proposal.md

Machine Learning Engineer Nanodegree

Capstone Proposal

Mrinal Jain
December 8th, 2017

Human Activity Recognition using Deep Learning

Domain Background

Introduction to Deep Learning

Deep learning is a sub-field of machine learning concerned with algorithms inspired by the structure and functionality of a human brain. Our human brain contains billions of cells (called neurons) which form a complex network (a neural network), that helps us perform our day-to-day tasks. These models are known as *artificial neural networks* and over the time, a number of variations of these networks have been implemented for a variety of different problems. Deep learning models give state-of-the-art performance in highly complex problems (like speech recognition, image recognition, object detection and many more). These problems may look very trivial to us (humans), but for a computer, they are extremely hard.

Video Recognition ?

Video Recognition is one such problem domain. But before moving on to video recognition, I would like to discuss some basic concepts.

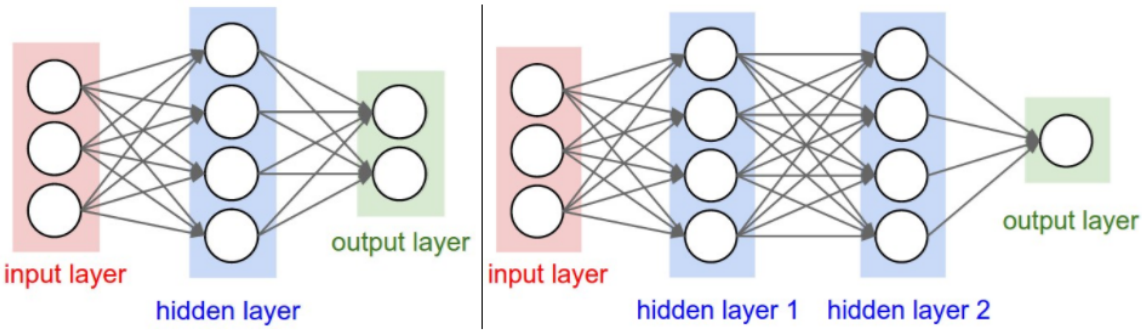
Intuitively, videos are nothing but a *running sequence of images*. Each individual image in a video is known as a *frame*. So, we can say that a video is composed of multiple frames, stacked after one another. Now, an image (technically a *digital image*) is a 2-d array of **pixels**. These pixels may represent gray levels, intensity values etc., that make up the image. Also, if we are dealing with color images, there are three 2-dimensional arrays of these pixels, one for each of the 3 channels - Red(R), Green(G) and Blue(B).

The problem can be written as - given a set of labelled videos, train a model so that it can give a label/prediction for a new video. Here, the label might represent what is being performed in the video, or what the video is about.

Why Deep Learning ?

We already know that neural networks perform very well for image recognition. In particular, a specific type of neural networks called Convolutional Neural Networks (CNNs) are best suited for the task of image recognition. I will now explain how the approach of convolutional neural networks differ from that of traditional neural networks.

