

# OBJECT TRACKING USING CONVOLUTIONAL AND RECURRENT NEURAL NETWORK

by

Mayuri Deshpande

A thesis submitted to the faculty of  
The University of North Carolina at Charlotte  
in partial fulfillment of the requirements  
for the degree of Master of Science in  
Computer Science

Charlotte

2018

Approved by:

---

Dr. Hamed Tabkhi

---

Dr. Min Shin

---

Dr. Samira Shaikh

---

Dr. Minwoo Lee



## ABSTRACT

MAYURI DESHPANDE. Object Tracking using Convolutional and Recurrent Neural Network. (Under the direction of DR. HAMED TABKHI)

In this master thesis, recurrent neural network based method for visual tracking in videos is introduced that learns to predict the bounding box locations of a target object at every frame. Region information and distinctive visual features are obtained from applying Convolutional Neural Network on each of the frames in the video. Our Recurrent Neural Network (LSTM) exploits these history of locations along with the high level visual features learned by the deep neural networks. In order to increase the tracking accuracy and reduce the computation cost, a novel approach is proposed to construct a larger LSTM Network which we call it as Sparsely stacked LSTM (S2LSTM). The promise of S2LSTM is to offer a systematic solution to scale LSTM networks capture longer and more complex sequences, compared to mainstream LSTM design. S2LSTM is scalable and contains discrete non-overlapping training stacks, offering a modular design for building complex LSTM networks. S2LSTM offers a discrete training mechanism which significantly helps to grow the complexity without retaining the next network. The key significance of S2LSTM is adding a time pooling module across stacked LSTM layers. It reduces the number of time steps propagating from first LSTM to the second LSTM by filtering out the "Intermediate Outputs" across the stacked layers. In S2LSTM, the output of each stack LSTM is compared with respective ground truth and are trained as separate paradigms. At the same time, it is less computationally Intensive compared to regular stacked LSTM. Our experiment on video data demonstrates that S2LSTM increases the tracking overlap accuracy by 15% compared to baseline ROLO implementation.

## ACKNOWLEDGEMENTS

I sincerely thank my advisor, Dr. Hamed Tabkhi, for giving me an opportunity to work with him on a great idea like this and for all his mentoring, support, and patience during my research. He is a great mentor and continuously steered me in the right direction.

A special thanks to my lab mates Vikas Ramachandra for his consistent support, guidance, and contribution to this research. A big thanks to UNC Charlotte for giving me an opportunity to present this research.

Lastly, I sincerely express my profound gratitude to my parents and my brother for their support and encouragement, without whom this accomplishment would not have been possible.

## DEDICATION

Dedicated to my dearest Parents, Sayali Deshpande and Vijay Deshpande.

## TABLE OF CONTENTS

LIST OF FIGURES	viii
LIST OF TABLES	x
LIST OF ABBREVIATIONS	1
CHAPTER 1: INTRODUCTION	1
1.1. Motivation	1
1.2. Problem Statement	2
1.3. Contributions	2
1.3.1. Sparsely Stacked LSTM (S2LSTM)	2
1.3.2. Data Classification based on Relative Motion between Camera and Object	3
CHAPTER 2: RELATED WORK	4
2.0.1. Spatially Supervised Recurrent Convolutional Neural Networks for Visual Object Tracking	5
2.0.2. Re3: Real-Time Recurrent Regression Networks for Vi- sual Tracking of Generic Objects	5
CHAPTER 3: BACKGROUND	8
3.1. Convolutional Neural Network	8
3.1.1. The LeNet Architecture (1990s)	8
3.2. Recurrent Neural Network	10
3.2.1. Long Short Term Memory Units(LSTMs)	11
3.2.2. Stacked LSTM	17

CHAPTER 4: PROPOSED APPROACH	19
4.1. Sparsely Stacked LSTMs	19
4.1.1. Baseline Implementation of ROLO(YOLO+LSTM)	19
4.1.2. Stacked LSTM Implementation (YOLO+Stacked LSTM)	23
4.1.3. Sparsely Stacked LSTM Implementation (YOLO+S2LSTM)	24
4.2. Data Classification based on Complexity of Scenarios in Dataset	26
4.2.1. Dataset	27
CHAPTER 5: EXPERIMENTAL RESULTS	30
5.1. Target Dataset	30
5.2. Experimental Setup	30
5.2.1. Software Setup	30
5.2.2. Hardware Setup	30
5.3. Tracking Performance Evaluation	31
5.4. Experimental Result Analysis	32
5.4.1. Quantitative Results	33
5.4.2. Qualitative Results	35
CHAPTER 6: CONCLUSIONS	39
6.1. Summary of Findings	39
6.2. Future Work	40
REFERENCES	41

## LIST OF FIGURES

FIGURE 2.1: Structure of ROLO. (From Guanghan Ning, Zhi Zhang, Chen Huang, Zhihai He, 2016)	6
FIGURE 2.2: Structure of $Re^3$ Networks. (From Daniel Gordon, Ali Farhadi, and Dieter Fox, 2017)	7
FIGURE 3.1: LeNet neural network structure. (From LeCun, Yann and Bottou 1998)	8
FIGURE 3.2: recurrent neural network and the unfolding in time of the computation involved in its forward computation	10
FIGURE 3.3: The vanishing gradient problem for RNN. The shading of the nodes in the unfolded network shows their sensitivity to the inputs at time one (the darker the shade, the greater the sensitivity). The sensitivity decrease over time as new inputs overwrite the activations of the hidden layer, and the network 'forgets' the first inputs. (From Alex Graves, 2012)[36]	12
FIGURE 3.4: LSTM memory cell, (From Kyunghyun Cho Pierre Luc Carrier, 2017)	13
FIGURE 3.5: Preservation of gradient information by LSTM.(From Alex Graves, 2012)	14
FIGURE 3.6: The repeating module in a standard RNN contains a single layer (From Christopher Olah, 2015)	14
FIGURE 3.7: The repeating module in an LSTM contains four interacting layers (From Christopher Olah, 2015)	15
FIGURE 3.8: : Information about different parts of LSTM diagram (From Christopher Olah, 2015)	15
FIGURE 3.9: cell state in the LSTM,(From Christopher Olah, 2015)	16
FIGURE 3.10: Gate in the LSTM, (From Christopher Olah, 2015)	16
FIGURE 3.11: LSTM Stacking	18
FIGURE 4.1: YOLO Model	20



FIGURE 4.2: YOLO Architecture. (From Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhad, 2015)	21
FIGURE 4.3: Proposed Network	22
FIGURE 4.4: Average IOU scores and fps over various number of steps (From Guanghan Ning, Zhi Zhang, Chen Huang, Zhihai He, 2016)	23
FIGURE 4.5: Stacked LSTMs Vs. Sparse Stacked LSTMs	24
FIGURE 4.6: Sparse Stacking of LSTM's	26
FIGURE 4.7: Sparse Stacked LSTM on ROLO	28
FIGURE 4.8: First frame of the target object for sequences that used in this project	28
FIGURE 5.1: Intersection over Union Equation	32
FIGURE 5.2: Average IOU score for all videos and classified videos based on Relative Motion	34
FIGURE 5.3: 6 continuous sequences in Skater2 Video	36
FIGURE 5.4: 6 continuous sequences in Human2 Video	37
FIGURE 5.5: 6 continuous sequences in Blur Car Video	38
FIGURE 5.6: 6 continuous sequences in Skating1 Video	38
FIGURE 5.7: 6 continuous sequences in Transformer Video	38

## LIST OF TABLES

TABLE 4.1: Input and Output Time steps for LSTM1 and Sparse Stacked LSTM Prediction	27
TABLE 4.2: Challenging aspects of Object Tracking in dataset	27
TABLE 5.1: Summary of Average Overlap Scores (AOS) results for all 15 test videos	33
TABLE 5.2: Training and Inference time for the trackers	35

## CHAPTER 1: INTRODUCTION

### 1.1 Motivation

Object tracking is an important task within the field of computer vision. With the emergence of GPUs with tremendous computing, the availability of high quality and inexpensive video cameras, and the increasing need for automated video analysis has generated an enormous interest in object tracking algorithms. Hence, It has become an essential component in modern applications such as robotics, video surveillance, self driving cars, human-computer interaction, etc. There are three key steps in video analysis: detection of moving objects, tracking of such objects from frame to frame, and analysis of object tracks to recognize their behavior. There is a lot of research going on in Object Detection and tracking problems with the help of computer vision and machine learning approaches. The main challenge in Object detection occurs due to variety of reasons such as target deformations, illumination variations, scale changes, fast and abrupt motion, occlusions, motion blur and background clutters. With booming deep learning, these challenges can be overcome. Convolutional Neural Networks (CNNs)[1] have demonstrated near-human or beyond-human object detection/classification accuracy. However they don't offer a scalable solution for tracking of objects over large videos. LSTM's have proven to be formidable for Object Detection problem[2] and many advancements[3] are being made to effectively utilize LSTM configurations to increase tracking accuracy[4]. However a lot of questions like "Do these state-of-the-art methods work well?", "What is the performance of these state-of-the-art methods", "Is it computationally burdening", "What is the best approach for tracking objects?", "Are Deep Learning algorithms the correct way to approach this problem?", "What are the potential problems that are likely to arise?"

which do not have a concrete answer yet. This thesis aims to find some answers by exploring two specific Deep Learning approaches Convolutional Neural Networks (CNNs or ConvNets) and Recurrent Neural Networks (RNNs), how they perform on this particular task of object detection and tracking and what are the potential areas of improvement.

## 1.2 Problem Statement

There has been a reasonable amount of research done in Object Tracking using CNNs and RNNs[1, 2]. However there has not been much of research to increase the performance of tracking while keeping the application computationally inexpensive. In order to improve performance of LSTMs and in turn the tracker, number of hidden layers are increased or LSTM's are stacked. But stacking LSTM[5] in case of Object Detection is computational and logically complex and is infeasible to both train and inference. Another challenge faced while object tracking is the tracking environment conditions such as illumination and scale variation, Occlusion, Deformation, motion blur, fast motion, In-plane rotation, out-of plane rotation, out-of view, background clutters and low resolution. Designing a generic tracking algorithm for above conditions has been a subject for research.

