UNIVERSITY OF CALIFORNIA, SAN DIEGO

Action Recognition from Videos using Deep Neural Networks

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

in

Computer Science

by

Rishikesh Sanjay Ghewari

Committee in charge:

Professor Garrison W. Cottrell, Chair
Professor Kamalika Chaudhuri
Professor Julian McAuley

2017

The thesis of Rishikesh Sanjay Ghewari is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

Chair

University of California, San Diego

2017

DEDICATION

I dedicate this thesis to my mother, father, and sister. Thank you for supporting me, guiding me through all my decisions, and for giving me the opportunity to pursue my Masters. This thesis or Masters would not have been possible without you.

I would also like to dedicate this to my friends for helping me stay focused.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank my advisor and committee chair Professor Garrison W. Cottrell for the valuable guidance he has provided over the course of this project and for the opportunity to be a part of GURU. I would like to thank members of GURU for their advice and help throughout the project.

I would like to thank my committee members, Professor Julian McAuley and Professor Kamalika Chaudhuri, for taking time out of their busy schedules to read this work and for being on my thesis committee.

ABSTRACT OF THE THESIS

Action Recognition from Videos using Deep Neural Networks

by

Rishikesh Sanjay Ghewari

Master of Science in Computer Science

University of California, San Diego, 2017

Professor Garrison W. Cottrell, Chair

Convolutional neural network(CNN) models have been extensively used in recent years to solve the problem of image understanding giving state-of-the-art results in tasks like classification, recognition, retrieval, segmentation and object detection. Motivated by this success there have been several attempts to extend convolutional neural networks for video understanding and classification. An important distinction between images and videos is the temporal information that is encoded by the sequence of frames. Most CNN models fail to capture this temporal information. Recurrent neural networks have shown promising results in modelling sequences.

In this work we present a neural network model which combines convolutional neural networks and recurrent neural networks. We first evaluate the effect of the convolutional network used for understanding static frames on action recognition. Following

this we explore properties that are inherent in the dataset. We combine the representation we get from the convolutional network, the temporal information we get from the sequence of video frames and other properties of the dataset to create a unified model which is trained on the UCF-101 dataset for action recognition. We evaluate our model on the pre-defined test set splits of the UCF-101 dataset. We show that our model is able to achieve an improvement over the baseline model. We show comparison between our models and various models proposed in other related works on the UCF-101 dataset. We observe that a good model for action recognition not only needs to understand static frames but also needs to encode the temporal information across a sequence of frames.

Chapter 1

Introduction

1.1 The Problem

In a world where images and videos are an integral part of our daily life, the quest to understanding them has lead to the development of various algorithms. These algorithms are focused on capturing the semantic content from images and videos for a wide range of applications like search and summarization. In the recent years, Convolutional Neural Networks(CNNs) [11] have proven to be highly successful models for image content understanding, which is demonstrated by their state-of-the-art performance on image recognition, segmentation, and object detection tasks. The success of CNNs can be attributed to the ability of learning complex features using trainable filters and feature pooling operations. An important enabling factor for the dramatic change in performance was the advancement in the computing power of GPUs, which led to the increase of network sizes to millions of parameters that can be learned from massive datasets.

Encouraged by the great success of CNNs, there have been several attempts to extend CNNs to video classification and action recognition. Videos contain additional information apart from that contained in a single static frame of the video. They additionally have a temporal component which provides important motion information. There are some actions that can be recognized by just looking at a single frame. For other actions, single frames can be ambiguous and the cues provided by the temporal component prove to be a clarifying factor. For example, still images for handstand

walking and handstand pushups, or breast-stroke and a crawl can be very similar to each other. In cases like these the motion cues help in distinguishing these actions from one another.

The primary challenge in modelling videos is to model variable number of frames using a fixed number of parameters. There are multiple methods proposed to tackle this problem which include feature pooling or recurrent neural networks. In this work we explore models using recurrent neural networks, primarily Long-short term memory(LSTM) models.

In this work we use the UCF-101 [15] dataset which is an action recognition dataset of realistic action videos. Action classification in videos has additional challenges such as camera motion, variation in motion and view points as compared to image classification. ImageNet has 1000 images per class as compared to 100 examples per class in UCF-101. UCF-101 has 101 action classes with roughly 130 videos per class. The low number of videos per class is a significant problem, as training complex temporal models like LSTMs with millions of parameters requires a great deal of data. This is a part of the reason for the low success for the action recognition task.

Moreover, we also exploit the fact that the 101 action classes for UCF-101 can be divided into 5 types. We can use this super-class information to get a better understanding of the semantic content. We explore the idea of learning motion features directly from a sequence of images without an input signal of precomputed optical flow. This project tries to take a step towards learning good video features in an unsupervised manner by learning to predict the next frame. The features obtained should be task independent and should be able to capture the contextual information in the video.

1.2 Layout of Thesis

The remainder of the thesis is organized as follows: Chapter 2 describes the general framework of convolutional neural networks and recurrent neural networks. It also covers details of specific CNN models that are used in this thesis. It further discusses relevant work related to action recognition that has been done over the past few years. Chapter 3 provides details about the UCF-101 dataset that is used in this thesis

and details preprocessing steps. Chapter 4 describes in detail the models that were in this thesis. It also provides details about the experimental settings and results of the experiments. Finally, chapter 5 gives a conclusion based on the results and discusses other possible methods that can be explored in future works.

Chapter 2

Background

2.1 Convolutional Neural Network

A convolutional neural network(CNN) is a feed forward neural network. CNNs are primarily designed to solve problems related to image recognition. These models, unlike ordinary feed forward neural networks, use convolutions in place of general matrix multiplication. They are biologically inspired variants of ordinary neural networks. CNN architectures are specialized for learning data that has a grid-like topology [4]. Primarily, CNNs are assume that the input data are images. This allows them to encode some properties in the network structure. A convolutional layer is the basic building block of a CNN. The parameters of a convolutional layer consist of a set of small learnable filters. When learning we convolve each filter across the image and take a dot product. The network learns filters that activate when they "see" some visual feature, for example, vertical edges, or wheel-like patterns. Convolutions build on the observation that in images, global features are constructed from combinations of local features in a hierarchical manner. We discuss in details the connectivity between units and their parameter sharing scheme.

Local Connectivity: When we have high dimensional images, having fully connected layers is not practical, as it increases the number of parameters by a huge amount. Instead local connectivity helps in learning filters which capture important image features without having to learn weights corresponding to global patterns.

Parameter Sharing: Parameter sharing refers to using the same weights across

more than one location. It is a scheme used to reduce the number of free parameters associated with the model. It is based on an assumption that if a filter is a useful feature at the position (x_1, y_1) then it is also useful at the position (x_2, y_2). This reduces the number of parameters of a convolutional layer dramatically. This contributes to the translational invariance of convolutional neural networks.

Another important concept used in CNNs is pooling. A pooling layer is often inserted between two convolutional layers. The primary function of the pooling layer is to provide translational invariance. It consequently reduces the spatial size of the representation of the image. Doing this also reduces the network parameters thus limiting the effect of overfitting. The most common operation used for pooling is the MAX operation. A very common form of the pooling layer is a pooling layer with a filter of size 2×2 applied with a stride of 2. In this case, the max is performed over 4 numbers and it discards $\frac{3}{4}$ of the activations. In addition to MAX, the pooling layer can also use other functions like average pooling, L2-norm pooling, or global average pooling.

A CNN architecture has multiple alternating convolutional and pooling layers. These layers extract meaningful features that are most important to describe the input image. After several such layers, we get a high level representation of the image. This representation is then used with a fully connected layer, which is the classifier, along with a softmax layer, which generates the image labels, for image classification.

Training a CNN involves training millions of parameters. This requires both a large amount of data and time. However, CNNs still learn much faster than an ordinary feed-forward neural network. It also performs much better as compared to a standard feed-forward neural network with similar number of parameters.

We further discuss the details of the two network architecture used in this thesis.

2.1.1 AlexNet

AlexNet [10] is the first work that popularized CNNs in the computer vision world. This network architecture won the ILSVRC-2012 contest by very large margin, achieving a top-5 test error rate of 15.3% compared to 26.2% achieved by the runner-up. The architecture is relatively simple.

AlexNet consists of 5 convolutional layers, 3 max-pooling layers and 3 fully

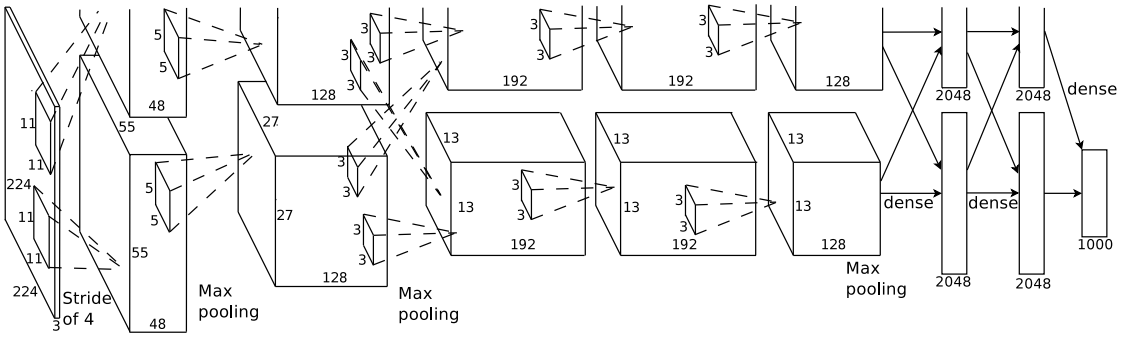

Figure 2.1: AlexNet, the CNN architecture which is trained on the ImageNet dataset to classify natural images and won ILSVRC-2012 [10]

connected layers. The network takes an image of size $227 \times 227 \times 3$ as its input. The input is connected to the first convolutional layer. It has 96 kernels of size 11×11 with a stride of 4. The output of this layer is passed through a normalization and a max pooling layer, giving it to the next convolutional layer. The second convolutional layer has 256 kernels with a size of 5×5, a stride of 1 and a padding of 2 . Again the output is passed through a normalization and a max pooling layer and is given to the third convolutional layer. The third convolutional layer has 384 kernels of size 3×3 with a stride of 1 and a padding of 1. The output of this layer is the input of the fourth convolutional layer. It has 384 kernels of size 3×3, stride of 1 and padding of 1. It is connected to the fifth convolutional layer which has 256 3×3 kernels with a stride and padding of 1. This is followed by a pooling layer. The output of the pooling layer is connected to 2 fully connected layers with 4096 units each. The output of the second fully connected layer is finally connected to a 1000-unit softmax layer which generates the class labels. Figure 2.1 shows that the network has two parallel channels. These channels were trained separately on 2 GPUs and are only cross-connected at two places. The kernels of the second, fourth and fifth convolutional layer are connected only to the kernel maps which reside on the same GPU. The kernels on the third convolutional layer are connected to all kernel maps in the second layer. AlexNet uses the Rectified Linear Units(ReLu) as a nonlinearity over its activation function.

