CS230

Photoshopped Image Detection with Deep Neural Networks

Thapelo Sebolai
Department of Computer Science
Stanford University
tsebolai@stanford.edu

Ozioma-Jesus Anyanwu
Department of Computer Science
Stanford University
ozijesus@stanford.edu

Abstract

As image manipulation techniques become more sophisticated, there exists a serious threat that such technology can be used for nefarious purposes. Therefore, there is an increasing need to detect such manipulations effectively. We present four different models that attempt to classify a given image as either photoshopped or pure. Our baseline model is a 4-layer Fully Connected Network. We also experimented with a 3-layer CNN, a ResNet, and an InceptionV3 Model. The latter gave us the best results with an accuracy of 68%.

1 Introduction

Today, anybody can interact and edit images and videos for reasons ranging from artistic expression to financial gain. However this increase in the sophistication of image manipulation technology has been exploited for malicious purposes. These range from creating a manipulated body type that is impossible to achieve for Instagram likes to creating realistic forgeries for identity theft. While these manipulations can be very difficult for the human eye to detect, they can accurately be detected by deep neural networks.

Although the task of image forgery detection is not a novel problem, we believe that there is not sufficient literature and publicly-available implementations that tackle the issues of image manipulations, particularly through the use of Photoshop. Consequently, we propose a novel approach of using four methods to detect image manipulation. Namely, we used a 4-layer Fully Connected Network as a baseline, followed by a 3-layer CNN, both trained end-to-end. We followed this up with a ResNeT18. And finally, we used an InceptionV3 network pre-trained on ImageNet. The intuition is that the pre-trained network will be able to extract low-level image features that could be passed through to a fine-tuned FC layer (3). For each of these networks, the input will be an image resized to fit the model's specifications. The output will be a 1 if the model predicts that the image is photoshopped, and a 0 otherwise.

2 Related work

Image Forensics has been tackled on numerous levels. Here are some approaches to the problem that don't involve NNs:

- Error Level Analysis method finds the difference in compression between Photoshopped regions and original regions through different JPEG compression qualities(11)
- A color filter array-based method that uses the camera filter array patterns and then produces the tampering probability for each pixel using its neighbours(13).

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

There is an even greater number of approaches that involve NNs, here are a few:

- Chen et al. used a low pass filter layer before a CNN that produced features that the CNN could use to detect median filtering tampering, particularly of the copy and paste variety(10).
- Bayar et al. who instead force the first layer of their CNN to suppress the image's content feature and instead learn the relationships between a pixel and its local neighbors(2).
- Zhang et al. utilized a Stacked Autoencoder to learn contextual information for each individual image patch and integrate that information to perform detection(18)
- Last year, a team at Adobe proposed a two-stream Faster R-CNN network and trained it to detect the tampered regions in the altered image(19). The team utilized a synthetic dataset constructed based on COCO to pre-train their model. This method achieved high accuracy of around 90% for each of the three common image manipulation techniques. Their approach was novel, however, we question whether they would achieve a similar accuracy with a collection of manually created alterations, like our dataset of Photoshopped images.

3 Dataset and Feautures

The PS-Battles dataset, (by Hellet et. al (6)) consists of 102,028 RGB images, grouped into 11,142 subsets. Each subset consists of 1 original image and then a variety of manipulated derivatives by a mix of amateur and professional digital artists.

Figure 1: Photoshopped to Original Pair

3.1 Data Processing

We had to take several preprocessing steps before we could train our models. First, the dataset contained several corrupted files which we deleted with Python Image Library. We also aimed to download only one photoshopped image per original image to create equal class distribution. We were left with 20,759 images, and therefore used a 70-15-15 split to get 14,533 images for training, 3,113 images for validation, and 3,113 images for testing. We then had to resize each image to fit into the (299x299x3) form required to be processed by models. As for data augmentation, we center-cropped each image and randomly performed a horizontal flips. For the test data, we only cropped the center of the image to fit the required size.

4 Methods

4.1 Problem Formulation

Our goal is to minimize the difference between our model's output probability that the input image has been photoshopped, and the ground truth. We therefore use cross-enntropy as our loss function.

4.2 Architecture

Following the work of Zhang et al. we hoped to achieve a classification accuracy in the range of 70%. Since our approach does not utilize any external metadata, we lowered our efficacy bound for this

reason as well as the complex content features of our images. Additionally, this benchmark is above the 50% baseline that could be achieved by random guessing.

We performed classification using four models:

Model 1 - Baseline: 4-layer Fully Connected Network

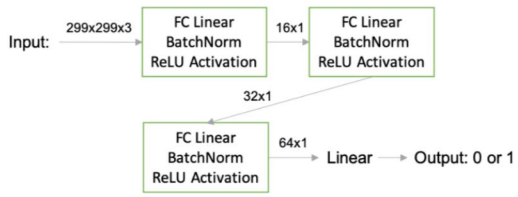

Figure 2: 4-layer Fully Connected Network

We utilize a fully-connected network to determine how well a simple model can learn the complex features of our images. The intuition behind this was to create a measurement of the feature complexity of our data. If a simple fully-connected model could perform better than a random guesser , then a more nuanced model could perform even better and capture a greater amount of the complexity. Each FC block had Batchnorm applied to it, followed by a ReLU activation function which would ideally increase training stability and result in better convergence.

