
Australian National University

# Multi-deep-models for Action Prediction (MDAP)

Abu Abraham

# ABSTRACT

One of the major challenges faced in very deep models for action prediction is the lack of correctly labelled distributed datasets. We eliminate this dependency on labelled data set by combining various existing state-of-the-art models. Making use of object-detection, environment detection and action recognition models, we establish relationships between actions and objects in the identified environment. This relationship and the flow of information is learned by an LSTM which predicts the upcoming human action. The model produced an accuracy of 64% in the ucf101 data set and 82% in the created BAS data set.

# Contents

# 1 Introduction

The ability to foresee actions and prevent them if necessary would be one of the ultimate goals of Artificial intelligence systems. Though there are several state-of-the-art action recognition mechanisms, action prediction techniques are still fledgling. Action prediction mechanisms are of growing importance with applications in surveillance cameras and also in autonomous cars, to predict the pedestrian movements [1].

Both action recognition and prediction tasks are challenging due to factors including, change of viewpoints, change in lighting conditions and also noise. In addition to these, learning a sequential flow between frames, each with a large amount of visual data poses a major challenge [33]. One of the common approaches to prediction is using a deep feature extractor and making the sequence model learn the extracted features []. However, these systems require a huge amount of varied labelled training data to avoid overfitting of the model and also to learn multiple scenarios [33].

We suggest an approach based on how humans perceive actions, recognizing actions along with narrative understanding [2]. Instead of using deeper models, we propose the use of multiple deep models, each assigned a specific task; an object detector, an action recognizer, environment detector, and a sequence predictor. Each model except the sequence predictor is trained independently. To train the sequence predictor, we use the relationship between objects, actions, and the environment extracted from each frame in the training input, therefore, we do not need any labelled training data to predict the sequence. This model, the Multi-deep-model for Action Prediction is discussed in detail in the upcoming chapters.

## Contribution

The main contribution of this report is, we suggest a method deviating from the usual approach of expanding and re-engineering the deep models. Instead, we devise a mechanism based on the cognitive prediction skill of the human brain by combining the existing state-of-the-art models and also, by using very limited labelled training data.

## Outline

We first introduce all the necessary background required for the suggested approach. Then, we discuss the existing prediction mechanisms in Chapter 3. This is followed by discussing the architecture of the introduced method and the decisions made in detail. Then, the results obtained by the model is shown followed by a brief discussion on the limitations of the suggested approach.

# 2 Preliminaries

## 2.1 Neural Networks

An Artificial Neural Network (ANN), is a computational model inspired by the human brain [1]. Neural Networks have attracted tremendous interests in various fields of study [4]. The ability of networks to understand and handle a large number of features makes it an attractive option over traditional statistical machine learning methods [4].

The basic component or unit in an ANN is a neuron, otherwise known as a node. It receives input from external events or other neurons, transforms it by multiplying with the associated weights (matrix) and then, applies an activation function. There are several activation functions such as sigmoid; squashes the real-valued input to a range [0,1], tanh; squashes to [-1, 1], ReLU; thresholds the real-valued input to 0, to name a few.

The feed-forward neural network is an ANN where all the connections go forward in one direction. A feed-forward network consists of 3 types of nodes [5].

1. Input: These nodes provide information from the outside world. No computations are generally performed on these nodes.

2. Hidden: They have no connection with the outside world. All the computations are performed by these nodes. A collection of hidden nodes are known as hidden layers and there can be multiple of them.

3. Output: These nodes provide the output.

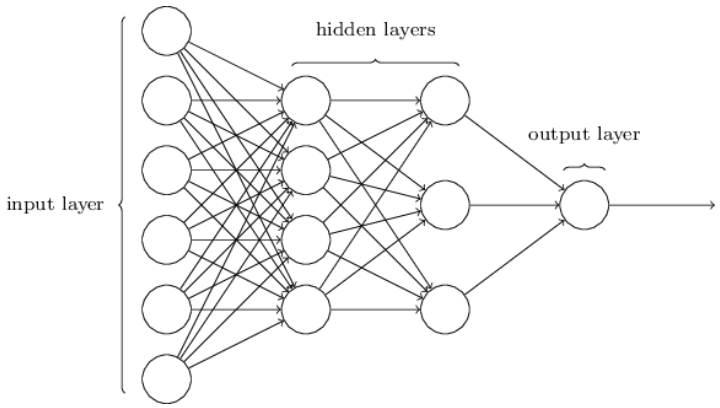



**Figure 2.1.1:** A Multi-Layer Perceptron with 2 hidden layers [5]

There can be multiple neurons or nodes arranged in a layer with connections between them. Two main examples of feed-forward networks are Single-layer Perceptron and Multi-layer

Perceptron. The former has no hidden layers and is not widely used. Whereas the later can have multiple hidden layers.

The network is trained by a technique known as backpropagation or backward propagation of errors [3]. This is a technique to update the weights used by each node to reduce the error [5]. The weights are updated by the formula;

*New weight = old weight — Derivative Rate * learning rate.*

Where the learning rate is a very small value used to smooth the weight updates and the derivate rate is the derivative of the difference between the expected value and the actual value, otherwise known as the error. The entire learning process of a neural network is shown below.

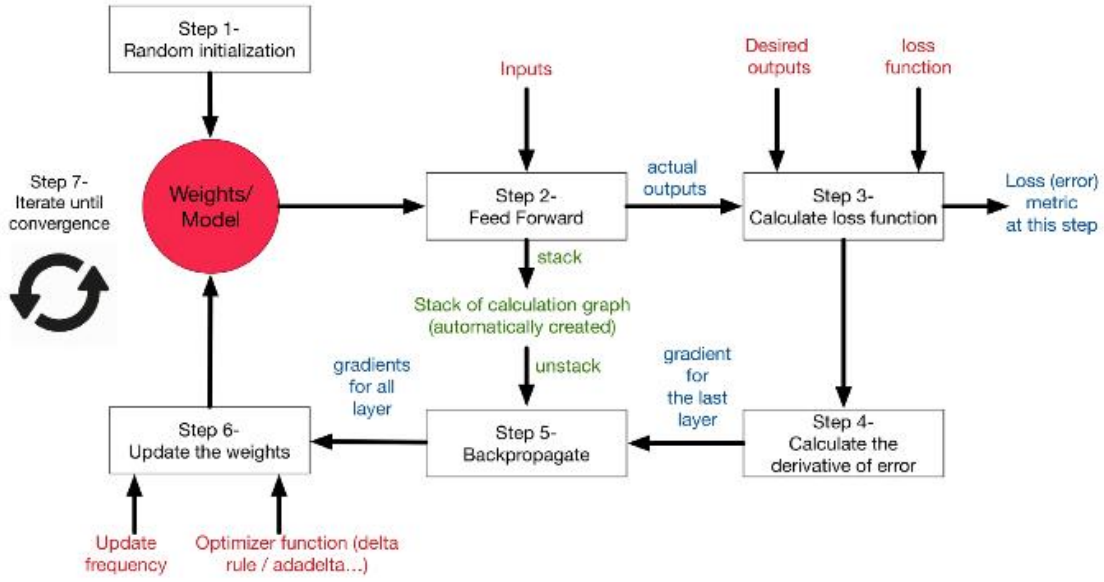



**Figure 2.1.2**: Step-by-step working of neural network during training phase [6]

## 2.2 Convolutional Neural Network

Similar to regular Neural Networks, Convolutional Neural Networks (CNN/convNet) are made up of neurons with weights and biases. One of the major drawbacks of a fully connected ANN is they don't scale well on images [7]. For instance, if an image of size 100x100x3 is taken as input, the first layer will have 30000 units and with additional layers, it will increase further. ConvNet's address this problem by using a constraint architecture [7]. Unlike the regular neural network layers of ConvNet's are arranged in 3-dimension; width, height, and depth. Also, 3 different types of layers are used; convolutional layer, pooling layer and fully-connected layer [7].

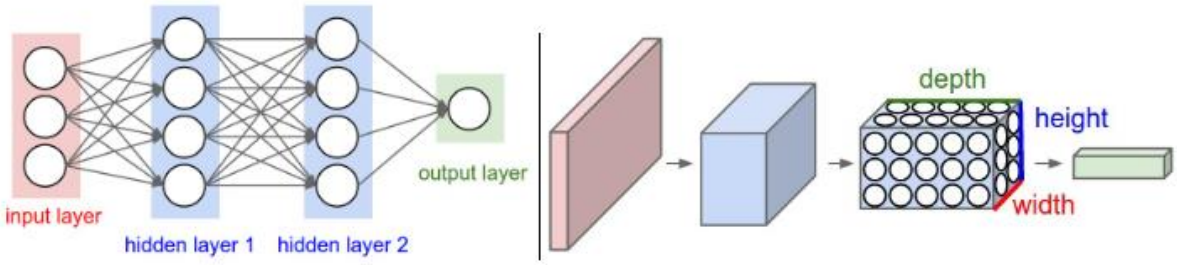



**Figure 2.2.1** Comparison between a regular 3 layer neural network and a convNet which arranges neurons in 3 dimensions (Name, Year)

The usage of a convolutional layer is one of the primary concepts introduced in CNN. In CNN, convolution is performed using filters/kernel to produce a feature-map. These filters slide over the entire image, based on the stride length and compute the similarity by multiplying the filter with the image.

We perform this operation for different filters and then use activation function to make each output non-linear. A ReLU activation function is commonly used. A stride in CNN is how many steps each filter moves when it iterates through the entire image. After the activation, we use pooling layer to pool/ select only higher-valued cells. Pooling layer hence helps in reducing the dimensionality of the input. The most common pooling mechanism used is a technique known as max-pooling (shown below). After the pooling layer, a fully connected layer is used to classify the image.