We use this model as our starting model to evaluate the performance of a convolutional network used to get representations of frames on the video classification task.

2.1.2 ResNet

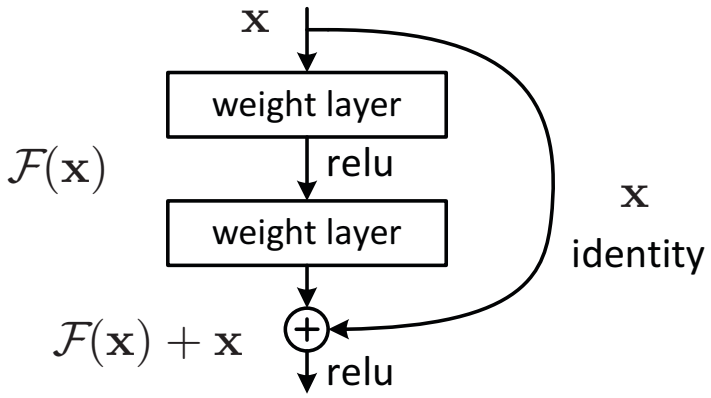

Figure 2.2: Residual block from the ResNet architecture [6]

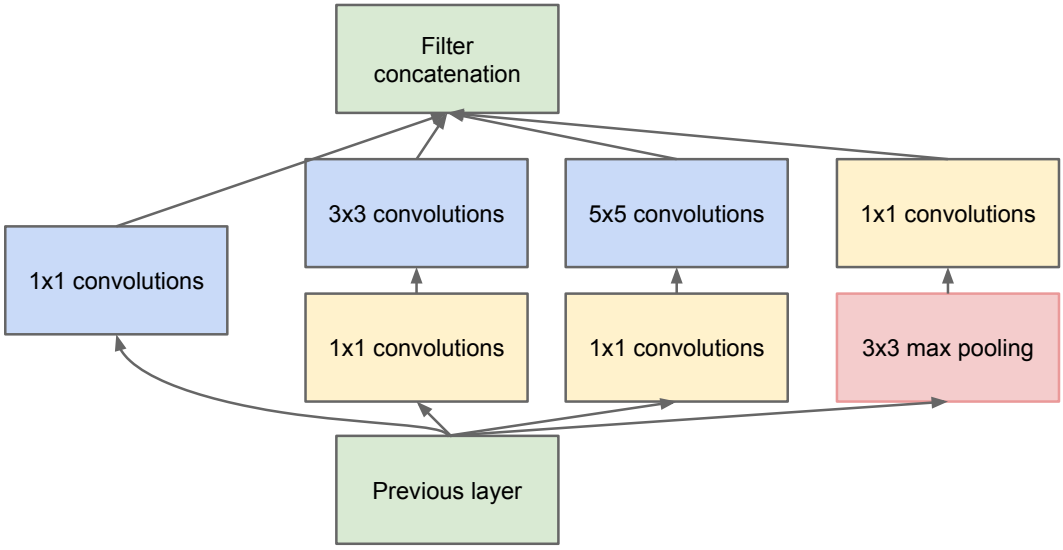

Figure 2.3: Inception module from the GoogLeNet architecture [17]

After AlexNet popularized convolutional neural networks for image classification, there have been several modifications and improvements to the network architecture. New convolutional architectures have provided improvements to the ImageNet image classification task every year since 2012. Various models have explored different aspects of building neural network architectures. One common theme among them being increase in the depth of the models. After AlexNet won ILSVRC-2012, VGGNet [14]

was a 16 layered deep convolutional neural network. It showed that the depth of the network is a critical factor in performance. VGGNet was the runner up of ILSVRC-2014. That year the competition was won by GoogLeNet [17], a 22 layer model. The main contribution of the GoogLeNet model was the Inception Module that reduced the number of parameters of the model drastically. The inception module consists of 1×1, 3×3, and 5×5 convolutions. Additionally, it contains a parallel pooling layer. Figure 2.3 shows the inception module used in GoogLeNet. The GoogLeNet model has 9 inception modules. It replaced the fully connected layer, which contributes the most parameters, with a global average pooling layer. GoogLeNet has only 5 million parameters as compared to AlexNet's 60 million.

Microsoft Research Asia's ResNet model was the winner of ILSVRC-2015. An important observation made in the ResNet paper by He et al. [6] is that simply stacking additional convolutional layers does not give a better model. They compared a 56 layer convolutional network with a 20 layer network. The experiment showed that the deeper 56 layered network had a higher training and test error on the CIFAR-10 dataset. They also observed a similar phenomenon with the ImageNet dataset.

The primary contribution of the ResNet model is the residual block. The residual block addresses the problem of vanishing gradients while training very deep models by introducing skip connections with identity mapping. These are skip connections with fixed weight of 1.0. The skip connections allow the model to grow very deep and still be trainable. Figure 2.2 shows the basic residual block [6]. Consider this network without the skip connections. We expect it to learn some underlying mapping $H(x)$ by learning the 2 weight layers. The residual block instead tries to fit to another mapping $F(x)$ such that,

$$H(x) = F(x) + x$$

Here $F(x)$ is a residual mapping with respect to identity. If the optimal mapping is identity then it is easy to learn the weights as 0. Otherwise, if the optimal mapping is close to identity, it is easier to find small fluctuations in addition to the identity mapping.

The main idea that went into building the ResNet-152 model was to stack a variant of these residual blocks to form a very deep network. Figure 2.4 shows the basic building block which is a slight variant of the basic residual block of the ResNet [6]

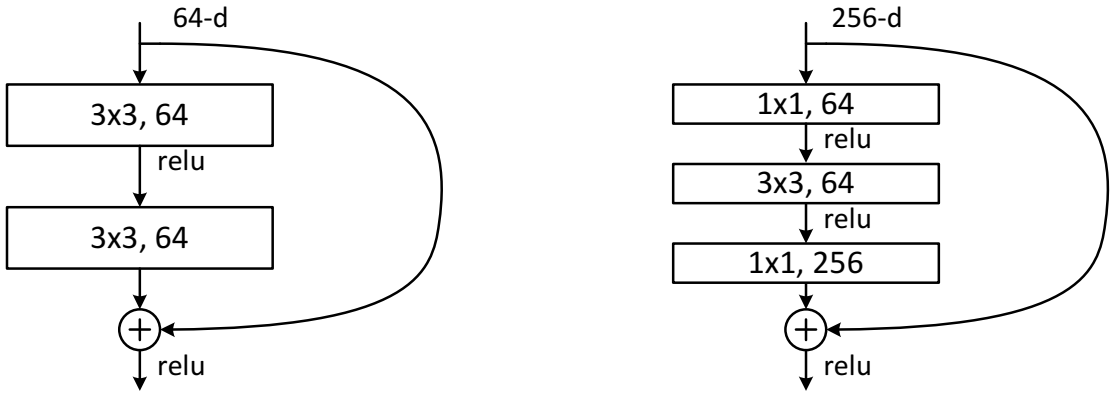

Figure 2.4: Building block of the ResNet architecture [6]: left - basic block used in an experimental 34 layer model, right - bottleneck structure used in the deeper 50,101,152 layered models

architecture. The input to the ResNet model is a $224 \times 224 \times 3$ image. Each residual block contains 3 convolutional layers. The first convolutional layer is 1×1 and is used to reduce the dimension. This is followed by a 3×3 convolutional layer with the same output dimension. Finally, this is followed by a 1×1 convolutional layer to increase the dimension back to the original input dimension.

The best model for ImageNet was a 152 layer ResNet model using the module in Figure 2.4(right). ResNet-152 achieves an error rate of just 3.57% on the ImageNet test set. For this thesis we use the ResNet-50 model as it performs very well and is not too large for our GPU computations.

2.2 Recurrent Neural Networks

2.2.1 Vanilla Recurrent Neural Networks

Recurrent neural networks(RNNs) are powerful models which have been used to generate and model sequences in various domains like text, image captions and music. RNNs can be used to process sequential data one step at a time. RNNs can also be used to generate new sequences by iteratively sampling from the network's output distribution and using the output as the input for the next step.

A fundamental limitation of feed forward neural networks is their inability to process variable length inputs. Feed-forward neural networks, both "vanilla" neural

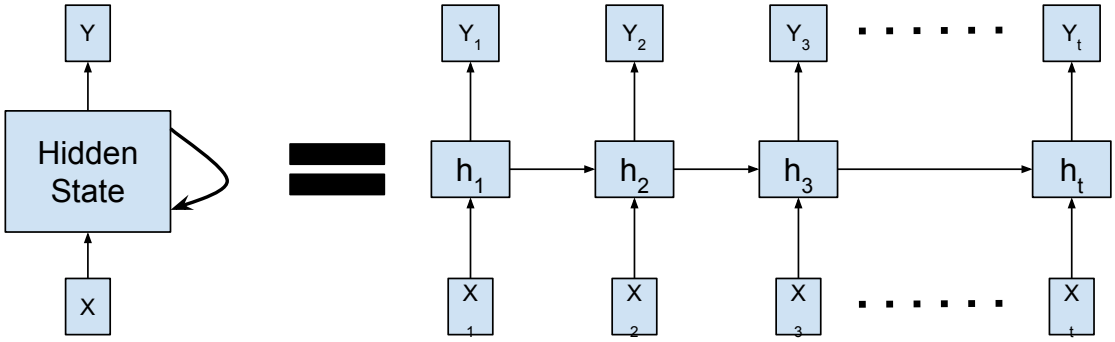

Figure 2.5: An unrolled recurrent neural network

networks and convolutional neural networks can accept a fixed size input vector and produce a fixed size output vector. Unlike these, recurrent neural networks are models which allow operations over a sequence of vectors.

