UNIVERSITY OF CALIFORNIA SAN DIEGO

Tennis Serve Classification using Machine Learning

A Thesis submitted in partial satisfaction of the
requirements for the degree of Master of Science

in

Electrical Engineering (Signal and Image Processing)

by

Aiman Jabaren

Committee in charge:

Professor Truong Nguyen, Chair
Professor Cheolhong An
Professor Pamela Cosman

2020

The Thesis of Aiman Jabaren is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California San Diego

2020

DEDICATION

To my mother, my father, and my family.

EPIGRAPH

"Don't limit your challenges; challenge your limits."

Jerry Dunn

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank and acknowledge my advisor Professor Truong Nguyen for his support and motivation. I would like to thank David for helping me with the data collection, and my friends, Jeremy and Aman, for their feedback. I would also like to express my deepest gratitude to the thesis committee members Professor Pamela Cosman and Professor Cheolhong An, my professors, fellow students, the UCSD tennis coach, my family, and everyone who has contributed to the project through discussion and volunteering for data collection.

VITA

2015	Bachelor of Science in Biomedical Engineering, Technion Institute of Technology, Haifa
2016	Bachelor of Science in Electrical Engineering, Technion Institute of Technology, Haifa
2020	Master of Science in Electrical and Computer Engineering, University of California San Diego

FIELDS OF STUDY

Major Field: Electrical and Computer Engineering

Studies in Machine Learning, Signal and Image Processing and Hardware

Previous Fields: Biomedical Engineering

Studies in Signal processing and Biomechanics

ABSTRACT OF THE THESIS

Tennis Serve Classification using Machine Learning

by

Aiman Jabaren

Master of Science in Electrical Engineering (Signal and Image Processing)

University of California San Diego, 2020

Professor Truong Nguyen, Chair

In this thesis, we propose a new approach to measure the quality of a serve in tennis. We formulate this as classifying tennis serves videos into amateur and professional category and introduce a novel two-stage architecture based on LSTM to address this task. In the first stage, we use a state-of-the-art CNN model to extract 3D human pose estimates in a temporal context. An LSTM is then used in the second phase to classify the human pose. We also investigate various 3D human pose estimation algorithms, LSTM architectures, and data collection methods and evaluate their performances for classifying tennis serves.

Chapter 1

Introduction

Measuring the quality of tennis serves in videos is a complex task that offers many challenges. It may be considered as a sports video classification task. In other words, in order to solve this, we need to look into video classification problems and harness current research in that field for our purpose. We also argue that solving the problem at hand will contribute to other sports and video classification problems.

1.1 Video Classification

Video classification is a recent problem that is being solved as a stepping stone for solving more complex computer vision tasks. Over the last decades, the number of videos and recordings has increased exponentially. Video data collection has become a very easy task that most people do on a daily basis. More than 300 hours of Youtube videos are uploaded to the internet every minute[1]. The gathered data may be processed and used to solve video related computer vision tasks. These videos could be used to infer more patterns and develop algorithms that would solve video classification tasks. More conventional video classification methods include SVM and PCA analysis. Most of the early video classifiers are based on video-audio features. Many of the early studies and methods are based on Hidden Markov Models[2][3] which allow visual and temporal feature extraction. More modern approaches harness neural networks for this task, making it a much easier problem and introducing a boost in performance. In this work, we take

inspiration from the state-of-the-art existing video classification algorithms.

Video classification models harness CNN based architectures for image feature extraction, multiple approaches have been investigated to tackle and extract the temporal context of the video. In this work, we ignore audio features and focus on visual representation. A popular approach is stacking convolutional networks and RNN based blocks. The convolutional block could be a pre-trained VGG16. The RNN block would extract the temporal features. The whole model is trained simultaneously. Another popular approach is based on 3D ConvNets developed by Paluri et al. [4]. In their approach, 3D Convolutional networks (3D ConvNets) are harnessed to extract spatiotemporal features. It outperformed 2D convolutional networks and other state-of-the-art convolutional classifiers.

The advancement in video classification has brought attention to an interesting and harder task which is the sports classification task. This is a subtask of video classification which has the same challenges and properties, in addition to introducing more complex settings and problems such as the human pose estimation.

1.2 Sports Classification

Many sports analysis tasks involve video content. To name a few: performance analysis, sports classification, action recognition, semantic analysis, event detection, and results prediction[5]. Traditional classifiers have been used in recent decades for sports and sports genre classification tasks, such as: HMMs[6], naïve Bayesian classifiers (NBCs)[7], decision trees[8], and support vector machines (SVMs)[9][10]. Many books and papers have contributed to the content analysis of statistics for baseball[11], basketball[12][13][14] and other sports. All of these algorithms contribute to sports computer vision tasks, such as performance analysis, recognition, classification, and prediction. We were inspired by their findings and motivation to measure sports performance through a classification task.

1.3 Human Pose Estimation

Human pose estimation problem has been of great interest in recent research. Many approaches have been developed to solve this problem. The recent advances of neural network architectures and computation capabilities, combined with the availability of data, greatly increase the possible obtainable accuracy. Some of the first problems to be solved were estimating 2D joints from 2D images, where multiple neural networks and convolutional network-based algorithms were developed for this end. Datasets were collected and labeled for training and evaluating these models. In the last couple of years, more recent models are being developed with the capability of estimating 3D coordinates and human mesh parameters from single 2D images[15].

We have been inspired by the different and successful approaches that divide the architecture into a pre-trained feature extraction CNN and then pass the spatial features into a temporal features extraction block which is based on RNN and LSTM. This approach is very popular in image tasks and image classification in particular. LSTM blocks have shown exceptional results in video and temporal tasks. A different approach that has been attempted is based on the pre-trained CNN image feature extraction block and replaces the LSTM block with a multi-layer perceptron. In other words, the extracted spatial features are flattened and inserted into a multilayer perceptron. Regardless of size and memory restrictions, this approach generates very promising results. In our case, this approach introduces some computational restrictions. Although we might reconsider it in the future, the currently developed model is not based on this approach. We are, however, eager to compare this approach with our current model in the future.

Chapter 2 discusses the methods and experiments implemented. It also describes dataset acquisition and processing. We go over different algorithms and approaches we have implemented in our pipeline for the human pose estimation problem. We discuss 2D joints estimation vs 3D joints estimation and the triangulation experiment applied on 2D joints estimation. In the 3D joints estimation subsection, we argue that 3D human pose estimation in a video is a more

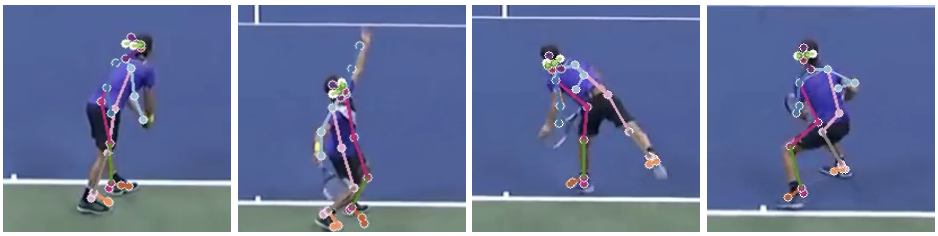

Figure 1.1. Tennis serve frames

complex problem thus a corresponding dedicated architecture is required. We also discuss the different LSTM architectures we have developed for the second phase. At the end of the chapter, we describe in detail the pipeline and the training process.

Chapter 3 displays the results of the different models. It summarizes the architectures' performance in accuracy and loss tables and training plots. In addition to the conventional training plots and test results, we display the models' performance on unseen subjects in order to fully verify the model's performance and compliance with our goal.

Chapter 4 summarizes and discusses the results of the different experiments while analyzing their meaning and significance. It analyzes the dataset acquisition and comments on its process. We also compare the different human pose estimation algorithms and the different LSTM architectures.

Chapter 2

Methods

2.1 Dataset

2.1.1 Data acquisition

The problem at hand is challenging on many levels. One of the major challenges in sports and video classification problems is the scarcity of available labeled data. Thus, we have developed a state-of-the-art two-stage architecture that harvests pre-trained feature encoders for more efficient training with limited datasets. Considering the proposed architecture and pipeline, we looked for a dataset that complies with our main assumption that sport video tasks rely mainly on players' pose features. With this in mind, and considering the scarcity of available tennis serve video datasets, we have collected labeled data from multiple sources and focused on the visible human pose. We recorded professional and amateur players on the tennis court and collected tennis serves from tennis games from the Internet. We now discuss the two main sources that we used for data collection.

