Internship Report
Deep Learning for Activity Classification in Egocentric Videos

Vardaan Pahuja
Mentor: Dr. Arijit Biswas and Dr. Om Deshmukh
Xerox Research Center India

Email: vardaanpahuja@iitkgp.ac.in

Abstract

The problem of classifying pre-segmented videos of daily living activities is addressed.We use CNN features to train an object recognition model and subsequently Temporal Pyramid based features are used in SVMs.In a second approach, LSTM based architecture is used to model the temporal context is videos.

1 Introduction

Activity recognition is a classic task in computer vision. Traditionally, the above limitations have been addressed by using actor-scripted video footage of posture-defined action categories such as skipping or jumping. Such categories maybe artificial because they tend not be functionally defined, a core aspect of human movement. We focus on the problem of detecting activities of daily living (ADL) from first-person wearable cameras such as Google Glass. The dataset contains of 20 videos ranging from 30-60 minutes long of people performing unscripted, everyday activities. The dataset is annotated with activities (along with segmentation) and object tracks. The goal is to classify these video segments into a set of pre-defined activity categories.

2 Applications

Tele-rehabilitation: A variety of clinical benchmarks used to evaluate everyday functional activities such as picking up a telephone, drinking from a mug, and turning on a light switch, etc for patients suffering from neural diseases. These evaluations are currently done in the hospital, but a computer-vision system capable of analyzing such activities would revolutionize the rehabilitative process, allowing for long-term, at-home monitoring.
Life-logging: It aids in continual logging of visual personal histories for memory enhancement for patients with memory-loss.

3 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is comprised of one or more convolutional layers (often with a subsampling step) and then followed by one or more fully connected layers as in a standard multilayer neural network. The architecture of a CNN is designed to take advantage of the 2D structure of an input image (or other 2D input such as a speech signal). This is achieved with local connections and tied weights followed by some form of pooling which results in translation invariant features. Another benefit of CNNs is that they are easier to train and have many fewer parameters than fully connected networks with the same number of hidden units.

3.1 Architecture

A CNN consists of a number of convolutional and subsampling layers optionally followed by fully connected layers. The input to a convolutional layer is a m x m x r image where m is the height and width of the image and r is the number of channels, e.g. an RGB image has r=3. The convolutional layer will have k filters (or kernels) of size n x n x q where n is smaller than the dimension of the image and q can either be the same as the number of channels r or smaller and may vary for each kernel. The size of the filters gives rise to the locally connected structure which are each convolved with the image to produce k feature maps of size m–n+1. Each map is then subsampled typically with mean or max pooling over p x p contiguous regions where p ranges between 2 for small images (e.g. MNIST) and is usually not more than 5 for larger inputs. Either before or after the subsampling layer an additive bias and sigmoidal nonlinearity is applied to each feature map. The figure below illustrates a full layer in a CNN consisting of convolutional and subsampling sub-layers. Units of the same color have tied weights.

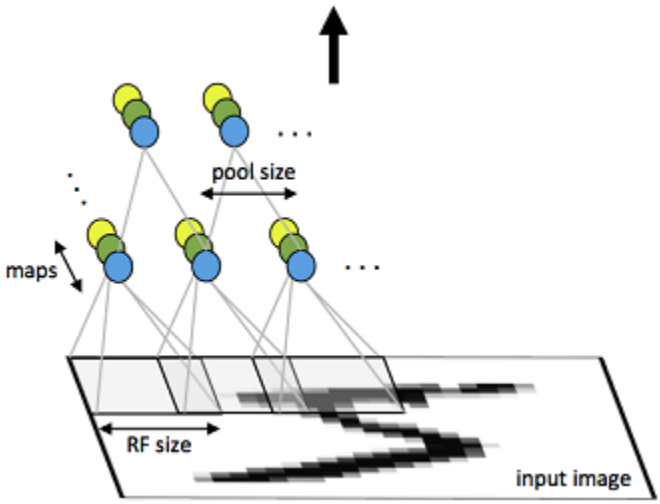

Figure 1: First layer of a convolutional neural network with pooling. Units of the same color have tied weights and units of different color represent different filter maps.

