Video Description using Deep Learning

Summer Undergraduate Research Award

Suyash Agrawal
2015CS10262
Computer Science
CGPA: 9.87
Mob: 9717060183
cs1150262@iitd.ac.in

Madhur Singhal
2015CS10235
Computer Science
CGPA: 8.66
Mob: 9540972599
cs1150235@iitd.ac.in

Supervisor:-
Subhashis Banerjee
Professor
Department of CSE
suban@cse.iitd.ac.in
IIT Delhi

Prof. S. Arun Kumar
Head of Department
Department of CSE
sak@cse.iitd.ernet.in

1 Introduction

Video Description is the process of discovering knowledge, structures, patterns and events of interest in the video data and describing them in natural language. Video Description is an open problem in computer vision and currently the only source of video description is manual labour.
Video Description has wide variety of applications. It can help visually impaired people "see" the world by describing the scene around them. It has also use in automated surveillance by analyzing the videos in real time and reporting malicious or unusual activities. It can also be used to efficiently index large video databases based upon their content for ease of accessibility.

Figure 1: Sample Video Description

2 Past Work

Much of the prior work in this field is focused on generating natural language descriptions from images. Recently the use of an end to end deep neural network architecture which takes an image as input and generates an english description has been shown to give excellent results. We extended this approach to a Video Description network which takes in a sequence of frames as input and generates an english description of the actions happening in the video. For this we use an encoder-decoder framework widely used in Machine Translation, which maps a variable length input to a variable length output.

3 Data

The data used is a crucial factor in the effectiveness of a complex neural network like ours. We primarily used the following two datasets for training and validation purposes.

1. **Microsoft Research - Video to Text (MSR-VTT)**: The dataset contains 41.2 hours and 200K clip-sentence pairs in total, covering the most comprehensive categories and diverse visual content, and representing the largest dataset in terms of sentence and vocabulary.

2. **Montreal Video Annotation Dataset (M-VAD)**: The M-VAD movie description corpus is another recent collection of about 49,000 short video clips from 92 movies. It is similar to MPII-MD, but only contains AD data and only provides automatic alignment.

Both of these datasets consist of short Youtube video clips (average length near 20 seconds) and english sentences (average length near 12 words) describing the clips. Most of the video clips also have a synchronized audio clip which we utilize in some of our experiments. Empirically we found out that these datasets were big enough for our network to learn proper English language sentence structure and grammar automatically, and the sentences we return are almost always gramatically correct. The datasets also identify a wide host of objects and actions which make the network applicable in general situations.
The main problem with these datasets is that they were not properly sampled and thus there was an inherent bias towards the types of videos seen on Youtube as opposed to those encountered in the real world. For example, the datasets had a large amount of videos of video games which often involve rapidly changing frames and our network began to associate things like car crashes to video game clips.

4 Methods

4.1 Overall Approach

Our approach to video description was motivated by the recent successful approach of using Encoder-Decoder system in Image Description. Our pipeline consisted of a CNN network that extracted feateures from the individual video frames, one LSTM network which encoded these features into a fixed length feature vector and another LSTM network which decoded the feature vector into a natural language description.

4.2 Word Embeddings

To generate sentences we need a vocabulary, we use a vocabulary which consists of about 13000 words as well as punctuation marks. We also add two special tokens ¡Start¿ and ¡End¿ to the vocabulary to represent the beggining and end of a sentence. Since a simple one-hot representation of a word is very sparse and unstructured, we

use a fully connected layer (called the “embedding layer” to convert a word to a dense vector space. We experimented with different embedding methods, including random initialization of a trainable embedding layer and using pretrained GloVe or Word2Vec embeddings. Empirically we found out that using pretrained embeddings gives almost identical results to randomly initializing the embedding layer and training it within our pipeline.
At the output stage we need another matrix to decode the outputs to words and we used another fully connected layer which gives an output of length vocabulary size with each element representing the probability of that word being generated. Note that while it may seem reasonable to use the same weights in the embedding and decoding layers as first sight, such an approach gives poor results in our experiments since the vector space of embedding outputs and the vector space of decoding inputs are different and unrelated to each other.