Field Data

A GoPro Hero 7 was used for collecting 70-110 frame videos where each video is a serve of 4-6 seconds and the frame rate is 30 fps. The recording angles were constrained to the front and back angles of the tennis player. We set our camera on a tripod while recording to mimic real-life tennis game recording view and collect different types of serves such as flat, slice,

and kick. Our recordings assume that the player's body pose is more important than the serve type. Our subject pool consists of males between 19-25 years old with either no experience or professional players from the UCSD tennis team. All these participants volunteered for collecting this dataset which was recorded on UCSD campus courts under the supervision of a tennis coach.

Internet Data

Our field data lacks video samples from professional players which are harder to find than amateur players. Therefore, in order to increase samples for this class, we collected tennis videos and game recordings from tennis matches and used Adobe Premiere Pro to crop and trim tennis serve samples. Each sample is a 30 fps video sequence of the same size having a single professional player in the scene who is performing a serve.

2.1.2 Data processing

As mentioned before, data collection and dataset handling are challenging, therefore, multiple methods have been adopted for more efficient feature extraction under the given limitations. The first step in preprocessing the videos is cropping and trimming the player while serving. There is a single human in every clip. A full tennis serve starts with the player holding the ball in one hand and the racket in the other hand and ends with both feet on the ground. This consistency in the samples would facilitate the classification problem and prevent unnecessary feature learning. As elaborated in the model, the first phase, which is robust, will extract human pose features that are later converted into a human skeleton. Having variance in the dataset parameters may be perceived as noise that the neural network would learn to ignore. Such parameters might be the height of the players or the angle of the serve. In order to tackle this issue and reduce the required dataset size, we have developed and added a data processing unit in order to minimize the variance of less important features. This unit includes the following procedures:

1. Normalization: we normalize the size of the skeleton, so all players have the same height

before classification. This would prevent the second phase of the model from learning the player's size and height as parameters.

2. Rotation: Skeletons have been randomly rotated around the normal axis. Although we are careful in the data collection and the recording, we wanted to make sure that the angle of the test samples or any future samples won't play part in giving inaccurate or biased results.

3. Shift: Due to the installment of the camera and the different environments of the recordings, the players' skeletons were shifted to the axes of origin. This is done in order to minimize the number of features the network needs to learn.

4. Interpolation: All samples used in the experiments are recorded at a constant rate of 30 fps. Nonetheless, for the purpose of keeping the model generic and robust in the future, an interpolation unit has been added in order to interpolate future samples that may be collected at a different rate, to the current 30 fps rate.

2.1.3 Data structure

The data structure is part of the architecture and was manipulated and processed in order to maximize efficiency and remain in compliance with the model's requirements. The input of the architecture is 2D RGB videos. Each video sample is about 80 frames, sampled at 30 frames per second. This input is fed to the pose estimation algorithm which gives a joints estimation human pose output. This output is converted into a skeleton and to a three-dimensional tensor as displayed in Figure 2.1. The size of the tensor is (number of Cartesian dimensions)x(number of joints)x(number of frames) = 3x25xframes. Due to a slight variation in the number of frames per sample, we pad and mask the dimension of the frames to achieve a constant number of frames. The tensor is padded to 110 frames, with a value of -1000. This padding value is of a different order than the real data order which allows the LSTM to learn to ignore the padding and extract

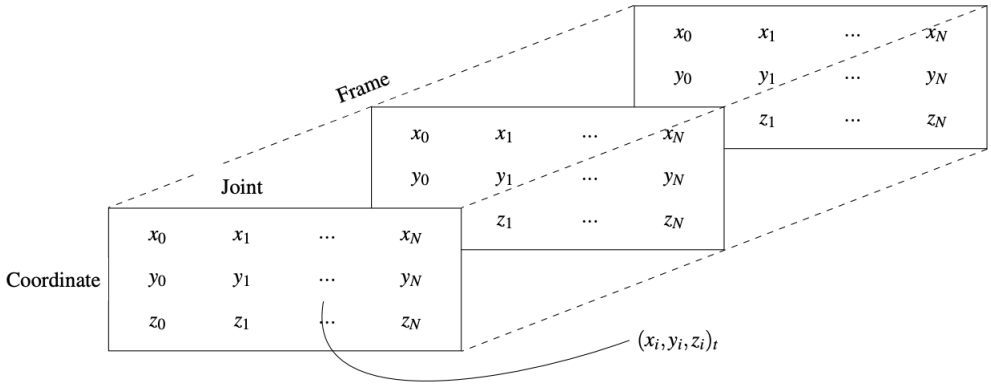

Figure 2.1. Tensor data structure in between both sub-models

the features out of the relevant frames only. This data structure is then fed into the second phase which is based on LSTM.

2.2 Human Pose Estimation

As discussed earlier, we implement computer vision algorithms in order to solve the human pose estimation problem. Through this project, we have developed and implemented multiple experiments in order to choose the best approach.

2.2.1 2D joints inference using 2 images

The first approach we attempted for joint inference was 2D joint inference using 2D images. It was important to explore this approach since it is more understood and studied. This approach offers lower errors and is very easy to apply. However, its output is limited to 2D which is open to many interpretations and misclassification when different poses are projected into a 2D plane. These limitations become significant when the size of the available dataset is small. Implementing this approach has performed well and given a good pose estimation. However, it is clear that it is not sufficient when considering sports tasks due to the complexity of the task.

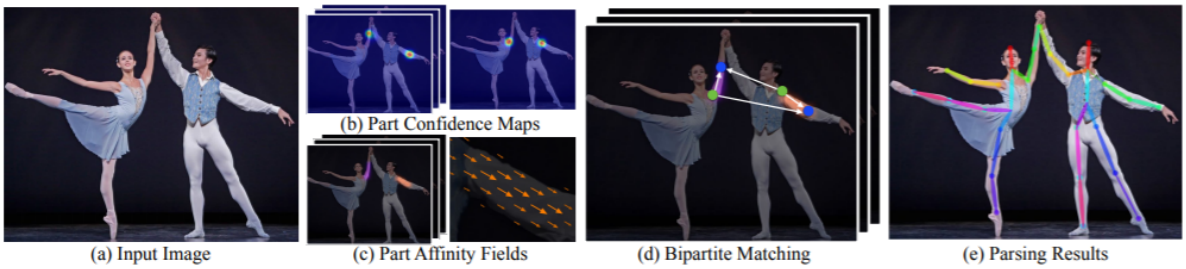

Figure 2.2. OpenPose Architecture. Image taken from [16]

OpenPose is a state-of-the-art 2D human pose estimation algorithm that utilizes part affinity fields[16]. This is a bottom-up approach rather than the conventional human detection approaches which detect the human first and then the body parts. In this architecture, Sheikh et al. utilize a bottom-up presentation of association scores via Part Affinity Fields (PAFs). Their work has significantly contributed to the computation speed, by reducing it, 200%, while increasing accuracy by 7%. The body parts of the humans are detected first and a final parsing is used to extract the human pose.

Architecture

Single images are passed through a baseline network, such as VGG-19, creating a feature map. Then multi-stage Convolutional layers are used in order to process the feature maps, thus generating a set of Part Confidence Maps and a set of Part Affinity Fields (PAFs):

1. Part Confidence Maps are a set of 2D confidence maps, S, for body part locations. Each joint location has a map.

2. Part Affinity Fields (PAFs) are a set of 2D vector fields, L, which encode the degree of association between parts. Each pair of predefined body parts has a PAF. Assuming C is the number of joints and c*c is the total number of joints pairs (body parts), then: $L = (L_1, L_2, ..., L_c)$ is the part affinity tensor.

In the end, a greedy algorithm is used in order to process the two sets and generate the pose estimate for each person in the image.