We have used the CNN architecture AlexNet the details of which are described as follows.

The first convolutional layer filters the 224 x 224 x 3 input image with 96 kernels of size 11 x 11 x 3 with a stride of 4 pixels (this is the distance between the receptive field centers of neighboring neurons in a kernel map). The second convolutional layer takes as input the (response-normalized and pooled) output of the first convolutional layer and filters it with 256 kernels of size 5 x 5 x

48. The third, fourth, and fifth convolutional layers are connected to one another without any intervening pooling or normalization layers. The third convolutional layer has 384 kernels of size 3 x 3 x 256 connected to the (normalized, pooled) outputs of the second convolutional layer. The fourth convolutional layer has 384 kernels of size 3 x 3 x 192 and the fifth convolutional layer has 256 kernels of size 3 x 3 x 192. The fully-connected layers have 4096 neurons each.

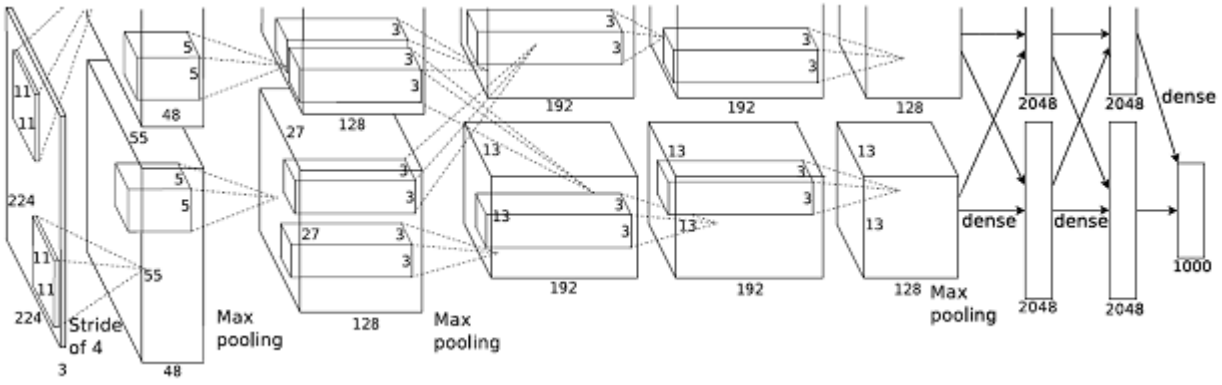

Figure 2: AlexNet Architecture

4 Training of Object Recognition Model

The AlexNet CNN pre-trained on ImageNet dataset which contains about 1.2 million natural scene images because the generic image features learned till fc7.The fc7 features are reused so as to avoid overfitting due to less amount of training data than required for training the CNN on video objects from scratch. A 1000 neuron softmax layer is replaced by 27 unit softmax (26 object categories + background) layer corresponding to our problem and then fine tuning is done using the object proposals extracted for 10000 iterations.

4.1 Training Data for Fine-tuning CNN

Region proposals of different objects in a frame of video are generated using Selective Search Technique. Around 2000 proposals per frame are generated most of which are negatives (do not correspond to real objects). A few proposals (around 10 per frame) are positive proposals having IoU (intersection over Union) overlap ratio with ground truth annotations ≥ 0.5. The training is performed on about 28k images consisting of batches of 140 consisting of 130 positive proposals and 10 negative proposals. The positive proposals are entirely used while a random sampling is performed to select the negative proposals. The batch size proportion is set such that each object class including the background has roughly same number of training examples. Using all background proposals leads to the object recognition model being more biased towards predicting negatives.