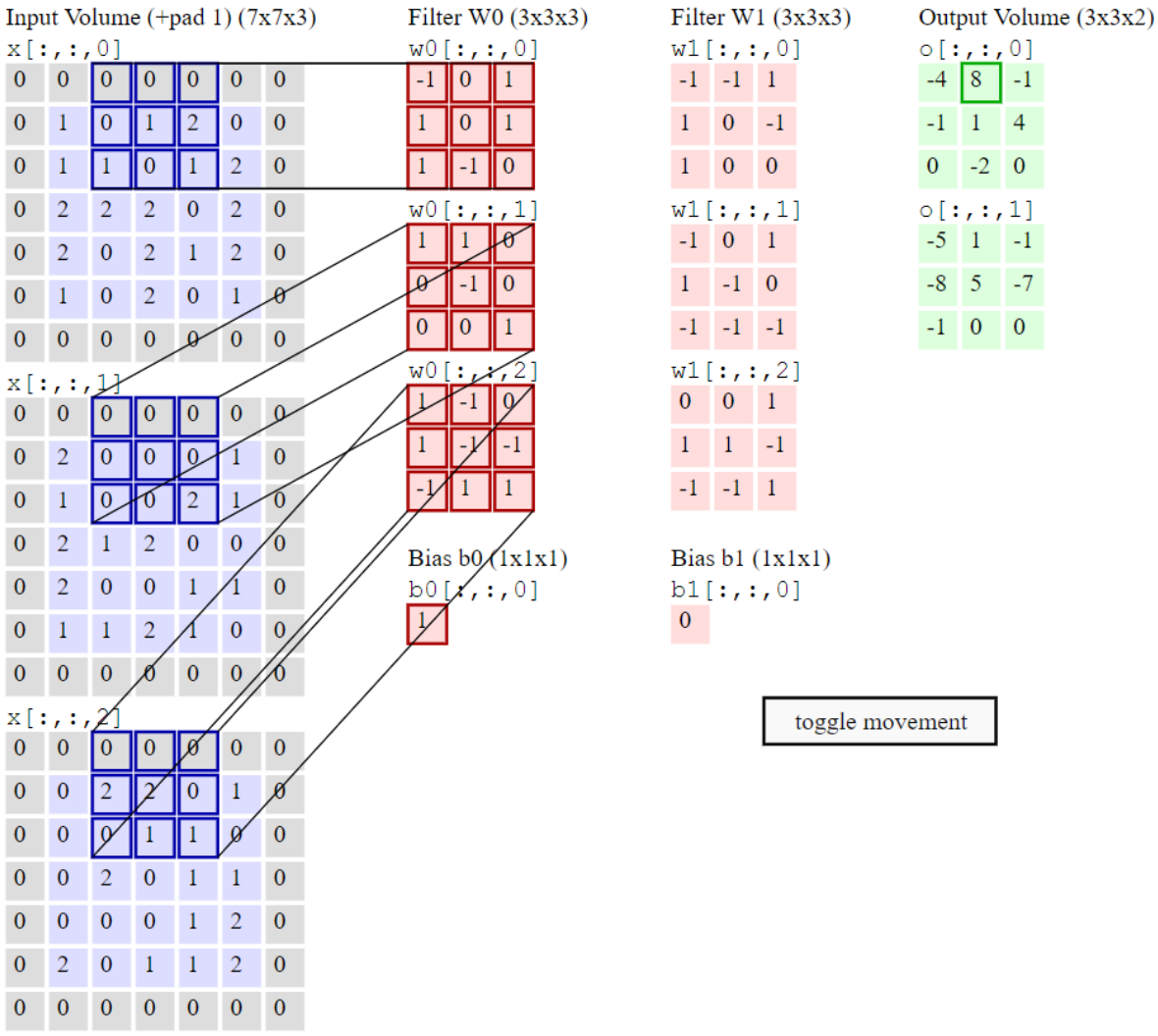
Traditional neural networks
The image is flattened into a 1-dimensional array, and this array is given as the input to our neural network. The problem with this approach is that the spatial pattern of the pixels (their position in their 2-d form) is not taken into account. Also, suppose we have an image whose dimension is 256x256 pixels. The input vector will then comprise of 65,536 nodes, one for each pixel in the image. That's a very large input vector, which could make the weight matrix very large and in turn, make the model very computationally intensive. And even after being so complex, the network would not be able to give any significant accuracy. As a result, this approach was not suited well for tasks like image recognition.



Left: A 2-layer Neural Network (one hidden layer of 4 neurons (or units) and one output layer with 2 neurons), and three inputs.
Right: A 3-layer neural network with three inputs, two hidden layers of 4 neurons each and one output layer. Notice that in both cases there are connections (synapses) between neurons across layers, but not within a layer.

Convolutional Neural Networks

The image is divided into regions, and each region is then assigned to different hidden nodes. *Each hidden node finds pattern in only one of the regions* in the image. This region is determined by a *kernel* (also called a filter/window). A filter is convolved over both *x-axis* and *y-axis*. Multiple filters are used in order to extract different patterns from the image. The output of one filter when convolved throughout the entire image generates a 2-d layer of neurons called a feature map. Each filter is responsible for one features map. These feature maps can be stacked into a 3-d array, which can then be used as the input to the layers further. This is performed by the layer known as *Convolutional layer* in a CNN. These layers are followed by the *Pooling layers*, that reduce the spatial dimensions of the output (obtained from the convolutional layers). In short, a window is slid in both the axes and the max value in that filter/window is taken (Max-Pooling layer). Sometimes Average pooling layer is also used where the only difference is to take the average value within the window instead of the maximum value. Therefore, *the convolutional layers increase the depth of the input image, where as the pooling layers decreases the spatial dimensions (height and width)*. The importance of such an architecture is that it *encodes the content of an image* that can be flattened into a 1-dimensional array.



