Viterbi Internship - Final Work Report

Arka Sadhu
Supervised by: Prof. Ram Nevatia

July 7, 2017

Contents

1 Abstract

2 Introduction

The work is done as a part of the MediFor Project. The MediFor project aims at pushing the state of the art research in the field of media forensics which in broad sense deals with the tampering of the media (image, video or audio) and its detection. This work only deals with image forensics. For each manipulated image the MediFor project demands the actual image on which manipulation is done (this is called the baseline image), the kind of manipulation, and in case of splice manipulation where one image is spliced onto another image it also demands the donor image. This work focuses only on the first part, where the aim is to find the baseline image. It is assumed that the world set contains the true baseline image. All experiments are done on Nimble Dataset which is publicly available for use.

3 Implementation Details

3.1 Basic Definitions

- Probe Image : This is the manipulated image given the probe folder.
- Probe folder : Folder containing the probe images.
- Baseline Image : This the actual image corresponding to a probe image with no manipulations
- Donor : In the case where the manipulation is such that a part of image A is pasted onto image B, then image A is called the Donor Image and B is the baseline image. The resulting image would be the probe image.
- World folder : Folder contaning all the images. This includes baseline, donor as well as the probe images.
- Provenance : Provenance is simple sense means the origin, so it defines the original image of a particular probe image.
- Provenance Graph : A relational graph which depicts all the transformations a particular baseline image would've undergone to reach the probe image. It is assumed that all the intermediate images are a part of the world dataset.

3.2 Datasets Used

The datasets used for this project are Nimble Datasets

Dataset version	# Probe Images	# World Images	# Provenance Images
NC2016	1124	874	-
NC2017 Dev1 Beta4	515	1631	65
NC2017 Dev3 Beta1	2261	4098	2157
Self-Generated	1000	4098	1000

The neural nets used for evaluations are

Neural Net Used	Dataset Trained on
AlexNet	Places365
AlexNet	ImageNet

3.3 Baseline Detection

All experiments have been done on the Nimble Datasets. For the neural nets, the corresponding caffe models are used. All code is written in Python.

The baseline detection problem is essentially finding the base image of the corresponding manipulated image. There can be different types of manipulation. The Nimble 2016 dataset has the following while the Nimble 2017 dataset has the following

For the purpose of baseline detection, two models are used. First is AlexNet trained on Places365 and second is AlexNet trained on ImageNet. The reason for using AlexNet over others like VGG or ResNet is primarily that it requires low computation memory and time. The models used are pre-trained caffemodels found from respective websites. As such a comparison between the Places365 and ImageNet dataset is also shown for the purpose of baseline detection.

Simple testing

- The first step was to understand which layer of the Net should be used.The final layer which is after the Softmax layer is good when the purpose is classification. But it didn't turn out to be as good when used for the purpose of image matching.
- Image matching problem is basically trying to understand wheather or not the two images are actually the same or not. By same we mean wheather the underlying scene is the same or not. This is where the Places365 kicks in because it is scene-centric database. So in cases where a completely new object is placed on the new original image, the ImageNet gives more weight for the new object, while the Places365 doesn't change too much. This is the intuition behind using the Places365 dataset.
- The metrics used are SSD (sum of square distance), SAD (sum of absolute distance), inner product , NCC (pearson's correlation).
- Of all the metrics the pearson's correlation coefficient turned out to be the most consistent, and hence all further computations are done using this metric only. For similar images the pearson's correlation coefficient gave results in the order of 0.95-0.99 while for dissimilar images it gave in the order of 0-0.2, and this is consistent for a number of test cases.
- Now onto which layer to choose. This is dictated by emperical results on the datasets (NC2016). The output of the last layer (Softmax layer) is good for classification purposes, but not for image matching. This is especially seen in this image
- On further investigation it is found that the fc8 layer performs the best out of all, i.e. the output of the layer just before the Softmax layer. This is intuitive in the sense that, the deeper down the layers one is, the more semantic features are observed. Also after going through the Softmax layer, it weighs the scenery so that it falls into one of the categories of the dataset. But there is always the chance that a part of the image heavily influences the scene to which it should be categorised, and the Softmax layer gives it a exponential boost in some sense, which is detrimental for the purpose of image matching. From experimentation as well as by intuition it is seen that considering the output of the fc8 layer which is a 365 x 1 vector is the most suitable.
- Interestingly it is also found that for small changes (manipulations less than 10%), the AlexNet trained on ImageNet outperforms AlexNet trained on Places365, but for significant changes (more than 25%) the AlexNet trained on Places365 does significantly better.

