



AMERICAN UNIVERSITY OF BEIRUT

The benefits of synthetic data for action  
categorization

by  
Mohamad Ballout

A thesis  
n submitted in partial fulfillment of the requirements  
for the degree of Master of Engineering  
to the Department of Mechanical Engineering  
of the Faculty of Engineering and Architecture  
at the American University of Beirut

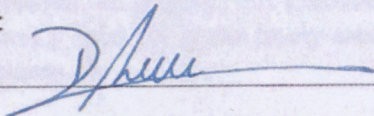
Beirut, Lebanon  
July 2019

# AMERICAN UNIVERSITY OF BEIRUT

## The benefits of synthetic data for action categorization

by  
Mohamad Ballout

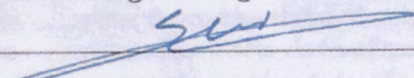
Approved by:



Dr. Daniel Asmar, Associate Professor

Advisor

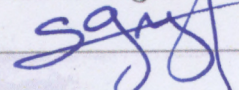
Mechanical Engineering



Dr. Elie Shammas, Associate Professor

Member of Committee

Mechanical Engineering



Dr. George Sakr

Member of Committee

Electrical and Computer Engineering

Date of thesis defense: July 4, 2019

# AMERICAN UNIVERSITY OF BEIRUT

## THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name: Ballout Mohamed Adnan  
Last First Middle


Master's Thesis       Master's Project       Doctoral Dissertation

I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes after: **One** \_\_\_ year from the date of submission of my thesis, dissertation or project.

**Two** \_\_\_ years from the date of submission of my thesis, dissertation or project.

**Three** ← years from the date of submission of my thesis, dissertation or project.

 4/7/19  
Signature Date

This form is signed when submitting the thesis, dissertation, or project to the University Libraries

# Acknowledgements

First and foremost, I would like to thank Allah, the Almighty and merciful for giving me the strength, knowledge, and opportunity to undertake this research study and to persevere and complete it satisfactorily. He provided me with power when I needed it the most, without his blessings, this achievement would not have been possible.

I am deeply thankful to my advisor, mentor, and my role model Professor Daniel Asmar for giving me support, help, and guidance throughout the masters years. He helped with his academic advises as well as providing motivation to push me forward to accomplish my goal.

Also, I would like to thank Professor Elie Shamma and Professor George Sakr who played a huge role in this achievement. Their comments and questions were always on point and helped me to proceed and finish my thesis.

Thanks to my labmates for the fun and the support. I am proud to be a member of the Vision and Robotics Lab at AUB. Many thanks to Abbas Sidawi, Mohamad Kassem EL Zein, Maya Antoun, Sevag Babikian, Mohamad Tuqan, and Rema Daher. Thank you for listening to me and giving advises throughout this process.

Last but not least, a very special thanks to my parents and my fiance. My parent who raised me to believe I can achieve anything that I set in my mind. I cannot thank you enough for being great support emotionally and financially. Finally, to my beloved fiance Jinan, thank you for being in my life.

# An Abstract of the Thesis of

Mohamad Ballout for Master of Engineering  
Major: Mechanical Engineering

Title: The benefits of synthetic data for action categorization

In this thesis, we will show the importance of video analysis using deep network. We are going to introduce some deep learning methods to detect and recognize faces in a video stream as well as emotion recognition. The three preceding networks are based on image analysis systems. Another way of analyzing videos is to do action recognition system, where the order of the frames become important. We propose a new 3DCNN+LSTM system for action recognition. However, the proposed system did not outperform the state of the art systems on UCF-101 dataset. In fact, it scored around 80% on UCF-101 while the state of the art system scores above 90%. In addition, we studied the value of using synthetically produced videos as training data for neural networks used for action categorization. Motivated by the fact that texture and background of a video play little to no significant roles in optical flow, we generated simplified texture-less and background-less videos and utilized the synthetic data to train a Temporal Segment Network (TSN). The results demonstrated that augmenting TSN with simplified synthetic data improved the original network accuracy (68.5%), achieving 71.8% on HMDB-51 when adding 4,000 videos and 72.4% when adding 8,000 videos. Also, training using simplified synthetic videos alone on 25 classes of UCF-101 achieved 30.71% when trained on 2500 videos and 52.7% when trained on 5000 videos. Finally, results showed that when reducing the number of real videos of UCF-25 to 10% and combining them with synthetic videos, the accuracy drops to only 85.41% from 96.60%, compared to a drop to 77.4% when no synthetic data is added.

# Contents

<b>Acknowledgements</b>	<b>v</b>
<b>Abstract</b>	<b>vi</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Convolutional Neural Network</b>	<b>4</b>
<b>3 Convolutional Neural Network to Analyze videos</b>	<b>8</b>
3.1 Face detection . . . . .	8
3.2 Face Recognition . . . . .	10
3.3 Emotion Recognition . . . . .	12
<b>4 Action Recognition Literature Review</b>	<b>14</b>
4.1 Action Recognition Methods . . . . .	14
4.2 The Use of Synthetic Data . . . . .	16
<b>5 Failure Case: a 3DCNN + LSTM Network</b>	<b>18</b>
5.1 3D-CNN . . . . .	18
5.2 LSTM . . . . .	18
5.3 3DCNN + ConvLSTM . . . . .	21
5.4 Results . . . . .	22
<b>6 Methodology</b>	<b>25</b>
6.1 The Value of Appearance Data in Action Categorization . . . . .	25
6.2 Synthetic Data . . . . .	26
<b>7 Results</b>	<b>30</b>
7.1 Datasets . . . . .	30
7.2 Networks tested . . . . .	33
7.3 Analysis of the Results . . . . .	34
7.3.1 The Effect of Background Removal . . . . .	36
7.3.2 Effect of Adding Synthetic Data to Training . . . . .	36
7.3.3 The Effect of Decreasing the Number of Real Videos . . . . .	37

7.3.4	Robustness Test . . . . .	40
7.3.5	Reduce Overfitting . . . . .	40
<b>8</b>	<b>Conclusion</b>	<b>41</b>



# List of Figures

1.1	Synthetic data can be used to train a network from scratch or augment a pre-trained network to improve its performance. . . .	2
2.1	Regular neural network(left) takes 1-d input while convolutional neural network(right) takes 3-D input . . . . .	5
2.2	The result of a convolution over an image is the element-wise product and sum of the filter matrix and the original image. . . . .	6
2.3	Convolution network includes an convolutional layers, pooling layers, activation functions, and fully connected layers. . . . .	7
3.1	Haar-kernel moves across the image in order to detect edges . . .	9
3.2	Results showing YOLO face detection to the left Haar Cascade face detection to the right . . . . .	10
3.3	Triplet loss function optimizes the embedding directly. . . . .	11
3.4	Friends faces recognized using live webcam stream. . . . .	12
3.5	Friends faces recognized using live webcam stream. . . . .	13
5.1	3D-CNN includes an additional dimension that helps the network to conserve the temporal relationships between the frames . . . .	19
5.2	RNN's have loops that enables them to hold information of previous layers . . . . .	20
5.3	LSTM have the ability to maintain information for long time dues to its ability to learn to drop the useless features and keep the useful ones. . . . .	21
5.4	Model create by [1] combined 3DCNN with ConvLSTM and 2DCNN to do gesture recognition . . . . .	22
5.5	Our proposed model to do action recognition combining 3DCNN with ConvLSTM. . . . .	24
6.1	The simplified data were generated using Unity without setting up a 3D scene . . . . .	27
6.2	Examples of videos created with background (left column) and simplified videos (right column) . . . . .	28

7.1	UCF-101 dataset includes 101 classes such as diving, walking, riding horse etc. . . . .	30
7.2	HMDB-51 dataset includes 51 classes such as haircut, rafting, shaving beard etc. . . . .	31
7.3	Sample of the created synthetic dataset with background . . . . .	32
7.4	Sample of the created synthetic dataset with background without background . . . . .	32
7.5	TSN is designed to have a longer memory by choosing snippets of the videos . . . . .	33
7.6	Networks tested, Augmented TSN with 4 and 3 streams, Optical Flow TSN, and 3D-ResNet . . . . .	35

# List of Tables

5.1	The effect of changing parameters on the network . . . . .	23
6.1	Number of videos created of objects corresponding to those found in benchmark datasets: background videos stands for the videos that were made using a 3D setup. . . . .	27
7.1	Network 2 versus Network 1: note that domain randomization is effective in simulating real outdoor data . . . . .	36
7.2	The effect of combining real and simplified synthetic videos on sub-datasets. . . . .	37
7.3	Benchmarking Network-2 versus state-of-the-art networks . . . . .	38
7.4	The effect of decreasing the number of real videos on the accuracy of UCF-25 while adding 2500 and 5000 synthetic videos . . . . .	39

# Chapter 1

## Introduction

In the drive towards a pervasive Internet-of-Things (IoT) society, machines will have to interact much more with humans, and to do so, they will have to understand human actions and activities. Action categorization is the process of classifying a trimmed video that contains a single action. For example, a four second video of a person biking should be classified as ‘biking’. On the other hand, the process of detecting the time interval of each action in a video and labeling them is called temporal action detection. In this thesis, we are dealing with the problem of action categorization.

The state-of-the-art action categorization systems are mostly built on deep network architectures. The major challenge for these networks is the collection and annotation of a sufficient number of videos for training. The deeper the networks are, the more data is required. To give some perspective to the problem, training the 3D ResNet [2] on the UCF-101 dataset failed, despite the fact that UCF-101 contains more than thirteen thousand videos of 101 classes. In fact, to successfully train 3D ResNet, a much larger dataset (Kinetics [3]) was required, which includes more than 300,000 videos. Manually annotating all these videos is a daunting task, and thus the need for a method that generates annotated videos in a simpler manner (Fig. 1.1).

One approach to overcome the requirement for large amounts of data is to do unsupervised action localization, which does not require any annotated videos, and aims to automatically group videos of a similar action into one class. Such systems rely on local features that can detect similar actions. Unfortunately, the results of these systems to date [4][5] are not comparable to supervised systems, with accuracy values as low as 60% on UCF-101.

Another approach to mitigate the problem of video collection and annotation, is to rely on simulated data instead of videos of real recorded actions. Such approaches have been attempted on still images generated through graphics simulators. For example, synthetic data was used in object detection [6][7]; it was also used in segmentation [8], and also in the evaluation of optical flow solutions [9]. Creating simulated realistic full scenes with complete background information is

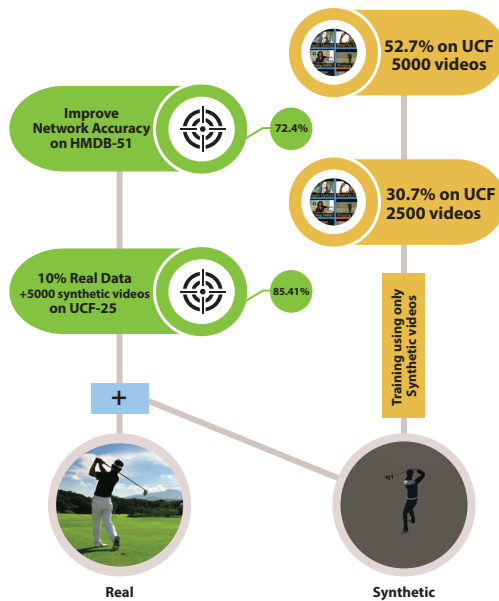


Figure 1.1: Synthetic data can be used to train a network from scratch or augment a pre-trained network to improve its performance.

difficult, and not many people have attempted it [10]; however, what is interesting to note is that the difficult part is mostly in the creation of the background. In fact, if one could disregard the background completely, creating actions in simulation would be relatively simple to do.

The contributions of this thesis include the following:

- First, we are going to explain multiple applications of CNN including face detection, recognition, and emotion recognition.
- Second, we propose a new 3DCNN + LSTM system that scores around 80% on UCF-101 dataset.
- Third, we prove the efficiency of using simplified synthetic data for action recognition, in which only optical flow data is considered. Domain randomization is applied by shaking the camera in a random fashion, as well as changing the lighting conditions in the recorded videos.
- Fourth, augmenting the vanilla TSN [11] by including as input an additional stream of synthetic data on top of the real videos. Our proposed augmentation outperforms the vanilla TSN by a significant margin.

