

Machine Learning Engineer Nanodegree

Capstone Project

RISHAB KUMAR

June 12th, 2017

I. Definition

Project Overview

Convolutional deep learning networks (CNN) have had a dramatic impact on the image recognition space. Even simple models are able to make highly accurate predictions on datasets like the MNIST database of handwritten digits. Because of the ability of CNN to detect images with high accuracy, it has been deployed to solve many real world problems from cell phone passwords to car-plate recognition with efficiency. Using the power of CNN and its ability for image detection, I look forward to solve one of the problem that I personally face in my daily routine.

I like watching cricket matches on television. But due to scarcity of time ,sometimes I am not able to watch it live so usually have to watch the recorded match. But the recorded match have both game as well as advertisements. Advertisements not only waste time but also a lot of memory when recorded. For example, if we have a show of 30 mins , 12 mins out of them are advertisements which contributes to 40% of show time and takes 40% of memory which is useless.The saving system has to be a little intelligent for sure. So I have tried my hands on implementing one such feature for the saving system. When a computer sits free, there are a thousand processes still going on in background. I wish to add one more. I aim to build a model using computer vision and deep learning that could distinguish between a cricket game(in fact any game) and advertisements (so that computer can detect whether the video running on T.V. is game or advertisement and record only game) so that I can see the game while not being anymore frustrated.

Problem Statement

Deep learning has been used to classify pre recorded video clips and even caption each clip, with each comprising a single action or subject. In this project I aim to continuously classify video as it's captured. Continuous classification allows us to solve all sorts of interesting problems in real-time, like understanding what's in front of a car for autonomous driving applications to understanding what's streaming on a TV.

Now in this project I will be classifying what we see on our TV as either a cricket game or an advertisement using the power of Convolutional Neural Network(CNN). Using computer vision and deep learning , I propose a model to classify whether the program running on TV is cricket match or advertisement in real time.

For the proposed model, at each second image frame will be captured from TV of resolution 1960 x 1080 and then will be pre-processed first. In preprocessing ,first image will be converted to grayscale. Converting image to grayscale decreases the input data size 3 times now as RGB channel gets converted to single gray channel. This makes the training step a lot faster and computationally inexpensive (by the curse of dimensionality, $2^1=2$ for one channel data, $2^3=8$ for image with RGB channel.). Then the image frame will be resized to 128 x 76 (decreased 15 times). This furthers decrease the input data size making model fast enough to predict faster in real time. This preprocessed image will then be normalized and will be given as input to 2D-CNN for its appropriate classification whether the captured frame is cricket game or advertisement.

Metrics

Imagine a situation you spell out your name R-I-C-K but the model being less accurate mis-spells the "R" with "D" or a thumbs up sign being misinterpret as thumbs down one. No one would want to use such a model. For accepting a model, it's very necessary that the model be accurate and precise. The evaluation metrics that will be used in the project is accuracy ,precision and recall.

One can define accuracy as the number of correct predictions over number of total predictions. Since accuracy is directly proportional to the happiness index of the person using the model, it shall be a good way to measure the performance and correctness of the model.