Accelerating the process : To accelerate the testing process the following measures were explored.

- Opening multiple terminals to run the same python code. While this works well in practice, it often requires the initialization of the whole net multiple times, which in some sense is a waste of memory.
- It was found that the time consuming step is in fact the preprocessing step. This involves reading the image, resizing the image, swapping the color channels (all required by Caffe). For this reason, the multiprocessing library was used. This led to initialization of the net only once. The multiprocessing library spawns workers which handle the preprocessing

step, and adds each preprocessed image into a queue and a main worker then takes on image from the queue. This in effect reduces the test time by approximately a factor of 10.

- This can also be used for the purpose of training, but since in the project most of the work is done using pre-trained model, this functionality is not used in particular.
- Caution had to be taken to keep one worker solely for the purpose of handling the neural net, since it is not possible to spawn multiple process of the same neural net in the same python code : this results in CUDA error

Parsing Datasets :

- A time consuming problem in this project was parsing data from different datasets. Even among the nimble datasets, there were considerable differences. To overcome this, the problem was divided into two halves. The first parsing the dataset and storing all the information into a single file. For this the Python library pickle used which allows the user to dump any data in any class easily. The second part being directly using this pickle file. So essentially parsing has to be done only once and stored into a .pkl file and then it can be directly used.

Mogrifying jpg to png :

- There are a number of troublesome images in the Nimble Dataset especially in the 2017 datasets where the image is marked .jpg file but is actually .png file. This leads to some error in reading the file. These troublesome files are identified by their mime types and converted to .png files.
- For some of the images an extra step is required : to mogrify (command provided by ImageMagick). If this is not done, libpng throws an error for not being able to file.

Keeping the approach modular :

- This is done by saving the corresponding files at each step. For example the all the feature vectors of the corresponding files are saved at once and storing it into a file, rather than recomputing it again and again.
- As a first step, we directly try to select the top-k matches using both the nets.

Slicing the images to get better matches :

- It is noted that in many cases the manipulation is restricted to a part of the image. Therefore to improve the performance, we slice the image into two halves or four sections. Then we consider the maximum of the two (or four) matches, and the strongest one gives us the top-1 match.
- This method is bad in the sense that it simply doubles or quadruples the computation time for feature vectors.
- Another problem is that the manipulations might be in the center, and as a result of slicing, the manipulation as such may be lost, and lead to some random matching.
- The more important problem is that, it is not robust against rotation, and also it is less robust against large scale variation.

Things I tried on the Datasets

- First tried simple matching. This involved extracting the feature vectors from the image and then doing a brute force comparison with the other images in the world dataset. Then try to note if the ground truth image is in the top-k matches. Here k is chosen to be 1, 5, 10.
- K-means clustering : Direct clustering algorithm was applied with k chosen to be the same as the number of probe images. This gave quite decent results.
- Misleading results : The organization of the dataset is somewhat poor. Each probe image has multiple images in the world set which are closely related to it. And there are cases where two images with different ids are extremely similar. This results in a high correlation, even though it is not the ground truth.

Graph Structure

- To minimize the number of times the correlation had to be computed a graph structure was used. The library networkx is relied on for the data structure Graph.
- The nodes of the graph are all the world images. Each edge of the graph has a weight equal to the correlation between the two nodes.

4 Results

5 Discussion