## 1.3 Contributions

### 1.3.1 Sparsely Stacked LSTM (S2LSTM)

One of the primary goals of the thesis is to present a computationally inexpensive and High performance tracking algorithm where the LSTMs are sparsely stacked. This configuration is feasible to both train and inference compared to a fully stacked LSTM[6] configuration with an improved tracking performance and a reasonable computation complexity. This approach is called as Sparsely Stacked LSTM (S2LSTM). This configuration is an extension of Stacked LSTM configuration [7] which has a time pooling concept. In this time pooling concept, the most significant output predictions

from the first layer of LSTM is fed to the next layer of LSTM. Each of the LSTM predictions are individually compared to the ground-truth and back-propagation for error reduction is done.

### 1.3.2 Data Classification based on Relative Motion between Camera and Object

This research is done on raw video frames from a standard dataset of Object Tracking Benchmark(OTB)[8] and VOT2017 [9] which is a very famous computer vision challenge. The OTB benchmark contains 100 sequences with 11 attributes, which represents the challenging aspects in visual tracking such as illumination and scale variation, Occlusion, Deformation, motion blur, fast motion, In-plane rotation, out-of plane rotation, out-of view, background clutters and low resolution. A systematic analysis of each category of the video is carried out and a comparison study is presented based on how LSTM hyper-parameters like number of time step is related to the object and camera motion and how it affects the performance of the tracker.

## CHAPTER 2: RELATED WORK

As object tracking is excessively dependent on object motion as well as the camera motion and in turn viewpoint change, pose and scale variations, motion blur and so on, traditional tracking methods with fixed models of a target prior to the start of tracking task normally fail because of the inevitable appearance changes. Therefore, to handle these variations effectively, adaptive methods have been proposed to update the representation of a target incrementally over time. That is to say, new models are learned online. Most trackers can be divided into two sub-models. The state-of-the-art single-target visual tracking methods with online updating schemes can be classified into generative and discriminative online learning methods. Traditional generative online learning methods are adopted to track an object by searching for regions most similar to the target model. The online learning strategy is embedded in the tracking framework to update the appearance model of the target adaptively in response to appearance variations. Discriminative online learning methods, treat object tracking as a classification problem and utilize information of the target and background simultaneously. A binary classifier is trained to distinguish the target from the background and is updated online to handle appearance and environmental changes[10]. The modern approach of deep learning for Object Tracking make use of Convolutional Neural Network to classify and detect the location of the object frame by frame, and Recurrent Neural Network to track and predict the object location in future frame[11, 12, 13]. This problem to predict the location at next time step which is dependent on the previous locations is a sequential problem and LSTMs have proven to be suitable and effective since ROLO[12]. Still highest accuracy achieved until now is XX and there is clearly room for improvement. Apart from LSTMs, in

TUBE-CNN[14], CNN[1] is used for tracking by feeding continuous frames as a single input. It feeds all the sequential frames object coordinates combined as a single input to CNN model and predicts the coordinates. The papers which were read during literature survey use deep learning approach and have been summarized below.

#### 2.0.1 Spatially Supervised Recurrent Convolutional Neural Networks for Visual Object Tracking

This method [12] ROLO(recurrent YOLO) uses YOLO[15] on the frames of videos to collect rich and robust visual features as well as preliminary location inferences; and Then LSTM network is used in the next stage as it is spatially deep and appropriate for sequence processing. ROLO proposes to study the regression capability of a single layered LSTM and concatenates the high level visual features which are in the form of 4096 sized feature maps along with 4 location coordinates and 2 class probabilities obtained from the convolutional layers and are fed to LSTM as input at each time step and get the next time step prediction. There are three phases for the end-to-end training of the ROLO model. The structure of the tracking procedures is illustrated in Figure 1.1. The pre-training phase of convolutional layers for feature learning, the traditional YOLO training phase for object proposal, and the LSTM training phase for object tracking. In ROLO[12] LSTM network uses raw video frames from a standard dataset such as Object Tracking Benchmark(OTB)[8] which is a very famous computer vision challenge. This challenge conveniently provide researchers with valuable datasets as well as annotated ground truth, and detailed evaluation methods, which makes the methods comparable with each other.

#### 2.0.2 Re3: Real-Time Recurrent Regression Networks for Visual Tracking of Generic Objects

In this method,[4] Image crop pairs are fed in at each timestep. Both crops are centered around the object's location in the previous frame, and padded to two times

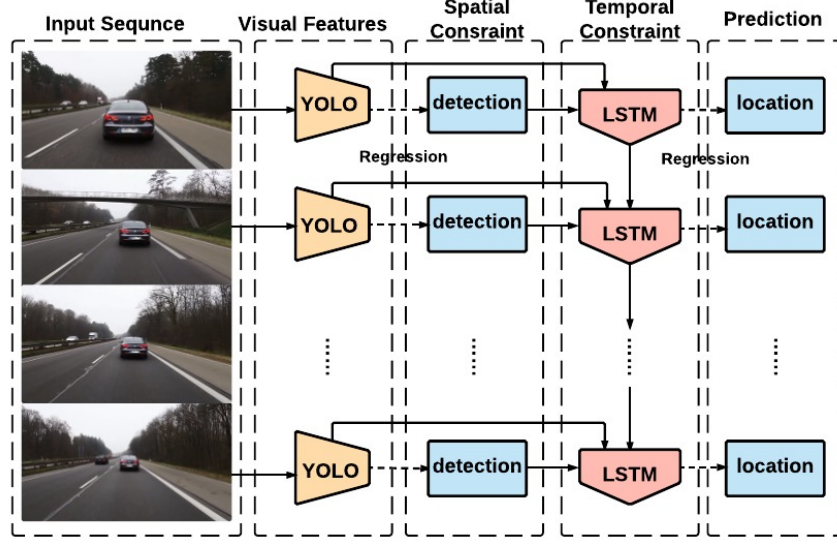


Figure 2.1: Structure of ROLO. (From Guanghan Ning, Zhi Zhang, Chen Huang, Zhihai He, 2016)

the width and height of the object. Before every pooling stage, a skip layer is added to preserve high-resolution spatial information. The weights from the two image streams are shared. The output from the convolutional layers feeds into a single fully connected layer and the LSTM network. The network predicts the top left and bottom right corners of the new bounding box. CaffeNet convolutional pipeline is used to initialize with the CaffeNet pretrained weights for the convolutional layers. The embedding fully-connected layer has 2048 units, and the LSTM network is a 2 stacked LSTM network which has 1024 units each. The structure of the model is illustrated in fig. 1.2. The training begins with few unrolls of time, and slowly increases the time horizon that the network sees to teach it longer-term relationships. All new layers are initialized with the MSRA initialization method. ADAM gradient optimizer is used with the default momentum and weight decay and an initial learning rate of  $10^{-5}$ , which is decrease to  $10^{-6}$  after 10,000 iterations and continue for approximately 200,000 iterations. The tracker model is trained on on two large object tracking datasets: the training set from the ILSVRC 2016 Object Detection from Video dataset (Imagenet Video) which has 3862 training videos with 1,122,397 images, 1,731,913



object bounding boxes, and 7911 unique object tracks and the Amsterdam Library of Ordinary Videos 300++ (ALOV) which consists of 314 videos. The testing of the videos is done on 7 videos that also occur in VOT 2014. The training time for this model is approximately a week.

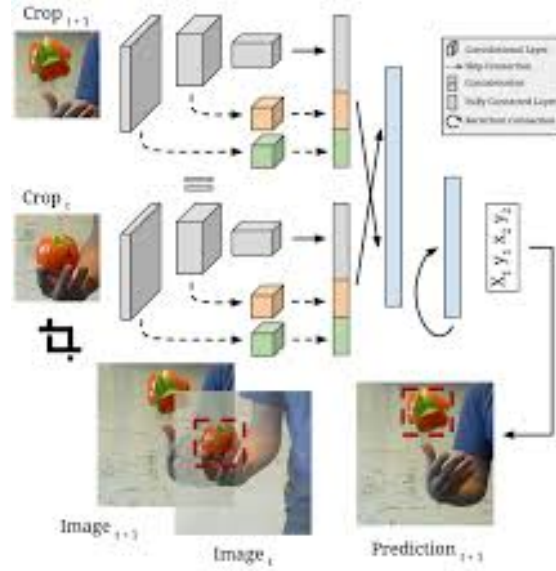


Figure 2.2: Structure of  $Re^3$  Networks. (From Daniel Gordon, Ali Farhadi, and Dieter Fox, 2017)

## CHAPTER 3: BACKGROUND

### 3.1 Convolutional Neural Network

Convolutional Neural Networks (ConvNets or CNNs) are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. ConvNets have been successful in identifying faces, objects and traffic signs apart from powering vision in robots and self driving cars.

#### 3.1.1 The LeNet Architecture (1990s)

LeNet was one of the very first convolutional neural networks which helped propel the field of Deep Learning. This pioneering work by Yann LeCun was named LeNet5 after many previous successful iterations since the year 1988(Ref). At that time the LeNet architecture was used mainly for character recognition tasks such as reading zip codes, digits, etc.

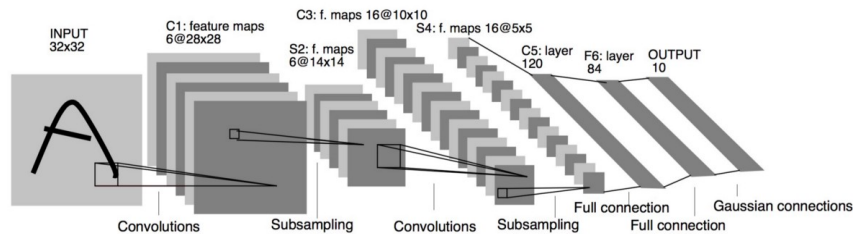


Figure 3.1: LeNet neural network structure. (From LeCun, Yann and Bottou 1998)

CNN has four main steps: convolution, subsampling, activation and fully connect-  
edness. The most popular and first implementation of the CNN is the LeNet, which  
was introduced by Yann LeCun in 1998 [33]. Figure 2.1 illustrates the LeNet struc-  
ture. First step in CNN is convolution. The main idea of using convolution in first  
layers is extracting features from the input image. There are some filters that act

as feature detectors from the original input image. In other words, convolution is a process where input signal is labeled by the network based on what it has learned in the past. If the network decided that the input signal looks like previous cat images that it has learned previously, the "cat" reference signal will be convolved with the input signal. The resulting output signal is then passed on to the next layer. In the second step, subsampling, for reducing the sensitivity of the filters to noise and variations, the inputs from the convolution layer are smoothed. Subsampling also reduces the dimensionality of each feature map but save the most important information. This smoothing process is named subsampling or downsampling or pooling. Subsampling can be achieved by different methods of max, average, sum etc. Third step is activation. The activation layer controls how the signal flows from one layer to the next layer. In different structures of CNNs, a wide variety of complex activation functions could be chosen to model signal propagation. One of the most famous function is (ReLU), which is known for its faster training speed. The ReLU has the mathematical form of:

$$f(x) = \max(0, x) \quad (3.1)$$

The forth step is fully connected. The last layers in the most Convolutional network are fully connected. It means that neurons of previous layers are connected to every neuron in next layers. The output from the convolutional and pooling layers contain high-level features of the input image. Features of fully connected layers are used by softmax layer and the input image is classified into different classes based on the training data. Also fully-connected layers help to learn non-linear combinations of mentioned features. Combinations of those features might be better for classification or other application of CNN.