4.3 Video Feature Extraction

We used Convolutional neural networks for extracting features from video frames. In recent times CNNs have shown amazing results in object detection in images and thus they form a natural choice in extracting features from images. Also, since videos are a sequences of images, we downsample video frames at a fixed frame rate and then individually extract features from each video frame and concantenate them to get the feature vector of the video.
We used Inception V4 net as our choice of CNN for video feature extraction as it has shown the best results in ImageNet challenge and is thus more likely to give better feature extraction from image frames. We also experimented with audio features in the video. Specifially we looked at the MFCC features of the audio and we processed these features along with image features in our encoder network to get better representation of the video. But we later abandoned this approach as it did not show any significant improvement in the results and also added audio dependency to our network which would have made this un-usable in the case of video surveillance and other areas where audio is not available.

4.4 Video Feature Encoding

We used a LSTM based network for encoding feature vectors of individual video frames into a single feature vector for the whole clip. This approach is largely inspired by similar approach in machine translation which also use LSTMs for encoding decoding between languages. Here we used the feature vector of video frames as inputs to LSTM units and the final hidden state of the LSTM network as the video

encoding.
We experimented with various types of encoding networks like:

- Single layer LSTM network
- Multi layered LSTM network (2-5 layers)
- Residual Network Multi Layered LSTM network
- Encoder network with Attention Model

Apart from the above, we also experimented with dropout in between layers and also experimented with audio features concatenated with video features.

4.5 Decoding to Natural Language

The previous stage gives us a fixed length feature vector which represents the whole video clip. We use this feature vector as the initial hidden state of our decoder LSTM network. After this we feed words (embedded in a vector space as described above) one by one into the LSTM, which gives the next word. There is a special ¡Start¿ token which represents the beginning of a sentence and is always the first one to be given as input. The ouput word is deduced from the output of the LSTM network using a fully connected layer with softmax activation, which results in a vector of length equal to the size of our vocabulary and where each element can be interpreted as the probability of that word coming next in the sentence.
Initially we used a greedy approach to sentence formation by taking the word with maximum probability at the output of each step, adding it to our sentence and feeding it as input in the next step. This approach often resulted in convoluted and inaccurate sentences, since some degree of look-ahead is needed for proper sentence structure. Finally we used **Beam Search Decoder** to provide us with a certain amount of look-ahead, and observed our sentences and results both improve dramatically. A Beam Search is a compromise between the totally greedy approach which can be short-sighted and the breadth-first search approach which is not computationally tractable. Instead of choosing the best setence at the end of each step, here we keep a 'beam' of sentences (three sentences in the beam were used in practice). At each step, for each sentence in the beam we consider the top three predictions for the next word and make nine sentences out of those, then we filter down from these nine sentences to three sentences on the basis of the combined probability for the original sentence and the added word.

4.6 Training Phase

4.7 Inference Phase

4.8 Dealing with Overfitting

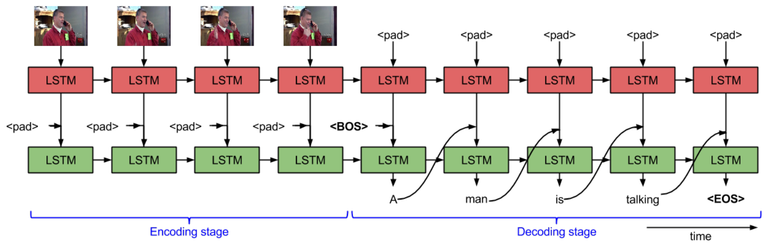

Figure 2: Video description model with 2 LSTM levels

5 Uses and applications

- Assisting visually impaired people to get description of their surroundings, thus enabling them to "see".
- Very useful for automated surveillance and theft detection by being able to analyze large amounts of data which is unfeasible to be done by humans.
- Allowing content based video retrieval by describing the contents of video in textual format which is indexable by web crawlers.
- This can also be used to detect catastrophic events through security cameras like fire breakout, murder etc.
- This project can also be applied in helping robotic vision as this project basically allows one to understand what is happening in the video and thus robots will be able to get a "true" sense of their surroundings.

