# Tennis Shots Classification With Deep Learning

MSCA 37011 Spring 2022

Vritti Gandhi, Shikhar Madrecha, Haider Waseem, Nihaal Zaveri

# Agenda

- Abstract
- Exploratory Data Analysis
- Modeling
- Results and Future Work

# Abstract


- Goal: identify the type of tennis shots from a video
- Data taken from Thetis dataset
- Implemented three different models
  - Base CNN model
  - Binary classification CNN model
  - LRCNN

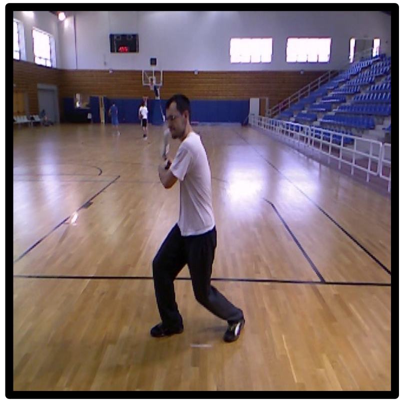
# EDA

# Data Dictionary

The data has been taken from ***THree DimEnsional TennIs Shots (THETIS)*** dataset.

THETIS set is comprised of a set of 12 basic Tennis shots performed by 31 amateurs and 24 experienced players. Each shot has been performed several times resulting in 1980 (single period cropped) videos, converted to AVI format. Videos were captured using an Xbox Kinect.

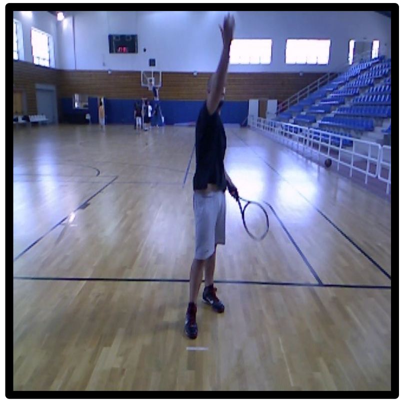
The total duration of the videos is 2 hours and 15 minutes.