### 3.2 Recurrent Neural Network

Recurrent Neural Networks (RNNs) are popular models that have shown great promise in many works which need the information of history such as language processing or video processing. The main idea behind RNNs is to use sequential information. In other neural networks, all inputs and outputs are independent of each other. But for many tasks it does not work well. For example, If you want to predict the next word in a sentence, it is better to know which words came before it. RNNs are called recurrent because they perform the same task for every element of a sequence, and the output depends on the previous computations. Also we can say that RNNs have a "memory" which captures information about what has been calculated so far. In theory RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to looking back only a few steps. Here is what a typical RNN looks like: Figure 2.2 shows a RNN is unrolled or unfolded into a full network.

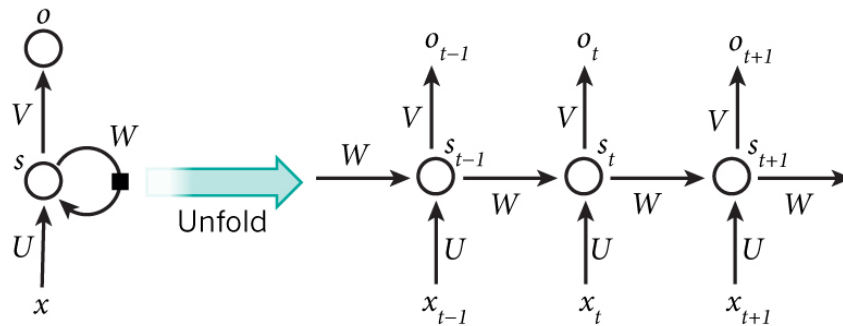


Figure 3.2: recurrent neural network and the unfolding in time of the computation involved in its forward computation

With unrolling we simply mean that we write out the network for the full sequence. For example, if we want to use 5 sequences of a video, the network would be unrolled into a 5-layer neural network, one layer for each sequence. More details about the parameters in figure and the formulas of RNN are as follows:  $x_t$  is the input at time step  $t$ . For example,  $x_1$  could be a vector corresponding to the second sequence of a video.  $s_t$  is the hidden state at time step  $t$ . It's the "memory" of the network.  $s_t$  is

calculated based on the previous hidden state and the input at the current step:

$$s_t = f(U_{x_t} + W_{s_{t-1}}) \quad (3.2)$$

The function  $f$  usually is a nonlinearity such as  $\tanh$  or  $\text{ReLU}$ .  $s_{-1}$ , which is required to calculate the first hidden state, is usually initialized to zero.  $o_t$  is the output at step  $t$ . For example, if we wanted to predict the the position of human in the next time stamp in a video it would be a vector of probabilities.

$$o_t = \text{softmax}(V_{s_t}) \quad (3.3)$$

Training a RNN is similar to training other neural network. Backpropagation algorithm is used here, but with a little change. As the parameters are shared by all time steps in the network, the gradient at each output is calculated not only based on the calculations of the current time step, but also on the previous time steps. For example, in order to calculate the gradient at  $t=6$  we would need to backpropagate 5 steps and sum up the gradients. This method is named Backpropagation Through Time (BPTT). RNNs have shown great success in many tasks. But the most commonly used type of RNNs are LSTMs, which are much better at capturing long-term dependencies than vanilla RNNs are. LSTM network will be explained in the next section.

### 3.2.1 Long Short Term Memory Units(LSTMs)

LSTM is a special kind of recurrent neural network which works for many tasks, much better than the traditional version [34]. One of the appeals of RNNs is the idea that they might be able to connect previous information to the present task, such as using previous video frames might inform the understanding of the present frame or predict future frame. But RNN does not have long memory for doing a perfect predic-

tion. Sometimes, we only need to look at recent information to perform the present task, but some times we need to have long term memory for our prediction. Long Short Term Memory networks (LSTMs) are a special kind of RNN, capable of learning long term dependencies. They were introduced by Hochreiter and Schmidhuber (1997) [35]. In a standard recurrent neural network, in the gradient back-propagation phase, the gradient signal is multiplied by the weight matrix of recurrent hidden layer, a large number of times (related to the number of timesteps). This is the reason of importance of magnitude of weights in the transition matrix. If the weights in this matrix are small (smaller than 1.0), it causes vanishing gradients because the gradients become so small and learning becomes very slow or stops working. So the task of learning long-term dependencies in the data would be impossible. The vanishing gradient problem is illustrated in Figure 2.3. On the other hand, if the weights in this matrix are large (larger than 1.0), it could cause exploding gradients, it means that the gradients become so large that it can cause learning to diverge. These problems of

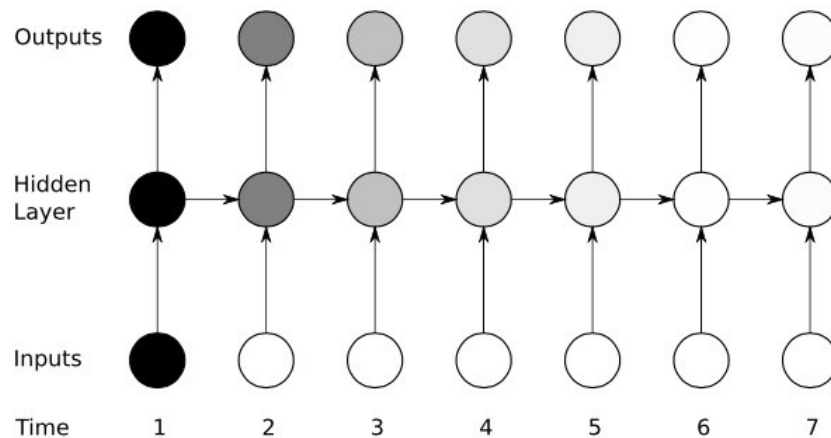


Figure 3.3: The vanishing gradient problem for RNN. The shading of the nodes in the unfolded network shows their sensitivity to the inputs at time one (the darker the shade, the greater the sensitivity). The sensitivity decrease over time as new inputs overwrite the activations of the hidden layer, and the network 'forgets' the first inputs. (From Alex Graves, 2012)[36]

RNN are the main motivation for designing the LSTM model which have a memory cell which is shown in Figure 2.4. A memory cell has four main elements: an input

gate, a neuron with a self-recurrent connection (a connection to itself), a forget gate and an output gate. The weight of the self-recurrent connection is 1.0 and ensures that the state of a memory cell can remain without change in different timestep. The input gate can let incoming signal change the state of the memory cell or block it. Also, the output gate can let the state of the memory cell change other neurons or prevent it. The forget gate can let the cell to remember or forget its previous state, as needed. In the following paragraphs, these elements will be discussed further. Gradient information is preserved by LSTM. Figure 2.5 illustrates preservation of

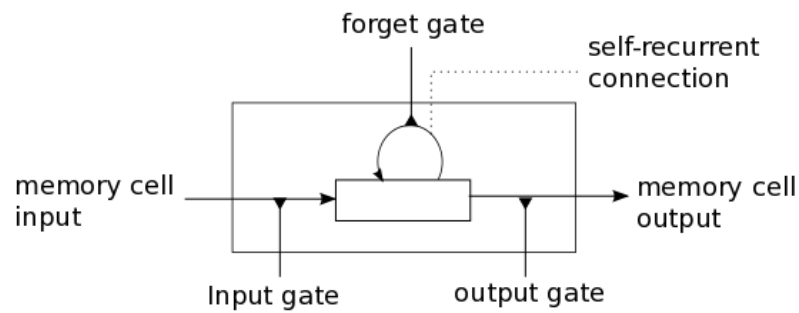


Figure 3.4: LSTM memory cell, (From Kyunghyun Cho Pierre Luc Carrier, 2017)

gradient information by LSTM. As it was showed in Figure 2.3 the shading of the nodes shows their sensitivity to the inputs at time one; in the LSTM, the black nodes are maximally sensitive and the white nodes are completely insensitive. The input, forget, and output gates are illustrated below, to the left and above the hidden layer respectively. All gates are either entirely open ("O") or closed ("-"). The memory cell "remembers" the first input until the forget gate is open and the input gate is closed. The sensitivity of the output layer can be switched on and off by the output gate without affecting the cell [36]. All recurrent neural networks have the form of a chain of repeating modules of neural network. In traditional RNNs, this repeating module will have a very simple structure, such as a single tanh layer. LSTMs also have similar chain structure, but the repeating module is a bit different. Instead of

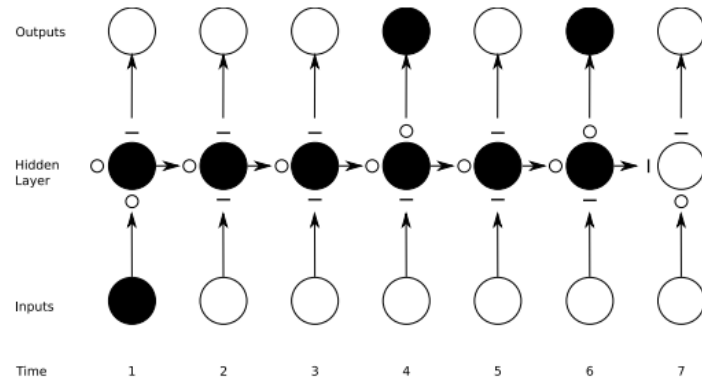


Figure 3.5: Preservation of gradient information by LSTM.(From Alex Graves, 2012)

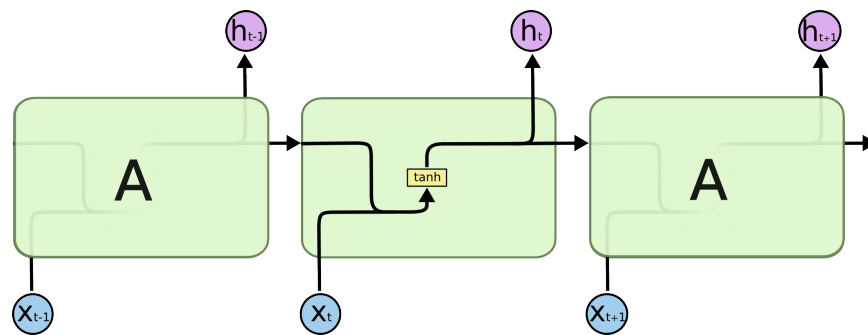


Figure 3.6: The repeating module in a standard RNN contains a single layer (From Christopher Olah, 2015)

having a single neural network layer, there are four, interacting in a very special way. In the Figure 2.7, each line includes an entire vector, from the output of one node to the inputs of others. The pink circles represent operations, such as vector addition, and the yellow boxes are learned neural network layers. Lines merging shows concatenation, and a line forking address its content being copied and the copies going to different locations. This information is showed in the following figure. The one important factor in LSTMs is the cell state, the horizontal line running through the top of the diagram shows cell state. The cell state is similar to conveyor belt. It connects the entire chain, with a linear interactions and information flow along it without change.