I have given a very brief introduction to the above mentioned concepts. For more in-depth details, you may reference the following:

- What is Deep Learning ?
- Intro to Neural Networks
- Convolutional Neural Networks

Problem Statement

There are potentially a lot of applications of video recognition such as:

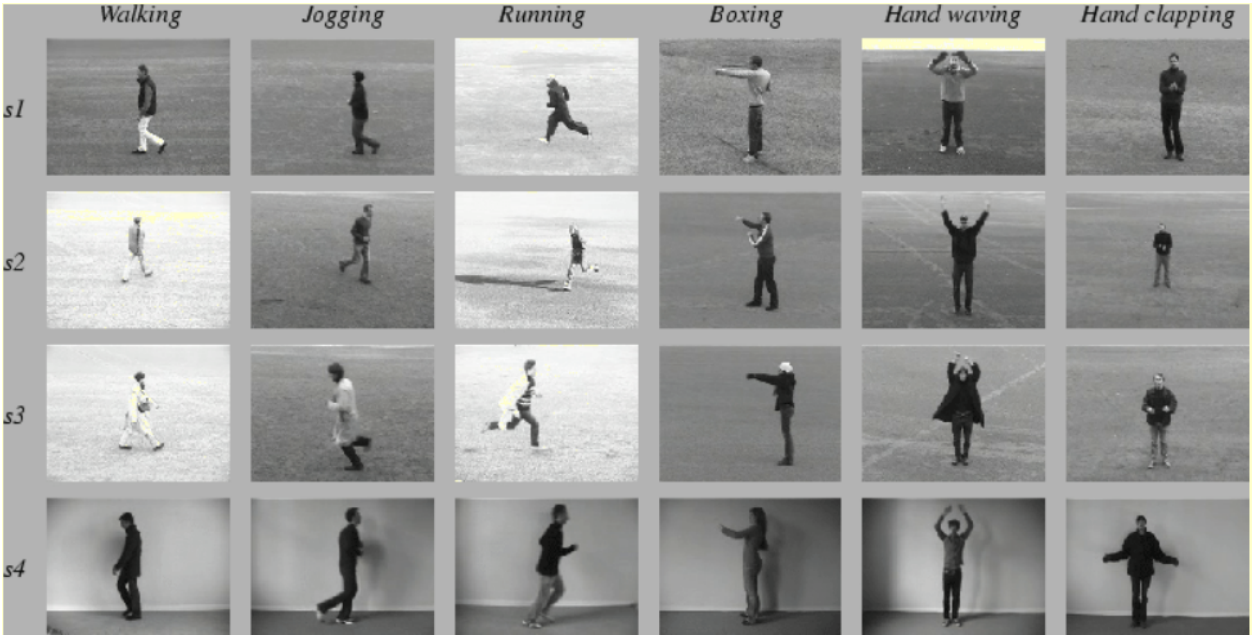
- *Real-time tracking of an object*: This could be very helpful for tracking the location of an object (like a vehicle) or a person from the video recorded by a CCTV.
- *Learning the patterns involved in the movement of humans*: If we are able to create a model that can learn how we (humans) perform various activities (like walking, running, exercising etc.), we can use this model for proper functioning of the movement mechanisms in autonomous robots.

The aim of this project is to create a model that can identify the basic human actions like running, jogging, walking clapping etc. The model will be given a set of videos, where in each video, a person will be performing an action. The label of a video will be the action that is being performed in that particular video. The model will have to learn this relationship, and then it should be able to predict the label of an input (video) that it has never seen. Technically, the model would have to learn to differentiate between various human actions, given some examples of these actions on which the model will train.

Datasets and Input

The dataset can be obtained here - Recognition of Human Actions

The video database containing six types of human actions *(walking, jogging, running, boxing, hand waving and hand clapping)* performed several times by 25 subjects in four different scenarios: outdoors *s1*, outdoors with scale variation *s2*, outdoors with different clothes *s3* and indoors *s4*. The videos were captures at a frame rate of *25fps* and each frame was down-sampled to the resolution of 160x120 pixels.



Further details and instructions to download the dataset is mentioned in the README.md file.

Citation
Recognizing Human Actions: A Local SVM Approach

Solution Statement

The primary model that will be used is a Convolutional Neural Network, where the input to the model will be a 3-dimensional stack of frames (a video). This would be a labelled input by which the model will train.