Two handed backhand

Forehand Flat

One handed backhand

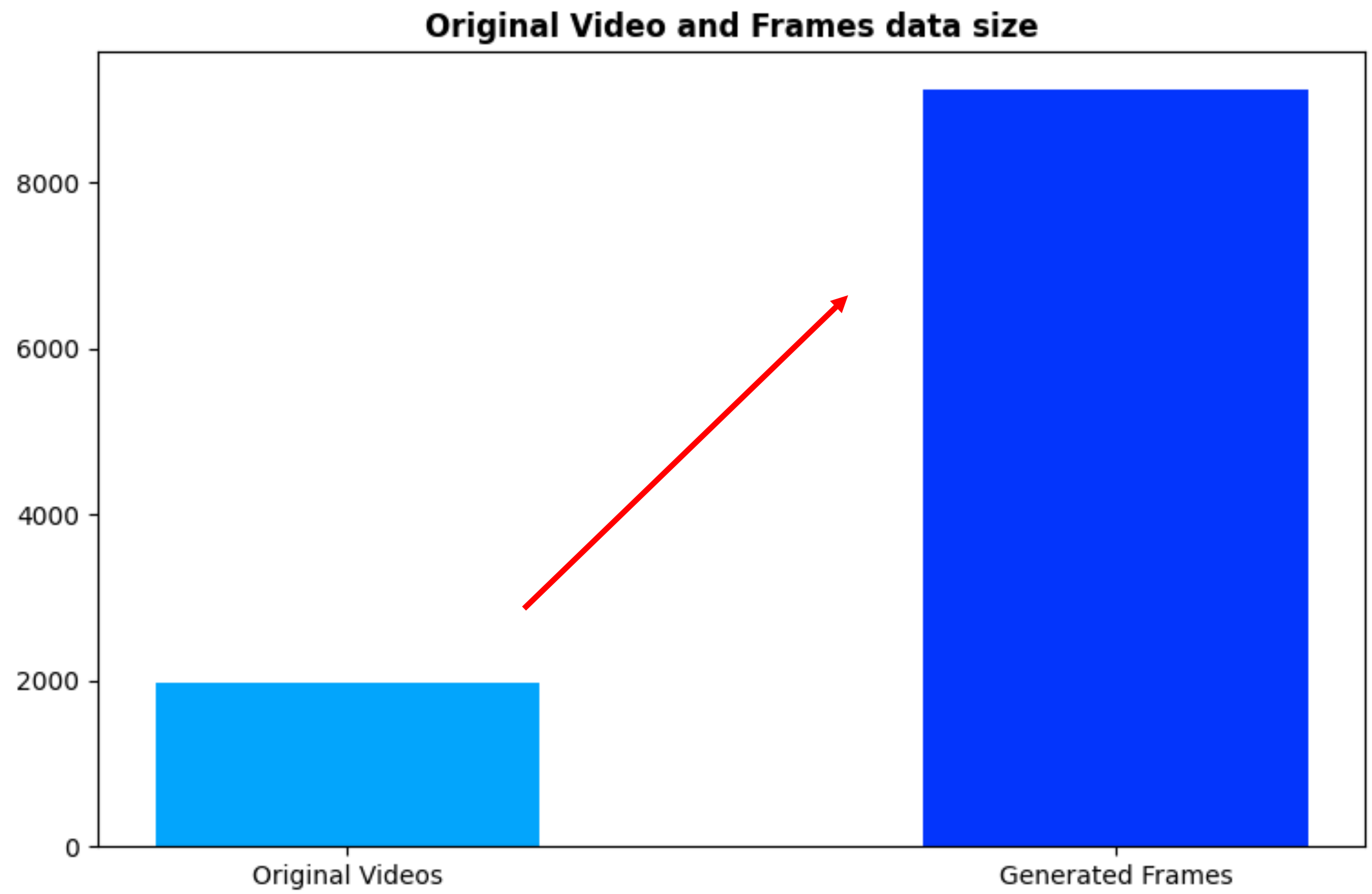


Smash

The complete set of classes which are included in the dataset are:

1. *backhand,*
2. *backhand 2 hands,*
3. *backhand slice,*
4. *backhand volley,*
5. *flat service,*
6. *forehand flat,*
7. *forehand open stands,*
8. *forehand slice,*
9. *forehand volley,*
10. *kick service,*
11. *slice service,*
12. *smash*

# Distribution of Classes and Data Size



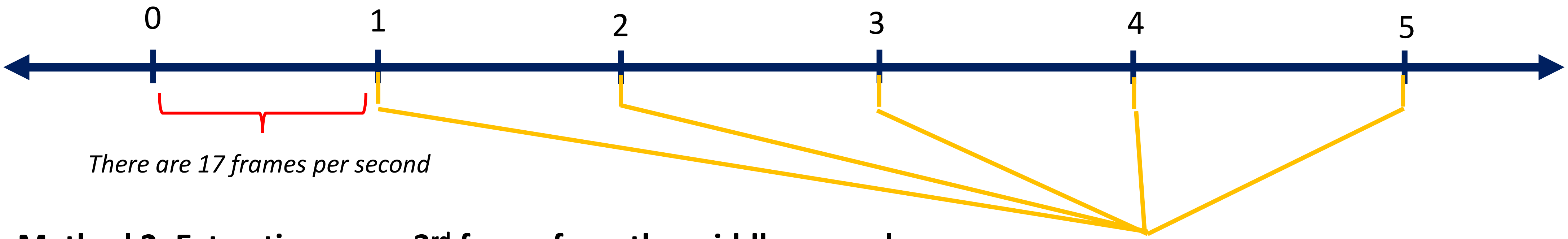

The THETIS dataset contains 5 second videos for each shot. To be able to input these videos in the model, we split every video in different frames and use these frames as an input into the model.