2.2.2 Triangulation: 2 Cameras

Triangulation refers to the process of determining a point in 3D space given its projection onto 2D images. This is a reconstruction task that requires camera parameters. It's based on the fact that a point on a 2D plane corresponds to a line of points in 3D space. In other words, all points on that line in 3D space have the same projection onto 2D. Therefore, one 2D image is not enough in order to reconstruct the 3D space. The 3D reconstructed structure is the intersections of all of the lines in 3D space[17]. This requires a thorough and multi-view camera reconstruction. Although multi-camera approaches have been developed before and have achieved a good reconstruction performance, the lack of available data and the complexity of multi-view camera recordings constrain us and make this an unfavorable approach. Nonetheless, we have examined this approach carefully. We have developed experiment settings using 2 GoPro Hero cameras for 3D reconstruction.

In theory, this is a simple and easy task. However, the nature of the problem makes it more complicated and an ill-posed problem. The problem we are trying to solve doesn't have accurate 2D projections. The projection images are RGB images and in order to reconstruct the human skeleton in 3D space, we apply a human pose estimation algorithm on each of the projections and extract tensors of 2D human joints. Then we reconstruct the 3D human pose through triangulating matching human joints from the two processed images. This nature of multi and sequential processing leads to error propagation. In other words, the error is compounded and amplified in each phase of the pipeline. Camera's estimated parameters have given errors due to the cameras' parameter variance and imperfections. Most of the problem parameters are extracted through calibration.

When implementing triangulation on video frames, a major issue is triangulating the matching frames from the same time steps. To do so, we have calibrated the recorded video

clips using the flashlight method. In this method, a flashlight is turned on at the beginning of the recording so we can match the first matching frames from both clips (cameras). We have implemented OpenPose and extracted human pose from matching 2D images. Then we implement triangulation on the generated 2D human pose from the matching frames from the 2 cameras. The extracted matching 2D human joints are triangulated and a 3D reconstruction is created for each time step and frame. When the 3D reconstructed frames were plotted as a function of time, huge errors emerged due to multiple factors: Camera parameters misestimation, 2D human pose estimation errors, and the fact that errors were inflated due to triangulation of error-based joints estimation. In other words, rather than applying triangulation on the actual projections of the 3D points, we apply triangulation on shifted and misestimated projections. Therefore, the reconstructed 3D joints are the intersections of the lines that pass through the shifted projections, leading to an accumulated error in 3D reconstruction.

2.2.3 Smoothing

The reconstructed 3D skeleton videos have been displayed and analyzed. Although the skeleton shape was preserved in most reconstructed samples, there was a significant error that is the result of multiple errors propagating through the pipeline. For the purpose of filtering the jitters and smoothing the skeleton movement, we have tried multiple filters such as a Kalman filter[18]. We use a Kalman filter as an iterative process for smoothing the estimated pose and get close to the true positions in space. The errors introduced in the pipeline are unpredicted, uncertain, and have random variations. A Kalman filter reduces the error but takes time to contain the error and the estimated human pose skeleton is not fully smooth and doesn't capture the fine movement in sports activities.

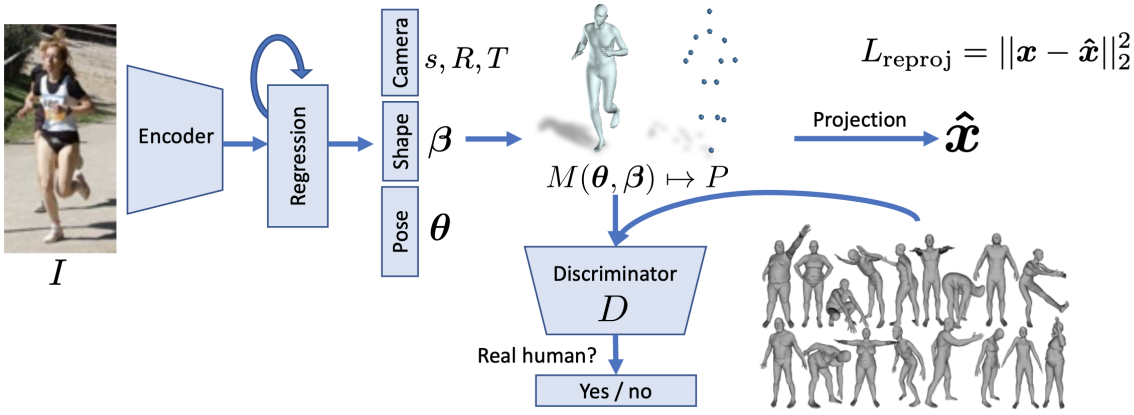

Figure 2.3. Human Pose estimation pipeline. Image taken from [15]

2.2.4 3D Joints Estimation

3D joints from image

Multiple approaches have been aimed at enhancing the 3D joints estimation problem. This is a challenging problem because of the nature and complexity involved. Some of these challenges are the ambiguity of the scene and scarcity of 3D joints ground truth labeling. Labeling 3D joints is a complex task that usually requires either multiple depth cameras or a set of trackers/sensors on the subjects. Some companies have dedicated teams for the collection task and for extracting human 3D pose using state-of-the-art cameras and body tracking sensors. 3D joints data collection is based on annotated data that was collected as RGB and depth images. They harness optical flow and time of flight techniques and cameras. Low-cost devices such as Microsoft Kinetics has made it easier to obtain and complement collected RGB data for better human pose annotation.

Multi-stage models have been developed to solve this problem. However, they are not optimal because they are based on many assumptions, such as the background or the camera angle, or require complex data collection sets. More efficient and simple approaches such as the SMPL algorithm [19] and “End-to-end recovery of human shape and pose” [15] by Kanazawa

et al. allow us to utilize the abundances of labeled 2D joints datasets for training 3D human pose estimation models. Kanazawa et al.'s method is based on the SMPL[19] model which parameterizes the mesh and the 3D joint angles and learns a low dimensional linear shape space. The model utilizes Resnet-50[20] feature extraction encoder as the architecture pre-trained on Imagenet classification task[21]. The Resnet output is average pooled, producing a feature map $\phi \in R^{2048}$ which represents the human pose features and the image encoding. The 3D regression module consists of two fully-connected layers with 1024 neurons, followed by a final layer of 85 neurons. It utilizes existing 2D joints labeled dataset for training and calculates the loss and error of the mesh grid parameters and the 3D joints estimation on the 2D projections which lays in the same plane as the original 2D image. A discriminator consisting of two fully connected layers with 10, 5, and 1 neurons, solves the human shape ambiguity thus inhibiting non-human 3D joints estimations. Hence the network is encouraged to output human parameters while the discriminator acts as weak supervision discouraging unusual body shapes. Model's 3D joints error was evaluated on Human3.6M dataset[22], a standard 3D pose benchmark captured in a lab environment and MPI-INF-3DHP dataset[23]. The reported error metrics in human pose estimation models are mean per joint positions error (MPJPE) and reconstruction error.

The loss function calculated during training is based on two sets: 2D annotated samples and 3D annotated samples.

$$L = \lambda(L_{reproj} + 1L_{3D}) + L_{adv}$$

where λ controls the weight of each objective, 1 is a boolean indicating whether a 3D annotation is available. 3D regression module is to output human shape feature vector, θ, given an image encoding ϕ.

$L_{reproj} = \sum_i ||v_i(X_i - \hat{X}_i)||$ where $X_i \in R^{2xK}$ is the 2D ground truth of the i-th joint, v_i is a boolean indicating whether the joint is visible or not. $\hat{X}$ is the projection of the module's output. Θ is regressed in an iterative error feedback loop[24][25][26], recurrent and progressive changes are made to the current estimate θ_t. In other words, ϕ and θ are concatenated and fed to the regressor

which would give a residual of the pose $\Theta \in R^{3k}$ where k is the number of the estimated joints. $\beta \in R^{10}$ is PCA analysis parameters of the space. As for L_{3D}, it is calculated and considered when 3D annotation is available.

$$L_{3D} = L_{3Djoints} + L_{3Dsmpl}$$
$$L_{joints} = ||(X_i - \hat{X}_i)^2||_2{}^2$$
$$L_{smpl} = |||[\beta_i, \theta_i] - [\hat{\beta}_i, \hat{\theta}_i]||_2{}^2$$

L_{3D} and L_{reproj} are calculated on the final estimate θ_t. However, the adversarial loss is calculated on every iteration of θ_t. As discussed earlier, the reprojection error triggers the network to create a 3D pose that has the same projection on the 2D plane as the ground truth. However, the produced projections may be misleading and a discriminator D is needed to distinguish whether the generated SMPL parameters are for a real human shape or not. This Discriminator has been pre-trained to distinguish real SMPL human parameters. The discriminator learns the parameters of human body shape and the possible angles and rotations between the joints thus preventing unconstrained human shapes. The previous approaches[19][27] have a priori assumptions and predefined degrees of freedom of the human skeleton model. The encoder adversarial is:

$$minL_{adv}(E) = \sum_i E_{\theta \sim pE}[(D_i(E(I)) - 1)^2]$$

The objective for each discriminator is:

$$minL(Di) = E_{\theta \sim p_{data}}[(D_i(\theta) - 1)^2] + E_{\theta \sim pE}[(D_i(E(I)))^2]$$

Dataset

2D annotated in the wild images dataset: LSP, LSP extended[28] MPII[29], and MS COCO[30]. Dataset sizes are 1k, 10k, 20k, and 80k images respectively. For 3D datasets: Human3.6M[22] and MPI-INF-3DHP[23]. Both datasets provide 150k training images with 3D annotations.