It is possible to add or remove information to the cell state by gates. Gates are



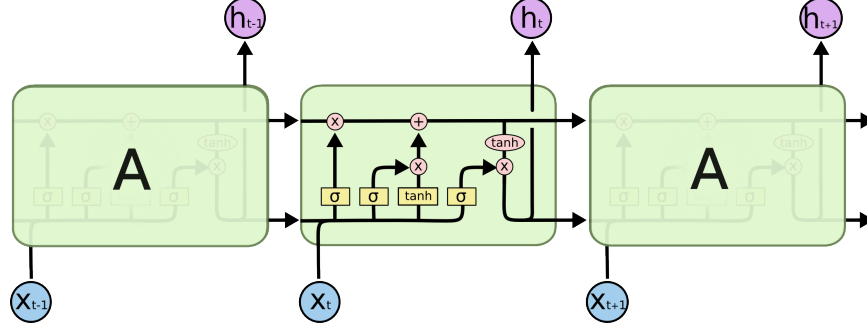


Figure 3.7: The repeating module in an LSTM contains four interacting layers (From Christopher Olah, 2015)

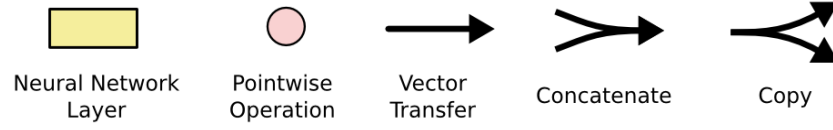


Figure 3.8: : Information about different parts of LSTM diagram (From Christopher Olah, 2015)

combination of a sigmoid neural net layer and a multiplication operation.

An LSTM has three of these gates, to protect and control the cell state. As a new input comes and input gate it is activated, new information will be accumulated to the cell. Also, if the forget gate  $f_t$  was on, the past cell status  $c_{t-1}$  could be "forgotten". The output gate  $o_t$  controls whether the latest cell output  $c_t$  propagated to the final state  $h_t$  or not. The LSTM architecture uses memory cells to store and use information of the history, to discover long-range temporal relations. Nonlinear sigmoid  $\sigma = (1 + e^{-x})^{-1}$ , outputs numbers between zero and one. Value of zero means "let nothing through," while a value of one means "let everything through!"[34]. The formula of input gate  $i_t$ , forget gate  $f_t$ , output gate  $o_t$  and final state  $h_t$  is define

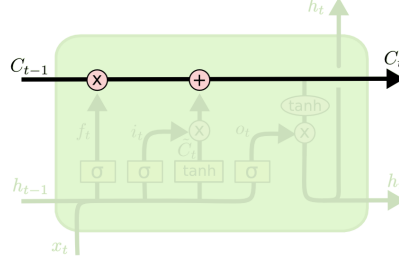


Figure 3.9: cell state in the LSTM, (From Christopher Olah, 2015)

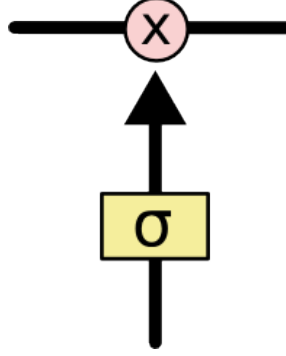


Figure 3.10: Gate in the LSTM, (From Christopher Olah, 2015)

as following:

$$i_t = \sigma(W_{x_i}x_t + W_{h_i}h_{t-1} + b_i) \quad (3.4)$$

$$f_t = \sigma(W_{x_f}x_t + W_{h_f}h_{t-1} + b_f) \quad (3.5)$$

$$o_t = \sigma(W_{x_o}x_t + W_{h_o}h_{t-1} + b_o) \quad (3.6)$$

$$g_t = \sigma(W_{x_c}x_t + W_{h_c}h_{t-1} + b_c) \quad (3.7)$$

$$h_t = o_t * \tanh(c_t) \quad (3.8)$$

$$c_t = f_t * c_{t-1} + i_t * g_t \quad (3.9)$$

The main difference of LSTM with classical RNNs is the use of the these gating functions  $i_t$ ,  $f_t$ ,  $o_t$ , which explained previously, and indicate the input, forget, and output gate at time t respectively. Weight parameters  $W_{x_i}$ ,  $W_{h_i}$ ,  $W_{h_f}$ ,  $W_{h_o}$ ,  $W_{x_f}$ ,  $W_{h_c}$ ,  $W_{x_o}$  and  $W_{x_c}$ , connect the different inputs and gates with the memory cells and outputs and biases  $b_i$ ,  $b_f$ ,  $b_c$  and  $b_o$ . The cell state  $c_t$  is updated with a fraction of

the previous cell state  $c_{t-1}$  that is controlled by  $ft$ [37].

### 3.2.2 Stacked LSTM

A sufficiently large single hidden layer Multilayer Perceptron can be used to approximate most functions. Increasing the depth of the network provides an alternate solution that requires fewer neurons and trains faster. Ultimately, adding depth is a type of representational optimization. Neural Networks[1] are known as function approximation, They try correlate inputs with outputs through weighted sums and activations. Hence Depth or complexity of Networks directly effect input, output correlation. Network Complexity increases as input output relation needed to established get complex[16]. Hence to increase accuracy or to make the network to approximate the input and output relation with high accuracy, complexity is needed in the network. In case of CNN's[16] number of layers are increased to increase accuracy, In case of LSTM's[17] increase in number of hidden units make the network wider capturing more feature maps, but doesn't increase complexity. Hence to increase depth and increase complexity LSTM's are stacked[5]. As shown in fig 1.3, a stacked LSTM architecture can be defined as LSTM model comprised of multiple LSTMs. First LSTM provides a sequence of output rather than a single value output to second LSTM. Specifically, one output per input time step, rather than one output time step for all input time steps. The second LSTM produces a single output for the given input time steps and error for backpropagation is computed on this prediction based on ground truth. Stacking LSTM[3, 5] makes the inputs to be processed through multiple gates of multiple LSTM's interacting and effecting cell state and memory states of multiple LSTM's thus creating a complex network and providing high level of abstraction aiming to improve the accuracy. But in case of large input sizes, with more stacked layers the complexity over-complicate the network making the network to converge very slow. Thus stacked LSTMs can suffer with the below problems. First, deployment of a stacked LSTM model consumes substantial storage, memory

bandwidth, and computational resources. Such demands may be too excessive for mobile phones and embedded devices. Second, stacked LSTMs are prone to overfitting but hard to regularize. Employing standard regularization methods that are used for feed-forward neural networks (NNs), such as dropout, in an LSTM cell is challenging. Third, the increasingly stringent run-time latency constraints in real-time applications make stacked LSTMs, which incur high latency, inapplicable in these scenarios. All these problems pose a significant design challenge in obtaining compact, fast, and accurate LSTMs.

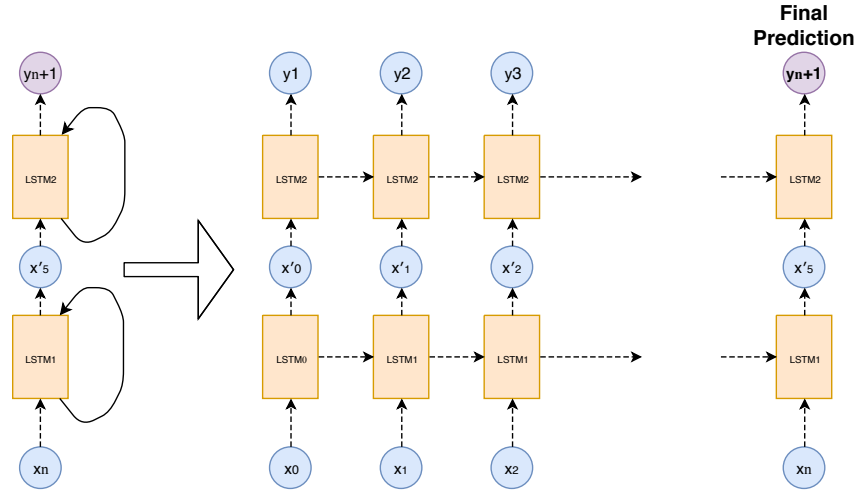


Figure 3.11: LSTM Stacking

## CHAPTER 4: PROPOSED APPROACH

### 4.1 Sparsely Stacked LSTMs

#### 4.1.1 Baseline Implementation of ROLO(YOLO+LSTM)

Tracker Module: Compared to other region proposal classification networks such as fast RCNN [18] which perform detection on various region proposals and thus end up performing prediction multiple times for various regions in an image, YOLO architecture is more like FCNN (fully convolutional neural network) [19] It trains on full images and directly optimizes detection performance. It divides the input image into an  $S \times S$  grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts  $B$  bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts. Formally confidence is defined as  $Pr(Object) * IOU$ . If no object exists in that cell, the confidence scores should be zero. Otherwise the confidence score is equal to the intersection over union (IOU) between the predicted box and the ground truth. Each bounding box consists of 5 predictions:  $x$ ,  $y$ ,  $w$ ,  $h$ , and confidence. The  $(x, y)$  coordinates represent the center of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image. Each grid cell also predicts  $C$  conditional class probabilities,  $Pr(Class_i | Object)$ . These probabilities are conditioned on the grid cell containing an object. Only one set of class probabilities per grid cell is predicted, regardless of the number of boxes  $B$ . At test time we multiply the conditional class probabilities and the individual box

confidence predictions,

$$Pr(Classi|Object) * Pr(Object) * IOU = Pr(Classi) * IOU \quad (4.1)$$

which gives class-specific confidence scores for each box. These scores encode both the probability of that class appearing in the box and how well the predicted box fits the object. Finally these predictions are encoded as an  $SX SX(BX5 + C)$  tensor.

