Fakultät für Mathematik, Informatik und Naturwissenschaften
Institude für Informatik II
Computer Graphik
Jun.Prof. Dr. Angela Yao

Lab Report

Unsupervised Learning of Video Representations using LSTMs

Cun Wang
Matriculation Number: 362024

July 2017

Advisor: Moritz Wolter

Abstract

In this lab, Long Short Term Memory (LSTM) networks are used to learn the representation of video sequences. It adpots the Encoder-Decoder framework. An encoder LSTM is used to map the inputs sequences to a fixed length representation. Then an decoder LSTM decodes this representation to reconstruct the input sequences. In order to capture spatiotemporal correlation better, the convolutional LSTM (ConvLSTM) is applied besides fully-connected LSTM (FC-LSTM).

Contents

1 Introduction

Understanding temporal sequences is important for solving many problems, such as speech recognition, caption generation. Sutksever *et al.* described a general sequence to sequence learning framework in which a recurrent network is used to encode a sequence into a fixed representation, and then another recurrent network is uded to decode the representation into a sequence [**?**]. Srivastava *et al.* use the LSTM Encoder-Decoder framework extending the general framework to learn video representations [SMS15]. A sequence of frames are fed in encoder LSTM to generate a representation. This representation is then decoded through another LSTM to produce a target sequence. Srivastava *et al.* consider two choices of the target. One choice is to use the inputs as the target sequence. The other is to predict the future frames. They also evaluate on two kinds of inputs. One is image patches, the other is the high-level "percepts" extracted by applying a convolutional net.
LSTM is an important part of Encoder-Decoder framework. LSTM is a specical kind of recurrent neural network, which can capture long-term temporal dependencies without suffering from the optimization problems such as gradient vanishing. In order to make use of frame structure, convolutional LSTM introduced in [SCW+15] is applied in Encoder-Decoder framework.

2 Methodology

2.1 LSTM architecture

As mentioned above, Encoder-Decoder framework is composed of LSTMs. We'll introduce vanilla LSTM architecture [GSK+15]. A schematic of the vanilla LSTM block is illustrated in figure 1. The key to LSTMs is the cell state, which is controlled by three kinds of gates, namely input gates, forget gates, output gates, adding information to or removing it from the cell state. The gates are composed of a sigmoid layer and a pointwise multiplication operation. The output of sigmoid layer ranges from zero to one, describing how much of each component should be let through. The forgate gate decides what information is going to be throwed away from the cell state. The input gate decides what information is going to be stored in the cell state. The output gate decides what is going to output.
The vector formulas for a vanilla LSTM layer forward pass are given below. $\boldsymbol{x}^t$ is the input vector at time t, the $\boldsymbol{W}$ are rectangular input weight matrices, the $\boldsymbol{p}$ are peephole weight vectors and $\boldsymbol{b}$ are bias vectors. Using peephole connections, the gate layers can look at the cell state. Functions σ, g and h are pointwise nonlinear activation functions. $\mathbf{i}$, $\mathbf{f}$, $\mathbf{o}$ are separately input gate, forget gate, and output gate. The pointwise multiplication of two vectors is denoted with $\odot$:

$$
\begin{aligned}
\mathbf{z}^t &= g(\mathbf{W}_z\mathbf{x}^t + \mathbf{R}_z\mathbf{y}^{t-1} + \mathbf{b}_z) \\
\mathbf{i}^t &= \sigma(\mathbf{W}_i\mathbf{x}^t + \mathbf{R}_i\mathbf{y}^{t-1} + \mathbf{p}_i \odot \mathbf{c}^{t-1} + \mathbf{b}_i) \\
\mathbf{f}^t &= \sigma(\mathbf{W}_f\mathbf{x}^t + \mathbf{R}_f\mathbf{y}^{t-1} + \mathbf{p}_f \odot \mathbf{c}^{t-1} + \mathbf{b}_f) \\
\mathbf{c}^t &= \mathbf{i}^t \odot \mathbf{z}^t + (f)^t \odot \mathbf{c}^{t-1} \\
\mathbf{o}^t &= \sigma(\mathbf{W}_o\mathbf{x}^t + \mathbf{R}_o\mathbf{y}^{t-1} + \mathbf{p}_o \odot \mathbf{c}^{t-1} + \mathbf{b}_o) \\
\mathbf{y}^t &= \mathbf{o}^t \odot h(\mathbf{c}^t)
\end{aligned}
$$

As pointed out by Srivastava *et al.*, the key advantage of using LSTM unit over a traditional neuron in an RNN is that the cell state in an LSTM unit sums activities over time. This is also the reason why LSTM unit don't suffer from gradient vanishing problem. Since derivatives distribute over sums, the error derivatives don't vanish too quickly as they get sent back into time.