Recurrent neural networks are consisted of recurrent units like one show in figure 2.5 (left). We allow the network to have cyclic connections. The recurrent units have connections that form a directed cycle. They contain a hidden state. To model temporal dynamics, RNNs map the input sequence to a hidden states and hidden states to the output. The mappings are governed by the following equation:

$$h_t = f(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$

$$y_t = f(W_{hy}h_t + b_y)$$

where x_t is the input at time t, f is a nonlinear function such as a sigmoid, rectified linear unit, or a hyperbolic tangent, h_t is the hidden state at time t, h_{t-1} is the hidden state at time $t-1$, and y_t is the output. W_{xh} is the weight matrix from the input to the hidden state, W_{hh} is the weight matrix from hidden to hidden, and b_y and b_h are bias units.

Though RNNs are very rich and dynamic it is difficult to train them to model long term dependencies. This is in part due to the vanishing and exploding gradient problem. These problems are faced due to propagation of the gradient through many layers of the unfolded recurrent network.

A very effective solution to the vanishing gradient problem is provided by LSTMs.

2.2.2 LSTM

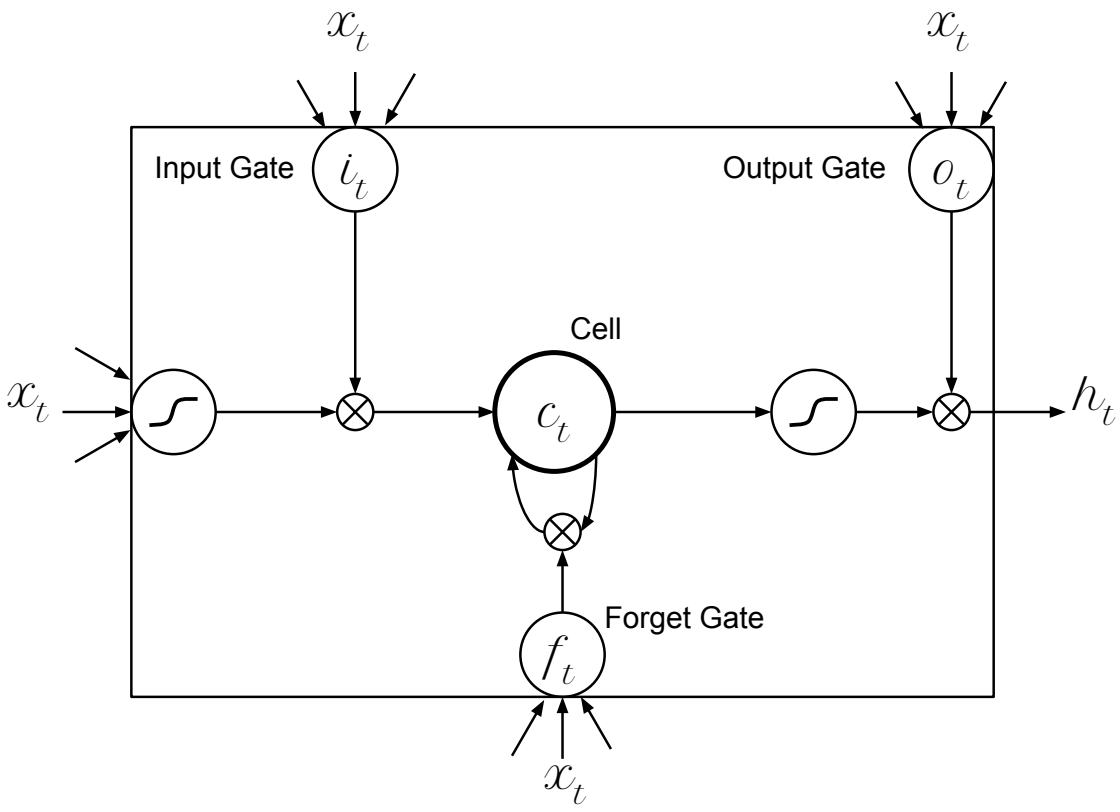

Figure 2.6: Long short term memory Cell [5]

Long Short Term Memory(LSTM) networks are a modified version of the RNN architecture. They were introduced by Hochreiter & Schmidhuber [7] in 1997. Many people have since refined and popularized the LSTM architecture. LSTMs are capable of learning long term dependencies. Figure 2.6 shows a basic LSTM unit. LSTMs are governed by the following set of equations:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{2.1}$$

$$h_t = o_t \odot tanh(c_t) \tag{2.2}$$

Here $\sigma(x) = (1+e^x)^{-1}$ is the sigmoid non-linearity. $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ is the hyperbolic tangent non-linearity. i, f, and o are the input gate, forget gate, and the output gate respectively. c is the memory cell. h is the hidden unit vector. x is the input. $x \odot y$ denotes the elementwise product of vectors x and y. The weight matrices are denoted by appropriate subscripts. For example,W_{xf} is the input to forget gate matrix, W_{ho} is the hidden to output gate weight matrix and so on for the other weight matrices.

The LSTM modifies the cell state using various operations, namely, the forget and input gates and the previous hidden state. It modifies the hidden state using information from the cell and output gate. The forget gate governs how much information is retained from the previous time step. It looks at the previous hidden state and the current input and outputs a number between 0 and 1. This number determines the amount of information that is retained from the previous cell state. The LSTM then has to decide what new information has to be added to the cell. It does so using the input gate and a candidate state generated using the input and hidden state of the LSTM units at the previous time step. The input gate, like the forget gate, modulates how much of the new candidate state percolates into the cell. Finally, using the forget gate, input gate, and the new candidate state the cell is updated according to equation 2.1. Next, the LSTM needs to decide what to output. The output of the LSTM is the updated hidden state. The update to the hidden state is controlled by two factors: the current cell state that we updated as stated above and an output gate. The current cell state is passed through a hyperbolic tangent function to get a value between -1 and 1. The output gate, like the forget and input gates, determines how much the cell state will affect the new hidden state and has a multiplicative interaction. The update to the hidden state is governed by equation 2.2.

The RNN has a state which we update at every time step. LSTMs have a cell state which is modified by an additive update to the previous state. The self-link on the cell state c (see Figure 2.6) has a weight of 1 and is gated by the forget gate f. The activation function of the cell state is linear. This allows the model to learn long term dependencies using backpropagation, because the gradient can flow through the weight of 1 all the way back to the input. This mitigates the vanishing gradient problem faced by Vanilla RNNs.

2.3 Related Work

Over the past few years there has been a lot of work in the field of action recognition from videos. There has been a great increase in accuracy because of learned features and several other deep learning methods and architectures. There have been several attempts to solve the action recognition problem using just CNNs and a combination of CNNs and RNNs or LSTMs.

In this section we discuss a few of these approaches that are relevant to this work. The remainder of this section will give a brief description of the models used to solve the action recognition problem.

2.3.1 CNN based video classification

Karpathy et al. [9] use CNNs to do video classification. They take advantage of the local spatio-temporal information by exploring various fusion techniques. The paper proposes 4 different types of fusion, namely, Single-frame, Early fusion, Late fusion, Slow fusion. Figure 2.7 shows a summary of the models used. Apart from this the paper also implements a two stream convolutional architecture. There is a context stream that focuses on the context information by taking a low resolution image as its input. The fovea stream takes high resolution center crop images as the input. This takes advantage of the fact that the object of interest is often at the center of the image. The two-stream network is mainly used to speed up the network as it reduces the input dimensionality in half. The paper also introduces the Sports-1M dataset which is a dataset with 1 million videos and 487 different sports classes. The model is trained on this Sports-1M dataset. It achieves an accuracy of 63.9% (top-1 prediction) and 82.4% (top-5 predictions) on that dataset.

2.3.2 Combination of CNN and RNN

Donahue et al. [1] proposed the long-term recurrent convolutional networks (LRCN) architecture. The paper proposes a general architecture to learn video representation that can be used in various tasks like action recognition, image captioning, video description. The LRCN model is a combination of convolutional neural networks