The algorithms used for the human pose estimation were trained on MPII[29], COCO[30], HumanEva[31], and Human3.6M[22]. The MPII human pose dataset is a multi-person 2D Pose Estimation dataset composed of different human activities collected mainly from YouTube videos. The COCO Keypoints dataset[30] is a multi-person 2D Pose Estimation dataset with collected images from Flickr. It is the largest 2D Pose Estimation dataset and is used as a benchmark for testing 2D Pose Estimation algorithms. HumanEva[31] is a single-person 3D Pose Estimation dataset made up of video sequences. Ground truth 3D poses were captured using marker-based motion capture (mocap) cameras. HumanEva was the first 3D Pose Estimation dataset of substantial size. Human3.6M[22] is a single person 2D/3D Pose Estimation dataset made up of short video sequences recorded using RGB and time of flight depth cameras. This dataset contains 11 subjects and 15 different activities. It is the largest 3D pose estimation dataset.

3D joints from video

Despite the many similarities, estimating the human pose in a video is more complex than pose estimation in a single image. Kanazawa, et al. have developed a state-of-the-art simple but effective temporal encoding of image features[32]. Adding the temporal features representation smoothes the 3D mesh predictions when analyzed in the temporal dimension. A per frame image human pose estimation without taking into consideration temporal information introduces a different type of error in every frame which causes a higher frequency error and leads to error amplification. However, the developed model recovers a current 3D mesh as well as its past and future motion which is used to predict adjacent frames. Annotated video datasets are limited. Therefore, they have harvested Internet-scale sources of unlabeled videos by running a 2D pose estimator, OpenPose[16], on the unlabeled videos, thus obtaining pseudo-ground truth. Their model outperformed other prediction models.

One frame at a time is passed through an image feature extraction block, Resnet-50[20], generating ϕ_t image features map. Every window of T=13 feature vectors ϕ_t, centered at t, is

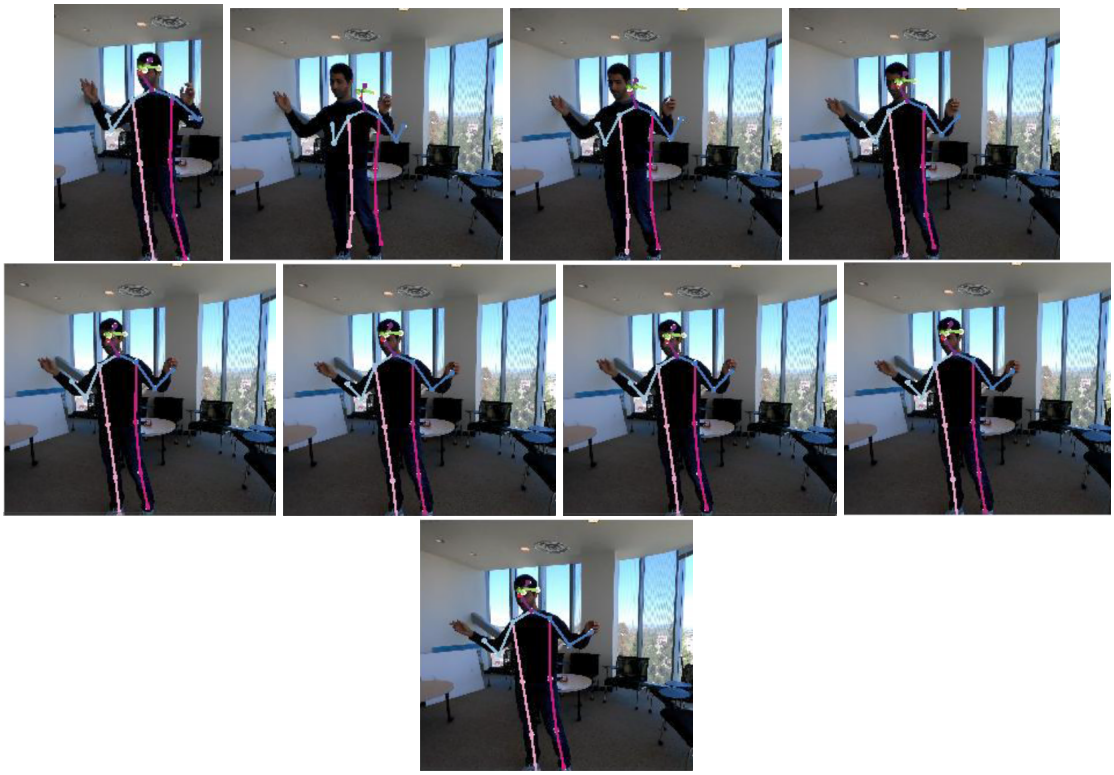

Figure 2.4. 3D human pose estimation applied to separate frames in a video

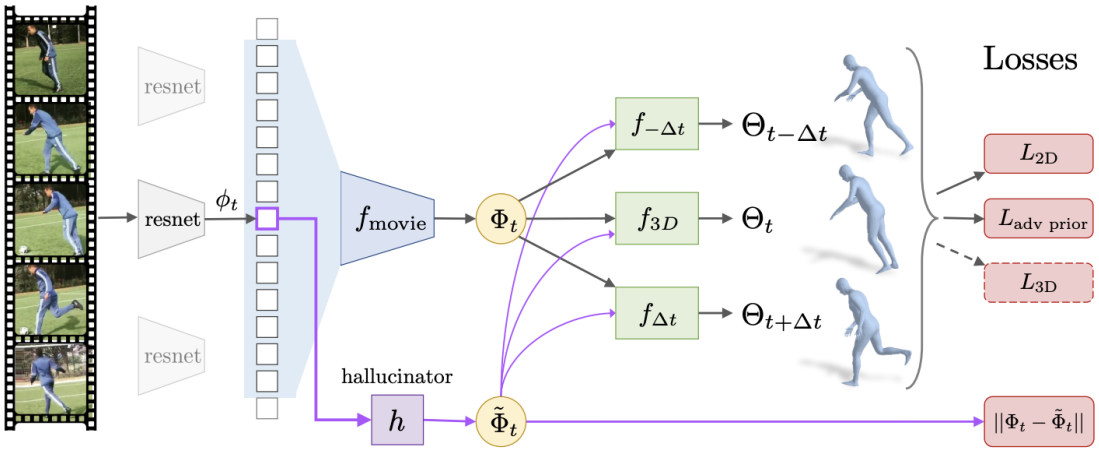

Figure 2.5. Dynamic Human Pose Estimation Pipeline. Image taken from [32]

encoded using f_{movie} generating a representation of the 3D human dynamics Φ_t. Φ_t is used to predict and estimate the human pose at time t: Θ_t and the neighboring Δt frames. The loss function is based on the reprojection loss and adversarial loss which validates the human pose. Moreover, 3D loss annotation is calculated when 3D annotations are provided. The loss function is described in the 3D human pose estimation in images section. The Hallucinator (Hallucination Block) takes a single image features ϕ_t and generates a temporal hallucination that would be used in creating temporal features out of a single image by predicting the changes in the nearby future and past frames. The added temporal content reduces ambiguity and unwanted jitter in the 3D estimation. Loss function at time t is defined as: $L_t = L_{2D} + L_{3D} + L_{adv_{prior}} + L_\beta$. The loss functions are described in the section above.