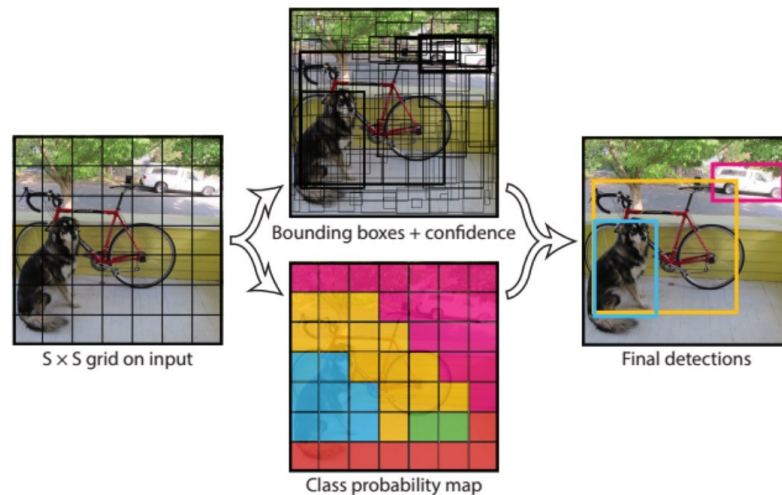


Figure 4.1: YOLO Model

Network Architecture: YOLO has 24 convolutional layers followed by 2 fully connected layers. Instead of the inception modules used by GoogLeNet, YOLO simply uses  $1 \times 1$  reduction layers followed by  $3 \times 3$  convolutional layers.

Why did we choose YOLO:

1. YOLO is extremely fast. YOLO runs at 45 frames per second with no batch processing on a Titan X GPU and a fast version runs at more than 150 fps. This means YOLO can process streaming video in real-time with less than 25 milliseconds of latency.
2. YOLO reasons globally about the image when making predictions. Unlike slid-



module using LSTM network. LSTMs are spatially deep as it is capable of interpreting the visual features and detecting objects on its own, which are spatially supervised by concatenating locations to the visual features. They are also temporally deep as they explore the temporal features as well as their possible locations. We design our LSTM network first using a single layer and then by stacking two LSTMs in order to improve the performance. The complete model of Detector and Tracker module is illustrated in the figure 3.4.

We design our LSTM network containing a single layer of LSTM and hidden units to

network.png

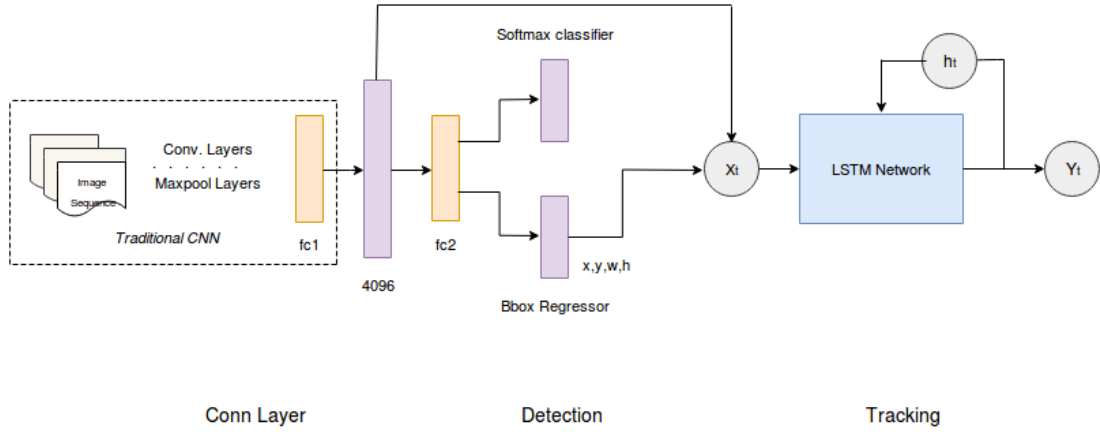


Figure 4.3: Proposed Network

be equal to the input size i.e. 4101 so that the complexity of the model is maintained in order to learn the input features by the model. Step size denotes the number of previous frames considered each time for a prediction by LSTM. ROLO [12] experimented with the step size and varied it from 1 to 9 in order to understand how sequence step of LSTM affects the overall performance and running time of the model. The results are illustrated in figure 3.5. As the step size of 6 for 1 layer LSTM was proved to be optimum in terms of performance and running time, we ran our experiments keeping step size number as 6. We evaluate the performance of the model using Mean square error while training and Average IOU score while inference.



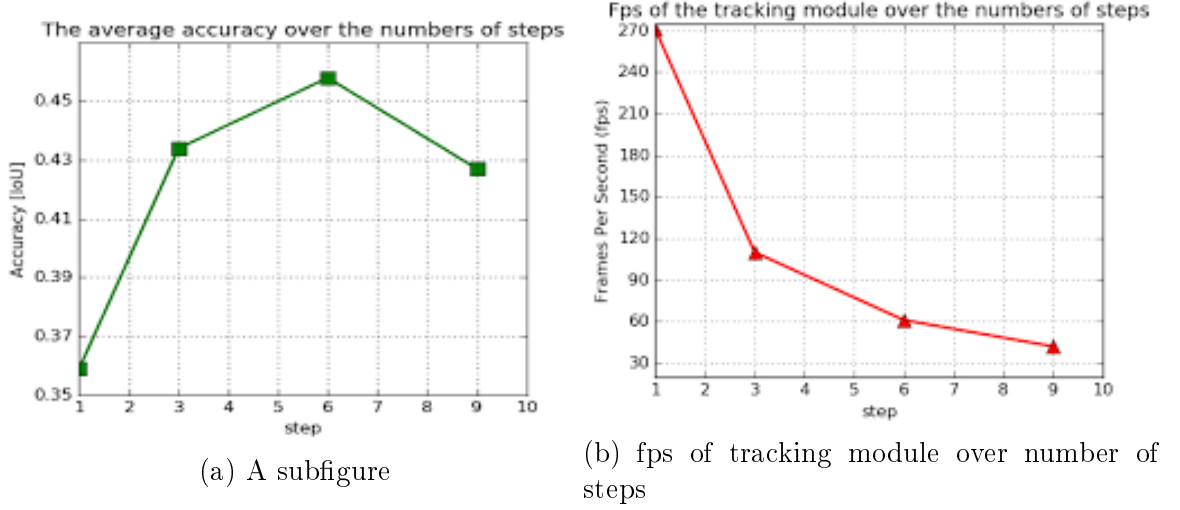


Figure 4.4: Average IOU scores and fps over various number of steps (From Guangan Ning, Zhi Zhang, Chen Huang, Zhihai He, 2016)

We performed 2 experiments with LSTM algorithm and studied its behavior on our dataset. The experiment setup and experiments are explained below.

#### 4.1.2 Stacked LSTM Implementation (YOLO+Stacked LSTM)

In our first experiment with the LSTM network we build a 2 layered stacked LSTM network to improve the performance of the tracker. We implement a 2 layered stacked LSTM with input number of hidden units in both the LSTMs, time step of size 6 and learning rate of 0.0001. We use Adam Optimizer for back propogation in Tensorflow. However, in first 10 epochs of training, the backpropogation tensor goes up to size of [8202, 32808] which forces even a server class GPU to go Out Of Memory while allocating the resources in tensorflow. Even when reducing the number of hidden units in each layer, the complexity of the network eventually increases and causes the GPU to go Out of Memory. We performed the training by using multiple multiple server class GPU's, however, the network did not converge due to over complexity of Network.

#### 4.1.3 Sparsely Stacked LSTM Implementation (YOLO+S2LSTM)

As mentioned in *Stacked LSTM* section, traditional stacking causes over complexity and creates a slow convergent network. One way to reduce the network complexity is by reducing the number of units in LSTM layer which in turn reduces the feature maps to train on and thus creates a converging network as done in [4], but due to reducing the number of units, under-fitting of the training data may occur. Deployment of stacked LSTM model consumes substantial storage, memory bandwidth, and computational resources. Stacked LSTM models become over complex with large input data, resulting into slow convergence during training. Stacked LSTMs incur high latency during inference, proving to be a limitation in real-time applications. All these problems pose a significant design challenge in obtaining compact, fast, and accurate LSTMs.[20] To avoid this we came up with idea of sparse stacking of LSTM i.e. to create a modular coarse tracking taking only final predictions of first LSTM at continuous time frames and feed them to second LSTM as time steps to predict future step.

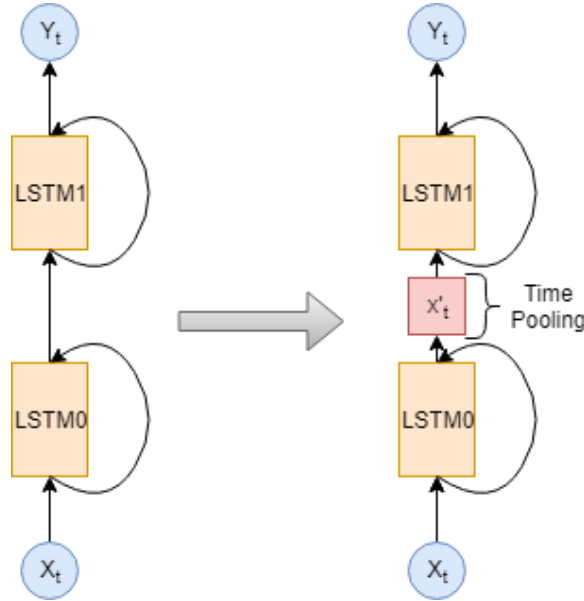


Figure 4.5: Stacked LSTMs Vs. Sparse Stacked LSTMs

In sparse stacking approach as shown in fig 3.4 below, rather than feeding all the intermediate time step outputs of first LSTM only last  $N^{th}$  time step output for LSTM[12] is fed to next LSTM and in similar way continuous N inputs from first LSTM are passed to the second LSTM as Nth time steps. Thus, second LSTM tracks or produces prediction for ' $N + 1^{th}$ ' time step while first LSTM [21] predicts ' $N + 1^{th}$ ' step for an Input of time steps from ' $N - N_t$ ' to ' $N$ ' as input for system where as ' $N_t$ ' represents the number of time steps. By using this approach second LSTM is fed with final predictions which are more important than warming up predictions of intermediate time steps and increases the accuracy of tracking and also training of LSTMs being completely independent as they have different ground truth enabled the approach to be modular. The Sparse Stacked LSTM is a separate LSTM running on ROLO predicting ' $N + 1$ ' frame based on predictions of ROLO. Thus, it uses the information from ' $N$ ' previous frames at a single time step to track the object with more accuracy.