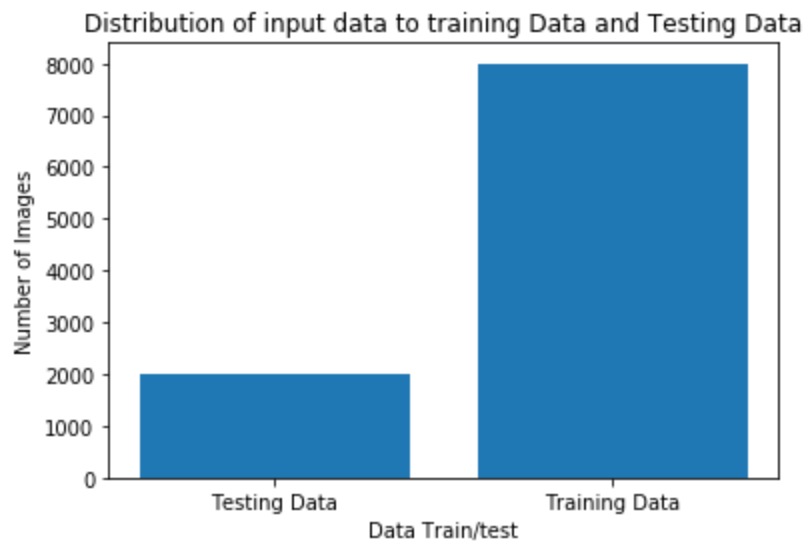
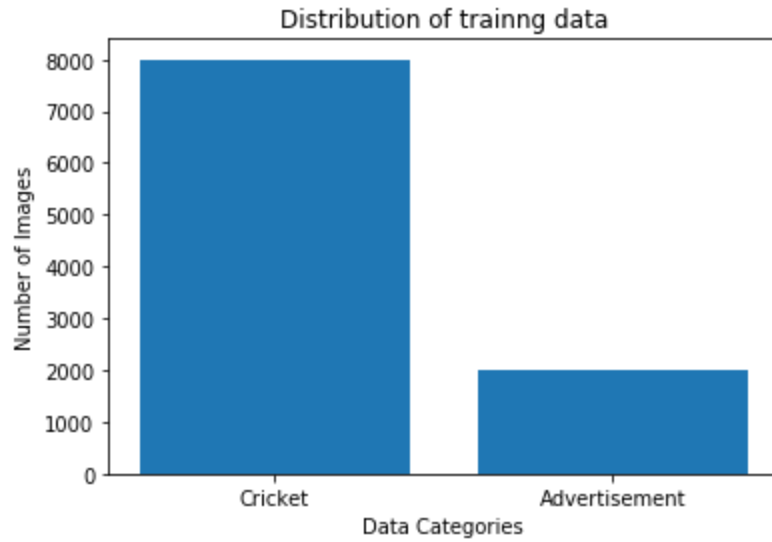
In information retrieval, precision is a measure of result relevance, while recall is a measure of how many truly relevant results are returned. A high area under the curve represents both high recall and high precision, where high precision relates to a low false positive rate, and high recall relates to a low false negative rate. High scores for both show that the classifier is returning accurate results (high precision), as well as returning a majority of all positive results (high recall).