- Fifth, training the TSN with only the generated synthetic dataset resulted in 52.7% accuracy when tested on the sub-dataset of UCF-101.
- Finally, we release a dataset of synthetic data with all the actions chosen from classes of benchmark datasets, including HMDB-51 and UCF-101. We name the datasets S-HMDB-38 and S-UCF-25, and make them publicly available for testing.<sup>1</sup>

---

<sup>1</sup><https://www.dropbox.com/s/isz6vsw6fzibbwk/datasets.zip?dl=0>

## Chapter 2

# Convolutional Neural Network

Deep learning is rather an old concept that was introduced in the 80's. It is a subset of machine learning algorithms that consists of multi-layer neurons designed based on the humans' brains. However, it faced an issue back-then due to its need to a huge computational power. With the advance of computational technology and the discovery of GPUs, the deep network has raised again and dominated the machine learning competitions since 2012. 2012 was a remarkable year in the deep learning history when Alex net [12] achieved an error of 15.3% on Imagenet competition more than 10.8 percentage points lower than that of the runner up. This result was a huge turn in the machine learning field and since then deep learning methods dominated and were used in most of the AI applications. In this section, we are going to explain Convolutional Neural Network (CNN) and in the next section, we will go through multiple applications that were done using it. First, we are going to apply face detection using a typical machine learning method called cascade and compare it with a deep learning method on videos. Second, two face recognition methods are applied and compared. Finally, a CNN network is used to detect the emotion of a person from a webcam.

Convolutional Neural Network have been successful in image competitions due to its explicit assumption that the inputs are images. Unlike the regular neural network, which takes an input of a single vector figure 2.3 [13](left), CNN takes an input of 3 dimensional neurons as shown in figure 2.3(right). The 3 dimensions are usually width,height ,and depth where the depth is equal to three when the image is RGB and to one when it is gray scale. Also, an additional feature to convolutional neural network is that neurons in a hidden layer are connected to a small region of the precedent layer instead of being fully connected. This feature gives the CNN a great advantage over regular nets as it reduces the number of parameters calculated.

The convolution operation is done by multiplying a sliding window, which is also called a kernel or a filter, over an image. The result of the element-wise product and sum of the original image with the filter is the output of the convolution, which is also called feature maps. The mathematical equation of the

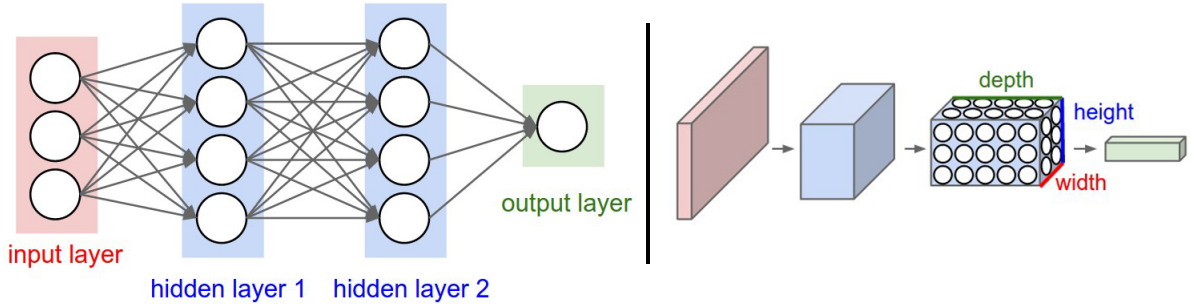


Figure 2.1: Regular neural network(left) takes 1-d input while convolutional neural network(right) takes 3-D input

convolution could be written as:

$$O(i, j) = (I * K)(i, j) = \sum_m \sum_n I(i + m, j + n)K(m, n) \quad (2.1)$$

where  $O$  is the output feature map,  $I$  is the input image, and  $K$  is the kernel.  $m, n$  are the width and height of the kernel, and  $i, j$  are the pixel at which the convolution is happening. As shown in figure 2.2 [14], the  $3 \times 3$  filter is slid over the original  $6 \times 6$  image to give a feature map of  $4 \times 4$ . The size could be kept as  $6 \times 6$  by using padding. Padding is adding pixels of zero to the border of the image in order to maintain the original size. Convolutional neural network is the combination of the steps mentioned above over multiple layers as shown in figure 2.3 [15]. At each layer multiple filters of altered values are applied in order to detect different types of features. In addition, a common practice in convolutional networks is to do Max pooling after each layer. Max pooling is used to reduce the spatial size of the feature maps in order to decrease the amount of parameters in the convolutional network. It is therefore one way to control overfitting in the system. Max pooling is a function that is applied on a part of the image ( $2 \times 2$  for example) where the maximum pixel is kept and the remaining values are discarded. Another function used in order to introduce non-linearity in the network is the activation function. Rectified Linear Unit, known as RELU, is an activation function that is widely used in order to apply element-wise non-linearity. RELU could be mathematically defined as:

$$f(x) = \max(0, x) \quad (2.2)$$

At the end of the network, the last two layers are usually fully connected layers where the number of neurons of the last layer is equal to the number of classes that we are trying to classify. For example, in a binary classification network, it could be designed to have a final layer of two neurons. Each neuron represents the probability of the input image being class  $a$  or  $b$ . Finally, the convolutional neural network could be summarized as follows (Fig.2.3)



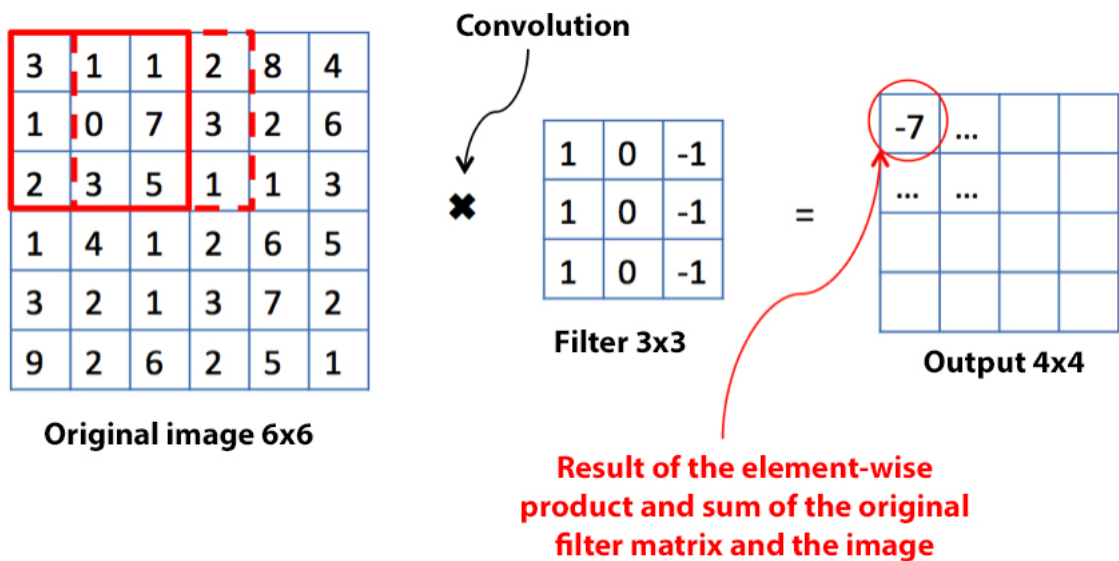


Figure 2.2: The result of a convolution over an image is the element-wise product and sum of the filter matrix and the original image.

- The input that is a raw pixel values of an image having three dimensions of height, width, and depth(RGB channels).
- Convolutional layer that gives the element-wise product and sum of the original image with the filter.
- Pooling layer serves as a downsampling function in order to reduce the number of parameters in the system.
- Activation function such as RELU to introduce non-linearity to the network.
- A 1-D Fully connected layer used to compute the final scores of each class.

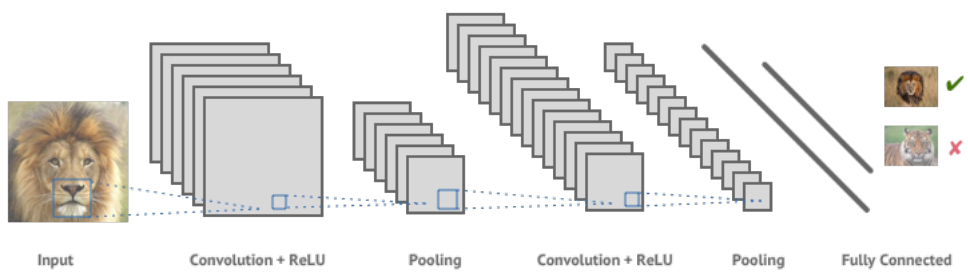


Figure 2.3: Convolution network includes an convolutional layers, pooling layers, activation functions, and fully connected layers.

## Chapter 3

# Convolutional Neural Network to Analyze videos

Convolutional neural networks are currently the state of the art when it comes to understanding images and videos. CNN are superior to any other methods in image classification, object detection etc. In this section, we present applications such as face detection, face recognition, and emotion recognition that use CNN. The input of all of the network presented below are videos from our webcam. The videos are usually decomposed into multiple frames and fed to the networks as images. Thus, dealing with videos is similar to dealing with images when applying face recognition or emotion recognition because the system is classifying each frame as a single image. In the next chapter, when trying to recognize an action going on in a video, the relation between frames will become important.

### 3.1 Face detection

In this section, we are going to compare a well know machine learning method to detect faces to a convolutional network in order to show the superiority of the CNNs. The two methods were tested to detect faces; one is Haar cascade method and the other is the well know object detection system YOLO (You Only Look Once). Haar cascade is a machine learning classifier that is trained to detect objects using positive and negative images. It is well known for being able to detect faces. Haar-kernel moves across the image in order to detect edges, lines, and centers as shown in figure 3.1[16]. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. In our case, I used the implemented Haar cascade method in Open-CV, which is pre-trained on the FERET dataset [17]. The dataset contains a total of 14,126 images that includes 1199 individuals and 365 duplicate sets of images. A duplicate set is a second set of images of a person already in the database and was usually taken on a different day. In the training process,

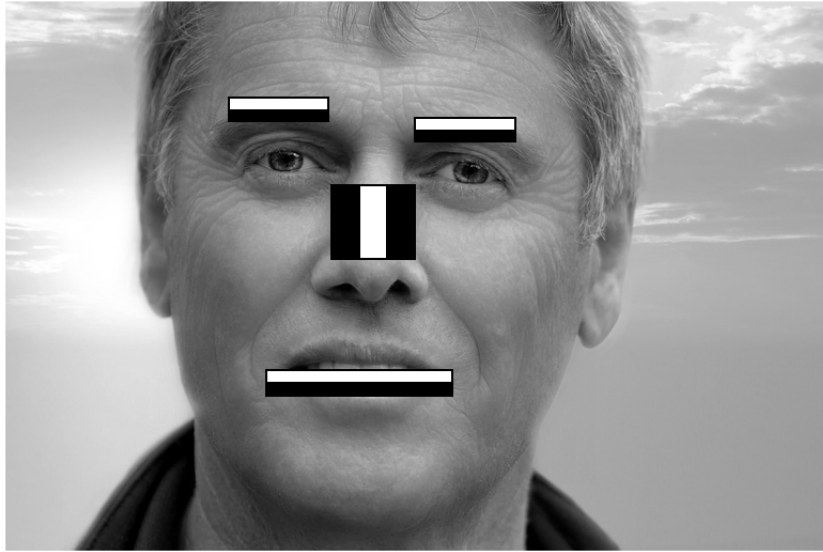


Figure 3.1: Haar-kernel moves across the image in order to detect edges

features are extracted from the images. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under the white rectangle from sum of pixels under the black rectangle.

Another robust way of detecting faces is using the famous deep learning-based method YOLO. YOLO is a famous, robust, and fast object detection method done by Redmond et. al [18]. It is based on a deep convolutional neural network that has 24 convolutional layers followed by 2 fully connected layers. The system used is pre-trained on the ImageNet data (224 224 input image) and then the resolution is doubled for detection. I used a finetuned YOLO on a dataset called Wider Face [19]. Wider Face dataset is a face detection that contains 32,203 images and label 393,703 faces with a high degree of variability in scale, pose and occlusion. We tested both networks on the same images and the results were as expected. YOLO, which is a convolutional network, showed a better performance where it was able to detect occluded faces as well as faces that are not looking directly to the camera as shown in figure 3.2. The left pictures in figure 3.2 are the output of the YOLO network whereas the right column shows pictures detected using HAAR Cascade method. For example, in the top picture, YOLO detected only the three faces without any false positives whereas the machine learning method detected 7 faces, 4 of them are not present. In addition, in the bottom picture, YOLO was able to detect the face of the first lady to the right even though it is not complete whereas HAAS Cascade was not able to detect it.



