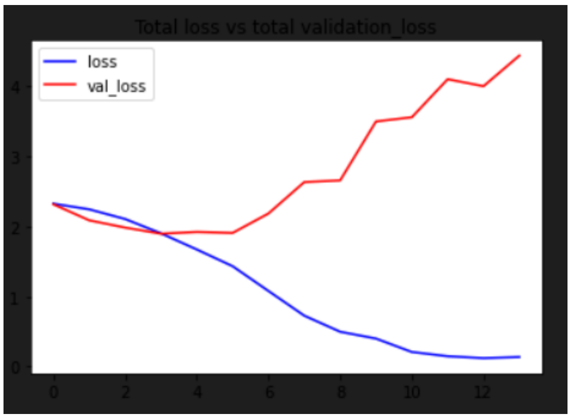
- Functions defined are as follows :
    - The first function **frame_extraction()** is defined that is used to extract certain number of frames from the videos. In this case we extract 15 frames from the entire video each extracted at a fixed interval. This is to give an idea of the entire videos
    - The second function **create_dataset()** is used to create a numpy array of the labels and the extracted frame features of the dataset.
        - While the labels is a 2D matrix after onehot encoding the features extracted form a 5D tensor.
    - Another helper function **plot_metric()** is created that is used to create plots for the loss and accuracy.
    - Further functions were defined that were used to create all the different models. There was one function for each model:
        - **create_cnn_gru()** for the cnn gru model
        - **create_dcnn_gru()** for creating the dialated cnn gru model
        - further models for vgg16 with 4 and 8 frozen hidden layers were also created.
        - A new ConvLSTM model was created using the **create_convLSTM_model()** function.
            - In this we use the ConvLSTM2D recurrent layer.
            - This takes the number of filters and kernel size required for applying the convolution operation
            - Further description is in the following section.
        - A new LRCN model is also used. LRCN stands for Long Term Recurrent Convolutional Network. This combines the LSTM and CNN layers in a single model.
- After this an analysis is done of all the models and the model with the highest accuracy is use to evaluate test results and classify youtube videos with the help of a function

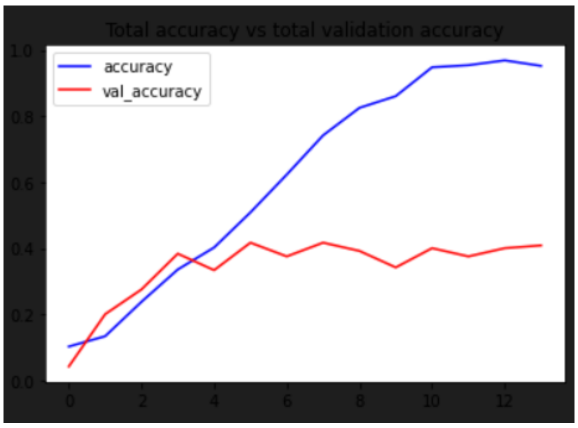
predict_on_video() which is take the video file path, sequence length and model name as inputs and produces and output.

PERFORMANCE ANALYSIS OF THE MODELS

1) **convLSTM** model had the following metrics:
    a. training - loss = 0.138 accuracy = 0.9521
    b. validation – loss = 4.436 accuracy = 0.4083
    c. test – loss 1.8538 accuracy = 0.3333

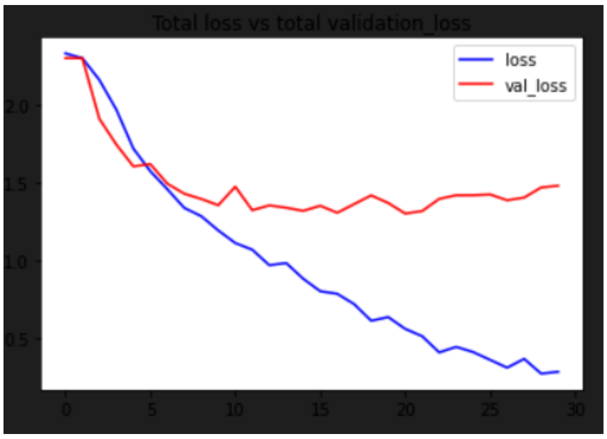
   We also conclude from the graph that the models encounter large amount of overfitting on the entire dataset