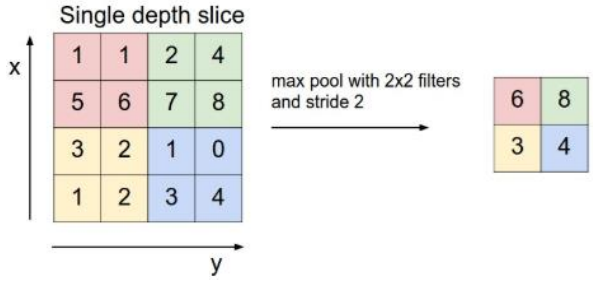



**Figure 2.2.2:** Downsampling using max-pooling [7]

### 2.2.1 3-Dimensional CNN

Traditional ConvNet's can act directly on raw-inputs, however, they can only handle 2-dimensional inputs. To consider both the temporal and spatial attributes, we need to use 3-dimensional ConvNet's with 3-d filters [8]. In video analysis problems it is necessary to capture the sequential flow of information in multiple contiguous frames [8]. We can use 3-d filters/kernels to capture these sequential flows of information. The other layers in the network

behave similarly to the traditional convNet. 3-d CNN is further explained in section 4.3 while discussing various methods for action recognition.

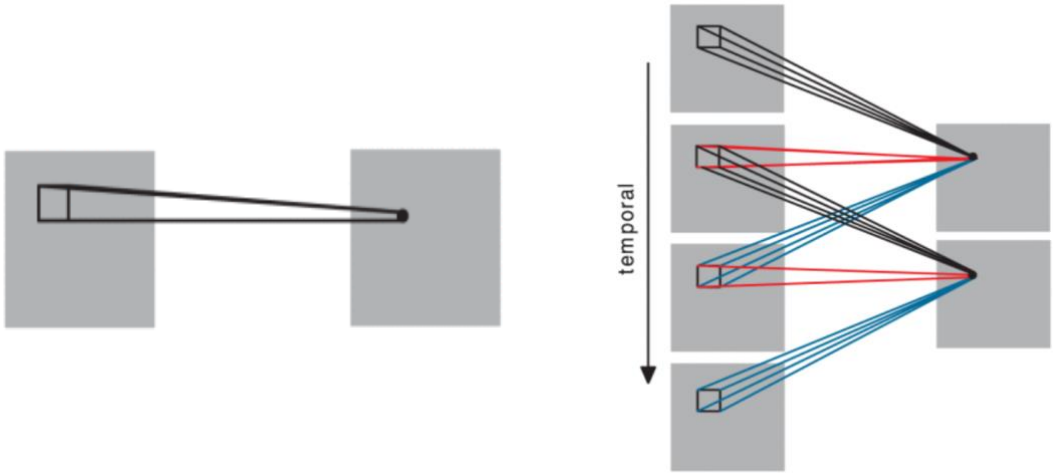



**Figure 2.2.1.1** Comparison of 2-D (left) and 3-D convolutions [8]

## 2.3 Sequence Networks

### 2.3.1 Recurrent Neural Networks

Traditional Neural Networks such as MLP and CNN's fail to capture the sequential flow of information [9]. These networks consider each input as independent of each other. For sequential tasks, such as predicting the next word in a sentence we need to keep track of the words seen so far. Recurrent Neural Networks (RNN) address this problem. RNN's uses networks with loops to act as a memory unit to store the information seen so far [9].

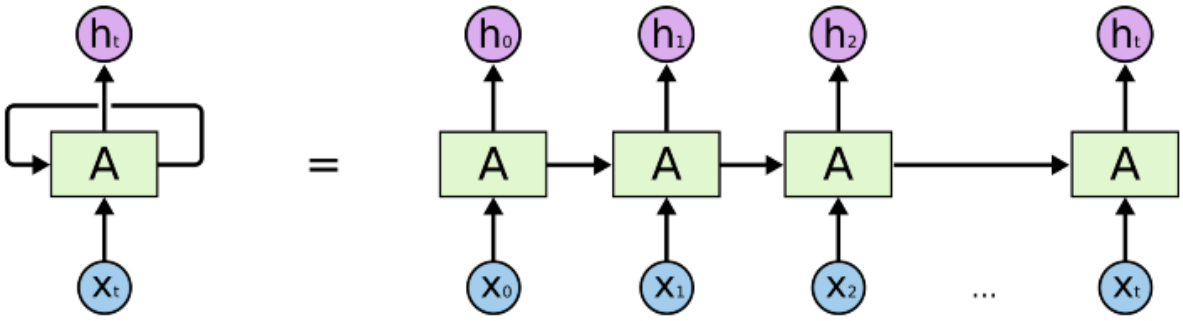



**Figure 2.3.1.1:** An unrolled Recurrent Neural Network [10]

Though RNN's may seem like an ideal solution for sequential tasks, they suffer from the *vanishing gradient* problem. Sometimes, we may come across a scenario where we must

remember an event that occurred long before compared to the recent ones. RNN's are unable to do this as weights are updated to remember the most recent information [9].

### 2.3.2 Long Short-term Memory Network

Long Short-term Memory network (LSTM) is a variant of Recurrent Neural Network, capable of handling the long-term dependencies and avoiding the vanishing gradient problem [11]. They are key to many of the sequence prediction tasks and are used in many areas in this project. The repeating module in RNN consist of a single unit whereas in LSTM there are four; input gate, forget gate, output gate, and a memory cell to control the storage of data.

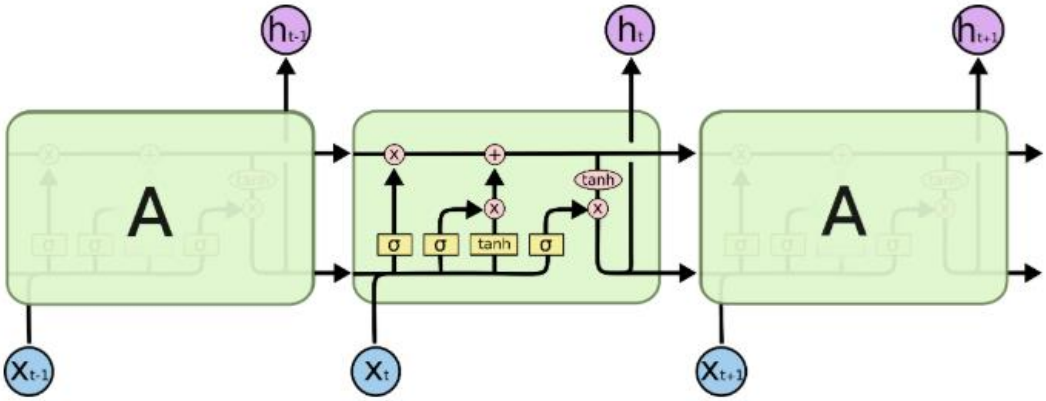



**Figure 2.3.2.1:** Repeating nodes of an LSTM [10]

The key idea in LSTM is to determine what information to store and what to forget. This is handled by the forget gate by considering the previous value, $h_{t-1}$ and the current input $X_t$. The forget gate uses a sigmoid with the value 1 to store and 0 to completely forget.

$$f_t = \sigma\big(W_f(h_{t-1}, x_t) + b_f\big)$$

To determine what information to store we use 2 gates. An input gate, a sigmoid layer to determine what values to update and a tanh layer to determine the candidate values to be added to stored values.

$$i_t = \sigma(W_i(h_{t-1}, x_t) + b_i)$$
$$\vec{C}_t = \tanh(W_C(h_{t-1}, x_t) + b_C)$$

The output of the LSTM cell is computed by determining what part of the cell state to output, by a sigmoid layer and a tanh layer.

$$o_t = \sigma(W_o(h_{t-1}, x_t) + b_o)$$
$$h_t = o_t\big(\tanh(C_t)\big)$$

## 2.4 Encoding

In many machine learning applications, the input data could be categorical i.e. some values could be labels or characters. By way of example, if we have a column representing cities, then its values could be [“Sydney”, “Canberra” ...]. However, most machine learning models except for decision trees cannot work with these categorical labels and we need to convert them into numerical values [12]. To convert these to integer format, we can use 2 approaches.

1. Integer Encoding: Integer encoding is the process of assigning a unique value for each label. In our examples of cities, we encode “Sydney”: 1, “Canberra”: 2 and so on.

2. One hot encoding: Using an integer encoding may result in model assuming an ordering of data and learning higher valued categories more quickly [12]. To avoid these, we can use one hot encoding where we encode labels as vectors. For instance, in our example we encode "Sydney": [1, 0, 0 ...], "Canberra": [0, 1, 0 ...].

## 2.5 Object Detection

Object detection is the process of identifying both objects and their position in an image frame [13]. Object detection differs from the image recognition problem. The primary objective of an image classification task is to classify the entire image to a labeled category. Whereas, object detection involves detecting multiple objects and their position in the input image. To identify multiple objects, we extract features in each frame using various techniques. This chapter will briefly discuss the feature extraction techniques and also the multiple object detection models.

### 2.5.1 Feature extraction

Feature extraction is the process of extracting features from the image. CNN’s are primarily used for these tasks and have been very successful and widely used since the introduction of AlexNet [14].

#### 2.5.1.1 AlexNet

AlexNet was one of the first deep convNet models to produce state-of-the-art accuracy in image recognition tasks. The model has 5 convolutional layers followed by 3 fully connected layers [14]. For faster training, AlexNet uses ReLU (Rectified Linear activation Unit), instead of tanh/sigmoid [14]. Also, dropouts are used to avoid overfitting [13].