Figure 3.2: Results showing YOLO face detection to the left Haar Cascade face detection to the right

## 3.2 Face Recognition

Another important application in understanding videos is the ability to not only to detect faces, but recognize it as well. It could be used for security, educational, and managerial purposes. For example, a webcam could be installed at the entrance of an office, and the attendance of each employee could be detected using face recognition system. State of the art systems done by Google and Facebook in face recognition are now at the brink of human level accuracy. DeepFace[20] (Facebook) and FaceNet[21] (Google) have accuracies of 97.35% and 99.63% respectively on Labeled Faces in the Wild (LFW) dataset [22]. The dataset contains more than 13,000 images of faces collected from the web. Each face has been labeled with the name of the person pictured. 1680 of the people pictured have two or more distinct photos in the data set.

DeepFace is a recognition method that applies the conventional pipeline in face recognition Detect - Align - Represent - Classify Their addition was in the alignment and the representation of the figures. They created a new way of aligning faces using 3D alignment. They employed an explicit 3D face modeling in order to apply a piecewise affine transformation and derive a face representation from a nine-layer deep neural network. The other contribution done by DeepFace is the network used. They used a 9 layers deep learning network that has around 120 million parameters. The first 3 layers are regular convolutional layers whereas

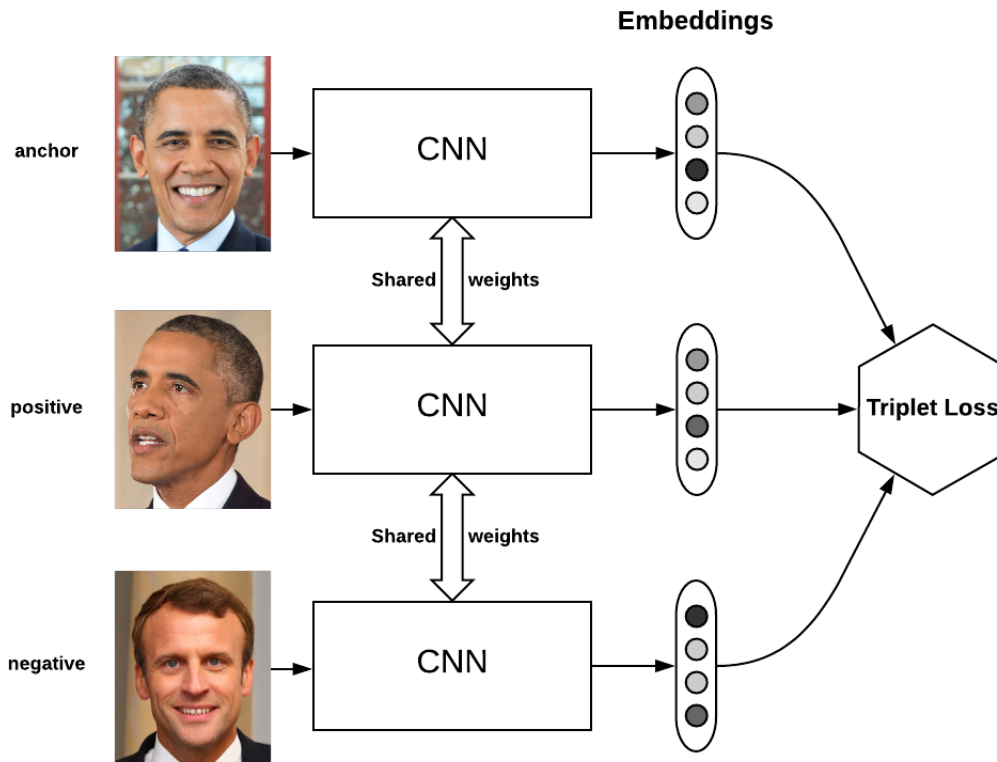


Figure 3.3: Triplet loss function optimizes the embedding directly.

L4, L5 L6 are like convolutional layers but every location in the feature map learns different set of filters. Finally, DeepFace could be done using an end-to-end metric learning approach, known as Siamese network once learned, the face recognition network (layer F8) is replicated twice (one for each input image) and the features are used to directly predict whether the two input images belong to the same person. This is done by learning a function that differentiates between the vector feature of each image.

On the other hand, FaceNet uses a method where deep convolutional network trained to directly optimize the embedding itself, rather than an intermediate layer as in DeepFace. The training is done using a triplet loss function. Triplets consist of two matching face thumbnails and a non-matching face thumbnail and the loss aims to separate the positive pair from the negative by a distance margin. For example, to learn the parameters of the network, the system looks at pair of pictures at the same time. The embeddings of Obamas pictures should be similar, while the embedding of Obamas and Macrons images should be different. Therefore, the Triplet Loss minimizes the distance between an anchor and a positive, both of which have the same identity, and maximizes the distance between



Figure 3.4: Friends faces recognized using live webcam stream.

the anchor and a negative of a different identity as shown in figure 3.3.

I ran a demo of FaceNet on my webcam and the results are shown in figure 3.4. I used a pre-trained model of FaceNet and fine-tuned it on one picture of each person recognized in the figure below. The system only requires one picture of the person in order to be able to recognize it.

### 3.3 Emotion Recognition

Finally, convolutional neural network could also be used in emotion recognition of faces in videos. Emotion recognition is usually done by analyzing the facial expression of the person at the first place. The voice of the person could also be analyzed to detect emotion. However, in our case, we are only considering facial expression as they are usually enough to detect the emotion of a person. Thus, information on the facial expressions are often used in automatic systems of emotion recognition. The system I used is taken from [23]. They used a mini-Xception model in order to be applied in real time. The accuracy is 66% on FER-2013 [24] while the state of the art is around 71%. The FER-2013 dataset contains 35886 gray scale images that have 7 emotions: Happy, Sad, Surprise, Disgust, Fear, Calm, Anger. Figure 3.5 shows snapshot a real-time webcam capture.

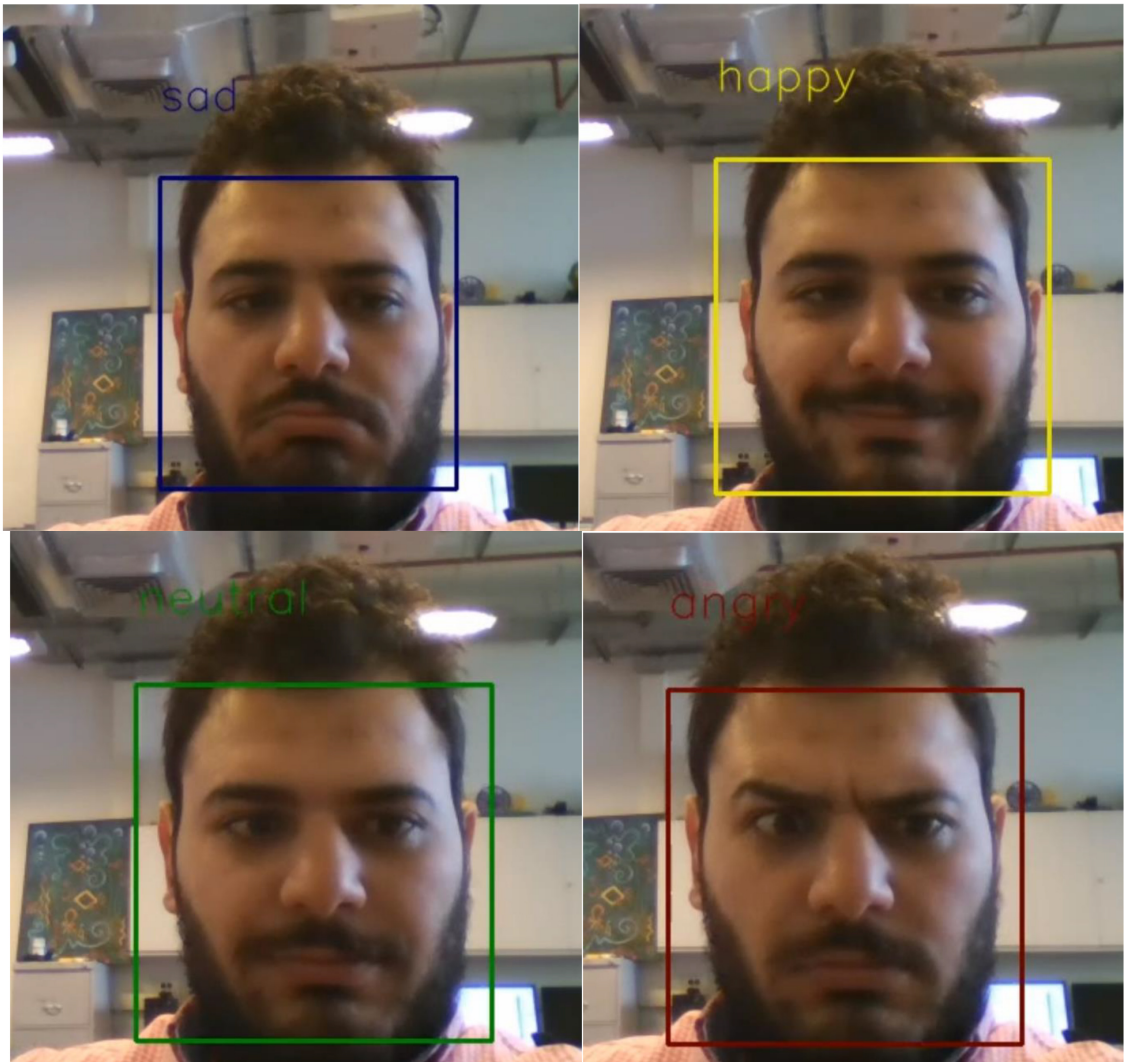


Figure 3.5: Friends faces recognized using live webcam stream.



# Chapter 4

## Action Recognition Literature Review

In this section, we are going to go through a literature review over the action recognition methods as well as the use of synthetic data in deep learning.

### 4.1 Action Recognition Methods

Action detection or categorization is a current active area of research due to its importance and its several real-world applications. It could be used in industrial applications such as inspecting the employees behavior and effort. Also, it could be used for security purposes to detect suspicious actions. First it is important to differentiate between action detection and categorization. Action detection is usually used in long videos to detect and classify several actions. For example, one of the widely datasets used is the MPII Cooking Dataset. The videos show actions taken in the kitchen where the person do variety of movements such as peeling, cutting, washing etc. The process of detecting each action in the video and labeling them is called action detection. On the other hand, action categorization is the process of recognizing and labeling a single action to be selected out of N number of actions in a trimmed video. Usually, the temporal component is more important in action detection than in action recognition since the detection process involves longer duration videos. Usually, the term fine-grained action detection is used to indicate how delicate and subtle the differences between the identified classes are. For example, in order to grasp the difficulty of the task, consider the difference between peeling, cutting, and grating. Hence, the action detection is usually considered to be a more difficult subject than action categorization. There are several approaches to solve an action detection problem. Usually, the difficulty lies in the temporal localization of actions in long videos. The standard approach in action detection is to split the videos into multiple frames. Since they have showed superiority in detecting features

of images, Convolutional Neural Networks are used for detecting spatial features of each frame. A fully connected layer is used from the CNN structure located before the activation layer represented in a vector form. Usually, this is the common part of any action detection structure. Next, in combining the spatial and detecting temporal features, different approaches are considered. One poor approach is to simply pool each frame features extracted from CNN to detect the actions. This method discards the importance of the temporal order of the frames and does not give efficient results. Cherian et al. [25] invented a novel pooling method that preserved the temporal order of the frame. Even though it showed promising results, it did not match the current state of art results in action recognition. A better approach that matches the state of art results is to use Recurrent Neural Network after extracting the spatial features using CNN. RNN can handle different input and output lengths as opposed to CNN which is restricted to a fixed size input. The idea behind the RNN is to make use of the sequential information. It allows understanding the video in a sequential form such that the previous frame information is passed to the current input in addition to the new frame. Thus, RNN memorizes what computations it has done to a certain extent. However, RNN is still considered to have a short memory meaning that the last layers might forget the initial information. RNN has an updated version called LSTM referring to Long Short-Term Memory which extends its memory. LSTM do not have a different structure than the RNN, but they differ in their ability to be trained to forget the inadequate information and remembering the important one. Thus, a good architecture for action detection is to use the output of a CNN (fully connected layer) as an input to LSTM [26]. Another approach would be an updated version of a CNN. It is called a 3D convolutional neural network [27]. The third additional dimension is time. However, this way only covers a short range of the sequence before it starts forgetting the initial frames features. Singh et al. [28] proposes a method where the CNN is updated to become a multi stream CNN. The additional stream includes information about the motion in order to capture more details for the temporal features. In more details, they applied CNN to optical flow representations. The last two methods we mention in this literature are the convolutional two-stream network [29] and the Temporal Segment Network [11]. The two stream flow applies CNN to both streams, which one of them represents spatial features of the frames and its input is a regular RGB frames. The other flow is to detect the motion between the frames and has an input of optical flow pictures. TSN is an upgraded version of the two stream network as it uses a deeper network, and it benefits from a 2D-CNN model trained on ImageNet. Another advantage of TSN versus a classic two stream convolution network, is its ability to model long-range temporal structures by extracting sparse snippets of the video, instead of using consecutive frames that are highly redundant.