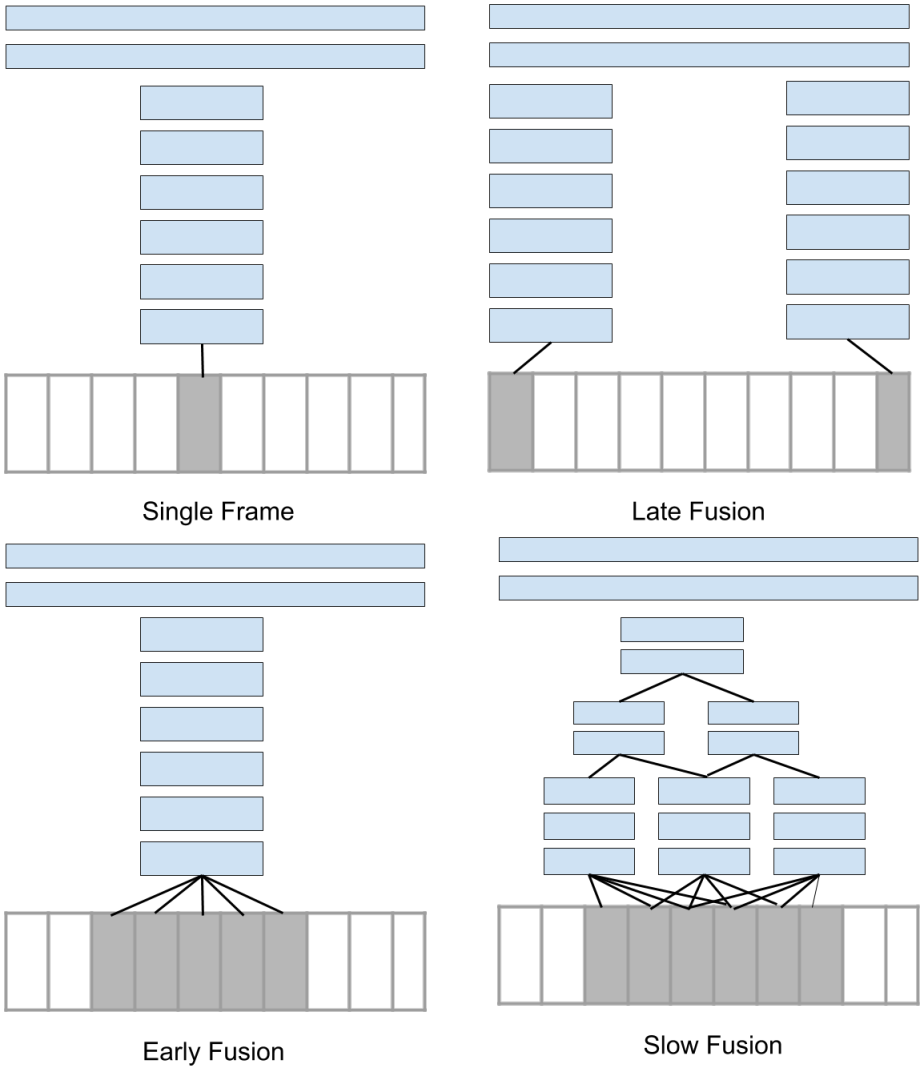

Figure 2.7: Various fusion methods used in the Karpathy [9] paper

and recurrent neural networks. Having an RNN allows the network to accept a sequence as its input. This is ideal for videos as video is a sequence of frames. The CNN produces a good representation of a static frame that is combined with long term temporal information carried by a sequence of frames. Figure 2.8 shows the general architecture of the LRCN model. For action recognition the paper uses a pre-trained hybrid CaffeNet (which is a minor variant of the AlexNet model) for the CNN part of the model. The best model uses the output of the fc_6 layer of the CaffeNet followed by a 256 hidden unit LSTM. Optical flow gives very important motion information about videos. The paper also combines optical flow along with just using RGB images. When using flow

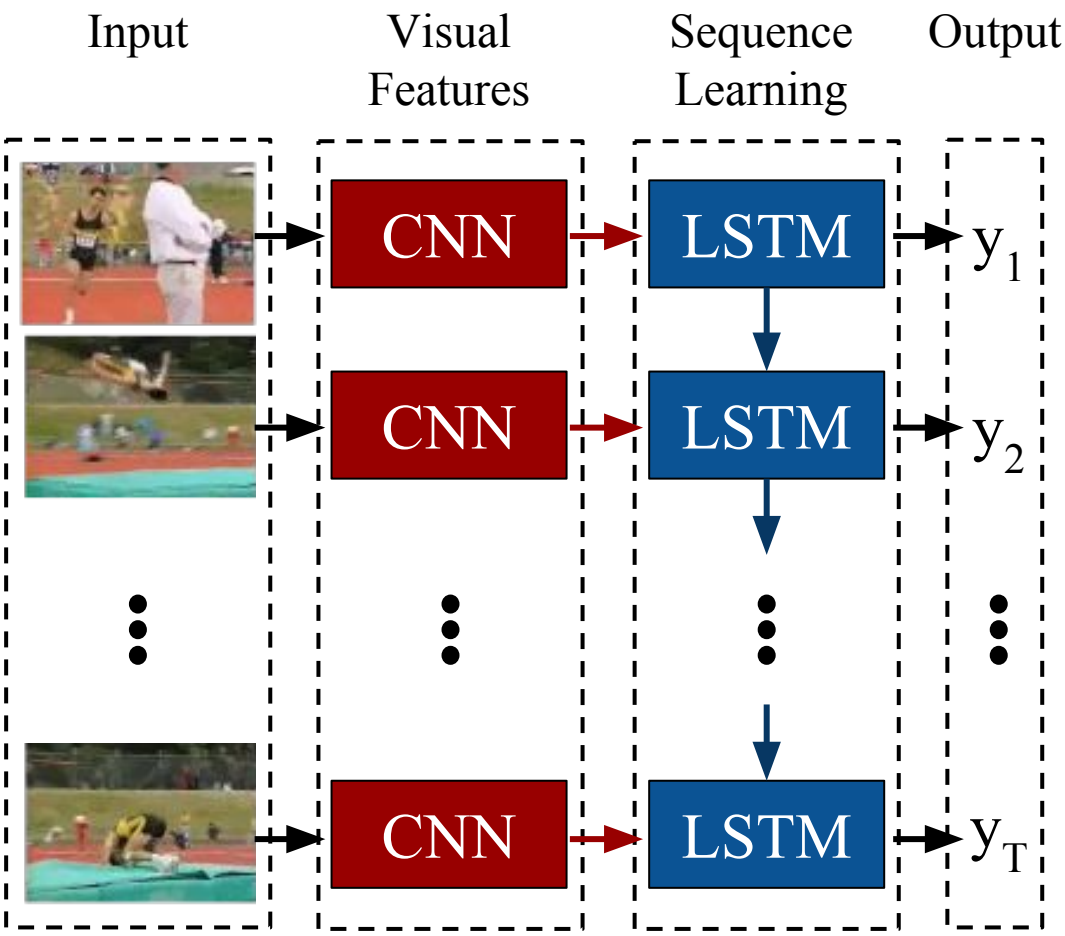

Figure 2.8: The LRCN architecture [1]

as an input, the network again uses the fc_6 layer but followed by 1024 hidden LSTM units. A weighted combination of these models is then used for evaluation. This results in a performance accuracy of 77.28% on the UCF-101 test set.

Ng et. al [18] use a model similar to [1]. In addition to using a combination of CNNs and LSTMs, the paper also tries just pooling over the output of a CNN that is trained from scratch on the Sports-1M dataset. The paper explores various pooling methods including some from [9], namely, Late Pooling, Slow Pooling, Conv Pooling, Local Pooling and Time-Domain Pooling. Instead of having separate networks using raw frames and flow information, this model combines them by concatenating the two together. Their best model achieves an accuracy of 73.1% on the Sports-1M dataset and an accuracy of 88.6% for a network fine-tuned on the UCF-101 dataset. Figure 2.9 gives

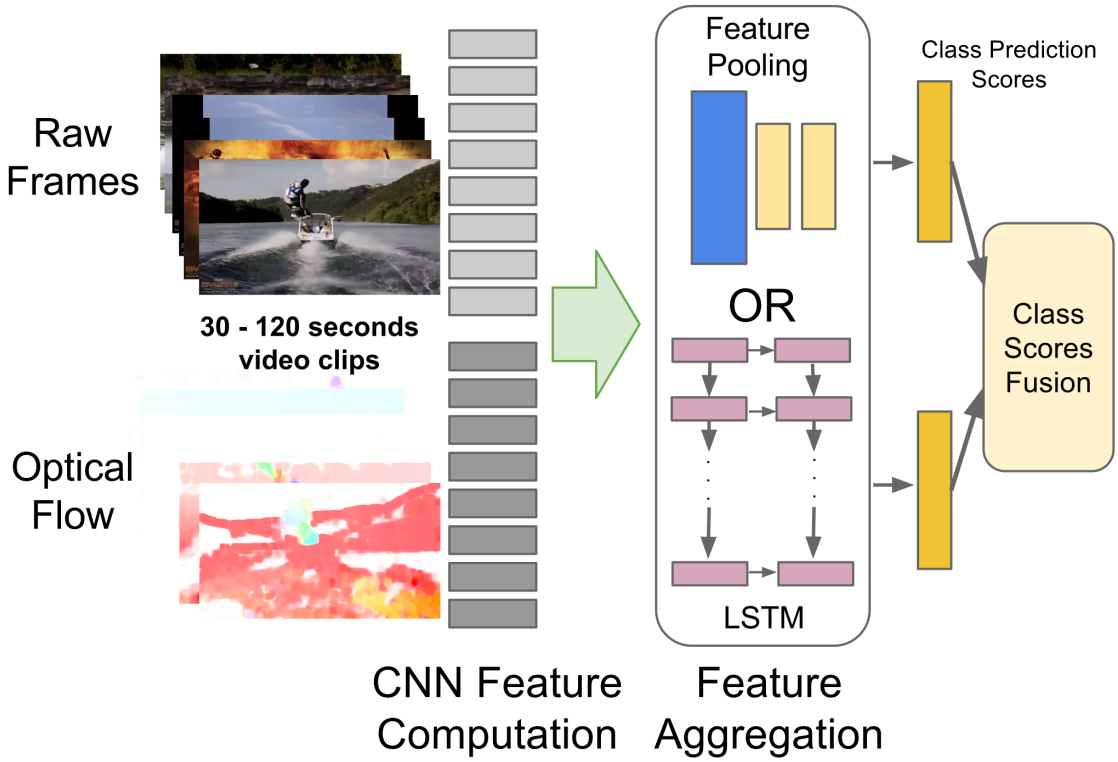

Figure 2.9: A summary of methods used in Beyond Short Snippets: Deep Networks for Video Classification [18]

a summary of the methods tried in the paper.