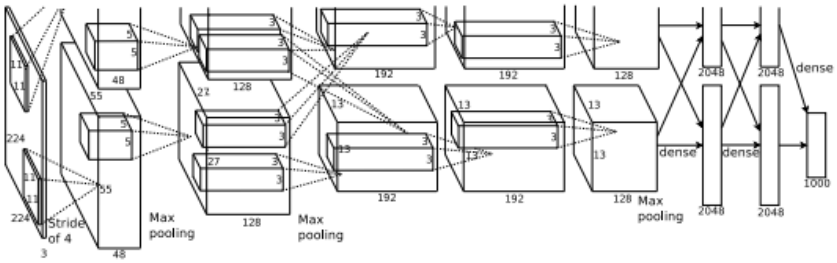



**Figure 2.5.1.1.1:** Illustration of the architecture of AlexNet [13]

### 2.5.1.2 VGG16

The VGG16 model was proposed by the VGG Group at Oxford. This model differs from AlexNet by replacing the large filters of the AlexNet by multiple 3x3 kernel filters one after the other [15]. Multiple non-linear layers can increase the depth of the network enabling it to learn more intricate features. Hence, VGG16 could achieve a much higher accuracy compared to AlexNet on the ImageNet dataset [13].

### 2.5.1.3 Inception

Prior to the introduction of inceptions, engineers were stacking up layers to improve the performance of the system. Inception model brought a change to this system by introducing multiple layer filters and combining those together [13]. One of the major concepts behind the design of inception was, the objects in an image could be of various sizes, and hence it is difficult to determine a fixed kernel size. Below is a naïve architecture of inception model, it has 3 different filters of varied size and a max pooling unit [13]. The output from these filters is then concatenated. In the first Inception model released by Google, 9 inceptions layers, (modified with additional 1*1 convolution to improve performance) were stacked up.

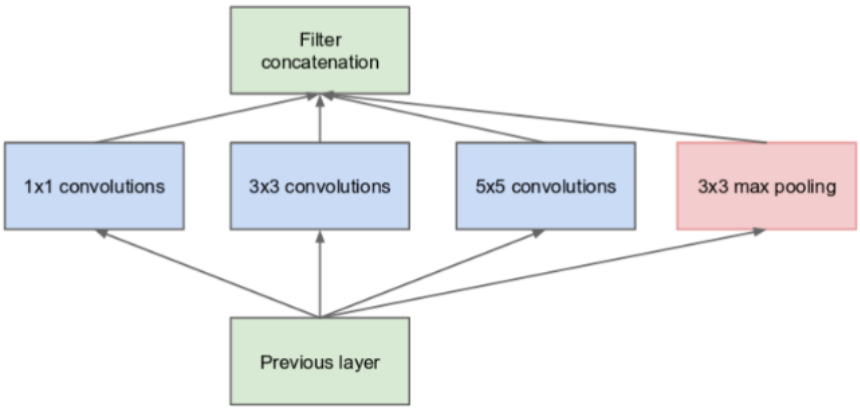



**Figure 2.5.1.3.1:** Architecture of a naïve inception module [16]

#### 2.5.1.4 ResNet

The main problem with increasing depth of networks is that the change in weights arising at the final layers of the network decays when it reaches the initial layers [13]. This problem is known as the vanishing gradient. As a result of this problem, the network is prone to lesser learning at the initial layers. ResNet avoids this by constructing networks using residual blocks with key idea being the so-called “identity shortcut connection” that steps one or more connections [13].

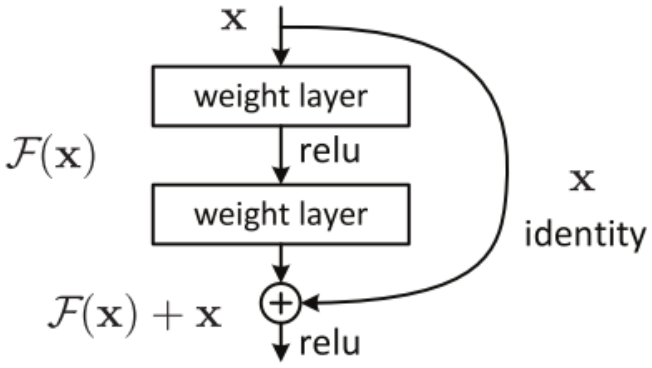



**Figure 2.5.1.4.1:** A basic residual block [17]

### 2.5.2 Object detection

#### 2.5.2.1 Sliding Window

This is a naive approach in object detection. A fixed size sliding windows can be used across the frame to localize images. At each sliding window, features are extracted using a feature detector. If the score of the detected object is greater than a threshold, we mark the bounding box with the identified label [18].

#### 2.5.2.2 Region-based detectors

Region-based detectors avoid the drawbacks of sliding window approach such as exhaustively searching the entire frame. One of the key features in a region-based approach is the usage of selective search. Selective search combines the good of exhaustive search and image segmentation [19]. With selective search, we group each pixel to the most similar ones around in a hierarchical manner.

Region-based convolutional network (R-CNN) is one of the earliest region-based approach using selective search. R-CNN combines selective search and convNet to detect regions proposals and detect objects in the detected regions. R-CNN uses 2000 regions of interest and each of these is then fed to the CNN network individually [19]. It is then followed by a fully connected network to calculate the final result [19].

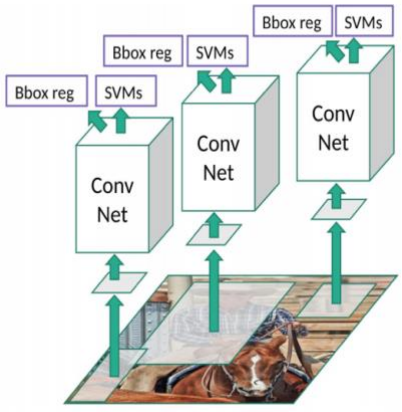



**Figure 2.5.2.2.1:** The working of R-CNN [18]

However, this approach is still very time-consuming as we analyze a high number of regions. With Fast Region-based Convolutional network (Fast R-CNN), we can minimize the running time by extracting features using a feature extractor on the feature map [20]. The identified Regions of interest are then used to detect features with a softmax classifier [20]. Both R-CNN and Fast R-CNN uses selective search to find the region proposals. This can be avoided by using a separate network to predict each region, this approach is called Faster R-CNN [21].

Faster R-CNN computes region proposals using another CNN, a Region Proposal Network (RPN) [21]. RPN takes an entire image frame as input and outputs rectangular proposals. This is done by using a 3x3 size sliding window over the image. Each selected window is termed an anchor. These selected anchors are in-turn fed to the next layer, a Fast R-CNN [21].

#### 2.4.2.3 **You Look Only Once**

You look only once (YOLO) is an approach that doesn't look at the entire image, but parts of it. It splits the image into a grid of S*S and then selects m boxes from these grids. Features are extracted from each of the 'm' boxes and the identified features greater than a threshold are selected. This approach is fast; however, fails to identify minute details in images [18].

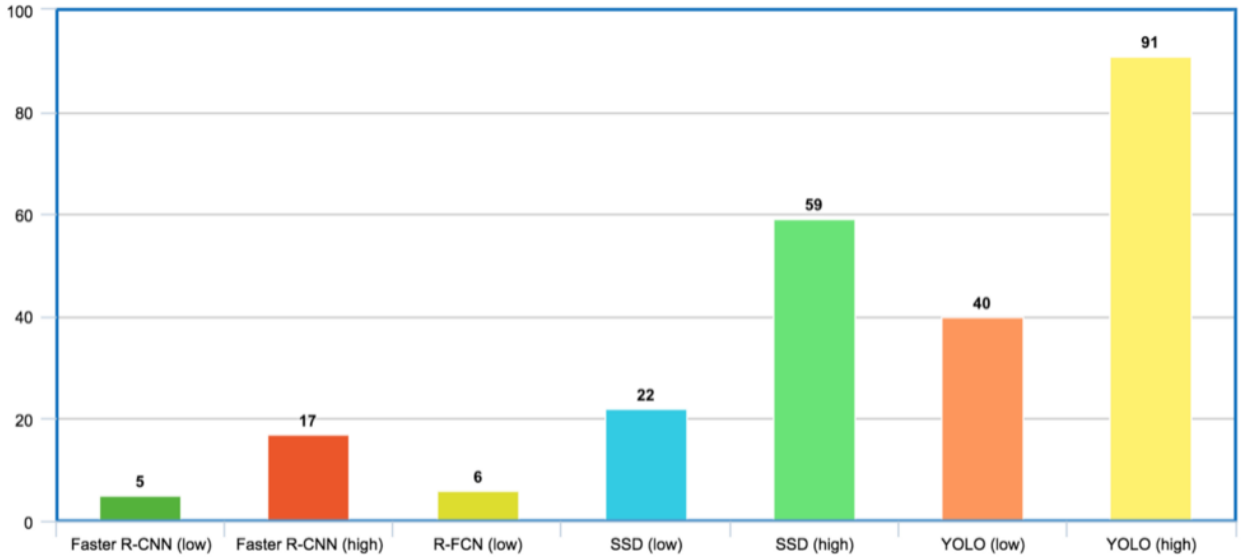



**Figure 2.4.2.3.1:** Frames per second comparison of variants of R-CNN, YOLO and SSD [18].

## 2.6 Action Recognition

Action recognition is the task of recognizing and labeling actions in a video sequence [22]. Automatically detecting actions are useful in many applications including, video surveillance and human computer interaction. Unlike Image recognition, action recognition is a more challenging problem due to factors such as change of illumination, camera view point and the similarity of visual contents.

Over the last decade, researchers have tried many hand-crafted methods for action recognition. These systems extract low-level features from the video frames and are then fed to either a Support Vector Machine or a decision tree. However, this method fails to provide accurate results when two actions have similar actions and shape [22].

With deep neural networks, there are different ways of overcoming these challenges in detecting human actions. Similar to the Image recognition task, CNN's have shown to achieve good performance in action recognition. Due to this reason, many approaches try to use the same 2-D convNet kernels. However other approaches such as using a 3-D convNet or 2 stream networks capturing spatial and temporal information between frames are proposed [3]. These methods are briefly discussed below.