$$\text{Precision} = \frac{tp}{tp + fp}$$
$$\text{Recall} = \frac{tp}{tp + fn}$$

II. Analysis

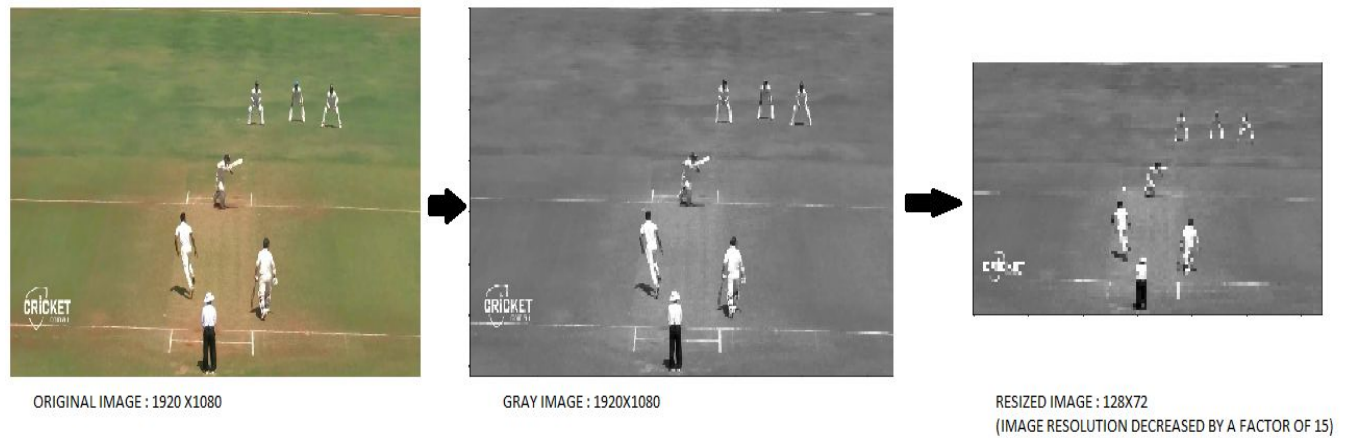
Data Exploration

Since there is no publically available dataset, I have made dataset on my own. I have taken clips from cricket match and advertisement. The cricket match is about 16 mins of length and advertisement of about 4 mins. The total length of video taken as input is about 20 mins. Each video has fps(frames per second) rate of 25. So for making data, every third frame was taken from the video making approximately 8-9 frames per second. Then first that frame is converted to grayscale and then resized to 128 x 72 pixels. Pixel values are taken and pixel array is flattened to form array of size 1 x 9216. Then these pixel values are normalized making them to lie in range of 0 to 1. This pixel array is then stored in data frame. From cricket clip 8000 frames are taken and from advertisement clip 2000 frames making total data of 10000 frames. So the size of input data is 10000 x 9216. Corresponding to that a label file is made taking label 0 for advertisement and label 1 for cricket. So this contributes to our dataset. This data set is divided into training and testing data with test size of 0.2. Total images available for training is 8000 and for testing is 2000. Proper code of making data is given in "Data Generation.ipynb".



Exploratory Visualization

Following image shows the visualization of the input data. First a frame of size 1920 x 1080 pixel is captured. Then this frame is converted to grayscale giving gray image of size 1920x1080. Then this image is resize to 128x72 pixels. This final image pixel array is taken, normalized and flattened. This array of shape 1x9216 with each element value in range of 0 to 1 is stored and given as input to 2D-CNN for its classification.



Algorithms and Techniques

In the project I have used multiple algorithms and techniques from both computer vision and machine learning. Following are the algorithms used from machine learning :

1. Decision Trees :

All decision tree algorithms have the same structure. Basically, it's a greedy divide and conquer algorithm.

1. Take the entire data set as input.
2. Search for a split that maximizes the "separation" of the classes. A split is any test that divides the data in two (e.g. if attribute 5 < 10).
3. Apply the split to the input data (the "divide" step) into two parts.
4. Re-apply steps 1 and 2 to each side of the split (the recursive "conquer" step).
5. Stop when you meet some stopping criteria.
6. (Optional) Clean up the tree in case you went too far doing splits (called "pruning")

Where the algorithms differ are:

Finding a split : Methods here vary from greedy search (e.g. C4.5) to randomly selecting attributes and split points (e.g. Random forests).

Purity measure : Measures here include: Information Gain, gain ratio, Gini coefficient, minimum description length and Chi-squared values.

Stopping criteria : Methods here vary from a minimum size, to a particular confidence in prediction, to certain purity criteria.

Pruning method: Methods here include no pruning, reduced-error pruning, and in ensemble cases such as bagging, out-of-bag error pruning.

2. Convolutional Neural Networks

There are four main steps in CNN: convolution, subsampling, activation and full connectedness.

The most popular implementation of the CNN is the LeNet, after Yann LeCun. The 4 key layers of a CNN are Convolution, Subsampling, Activation and Fully Connected.

Step 1: Convolution

The first layers that receive an input signal are called convolution filters. Convolution is a process where the network tries to label the input signal by referring to what it has learned in the past. If the input signal looks like previous cat images it has seen before, the “cat” reference signal will be mixed into, or convolved with, the input signal. The resulting output signal is then passed on to the next layer.

Convolving Wally with a circle filter. The circle filter responds strongly to the eyes.

Convolution has the nice property of being translational invariant. Intuitively, this means that each convolution filter represents a feature of interest (e.g whiskers, fur), and the CNN algorithm learns which features comprise the resulting reference (i.e. cat). The output signal strength is not dependent on where the features are located, but simply whether the features are present. Hence, a cat could be sitting in different positions, and the CNN algorithm would still be able to recognize it.