2.3.3 Using optical flow

Ng et. al [12] build upon the idea that optical flow is a very important feature for video classification. The state-of-the-art systems still uses hand-crafted motion features like optical flow when doing action recognition. In their previous paper, mentioned above, the network uses optical flow as an input for action recognition. Here, they learn the motion representation by learning to predict the optical flow between T input frames. Figure 2.10 shows the architecture for the ActionFlowNet. They use a 3D-ResNet model for multi-frame optical flow estimation. They also incorporate a future prediction module to predict the optical flow on the last frame, which is the optical flow between the T^{th} and $(T+1)^{st}$ frames. This future prediction module is based on the idea that a model needs to have semantic reasoning to extrapolate the future optical

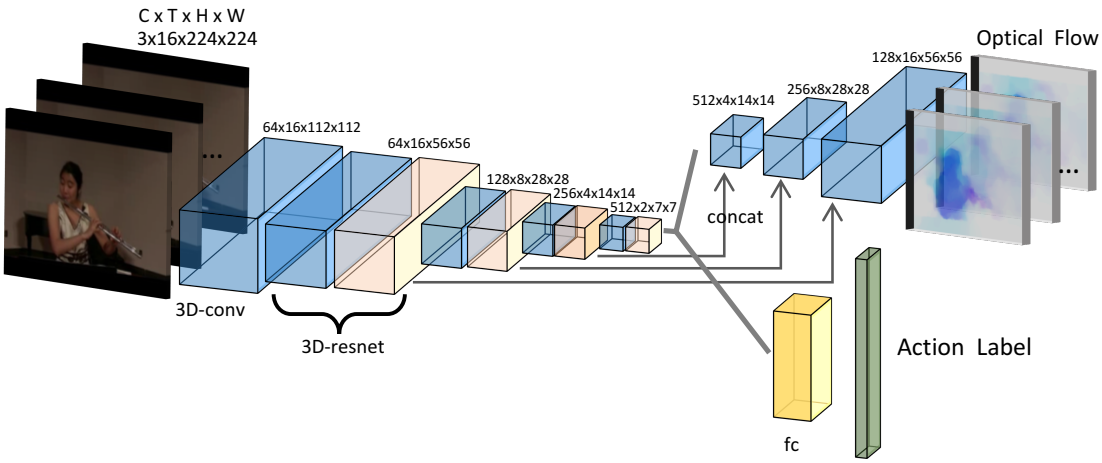

Figure 2.10: ActionFlowNet for jointly estimating optical flow and recognizing actions [12]

flow. This trains the model to learn better motion features. The network uses an L_2 loss function between the estimated flow and the ground truth flow over all pixels. The model uses a 3D ResNet as the encoder and has deconvolutional layers that are extended in a similar fashion. The network then jointly estimates the optical flow while doing action recognition. The network is directly trained on UCF-101, unlike the previous paper where it was trained on the Sports-1M dataset. The model achieves results competitive with other networks that have been pre-trained on a large dataset like Sports-1M or ImageNet. The best model achieves an accuracy of 83.9% on the UCF-101 dataset.

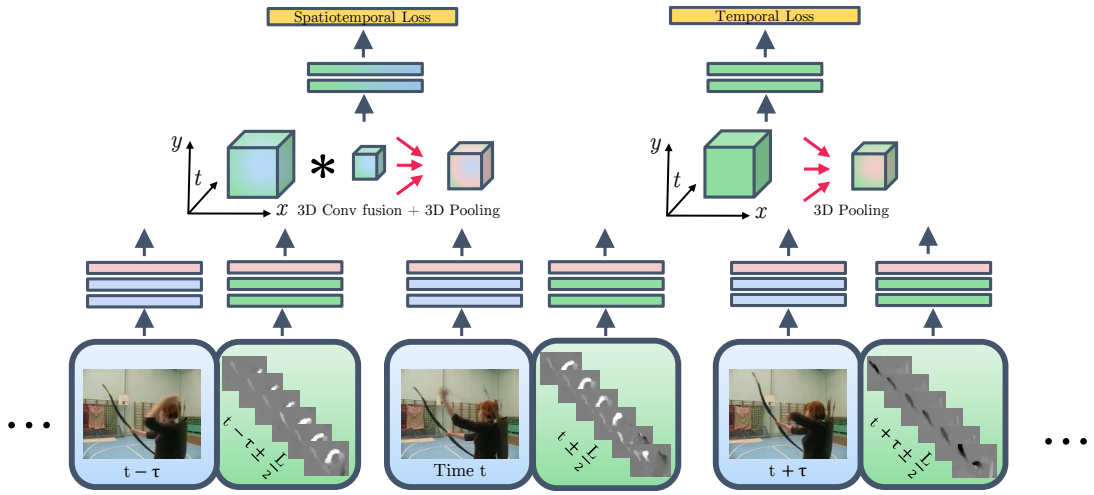

Figure 2.11: Architecture for the Convolutional Two Stream Network Fusion [3]

Feichtenhofer et. al [3] use a two stream network that operates on raw images

and optical flow. They explore various ways to fuse the two streams spatially and temporally. The network architecture is shown in figure 2.11. Out of the multiple ways of fusion, convolution works best to fuse spatially, and for temporal fusing, 3D convolution + 3D pooling works the best. The paper relies on the VGG-16 architecture for the convolutional networks. The model achieves an accuracy of 93.6% on the UCF-101 dataset.

2.3.4 Unsupervised video representation

The Srivastava et al. [16] paper is one of the models that is closest to what we have explored in this thesis. The paper proposes a future predictor model similar to what we do in this thesis. The model is trained for a composite task that reconstructs the input and predicts the future. It combines an autoencoder with a module that predicts the future. Their best composite model gets an accuracy of 75.8%. While a combination of this model with a flow model gets an accuracy of 84.3%.

2.4 Conclusion

As we have seen in this section, capturing motion information is crucial for action recognition. Almost all currently available models use a pre-calculated optical flow or rely on a very large dataset for pre-training. In this work, we try to do away with both of these restrictions. We explore the idea of learning motion features directly from a sequence of images without a supervision signal for optical flow. We do so by making the network predict the next frame of the video. We also only use the UCF-101 dataset.

Chapter 3

Data

3.1 Dataset

The UCF-101 [15] dataset consists of 13,320 videos with 101 different action classes. These actions cover a broad range of activities, which can be divided in to 5 types: Human-Object Interaction, Body-Motion Only, Human-Human Interaction, Playing Musical Instruments, and Sports. It consists of about 27 hours of video with an average clip length of 7 seconds. All the videos have a fixed frame rate of 25 FPS and a resolution of 320×240.

Table 3.1: Summary of UCF-101 dataset characteristics

Actions	101
Clips	13320
Average Clip Length	7.21s
Min Clip Length	1.06s
Max Clip Length	71.04s
Frame Rate	25
Resolution	320×240

The UCF-101 dataset provides a predefined train/test split. The training set contains 8455 videos and the remaining set of 1082 videos are the validation set.

Figure 3.1 shows one frame from each of the 101 action classes.

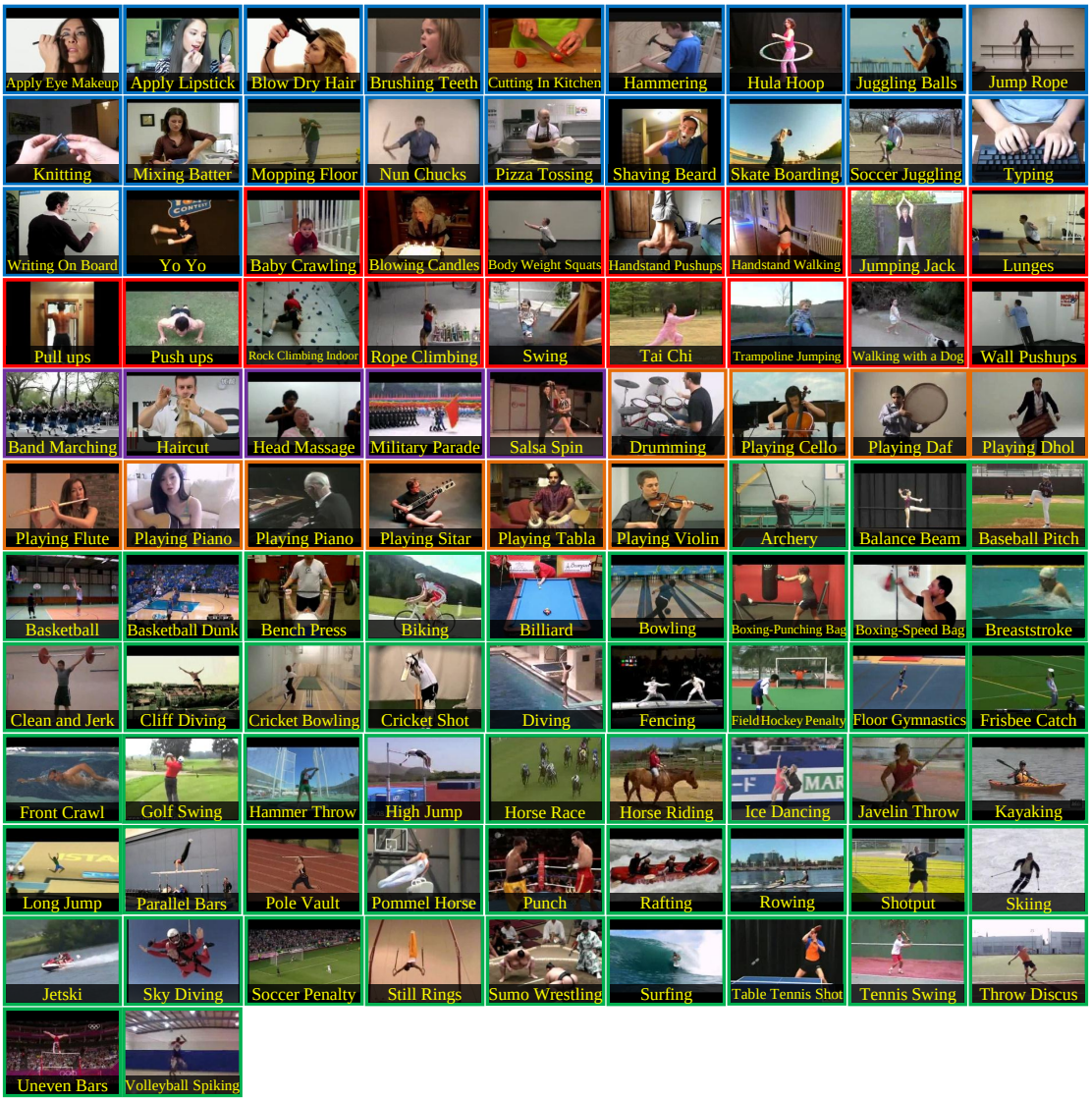

Figure 3.1: 101 actions included in UCF101 [15] shown with one sample frame. The color of frame borders specifies to which action type they belong: Human-Object Interaction (Blue), Body-Motion Only (Red), Human-Human Interaction (Purple), Playing Musical Instruments (Orange), and Sports (Green).

3.2 Preprocessing

The UCF-101 dataset consists of videos in .avi format with a resolution of 320 $\times$ 240 with a frame rate of 25 fps. For this work we use the avconv utility from open source video processing library libav. We extract 30 frames equally-spaced over the length of the video.

We use the 30 frames as input to our CNNs. For AlexNet, we resize the frames to 227×227. For ResNet-50, we resize the frames to 224×224.