$$L_{2D} = ||v_t(x_t - \hat{x_t})||_2{}^2, L_{3D} = ||\Theta_t - \hat{\Theta}_t||_2{}^2$$

The adversarial prior, as discussed earlier, trains to discriminate the shapes of the joints.

$$L_{advprior} = \sum_k (D_k(\Theta) - 1)^2$$

Due to the assumption that the person filmed is the same, a constant shape loss is added that tries to minimize the changes in the recorded person shape.

$$L_{constshape} = \sum_{t=0}^{T-2} ||\beta_t - \beta_{t+1}||$$

The main goal of the Hallucination block is to learn a function that maps a human pose features from one image into features representing the human pose in multiple frames. The Hallicinator (Hallucination block) is self-supervised and is trained using the ground truth extracted from the movie strip.

$$I_{hal} = ||\Phi_t - \hat{\Phi}_t||_2$$

To sum up, the temporal encoder is trained with the hallucinator:

$$L = L_{temp} + L_{hal} + L_t(\hat{\Phi}_t) + \Sigma L_{t+\Delta t}(\hat{\Phi}_t)$$

Dataset

This model has also been trained on multiple datasets according to protocol[15]. One of these datasets is Human3.6M[22] which has ground truth 3D annotation. Another used dataset with 2D annotation is the Penn Action dataset[33] and a collected NBA dataset. In addition to the already existing labeled datasets, the model was trained on youtube video collections, VLOG lifestyle dataset[34], and VLOG people and InstaVariaty from Instagram. To generate ground truth, OpenPose[16] algorithm was applied to these videos to produce human skeleton ground truth videos.

2.3 LSTM

2.3.1 LSTM Unit

LSTMs are RNN units that have been developed for sequence prediction and classification. In a sequence, the imposed order of the samples is important, and we need to preserve it in a learning task. In the given problem, the sequence is the coordinates of the human joints as a function of time. Although, as discussed earlier, a more conventional multi-layer perceptron may be applied, they have some limitations and restrictions that make them undesirable for the classification phase of the pipeline. Some of the Multi-layer perceptron limitations are that they learn a fixed-function approximation, unawareness of temporal structure of inputs, and a fixed-sized input. However, the problem we are trying to solve is temporal dependent and has multivariate inputs. Moreover, given the complexity of the problem and the given data sizes, mapping time into space, CNNs or multi-layer perceptron would introduce computational complexities. RNNs[35] are based on the idea of mapping time into state, where the data is processed based upon what has come before. They are very powerful since they have distributed hidden state that allows them to store past information very efficiently in addition to their non-linearity property. RNNs are harder to train due to gradient exploding and vanishing problems especially when trained on more time steps.

Some of the best solutions to mitigate the problems above are LSTMs[36] and GRUs[37] which have given us very similar and enhanced results while possessing memory to overcome the issues of long term temporal dependency. We have used LSTM units in our architecture. LSTM units are based on multiple gates: input gate, forget gate, and an output gate. The input gate indicates whether the cell state is updated and which values from the input to update the memory cell. The forget gate indicates whether the memory is set to zero or how much of the old cell state should be forgotten or remembered. The output gate indicates whether the current info is visible and what to output. These gates are normalized using a sigmoid function. Each gate is fed the hidden states and the current inputs. The cell state is updated while using a tanh(x) function to prevent vanishing or exploding gradients.

LSTM Equations:

$$
\begin{aligned}
i_t &= \sigma(x_t U^i + h_{t-1} W^i) \\
f_t &= \sigma(x_t U^f + h_{t-1} W^f) \\
o_t &= \sigma(x_t U^o + h_{t-1} W^o) \\
C_t &= \sigma(f_t * C_{t-1} + i_t * \hat{C}_t) \\
\hat{C}_t &= tanh(x_t U^g + h_{t-1} W^g) \\
h_t &= tanh(C_t) * o_t
\end{aligned}
$$

W, U: Input and hidden state weight matrices respectively

2.3.2 LSTM Architecture

1. Basic Vanilla LSTM:

   Vanilla LSTM architecture is based on one LSTM unit and was first defined in 1997 LSTM paper[36]. It is composed of an input layer, a fully connected hidden layer, and a fully connected output layer.

2. Stacked LSTMs:

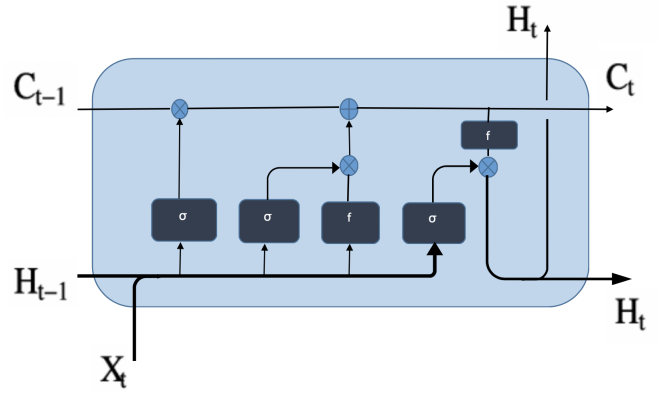

Figure 2.6. LSTM Block

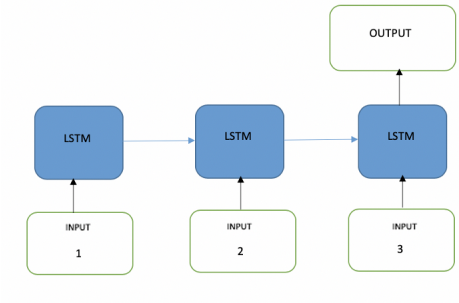

Figure 2.7. LSTM Block used in video tasks

Stacked LSTM is a model consisting of multiple hidden LSTM layers, each containing multiple memory cells. Increasing the number of hidden LSTM layers makes the model deeper and like most deep learning techniques, enhances performance[38].

According to the Universal Approximation Theorem, any function can be approximated by a forward shallow network made of one large single hidden Multilayer Perceptron with a sufficient number of neurons[39][40]. However, a large and shallow single hidden layer is not guaranteed to be trained successfully. Deep and hierarchical models can increase function approximation accuracy performance exponentially when compared to shallow ones[41]. They give similar results to large and shallow single hidden layers while using fewer neurons and training in less time. We use the same concept with LSTMs, adding levels of abstractions over time. Stacking layers may result in extracting more abstract and long-term features[42]. They were introduced first by Graves, et al[43]. According to their findings, the depth of the neural network had contributed to performance more than the number of memory cells in a layer. As shown in Figure 2.8 an LSTM layer passes a sequence output rather than a single value output to the following LSTM layer: One output per input time step.

3. CNN LSTMs:

   Another architecture we have investigated was a CNN LSTM which combines convolutional layers and LSTMs. They are very useful in vision-related tasks since they extract spatial and temporal information. The CNN layers may be pre-trained on relevant tasks, extracting features relevant to the problem being solved. It is very useful in spatial and temporal structures. Our pipeline is similar to CNN LSTM in concept but more complex. The whole architecture can be divided into two main sub-models: A CNN based sub-model for features extraction and an LSTM based model for processing and interpreting the features across the temporal dimension. The CNN is made of multiple 2D convolutional layers and maxpooling layers. The CNN model is applied to the input, images or videos in this case,