Step 2: Subsampling

Inputs from the convolution layer can be “smoothened” to reduce the sensitivity of the filters to noise and variations. This smoothing process is called subsampling, and can be achieved by taking averages or taking the maximum over a sample of the signal. Examples of subsampling methods (for image signals) include reducing the size of the image, or reducing the color contrast across red, green, blue (RGB) channels.

Sub sampling Wally by 10 times. This creates a lower resolution image.

Step 3: Activation

The activation layer controls how the signal flows from one layer to the next, emulating how neurons are fired in our brain. Output signals which are strongly associated with past references would activate more neurons, enabling signals to be propagated more efficiently for identification.

CNN is compatible with a wide variety of complex activation functions to model signal propagation, the most common function being the Rectified Linear Unit (ReLU), which is favored for its faster training speed.

Step 4: Fully Connected

The last layers in the network are fully connected, meaning that neurons of preceding layers are connected to every neuron in subsequent layers. This mimics high level reasoning where all possible pathways from the input to output are considered.

Benchmark

For the benchmark model, I would want to try out decision tree classifier. They are very good classifiers in the sense that we can reason why the particular image returned the particular output, something which isn't possible with a CNN, CNN being a black-box model. Both Decision Tree and CNN have been discussed earlier. A part of the explanation of how these two are used and which one is better of the two is given in the results section. To give an idea, because the data is about image pixel values , not so related and with a lot of features , decision tree classifier did

not work well giving accuracy around just 65% whereas CNN because of its ability to detect images gives higher accuracy around 97-98%.

III. Methodology

Data Preprocessing

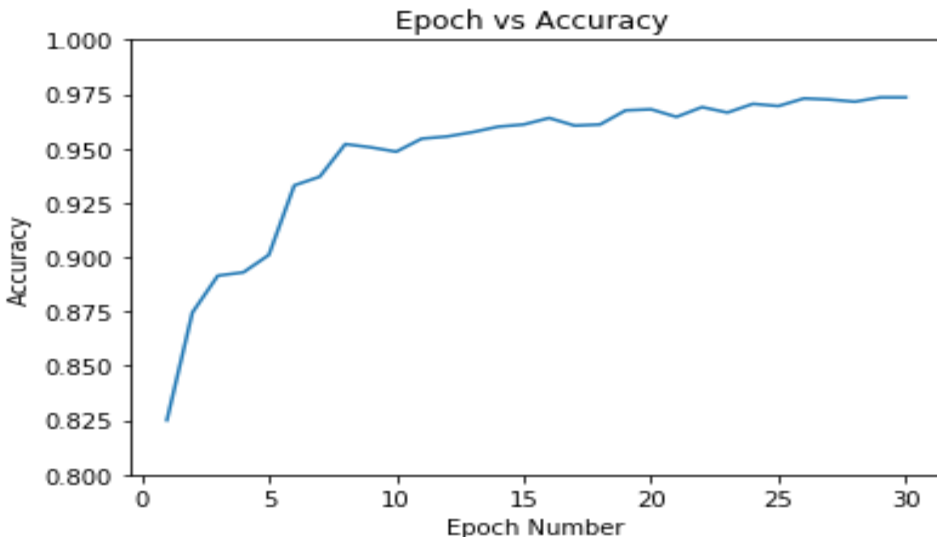
All the Data Preprocessing has already been discussed in **Data Exploration of Analysis** section above. After all data is preprocessed , then only it is fed to 2D CNN for the purpose of its classification.

Implementation

2D Convolutional Neural Network(CNN) has been used for the implementation purpose. First the traindata_gray.csv and labeldata_gray.csv files are loaded containing all the input data. Then this data is splitted into training and testing data (with random state=200) with test size=0.2 making total training about 8000 images. As this data is very huge so it is first converted into 64 batches with 125 datapoints(images) in each batch and this batch is fed into CNN.