We have total **1980** videos split equally between 12 classes, i.e., **165 videos for each class.**

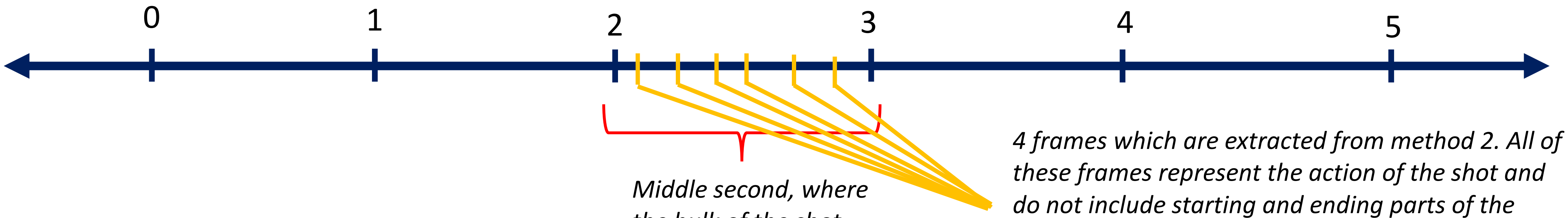
From these videos, we extract the frame at the end of every second which leads to generation of c. **9k frames** in total and around **700-800 frames for each class.**

# Extracting Frames

**Method 1: Extracting the last frame of every second**




**Method 2: Extracting every 3rd frame from the middle second**




**Method 3: Extracting 16 frames from the entire video**


*For the LRCN model, we extract 16 frames evenly spread out throughout the video.*

# Resizing Frames

The original size of the frames extracted from videos was 480*640 pixels. This was too large to be used as an input on our local machines and was making the model run into OOM errors.

We decided to resize the images to 120*160 pixels to improve model speed and performance.

We also normalize all images by dividing the pixels by 255.


# Infrastructure Used

As running CNNs can be computationally very expensive, we have tried variety of platforms / infrastructures to boost training and prediction speed.

THE UNIVERSITY OF CHICAGO | Office of Research and National Laboratories
Research Computing Center