But, some pre-processing of the input data needs to be done before we send in that data to our model for training/testing. This would include:

- Reading in the video, frame by frame.
- The videos were captured at a frame rate of *25fps*. This means that for each second of the video, there will be 25 frames. We know that within a second, a human body does not perform very significant movement. This implies that most of the frames (per second) in our video will be redundant. Therefore, only a subset of all the frames in a video needs to be extracted. This will also reduce the size of the input data which will in turn help the model train faster and can also prevent over-fitting.
  Different strategies would be used for frame extraction like:
  - Extracting a **fixed number of frames from the total frames** in the video: say only the first 200 frames (i.e., first 8 seconds of the video).
  - Extracting a **fixed number of frames each second** from the video: say we need only 5 frames per second from a video whose duration is of 10 seconds. This would return a total of 50 frames from the video. This approach is better in the sense that we are extracting the frames sparsely and uniformly from the entire video.
- Each frame needs to have the same spatial dimensions (height and width). Hence each frame in a video will have to be *resized* to the required size.
- In order to simplify the computations, the frames can be converted to gray scale.
- **Normalization**: The pixel values ranges from 0 to 255. These values would have to be normalized in order to get a better performance from our network. Different normalization techniques can be applied like:
  - *Min-max Normalization*: Get the values of the pixels in a given range (say 0 to 1)
  - *Z-score Normalization*: This basically determines the number of standard deviations from the mean a data point is.

We would finally get a 5-d tensor of shape - (`< number of videos >`, `< number of frames >`, `< width >`, `< height >`, `< channels >`)

- `channels` can have the value 1 (gray scale) or 3 (RGB)
- `number of frames` : the extracted frames (will have to be the same for each video)

Model Architecture
The model will be a combination of Convolutional and Max-pooling layers, that would make the input array deeper and deeper, and decrease the spatial dimensions. These layers will be followed be a set of fully connected layers, where the final (output) layer will have 6 neurons, one for each of the described actions. The model will give a probability for the input video to be of a particular category. The class label with the highest probability will be the predicted label of that video.

The whole dataset will be divided into training and testing data. The model will be trained on the training data and will be tested on the test data, that it would have never seen before.

Benchmark Model

The existing models use the notion of local features in space-time to capture and describe local events in a video. The general idea is to describe such events is to define several types of image descriptors over local spatio-temporal neighborhoods and evaluate these descriptors in the context of recognizing human activities. These points have stable locations in space-time and provide a potential basis for part-based representations of complex motions in video.

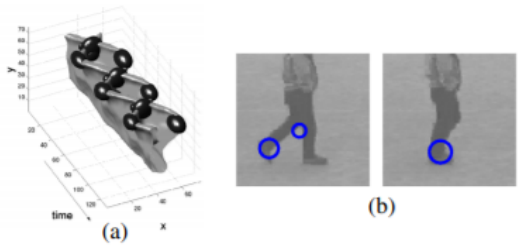


Figure 1. Local space-time features detected for a walking pattern: (a) 3-D plot of a spatio-temporal leg motion (up side down) and corresponding features (in black); (b) Features overlaid on selected frames of a sequence.

These models are able to achieve an overall recognition rate of about 80-85%. I intend to develop a model with an approach that is fundamentally different from that of these existing models.

References:

1. Recognizing Human Actions: A Local SVM Approach
2. Local Descriptors for Spatio-Temporal Recognition

Evaluation Metrics

Once the model has been trained, it will be tested using the testing data.

`Accuracy` will be used for evaluating the performance of the model on the test data.
`Confusion Matrix` will be created in order to compare the model with the Benchmark model.

A *confusion matrix* is used to describe the performance of a classification model.
Example -