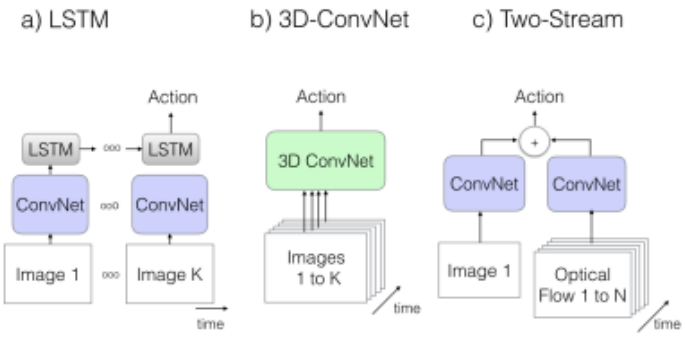



**Figure 2.6.1:** Video architectures considered in this project [23]

### 2.6.1 CNN+LSTM

CNN has already been established as an efficient method for classification and representation of images. It makes sense to make use of the already well performing CNN models such as an ImageNet/ResNet for video classification. With 2-D convNet, we can extract features from each frame in the video sequence and pool them across the whole video [24]. However, this approach fails to learn the temporal features, i.e. it wouldn't be able to determine, whether the action is opening the door or closing it [24].

A better design choice is using LSTM to learn the sequence of images rather than pooling them [24]. With this approach, each frame is represented with a CNN and is fed in sequence to an LSTM. CNN finds hidden patterns in images and these are learned using an LSTM. The performance of these networks can be increased by using a multiple-layers of LSTM's in a hierarchical manner. By stacking LSTM layers, each layer receives input from the previous hidden layer as input. Further improvements can be achieved using Bi-Directional LSTM's as the output at the current frame is not just dependent on the previous frame, but the next as well [24]. The architecture of this model is shown below.

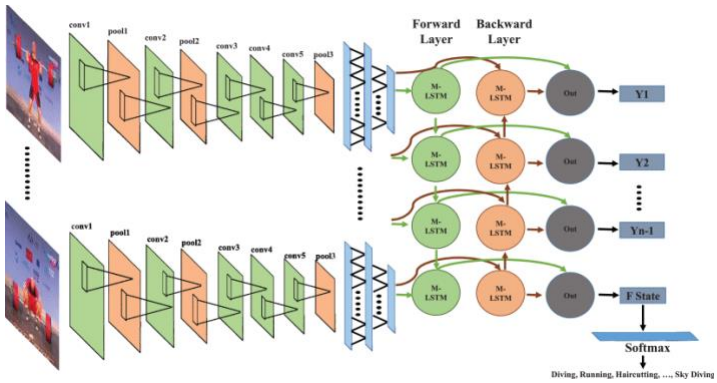



**Figure 2.6.2.1:** Architecture of a CNN-LSTM model [24].

### 2.6.2 3-Dimensional CNN

3D ConvNet's are similar to standard ConvNet, though with additional spatio-temporal filters. Here, we adopt 3D kernels instead of 2D and hence use 3-dimensional features. This approach has a huge limitation, with the additional dimension, we need much more parameters than a 2D model. Training them on datasets such as UCF101 and HMDB51 leads to lower performances compared with LSTM models [23].

To address these limitations many techniques have been suggested, most notably by Carreira et al. and Kensho Hara et al. [22, 23]. This include, inflating 2D kernels to 3D, i.e. endowing them with additional temporal dimension making them cubic [22]. Bootstrapping is another technique, where we bootstrap parameters from 2D ResNet models and copy them in a sequence by repeating weights N times across the 3rd dimension and rescaling them [22].

### 2.6.3 2-Stream Networks

While the LSTM models can represent high level features, they fail to capture low-level variations. A different approach to this is suggested by Simonyan et al [25]. Videos can be broken down into the spatial and temporal parts, where the spatial part forms the frame appearance and carries information about scene and objects across the frames. Whereas, the temporal part conveys the movement of camera and the change in viewpoints [25].

In the 2-stream model, these temporal and spatial streams are identified using deep convolutional networks and are then fused together. For temporal, averaging predictions from RGB frames and for spatial, a stack of 'n' optical flow frames [25]. An optical flow is a way of representing how objects flow across the frames. If 'w' and 'h' is the width and height of the video and d, the displacement across the frames, the spatial information across L frames is constructed as,

$$I_\tau(u, v, 2k-1) = d^x_{\tau+k-1}(u, v),$$
$$I_\tau(u, v, 2k) = d^y_{\tau+k-1}(u, v), \quad u = [1; w], v = [1; h], k = [1; L].$$

The channel encodes $I_T$ *(u, v, c), c = [1:2L],* encodes motion at point (u, v) across L frames. Also, d is the displacement vector and, $d^x$ and $d^y$ are the image channels in horizontal and vertical directions respectively.

## 2.7 Attention Mechanism

Attention mechanism in the neural network is closely related to human attention. We often focus on certain things and not on the entire information we have [26]. Similarly, with attention mechanism, we model the network to prioritize certain information.

### 2.7.1 Attention mechanism in Neural Machine Translation

One of the earliest use of attention mechanism is in Neural Machine Translation (NMT). In NMT, we encode the input string into a fixed length vector and generates translation based on that [27].

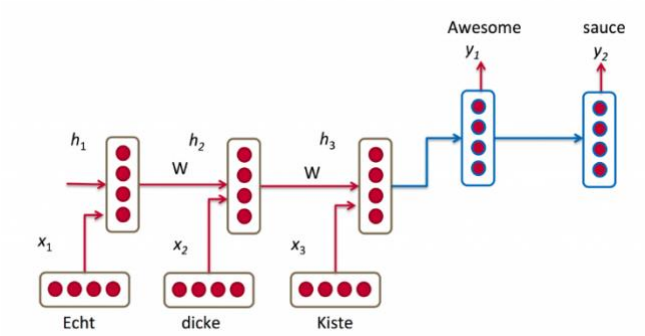



**Figure 2.7.1.1:** How NMT works without attention [27]

In the picture above, the decoder is supplied solely with the output of the last hidden state. It is quite hard to imagine that we can encode a large string say, with 40 words in a single vector. Attention mechanism comes into play here. With attention mechanism, we do not encode the whole sentence into one based on the last hidden layer. Instead, we feed all the outputs to a model, which keeps what is important in the sentence [27].

### 2.7.2 Attention mechanism in video captioning

The initial Video Captioning models used a CNN to extract features (encode) and an RNN to generate sentences based on these extracted features (decode). One of the earliest such models is the Show and Tell [28]. However, in addition to this, the Show Attend and Tell model introduced the usage of attention in image frames [29]. The concept is closely related to the NMT model, instead of storing all the features, we do a selective method of giving more importance to certain details.

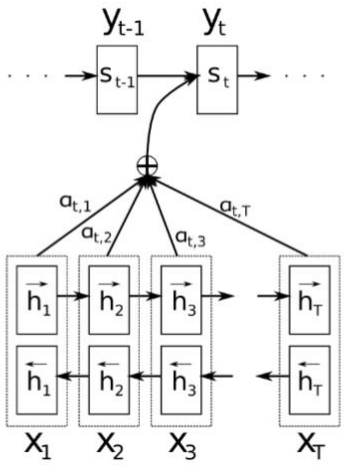



**Figure 2.7.2.1:** Attention layer used in bi-directional network

Show Attend and Tell introduced 2 types of attention mechanisms: Hard and Soft Attention [29].

1. Soft Attention: Also known as deterministic attention is a technique were, we multiply an attention map onto the feature map.

2. Hard Attention: Instead of using a deterministic method, we use a stochastic one. With hard attention, we strictly pay attention and selects only the features identified as required by the attention model.

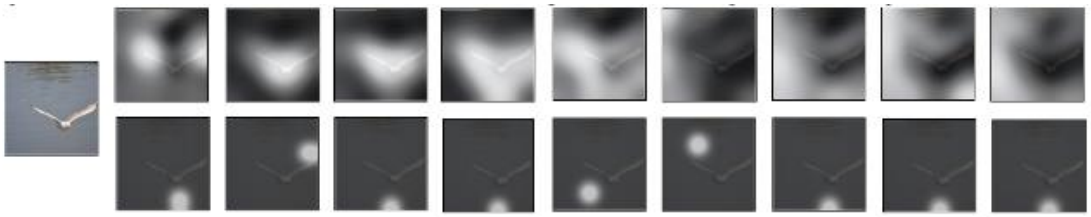


**Figure 2.7.2.2:** Examples of attending to correct object using soft (above) and hard (below) attention [29]

## 2.8 Sequence Prediction

Sequence prediction is the process of predicting the next value in a sequence based on the previous values. This chapter will discuss various sequence predicting mechanisms besides the sequence models such as LSTM's and RNN mentioned in section 2.d.

### 2.8.1 Markov Models

We can use conditional probability distributions to predict the event at $X_n$, based on the previous events, p ($X_n|X_{n-1} \dots X_1$). With the growing value of n, this becomes computationally infeasible [30]. However, we can solve this problem using the Markovian assumption. According to Markov's principle, the probability of an event at a state is only dependent on the immediate predecessor states i.e. for a k-order markov chain, p ($X_n|X_{n-1} \dots X_{n-k}$) [30].

Markov's chain is applicable only in areas where we can directly observe the results. In such scenarios, where only the hidden or indirect values can be observed, we can use Hidden Markov Models. We use Bayes rule to predict events based on the observed hidden values.

$$P(X_1 \dots X_n|h_1 \dots h_n) = P(h_1 \dots h_n|x_1 \dots x_n) \times P(x_1 \dots x_n) / P(h_1 \dots h_n)$$

Markov models are built on strong assumptions that state transitions are dependent only on the current state [30]. Also, we just consider the 'k' prior states in memory and is hence limited in remembering old events. Though we can increase the value of k, it comes at the expense of huge computational complexity. Therefore, LSTMs are preferred over Markov chain methods, especially when we have enough training data [31].