RCC - Skyway

Midway

M1 PRO

NVIDIA

# Modeling

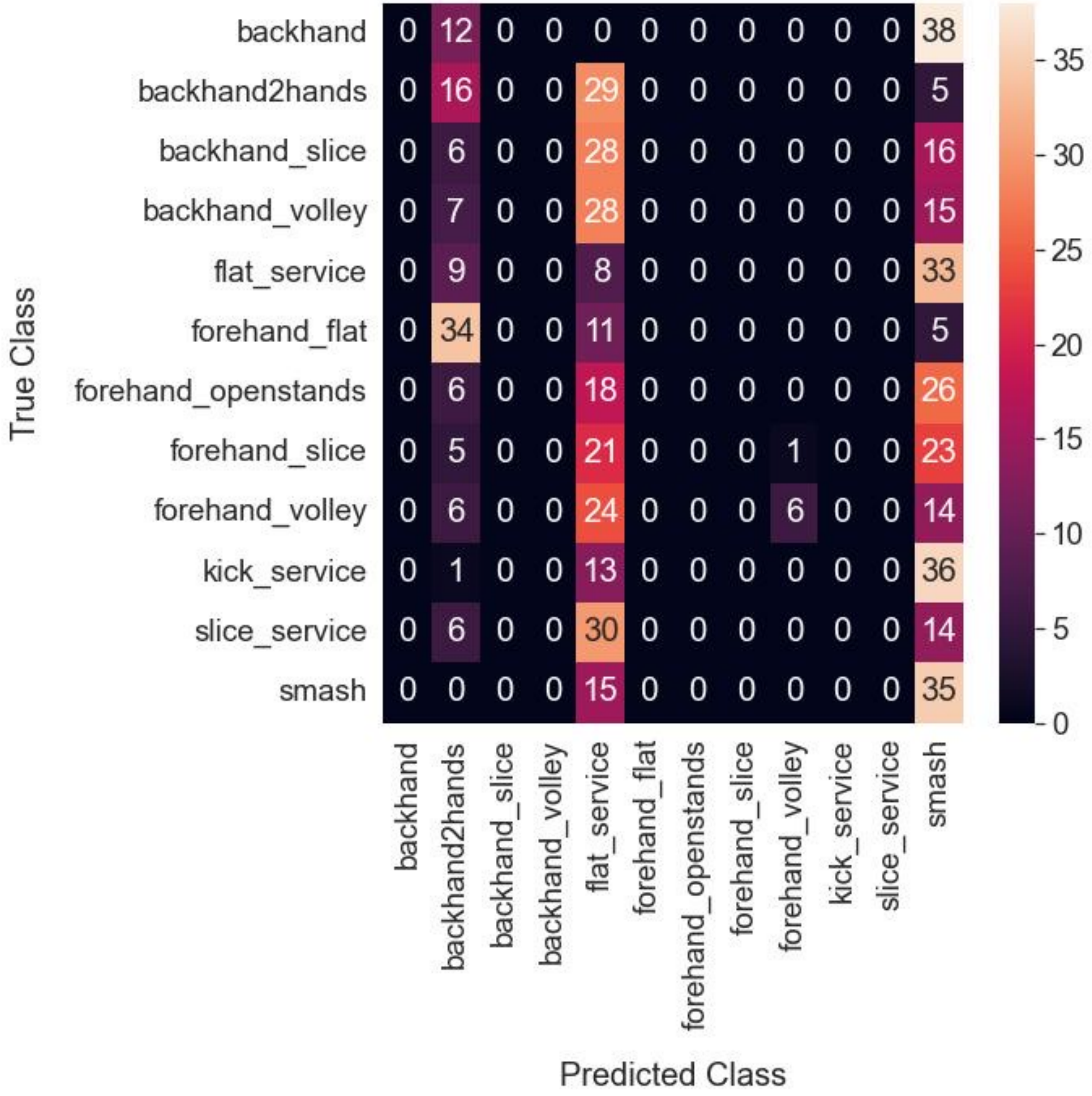
# Base Model

- For our baseline model, we use all 12 shot classes
- We selected 200 frames of each shot type for training the model
- 50 frames of each shot were used for validation

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 118, 158, 32)      896

 conv2d_1 (Conv2D)           (None, 114, 154, 64)      51264

 batch_normalization (BatchN  (None, 114, 154, 64)     256
 ormalization)

 conv2d_2 (Conv2D)           (None, 114, 154, 64)      102464

 max_pooling2d (MaxPooling2D  (None, 57, 77, 64)       0
 )

 batch_normalization_1 (Batc  (None, 57, 77, 64)       256
 hNormalization)

 flatten (Flatten)           (None, 280896)            0

 dense (Dense)               (None, 12)                3370764

=================================================================
Total params: 3,525,900
Trainable params: 3,525,644
Non-trainable params: 256
_________________________________________________________________
```

# Base Model


