CS230

Doctored or Not: Detecting Photoshopped Images

Kavin Anand*
Department of Computer Science
Stanford University
akavin@stanford.edu

Michael Cao
Department of Computer Science
Stanford University
caom@stanford.edu

Abstract

Project Category: Computer Vision
The rise of photoshopped images on the web and social media sites are a serious threat. Not only does it contribute to the rapidly growing epidemic of mental health cases amongst the younger generations, it is also a direct attack against one's integrity and opens the door for myriads of political scandals. In the age of fake news and mistrust of media, manipulated images could shatter the public trust of politicians and the world. Hence, in this project we aim to address this problem and design an algorithm that will be able to detect if images have been manipulated, classifying the input features as either true images or photoshopped copies. Settling on a ResNet modified to include the Adam Optimizer, we trained the model on over 100,000 images from a Flickr face dataset, using a Photoshop script to generate the negative class. The model's best performance was 94.9% accuracy on the development set and 94.3% on the test set.

1 Introduction

Photoshopped images are an increasingly prevalent problem. As the technology to fabricate faces and true images improves, our capabilities of detecting them fails to keep pace. This is especially common in social media sites, where models, influencers and other high-profile users consistently enhance their appearance to their millions of fans, setting false expectations of the beauty standard. Substantial research has focused on this subject, and top opinions arrive at a consensus that this constant barrage of fake standards adversely affect the mental health of the primarily adolescent audience that look up to these figures. In fact, it has been shown to correlate with the high rates of depression and anxiety linked with Generation Z. Beyond this, if falsified images continue to spread, it can destroy public trust in politicians and world leaders as their messages can easily be forged. Especially with our already fragile political climate and "fake news," it is imperative to be able to identify these images and accurately classify them as fake. Our project aims to fill this knowledge gap, given an input image our deep learning neural network should correctly classify that as either a truth or falsified image.

2 Related Works

Existing methods to detecting photoshopped images generally fall into two categories: algorithmic approaches and deep learning approaches. On the algorithmic side, there are a variety of image analysis algorithms that have been applied to our problem, such as Error Level Analysis (ELA)

*Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies.

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

[1], Principal Component Analysis (PCA) [2], wavelet transformation, and light direction detection. These methods have proven to be rather effective, as they beat out human performance, however they are far from perfect. In recent years, researchers have been interested in improving the accuracy of photoshop detection, applying deep learning techniques to the problem.

Most notably, researchers from UC Berkeley and Adobe, the creator of Photoshop, developed a CNN-based model that detects images that were altered using Photoshop's "Face-aware Liquify" tool, which is extremely difficult to spot with the human eye [3]. In specific, their approach is a Dilated Residual Network variant (DRN-C-26). After further research, we have decided not to pursue their method and rather strike a different approach. A key intuition is that the doctored regions are exceedingly small, with the target feature zone being a focused subsection of the original image. We first seek to understand dilated convolution to see why this model isn't optimal for those features. Compared to the normal convolution, the dilated version introduces a hyperparameter called the dilation rate which injects spaces into the kernel to increase the receptive field of each filter [4]. This is counter-intuitive to what our model's goal of identifying focused, regionalized features so we opted to forego this approach.

3 Dataset and Features

Our current dataset is a high quality collection of faces that were crawled from Flickr, known as the Flickr-Faces-HQ-Dataset (FFHQ). These images were all automatically aligned and cropped to 512x512 resolution with dlib and maintain a considerable amount of diversity with various ages, ethnicities and backgrounds represented. The data has also undergone extensive pre-processing with Amazon Mechanical Turk to remove any statues or paintings that could confound our neural network. The FFHQ set is available on Kaggle at this link: https://www.kaggle.com/arnaud58/flickrfaceshq-dataset-ffhq. The dataset currently contains 52k real/fake image pairs, resulting in 104k images total, which is an ample upgrade from the Real/Fake face detection set we were initially using that had less than 2000 images. This old dataset, which we also discovered from Kaggle, contained expert-generated photoshopped images with varying levels of detection difficulty, which although useful and may help train a more robust model, was limiting in the volume of trainable images. As such, we were running into overfitting problems as there was just too few training examples to build a robust network on. When we moved on from this dataset, we had to find a way to recreate the fake set that had been provided to us before, or else we'd have no negative class for our model.

To solve this problem, we reached out to the authors of the Adobe paper [3] and obtained links to the script they used to automate warps for an image (https://github.com/PeterWang512/FALdetector/issues/3). This script utilizes the Face-Aware Liquify tool to select random features to warp, such as the right eyelid or left cheekbone, which ensures development of a robust train set. We utilized this tool to effectively double our FFHQ dataset by generating a photoshopped copy for each truth image present. A sample from the expanded dataset is provided in Figure 1.