Model 2: 3-layer Convolutional Neural Network

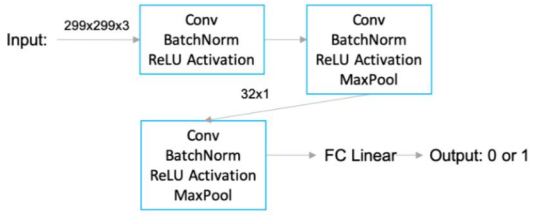

Figure 3: 3-Layer Convolutional Neural Network

We then constructed a basic CNN consisting of 3-layers. Each Conv layer consisted of a Convolution, followed by BatchNorm and a ReLU activation. On the last two we added a MaxPool layer, which allowed us to down-sample. This is important because it allows us to capture only the most important features. In this case, instead of simply capturing content, we want to highlight differences like extreme deviations amongst neighboring pixels.

Model 3: ResNet-18

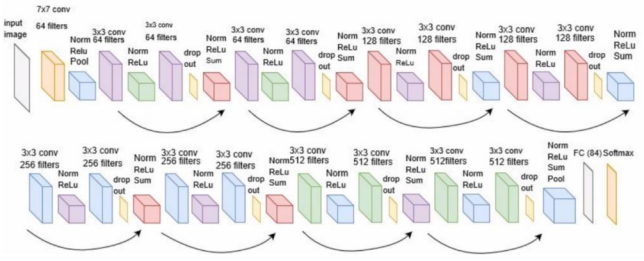

Figure 4: ResNet-18 Neural Network Schema(16)

ResNets are powerful because they allow us to train much deeper networks while still attaining a lower cost. This is because the skip connections within the network allow earlier layers to transfer

their information over to later ones, making it easy for a deeper network to perform no worse than a shallower one. We trained the 18-layer ResNet in two different ways:

- **End-to-End Implementation**: We trained all the weights from scratch using our PS-Battle dataset. One thing to note is that ResNet-18 uses a lot of "same" convolutions to keep the dimensions of the layers the same to allow for the skip connections.
- **Fine-Tuned Implementation**: We froze all layers in the network except for the last one. We retrained the weights of the last layer and changed the final activation to a sigmoid. The network was pre-trained on ImageNet

Model 4: Inception V3 Neural Network

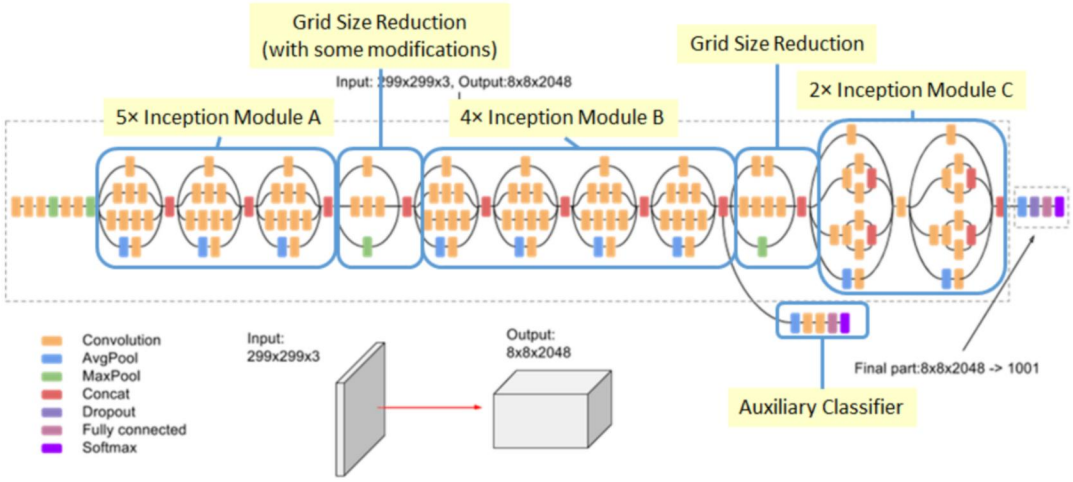

Figure 5: Original InceptionV3 Neural Network Schema(17)

Our final classification model is a fine-tuned InceptionV3 network trained on ImageNet (3). To use the InceptionV3 as a feature extractor, we froze all the lower pre-trained convolutional layers, replacing the last fully-connected layer with randomly initialized weights, and trained the parameters connected to the final activation layer using our dataset. Our intuition is that like the RGB and Noise stream networks utilised as feature-creation models in (19), the pre-trained portion would be able to detect hidden features of each image.
The key ideas of the InceptionV3 are the following (17):

- **Factorized Convolutions**: Instead of using larger 5x5 or 7x7 convolutions, you can instead stack smaller kernels and thus reduce the number of parameters needed for learning. InceptionV3 achieved this by stacking Inception modules of different structures with dimensionality reductions between each. Reducing number of parameters means we can create a deeper network but still reduce overfitting.
- **Auxiliary Classifiers**: According to (17), these Auxiliary Classifiers act as regularizers by inserting additional gradient. This acts to prevent overfitting and thus allow the model to generalize better.