```
Test loss: [36.97636795043945, 0.10833333432674408]
Train loss: [16.63172721862793, 0.2212500125169754]
```

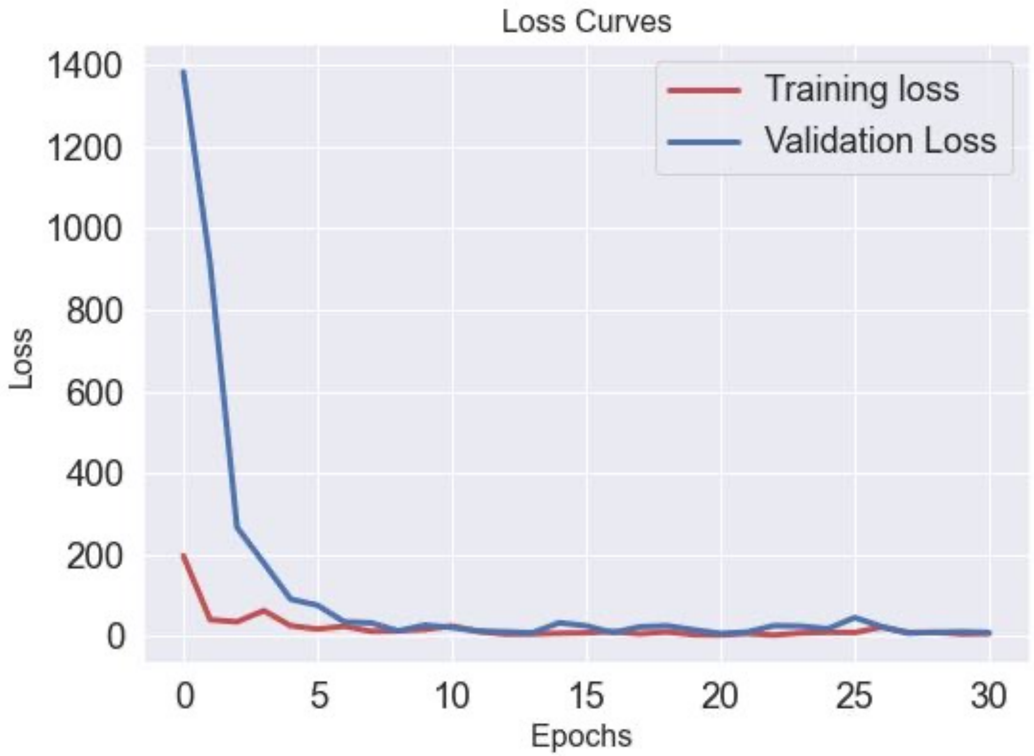


# Binary Classification

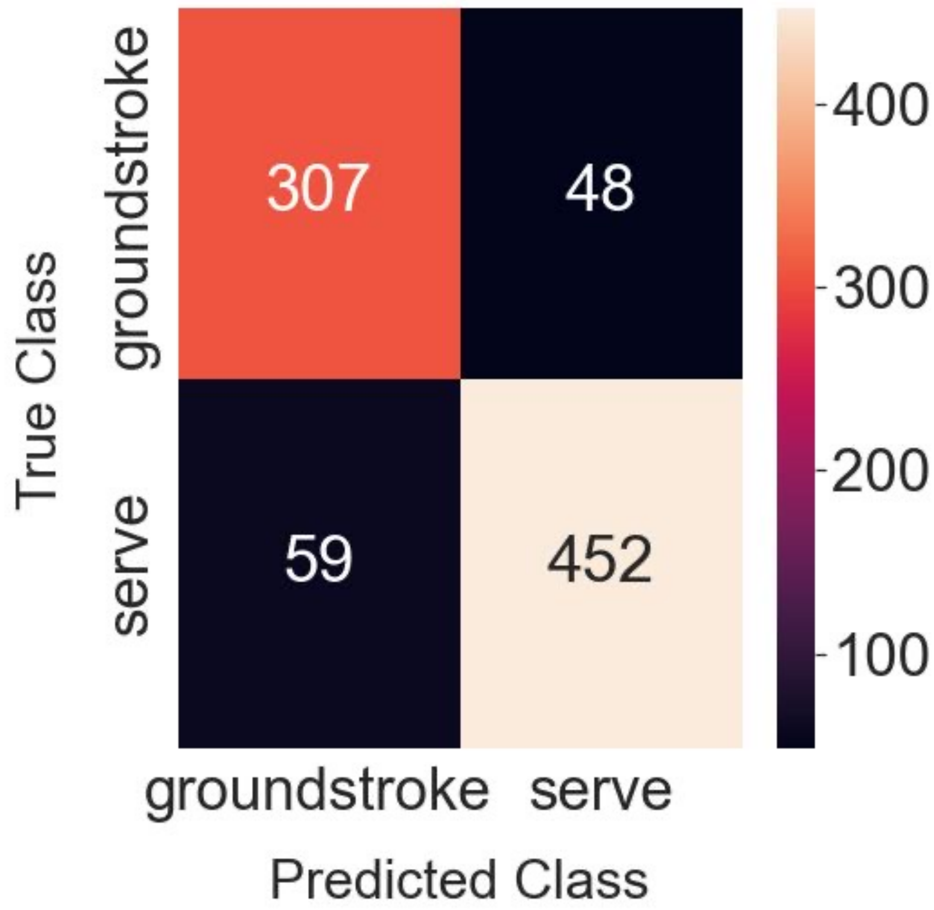
- Instead of using all 12 shot classes, we simplified our approach to a binary classification
  - Groundstrokes: 2 handed backhand and flat forehand
  - Service: combine three types of serve – slice, flat, kick
- 1776 groundstroke frames
- 2553 service frames

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 118, 158, 32)      896

 conv2d_1 (Conv2D)           (None, 116, 156, 64)      18496

 batch_normalization (BatchN  (None, 116, 156, 64)     256
 ormalization)

 conv2d_2 (Conv2D)           (None, 116, 156, 64)      65600

 max_pooling2d (MaxPooling2D  (None, 58, 78, 64)       0
 )

 batch_normalization_1 (Batc  (None, 58, 78, 64)       256
 hNormalization)

 flatten (Flatten)           (None, 289536)            0

 dense (Dense)               (None, 2)                 579074

=================================================================
Total params: 664,578
Trainable params: 664,322
Non-trainable params: 256
_________________________________________________________________
```