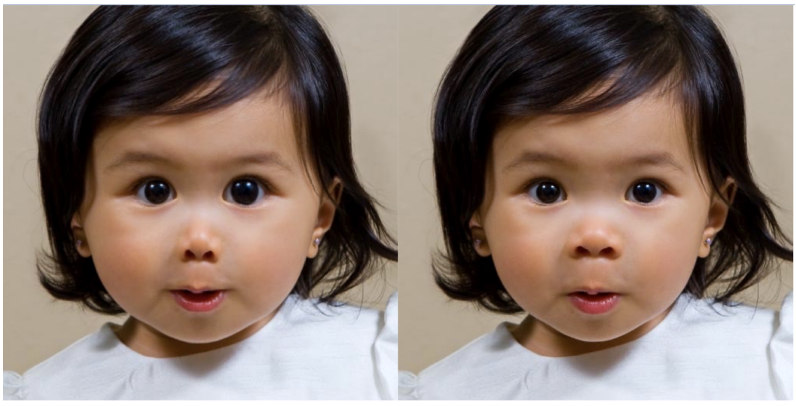


Figure 1: Scripted image (left) in comparison to the truth image (right) from the FFHQ dataset

4 Methods

We opted for a supervised learning approach where the final dataset contained both truth and false images in equal proportion, with each image labeled. We did extensive preprocessing by normalizing the images to the same size, as the photoshop script runs a compression on the input image we had to compress the FFHQ dataset, and ensuring both classes contained all relevant information fields. We then followed Prof Ng's advice and split the dataset 90/5/5 to put as many examples as possible into our training set, as we felt this would give us the maximal training power and there were enough images in both the dev and test set to validate our results.

We arrive at a deep Residual Neural Network as the optimal architecture for our problem. ResNet's introduction of an "identity shortcut connection," to skip multiple layers when building a deep model, is critical in dealing with the vanishing gradient issue that plagues other architectures of similar depth. Especially as our model needs to learn exceedingly fine features, it's almost a necessity for it to be deep. We tested two variants of residual networks, ResNet-18 and ResNet-50, that are 18 and 50 layers deep, respectively, and compared results. To further boost performance we used pretrained weights from the ImageNet database, which contains over one million images.

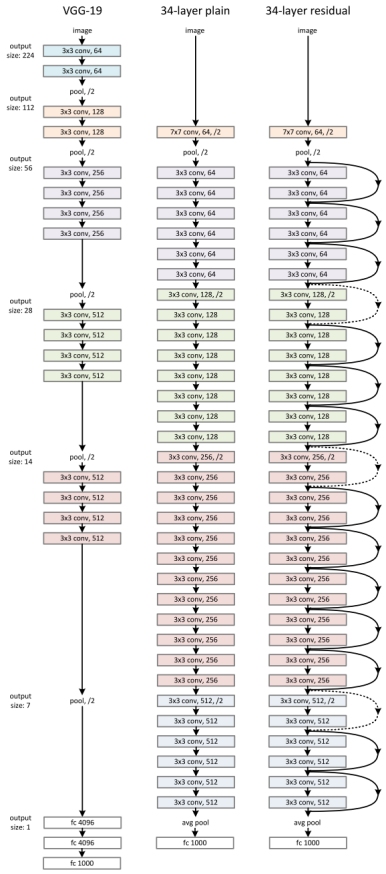


Figure 2: The residual network (right) escapes the vanishing gradient problem persistent in the traditional (middle) network, while still being deep enough to compute meaningful results [5].

5 Results and Discussion

As our baseline, we have trained a ResNet-18 model from PyTorch with all default settings. In literature, ResNet has been shown to perform far better than a typical CNN so we felt this will be the most up-to-date baseline for our task.

After training our baseline with the Real/Fake face dataset, we were able to achieve a dev loss of a little over 60 percent. For our baseline, we used SGD with momentum of 0.9 for our optimizer, a learning rate of 0.001, and a batch size of 16. These were the parameters recommended by PyTorch, but we believe that we can improve performance and convergence speed by using Adam as our optimizer.

We found that our training loss decreased the entire time while our dev loss plateaued, signalling possible overfitting and a high variance issue. We believe that this is due to the fact that the Kaggle dataset is rather small, so we considered finding a larger dataset and/or doing some scripting to generate our own dataset. Another tool we have to tackle the high variance issue is regularization and introduction of batch normalization.

After discovering what needed to be improved, we created an updated baseline with the same model architecture and hyperparameters but with our new dataset, the FFHQ. We initially trained a model with 44k images, which is almost half of our full dataset, and we found that performance increased significantly, as both dev and test accuracy jumped to 88.6 percent and 86.7 percent, respectively. With this success, we trained a model on our full dataset of 104k images and obtained a best dev loss of 90.2 percent and a best test accuracy of 88.9 percent.