### 2.8.2 Compact Prediction Tree

Though LSTMs, are really good sequence prediction models, they have certain limitations – they need to be retrained for sequences not contained in previous situations. Also, the training time for LSTM's is high.

Compact Prediction Tree (CPT) improves on both these disadvantages of LSTM's [32]. CPT works by creating a prediction tree, an inverted index and a lookup table in the training phase [32]. Prediction tee is a tree of nodes created based on each sequence. An inverted index is a dictionary, with the key an item and values the sequence id's in which they occur. Lookup tables store the last element of the sequence along with the sequence id. An example is shown below. However, since CPT prioritize each state equally, it fails to predict correctly on occurrence of noises in dataset, whereas LSTM's are much more tolerant [31].

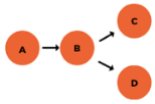
<table>
<tr>
<td>Sequence:<br><br>Seq1: A, B, C<br>Seq2: A, B, D</td>
<td>Prediction Tree<br><br> 
</td>
</tr>
<tr>
<td>Inverted Index<br><br>
<table>
<tr><td>A</td><td>Seq1, Seq2</td></tr>
<tr><td>B</td><td>Seq1, Seq2</td></tr>
<tr><td>C</td><td>Seq1</td></tr>
<tr><td>D</td><td>Seq2</td></tr>
</table></td>
<td>Look-up Table<br><br>
<table>
<tr><td>Seq1</td><td>C</td></tr>
<tr><td>Seq2</td><td>D</td></tr>
</table></td>
</tr>
</table>

**Figure 2.8.2.1:** Working of Compact Prediction Tree

In the prediction phase, we find the sequence which is most similar to the input. For each item in the input, the value in the inverted index is taken and the values with the highest scores are then returned.

# 3 Related Work

Understanding human actions and interaction is one of the ultimate goals of Artificial Intelligence. With the capability to predict actions, computers can serve us in many areas including prevention of vehicle accidents and in surveillance systems. Computer vision research on human actions ranges from simple limb movements to joint movements of the human body [33]. As we saw in section 2.6 action recognition tasks classifies action after processing the entire set of sequences. However, to prevent potential threats, we need to identify things beforehand.

Both action recognition and prediction involve a lot of challenges in detecting the current actions [33]. These include;

1. Intra-inter class variations: Each action could be executed in different ways by different people. Also, each action can be done in a variety of ways, for instance, sprinting and jogging can be classified correctly as running.

2. Cluttered background: Background noise is another concern in action recognition tasks. Most of the models work well in an indoor environment, whereas their performance drops significantly in an uncontrolled environment. Camera motion and change of viewpoint is another concern.

3. Insufficient training data: Deep models such as 3d CNN are on the rise, however, we do not have big datasets of correctly labeled data. In datasets such as YouTube 8-M, annotations are generated by the relative method and hence not very accurate [33].

## 3.1 Action Prediction

Effectively identifying and classifying actions can be done by many action recognition methods [34]. However, though there are various approaches to action prediction, it is still a growing field. Action prediction tasks can be classified into 2; Short-term and Long-term.

Short term tasks involve shorter duration videos that last for seconds. The primary challenge of these tasks is to predict the final action, given a sequence of initial frames. By way of example, if we have frames *1 ... t*, we predict *y: $X_1$ ... t*. For short-term predictions, a bag of words approach is proposed where we average features of a particular level in the same category [35]. Also, approaches focusing on the structure of human movements by Lan et al [36] are also fairly successful. Recently, with the emergence of deep-networks, many deep learning sequence methods such as LSTM's are used in these tasks [33].

Long-term prediction otherwise known as intent prediction is a much more challenging problem. As keeping the entire sequences in memory and establishing relationships is infeasible, most methods consider interactions of objects with actions. One of the earliest intent-prediction methods was the one using and-or graph by Pei et. al [37]. This model represented agent-object interactions using a stochastic context-sensitive grammar and predictions were carried out by finding maximum posterior probability from and-or graphs.

Lie et al. presented a method using a Probabilistic Suffix Tree (PST) which captures the dependencies between actions and other objects [38]. They determine the most likely agent and associated actions along with the interactions between agents and surroundings using a Predictive Accumulation function. The prediction accuracy for this model is over 0.75 on a tennis game dataset created using YouTube by the author [38].

# 4 Method

Action representations such as space-time interest point, holistic representation are used by predictive models such as HMM and CPT. Deep-learning models are preferred over HMM and CPT since they do not require handcrafted features [32]. Handcrafting the features requires significant additional overhead and hence, deep-learning methods are used in this project. As for deep-learning models, models can be improved by increasing the width or depth of models which then will learn more details, provided large enough datasets and sufficient computational resources [33]. However, it is difficult to obtain a huge volume of correctly labeled balanced datasets. Labels for most of the video datasets including YouTube 8M, Sports 1M are generated by the retrieval method and are hence not very accurate [33].

The introduced, Multi-deep-model for Action Prediction (MDAP) method eliminates these problems by trying to design models the way humans perceive actions and events. The ability to understand the gestures and actions of fellow humans is one of the important skills we have. Action recognition by humans is a complex cognitive capability which involves action recognition, intention understanding, and narrative understanding [39]. The problems of action recognition and narrative understanding can be solved using the existing state-of-the-art mechanisms mentioned in section 2. In this project, I have tried to model a prediction mechanism based on human cognitive capabilities and by making use of the existing state of the art mechanisms. This chapter explains the architecture of the system along with each sub-module used in detail.

## 4.1 Architecture

MDAP consist of 4 models which executes simultaneously. The video frames are streamed from the client machine to the server side and are listened by an active listener. The listener, listens to the incoming messages and transfer each frame to both the Object Detector module and the Environment Detector.

Once, enough frames of a particular human are identified, the object detector invokes the action recognizer with the sequence of image frames to perform action recognition. Once, an action is identified, all the outputs generated by the object detector and environment detector are then fed to the sequence prediction module. The schematic diagram of the model is shown below.

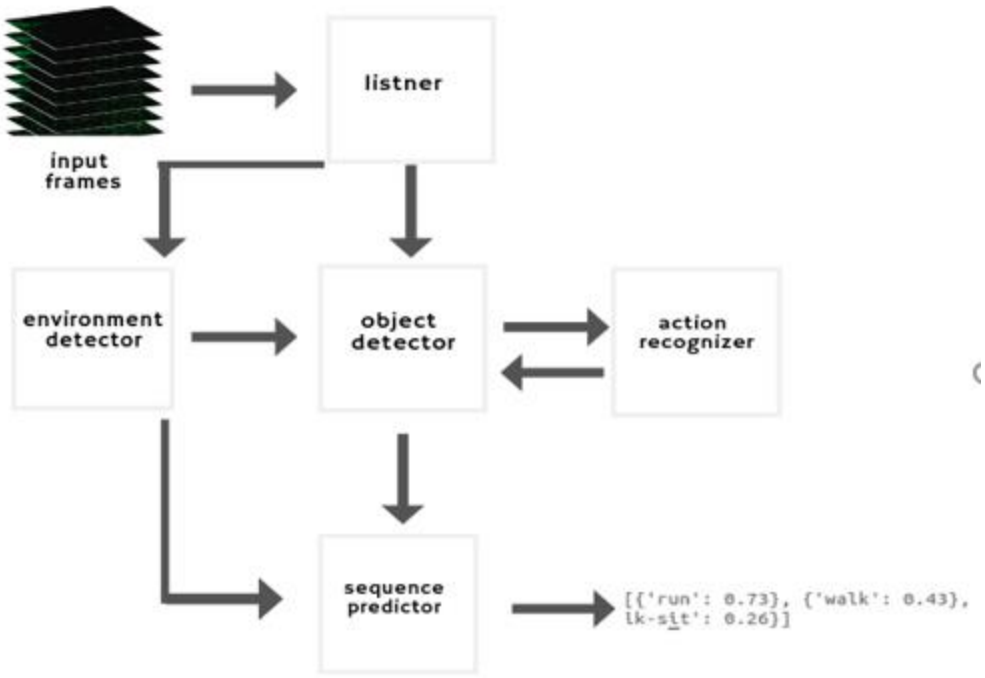



**Figure 4.1.1:** Architecture of the proposed model

## 4.2 Object detector

Once the image frame is received at the server, it is passed on to the object detection module. In the object detection module, we detect objects along with its location within the frame. The object detection is done using a Single Shot Multi-Box Detector (introduced below). A stochastic attention process is then applied to the detected objects in each frame. For example, in a cricket match, we should refine to objects commonly seen in cricket and not detect a car. One of the other major challenges in this object detection module is to keep track of the identified objects. A moving object would change coordinates in each frame, however, the object detector should still view it as the same person and pass to action-recognition to correctly identify the actions.

### 4.2.1 Single Shot Multi-Box Detector

As we saw in section 2.5.2 region based models such as Fast R-CNN or the Faster R-CNN are used by most of the state-of-the-art classifiers in COCO, PASCAL VOC, ImageNet datasets [40]. However, these object detectors are slow with the capability to process 7 frames/sec [40]. In applications where real-time object detection is required, this region based approaches are hence not suitable.

Single Shot Multi-Box Detector (SSD), however, is based on YOLO and is hence a lot faster along with providing results of much higher accuracy compared to YOLO [40]. One of the major drawbacks we discussed in Section 2 for YOLO was its inefficacy to detect small features within the image. SSD overcomes these by producing a prediction of different scales from features of different sizes [40].

The SSD object detector detects the image in a single frame using deep neural networks [40]. They discretize the output space into a set of default bounding boxes and generate a score for each bounding box. Also, unlike YOLO detection technique we saw in Section 2, SSD adjusts the shape of bounding boxes to increase its score. This flexibility, by modifying the dimensions of the bounding boxes enables SSD to detect objects of varied sizes, both larger and much smaller than the box.