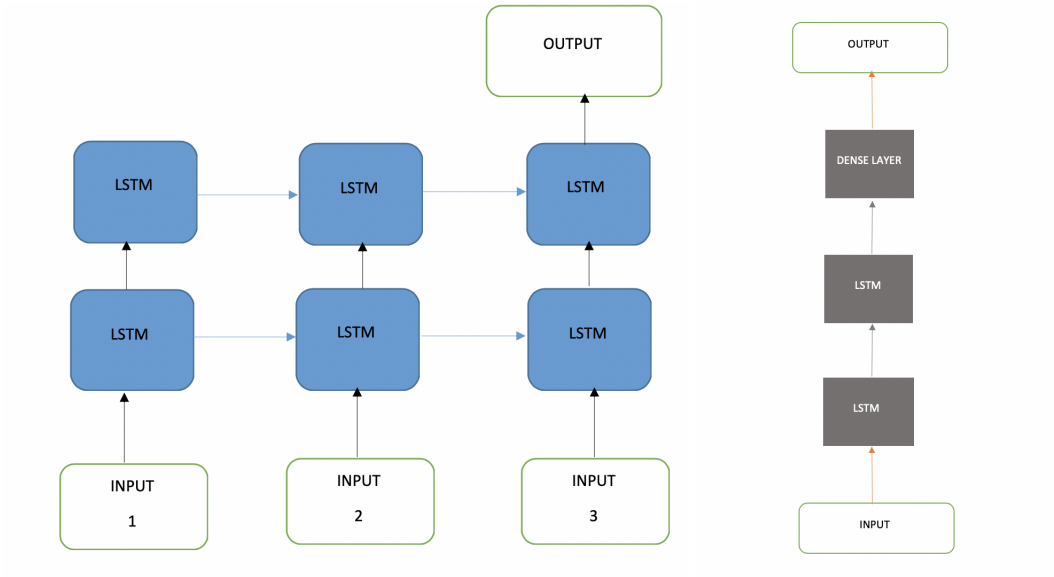

Figure 2.8. Stacked LSTM Block

and the output is passed to the LSTM as single time steps. As expected, the additional CNN layer didn't contribute to classification especially since we have already extracted the features out using a pre-trained Resnet-50[20].

Two types of CNN LSTM were developed in this work. The first one is 1D CNN LSTM where we have flattened the images into 1D and concatenated them into one vector, then a convolutional layer was applied to it and the product was fed into the LSTM portion of the model. However, in the 2D CNN LSTM approach, the clip frames were flattened into a 2D matrix and a 2D convolution was applied. The product was followed by the LSTM block. In our project adding CNN didn't improve performance and has shown redundancy. This is because the pre-trained model already contains CNNs and convolutional layers.

4. Bidirectional LSTM:

Another version of LSTM is bidirectional LSTM which is an upgrade of vanilla LSTM. It extracts more features of the input sequence by stepping through input time steps in both

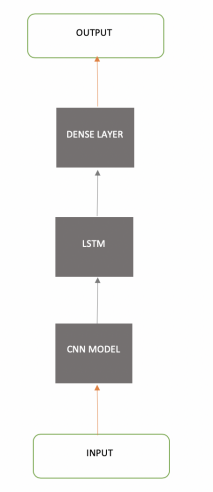

Figure 2.9. CNN LSTM Block

directions forward and backwards[44]. In practice, this architecture duplicates the first LSTM layer so that there are two layers side by side as shown in Figure 2.10.

5. Stacked Bidirectional LSTM:

   Stacked Bidirectional LSTMs are very similar to bidirectional LSTM in that features are extracted by stepping through time steps in both directions forward and backwards[44]. However, stacked bidirectional LSTMs have multiple layers of bidirectional LSTMs. As shown in Figure 2.11, a bidirectional LSTM layer passes a sequence output rather than a single value output to the following bidirectional LSTM layer.

2.4 Pipe Line and Training

As previously stated, the goal of this work is to classify tennis serves and to build a generic system that would learn new features and process them in real-time.

Pipeline Summary:

• Input:

Tennis Serves Videos

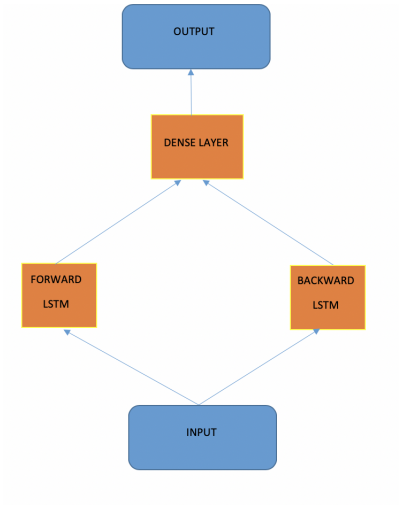

Figure 2.10. Bidirectional LSTM

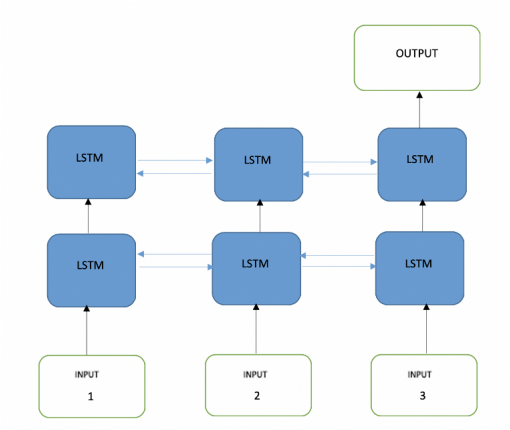

Figure 2.11. Stacked Bidirectional LSTM

Label Each video with a grade: professional/ amateur

- System:

  Analyze the video

  Extract Human pose:

    3D joints coordinates

  Process the 3D joints coordinates

    Feed the joints coordinates series and their label to the LSTM Neural Network

  Train the LSTM Neural Network

  Test the network

- Output:

  Professional/ amateur serve

The input of the model is short videos of tennis serves recorded at 30 fps. Their duration is 3-5 seconds. For this task, we have labeled the data according to serve proficiency: Professional or Amateur. The model is composed of two main submodels: Human pose feature extractor which performs as features encoder and an LSTM based classifier that processes and analyzes the simplified and extracted features.

The videos were processed in chunks using python scripts and the features were extracted using the first submodel which served as a features encoder. The first submodel processes the frames one by one on different levels. A pre-trained image Resnet-50[20] is used to extract features out of the human-based images. The extracted features are of the human pose at time step t_0. Then a temporal encoder f_{movie} learns the 3D human representation Φ_t over the temporal window centered at frame t. From Φ_t we predict the 3D human pose and shape as well as the deviation in the pose in neighboring frames $\pm\Delta(t)$. The primary loss is a 2D reprojection error with an adversarial prior to make sure the recovered poses are valid. The hallucinator h takes a single image feature map and learns to hallucinate its temporal representation and feeds the hallucinated features to neighboring frames thus leading to a smoother movement[32].

The outputs of this model are sequences of 25 3D joints coordinates of the human skeleton per frame, saved as a JSON file. The skeletons are plotted in 3D, as well as their projection on the 2D plane. For this research and the data collection, we have plotted the skeleton per frame, creating a video and investigated and observed it to verify that the pipeline has been successful, to this point, in extracting the correct and relevant features. The skeleton was further processed, shifted to the origin, and normalized by dividing by the length of the spine to reduce the variety and minimize the required features to be learned. The different samples were also randomly rotated to vary the recording angles and to prevent any unintentional orientation bias. The processed skeletons were inserted as an input to train the second submodel. We have developed multiple experiments and applied multiple variations of LSTM architectures in the second submodel:

1. Vanilla LSTM
2. Stacked LSTM
3. CNN LSTM
4. Bidirectional LSTM
5. Stacked Bidirectional LSTM

Different combinations of LSTM inner parameters were used to fine-tune the model. Adam optimizer has been used with a learning rate of 0.001. The dataset was split into a training and a validation set according to protocol. Multiple depths and LSTM variations have been experimented. LSTM architecture of 4 layers has generated the best results. For the best and final results, we have developed stacked bidirectional LSTM architecture.