5 Temporal Pyramids based Approach

- Bag of Features representation simply averages features corresponding to each frame ignoring temporal information.
- We represent features in a temporal pyramid, where the top level j = 0 is a histogram over the full temporal extent of a video clip , the next level is the concatenation of two histograms obtained by temporally segmenting the video into two halfs, and so on.
- A coarse-to-fine representation is obtained by concatenating all such histograms together.

$$x_i^{j,k} = \frac{2^{j-1}}{T} \sum_{t \in T^{j,k}} f_i^t \; \forall \; k \; \in \{1...2^j\}$$

  where $T^{j,k}$ is the temporal extent of the k^{th} segment on the j^{th} level of the pyramid and $x_i^{j,k}$ is the feature for the i^{th} object detector on that segment.
- The features corresponding to the active objects are biased by a threshold greater than passive objects as they are more likely to represent the prominent objects in the video frame.
- A two-level temporal pyramid is constructed in which at each level, the frames corresponding to the span of the corresponding level segment are averaged.
- For each activity the feature vectors of different segments are concatenated to obtain the feature vector.
- A linear kernel SVM is used to classify the remaining 14 videos (object detection is trained on first 6 videos only).

6 Activity Classification using LSTMs

Recently Long Short Term Memory(LSTM) units have been widely successful in modelling sequence based information both in input and output.

Recurrent Neural Networks(RNNs) can learn complex temporal dynamics by mapping input sequences to a sequence of hidden states, and hidden states to outputs via the following recurrence equations.

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

$$z_t = g(W_{hz}h_t + b_z)$$

where g is an element-wise non-linearity, such as a sigmoid or hyperbolic tangent, x_t is the input, $h_t \in R^N$ is the hidden state with N hidden units, and y_t is the output at time t. For a length T input sequence $x_1, x_2, ..., x_T$, the updates above are computed sequentially as h_1 (letting $h_0 = 0$),$y_1, h_2, y_2, ..., h_T, y_T$.

In addition to a hidden unit $h_t \in R^N$, the LSTM includes an input gate $i_t \in R^N$, forget gate $f_t \in R^N$, output gate $o_t \in R^N$, input modulation gate $g_t \in R^N$, and memory cell $c_t \in R^N$. The memory cell unit c_t is a summation of two things: the previous memory cell unit c_{t-1} which is modulated by f_t, and g_t, a function of the current input and previous hidden state, modulated by

the input gate it. i_t and f_t can be thought of as knobs that the LSTM learns to selectively forget its previous memory or consider its current input. Likewise, the output gate o_t learns how much of the memory cell to transfer to the hidden state. These additional cells enable the LSTM to learn extremely complex and long-term temporal dynamics the RNN is not capable of learning. Additional depth can be added to LSTMs by stacking them on top of each other, using the hidden state of the LSTM in layer $l-1$ as the input to the LSTM in layer l.

The LSTM updates for timestep t given inputs x_t, h_{t-1}, and c_{t-1} are:

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\
g_t &= \sigma(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \varphi(c_t)
\end{aligned}
$$