In sparse stacked LSTM configuration, each LSTM training can be done together in parallel or sequentially one after other training. Each Network weights are optimized based back propagation with their own ground truths. In stacked LSTM's[22] output of all the steps of first LSTM are passed as input to next LSTM as inputs for each input time step. In Sparse Stacked LSTMs only the significant time step output of first LSTM is fed to second LSTM reducing the data input by ' $n$ ' times for second LSTM for each time step. In sparse stacked LSTM second LSTM predicts ' $N + 2^{th}$ ' time step while first LSTM predicts ' $n + 1$ ' time prediction. Our Network is trained on 30 videos from OBT[8]. This configuration being modular, each LSTM can be trained separately with respective ground truth and each LSTM is optimized to respective training steps allowing the network to be trainable with less computational complexity.

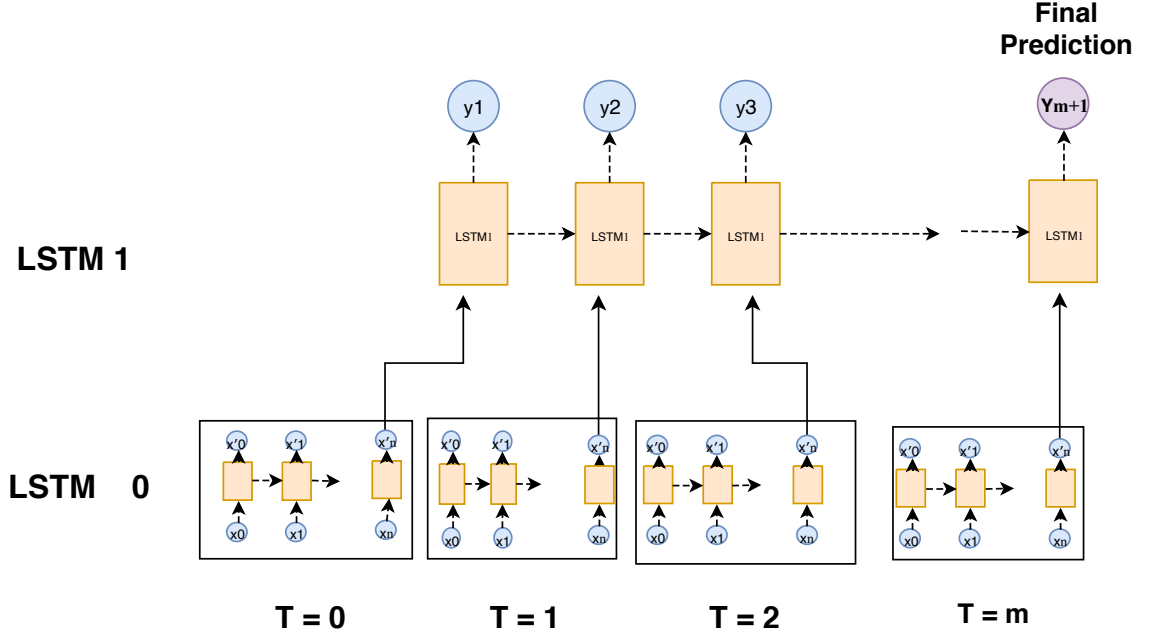


Figure 4.6: Sparse Stacking of LSTM's

Considering the case where  $N_t = 3$  which means number of time steps are '3'. With input frames starting from 'n' below table 4.1 represents the outputs of First LSTM and output of Sparse Stacked Second LSTM for 't+1' and 't+2' frame. As each training is modular training the network is flexible and doesn't require multiple server class GPU's as required in case of Stacked LSTM [3, 5] and also training weight tensors are not combined as in case of traditional stacking of LSTM's easing up the training process and modularity. Sparse Stacking also saves time as retraining of trained module can be avoided incase of any development.

Fig.4.7 shows the complete structure of system from Image input Sparse Stacked LSTM output including intermediate stages.

#### 4.2 Data Classification based on Complexity of Scenarios in Dataset

In the above two approaches, we observed that the overall accuracy of S2LSTM is improved compared to 1 layered LSTM. However, the overlap accuracy varied for different videos. Videos having shaky camera motion, fast camera and object motion,

Table 4.1: Input and Output Time steps for LSTM1 and Sparse Stacked LSTM Prediction

Input	First LSTM Output	Second LSTM Input	Sparse Stacked LSTM predicting $t+1$
$n$	-	-	-
$n$ to $n+1$	-	-	-
$n$ to $n+2$	$n+3$	$n+3$	-
$n+1$ to $n+3$	$n+4$	$n+3$ to $n+4$	-
$n+2$ to $n+4$	$n+5$	$n+3$ to $n+5$	$n+6$
$n+3$ to $n+5$	$n+6$	$n+4$ to $n+6$	$n+7$
$n+4$ to $n+6$	$n+7$	$n+5$ to $n+7$	$n+8$
$n+5$ to $n+7$	$n+8$	$n+6$ to $n+8$	$n+9$

out of plane object rotation, low brightness and low resolution had low performance compared to the rest of the videos.

#### 4.2.1 Dataset

In recent years, Different benchmark datasets have been developed for various vision problems. Some standard datasets such as Object Tracking Benchmark (OTB)[8] and VOT2017 [9] are very famous for Tracking task. These two challenges conveniently provide researchers with valuable dataset as well as annotated ground truth. The dataset contains different challenging visual situations which are classified into 11 attributes. These attributes are explained in table 4.2 We use 30 videos from OBT

Table 4.2: Challenging aspects of Object Tracking in dataset

Attribute	Description
Illumination Variation	the illumination in the target region is significantly changed.
Scale Variation	the ratio of the bounding boxes of the first frame and the current frame is out of the range $t_s, t_s > 1$ ( $t_s=2$ ).
Occlusion	the target is partially or fully occluded.
Deformation	non-rigid object deformation.
Motion Blur	the target region is blurred due to the motion of target or camera.
Fast Motion	the motion of the ground truth is larger than $t_m$ pixels ( $t_m=20$ ).
In-Plane Rotation	the target rotates in the image plane.
Out-of-Plane Rotation	the target rotates out of the image plane.
Out-of-View	some portion of the target leaves the view.
Background Clutters	the background near the target has the similar color or texture as the target.
Low Resolution	the number of pixels inside the ground-truth bounding box is less than $t_r$ ( $t_r=400$ )

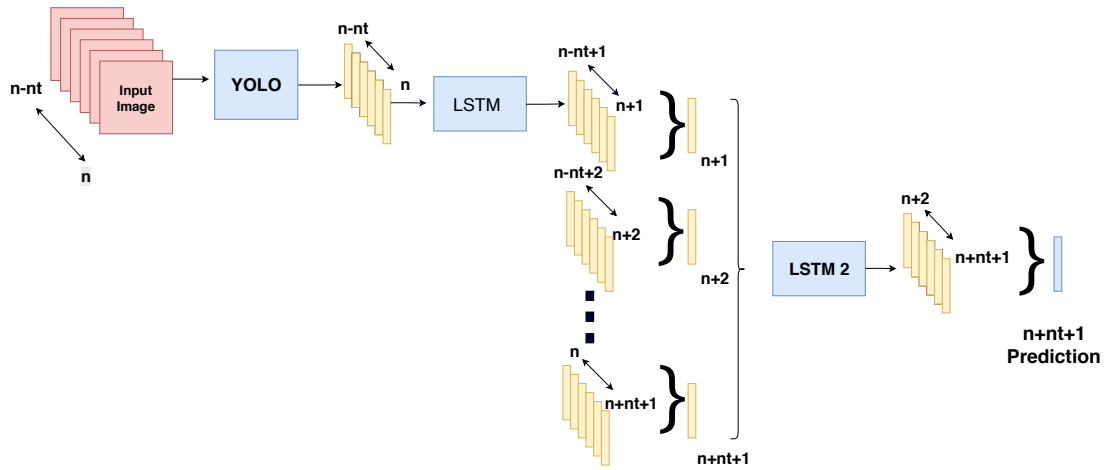


Figure 4.7: Sparse Stacked LSTM on ROLO

which includes all 11 attributes stated above for training and testing the model. We

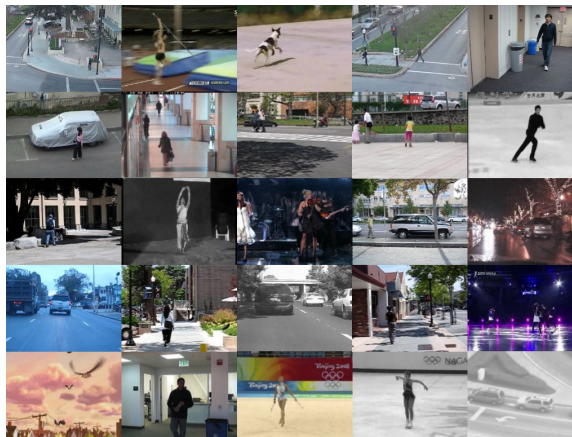


Figure 4.8: First frame of the target object for sequences that used in this project

classify the videos into various categories: Shaky, High, Medium and Low Camera Motion, High, Medium and Low Object Motion. Based on the relative object and camera motion observation in all the vidoes, we further classify videos into High, Medium and Low Relative motion. Then we classify the videos based on resolution, brightness and Object deformation. Accurate object detection in case of resolution, brightness and object deformation can be achieved using better detection algorithms like [23]. In order to increase accuracy of videos with shaky camera motion, fast camera and object motion, out of plane object rotation, we experiment with the

number of steps of LSTM layers. Since videos with high speed need more number of historical frames to predict the location coordinates in future frame, we set the time step size of the dataset based on different complexity of videos. For high relative motion, we set the time step number as 8, for medium relative motion, we set time step number as 7 and for low relative motion, we set the time step number as 6.