At each layer, SSD produces a fixed set of prediction using a set of CNN filters. For a feature layer of m x n with p channel, base element for detection is a very small kernel of size 3 x 3 x P.

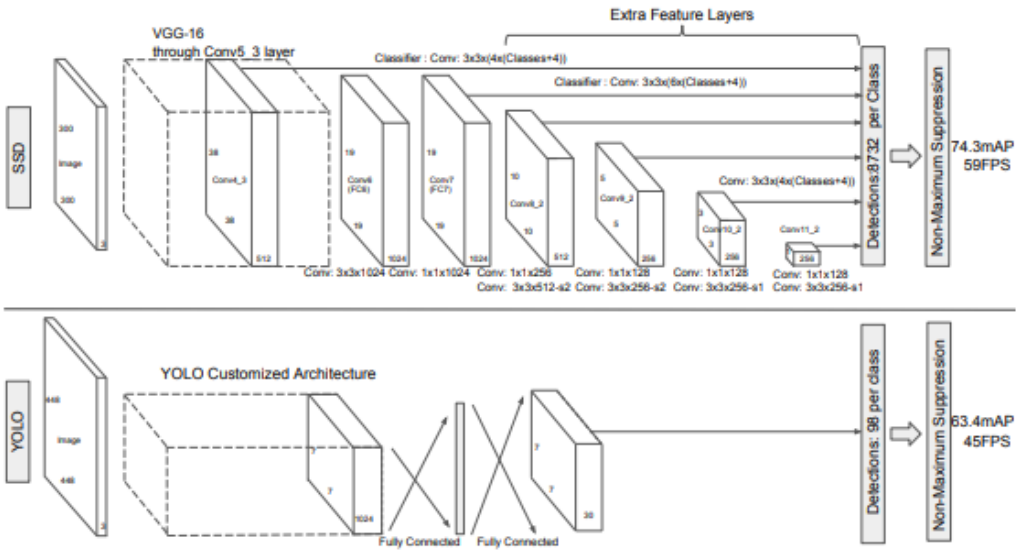



**Figure 4.2.1.1:** Architectural comparison between YOLO and SSD [40]

One of the main challenges in modelling object detectors is during the training phase. In the training phase, we need to determine not just whether an object is been detected, but also where it is located. We match each ground truth with best overlap and select the ones with scores greater than 0.5.

The selected bounding boxes along with their location coordinates are then sent to the sequence prediction model and the action recognition models. Pre-processing of the identified boxes is required before sending to the action recognition model, as the action recognition model expects detected human images to be of size 400x400. If the bounding box we detected is greater than the required size, we trim the excess pixels by maintaining the centroid of the

bounding box. Otherwise, the neighboring pixels are added to bounding box until it is of the required size. Once, action recognition returns the detected action, we feed the results from action recognition model along with the identified object suggested by environment module to the Action Predictor.

## 4.3 Action recognizer

To predict intent, we need to understand what the current action is. Action recognition is the process of identifying or classifying actions in a set of image frames. In this project, we confine to human actions. The humans identified by the object detector iare fed to the action recognition system to detect actions. As mentioned in section 2.6, there are many action recognition mechanisms such as the CNN-LSTM model, 3-d CNN and 2-stream networks. MDAP uses the 3-d CNN model for this task since, the 3-d CNN methods are capable of learning the correlation between 3-d signals in image sequences compared to the CNN-LSTM model [41]. Also, they are simpler and provide comparable accuracy to the 2-stream convolution models in variety of data-sets such as UCF101 and YouTube 8-M [44].

### 4.3.1 Dataset for training the Action Recognizer

One of the primary objectives of this project is the use of a basic subset of human actions instead of using complicated actions during training. The common datasets used in most of the state-of-the-art recognizers recently include YouTube 8-M, Kinetics and Sports 1-M [42]. However, all these datasets focus on providing more complicated classes. For instance, Kinetics dataset provides a distribution of 400 classes with complex actions such as walking with a horse and wrapping present [42]. As MDAP focuses on finding relationships with simple actions and surrounding objects, all the mentioned datasets are complicated. Also, for action-recognition the commonly used datasets such as HMDB51 and UCF101 are too small to train a 3-d network without overfitting [42].

The I3-D Post Multi-View Human Action Dataset provides 24 videos for 12 basic human actions; Walk, run, jump forward, jump in place, bend, one hand wave, sit down - stand up, walk - sit down, run - fall, run - jump - walk, two persons handshaking, one person pulls another each with different actors [43]. Providing a refined category of simple actions with multi-angle camera views is ideal for the MDAP project. Also, the dataset is created in a controlled, noise free environment in addition to being much smaller than ucf101. These factors make I3-D dataset an ideal choice and is hence used in training the action recognizer.

### 4.3.2 3-D CNN Model

As we saw in section 2.6, 3-d CNN uses 3-D convolutions instead of the 2-D kernels used in tradition convNets. The 3-D convolutions are created by stacking up the sequential image frames. These convolutions extract spatio-temporal sequences directly from the videos and are

hence very effective [44]. Tran et al. in their research to extract spatio-temporal features from videos found convolution filters of 3x3x3 to achieve the best performance [47].

As we saw in section 2.5.1, ResNet is one of the most popular and effective techniques for feature extraction [22]. Since ResNet bypasses connections through all layers, they can be used for training very deep networks. A basic ResNet consists of 2 convolutional layers each followed by batch normalization and ReLU activation [44].

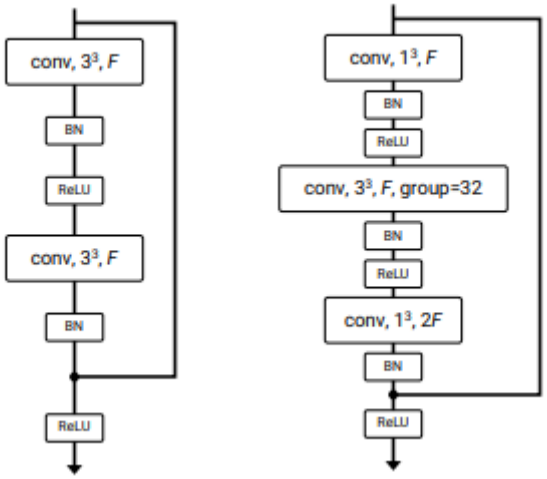



**Figure 4.3.2.1:** Architecture of basic ResNet (left) and ResNeXT (right)

Improved ResNet models such as ResNeXt consisting of 3 convolutional networks along with dividing feature map into small groups provides higher accuracies compared to the vanilla ResNet model [44]. Also, increasing the number of convolutional layer group in middle convolutional network results in higher accuracy [44], hence, 32 groups were used in the middle layer.

The action recognition model was trained with stochastic gradient descent with momentum. The input images are assumed to be of size 400x400, the object-detection ensures that all images sent to the action recognizer have the required size. Each video is then split into 16 frames; hence the total size of each training input would be 16 x 400 x 400 x 3. The action recognition labels the incoming frames to the identified action, which is then returned to the object detection module.

## 4.3 Environment Detector

Unlike the previously seen Object detector and the Action recognizer the environment detection module has 2 tasks. The first is to detect the scene or the environment in the image frame and the second is to provide probabilistic distribution for various actions and objects in the identified scene.

### 4.3.1 Scene Identifier

This sub-module is responsible for identifying the scene in the video frame. For instance, if the video sequence is a football match, the model identifies the scene as a football match. This can be accomplished by modeling an image classifier using any of the state-of-the-art CNN's such as VGG-16, Inception or a ResNet [19].

Though scene understand seems like a trivial problem, the usage of an appropriately balanced dataset is quintessential. The Places205 [45] or the improved Places365 [46] from MIT Labs is perfect for scene understanding [45]. In this project, we have used the Places365 datasets trained on a VGG-16 convNet since VGG-16 achieved the highest accuracy for classification in the ECCV 2016 Place-365 Challenge [50]. The sub-module returns the probability of each identified scene and also returns whether it is indoor or outdoor.

| | Validation Set of Places365 | | Test Set of Places365 | |
|---|---|---|---|---|
| | Top-1 acc. | Top-5 acc. | Top-1 acc. | Top-5 acc. |
| Places365-AlexNet | 53.17% | 82.89% | 53.31% | 82.75% |
| Places365-GoogLeNet | 53.63% | 83.88% | 53.59% | 84.01% |
| Places365-VGG | **55.24%** | 84.91% | **55.19%** | 85.01% |
| Places365-ResNet | 54.74% | **85.08%** | 54.65% | **85.07%** |

**Figure 4.3.1.1:** Accuracy comparison of various convNet models in Places-365 [50]

### 4.3.2 Probability Generator

The second task of the environment detector module is to provide a probabilistic distribution of various objects and actions for a particular scene. By way of example, if a scene is identified as football, we would get a probability distribution with higher values for football and goal post and much lower value for say, a car.

This probability distribution is generated using the data from training set. Each time an object is found in a frame, its counter is increased and the probability is found by the total number of times each object is seen and the total number of frames processed for each scene. The attention mechanisms used in this project solely depends on the probability values returned by this module. The object detector uses a stochastic/hard attention as stated in section 4.2 and the sequence prediction uses the soft attention method.

The occurrence of each object within a scene is stored in a map, with the identified scene as the key and the object and its occurrence the values. A separate training process with the training dataset is required for this environment detector and it should be done prior to the training on sequence model. This is primarily because we need the environment detector to provide values for the attention layer during the training phase of the action predictor. During the training phase of the Action predictor, this module provides probability distribution of various objects

within the identified scene. This result provided is used by the object detector to prioritize and select only the relevant objects in a scene and by the action predictor to prioritize certain actions based on the environment.