Total	Predicted: Yes	Predicted: No
Actual: Yes	TP	FN
Actual: No	FP	TN

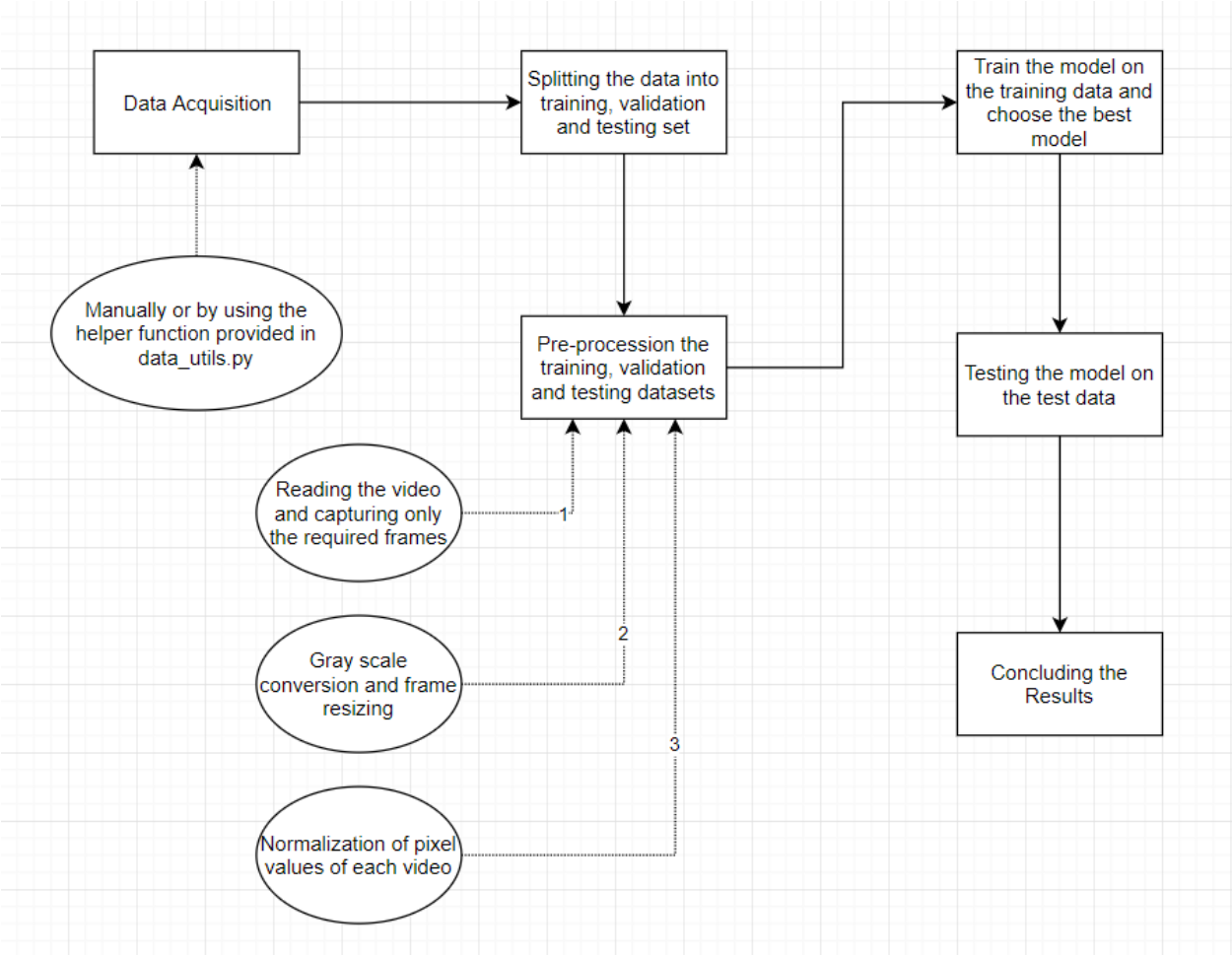
True Positives (TP): The model predicted 'YES' and the prediction was *True*
False Positives (FP): The model predicted 'YES' but the prediction was *False*
False Negatives (FN): The model predicted 'NO' but the prediction was *False*
True Negatives (TN): The model predicted 'NO' and the prediction was *True*

```
Accuracy = (TP + TN)/Total
```

References:

1. Confusion Matrix Terminology

Project Design



- Firstly, we obtain the data and extract it in the required format. Follow the instructions here in order to get the data.
- Next, we have to *read in the videos* and extract only a selected frames, in a specific format (say the spatial dimensions should be 128x128 pixels, in gray scale), and create a *5-dimensional tensor* which would be used as the input to our model.

- Ideally, this is to be done by a *single wrapper function*, where the function will receive the paths (absolute paths) of the videos to be read, and it should return the required 5-d tensor.

- The label of each video will be in range (0...5), with 0 for `boxing`, 1 for `handclapping`, and so on. These labels cannot be used directly. We would apply **One-hot Encoding** on the labels.

  One-hot Encoding: It is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction.
  Example -

  Before one-hot Encoding

CompanyName	Categoricalvalue	Price
VW	1	20000
Acura	2	10011
Honda	3	50000
Honda	3	10000

  After one-hot Encoding

VW	Acura	Honda	Price
1	0	0	20000
0	1	0	10011
0	0	1	50000
0	0	1	10000

  For further reference:

  - One-hot Encoding

- The whole data is divided into **training, validation and testing set**. The training set is used to train the model. After each epoch of training, the model is evaluated on the validation set. Once the training is completed, *the model that performed the best on the validation set is loaded*. This model is then tested on the test data using the above mentioned metrics.

- Finally the results are compared with the Benchmark model, and any possible scope of improvement in the model is analyzed.