## CHAPTER 5: EXPERIMENTAL RESULTS

### 5.1 Target Dataset

Object Tracking Benchmark (OTB) and Visual Object Tracking 2017(VOT2017)challenge dataset is used to train and test the models. Preprocessing of the data was required to get OTB and VOT2017 groundtruths in the same format. OTB dataset has the groundtruth coordinates in x,y,width and height format where x and y denotes the bottom left corner coordinates, where as VOT dataset has the groundtruths in the format of x1,y1, x2,y2,x3,y3 and x4,y4 i.e. they denote coordinates of all the 4 corners of the rectangle. Thus a python program was written to convert coordinates of VOT dataset in the format of OTB dataset. We use 20 videos from OTB dataset to train the models; and 10 videos from OTB and 5 videos from VOT for testing on it.

### 5.2 Experimental Setup

#### 5.2.1 Software Setup

We use Tensorflow 1.8.0 and Python2.7 to train and test our network. We used pre-trained weights of the YOLO detection model extracted into python from Darknet which were trained using Darknet framework. Libraries used are Numpy and OpenCV 3.6.

#### 5.2.2 Hardware Setup

Experiments are conducted on NVIDIA Tesla P100 GPU with 12GB RAM and NVIDIA Quadro with 8GB RAM.



### 5.3 Tracking Performance Evaluation

Performance of the trackers could be evaluated by several evaluation criteria. One widely used evaluation metric for object tracking is the average center location error, which calculates the average Euclidean distance between the center locations of the tracked targets and the labeled ground-truth positions of all the frames. When a tracking algorithm loses track of a target object, the output location can be random, and thus, the average error value does not measure the tracking performance correctly [24]. If the estimated locations are within a given threshold distance of the ground-truth would be evaluated, it would be more accurate evaluation[24] The raw videos obtained from OBT and VOT 2017 dataset are passed through the pre-trained YOLO model and visual features from convolutional layers and location information is obtained for each frame of the videos. We refer to the visual feature vector as  $X_t$  and the location bounding box vector as  $B_t$ . This stream of data is passed into the LSTM network during training. In addition to  $X_t$  and  $B_t$  there is another input to the LSTM which is the output of hidden state from the previous time state  $h_{t-1}$ . In out objective module we use Mean Squared Error(MSE) for training:

$$LMSE = 1/n \sum_{i=1}^n ||B_{target} - B_{pred}||^2 = 1 \quad (5.1)$$

Another commonly used evaluation metric is the overlap score[25]. The average overlap score (AOS) can be calculated using Intersection Over Union method which is simply a ratio where numerator computes the area of overlap between the predicted bounding box and the ground-truth bounding box and denominator is the area of union, or more simply, the area encompassed by both the predicted bounding box and the ground-truth bounding box. The method can be clearly explained in fig 5.1. In addition, the overlap scores can be used for determining whether an algorithm

successfully tracks a target object in one frame, by testing whether  $S$  is larger than a certain threshold  $t_0$  (for example, 0.5). Thus the predicted location coordinates obtained at time step  $t$  from tracker module are compared with the ground-truth coordinates of next time-step i.e.  $t+1$  and Intersection Over Union is calculated for each frame. Average IOU is the average of IOU for all the frames in each video.


$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


Figure 5.1: Intersection over Union Equation

#### 5.4 Experimental Result Analysis

It is difficult to say that which of the qualitative or quantitative evaluation is better for tracking problem. Also it is a challenging task to measure the accuracy of a tracking method with quantitative metrics. Many factors such as ground-truth position accuracy, robustness, tracking speed, memory requirement can be considered. Even in one frame with the tracking output and ground-truth object state, there can be several metrics to measure accuracy. Some times tracking results show that tracker does not lose the object but the size of the bounding box is smaller than ground

truth. The tracker performance depends a lot on the detector performance. When in a given frame if the object is partially occluded, the detector output in real time is zero which is fed to the tracker. Even if the prediction for that frame is accurate, the performance of the tracker in future frames is affected. In some cases, even if the tracker loses the object in one frame and then tracks it successfully in the next frames, the overall performance evaluated over all frames in sequence is affected.

#### 5.4.1 Quantitative Results

Table 5.1: Summary of Average Overlap Scores (AOS) results for all 15 test videos

Relative Motion	Sequence	YOLO + LSTM with varied steps#	YOLO + S2LSTM with varied steps	YOLO + LSTM with fixed time step = 6	YOLO + S2LSTM with fixed time step = 6
Low	Human6	0.361	0.382	0.12	0.155
	Car4	0.344	0.380	0.303	0.383
	Skater2	0.521	0.588	0.318	0.371
	Dancer2	0.553	0.558	0.417	0.434
Medium	Human2	0.279	0.321	0.0529	0.601
	SUV	0.447	0.502	0.187	0.278
	Walking2	0.203	0.486	0.082	0.254
	IceSkater1	0.203	0.234	0.124	0.125
	Skating1	0.415	0.386	0.267	0.288
	Dog	0.259	0.216	0.032	0.099
High	0.258	0.223	Human9	0.047	0.057
	BlurBody	0.374	0.322	0.466	0.401
	Ice Skater2	0.231	0.247	0.142	0.156
	David3	0.173	0.113	0.052	0.104
	Bird2	0.102	0.108	0.036	0.039
	BlurCar3	0.372	0.375	0.197	0.215

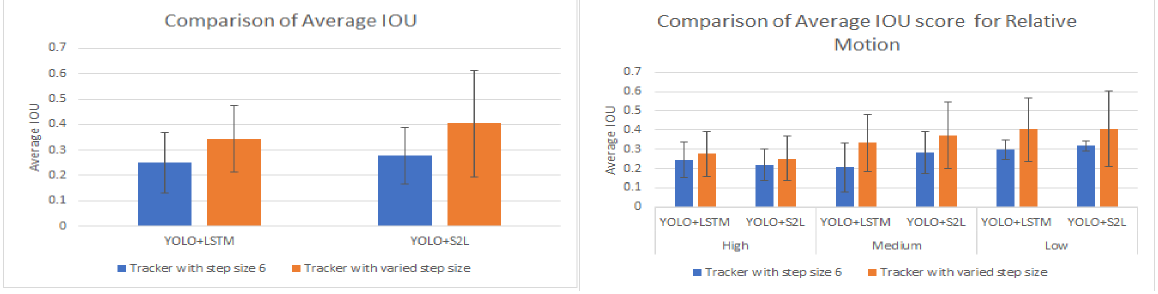


Figure 5.2: Average IOU score for all videos and classified videos based on Relative Motion

Table 5.1 denotes the summary of overlap score for all the models on the videos in OTB30 based on their object-camera relative motion. It is clear from the results that the performance of sparse stacked LSTM is better by approximately 15 % than the single layered LSTM. Also, the results from experiment 3 with tracker having varied time step size are better than experiment 2 with tracker having fixed time step size. This can be seen from fig 4.3 (a). In experiment 2, the overall overlap score for some videos like BlurBody, BlurCar1, Singer2, Skating1 which lie in the Medium and High Camera-Object Relative Motion category has low overlap accuracy. With varied time step tracker, the corresponding overlap accuracy increases by 15-20% which can be seen in Table 5.1 and fig 5.2(b). Another point that should be noted is that the performance of S2LSTM tracker with varied time steps though has overall increased performance in comparison to the S2LSTM tracker with fixed step size of 6, it slightly under-performs compared to single layered LSTM in the High Relative Motion Category. This happens mostly because S2LSTM cannot learn the high motion of the object and camera with the doubled time step number.

Table 5.2 shows the training and Inference time required for trackers in experiment 1,2 and 3. We trained S2LSTM tracker sequentially i.e. both the LSTM layers were trained one after the other on NVIDIA Tesla P100.

Table 5.2: Training and Inference time for the trackers

	LSTM with fixed time step	S2LSTM with fixed time step	LSTM with varied time steps	S2LSTM with varied time steps
Training time	12.5 hours	33.5 hours	12.5 hours	33.5 hours
Inference time	20 fps	16 fps	20 fps	16 fps

#### 5.4.2 Qualitative Results

Qualitative evaluation methods could be more comprehensible for human to see the results in different condition and judge about the performance of the tracking. We try to show the qualitative results in different challenging conditions such as occlusion, motion blur, high, medium and low relative motion, illumination variation, wrong detection information tracked by YOLO+S2LSTM with varied time step approach and discuss about them.

Figures in figure number 5.3 are show 6 images in sequence in video Skater2. This video comes in the category of Scale variation, deformation, in-plane rotation and out-of plane rotation. This video also has low relative motion between the camera and object. Comparison between YOLO+LSTM and YOLO+S2LSTM approach has been done in the images. YOLO+S2LSTM proves to work better on this category. Green bounding box shows tracking results of YOLO+S2LSTM and blue bounding boxes show tracking results of YOLO+LSTM. As you can see S2LSTM tracker tries to cover the entire body while LSTM tracker covers only half of the body. Similar case can be seen in case of Human2 images in figure 5.4. When the target is turning, both the trackers follow the human but only S2LSTM tracker is able to cover the entire body.

Figures in figure number 5.5 are show 6 images in sequence in video BlurCar3. This video comes in the category of Motion Blur. This video also has high relative motion between the camera and object. The Green bounding box shows tracking results of



Figure 5.3: 6 continuous sequences in Skater2 Video

YOLO+S2LSTM and blue bounding boxes show tracking results of YOLO+LSTM. YOLO + S2LSTM tracker can follow the car more effectively than YOLO + LSTM tracker. Even though, at some frames YOLO + S2LSTM loses the target it gets back to the target in the next frame. In this case the tracker follows the target in most of the frames but the bounding box size is small compared to the ground truth bounding box size hence the overlap calculated is low.

Figures in fig. 5.6 shows the case where the target person size is very small in the frame and the background is dark. The target human is taking a 360 degrees turn. The blue bounding boxes which shows the tracking results of YOLO + S2LSTM, could follow the target perfectly while the green bounding box i.e. the YOLO + LSTM tracker loses the target completely. It is because S2LSTM uses the information from the  $N+N$  number of previous the frames where as LSTM uses only  $N$  number of previous frames where  $N$  is the step size.