The CNN model is constructed using Tensorflow. My CNN model consist of 5 layers in total. First there is a convolutional layer with convolution filter of size of 1x1 and RELU as its activation layer. The depth of first layer is taken as 32. So, initially our input(image data) is of shape 128x72x1 which gives output of shape 128x72x32. Then after first convolutional layer, a max pooling layer is used with filter of size 2x2 which converts input to shape of 64x36x32. After this, again a convolutional layer is used again with convolution filter of size 1x1. This time our convolutional layer has a depth of 64. This converts our input of shape 64x36x32 to 64x36x64. Again RELU function is used as our activation function for this layer. Again a max pooling layer is used with filter of size 2x2 giving us the resultant input of shape 32x18x64. Then finally a fully connected layer is used which joins our resultant input to 1024 nodes. So a full connection of 32x18x64 input data is made to 1024 nodes with again RELU as the activation function. Finally, these 1024 nodes are connected to 2 output units each for advertisement(0) and cricket(1) with softmax function used for probability distribution and prediction. At each layer dropout with probability 0.8 is used to avoid overfitting. This describes our CNN architecture.

Now , all 64 batches are given to our CNN one by one. For each batch loss is calculated with ***tf.nn.softmax_cross_entropy_with_logits*** function and backpropagation is initiated to adjust weights for minimizing loss. After all 64 batches are given as input , accuracy using testing dataset is calculated after each epoch. For testing data, batch of 100 images is made and accuracy on each batch is calculated and stored. After all 20 testing batches, all the accuracy for all 20 batches is averaged. This same procedure is repeated for 30 epochs with accuracy increasing with each epoch. Following graph shows increase in accuracy with epoch from 1 to 30.



Like this our model is trained with 8000 images and tested with 2000 images after each epoch. After training, the model so trained is saved and stored for use in future under name '**my_model**'.

The code for model training is given in file '**Model Constructed.ipynb**' file.

Refinement

Refinement is fairly important for most of the classifiers because tuning a classifier can increase the accuracy. But for this project, refinement is not required for CNN because it is already performing very well. All the refinement that was required was in data making. That refinement has already been discussed in the data making part.

IV. Results



Model Evaluation and Validation

For our problem, results given by CNN are exceptionally good. All the results are given in file '**Model Prediction.ipynb**' file. The results of CNN on our test set with 2000 images is given below :

Accuracy : 0.97350

Labels :	0 (Advertisement)	1 (cricket)
Precision	0.99130435	0.97092671
Recall	0.87244898	0.99689055

For 2000 test images confusion matrix is as follows :

Predicted 	0 (Advertisement)	1 (cricket)
Actual 		
0 (Advertisement)	344	48
1 (cricket)	5	1603

So from accuracy, precision, recall and confusion matrix we can see that our model really worked good.

On the other side, for decision tree classifier on same data set results are not so good. All its results are given in file '**Benchmark.ipynb**' file with code. The results of DecisionTreeClassifier on our test set with 2000 images is given below :

Accuracy : 0.6745

Labels :	0 (Advertisement)	1 (cricket)
Precision	0.20574163	0.79835651
Recall	0.21234568	0.79184953

From accuracy and tables above , we can clearly see that our CNN performed way better than DecisionTreeClassifier which is very well validated by our test sets tested on our both models.

Hence , CNN being the choice to solve this problem is well justified by the results. The model in its final shape with all the preprocessing seems to do a fairly good job, especially when it comes to precision of the model. And since the model is tested on real dataset and that too on so many images, the model is highly generalized. Therefore, the model is reasonable to use.

To add to the reason of expectations, the model has exceeded my expectations. For CNNs that I have personally come across, accuracy of 96% for the CIFAR dataset was my previous best. But the point here is that the data in CIFAR and this dataset differs a lot when it comes to preprocessing. I believe that the preprocessing steps are what makes the model more accurate, fast and reliable.

Justification

From the results shown above, it can be seen that in our problem CNN works way better than the classifier. Our benchmark model outperformed in front of our CNN. We are able to construct a model which can classify between cricket and advertisement efficiently and better than traditional classifiers. As image processing and image recognition was involved , so CNN was the obvious choice because it is the algorithm made specifically for image recognition and my results clearly justified my choice of choosing CNN for my problem.