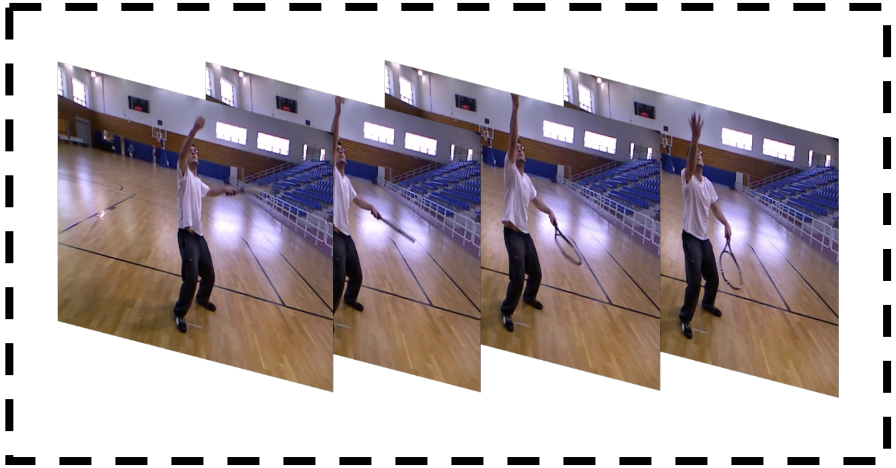
# Binary Classification




```
Test loss: [6.450009822845459, 0.8764433860778809]
Train loss: [3.3866405487060547, 0.9347386956214905]
```



# Binary Classification – 4D Input



One input

Instead of using each frame as a single data point, we also decided to use 4 frames from each video as a single data point and created a 4-dimensional input array instead of a 3-dimension input array.

```
[ ]  x_train_resized.shape

     (2304, 120, 160, 3)

[ ]  x_train_vids.shape

     (576, 4, 120, 160, 3)
```

Old input array (3d)

New input array (4d)