Table 2.1. Model Architecture

<table>
<tr><td>Block</td><td colspan="3">Description</td></tr>
<tr><td>Encoder:</td><td colspan="3"></td></tr>
<tr><td>Input
Resent-50

Average Pool Layer
Resnet-50 output
f_{movie}

Hallucinator

$f_{3D}regressor$

output</td><td colspan="3">Rescale image to 224x224
Resnet-50[20] based architecture
pretrained on 3D human pose[15]

Per image features vector $\phi_t \in R^{2048}$
Every 13 frames are processed into Φ_t
f_{movie} serves as a temporal encoder
It is composed of 3 sequential residual blocks[32]
Composed of 2 fully connected layers
with filter size of 2048
output added to original ϕ as a skip connection
Composed of 2 fully connected layers
with 1024 neurons and a dropout layer
takes ${\Phi_t}^j$ and outputs ${\Phi_t}^{j+1}$
Pose feature vector $\Theta = [\beta, \theta, \pi] \in R^{85}$
3D human joint coordinates are extracted $\in R^{75}$</td></tr>
<tr><td>Decoder:</td><td colspan="3"></td></tr>
<tr><td>LSTM Architecture</td><td colspan="3"><table>
<tr><th>Layer</th><th>Output Shape</th><th>Number of trainable parameters</th></tr>
<tr><td>Masking Layer</td><td>(3100,3)</td><td>0</td></tr>
<tr><td>bidirectional LSTM</td><td>(3100, 18)</td><td>936</td></tr>
<tr><td>bidirectional LSTM</td><td>(3100, 18)</td><td>2016</td></tr>
<tr><td>bidirectional LSTM</td><td>(3100, 18)</td><td>2016</td></tr>
<tr><td>bidirectional LSTM</td><td>(3100, 36)</td><td>5328</td></tr>
<tr><td>bidirectional LSTM</td><td>(36)</td><td>7920</td></tr>
<tr><td>Dense Layer</td><td>(1)</td><td>37</td></tr>
</table></td></tr>
<tr><td>Output:</td><td colspan="3">Amateur/ Professional Serve</td></tr>
</table>

Chapter 3

Results

As mentioned previously, we compare the performance of the different developed architectures. The different models perform differently and output different accuracies and losses. As expected, we have noticed that LSTM is very beneficial in classifying temporal-related tasks. The best performance was achieved when stacked bidirectional LSTM was used. This is due to the complex mechanism and the efficient temporal feature extraction in the LSTM blocks. In addition to the accuracy and loss achieved while training and testing on different samples collected from the same subjects, we have tested the models on samples collected from unseen subjects. This means that we test whether the models could generically generate proficiency feedback on unseen subjects that might have different individual professional/ amateur features. If successful, this implies that the models learned the real features of professional and amateur serves, and not the subjects serve techniques.

We have also plotted the different training and validation loss and accuracy convergence plots. These plots were very useful in fine-tuning and comparing the models afterward. We observe that different LSTM architectures have different performance and give different outcomes. For our task and sports videos related tasks in general, LSTM based models generate promising outcomes. As expected, more complex LSTM architectures extract more information and give better results and outcomes. In our case stacked bidirectional LSTM models gave better classification results and the best performance. We can also infer from the smooth convergence

Table 3.1. Training Summary

Experiment	T Accuracy	T Loss	Val Accuracy	Val Loss
Vanilla LSTM	0.805	0.390	0.817	0.403
Stacked LSTM	0.903	0.306	0.863	0.342
Bidirectional LSTM	0.954	0.145	0.953	0.141
Stacked Bidirectional LSTM	0.953	0.145	0.956	0.150

Table 3.2. Test Results on unseen subjects

Experiment	Accuracy	Loss
Vanilla LSTM	0.643	0.665
Stacked LSTM	0.857	0.367
Bidirectional LSTM	0.791	0.413
Stacked Bidirectional LSTM	0.857	0.282

in stacked bidirectional graphs that the problem became easier and more convex when compared to a single LSTM or other architectures. When we compare the validation or the test results when sampled from the same pool of samples and the same subjects, bidirectional LSTM generated good results with better accuracies when compared to other architectures. However, when analyzing the performance on unseen subjects, a harder classification task, we see that the accuracy of stacked bidirectional LSTM has performed significantly better than others and we may say that this model has learned generic features of amateur and professional serves.

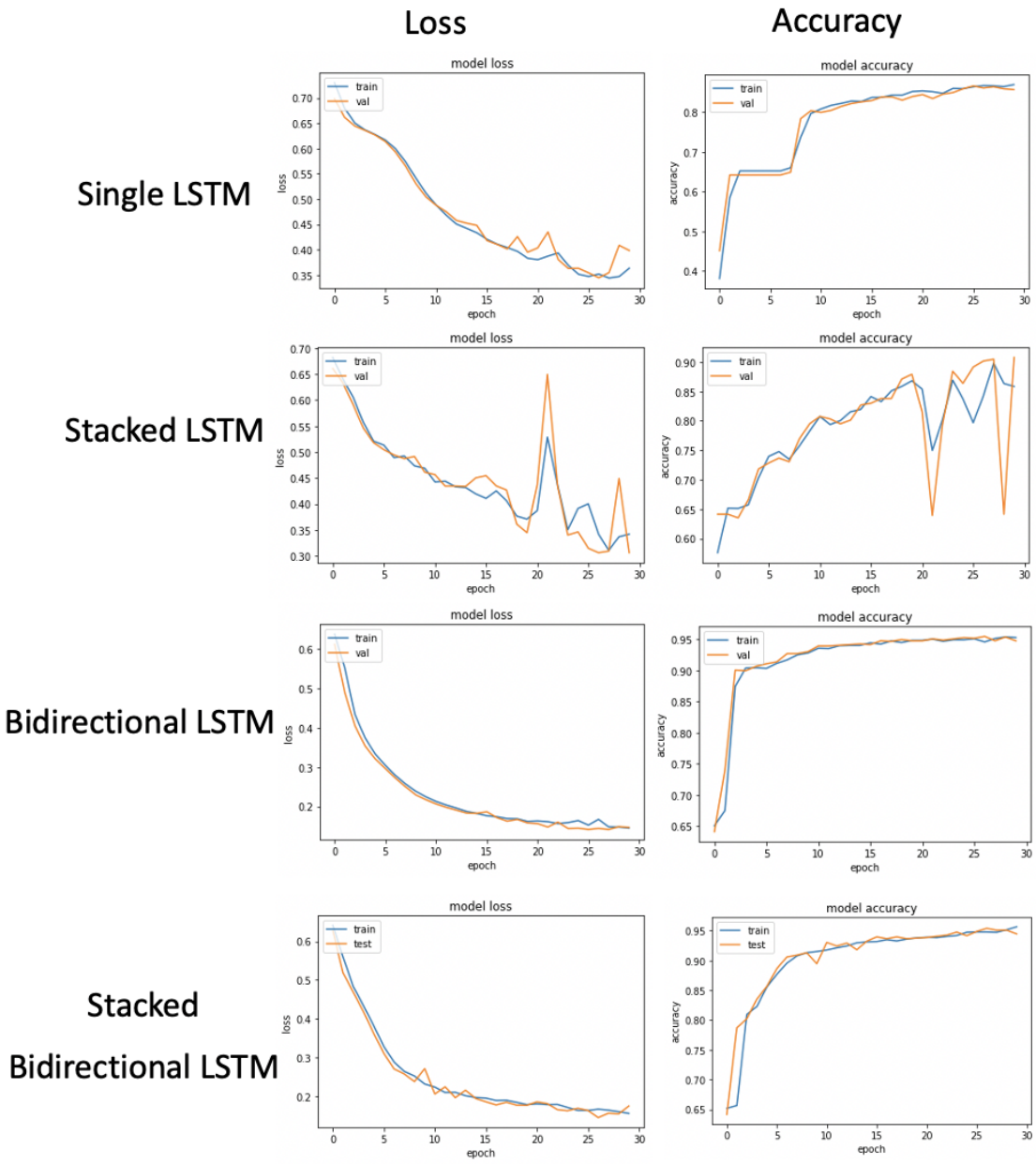

Figure 3.1. Training and accuracy convergence plots

Chapter 4

Discussion and Conclusion

In this work, we have implemented multiple approaches and experiments in order to achieve our goal. We looked forward to developing an end-to-end efficient architecture that would solve sports performance classification problems. We have built multiple submodels and compared their performance.

1. Through experimenting with multiple approaches to human pose features extraction, we were able to support the claim that 3D information is very important when handling human-related problems, especially in human sports tasks and when fine movements are involved. This was obvious when we compared the performance of 2D human joints extractors vs 3D human joints extractors. Although OpenPose is very accurate in estimating the human joints in 2D, the generated information wasn't sufficient in providing the necessary features for performance classification. We argue that adding depth information will facilitate most computer vision tasks. This is a very crucial observation and gives us the motivation to invest in developing efficient and accurate depth estimation algorithms and 3D human pose estimation algorithms.

2. We were able to observe different human pose estimation algorithms. OpenPose is a state-of-the-art algorithm that extracts a very accurate 2D human pose estimation. This algorithm doesn't produce depth information or 3D joints estimation. We have developed

and harnessed multiple cameras to estimate the 3D human pose information. This showed good results when applied to single images. However, we weren't satisfied with the results when applied to video frames due to error propagation and inflation.