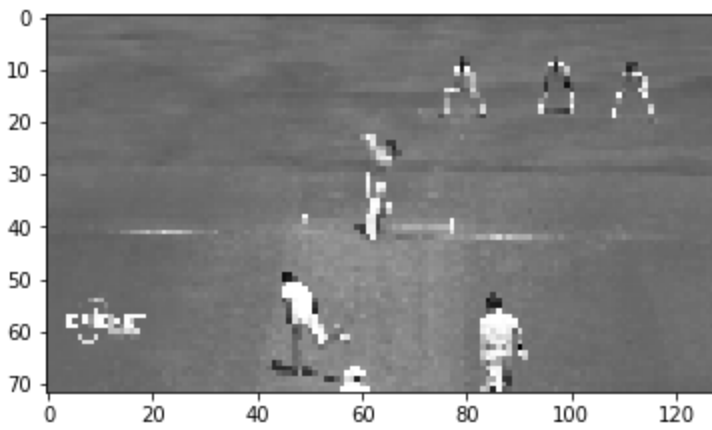
V. Conclusion

Free-Form Visualization

The Free-Form evaluation is carried in file '**Model Prediction.ipynb**' file. In this file, first the store model is restored and again accuracy is calculated which is the same as 0.97350. This confirms that model was saved correctly and was restored efficiently and is now ready to make predictions. To show image and its prediction one out of 20 test batch (batch number 4)is taken and our trained CNN is run over the batch to predict each image. The batch contains 100 images out of which 98 are predicted correctly.

All the images and predictions are given in above stated file. Some of them are :

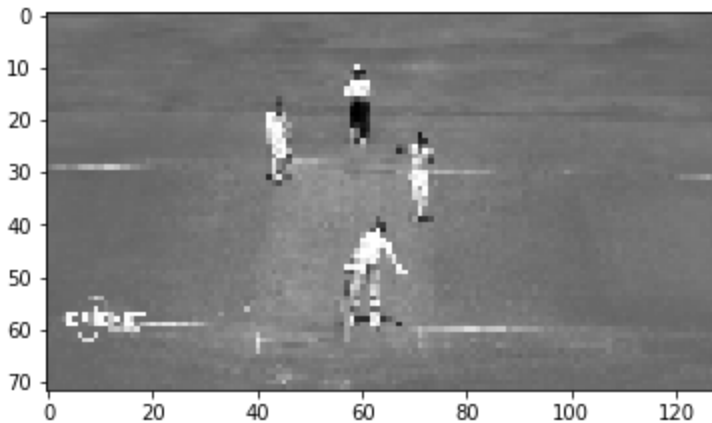
1. Image predicted is Cricket and originally image is Cricket



2. Image predicted is Advertisement and originally image is Advertisement



3. Image predicted is Cricket and originally image is Cricket



Reflection

At the start of the project, I was worried about the real time implementation of the CNN. As I expected, the training purpose took more than 6 hours on my computer whereas the decision tree classifier took just about a minute. The difference is very very large for a slight increase in accuracy. But it works fine in case of prediction as it took under 30 seconds to predict 2000 images in training data.

Initially, I thought that CNN would be the toughest part but the overall process is fairly simple. Opposite to my expectations, TensorFlow made it very very easy to use CNN. Fortunately, TensorFlow has a flexible data flow model that allows even complicated operations to be represented as sequences of simple steps. Using TensorFlow's advanced flow control operations, we could create callable Python lambdas that would be invoked by TensorFlow at runtime to perform arbitrarily complex operations. We imagine that leveraging these abilities will become more important as we move beyond simple image classifiers.

Improvement

There is always a room for improvement. For this project, I improved the model according to the limited knowledge and skills that I have. Maybe a better CNN model can be made with more convolutional layer increasing the accuracy of the model. Maybe a better activation function or different kind of pooling can be used. Overall, I am quite happy with the results and this nanodegree has given me enough knowledge to develop something that could be useful for me as well as society.