## 4.2 The Use of Synthetic Data

Training deep network models using synthetic data has shown promising results in multiple computer vision applications, such as object detection [6][30][7][31], segmentation [32][8][33], optical flow estimation [9][34], action recognition [10][35][36][37], and pose estimation [38][39]. Some of these contributions are in the methods used for generating synthetic data. For instance, Dwibedi [7] suggested an easy yet effective way to generate synthetic images using what they call ‘cut and paste’. Cropped pictures were placed inside any background picture to create realistic training images. The most important point in their contribution is that images are generated quickly, and their system outperforms the existing synthesis detection approaches by 21% when combined with real data.

Another important aspect, referred to as cross-domain generalization, is the ability of synthetic data to generalize to real world data. It would not be useful to train a system on synthetic data, and then have the trained network produce worse detection and classification on real world data. To address this problem, Tobin [40] used domain randomization [34] for object detection; the concept is based on introducing random variations in the simulator in such a manner that the world, after randomization, appears different to the AI system. In the simulator of Tobin [40], the parameters for lighting, pose, and textures were set in a random and non-realistic manner. Their idea was to provide enough variability when training in a way that the system would then be able to generalize to the real world during testing. By training only on synthetic data, their proposed network produced compelling results on a benchmark object detection dataset.

The method we are proposing is similar in spirit to that of De Souza [10]; however, three main differences exist. First, the network we use is an *augmented* TSN, which includes an additional stream or two for synthetic data compared to their cool-TSN, which feeds the synthetic and the real data together using mini-batches. Second, we propose using a reduced form of synthetic data (hereafter referred to as *simplified synthetic data*), which leads to a better accuracy than that of [10]. Third, the synthetic dataset we created and used for training is considerably smaller than that of [10] (8529 versus 39,982), and yet the improvement we achieve is higher than what they achieve (3.9% versus 1% on HMDB-51). Our experiments are more comprehensive, in which we test our augmented TSN on different inputs, sometimes using synthetic data alone, and others using a mixture of real and synthetic data.

To the best of our knowledge, along with [37], only a few papers assess the effect of training a network on synthetic videos alone, and test it on a dataset such UCF-101. Our results revealed that the more synthetic data is used to train a network, the higher the accuracy becomes.

In this thesis, we propose to evaluate the advantage of using simplified synthetic data for training neural networks for action categorization. We test our approach on the Temporal Segment Network (TSN) [11], which learns both spa-

tial and temporal streams separately; the temporal stream is fed with optical flow fields, and is not affected by background. For instance, if the action to be modeled is ‘diving from a cliff’, whether the surrounding environment is simulated in the scene or not, the optical flow frames are not affected as long as the background is static. Thus, our videos generated using a physics engine, such as Unity, are background-free and no complex scene design was required.

## Chapter 5

# Failure Case: a 3DCNN + LSTM Network

In this section we are going to combine 3D-CNN with LSTM in a novel way. But first, we are going to take a closer look on what 3DCNN and LSTM are

### 5.1 3D-CNN

As its name indicates, 3D-CNN is the three dimensional version of CNN. As mentioned in the previous sections, and since CNN is the state of the art in image classification, [27] tried to extend this superiority to action recognition where we are dealing with videos. Since videos are bunch of assembled images, 3D-CNN includes as an additional dimension that is considered as time (Fig.5.1). The filters used to convolve over the images are now 3D filters in order to conserve the temporal relation between the frames over time. Also, as shown in the bottom of figure 5.1, and similar to the regular CNN, 3D-CNN has multiple convolutional layers as well max-pooling and fully connected layers. In [27], they applied deconvolution on conv5b feature maps in order to visualize what the 3D-CNN is learning. They discovered that the 3D-CNN starts by focusing on appearance in the first few frames and then tracks the motion in the subsequent frames, which proved the ability of 3D-CNN to learn both motion and appearance. Also, similar to the regular CNN, 3D-CNN is could be pre-trained on large video datasets Sports-1M, which contains more than million sport videos. In fact, 3D-CNN has to be pre-trained on large datasets in order to avoid overfitting when trained on small datasets.

### 5.2 LSTM

Recurrent Neural Network (RNN) is a special kind of neural networks that handles sequential data such as voices, texts, videos etc. Similar to other deep

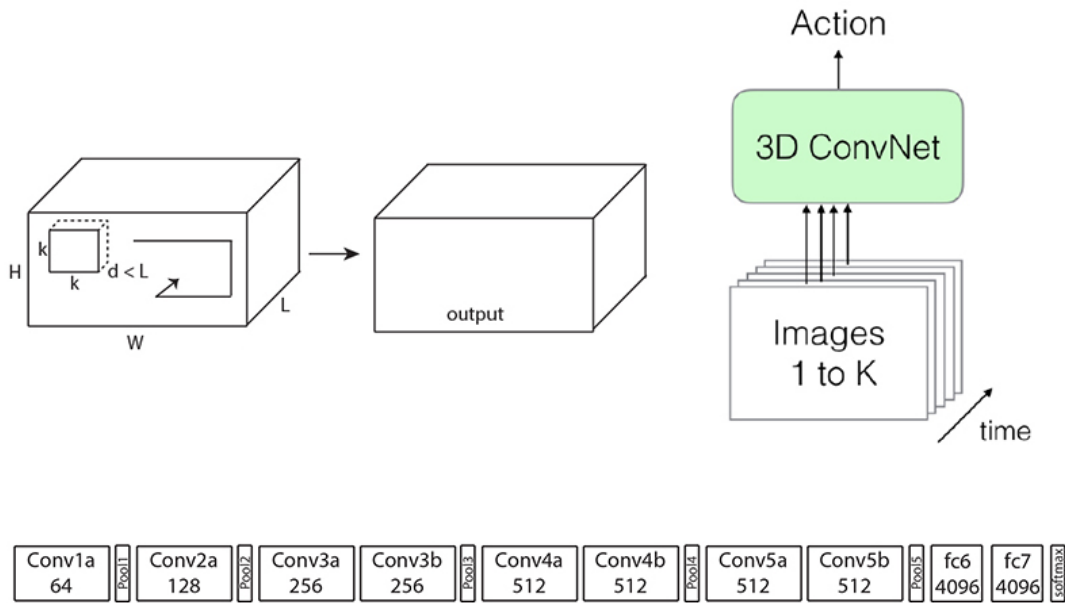


Figure 5.1: 3D-CNN includes an additional dimension that helps the network to conserve the temporal relationships between the frames

learning algorithms, RNN's are relatively old and were invented in the 1980's. Unlike CNN who accepts a fixed size input such as an array of an image, RNN's have a more flexible structure where it can take a sequential input or a sequential output. It is also capable of having both sequential input and output. In addition to this flexibility, RNN's are designed to have internal memory as they are in fact the only networks to have one. The memory, which enables the network to persist information, is done by creating loops inside the network that allow information to be passed from a hidden layer to another as shown in figure 5.2. The equations of RNN could be represented such as :

$$h_t = g(W_{xh} * x_t + W_{hh} * h_{t-1} + b_h) \quad (5.1)$$

$$z_t = g(W_{hz} * h_t + b_t) \quad (5.2)$$

where  $g$  is a non-linearity function such as sigmoid,  $x_t$  is the input,  $h_t$  is the hidden layers,  $N$  is the hidden state with  $N$  hidden units, and  $z_t$  is the output at time  $t$ .  $W$  is the weight between two layers, and  $b$  is the bias term. However, there are two problems with the regular RNN's called "exploding gradients" and "vanishing gradients". Exploding gradients means that the system gives a huge number for the weights without a logical reason. This problem could be

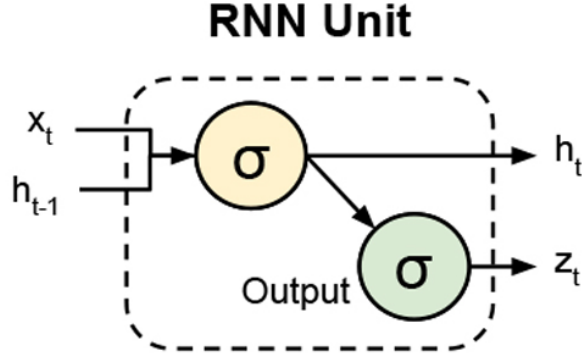


Figure 5.2: RNN's have loops that enables them to hold information of previous layers

solved by normalizing the weights. However, the second problem, which is the vanishing gradients, which means the gradients of the system become too small and eventually vanish. This problem was solved by creating an updated version of RNN's called "Long Short Term Memory". LSTM networks, proposed by [41], are special type of recurrent neural network RNN that has the ability to memorize long-term variables and solves the vanishing gradients problem. LSTM's have the ability to maintain the information for long time because of they are able to delete useless information and keep the useful one. The mathematical equations of LSTM could be shown as follow :

$$i_t = g(W_{xi} * x_t + W_{hi} * h_{t-1} + b_i) \quad (5.3)$$

$$f_t = g(W_{xf} * x_t + W_{hf} * h_{t-1} + b_f) \quad (5.4)$$

$$o_t = g(W_{xo} * x_t + W_{ho} * h_{t-1} + b_o) \quad (5.5)$$

$$g_t = \tanh(W_{xc} * x_t + W_{hc} * h_{t-1} + b_c) \quad (5.6)$$

$$c_t = f_t ** c_{t-1} + i_t ** b_c \quad (5.7)$$

$$h_t = o_t ** \tanh(ct) \quad (5.8)$$

where \*\* denotes element-wise product between the vectors, and i is input gate ,f is forget gate ,o is output gate,g is input modulation gate , and c is memory cell unit. More details about LSTM could be found in [41] and [26] where the above equations and figures 5.2 and 5.3 are taken from.

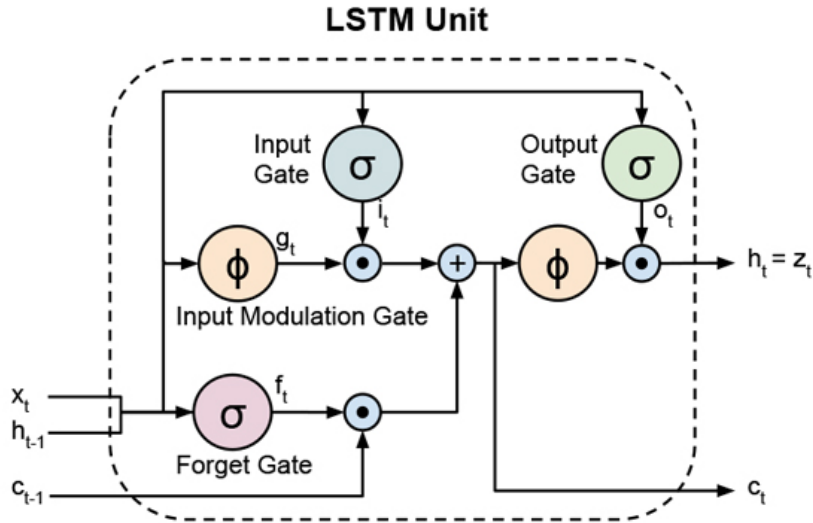


Figure 5.3: LSTM have the ability to maintain information for long time dues to its ability to learn to drop the useless features and keep the useful ones.