## 4.4 Action predictor

The primary purpose of the Multi-Deep models for Action Prediction (MDAP) system is to predict the next action in a sequence. Though there are many state-of-the-art action recognition systems, predicting the next action is a tough problem to solve. As we saw in section 3.1, the earlier models by Lie et. al [38] and Pei et. al [37] uses handcrafted features for identifying and predicting actions. However, with deep networks we can eliminate this usage. The recent prediction models such as Encoding Temporal Evolution for Real-time Action Prediction by Fahimeh et. al. represents frames as Dynamic Images based on rank pooling. These Dynamic Images focuses on certain actions and objects and pads the rest of the information in image [48]. The prediction of the next actions is then done by a LSTM model trained on the Dynamic Image sequences. However, Dynamic Images consist of many insignificant values which hinders the model's capability to learn. MDAP tries to combine the best of the earlier models and the recent ones. We use deep-models to detect and identify the object-action relationship and then only the fine-grained result to train and predict the next action.

As we saw in section 2.8, variants of Markov chain and Compact Prediction Trees can be used for sequence prediction. However, these models suffer from the vanishing and exploding gradient problem we discussed in section 2.3. Hence, we use the Long Short-term Memory (LSTM) networks since LSTM's can learn long-term dependencies and does not suffer from the vanishing gradient problem [31].

The network is trained with the sequence of information from the object detector and the environment detector. The object detector feeds in the detected action along with close by object and the Euclidean distance between them. For efficient learning of the sequence, only a single-pair of object-action is fed at a time. During the training phase, if multiple object-action pairs are detected by the earlier models, it is split into multiple fractions of object-action pairs and is fed to the model.

Given the sequential data, LSTM or its variants can learn the temporal dependencies within the sequence. One of the shortcomings of LSTM's is they are only able to make use of the previous information [49]. With Bidirectional LSTMS, BDLSTM, we can process data in both directions. BDLSTM connects 2 hidden layers, one in forward direction and the other in backward direction to the same output [49]. The output of a BDLSTM cell is given by;

$$\sigma\left(\underset{h}{\rightarrow}, \underset{h}{\leftarrow}\right)$$

where $\overleftarrow{h}$ is calculated using the sequence in backward direction and $\overrightarrow{h}$ in forward direction.

Since, BDLSTM allows learning based on the data in both directions, we use a BDLSTM for our sequence prediction tasks. The hyper parameters used by the BDLSTM models includes Mean-Squared-Loss error for training and L-BFGS optimizer.

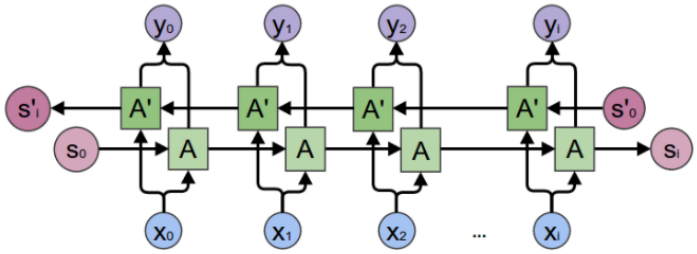



**Figure 4.4.1:** A bi-directional LSTM network

Attention mechanism helps our network to focus on certain parts of data. To confine our predictions, we use a deterministic or soft attention model. The calculated attention weights are multiplied with the results of each LSTM cell to create a weighted combination. The attention weights provided by the environment detector is the probability of the action in the identified scene.

# 5 Results

This chapter will discuss the results achieved in this project. Results achieved for each of the sub-modules along with the final results are discussed in detail.

## 5.1 Dataset

One of the major challenges in this approach is the lack of dataset to evaluate the result. Object detection, action-recognition, and prediction can all be verified with pre-existing datasets. However, the entire combination of the sub-modules cannot be verified on any of the existing datasets, as long sequences of video labelled dataset for each frame are currently not available. For correctly evaluating the model, we need a video dataset with long videos with detailed labelling for each frame. For instance, if a video is of people going towards and sitting in chairs, we need the label for each frame to include all objects seen, the action of each person involved.

For these reasons, all the sub-modules are evaluated with the existing datasets and the combination by a newly created labelled dataset, the Basic Action Set (BAS). The new dataset, BAS, involves multiple actions such as walking and sitting, running and kicking, continuously running, running and jumping, walking and crawling. Each frame in this dataset is labelled in detail.

## 5.2 Object Detection

Object Detection module was trained using the MS-COCO dataset. Though faster region based methods such as fast R-CNN and faster R-CNN gave better accuracy, a single shot multi-box detector model was used for performance reasons. The SSD is nearly 6 times faster than the region based models [31]. The model was evaluated on the MS-COCO dataset and produced a mean average precision of 0.32.

In MDAP, the objective of the object detection module is to identify each object, localize them and to pass the identified humans uniquely to the action-recognition model. The below are some of the results produced by the object detection module.

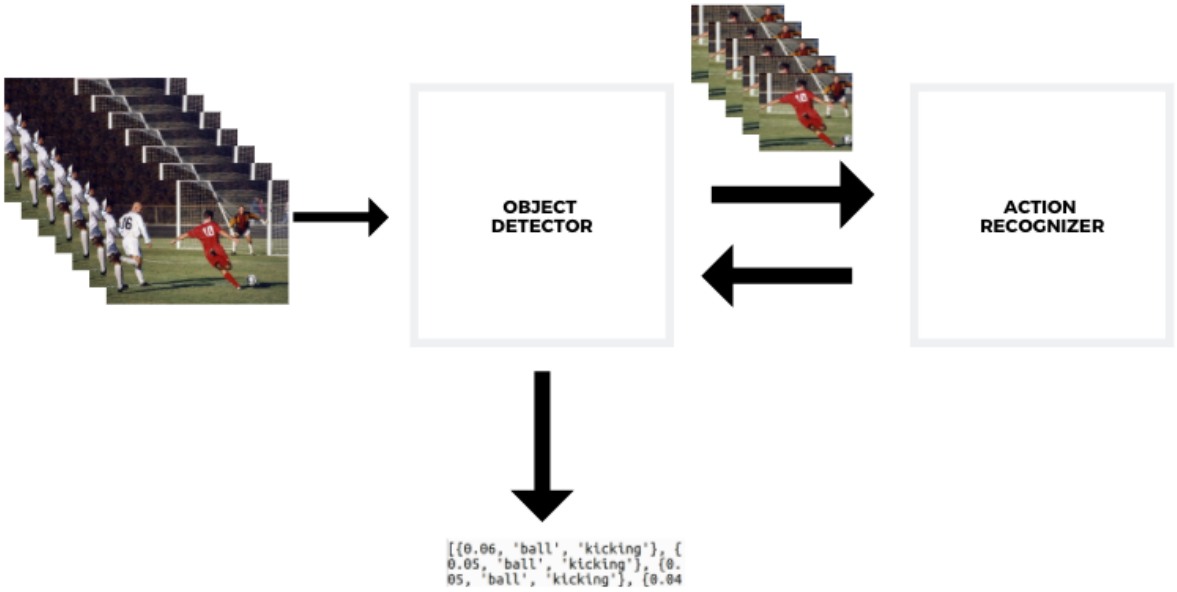



**Figure 5.2.1:** An example output generated by object detector.

## 5.3 Action Recognition

Action recognition is done using the 3-d CNN model trained on the i3-D post-multi-view action dataset and also on the UCF101 dataset. The initial training and evaluation were done on the ucf101 dataset by directly using the publicly available pre-trained models. However, with the progression of the project, the i3-D dataset was used and the model produced an accuracy of 0.78. The results obtained by our 3-d CNN model are given below.

| Dataset | Accuracy |
|---|---|
| UCF101 | 0.74 |
| I3-D Post Multi-view | 0.81 |

**Table 5.3.1:** The accuracies obtained by the Action recognition model

One of the key elements of the project, this action recognition module produces a probability of the input frames belonging to each action classes. All the inputs given to the recognition module are a sequence of 10 frames of the same human identified by the object detector. An example of this module returning the probability of each action class to the object detector module is shown below.

## 5.4 Environment Detection

The primary goal of environment detection is to classify the image to various scenes. Since the pre-trained weights of the scene-365 model trained on vGG-16 are publicly available, I could directly use it. The environment detector gave an accuracy of 0.85 on this pre-trained network in the scene-365 dataset.

The other task of the environment detector is to keep track of each object and action encountered for a particular scene. During the training phase, a counter is maintained as a map for each of the scene-object pair. For each matching occurrence, the identified entry in the map is updated. And in the evaluation phase, object detector and sequence predictor is fed with the probability distribution for each object and action in the identified scene.


**Figure 5.4.1:** The output generated by scene detector for the shown input.

## 5.5 Action Prediction

The action prediction along with the attention mechanism is trained and tested on the output of the object detector. The object detection outputs an object-action relationship for each object-action pair identified. This is then stored in a file in .pt format and used while training the sequence prediction model. Each of the object-action pairs is then extracted from the file and is used to train the model. With a separate test-dataset, the model produced an accuracy of 82% during the testing phase in the newly created dataset. Also, the model produced an accuracy of 64% on the UCF101 dataset. However, the verification of prediction in ucf101 is done based on the actions recognizer by the action recognizer as labelled frames are not available in ucf101. An example of the final result produced by the model is shown below.