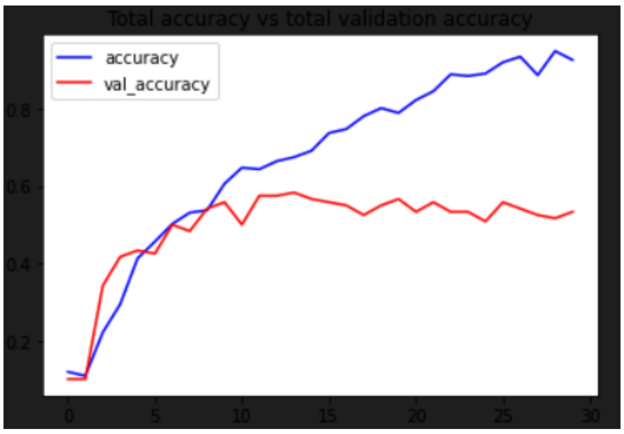




2) **LRCN** has the following metrics
    a. training - loss = 0.2828 accuracy = 0.9271
    b. validation – loss = 1.4818 accuracy = 0.5333
    c. test – loss 1.5014 accuracy = 0.5467

   from the plotted graphs we can see that even this model even though it performs better encounters a large amount of overfitting after training in the dataset





MODEL ARCHITECTURE SUITABLE FOR THE PROBLEM STATEMENT

From research and analysis of the problem statement as well as the lecture slides it can be proposed that some sort of encoder decoder architecture is best suited for the problem. The proposed encoder must be some sort of RNN (can be an LSTM or GRU) of CNN's and the decoder has to be a fully connected layer or a group of fully connected layers that converge to classification.

TABULATION OF RESULTS

Model Name	Train loss	Validation Loss	Test Loss
convLSTM	= 0.138	4.436	1.8538
LRCN	0.2828	1.4818	1.5014

Model Name	Train Acc.	Validation Acc.	Test Acc.
convLSTM	0.9521	0.4083	0.3333
LRCN	0.9271	0.5333	0.5467

RESULT ANALYSIS

From the above results we can conclude that both the models that we built encounter substantial overfitting. While the convLSTM model had a test accuracy of 33.33% the LRCN model showed an accuracy of 54.67%. One of the many reason for the models showing overfitting can be attributed to the fact that the size of the dataset that was used for training the models was substantially reduced from 1888 videos down to 750 videos. This was done purely in-order to be able to work with the existing computational resources. The respected analysis curve showing overfitting have been attached above.

REASON FOR SELECTING HYPERPARAMETERS

In the **LRCN** model the hyperparameters were selected for the following reasons:

- Here we will implement a Long Term Recurrent Convolutional Network. This combines the LSTM and CNN layers in a single model. CNN used to extract features from the frames while the LSTM is used to pass the features in a sequence
- Each layer was wrapped in a **TimeDistributedLayer** as this **wrapper** allows to apply a layer to every temporal slice of input.

In the **convLSTM** model the hyperparameters were selected for the following reason:

- We started with 4 filters being used and the number of filters being used increased to 8, 14, and 16 in the successive convolution layers.

In both the models the output layer has a **softmax** activation function as our problem is a case of multiclass classification.

An instance of **early stopping** is also used which is a form of regularization used **to avoid overfitting**. **A patience of 10** is also used meaning that the model will stop training if it doesn't show improvement over 10 instances.

Both the models are then compiled using a **loss function of categorical cross entropy** while using an **Adam optimizer**.

The **convLSTM model** was trained over **20 epochs**. This was reduced from an original 50 epochs as the model did not show any significant improvement beyond this point.

The **LRCN model** was trained over **30 epochs**. This too was reduced from the original 70 epochs as the model did not show any significant improvement beyond this point.

Both the models regardless of the hyperparameters being changed over the course of the experiments **showed overfitting**. This can be attributed to reducing the size of the dataset that we were dealing with, reducing it from 1888 videos extracting 20 frames per video to 750 videos, extracting 15 frames per video. (The reduction of the dataset was done to deal with constrained computational resources)

CONCLUSION

The project 'CRICKET SHOT CLASSIFICATION' dealt with the classification of 10 different cricket shots. The entire project was built / worked upon using a novel 'CRICSHOT10' dataset (compiled by Anik Sen, Kaushik Deb, Pranab Kumar Dhar from the Department of Computer Science & Engineering, Chittagong University of Engineering & Technology (CUET), Chattogram 4349, Bangladesh).

the models that were used were some sort of a combination of a CNN for feature extraction and an RNN, LSTM or a GRU for treating the frames as a series input .The LRCN model worked better than convLSTM model showing a better test accuracy of ~ 54 %. The LRCN model trained faster than the convLSTM model as well.

FUTURE WORKS

- Creating an application for real time cricket shot classification using a larger dataset to classify an even larger number of cricket shots.
- Creating a larger more detailed dataset of cricket shots to enable more research in the field of sports activity recognition (specially cricket as not a lot of video data was available on the problem statement)