### 5.3 3DCNN + ConvLSTM

Our model for 3DCNN+LSTM was inspired by a gesture detection algorithm made by [1] and shown in figure 5.4. However, our system differs from their system by discarding the 2DCNN layers at the top of the model and using one directional ConvLSTM. These two simplifications are made to decrease the number of parameters in the system. ConvLSTM designed by [42] differs from the LSTM explained in the previous section by its ability to take an image as an input. The Vanilla LSTM takes vectorized features as an input, which results in the loss of the spatial information of an image. Thus, ConvLSTM is an upgraded version of LSTM that is identical to LSTM except for its hidden layers that are convolutional layers. Another main difference between our proposed system and system made by [1] is the way that the 3DCNNs' outputs are fed to the ConvLSTM. Instead of feeding the last fully connected layer, we fed each cell of the LSTM by convolutional hidden layer as shown in figure 5.5. The idea behind this adjustment was to explicitly help the ConvLSTM to have a longer memory since it was stated by [27] that the first few layers of the 3DCNN are responsible of the spatial information while the subsequent layers are responsible of the temporal ones.



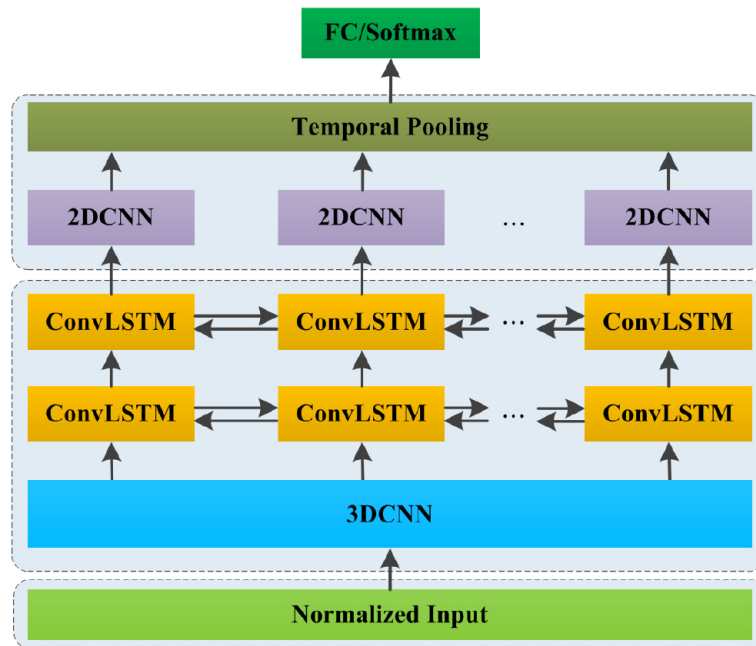


Figure 5.4: Model create by [1] combined 3DCNN with ConvLSTM and 2DCNN to do gesture recognition

## 5.4 Results

We used a compact 3DCNN that has only 4 hidden layers and fed them to the ConvLSTM network. The 3DCNN was first trained on Sports-1M and then trained on UCF-101. We tried different set of parameter shown in table 5.1. The parameters changed were kernel size, dropout, batch size, optimization method, number of filters, input image size and epochs. The last combination in table 5.1 was the best one, where we achieved an accuracy of 80.1% on UCF-101 dataset. More details about the dataset is given in section 7. The results achieved which is 80.1% is way below the state of the art systems, which achieve above 90% accuracy on this dataset as shown later in table 7.3. Thus, we tried a different way to improve the accuracy of another deep learning system called TSN using synthetic dataset.

Table 5.1: The effect of changing parameters on the network

Kernel size	Dropout	Batch size	Optimization method	Number of filters	Input Image size	Epochs	Test Accuracy
1x1x1	0.5	2	RmsProp	32	16x16x15	50	64.0
3x3x3	0.5	2	RmsProp	32	16x16x15	50	67.5
3x3x3	0.8	2	RmsProp	32	16x16x15	50	63.3
3x3x3	0.5	10	RmsProp	32	16x16x15	50	68.3
3x3x3	No Dropout	10	RmsProp	32	16x16x15	50	60.0
5x5x5	0.5	10	RmsProp	32	16x16x15	50	69.0
5x5x5	0.5	10	RmsProp	64	16x16x15	50	69.2
5x5x5	0.5	10	RmsProp	32	16x16x15	50	69.1
5x5x5	0.5	10	ADAM	32	16x16x15	50	65.1
5x5x5	0.5	10	RmsProp	32	16x16x15	50	71.6
5x5x5	0.5	10	RmsProp	64	16x16x15	100	73.3
5x5x5	0.5	10	RmsProp	64	16x16x30	50	78.3
5x5x5	0.5	10	RmsProp	64	50x50x50	75	80.1

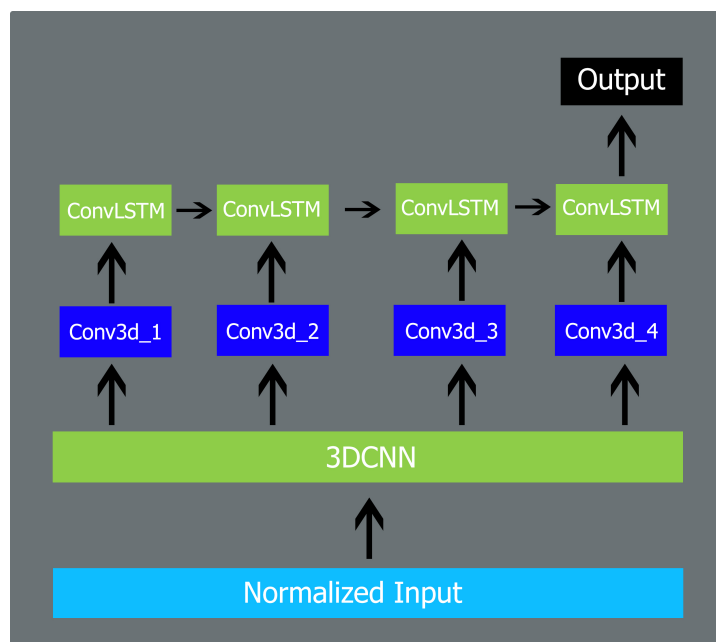


Figure 5.5: Our proposed model to do action recognition combining 3DCNN with ConvLSTM.

# Chapter 6

## Methodology

This section presents the proposed methodology, including an analysis on the value of appearance data versus optical flow in action categorization, as well as the assessment of what part of synthetic data is most relevant for the sake of action categorization.

### 6.1 The Value of Appearance Data in Action Categorization

In action recognition networks, it is common to include both spatial and temporal ConvNet streams. In TSN [11], for example, the spatial stream takes RGB frames as input, and the temporal stream processes optical flow. Each stream is trained separately and then their scores are fused at the output layer. When tested on the UCF-101 dataset, after training on the spatial stream alone, TSN scored an accuracy of 84.5%, and when trained on the temporal stream alone, it scored an accuracy of 87.2%. Fusing both streams resulted in an improved accuracy of 92.0%. Looking closer at these results, the relatively moderate improvement attained when adding appearance data questioned its value in action categorization. This result motivated us to use *only* temporal information in our synthetic data; an idea that agrees with the results presented in [43]. Moreover, using only temporal information introduces substantial simplifications to the synthesis of videos, as will be seen below.

Optical flow is the pattern of motion in the visual scene reflecting the relative motion between the object and the camera. For the sake of reproducing the action, a question arises regarding the background of the scene and its impact on the optical flow. Under ideal conditions, where the camera is fixed, and the lighting condition is consistent among all the videos within the dataset, the background does not have a significant contribution to the optical flow pattern. However, in the absence of ideal conditions, relative motion can be detected from the pixels in the background of the scene. This happens, for instance, when a

camera is held by a person who is moving or walking. This random movement or ‘shake’ in a camera can distort the ideal conditions that are replicated in a gaming engine.

One approach to tackle this problem is domain randomization [6] [34], which is used in object recognition. In domain randomization, an object in a scene is placed out of context in order to train the network to deal with the possible variability in the scene appearance when detecting that object. In our thesis, we borrow the idea of randomization from the field object recognition, and apply it to actions by adding random changes in light intensity in the scene, and random camera movements. The variations are applied either within the vicinity of its original position to introduce the shaking effect into the camera, or by tracking the action of interest. Therefore, the synthetic videos we use constitute a combination of scenes under ideal and non-ideal conditions, either by introducing distortion in light intensity, camera position, or a combination of both.

## 6.2 Synthetic Data

Creating realistic backgrounds is one of the main challenges one faces when generating synthetic data. In an outdoor environment, this challenge is even more difficult, requiring graphics experts capable of reproducing computer models of natural objects, such as trees and rocks. It would seem that in the case where background information is critical, the required computer effort could be better spent collecting videos of real scenes. On the other hand, if one could completely disregard the background and create simple videos of foreground alone, synthetic data creation becomes considerably simpler (see Fig. 6.1).

We investigate the significance of appearance on network performance for videos of human actions on simulated data we created, as well as on the benchmark datasets HMDB-51 and UCF-101. The synthetic actions were downloaded from Mixamo.com, or from the Unity asset store. To test for the effect of background, various backgrounds and scenes were downloaded from the Unity asset store. For the HMDB-51 classes, 38 were reconstructed; whereas for the 101 UCF, 25 classes were reconstructed. Table 6.1 shows the number of videos and classes for actions corresponding to those found in the benchmark datasets. It is worth noting that we only synthesized videos for 38 out of the 51 HMDB classes, and 25 out of the 101 UCF classes, since other classes were not available to download.

In each dataset, different characters are used to do the same action; and for each action, there are multiple animations that differ from each other in the way of doing the action. For example, if the action is ‘riding a bike’, there are several ways of riding it: it could be ridden at a quick or slow speed, it could be ridden standing up or sitting down (see Fig. 6.2). Also, for further generalization, some of the actions were done under different lighting conditions such as under dark,

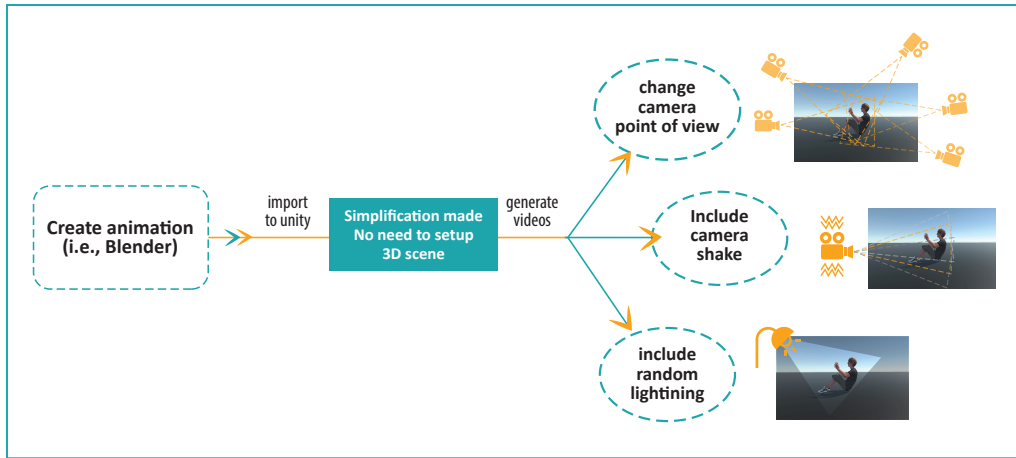


Figure 6.1: The simplified data were generated using Unity without setting up a 3D scene

shadow, or bright lighting conditions. When removing appearance and relying on optical flow alone, none of these nuances are relevant any longer. Optical flow is not affected by the lighting, nor by the color and texture of the garment of a character. This fact allowed us to use the same character for most of the videos where appearance is disregarded.