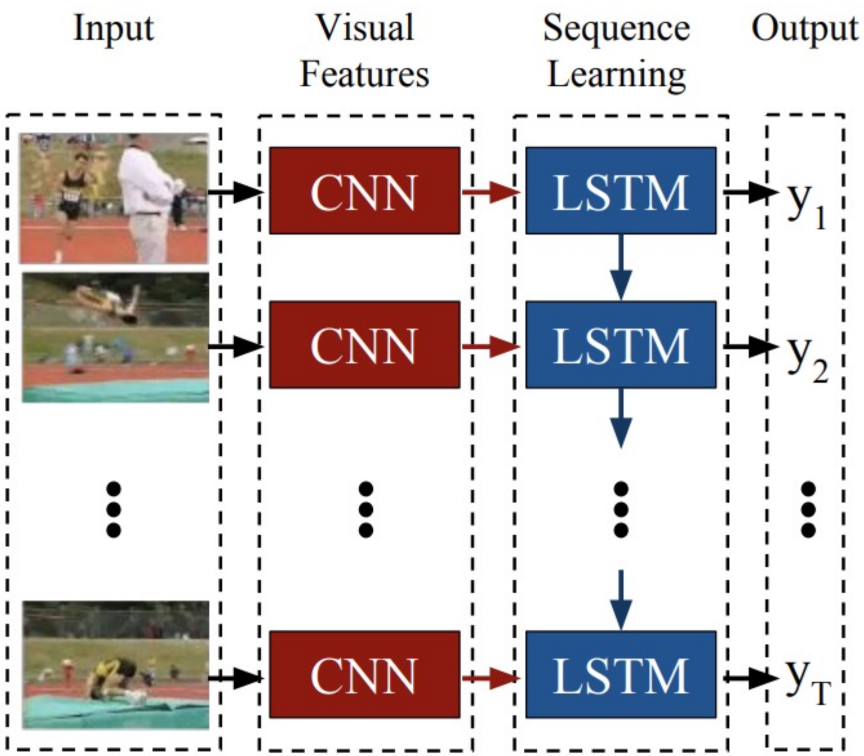
The idea behind this approach was to somehow provide a sequence of inputs in the traditional CNN architecture.

The **accuracy** from the binary classification problem is **around 60%** from this model.

Model Architecture

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_15 (Conv2D)          (None, 4, 118, 158, 32)   896

 conv2d_16 (Conv2D)          (None, 4, 114, 154, 32)   25632

 max_pooling3d_10 (MaxPoolin  (None, 1, 57, 77, 32)    0
 g3D)

 batch_normalization_5 (Batc  (None, 1, 57, 77, 32)    128
 hNormalization)

 dropout_10 (Dropout)        (None, 1, 57, 77, 32)     0

 conv2d_17 (Conv2D)          (None, 1, 57, 77, 64)     51264

 max_pooling3d_11 (MaxPoolin  (None, 1, 29, 39, 64)    0
 g3D)

