Machine Learning II_DATS_6203_11
Dr. Amir Jafari

ML2 - Final Project

Individual Report

Photoshop Detector

Chirag Jhamb

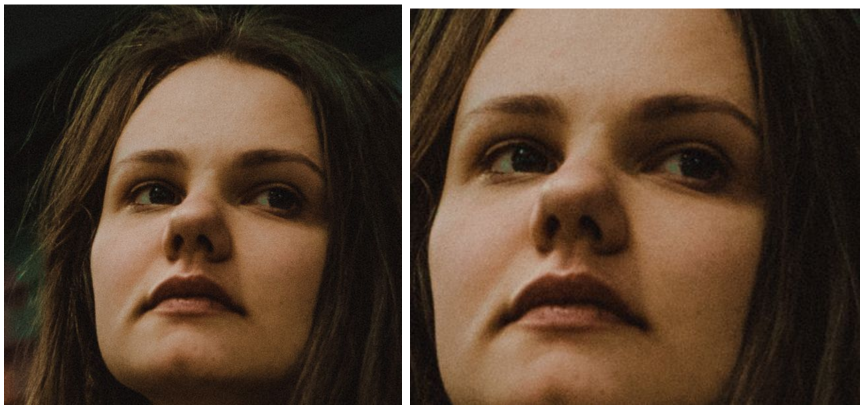
Abstract

The aim of this project is to use Neural Networks to determine whether a photo has been altered by photoshop. During the course of this project, we have attempted to explore several different types of neural networks (like Dilated Residual Networks, ResNets, vgg’s) and frameworks (pytorch, keras, tensorflow). In addition we also experimented with fastai library is order to speed up the training. The main focus throughout has been to achieve better accuracy in identifying photoshopped images. We attempted to isolate the faces in the images in order to reduce the noise in the background. This research is part of a broader effort to improve digital image forensics; a field dedicated to detecting the authenticity of images.

My Work

Dataset Preprocessing

Cropping images: Getting the faces and cropping was done using MTCNN, a face detector package based on the paper facenet. It gives a box of x, y, width and height values, which we can use to crop the face and ignore the rest of the image. In a lot of cases the chin area of the face was cropped out as well, so we did +15 to the height to adjust for that.



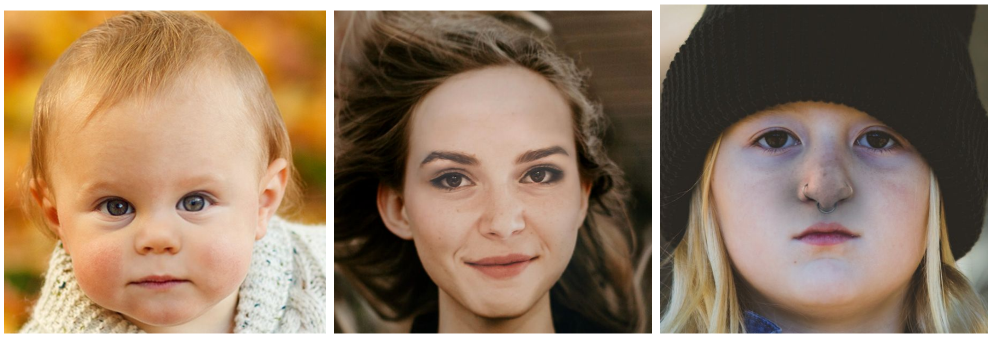
Original Image vs Cropped Image

MTCNN:

Based on facenet, it was trained, using triplets of roughly aligned matching / non-matching face patches generated using a novel online triplet mining method. The benefit was much greater representational efficiency.

Subsetting dataset:

For fake, we had easy, medium and hard images based on the level of photoshop editing performed. In order to avoid getting just one category in test and getting a very good or bad accuracy, we took 20% of each difficulty level of fake images and 20% of real images for the test set.



Hard *Medium* *Easy*

Photoshop levels

fastai

Fast.ai is synonymous to transfer learning and achieving great results in a short amount of time. The fastai library simplifies training fast and accurate neural nets using modern best practices. It's based on research in to deep learning best practices undertaken at fast.ai, including "out of the box" support for vision, text, tabular, and collab (collaborative filtering) models.

I used fast.ai for data augmentation as well as transfer learning.

Data Augmentation:

Before adding the cropping of images in the data preprocessing as mentioned above, we tried our hands on data augmentation package. We tried the following techniques and got the respective changes in the accuracy while using resnet50 (original accuracy was 63%):

- Cutout, decreased accuracy by 4%
- Contrast, increased accuracy by 2%
- Brightness, increased accuracy by 2%
- Dihedral, increased accuracy by 1%
- Jitter, decreased accuracy by 4%
- Perspective_warp, decreased accuracy by 6%
- Rgb_randomize, decreased accuracy by 4%

Transfer Learning

Below is a table of results that were obtained after I tried various transfer learning methods using fastai library:

Network	**Epochs**	**Batch Size**	**Accuracy**
FastAI-ResNet152	n/a	n/a	Out of Memory
FastAI-resnet50	14	64	63%
FastAI-VGG16	14	64	64%
FastAI-Densenet101	10	64	61%

Summary

I learned a lot about the different approaches being used to for detections using CNN. For example, MTCNN is using deep metric where embedding where semantically similar images are embedded to nearby locations, and semantically dissimilar images are embedded to distant locations.
Fastai data augmentation is much more easier to use than cv2 when we need quickly to see which approach works better. From now on, I would prefer testing my model on fastai and then after I know which model works best, I would use pytorch to customise it and ensemble with other well performing models.

Percentage of code

Total lines of code = 170
Lines copied = 100
Lines modified = 53
Percentage = 100*(170-53)/(170+100) = 43.33%

References

1. https://arxiv.org/pdf/1705.09914.pdf
2. https://peterwang512.github.io/FALdetector/
3. https://www.kaggle.com/ciplab/real-and-fake-face-detection
4. https://arxiv.org/pdf/1512.03385.pdf
5. https://en.wikipedia.org/wiki/Residual_neural_network
6. https://docs.fast.ai/
7. https://arxiv.org/abs/1801.04381
8. https://ai.googleblog.com/2018/04/mobilenetv2-next-generation-of-on.html
9. https://neurohive.io/en/popular-networks/vgg16/
10. https://towardsdatascience.com/review-densenet-image-classification-b6631a8ef803
11. https://github.com/ipazc/mtcnn
12. http://arxiv.org/abs/1503.03832