As a result, in this thesis we suggest using only the optical flow of synthetic data (which is simple to obtain), and disregard the background scene information (which can be difficult to obtain). For example, in producing the ‘climb stair’ action, it is enough to only reproduce the action of climbing the stairs without the need to place the stairs in the scene. For the ‘pitching’ action, it is sufficient to simulate the act of pitching a ball without having to simulate a ball in the scene, as its contribution to the optical flow is negligible. In other words, the only factor

Table 6.1: Number of videos created of objects corresponding to those found in benchmark datasets: background videos stands for the videos that were made using a 3D setup.

<b>Dataset</b>	Classes (#)	Background Videos (#)	Background-less Videos (#)
HMDB-51	38	3817	8528
UCF-101	25	–	5514

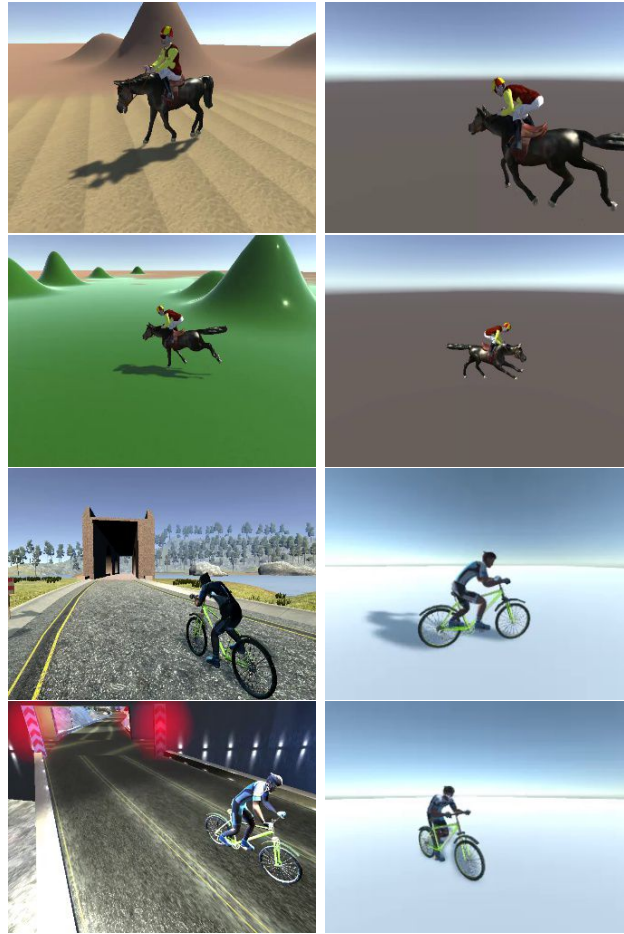


Figure 6.2: Examples of videos created with background (left column) and simplified videos (right column)

contributing in the synthesis of the scene is the action of the character itself.

Each action includes between 200-250 videos, requiring between 60-75 minutes in total to generate. Similar to real videos, the generated videos are between two to six seconds. Camera acquisition is set to thirty frames per second in all of the generated videos. The aspect ratios of the videos are close to those of the UCF-101 videos (320x240 pixels). To complete the dataset, each action is reproduced from different camera viewpoints and environment conditions (light source intensity and camera position). Note however, that not all the actions are one-person synthetic. Some actions such as ‘salsa spin’ or ‘boxing’, require including the second person in order to correctly interpret the overall meaning of the action (two-people synthetic).

The results of this comparison were quite surprising: training using optical flow alone on both the real and synthetic datasets produced classifications results

superior to those in which appearance data was also included. As a result, it was decided that in simulations, we would only vary the camera viewpoint while performing the various actions. Also, purposely shaking the camera in simulations resulted in conditions that are close to what is experienced in the real world.



# Chapter 7

## Results

Several experiments were performed to assess the effectiveness of using synthetic data in training of action recognition networks. In what follows, we first present the datasets we used for the experiments, then discuss the proposed networks, and finally discuss the results of our experiments.

### 7.1 Datasets

Five different datasets were used for the proposed experiments, two of which comprise videos of real actions, including UCF-101 and HMDB-51, and three that are synthetic, which we simulated:

- UCF-101: a dataset of human actions from videos in the wild, containing 13320 videos distributed in 101 action categories. Some captures of the videos are shown in Figures 7.1

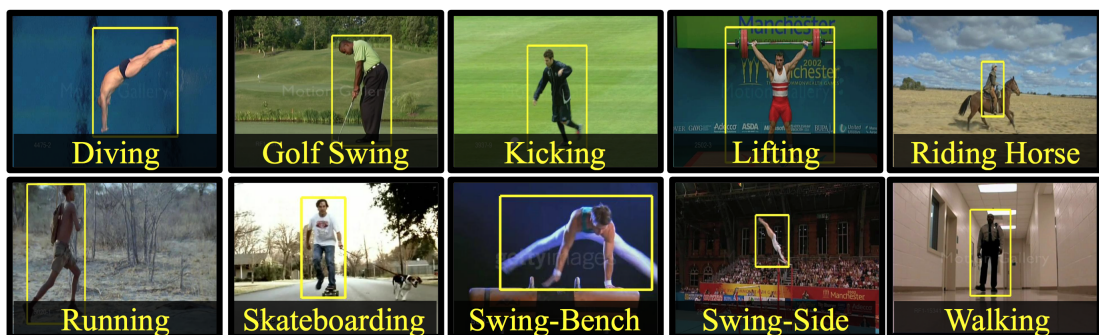


Figure 7.1: UCF-101 dataset includes 101 classes such as diving, walking, riding horse etc.

- HMDB-51: a video database for human actions, mostly extracted from movies and web videos. It consists of 6766 videos from 51 different action categories. Some captures of the videos are shown in Figure 7.2

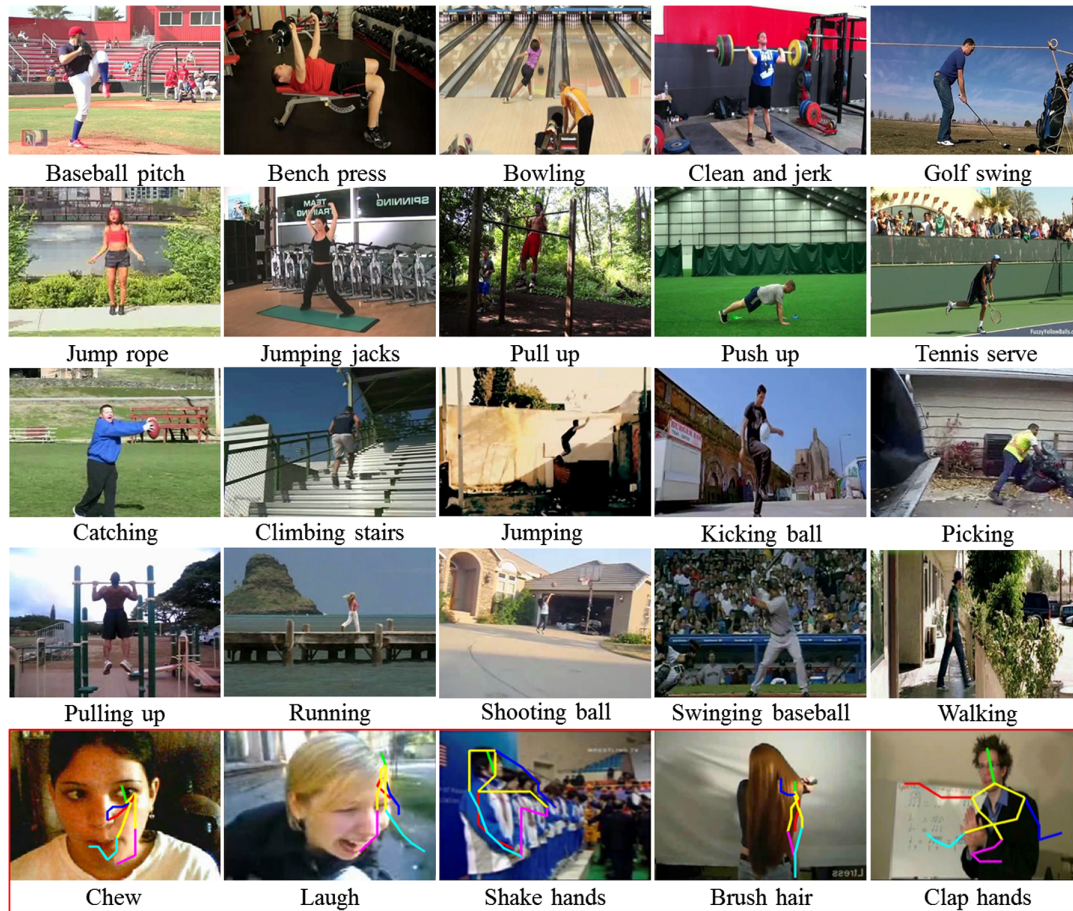


Figure 7.2: HMDB-51 dataset includes 51 classes such as haircut, rafting, shaving beard etc.

- Synthetic-appearance (HMDB-38): approximately 4000 synthetic videos were created including background, chosen from 38 classes of real HMDB-51. Example is shown in Fig. 7.3. Note that the videos are made of low resolution in order to simulate the videos from the real dataset.
- Synthetic-simplified (HMDB-38): approximately 8000 synthetic videos were created using no-background, chosen from 38 classes of real HMDB-51. Example of the video is shown in Fig. 7.4
- Synthetic-simplified (UCF-25): approximately 5000 synthetic videos were created using no-background, from 25 classes of the real UCF-101. Example of the video is shown in Fig. 7.4



Figure 7.3: Sample of the created synthetic dataset with background



Figure 7.4: Sample of the created synthetic dataset with background without background

## 7.2 Networks tested

To investigate the potential of using simplified synthetic data for training, we tested four different networks (Fig. 7.6), including TSN, Augmented TSN (Network-1 and Network-2), and one-flow TSN (Network-3).

**TSN** Temporal Segment Networks [11] are an upgraded version of the two stream convolutional network [29]. TSN uses a deeper network than the two stream network, and it benefits from a 2D-CNN model trained on ImageNet [44]. The 2D-CNN could be Inception, 2D-ResNet, or Batch-Normalized-Inception. Another advantage of TSN versus a classic two stream convolution network, is its ability to model long-range temporal structures by extracting sparse snippets of the video, instead of using consecutive frames that are highly redundant.

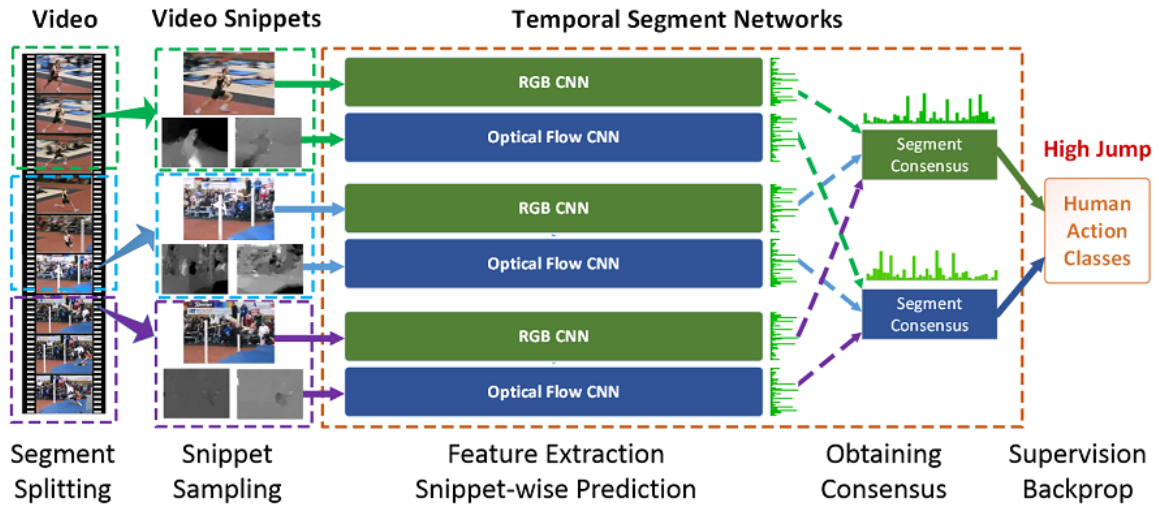


Figure 7.5: TSN is designed to have a longer memory by choosing snippets of the videos

**Augmented-TSN (Network-1).** This network is based on TSN, with two additional streams. The first addition is the spatial stream that takes as input synthesized + real appearance data. The second additional stream is the temporal one, which takes as input the optical flow of both the synthetic and the real datasets. Fusing the score was done by giving the real+synthetic flow a weight of 2.0, real flow 1.0, real RGB 1.0, and synthetic+real RGB 0.5.