6 Background

6.1 Deep Learning

Deep Learning is a branch of machine learning in which multiple parameter based models are used in series. In a deep network, there are many layers between the input and output, allowing the algorithm to be executed in multiple processing steps, composed of **multiple linear and non-linear transformations**. At each layer, the signal is transformed by a processing unit, like an artificial neuron, whose parameters are **'learned'** through training. Deep Learning has been shown to excel in tasks where the goal is to find **intuitive** patterns in the data.[8] In particular, in the field of Computer Vision, deep networks are increasingly used to extract **feature descriptions and inter-relationships between features** from images.[10]

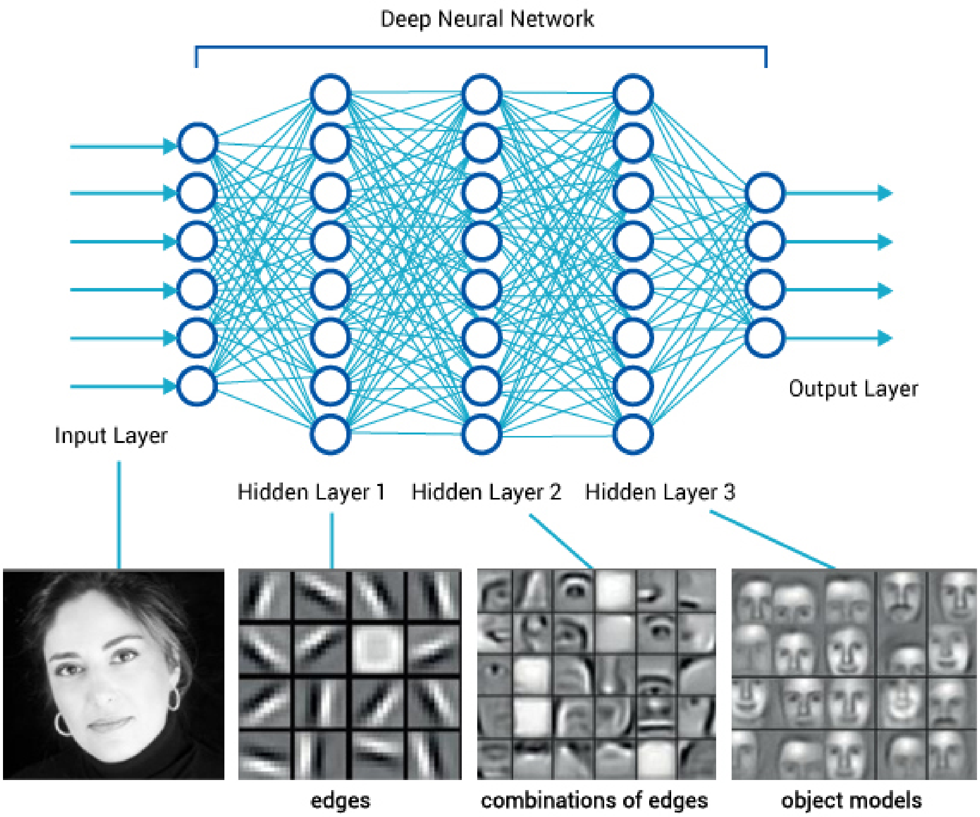

Figure 3: Illustration of Deep Learning as applied to Vision

6.2 Convolutional Neural Networks

Convolutional Neural Networks (CNN, or ConvNet) are a type of feed-forward artificial neural network in which the connectivity pattern between the neurons is inspired by the organization of the animal visual cortex. Individual cortical neurons respond to stimuli in a restricted region of space known as the **receptive field**. The receptive fields of different neurons partially overlap such that they tile the visual field. The response of an individual neuron to stimuli within its receptive field can be approximated mathematically by a **convolution operation**. A Convolutional Neural Network consists of the following layers.[3]