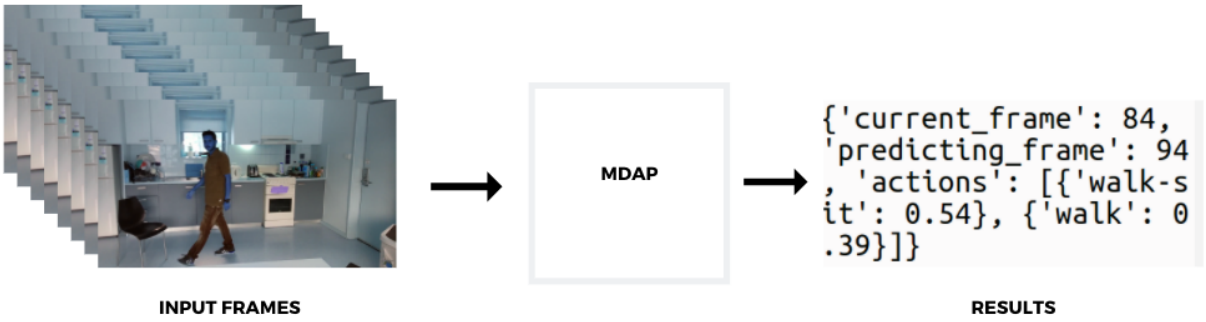



**Figure 5.5.1:** A result generated by model on the BAS data set.

# 6 Limitations

Though this model is capable of working with a very limited set of labelled video data, it has certain limitations.

1. No 3-D Distance Computed: In MDAP, we are just computing the 2-d Euclidean distance between various objects. As no 3-d distance is computed, we are not able to utilize the three-dimensionality of videos. One of the main problems with this is, even if say, a chair is lying behind and a person walks across, the system could predict that the next action is sitting.

**Figure 6.1:** Only 2-D distance is computed. The chair and oven is considered to be equidistant through in reality oven is behind and much farther the person walking.

2. Works only with single-viewpoint dataset: MDAP computes the relationship between objects and looks for a sequential connection between objects in frames. However, this approach will be erroneous if the viewpoint of the video changes. As in, if a person is walking to a chair and the video zooms out, the distance between the objects dramatically increases.

# 7 Future Work

The proposed model, Multi-deep-model for Action Prediction (MDAP), is capable of predicting actions with limited labelled training data. With the advancements in computer vision techniques handling the aforementioned limitations would certainly be possible. For instance, we could address the limitation of changes in viewpoint by methods such as the one by Daniel et. al. [51].

Besides, MDAP focuses on predicting only human actions, it can be extended for predicting the movement of other objects such as cars. The ability to predict changing lanes of the cars in front would be beneficial in an autonomous driving environment.

Overall, MDAP meets its purpose of predicting actions with very limited labelled data set and can be used as a base method for further research in the sequence prediction domain.

# 8 Conclusion

The Multi-deep-model for Action Prediction (MDAP) is a novel approach requiring only limited labelled data to be trained on. Instead of the common approaches where we train with huge distributed datasets, MDAP uses only a small labelled action dataset along with unlabelled dataset for further training.

MDAP uses four different models which execute in parallel along with usage of both stochastic and deterministic attention. The video frames are initially processed to compute the scene using an environment detector. Once the scene is detected, relevant objects are extracted using the object detection and actions are computed for the identified humans. These inputs are further fed to the sequence predictor to predict the upcoming actions.

Though MDAP is an attractive model for predicting future actions in a dataset, it has certain limitations including not addressing the 3 dimensionalities and video with multi-views. Also, the model is slow compared to the progress of the video. However, these could be avoided through further research in the area.

# 8 References


[1] Gandhi, T., & Trivedi, M. M. (2007). Pedestrian protection systems: Issues, survey, and challenges. *IEEE Transactions on intelligent Transportation systems*, *8*(3), 413-430.\

[2] Polkinghorne, D. E. (1988). *Narrative knowing and the human sciences*. Suny Press.

[3] Fahlman, S. E. (1988). An empirical study of learning speed in back-propagation networks.

[4] Wang, Z., He, K., Fu, Y., Feng, R., Jiang, Y. and Xue, X. (2017). Multi-task Deep Neural Network for Joint Face Recognition and Facial Attribute Prediction. *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval - ICMR '17*.

[5] Sathyanarayana, Shashi. (2014). A Gentle Introduction to Backpropagation. Numeric Insight, Inc. Whitepaper.

[6] Neural networks and backpropagation explained in a simple way. (2018). Retrieved from https://medium.com/datathings/neural-networks-and-backpropagation-explained-in-a-simple-way-f540a3611f5e

[7] CS231n, S. (2017). Convolutional neural networks for visual recognition.

[8] Ji, S., Xu, W., Yang, M., & Yu, K. (2013). 3D convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, *35*(1), 221-231.

[9] Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*.

[10] Olah, C. (2015). Understanding lstm networks, 2015. *URL http://colah. github. io/posts/2015-08-Understanding-LSTMs*.

[11] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

[12] Potdar, K., Pardawala, T. S., & Pai, C. D. (2017). A comparative study of categorical variable encoding techniques for neural network classifiers. *International Journal of Computer Applications*, *175*(4), 7-9.

[13] Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Hasan, M., Van Esesn, B. C., ... & Asari, V. K. (2018). The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches. *arXiv preprint arXiv:1803.01164*.

[14] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

[15] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[16] A Simple Guide to the Versions of the Inception Network. (2018). Retrieved from https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202

[17] An Overview of ResNet and its Variants – Towards Data Science. (2018). Retrieved from https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035

[18] What do we learn from single shot object detectors (SSD, YOLOv3), FPN & Focal loss (RetinaNet)?. (2018). Retrieved from https://medium.com/@jonathan_hui/what-do-we-learn-from-single-shot-object-detectors-ssd-yolo-fpn-focal-loss-3888677c5f4d

[19] Dai, J., Li, Y., He, K., & Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems* (pp. 379-387).

[20] Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440-1448).

[21] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91-99).

[22] Qiu, Z., Yao, T., & Mei, T. (2017, October). Learning spatio-temporal representation with pseudo-3d residual networks. In *2017 IEEE International Conference on Computer Vision (ICCV)* (pp. 5534-5542). IEEE.

[23] Carreira, J., & Zisserman, A. (2017, July). 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.

[24] Ullah, A., Ahmad, J., Muhammad, K., Sajjad, M., & Baik, S. W. (2018). Action recognition in video sequences using deep Bi-directional LSTM with CNN features. *IEEE Access*, *6*, 1155-1166.

[25] Simonyan, K., & Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems* (pp. 568-576).

[26] Rensink, R. A. (2001). Change blindness: Implications for the nature of visual attention. In *Vision and attention* (pp. 169-188). Springer, New York, NY.

[27] Luong, M. T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

[28] Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3156-3164).

[29] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., ... & Bengio, Y. (2015, June). Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning* (pp. 2048-2057).

[30] Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257-286.

[31] Panzner, M., & Cimiano, P. (2016, August). Comparing hidden markov models and long short term memory neural networks for learning action representations. In *International Workshop on Machine Learning, Optimization and Big Data* (pp. 94-105). Springer, Cham.

[32] Gueniche, T., Fournier-Viger, P., & Tseng, V. S. (2013, December). Compact prediction tree: A lossless model for accurate sequence prediction. In *International Conference on Advanced Data Mining and Applications* (pp. 177-188). Springer, Berlin, Heidelberg.

[33] Kong, Y., & Fu, Y. (2018). Human Action Recognition and Prediction: A Survey. *arXiv preprint arXiv:1806.11230*.

[34] Yao, B., & Fei-Fei, L. (2012). Recognizing human-object interactions in still images by modeling the mutual context of objects and human poses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *34*(9), 1691-1703.

[35] Ryoo, M. S. (2011, November). Human activity prediction: Early recognition of ongoing activities from streaming videos. In *Computer Vision (ICCV), 2011 IEEE International Conference on* (pp. 1036-1043). IEEE.

[36] Lan, T., Chen, T. C., & Savarese, S. (2014, September). A hierarchical representation for future action prediction. In *European Conference on Computer Vision* (pp. 689-704). Springer, Cham.

[37] Pei, M., Jia, Y., & Zhu, S. C. (2011, November). Parsing video events with goal inference and intent prediction. In *Computer vision (iccv), 2011 ieee international conference on* (pp. 487-494). IEEE.

[38] Li, K., Hu, J., & Fu, Y. (2012, October). Modeling complex temporal composition of actionlets for activity prediction. In *European conference on computer vision* (pp. 286-299). Springer, Berlin, Heidelberg.

[39] Keestra, M. (2015). Understanding human action. Integrating meanings, mechanisms, causes, and contexts.

[40] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). Ssd: Single shot multibox detector. In *European conference on computer vision* (pp. 21-37). Springer, Cham.

[41] Liu, H., Tu, J., & Liu, M. (2017). Two-Stream 3D Convolutional Neural Network for Skeleton-Based Action Recognition. *arXiv preprint arXiv:1705.08106*.

[42] Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., ... & Suleyman, M. (2017). The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*.

[43] Starck, J., & Hilton, A. (2007). Surface capture for performance-based animation. *IEEE computer graphics and applications*, *27*(3).

[44] Hara, K., Kataoka, H., & Satoh, Y. (2018, June). Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and ImageNet. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA* (pp. 18-22).

[45] Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., & Oliva, A. (2014). Learning deep features for scene recognition using places database. In *Advances in neural information processing systems* (pp. 487-495).

[46] Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., & Torralba, A. (2018). Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, *40*(6), 1452-1464.

[47] Liu, K., Liu, W., Gan, C., Tan, M., & Ma, H. (2018). T-C3D: Temporal Convolutional 3D Network for Real-Time Action Recognition. In *AAAI*.

[48] Rezazadegan, F., Shirazi, S., Baktashmotlagh, M., & Davis, L. S. (2018). On encoding temporal evolution for real-time action prediction. *International Journal of Computer Vision*.

[49] Cui, Z., Ke, R., & Wang, Y. (2016). Deep Stacked Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction. In *6th International Workshop on Urban Computing (UrbComp 2017)*.

[50] Zhou, B. (2018). Places: A 10 million Image Database for Scene Recognition. Retrieved from http://places2.csail.mit.edu/

[51] Glasner, D., Galun, M., Alpert, S., Basri, R., & Shakhnarovich, G. (2011, November). aware object detection and pose estimation. In *Computer Vision (ICCV), 2011 IEEE International Conference on* (pp. 1275-1282). IEEE.