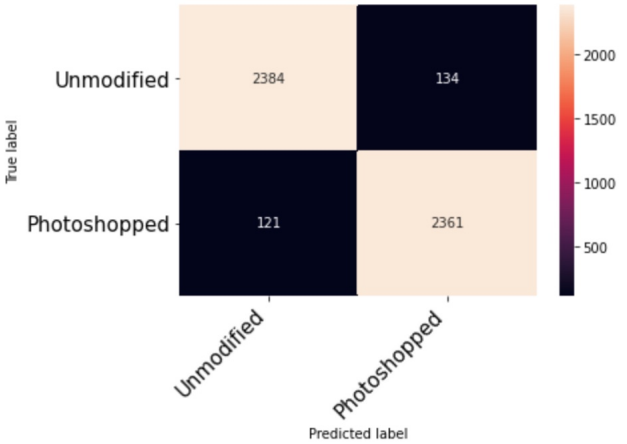
We noticed that the model converged very quickly and started to overfit after around 4-5 epochs, so to try to increase performance, we switched out the SGD optimizer with an Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ and lowered the learning rate to 0.0001. Changing the optimizer increased performance significantly, and our updated model was able to achieve 94.2 percent dev accuracy and 93.2 percent test accuracy. Test loss was also limited to 0.2939.

For our final model, we trained with the same hyperparameters on a deeper Resnet-50 model, and we saw even further improvement, especially with the test accuracy; we were able to achieve 94.9 percent dev accuracy and 94.3 percent test accuracy.

Table 1: Model Results

Model Name	# of Images	Dev Accuracy	Test Accuracy
ResNet18 + SGD	2000	62.4	61.6
ResNet18 + SGD	44000	88.6	86.7
ResNet18 + SGD	104000	90.4	88.9
ResNet18 + Adam	104000	94.2	93.2
ResNet50 + Adam	104000	**94.9**	**94.3**

Figure 3: Final Model Confusion Matrix - Dev Set



As our model achieved 94.9 percent dev accuracy and 94.3 test accuracy, we are very pleased with our results and accuracy on our generated dataset. We performed an error analysis and generated a confusion matrix for our model, and we found that our model tends to classify unmodified photos as

photoshopped slightly more than the other way around. This was actually quite surprising, as we hypothesized that our model would struggle to classify photos that were photoshopped very subtlely.

After training our final model, we wanted to test the whether our model would generalize to other types of doctoring methods. We ran our model on our original Kaggle dataset which contained patching and warping, and we achieved an accuracy of 63 percent. Although this is far from perfect, it is still noticeably better than random guessing. Next, we ran our model on scripted images with the random parameters altered to less aggressively warp the images, and our final model achieved 84 percent accuracy. This decrease in performance is likely due to the low resolution of our model, so subtler changes would be hard to pick out. Nonetheless, our model is still quite robust and achieves good performance in detecting face-liquified images. We believe that either performing data augmentation and/or including images that include other types of warps can help our model generalize better, which is crucial to being able to deploy our model in the real world.

6 Conclusion and Future Works

The final model was trained for 15 epochs, on 104,000 images with a 50/50 split between the binary classes and resulted in a 94.2% accuracy on the dev set and a 93.2% accuracy on our test set. The model is somewhat able to generalize to other warps, and it still performs well on more subtly warped images. The final algorithm that worked best, the modified ResNet with the Adam optimizer, fit our initial hypothesis as being the best architecture for our problem.

There are many avenues to take this work in the future. For one, if we had more team members something we'd considered is experiment breaking up our input images into small chunks. This is to better capture the lower level details of each image where we believe many of the features to reside. Of course, a difficulty is that with our supervised learning approach, it's hard to obtain labels for such a huge dataset especially because we cannot automatically allocate all the smaller slices to have the same class as the original. In the common case of where the photoshopped features are isolated to a small region, only that slice should have the negative class label and the others should be fixed to positive which is a difficult and arduous data preparation task. Another approach we could consider, given more manpower and time, is changing our architecture and adapting it to the novel ResNeXt variant. Without going too deep into the implementation, the ResNeXt varies from the traditional implementation by splitting into multiple paths in the "identity mapping" step introduced by the base model and depth-concatenating the different paths in the merge step [6]. Thus a new term, cardinality, was introduced that helps us control the model's capacity. This is an approach worth considering because preliminary research with ResNeXt shows that accuracy is gained more easily with this cardinality approach and the model generalizes better new datasets, something that our final implementation could benefit from [6].

7 Contributions

Both of us worked on idea formulation: combing through resources available on the 230 website, identifying problems that are prevalent today, especially so for our generation, and selecting one which a deep learning approach could make headway on solving. We both did extensive research to discover the optimal dataset, sifting through Kaggle, towardsdatascience and other websites before settling on FHHQ. Kavin helped with the script and sections 1,3,4 and 6 on the report. Michael contributed much to the code base and scripting algorithm on PhotoShop to generate a larger dataset. Michael also ran the baseline, final result and multiple intermediary training of the model and focused on section 2, 5, and 6 on the report.

References

[1] N. B. A. Warif, M. Y. I. Idris, A. W. A. Wahab, and R. Salleh. An evaluation of error level analysis in image forensics. In *2015 5th IEEE International Conference on System Engineering and Technology (ICSET)*, pages 23–28, 2015.

[2] Ian T. Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

[3] Sheng-Yu Wang, Oliver Wang, Andrew Owens, Richard Zhang, and Alexei A. Efros. Detecting photoshopped faces by scripting photoshop, 2019.

[4] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions, 2016.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[6] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *CoRR*, abs/1611.05431, 2016.