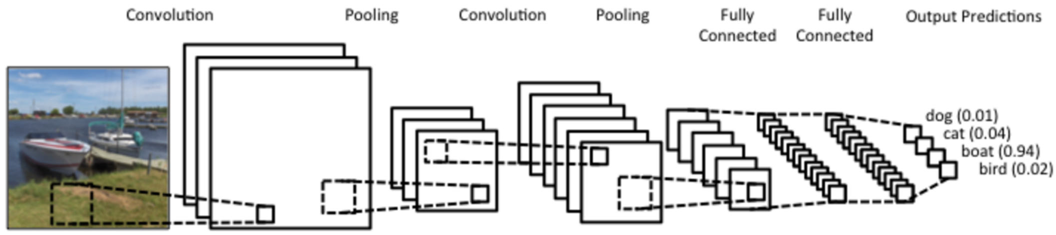

Figure 4: A Typical Convolutional Neural Network

6.2.1 Convolutional Layer

The convolution layer is the core building block of a CNN. The layer's parameters consist of a set of **learnable filters** (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter.[10] As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input.

6.2.2 Max Pooling Layer

It is common to periodically insert a Pooling layer in-between successive Conv layers in a ConvNet architecture. Its function is to **progressively reduce the spatial size** of the representation to reduce the amount of parameters and computation in the network, and hence to also control over-fitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the max operation.[10]

6.2.3 Fully-Connected Layer

Finally, after several convolutional and max pooling layers, the high-level reasoning in the neural network is done via fully connected layers. Neurons in a fully connected layer have **full connections to all activations** in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.[11] Thus output of the fully connected layer is a vector with elements representing the 'probability' (not in a strictly statistical sense) of the image containing specific objects or actions.

6.3 Long Short Term Memory Networks

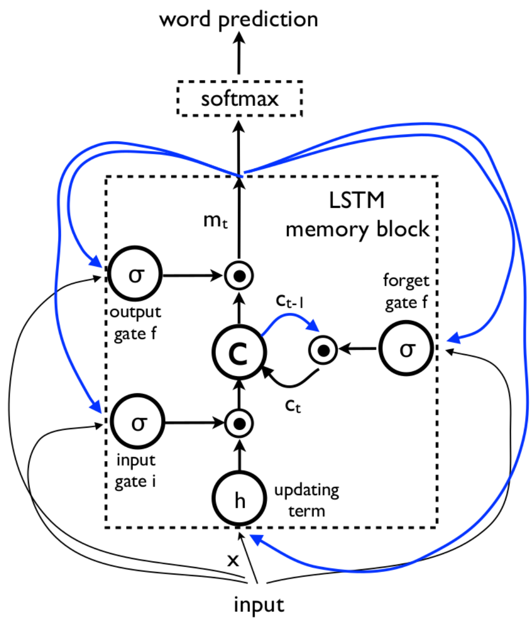

Figure 5: A single LSTM unit

Long Short Term Memory Networks are a type of Recurrent Neural Networks. These networks are based upon recursion, so that variable length inputs can be handled easily and sequential information can be processed with better results. LSTM's are specifically used for making RNNs learn long term patterns since traditional RNNs tend to **favour short term temporal dynamics**. It can be difficult to train traditional RNNs to learn long-term dynamics, likely due in part to the **vanishing and exploding gradients problem** that can result from propagating the gradients down through the many layers of the recurrent network, each corresponding to a particular time step[7]. LSTMs provide a solution by incorporating memory units that explicitly allow the network to learn when to "forget" previous hidden states and when to update hidden states given new information[4].