2.2 Convolutional LSTM

Video frames have spatio information in their structure, which is not captured by FC-LSTM, where full connections are used in input-to-state and state-to-state transitions. Because of full connections in FC-

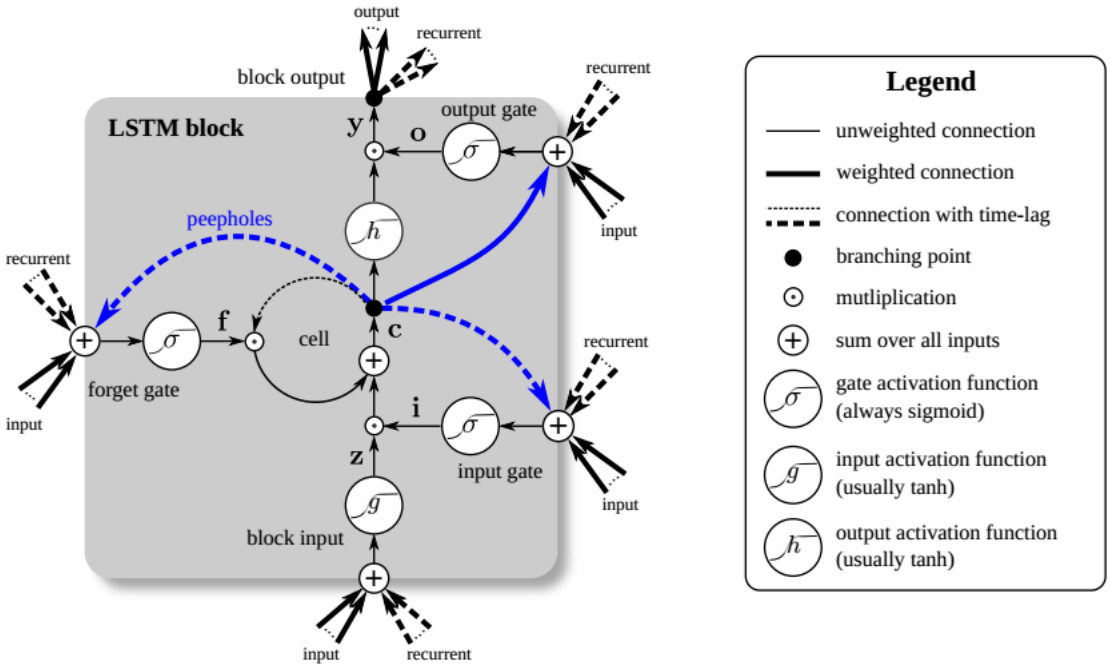

Figure 1: vanilla LSTM block architecture

Source: [GSK+15]

LSTM, it contains too much redundancy for spatial data To make use of spatial information, ConvLSTM is applied in this lab. In the ConvLSTM, the inputs $\boldsymbol{X}_1,...\boldsymbol{X}_t$, cell outputs $\boldsymbol{C}_1,...\boldsymbol{C}_t$, hidden states $\boldsymbol{H}_1,...\boldsymbol{H}_t$, and gates i_t, f_t, o_t are 3D tensors whose last two dimensions are spatial dimensions (rows and columns). The ConvLSTM determines the future state of a certain cell by the inputs and past states of its neighbors, using a convolutional operator in state-to-state and input-to-state transitions. The key equations are given below, where '$*$' denotes the convolutional operator and '$\odot$' denotes the Hadamard product:

$$
\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{xi} * \mathbf{X}_t + \mathbf{W}_{hi} * \mathbf{H}_{t-1} + \mathbf{W}_{ci} \odot \mathbf{C}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_{xf} * \mathbf{X}_t + \mathbf{W}_{hf} * \mathbf{H}_{t-1} + \mathbf{W}_{cf} \odot \mathbf{C}_{t-1} + \mathbf{b}_f) \\
\mathbf{C}_t &= \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot tanh(\mathbf{W}_{xc} * \mathbf{X}_t + \mathbf{W}_{hc} * \mathbf{H}_{t-1} + \mathbf{b}_c) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{xo} * \mathbf{X}_t + \mathbf{W}_{ho} * \mathbf{H}_{t-1} + \mathbf{W}_{co} \odot \mathbf{C}_{t-1} + \mathbf{b}_o) \\
\mathbf{H}_t &= \mathbf{o}_t \odot tanh(\mathbf{C}_t)
\end{aligned}
$$