3. Data acquisition has been very challenging in this task. The unavailability of sufficient samples and data has led us to harvest pre-trained models that facilitated the learning process by extracting relevant features to the task at hand. This approach of transfer learning has simplified the problem by significantly reducing the dataset size. Without harvesting pre-trained encoders, the required dataset size is estimated to be hundreds of thousands of samples as compared to the thousands of samples used in this work.

4. The positive results and successful training has led us to conclude that human pose estimation may be used to simplify complex sports-related problems. This approach may be applied to other tasks and this pipeline may be used to classify and give other sports performance feedback.

5. The proposed pipeline and model can be used in other sports performance tasks. It can be easily be applied to baseball, golf, or even basketball. With more data, the problem can be easily expanded in the future to give more feedback about the performance by ranking the tennis serve or the sports task instead of binary classification. Moreover, a more generic system can be built by adding an action recognition block at the beginning of the proposed pipeline in order to classify other sports tasks first and rank the performance afterward. In other words, the system would classify whether the video is a tennis serve, a baseball swing, or a basketball throw first, and only afterward it would classify the player's performance.

We hope that our work can be used in the future as a base for solving more complex tasks. The proposed architecture can be applied to many other problems. By increasing the dataset size and enhancing the video labeling, the model would give a more accurate performance assessment.

We look forward to sharing our work and dataset and keep improving it in the near future.

Bibliography

[1] S. Aslam, "Youtube by the numbers", *https://www.omnicoreagency.com/youtube-statistics/*, 2018.

[2] H. L. Wang, J. Huang, Z. Liu, Y. Wang, Y. Chen, and E. K. Wong, "Integration of multimodal features for video scene classification based on hmm", *IEEE Workshop on Multimedia Signal Processing*, pp. 53–58, 1999.

[3] Z. Liu, J. Huang, and Y. Wang, "Classification tv programs based on audio information using hidden markov model", *Workshop on Multimedia Signal Processing*, 1998.

[4] M. Paluri, D. Tran, L. Bourdev, R. Fergus, and L. Torresani, "Learning spatiotemporal features with 3d convolutional networks", *Computer Vision and Pattern Recognition*, 2015.

[5] H.-C. Shih, "A survey on content-aware video analysis for sports", *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, vol. 99, no. 9, 2017.

[6] J. Wang, C. Xu, and E. Ching, "Automatic sports video genre classification using pseudo–2d–hmm", *Proc. IEEE Int'l Conf. Pattern Recogn*, pp. 778–781, 2006.

[7] A. Ekin and A. M. Tekalp, "Shot type classification by dominant color for sports video segmentation and summarization", *Proc. IEEE Int'l Conf. Acoustics, Speech and Signal Process.*, pp. 173–176, 2003.

[8] E. Jaser, J. Kittler, and W. Christmas, "Hierarchical decision making scheme for sports video categorization with temporal post-processing", *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, pp. 908–913, 2004.

[9] L. Li, N. Zhang, L. Y. Duan, Q. Huang, J. Du, and L. Guan, "Automatic sports genre categorization and view-type classification over large-scale dataset", *Proc. ACM Multimedia*, pp. 653–656, 2009.

[10] X. Yuan, W. Lai, T. Mei, X. S. Hua, X. Q. Wu, and S. Li, "Automatic video genre categorization using hierarchical svm", *Proc. IEEE Int'l Conf. Image Process.*, pp. 2905–2908, 2006.

[11] M. Marchi, J. Albert, W. Lai, T. Mei, X. S. Hua, X. Q. Wu, and S. Li, "Analyzing baseball data with r. chapman and hall/crc press", *Taylor Francis Group*, 2013.

[12] D. Oliver, "Basketball on paper: Rules and tools for performance analysis", *Potomac Books*, 2004.

[13] S. M. Shea, "Basketball analytics: Spatial tracking", *CreateSpace Independent Publishing Platform*, 2014.

[14] S. M. Shea and C. E. Baker, "Basketball analytics: Objective and efficient strategies for understanding how teams win", *CreateSpace Independent Publishing Platform*, 2013.

[15] A. Kanazawa, M. J. Black2, D. W. Jacobs, and J. Malik, "End-to-end recovery of human shape and pose", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7122–7131, 2018.

[16] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields", 2019.

[17] T. Luhmann, S. Robson, S. Kyle, and J. Boehm, "Close-range photogrammetry and 3d imaging", 2013.

[18] G. Welch and G. Bishop, "An introduction to the kalman filter", 2001.

[19] F. Bogo, "Keep it smpl: Automatic estimation of 3d human pose and shape from a single image", *ECCV*, 2016.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks", *ECCV*, pp. 630–645, 2016.

[21] N. Silberman and S. Guadarrama, "Tensorflowslim image classification model library",

[22] C. Ionescu, "Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments", *PATTERN ANALYSIS AND MACHINE INTELLIGENCE (PAMI)*, vol. 36, no. 7, pp. 1325–1339, 2014.

[23] D. Mehta, H. Rhodin, D. Casas, P. Fua, O. Sotnychenko, W. Xu, and C. Theobalt, "Monocular 3d human pose estimation in the wild using improved cnn supervision", *3D Vision (3DV), 2017 Fifth International Conference on*, 2017.

[24] J. Carreira, "Human pose estimation with iterative error feedback", *Computer Vision and Pattern Recognition, CVPR*, 2016.

[25] P. W. M. Oberweger and V. Lepetit, "Training a feedback loop for hand pose estimation", *IEEE International Conference on Computer Vision*, pp. 3316–3324, 2015.

[26] P. W. P. Dollar and P. Perona, "Cascaded pose regression", *Computer Vision and Pattern Recognition, CVPR*, 2010.

[27] X. Zhou, "Deep kinematic pose regression", *ECCV Workshop on Geometry Meets Deep Learning*, pp. 186–201, 2016.

[28] S. Johnson and M. Everingham, "Clustered pose and nonlinear appearance models for human pose estimation", *BMVC*, pp. 12.1–12.11, 2010.

[29] M. Andriluka, "2d human pose estimation: New benchmark and state of the art analysis", *Computer Vision and Pattern Recognition, CVPR*, 2014.

[30] T. Lin, "Microsoft coco: Common objects in context", *European Conference on Computer Vision (ECCV)*, 2014.

[31] L. Sigal and M. Black, "Humaneva: Synchronized video and motion capture dataset for evaluation of articulated human motion", *Brown University, Tech*, 2016.

[32] A. Kanazawa, J. Y. Zhang, P. Felsen, and J. Malik, "Learning 3d human dynamics from video", *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[33] W. Zhang, M. Zhu, and K. G. Derpanis, "From actemes to action: A strongly-supervised representation for detailed action understanding", *CVPR*, pp. 2248–2255, 2013.

[34] D. F. Fouhey, W. Kuo, A. A. Efros, and J. Malik, "Lifestyle vlogs to everyday interactions", *CVPR*, 2018.

[35] Rumelhart, Hinton, G. E, Williams, and R. J, "Learning internal representations by error propagation", *ICS*, 1985.

[36] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[37] C. Kyunghyun, van Merrienboer Bart, G. Caglar, B. Dzmitry, B. Fethi, S. Holger, and B. Yoshua, "Learning phrase representations using rnn encoder-decoder for statistical machine translation", 2014.

[38] J. Brownlee, "Long short-term memory networks with python", 2017.

[39] G. Cybenko, "Approximation by superpositions of a sigmoidal function", *Mathematics of Control, Signals and Systems*, vol. 2, pp. 303–314, 1989.

[40] K. HorniK, Maxwell-Tinchcombe, and Halbertwhite, "Multilayer feedforward networks are universal approximators", *Neural Networks*, vol. 2, pp. 359–366, 1989.

[41] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to construct deep recurrent neural networks", 2014.

[42] M. Hermans and B. Schrauwen, "Training and analysing deep recurrent neural networks", *Advances in Neural Information Processing Systems*, pp. 190–198, 2013.

[43] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks", *ICASSP*, 2013.

[44] Z. Cui, R. Ke, Z. Pu, and Y. Wang, "Stacked bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction", 2018.