Chapter 4

Our Approach

4.1 Models

In the following sections we investigate a few different approaches to understand various aspects of video classification.

4.1.1 Impact of CNN

To understand the impact of the CNN used to process the frames of the video we use a model which uses only the feature representations of the static video images as the input to our network. We experiment with two different CNNs, Caffenet [8] (a minor variant of AlexNet) and resnet-50, which are pre-trained for the ImageNet classification task. We then add a fully connected softmax layer with 101 units to predict the class. We train the final fully connected layer to predict the output classes. We combine the output for all frames by averaging over all the frames to produce the final prediction.

As discussed in section 2.1, it is clear that deeper networks are better for extracting features from static images. We also saw that the ResNet-50 model performed significantly better as compared to the AlexNet model for ImageNet. So, it is not surprising that we see the Resnet-50 model performs significantly better as compared to the BVLC-caffenet model. We only use the Resnet-50 feature representation for the rest of this work. We consider this model to be our baseline.

4.1.2 Super-classes

The UCF-101 dataset has 101 actions which belong to 5 different types, namely, Human-Object Interaction, Body-Motion Only, Human-Human Interaction, Playing Musical Instruments, and Sports. We use this information to train our action recognition network. As in the previous model, the input to the network is the output representation of the pre-trained resnet-50 model. We take the output of the pool5 layer of resnet-50, which is the layer before the fully connected layer. The input is connected to a fully connected layer which has 512 units with a rectified linear units as the non-linearity. This layer is then connected to 2 fully connected layers that act as the output of the network. One layer has 101 units and a softmax to predict the classes. The other layer has 5 units and a softmax which predicts the type to which the action belongs. Hence, the model is a network with 2 heads.

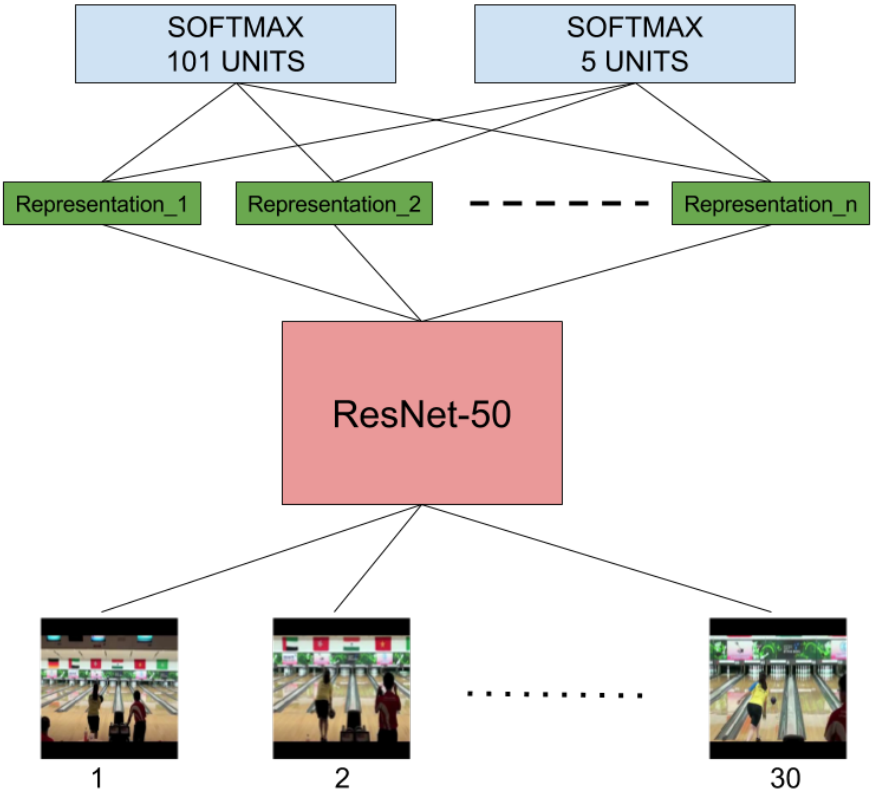

Figure 4.1: Network for the Super-Class model

4.1.3 Predicting the next frame

One good way to learn representations is to by trying to reproduce the input as the output of the network. In many networks that try to model sequences, training the network to predict the next element in the sequence has shown promise towards learning a good representation. RNNs have shown promising results in modelling sequences. However, it is difficult to train standard RNNs to model long sequence due to the problem of vanishing and exploding gradients. In contrast, LSTMs are know to be able to model sequences without these problems. For this reason, we use LSTMs to predict the next frame of the video. The network is show in Figure 4.2.

The input to the network is the 2048-dimensional representation of each frame taken from resnet-50. We have 2 LSTM layers on top of this, each with 2048 units. The 1st LSTM layer acts as the encoder and the 2nd LSTM layer acts as the decoder.

We train this network to learn the representation of the next frame. We use the squared error as the loss function to train this network.

We then use the output of the trained encoder LSTM layer, *R_i* (see Figure 4.2), as the input to another network which is the classifier network. This network has a fully connected network with 1024 units and rectified linear units as the non-linearity. Then, as in the networks in section 4.1.2, we have two fully-connected output layers with 101 and 5 units to predict the action and the category respectively.

4.2 Experiments and Results

We trained the above mentioned models on the UCF-101 dataset. We used the suggested train/test splits on UCF-101 for our experiments. We used adam as the optimizer for training all networks. We use an initial learning rate of 0.01. We used dropout at both the LSTM layers and the fully connected layer to prevent the network from overfitting. For the next-frame prediction model, we pre-train the network to predict the next frame for 30 epochs. After the pre-training is completed, we the add the classifier layers. Figure 4.4 shows the evolution of the loss with the number of epochs. Figure 4.5 shows the training and validation accuracy of the model over epochs. Table 4.1 gives the results for each model.

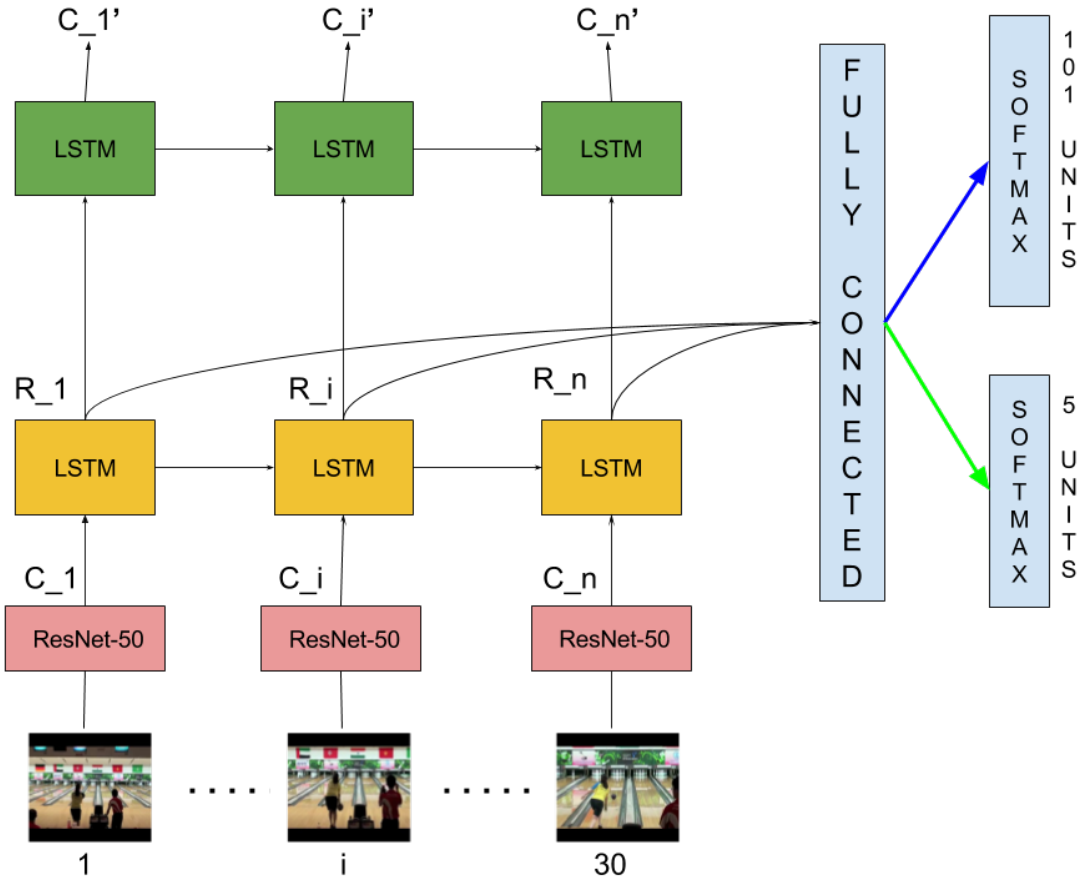

Figure 4.2: Network for the Next frame prediction model: The yellow LSTM layer acts as an encoder while the green LSTM layer acts as a decoder. Blue layers are fully connected layers

1. We observe resnet-50 CNN model performs significantly better as compared to the BVLC-caffenet model. This is not surprising, seeing the performance of these models on the ImageNet task. The image features generated by resnet-50 are better image representations. We get an accuracy of around 64% for the resnet model as opposed to a around 19% accuracy of the BVLC-caffenet model.

2. Given that the action classes belong to 5 types that are very different from each other, we expected that adding the additional supervision of the super class labels to improve the performance. We get an accuracy of around 67%, which is an improvement over the previous model.