2.3 LSTM autoencoder model

This model is composed of two Recurrent Neural Network, the encoder LSTM and the decoder LSTM, as shown in figure 3. The input to the model is a sequence of vectors, in our case, video frames. The encoder LSTM runs through these frames to come up with a representation, which is the state of encoder LSTM atfer the last frame has been read. The decoder LSTM then is expected to reconstruct target seqence as input sequence based on the representation, which requires that representation contains information about the appearance of the objects and the background as well as any motion in the video. Srivastava *et al.* point out that it makes optimization easier when target seqence is reversed compared to input sequence, because the model can look at low range correlation.
The model has two different design, one in which the decoder LSTM is contioned on the last generated frame and the other in which it is not. In other word, A conditional decoder receives the last generated

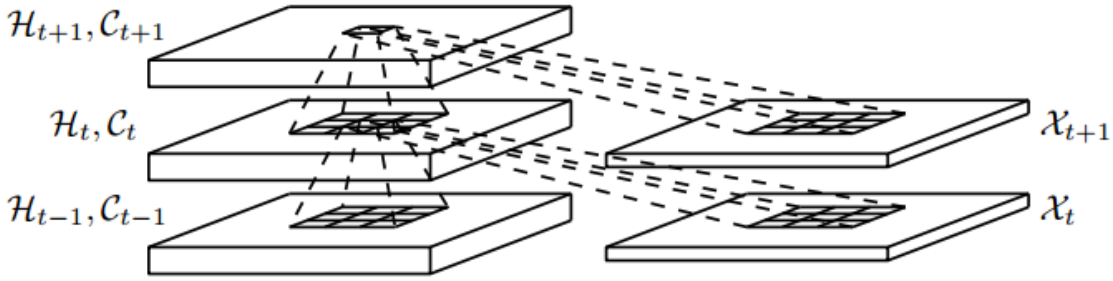

Figure 2: inner structure of convolutional LSTM

Source: adopted from slides of Kaiming He

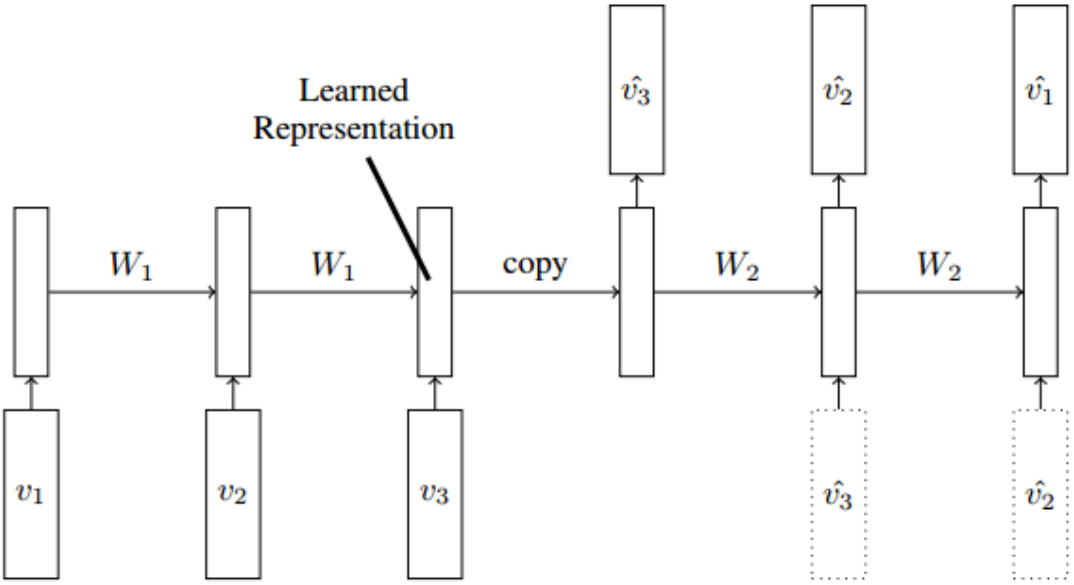

Figure 3: LSTM Autoencoder Model

Source: adopted from slides of Kaiming He

output frame as input, the dotted boxes in figure 3 are present.

3 Experiments

In this lab, experiments are performed to achieve the following objects:
compare the performance of conditional decoder and unconditioned decoder.
check how hyperparameters affect the autoencoder model
compare FC-LSTM and ConvLSTM
check my implementation of FC-LSTM and ConvLSTM

3.1 Datasets

The model is trained on a dataset of moving MNIST digits. Each video is 20 frames long and consist of 2 digits moving inside 64×64 patch. The moving digits are chosen randomly from the MNIST dataset and

placed initially at random locations inside patch. Each digit is assigned a velocity with random direction on a unit circle and uniformly random magnitude over a fixed range. The digits bounce off the edges of the frame and overlap if they are at the same location.

References

[GSK+15] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. LSTM: A search space odyssey. *CoRR*, abs/1503.04069, 2015.

[SCW+15] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 802–810, 2015.

[SMS15] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 843–852, 2015.