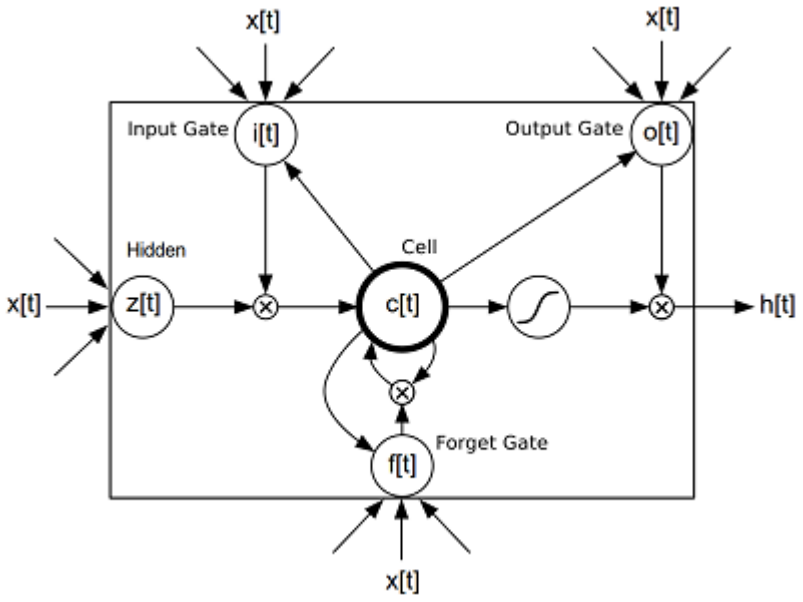

Figure 3: A single LSTM cell

6.1 LSTM Architecture

The architecture consists of a single layer LSTM with 256 hidden units followed by two fully connected non-recurrent layers of size 256 and 18 respectively followed by 18 dimensional softmax layer. The input to LSTM is a 26- dimensional vector for each frame and output is 18 dimensional vector representing the Negative log likelihood of the frame belonging to each of 18 video categories. The predictions corresponding to each frame are averaged to obtain a vector used for predicting the class of activity.

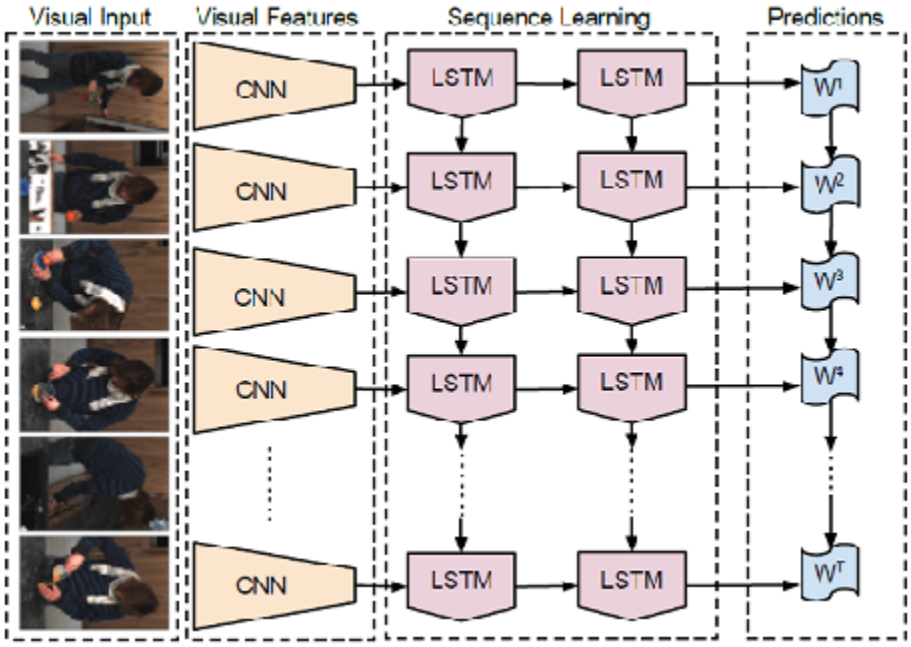

Figure 4: LSTM architecture for classification

6.2 Experiments on LSTM Classification Model

We tried with different number of hidden units for each layer in LSTM based model(64,256,512) and found best results with the 256 units model. Another variation is to take average of confidence scores at each frame instead of last frame.The averaging approach yields better results. Non-uniform distribution of different activities causes the model to be biased towards more frequently occurring activities.Also, the training data is insufficient compared to the large number of model parameters.In order to overcome these limitations, we divided each segments into sub-segments of 5 seconds duration each. Then we sample a number uniformly ranging from half of total number of segments and total number of segments and take these number of segments in topological ordering for training.The number of times sampling is done is in inverse proportion to the frequency of occurrence of that activity category.Thus, it results in a approximately uniform distribution of all activities.This gives encouraging results than the previous approach and further work is ongoing to improve the classification accuracy.