3. The next frame prediction model gives an accuracy of about 68%. We expected

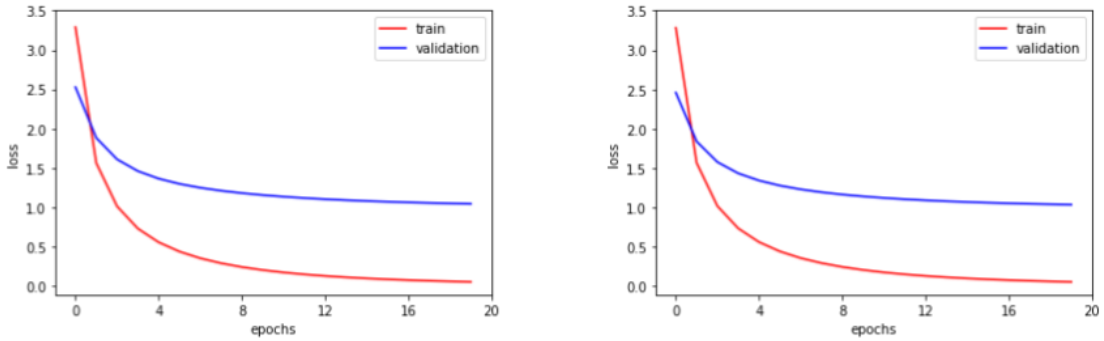

Figure 4.3: Training and validation loss as a function of number of epochs. Left: the super-class network, Right: ResNet with average over frames

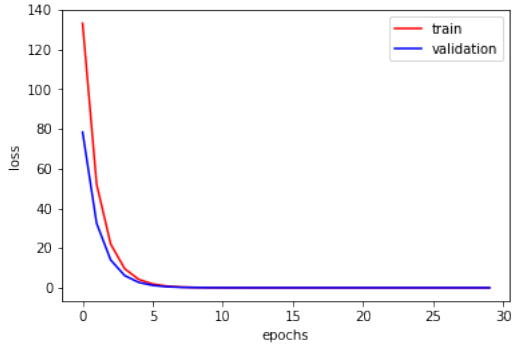

Figure 4.4: Training and validation loss for pre-training the next-frame prediction network as a function of number of epochs

predicting the next frame model to perform better, as it should learn a better representation of the temporal changes in a video. However, evidence shows that it does not improve as much as we had hoped for. The improvement, though small does show an ability to learn temporal information. Taking the number of classes and the difficulty of the task into account, even a small improvement is significant.

Table 4.1: Summary of accuracy of our models

Model	Accuracy(%)
CNN only(BVLC-caffenet)	19.31
CNN only(Resnet-50)	63.76
super-class-model	67.35
next-frame-prediction	68.19

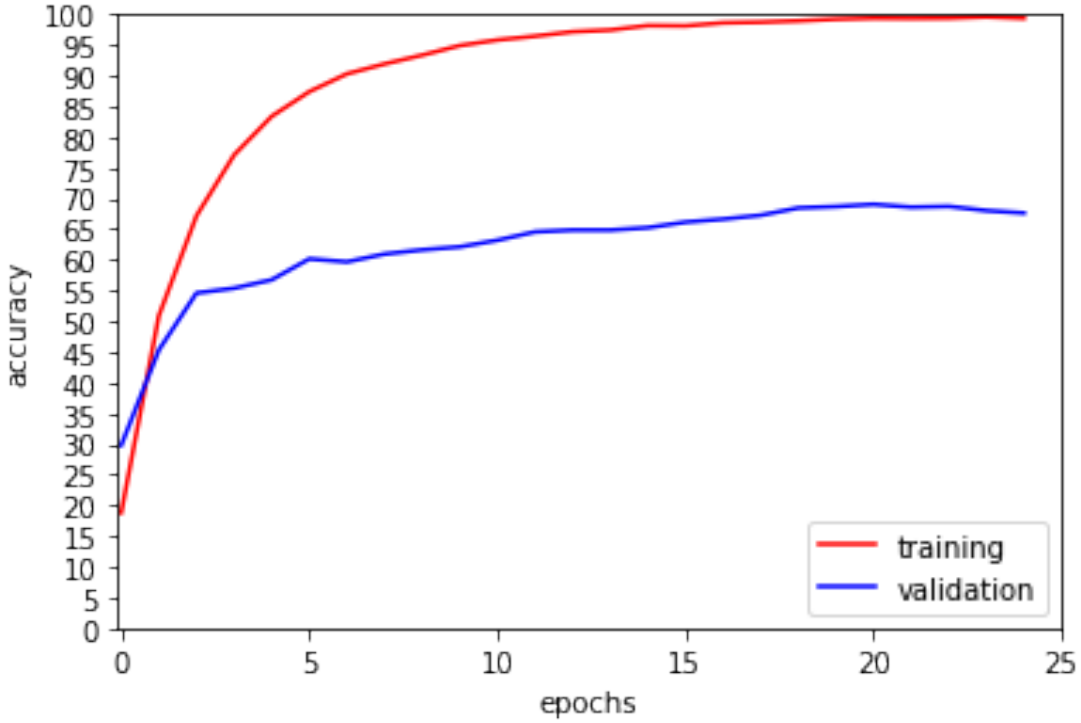

Figure 4.5: Training and validation accuracies for the classifier network with a pre-trained next-frame prediction as a function of number of epochs

Figures 4.6 and 4.7 show frames from several correctly-classified videos. There are some classes that are very similar to each other in terms of both spatial and temporal description. For instance, Applying Lipstick and Brushing Teeth are classes which not only have similar static images but also similar motion of the hand. The network performs particularly poor on such pairs. Figures 4.8 and 4.9 include some incorrectly classified examples.

Finally, table 4.2 shows the results of other works doing action recognition on the UCF-101 dataset. The Slow Fusion CNN model is from Karpathy et al. [9] and is discussed in section 2.3.1. This model uses a network that is pre-trained on the Sports-1M dataset and uses slow fusion to capture the spatio-temporal dynamics of the video. The LRCN [1] is one of the first models to combine CNNs and LSTMs for the video classification task. Section 2.3.2 describes the details of the [1] and [18] models. These models use a combination of optical flow and RGB images and are pre-trained on the Sports-1M dataset to compensate for the small amount of data in UCF-101. Srivastava

et al. [16] use an unsupervised method to train a model to predict the current and the next frame using LSTMs (see section 2.3.4). They combine predictions from RGB and flow models to achieve an accuracy of 84.3%. ActionFlowNet [12] is a model that learns to predict the optical flow and uses the learnt flow as an input to a classifier. Optical flow being a powerful motion feature is used in other state of the art models like the Convolutional Two stream fusion model in [3]. We discuss [3] and [12] model in section 2.3.3. [2] is the current state-of-the-art model which uses optical flow in combination with ResNets. It is evident from these results that optical flow plays a crucial role in understanding videos.

Table 4.2: Comparison with state-of-the-art action recognition models

Model	Accuracy(%) on UCF-101
Slow Fusion CNN [9]	65.4%
LRCN [1]	82.34%
ActionFlowNet [12]	83.9%
Composite LSTM [16]	84.3%
Two stream ConvNet [13]	88.0%
LSTM + CNN (Optical Flow + Image Frames) [18]	88.6%
Convolutional Two stream fusion [3]	92.5%
Convolutional Two stream fusion+IDT [3]	93.5%
ST-ResNet [2]	93.4%
ST-ResNet+IDT [2]	94.6%
(Ours)next-frame-prediction	68.19%

Sumo Wrestling

Rock Climbing Indoor

Writing On Board

SkiJet

Figure 4.6: Frames corresponding to videos that are correctly classified by the model

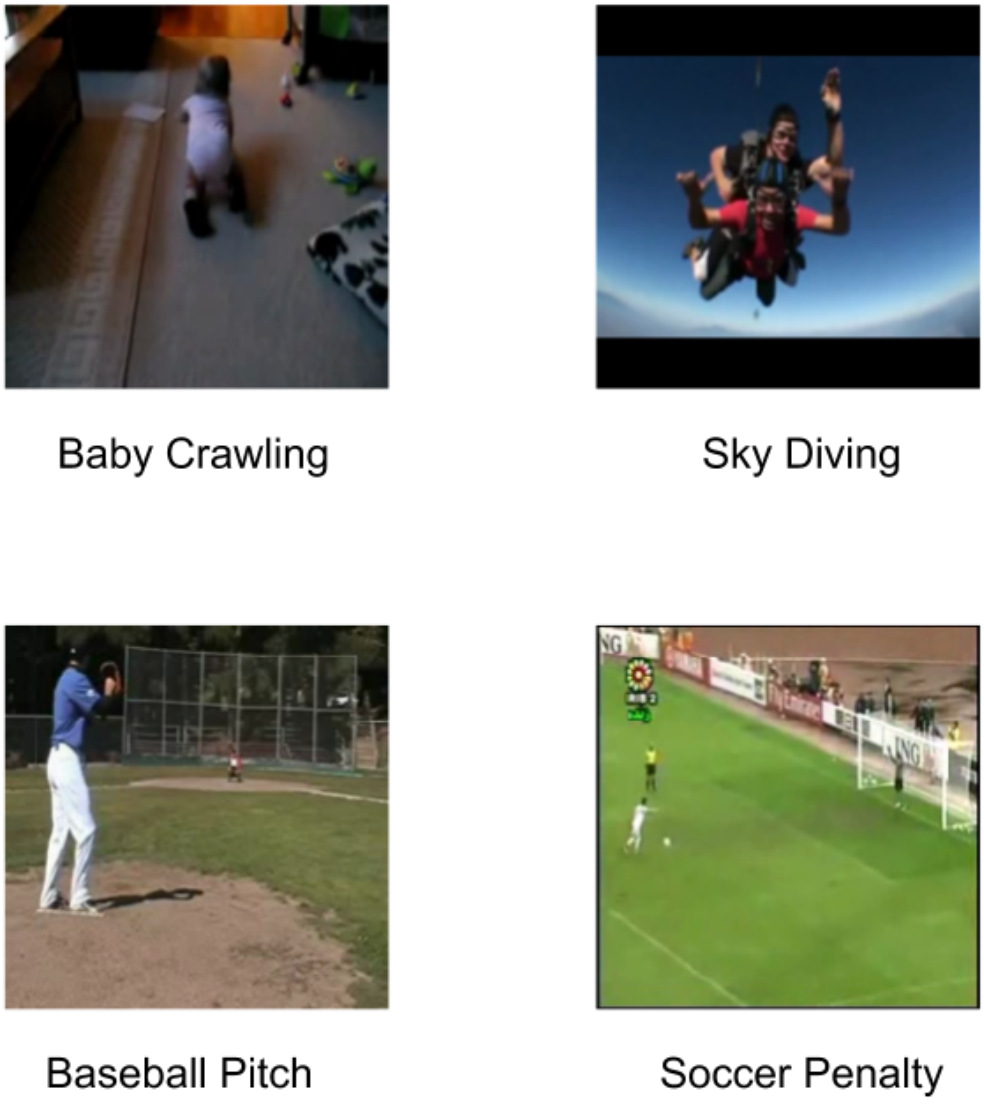

Figure 4.7: Frames corresponding to videos that are correctly classified by the model