6.4 Training Deep Neural Networks

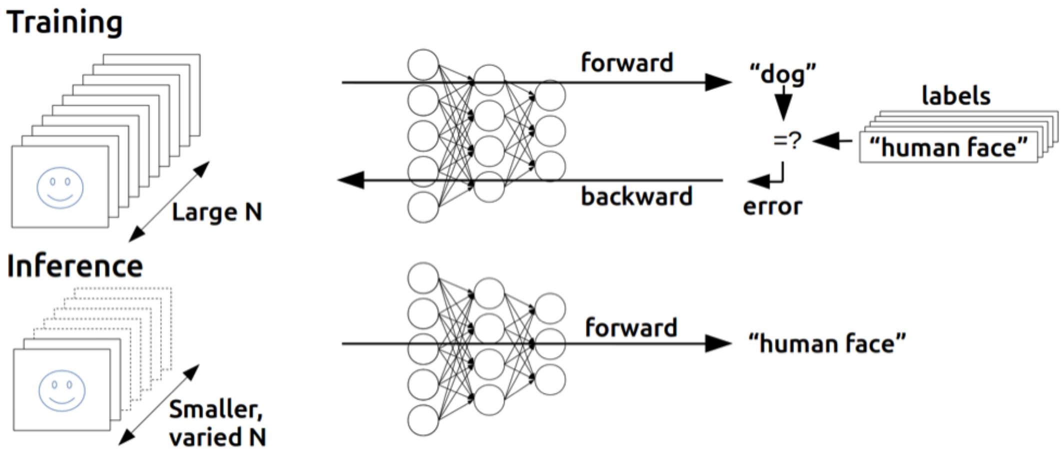

Figure 6: Training and Inference Processes

A Deep Neural Network is at it's core a parameter based function. All of these parameters are **trained** automatically from inputs and expected output tuples (training data). The training process revolves around minimizing a particular cost function using methods like **Stochastic gradient descent**. The input is given to the network in a feed forward fashion and the parameters are modified from the last layer to the first **(Backpropagation)**. Neural Networks, by design, require huge amounts of training data and take a large time to get trained. For some perspective, most current state of the art image classifiers have $>$ 100 million parameters and are trained on more than 1.2 million images.

6.5 Finetuning

Fine-tuning a network is a procedure based on the concept of **transfer learning**. We start training a CNN to learn features for a broad domain with a classification function targeted at minimizing error in that domain. Then, we replace the classification function and **optimize the network** again to minimize error in another, more specific domain. Under this setting, we are transferring the features and the parameters of the network from the broad domain to the special one.[9] In our project we will need to use the pre-trained image classification models to actually decode individual frames of the video, thus we are planning to **fine-tune those models with respect to the output of our LSTMs**.

References

[1] Subhashini Venugopalan *Natural Language Video Description using Deep Recurrent Neural Networks*, 2015.

[2] Venugopalan, Subhashini and Rohrbach, Marcus and Donahue, Jeff and Mooney, Raymond and Darrell, Trevor and Saenko, Kate *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015

[3] Oriol Vinyals and Alexander Toshev and Samy Bengio and Dumitru Erhan *Show and Tell: A Neural Image Caption Generator* 2014.

[4] Wojciech Zaremba, Ilya Sutskever *Learning to Execute* ICLR 2015

[5] Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, Aaron Courville *Describing Videos by Exploiting Temporal Structure* ICCV 2015

[6] Aishwarya Agrawal, Jiasen Lu , Stanislaw Antol, Margaret Mitchell, C. Lawrence Zitnick, Dhruv Batra, Devi Parikh *VQA: Visual Question Answering* 2016

[7] Jeff Donahue, Lisa Anne Hendricks, Marcus Rohrbach, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, Trevor Darrell *Long-term Recurrent Convolutional Networks for Visual Recognition and Description* 2016

[8] Bengio, Yoshua; LeCun, Yann; Hinton, Geoffrey *Deep Learning* 2015

[9] Angie K. Reyes, Juan C. Caicedo and Jorge E. Camargo *Fine-tuning Deep Convolutional Networks for Plant Recognition* LifeCLEF 2015

[10] Andrej Karpathy *CS231n Convolutional Neural Networks for Visual Recognition* http://cs231n.github.io/convolutional-networks/

[11] Santanu Chaudhury, Anupama Mallik, Hiranmay Ghosh *Multimedia Ontology: Representation and Applications*