Table 1: Activity classification accuracy with 2 layer LSTM with 512 units in each layer

No. of Iterations	**Training Set Accuracy**	**Validation Set Accuracy**
10	13.64	21.18
20	21.18	18.72
30	22.72	15.27
40	27.27	15.27
50	31.82	15.27
60	31.82	13.79

Table 2: Activity classification accuracy with 2 layer LSTM with 256 units in each layer

No. of Iterations	**Training set Accuracy**	**Validation set Accuracy**
10	13.64	12.80
20	20.0	20.69
30	22.72	17.24
40	21.82	15.27

Table 3: Activity classification accuracy with 2 layer LSTM with 64 units in each layer

No. of Iterations	**Training set Accuracy**	**Validation set Accuracy**
10	13.63	12.32
20	13.63	12.32
30	13.63	12.32
40	13.63	12.32

Table 4: Activity classification accuracy with 2 layer LSTM with 256 units in each layer and averaging approach

No. of Iterations	**Training Accuracy**	**Validation Accuracy**
40	14.54	12.80
50	15.45	14.77
54	17.27	15.76
56	18.18	16.26
58	17.27	17.73
60	16.36	19.70
64	16.36	23.15
68	18.18	23.64
70	19.09	24.14
72	18.18	24.13
76	19.09	22.66
80	19.09	21.67

7 Results

7.1 Results for Object Detection Model

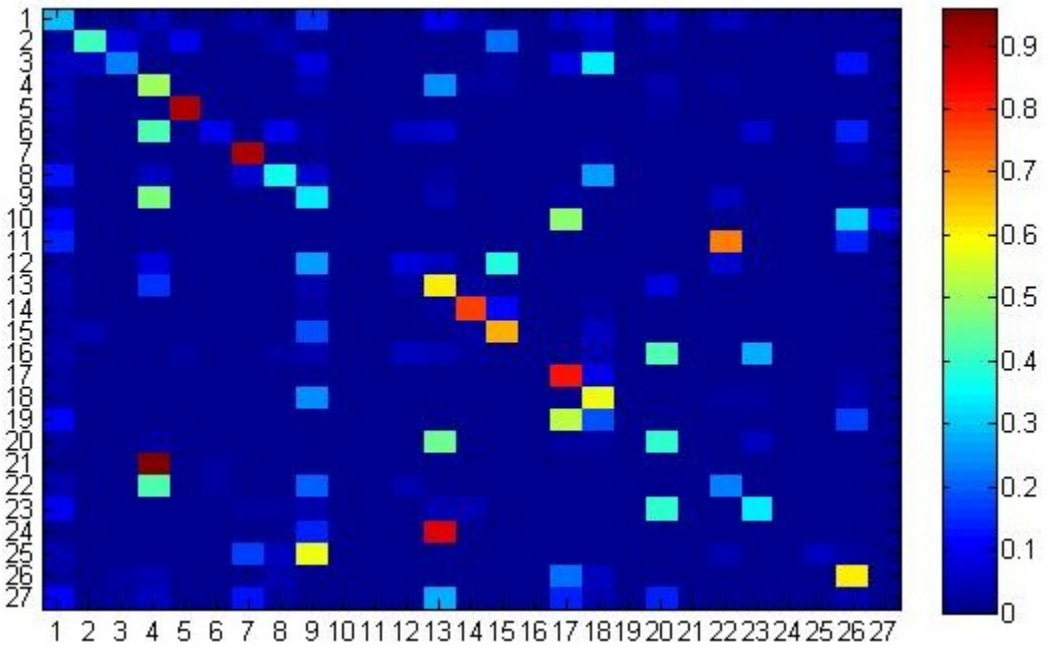

Figure 5: Confusion Matrix for Object classification on 26 objects + background using AlexNet CNN. Object Classification Accuracy = 44 %

- The AlexNet CNN model results in a 44 % object detection accuracy on the validation set.
- We get a 27- dimensional vector(26 object categories, 1 background) representing the confidence scores of the input proposal for each of object categories.
- For each frames, in order to obtain the confidence score for each object category, we took the absolute maximum among all proposals of that frame for each object category.
- The max 27-dim vector acts as a feature for each frame.