 dropout_11 (Dropout)        (None, 1, 29, 39, 64)     0

 flatten_5 (Flatten)         (None, 72384)             0

 dense_5 (Dense)             (None, 2)                 144770

=================================================================
Total params: 222,690
Trainable params: 222,626
Non-trainable params: 64
_________________________________________________________________
```

# Long Term Recurrent CNN




LRCN processes the (possibly) variable-length visual input with a CNN, whose outputs are fed into a stack of recurrent sequence models. A variable-length prediction is then produced.

Both the CNN and LSTM weights are shared across time, resulting in a representation that scales to arbitrarily long sequences.

# Long Term Recurrent CNN

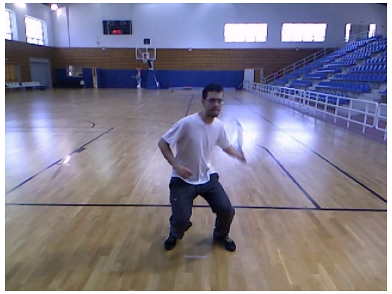
Backhand

Backhand Slice

Backhand2hands

→ Backhand

Forehand Flat



Forehand Open Stance

Forehand Slice

→ Forehand

All serves were combined and labeled as serves. Backhand volley, forehand volley, and smash are kept separate, resulting in 6 classes.

# Long Term Recurrent CNN

Train, validation and test split (80-10-10) → Extraction of Frames (16 per video) → Pretrained Inception V3 as feature extractor → LSTM Model

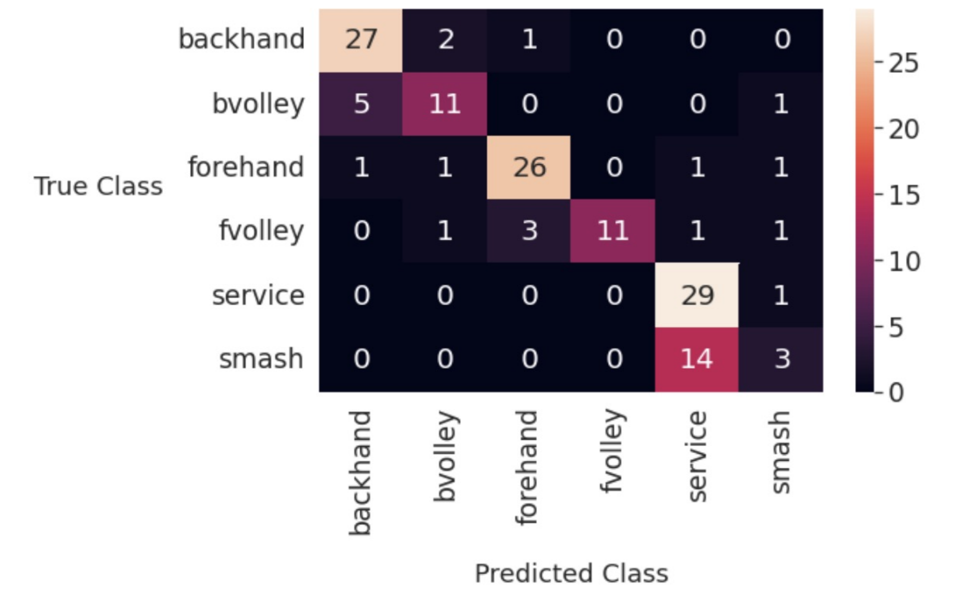
LSTM Architecture

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dropout_20 (Dropout)        (None, 16, 2048)          0

 lstm_10 (LSTM)              (None, 16, 128)           1114624

 time_distributed_20 (TimeDi  (None, 16, 128)          0
 stributed)

 time_distributed_21 (TimeDi  (None, 16, 6)            774
 stributed)

 lambda_10 (Lambda)          (None, 6)                 0

=================================================================
Total params: 1,115,398
Trainable params: 1,115,398
Non-trainable params: 0
```

# Long Term Recurrent CNN

```
Epoch 300/300
7/8 [=========================>....] - ETA: 0s - loss: nan - categorical_accuracy: 0.9592
Epoch 300: saving model to drive/MyDrive/DeepLearning_Test/experiments/base_model/checkpoints/lstm_weights.0300-0.743.hdf5
8/8 [==============================] - 0s 52ms/step - loss: nan - categorical_accuracy: 0.9615 - val_loss: 0.7426 - val_categorical_accuracy: 0.7899
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| backhand | 0.82 | 0.90 | 0.86 | 30 |
| bvolley | 0.73 | 0.65 | 0.69 | 17 |
| forehand | 0.87 | 0.87 | 0.87 | 30 |
| fvolley | 1.00 | 0.65 | 0.79 | 17 |
| service | 0.64 | 0.97 | 0.77 | 30 |
| smash | 0.43 | 0.18 | 0.25 | 17 |
| accuracy | | | 0.76 | 141 |
| macro avg | 0.75 | 0.70 | 0.70 | 141 |
| weighted avg | 0.76 | 0.76 | 0.74 | 141 |



# Conclusion & Future Work

| Model | Number of Output Classes Used | Test Accuracy |
|---|---|---|
| Base CNN | 12 | 11% |
| Binary Classification CNN | 2 | 87% |
| Binary Classification CNN – 4D input | 2 | 60% |
| LRCNN | 6 | 76% |

- Deploy model using Streamlit
- Build more complex model in the future and identify all 12 classes
- Solve overfitting problem
- Solve problem of Nans in training loss of LRCNN
- Have model run in real-time over a tennis video for live shot prediction

# References


- Donahue, Jeff, et al. *Long-term Recurrent Convolutional Networks for Visual Recognition and Description*, 2014
- https://github.com/ganasank/CS230
- https://github.com/chow-vincent/tennis_action_recognition
- http://thetis.image.ece.ntua.gr/textfiles/thetis.pdf
- http://thetis.image.ece.ntua.gr/