**S-TSN (Network-2).** Synthetic-Simplified TSN is based also based on the TSN model, with an addition stream that trains extracted optical flow for both

real and simplified synthetic data. Fusing the score was done by giving all of the streams the same weights.

**One-flow-TSN (Network-3).** This model uses only the temporal stream of TSN. It takes as an input the extracted optical flow of the Synthetic-UCF-25 dataset and it is tested on real videos from UCF-25. The intent of this experiment is to prove that synthetic data alone is sufficient to train a network that is later tested on real data.

**3D-ResNet-18 (Network-4).** Residual Networks in general, 2D or 3D, were created to train very deep networks for action recognition. In order to overcome overfitting, these networks typically require huge amounts of data. For example, 3D Res-Net-101 [2] (the number 101 refers to number of deep layers in the network) was trained on the Kinetic dataset [3], which contains 300k videos. 3D-ResNet uses the same terminology as 3D-CNN, where a 3D convolution filter is applied to RGB frames, with time as the third dimension. However, it uses a more efficient way in learning, called skip connections, or short cuts between layers, which allow resolving the vanishing gradient problem, and enable modeling very deep networks.

In our tests, we are using 3D-ResNet-18 and not 101 because we did not generate 300 k synthetic videos. However, it would seem intuitive that if our small number of synthetic videos is able to reduce overfitting in ResNet-18, a larger number of synthetic videos will surely be able to train ResNet-101. Therefore, the intention of this test is to verify whether synthetic data can help reduce overfitting in a pre-trained network. In this experiment, all our synthetic videos were added to the real training data.

## 7.3 Analysis of the Results

All of the network parameters were tuned to those of the original TSN [11], except for the drop-out ratio, which gave a better accuracy when set to 0.8 for the temporal stream. The network weights were initialized with pre-trained models from ImageNet using BN-Inception [45] as the ConvNet architecture. Also, mini-batch stochastic gradient descent was used with a batch-size set to 128 with a momentum of 0.9. The number of iterations was adopted from the TSN for both spatial and temporal streams. The optical flow frames were extracted using the code provided in the TSN framework. Finally, we followed the traditional evaluation on the three training/testing splits for all of the experiments.

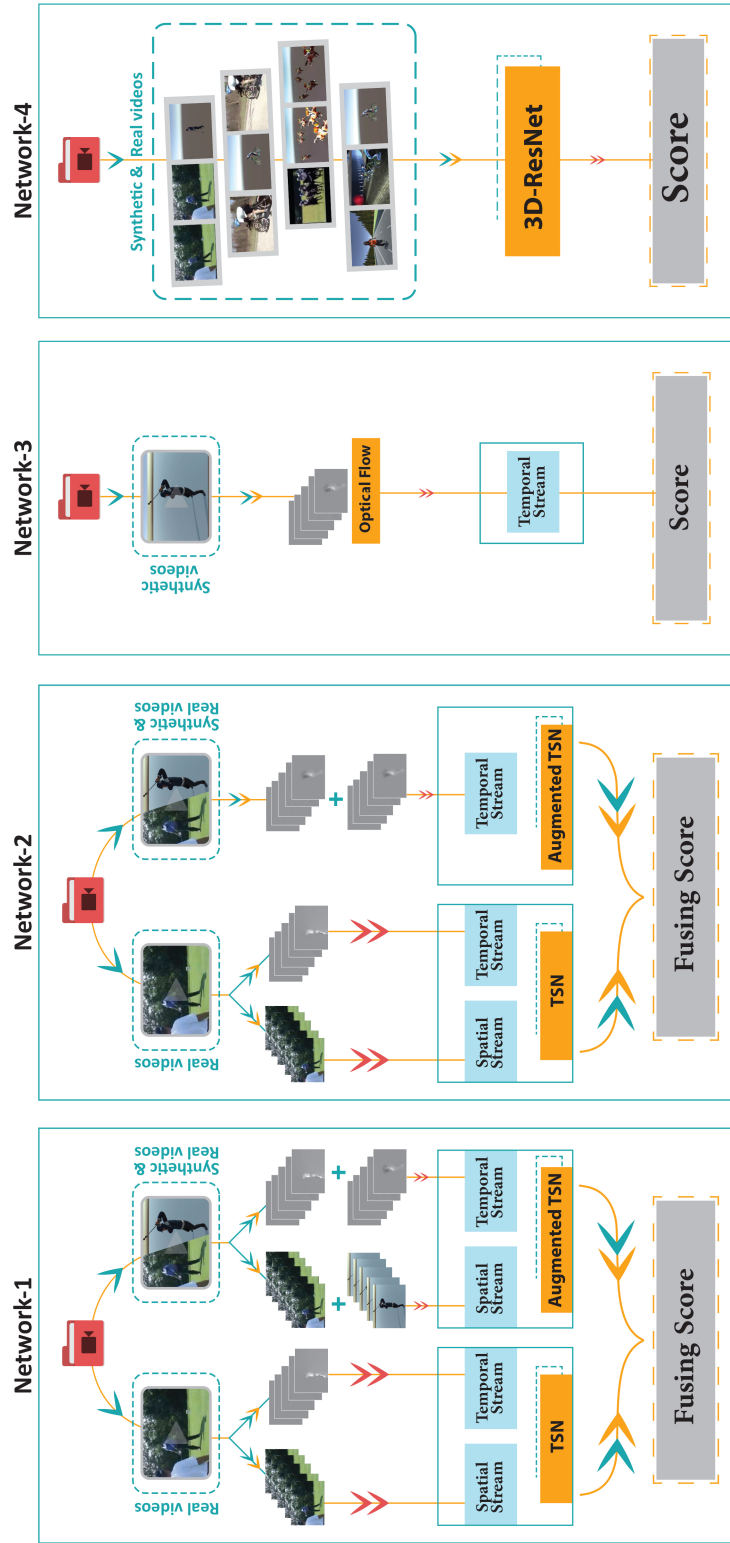


Figure 7.6: Networks tested, Augmented TSN with 4 and 3 streams, Optical Flow TSN, and 3D-ResNet

### 7.3.1 The Effect of Background Removal

In this experiment we tested the effect of using quality versus quantity videos. For Network-1, we added 4000 synthetic videos with an appropriate background setup. For Network-2, we added 4000 simplified videos and in another instance, we added 8000 simplified videos. What we observed ( Table 7.1) was that adding 8000 videos to Network-2 produced slightly better results than Network-1. Even though Network-1 includes an additional stream compared to Network-2, the latter performed better. We concluded from this experiment that a large number of simplified videos can outperform videos with background included.

Table 7.1: Network 2 versus Network 1: note that domain randomization is effective in simulating real outdoor data

Method	HMDB-51
Network-1 with 4000 background synthetic videos	72.3 %
Network-2 with 4000 simplified synthetic videos	71.8 %
Network-2 with 8000 simplified synthetic videos	72.4 %

### 7.3.2 Effect of Adding Synthetic Data to Training

This experiment was divided into two parts: the first one performed on a sub-dataset of HMDB-51 and UCF-101. As mentioned in Section 7.1, we reconstructed (synthetically) 38 classes of the HMDB-51 and 25 classes of the UCF-101. Thus, the training and the testing in this part is performed on 38 classes of HMDB-51 and 25 classes UCF-101. The purpose of this experiment is to study the effect of adding synthetic data to all of the dataset classes. First, we reproduced the TSN results for those sub-datasets using the original real videos. Next, we added half of the generated synthetic videos (2500 for UCF-25, and 4000 for HMDB-38) to a third stream using Network-2. Finally, we added all of the generated videos (5000 for UCF-25 and 8000 for HMDB-38) to the third stream in Network-2. Table 7.2 clearly shows that adding simplified synthetic videos improved the performance of TSN. It also shows that the more synthetic videos are added the better the network performs.

The second part of the experiment is done on the entire dataset, even though some of the classes are not augmented with synthetic data. In this part, we compared the performance of Network-2 versus state-of-the-art methods, on both the HMDB-51 and UCF-101.

Table 7.2: The effect of combining real and simplified synthetic videos on sub-datasets.

<b>Dataset</b>	HMDB-38	UCF-25
Real only	71.8	96.66
Half Synthetic Videos+Real	73.66	97.5
All Synthetic Videos+Real	74.62	97.8

**HMDB-51 Dataset.** First, we compare our Network-2 to Cool-TSN [10]. Cool-TSN includes 39,982 appearance videos for 35 action classes, where 21 of them are common with HMDB-51. In our Network-2, we generate only 8000 videos, with actions similar to the 38 classes of HMDB-51. In the third stream, the synthetic videos were added to the real videos in the 38 classes, while the remaining 13 classes had only real videos. Results show that Network-2 outperformed Cool-TSN by 2.9 %. As can be seen in Table 7.3, the Network-2 also outperformed all of the state of the art systems, except I3D and OFF. However, we must note that I3D [46] had to be pre-trained on 300,000 videos from the Kinetics dataset. On the other hand, although OFF uses 5 streams while Network-2 uses 3 streams, it outperforms our proposed system by only 1.8 %.

**UCF-101 dataset.** Similar to HMDB-51, we created a sub-UCF-101 synthetic, including approximately 5000 videos, representing 25 classes of the actions. Using the same procedure as that performed on the HMDB-51, we added the 2500 videos first and then added 5000 videos to 25 classes of UCF-101. Since we reconstructed only 25% (25 out of 101) of the UCF-101 classes compared to 75% ( 38 of 51) of the HMDB-51, here the improvement was lower. Network-2 improved 0.6% above the vanilla TSN when 5000 videos were added to get a final score of 94.6 %.

### 7.3.3 The Effect of Decreasing the Number of Real Videos

In this section, we investigate the ability of simplified synthetic videos to replace real videos. Three tests were performed: (1) reducing the amount of videos by half, (2) keeping only 10 % of the real videos, and (3) having no real videos at all in the training sets.

In the first experiment, half of the real videos were randomly removed from the training splits, and the performance of the network was evaluated with and without synthetic data. In the second experiment, only 10 % of the real data was kept and the same evaluation was repeated. Finally, in the third experiment,



Table 7.3: Benchmarking Network-2 versus state-of-the-art networks

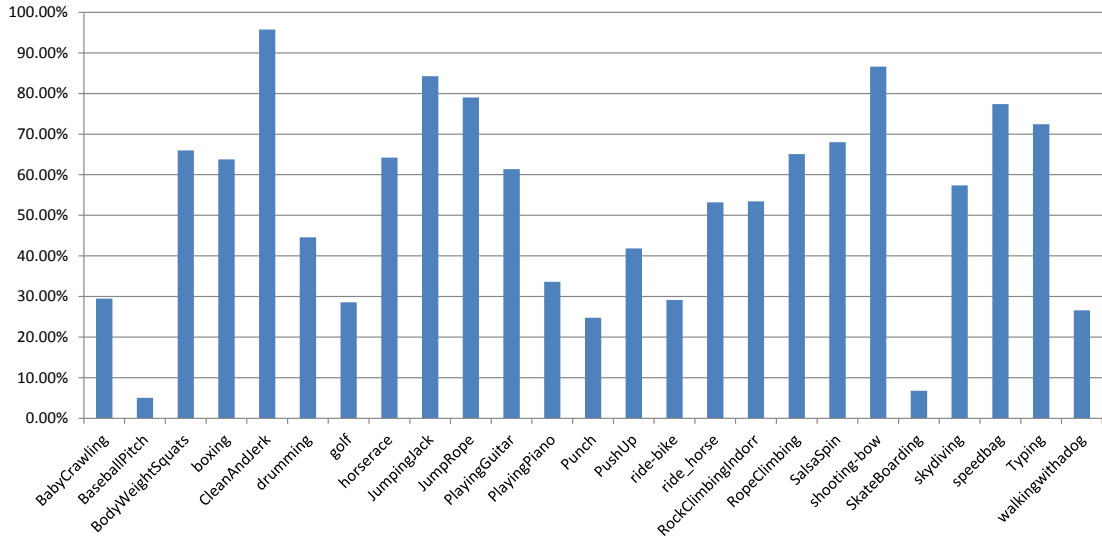
<b>Method</b>	UCF-101 (%)	HMDB-51 (%)
iDT+FV [47]	84.8	57.2
Tow-stream [29]	88.0	59.4
Two-stream LSTM [48]	88.6	-
L <sup>2</sup> STM [49]	93.6	66.2
TSN-2M [11]	94.0	68.5
TSN-3M [11]	94.2	69.5
Cool-TSN [10]	94.2	69.5
3D-ResNet-101 [2]	94.5	70.2
Two-stream MiCT [50]	94.7	70.5
CoViar [51]	94.9	70.2
OFF [52]	96.0	74.2
Two-stream I3D [46]	98.0	80.7
Network-2 half of the generated videos (Ours)	94.4	71.8
Network-2 all of the generated videos(Ours)	94.6	72.4

training was performed on only synthetic data, and testing on real data. The first two experiments were done using Network-2 while the last experiment was done using Network-3. Network-3 in Fig. 7.6 includes only the optical flow stream. Since there were some actions from the real datasets that could not be synthesized, during testing we limited the actions from UCF-101 to those that were produced synthetically. For UCF-101, 25 out of 101 classes were used. Table 7.4 shows that the accuracy of the network did not drop by much when half of the real videos were removed. However, when only 10% of the videos were kept, the accuracy dropped to 77.14 % with real videos, while it stayed relatively high 85.41% with the additional 5000 synthetic videos.