The remaining implementation details of the InceptionV3 implementation are explored in (17)

5 Experiments/Results/Discussion

In order to determine the optimal hyperparameters for each of our classifier models, we trained each of our models on 412 images for 10 epochs, using 88 images for validation and another set of 88 images for testing. We found that an Adam Optimizer, with an exponential learning rate scheduling system worked best for our model. Here are the hyperparameters that worked well for us:

Model	Learning Rate	Step Size	Gamma	Batch Size
FC NN	0.005	2	0.1	64
CNN	0.01	2	0.05	64
InceptionV3	0.005	2	0.05	64
ResNet End-to-End	0.005	2	0.05	64
ResNet Last-layer Trained	0.005	2	0.05	64

Using these parameters, we then proceeded to train each of the models using the following 70/15/15 split over 30 epochs:
Training Set: 14533
Validation Set: 3113
Test Set: 3113

Figure 1: Validation Accuracy vs. Number of Training Epochs on Full Dataset

The test results were as follows:

Model	Accuracy
FC NN	0.4916
CNN	0.5555
ResNet End-to-End	0.5859
ResNet Last-Layer Trained	0.5951
InceptionV3	0.6848

The results showcase that the InceptionV3 network performed best. Unfortunately, it didn't reach the benchmark of 0.7 accuracy. A possible reason for this is that freezing all the convolutional layers might not have worked in our benefit. However, we discovered that, when trained end-to-end, our ResNet models tended to overfit. This explains why we we didn't see a large increase in accuracy from our CNN to the ResNet models.

6 Conclusion/Future Work

We were able to create an algorithm that managed to classify Photoshopped and un-Photoshopped images correctly 68% of the time. This fell short of our original benchmark of 70% accuracy indicating that there is significant room for improvement. We believe that using a pre-trained network in the form of InceptionV3 trained on ImageNet was good intuition, since it outperformed the other classification models. One key aspect that we failed to work on was actively combating over-fitting and under-fitting using methods beyond solutions like dropout. We believe that the largest reason why the InceptionV3 worked the best was due to it being less prone to over-fitting to the data. The Auxiliary classifiers inject additional gradient, allowing for more regularization, than the other networks.

We would work on devising new ways to extract features from each image, ones that captured the most salient aspects would be a huge step. This could follow a similar paradigm to previous work we researched: Interpolating two streams, one that extracts low-level features like noise and another that created high level features like RGB values, and passing them through as a feature would have enabled our classifiers to function better.

7 Contributions

Thapelo Sebolai

- Built/ Integrated the remainder of the existing codebase - data pre-processing, other classifiers e.g ResNet, InceptionV3
- Built testing and hyperparam tuning scripts.

Ozioma-Jesus Anyanwu

- Wrote image-processing script to resize images and delete corrupted photos.
- Built baseline fully connected network.

References

[1] Barrat, Shane Pytorch Implementation of Inception Score https://github.com/sbarratt/inception-score-pytorch

[2] B. Bayar and M. C. Stamm A deep learning approach to universal image manipulation detection using a new convolutional layer. IH&MMSec, 2016.

[3] Chilumkurthy, Sasank. Transfer Learning Tutorial https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html

[4] CS230 Code Template https://github.com/cs230-stanford/cs230-code-examples/

[5] CS231N http://cs231n.github.io/

[6] Heller, Silvan, Luca Rossetto, and Heiko Schuldt. "The PS-Battles Dataset-an Image Collection for Image Manipulation Detection." arXiv preprint arXiv:1804.04866 (2018).

[7] Huang, Teeyo Pytorch Implementation of Pix2Pix https://github.com/TeeyoHuang/pix2pix-pytorch

[8] Inkawhich, Nathan. DCGAN Tutorial https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html

[9] Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[10] J. Chen, X. Kang, Y. Liu, and Z. J. Wang. Median filtering forensics based on convolutional neural networks. Signal Processing Letters, 2015.

[11] Krawetz, Neal, and Hacker Factor Solutions. "A Picture's Worth..." Hacker Factor Solutions 6 (2007).

[12] Nist nimble 2016 datasets. https://www.nist.gov/itl/iad/mig/nimble-challenge-2017-evaluation/.

[13] P. Ferrara, T. Bianchi, A. De Rosa, and A. Piva. Image forgery localization via fine-grained analysis of cfa artifacts. TIFS, 2012.

[14] Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015).

[15] Salimans, Tim, et al. "Improved techniques for training gans." Advances in neural information processing systems. 2016.

[16] Muhammad, Abdul Hassan, et al. "Isolated Bangla Handwritten Character Recognition with Convolutional neural network" 2017

[17] Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[18] Y. Zhang, J. Goh, L. L. Win, and V. L. Thing. Image region forgery detection: A deep learning approach.

[19] Zhou, Peng, et al. Two-stream neural networks for tampered face detection, https://arxiv.org/abs/1803.11276

[20] Zhou, Peng, et al. "Learning rich features for image manipulation detection." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.