Table 5: Object categories for Egocentric videos

Object ID	**Object Name**
1	Fridge (Active)
2	Microwave (Active)
3	Mug/cup (Active)
4	Oven/Stove (Active)
5	Soap Liquid(Active)
6	Bed
7	Book
8	Bottle
9	Cell
10	Dental Floss
11	Detergent
12	Dish
13	Door
14	Fridge
15	Kettle
16	Laptop
17	Microwave
18	Monitor
19	Pan
20	Pitcher
21	Soap Liquid
22	Tap
23	Tea bag
24	Toothpaste
25	TV
26	TV remote
27	Background

7.2 Results for Activity Classification using Temporal Pyramids

The temporal Pyramid based feature in SVM gives 44.03 % accuracy which is better than the current state of the art(36.8 %) on the same dataset.

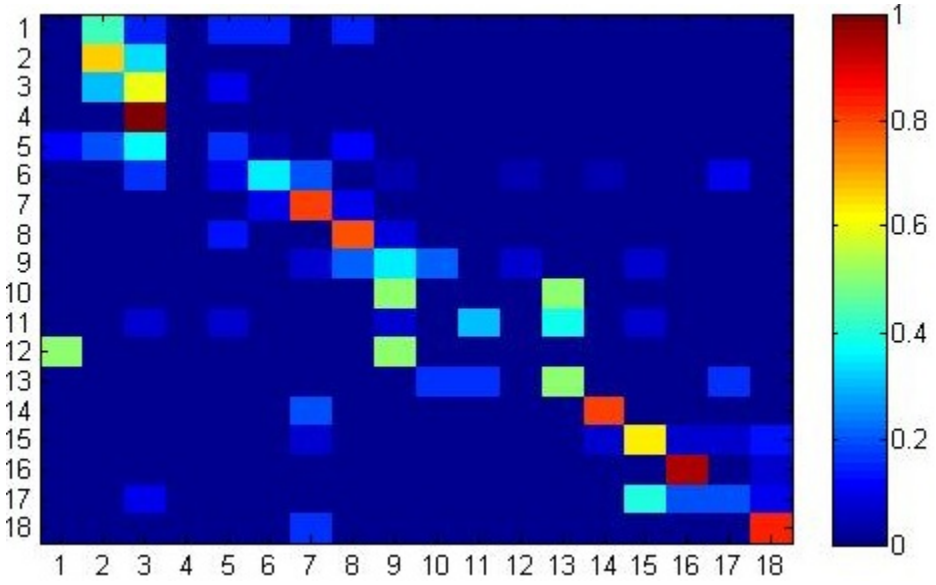

Figure 6: Confusion matrix for Temporal pyramid based video descriptors. Segment classification accuracy = 44.03 %

Table 6: Activity Categories in Egocentric Videos

Activity ID	**Activity description**
1	combing hair
2	make up
3	brushing teeth
4	dental floss
5	washing hands/face
6	drying hands/face
7	laundry
8	washing dishes
9	making tea
10	making coffee
11	drinking water/bottle
12	drinking water/tap
13	making cold food/snack
14	vaccuming
15	watching tv
16	using computer
17	using cell
18	reading book

7.3 Results for Activity Classification using LSTM architecture

Using the LSTM based approach, the maximum accuracy found so far is 24.14 %. using 2 layer LSTM architecture with 256 hidden units in each layer and use of averaging approach to make predictions. The low accuracy is attributed to the vanishing gradient problem in extremely long temporal frame sequences. Further research is needed to make this architecture perform better.