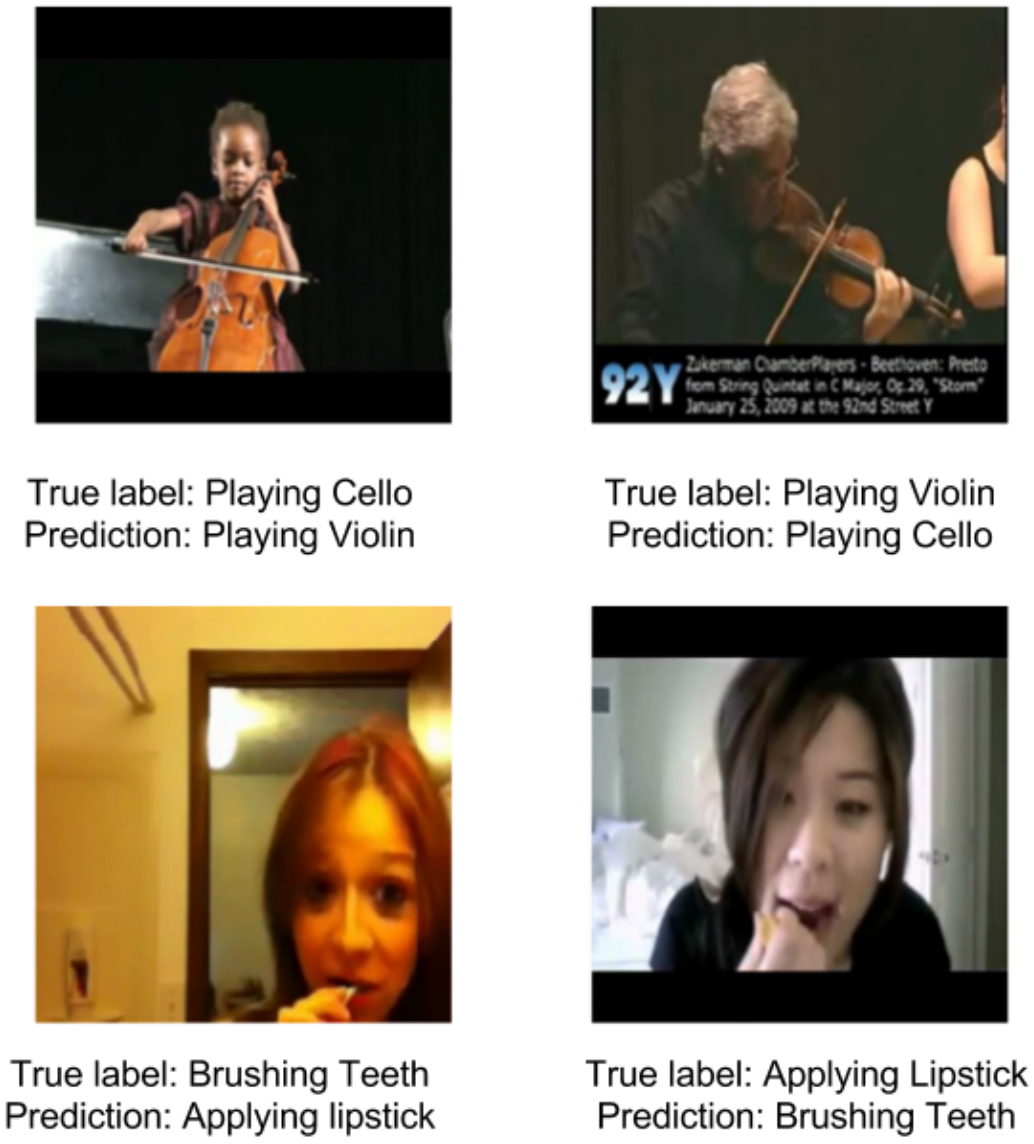

Figure 4.8: Frames corresponding to videos that are incorrectly classified by the model

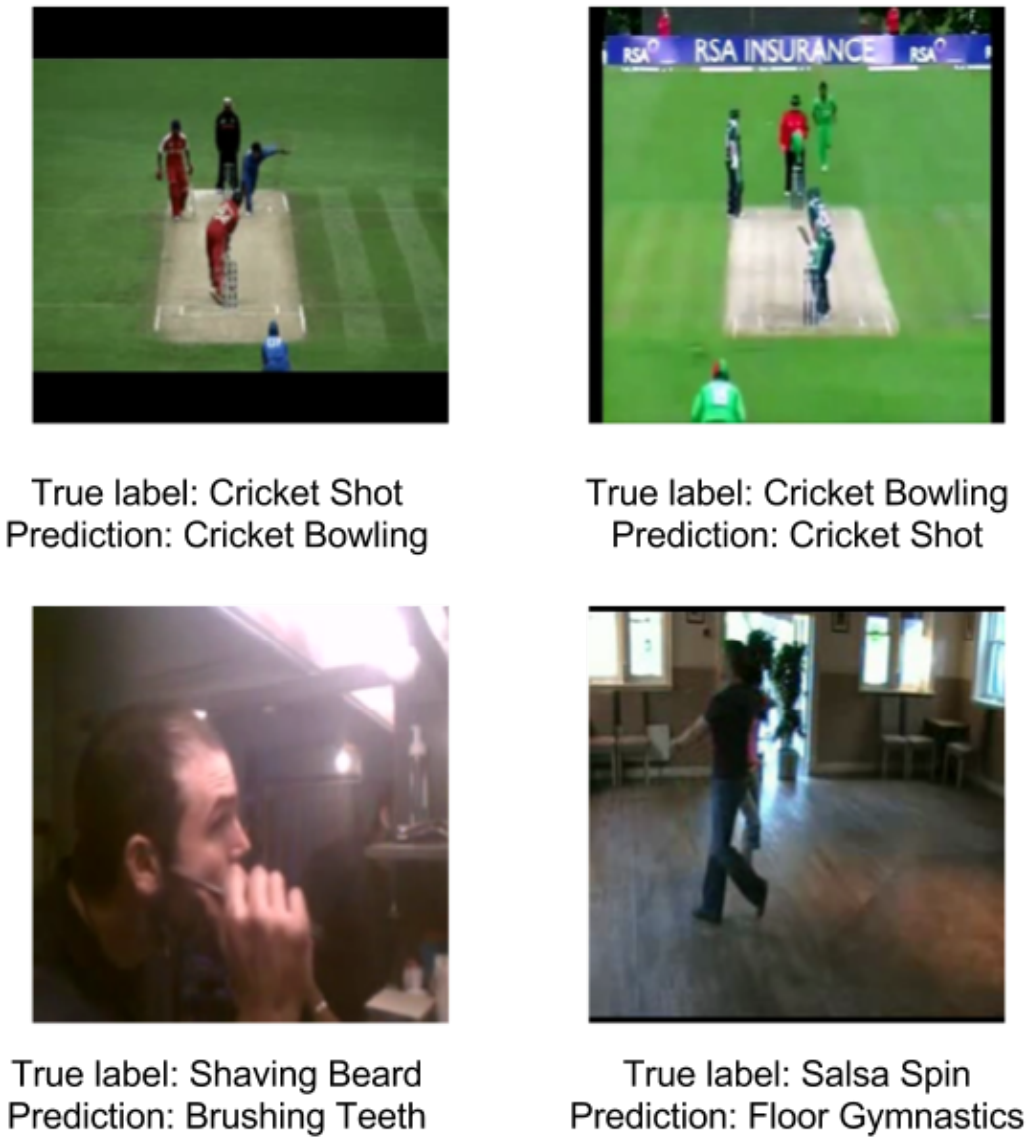

Figure 4.9: Frames corresponding to videos that are incorrectly classified by the model

Chapter 5

Conclusion and Future Work

5.1 Conclusion

We observe that the image representation from ResNet-50 significantly outperforms AlexNet. This shows that a good representation of static images is essential for good video classification. From related works we can conclude that optical flow is a good motion feature and captures temporal information that enables action recognition. Further we also observe that pre-training a model on larger relevant dataset like Sports-1M or ImageNet helps the network learn a better representation. This is expected because of large variety of data contained in these datasets.

Finally, predicting the next frame of a video seems to learn at least some relevant features. We observe a small improvement in accuracy, but given the number of classes and difficulty of the task, this small improvement is significant. The encoder-decoder based model for video representation can be used for a variety of video understanding task as the network's representation is not classification specific.

5.2 Future Work

Although the results from this work may seem like small improvements, this sets of models are worth exploring. There are have been various attempts to successfully model sequences by predicting the next element. Most recent promising results are in

the field of NLP. Here are a few ideas which could be explored in future work.

- Actions are predominantly determined by movement of joints of the person involved in the action. A dataset which contains both video and 3D skeletal data was released in CVPR'16. Using skeleton data along with the video would definitely yield better results in action recognition.
- Time-domain pooling has shown promising results when combining representation of sequences. Adding a convolutional layer before the LSTM layers in the encoder might yield interesting results.
- Attention models have worked well in various tasks. Use of attention to focus on the correct frames is another possible area to explore.
- Actions are also determined by the objects involved in the action. Explicit object detection can lead to better understanding the context of the video.

Bibliography

[1] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015.

[2] Christoph Feichtenhofer, Axel Pinz, and Richard P. Wildes. Spatiotemporal residual networks for video action recognition. *CoRR*, abs/1611.02155, 2016.

[3] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[5] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[8] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[9] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, NIPS'12, pages 1097–1105, 2012.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[12] Joe Yue-Hei Ng, Jonghyun Choi, Jan Neumann, and Larry S Davis. Actionflownet: Learning motion representation for action recognition. *arXiv preprint arXiv:1612.03052*, 2016.

[13] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, 2014.

[14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[15] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[16] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using LSTMs. In *ICML*, pages 843–852, 2015.

[17] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[18] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4694–4702, 2015.