Table 7.4: The effect of decreasing the number of real videos on the accuracy of UCF-25 while adding 2500 and 5000 synthetic videos

<b>UCF-25</b>	Real	Real + 2500	Real + 5000
100% Real	96.6	97.5	97.8
50%Real	96.3	97.0	97.7
10%Real	77.4	81.6	85.4
0% Real	-	30.7	52.7

Finally, when training with 5000 synthetic videos, the accuracy dropped to 52.7% while training on around 2500 synthetic videos scored 30.71%. These results are comparable to those by [37], who got 52.1% when trained only on their generated synthetic data. The results we put forward demonstrate the potential of training action recognition networks using simplified videos, especially that some of the trained classes, such as ‘CleanAndJerk’ achieved an accuracy above 90%, as shown in Figure 7.3.3. On the other hand, other classes (such as ‘BaseBallPitch’ and ‘Skateboarding’) scored as low as 5%, suggesting the inability of the simulator in accurately re-creating those classes.



Accuracy for individual classes using synthetic data alone

### 7.3.4 Robustness Test

This final experiment was performed to test for robustness. Here, a network was trained on one dataset, and then tested on a second, using only actions classes that are common between the two. More specifically, 5 common classes between UCF-101 and HMDB-51 were identified (shoot-bow, biking, horse-riding, push-up, and golf-swing). TSN was trained on the regular HMDB-51, but when tested, the common classes stated above are replaced by videos from UCF-101. This process is tested using a TSN and augmented TSN and the results shows that with TSN the average accuracy of these five classes dropped from 91.4% to 70.3%, while with augmented TSN it dropped to 80.5% yielding to 10 % improvement.

### 7.3.5 Reduce Overfitting

In this experiment, we investigate the potential of synthetic data in reducing over-fitting of 3D-ResNet-18 [2]. Because 3D-ResNets are very deep network, they require a large amount of data to successfully train. 3D-ResNet-18 got a validation accuracy of 16.2% when trained and tested directly on HMDB-51. 3D-ResNet-18 was subsequently pre-trained on Kinetic (300k videos) in order to improve its accuracy to 56.4% on HMDB-51.

In our experiment, no pre-training was done, but we added both optical flow and appearance synthetic videos to the real HMDB dataset (12,119 in total), and used it to train 3D-ResNet-18. Although the 2% improvement is not very large, it demonstrates that synthetic data can help reduce over-fitting in networks trained on real data.

# Chapter 8

## Conclusion

In this thesis, we explained the importance of video analysis using deep network. We introduced three important methods which are face detection and recognition as well as emotion recognition. Next, we proposed an innovative action recognition system that is based on 3DCNN+LSTM. However, this system scored below the state of the art action recognition systems. The most important part in this thesis was introduction of new a way of generating simplified synthetic data to improve the network accuracy. We analyzed the effectiveness of simplified synthetic data in the training of deep networks for the sake of action categorization. We validated that optical flow information was sufficient to train a network, and that appearance information could be disregarded. The caveat here is that the proposed actions do not require background interaction to differentiate between two different actions (, swimming vs flying). We also tested using synthesized data for training under two different scenarios: the first using synthesized data to augment TSN with an additional stream, and the second using only synthesized data to train the network from scratch. Both scenarios revealed good results, where in all cases we obtained notable improvements for TSN on both UCF-101 and HMDB-51. We are currently working on creating a large synthetic dataset that includes over 200 k videos in order to compare its effectiveness to that of a dataset of real actions of comparable size, such as Kinetics.

# Bibliography

- [1] L. Zhang, G. Zhu, P. Shen, J. Song, S. Afaq Shah, and M. Bennamoun, “Learning spatiotemporal features using 3dcnn and convolutional lstm for gesture recognition,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3120–3128, 2017.
- [2] K. Hara, H. Kataoka, and Y. Satoh, “Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 18–22, 2018.
- [3] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, *et al.*, “The kinetics human action video dataset,” *arXiv preprint arXiv:1705.06950*, 2017.
- [4] S. Jones and L. Shao, “Unsupervised spectral dual assignment clustering of human actions in context,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 604–611, 2014.
- [5] K. Soomro and M. Shah, “Unsupervised action discovery and localization in videos,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 696–705, 2017.
- [6] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, “Training deep networks with synthetic data: Bridging the reality gap by domain randomization,” *arXiv preprint arXiv:1804.06516*, 2018.
- [7] D. Dwibedi, I. Misra, and M. Hebert, “Cut, paste and learn: Surprisingly easy synthesis for instance detection,” in *The IEEE international conference on computer vision (ICCV)*, 2017.
- [8] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, “The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3234–3243, 2016.

- [9] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, “FlowNet: Learning optical flow with convolutional networks,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2758–2766, 2015.
- [10] C. R. De Souza, A. Gaidon, Y. Cabon, and A. M. L. Peña, “Procedural generation of videos to train deep action recognition networks,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2594–2604, 2017.
- [11] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, “Temporal segment networks: Towards good practices for deep action recognition,” in *European Conference on Computer Vision*, pp. 20–36, Springer, 2016.
- [12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, pp. 1097–1105, 2012.
- [13] S. Y. Fei-Fei Li, Justin Johnson, “Knuth: Computers and typesetting.”
- [14] M. Cavaioni, “Deep learning series: Convolutional neural networks.”
- [15] S. Tejani, “Artistic style transfer with deep neural networks.”
- [16] B. Holczer, “Computer vision - haar-features.”
- [17] P. J. Phillips, H. Wechsler, J. Huang, and P. J. Rauss, “The feret database and evaluation procedure for face-recognition algorithms,” *Image and vision computing*, vol. 16, no. 5, pp. 295–306, 1998.
- [18] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.
- [19] S. Yang, P. Luo, C.-C. Loy, and X. Tang, “Wider face: A face detection benchmark,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5525–5533, 2016.
- [20] O. M. Parkhi, A. Vedaldi, A. Zisserman, *et al.*, “Deep face recognition,” in *bmvc*, vol. 1, p. 6, 2015.
- [21] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

- [22] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, “Labeled faces in the wild: A database for studying face recognition in unconstrained environments,” in *Workshop on faces in ‘Real-Life’ Images: detection, alignment, and recognition*, 2008.
- [23] O. Arriaga, M. Valdenegro-Toro, and P. Plöger, “Real-time convolutional neural networks for emotion and gender classification,” *arXiv preprint arXiv:1710.07557*, 2017.
- [24] I. J. Goodfellow, D. Erhan, P. L. Carrier, A. Courville, M. Mirza, B. Hamner, W. Cukierski, Y. Tang, D. Thaler, D.-H. Lee, *et al.*, “Challenges in representation learning: A report on three machine learning contests,” in *International Conference on Neural Information Processing*, pp. 117–124, Springer, 2013.
- [25] A. Cherian, B. Fernando, M. Harandi, and S. Gould, “Generalized rank pooling for activity recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3222–3231, 2017.
- [26] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, “Long-term recurrent convolutional networks for visual recognition and description,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2625–2634, 2015.
- [27] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning spatiotemporal features with 3d convolutional networks,” in *Proceedings of the IEEE international conference on computer vision*, pp. 4489–4497, 2015.
- [28] B. Singh, T. K. Marks, M. Jones, O. Tuzel, and M. Shao, “A multi-stream bi-directional recurrent neural network for fine-grained action detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1961–1970, 2016.
- [29] K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” in *Advances in neural information processing systems*, pp. 568–576, 2014.
- [30] X. Peng, B. Sun, K. Ali, and K. Saenko, “Learning deep object detectors from 3d models,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1278–1286, 2015.
- [31] J. Wu, I. Yildirim, J. J. Lim, B. Freeman, and J. Tenenbaum, “Galileo: Perceiving physical object properties by integrating a physics engine with deep learning,” in *Advances in neural information processing systems*, pp. 127–135, 2015.

- [32] S. Sankaranarayanan, Y. Balaji, A. Jain, S. N. Lim, and R. Chellappa, “Learning from synthetic data: Addressing domain shift for semantic segmentation,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [33] A. Handa, V. Patraucean, V. Badrinarayanan, S. Stent, and R. Cipolla, “Understanding real world indoor scenes with synthetic data,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4077–4085, 2016.
- [34] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, “A naturalistic open source movie for optical flow evaluation,” in *European Conference on Computer Vision*, pp. 611–625, 2012.
- [35] H. Rahmani and A. Mian, “3d action recognition from novel viewpoints,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1506–1515, 2016.
- [36] H. Rahmani and A. Mian, “Learning a non-linear knowledge transfer model for cross-view action recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2458–2466, 2015.
- [37] C. Vondrick, H. Pirsiavash, and A. Torralba, “Generating videos with scene dynamics,” in *Advances In Neural Information Processing Systems*, pp. 613–621, 2016.
- [38] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, “Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2686–2694, 2015.
- [39] G. Rogez and C. Schmid, “Mocap-guided data augmentation for 3d pose estimation in the wild,” in *Advances in Neural Information Processing Systems*, pp. 3108–3116, 2016.
- [40] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pp. 23–30, IEEE, 2017.
- [41] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with lstm,” 1999.
- [42] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, “Convolutional lstm network: A machine learning approach for precipitation nowcasting,” in *Advances in neural information processing systems*, pp. 802–810, 2015.



- [43] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black, “Towards understanding action recognition,” in *Proceedings of the IEEE international conference on computer vision*, pp. 3192–3199, 2013.
- [44] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255, Ieee, 2009.
- [45] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167*, 2015.
- [46] J. Carreira and A. Zisserman, “Quo vadis, action recognition? a new model and the kinetics dataset,” in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pp. 4724–4733, IEEE, 2017.
- [47] H. Wang and C. Schmid, “Action recognition with improved trajectories,” in *Proceedings of the IEEE international conference on computer vision*, pp. 3551–3558, 2013.
- [48] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, “Beyond short snippets: Deep networks for video classification,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4694–4702, 2015.
- [49] L. Sun, K. Jia, K. Chen, D.-Y. Yeung, B. E. Shi, and S. Savarese, “Lattice long short-term memory for human action recognition.,” in *ICCV*, pp. 2166–2175, 2017.
- [50] Y. Zhou, X. Sun, Z.-J. Zha, and W. Zeng, “Mict: Mixed 3d/2d convolutional tube for human action recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 449–458, 2018.
- [51] C.-Y. Wu, M. Zaheer, H. Hu, R. Manmatha, A. J. Smola, and P. Krähenbühl, “Compressed video action recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6026–6035, 2018.
- [52] S. Sun, Z. Kuang, L. Sheng, W. Ouyang, and W. Zhang, “Optical flow guided feature: a fast and robust motion representation for video action recognition,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