Figures in 5.7 shows the case where the target object which is a transformer does not come in any of the training environments. Hence the tracking of the transformer has low accuracy. Though the tracker is able to follow the transformer, it cannot fully

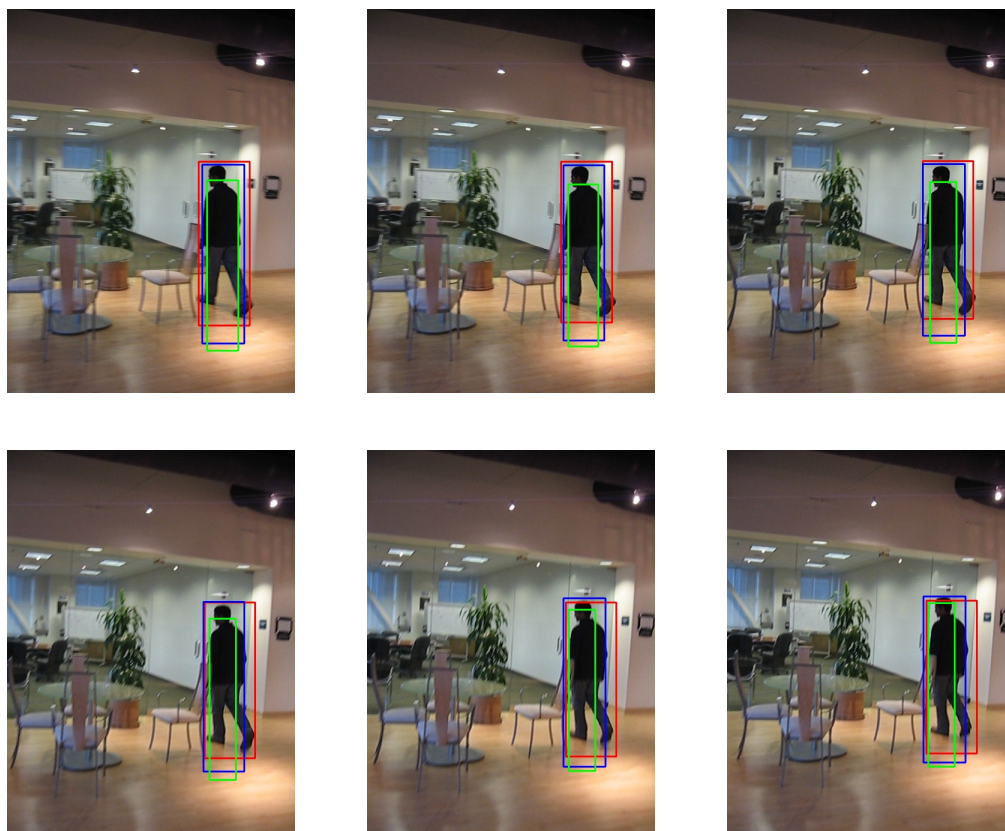


Figure 5.4: 6 continuous sequences in Human2 Video

cover the body.



Figure 5.5: 6 continuous sequences in Blur Car Video

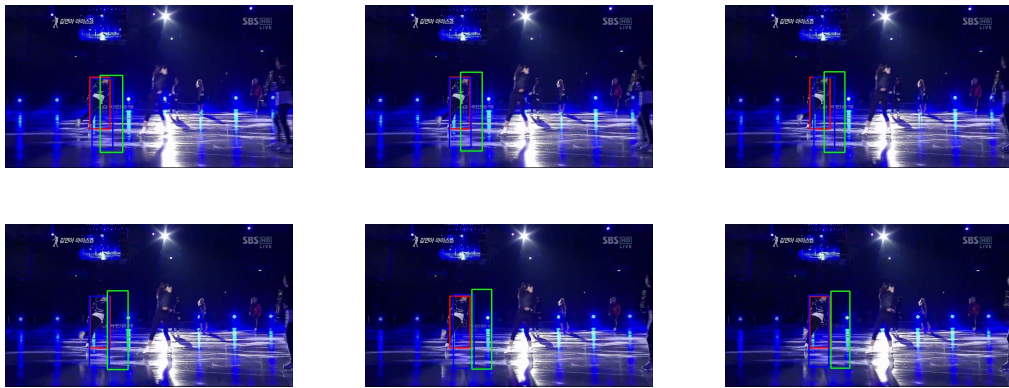


Figure 5.6: 6 continuous sequences in Skating1 Video



Figure 5.7: 6 continuous sequences in Transformer Video



## CHAPTER 6: CONCLUSIONS

This thesis has been proposed with the aim of developing a model for having robust object tracking. We proposed various models to achieve better results. Different approaches have been tried for different tasks during the development of the proposed algorithms implementation which was discussed in the previous chapters.

### 6.1 Summary of Findings

We have developed a method of Sparsely Stacked LSTM for visual object tracking. This method extends the multi layered stacking of LSTM. We change it in the way of using only the significant output from previous LSTM layer so as to reduce the computation complexity and avoid slow convergence of model. The OBT30 dataset was classified into 3 main categories taking the camera and the object motion in the video frames under consideration. With sparsely stacked LSTM being able to learn more number of steps in one time frame, it was able to learn the data more quickly. Using the information from 'N' previous frames at a single time step helps to track object in the cases of motion blur, occlusion, change of illumination etc. Also, the proposed method could track the human after disappearing for short time and appearing again in the video. However, when the Relative Motion between Object and Camera is too high, the history of N+N frames does not help S2LSTM algorithm to track the objects effectively. Hence, in case of high Relative Motion Videos, the S2LSTM performance is lower than 1 layered LSTM. With varying the time steps for different videos based on the relative motion, the LSTMs were able to track more effectively compared to fixed time step LSTMs. This is because LSTMs are temporally deep and by using temporal features set according to the relative speed of the object,

it becomes easier for LSTMs to learn the environment.

## 6.2 Future Work

Although, the proposed approach of Sparsely Stacked LSTM was able to track the object in many challenging conditions, its performance could be improved by doing some more changes in the proposed model. The limitations which reduce the robustness of the method and some suggestions for its improvement are explained below.

1. For improving the performance of S2LSTM approach in case of High Relative Motion, we will build a complex Neural Network with object visual features concatenated with location coordinates from 1 layered LSTM and S2LSTM as input. The prediction obtained from the Neural Network will be compared to the ground-truth. In this way, the Neural Network is trained to use the best of the two predictions and produce the output.
2. We will study three layers of sparsely stacked LSTMs and its behavior on different challenging conditions stated above. With increase in layers of stacked LSTMs, we can experiment with the number of time steps in order to learn the locations of objects with varying speeds.
3. We will explore the detection and tracking for the purpose of multi-object tracking.

## REFERENCES

- [1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS’12, (USA), pp. 1097–1105, Curran Associates Inc., 2012.
- [2] L. C. Jain and L. R. Medsker, *Recurrent Neural Networks: Design and Applications*. Boca Raton, FL, USA: CRC Press, Inc., 1st ed., 1999.
- [3] Z. Cui, R. Ke, and Y. Wang, “Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction,” *CoRR*, vol. abs/1801.02143, 2018.
- [4] D. Gordon, A. Farhadi, and D. Fox, “Re3: Real-time recurrent regression networks for object tracking,” *arXiv preprint arXiv:1705.06368*, 2017.
- [5] J. R. A. Moniz and D. Krueger, “Nested lstms,” in *Proceedings of the Ninth Asian Conference on Machine Learning* (M.-L. Zhang and Y.-K. Noh, eds.), vol. 77 of *Proceedings of Machine Learning Research*, pp. 530–544, PMLR, 15–17 Nov 2017.
- [6] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, pp. 1735–1780, Nov. 1997.
- [7] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, “Long short term memory networks for anomaly detection in time series,” in *Proceedings*, p. 89, Presses universitaires de Louvain, 2015.
- [8] Y. Wu, J. Lim, and M.-H. Yang, “Object tracking benchmark,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1834–1848, 2015.
- [9] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernandez, T. Vojir, G. Hager, G. Nebehay, and R. Pflugfelder, “The visual object tracking vot2015 challenge results,” in *Proceedings of the IEEE international conference on computer vision workshops*, pp. 1–23, 2015.
- [10] Q. Liu, X. Zhao, and Z. Hou, “Survey of single-target visual tracking methods based on online learning,” *IET Computer Vision*, vol. 8, no. 5, pp. 419–428, 2014.
- [11] B. Wu, F. Iandola, P. H. Jin, and K. Keutzer, “Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving,” 2016.
- [12] G. Ning, Z. Zhang, C. Huang, Z. He, X. Ren, and H. Wang, “Spatially supervised recurrent convolutional neural networks for visual object tracking,” *arXiv preprint arXiv:1607.05781*, 2016.

- [13] W. Luo, X. Zhao, and T. Kim, “Multiple object tracking: A review,” *CoRR*, vol. abs/1409.7618, 2014.
- [14] R. Hou, C. Chen, and M. Shah, “Tube convolutional neural network (T-CNN) for action detection in videos,” *CoRR*, vol. abs/1703.10664, 2017.
- [15] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” *CoRR*, vol. abs/1506.02640, 2015.
- [16] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *CoRR*, vol. abs/1512.03385, 2015.
- [17] M. Miwa and M. Bansal, “End-to-end relation extraction using lstms on sequences and tree structures,” *CoRR*, vol. abs/1601.00770, 2016.
- [18] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Advances in neural information processing systems*, pp. 91–99, 2015.
- [19] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.
- [20] X. Dai, H. Yin, and N. K. Jha, “Grow and prune compact, fast, and accurate lstms,” *arXiv preprint arXiv:1805.11797*, 2018.
- [21] Y. Zhai, M. B. Yeary, S. Cheng, and N. Kehtarnavaz, “An object-tracking algorithm based on multiple-model particle filtering with state partitioning,” *IEEE Transactions on Instrumentation and Measurement*, vol. 58, pp. 1797–1809, May 2009.
- [22] X. Du, H. Zhang, H. V. Nguyen, and Z. Han, “Stacked lstm deep learning model for traffic prediction in vehicle-to-vehicle communication,” in *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*, pp. 1–5, Sept 2017.
- [23] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.
- [24] B. Babenko, M.-H. Yang, and S. Belongie, “Robust object tracking with online multiple instance learning,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 8, pp. 1619–1632, 2011.
- [25] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (voc) challenge,” *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.