CS230

Deep Fake Detector: Using ML techniques to Distinguish Real Images from Fake

Jervis Muindi
Department of Computer Science
Stanford University
jmuindi@stanford.edu

Raj Prateek Kosaraju
Department of MS&E
Stanford University
rprateek@stanford.edu

Yash Lundia
Department of MS&E
Stanford University
ylundia@stanford.edu

Abstract

Fake and manipulated images have become extremely prevalent in recent years and have been credited with deceiving even the most informed. Spreading news and images has becoming so easy in the present times that a fake/manipulated image can be spread out to millions in the blink of an eye. This makes it incredibly easy for manipulated images to spread fake news and misinformation. This inspired us to develop a fake image detector using deep neural network techniques. We trained models using numerous CNN architectures and chose the most promising architecture. Using hyperparameter tuning, we arrived at a ResNet model that achieved 94% accuracy, 92% precision, and 93% recall when detecting fake images.

1 Introduction

With the emergence of social media, people around the globe are connected more than ever. They can share photos, videos and messages to anyone in the world in a matter of seconds. Given all the positive aspects of social media, unfortunately social media today, is also used to spread false news through fake or manipulated images. In current times, there has been a great increase in the proliferation of fake images on social media as well as other mediums. Most of these fake images are indistinguishable from real content to the unsuspicious reader. Apart from misinformation, manipulated images also aid document fraud. The spread of manipulated images is therefore a very serious issue. We aim to develop a fake image detector using convolutional neural networks that accurately detects the manipulated images and thus helps make people more informed about the authenticity of the images. The input to our algorithm is an image. We then use a convolutional neural network to output the predicted authenticity (authentic/fake) of the image. The complete code base for this project is also available for reference[14].

2 Related work

The proliferation of fake images has inspired increase in research to find ways to accurately detect fake images and thus curb its spread. Numerous approaches have been used in the literature to devise an accurate fake image detector. Huaxiao Mo, Bolin Chen and Weiqi Luo [1] developed a Convolutional Neural Network to identify Fake Faces generated by GANs. Their approach had a 99.4% accuracy. We found their approach of developing their CNN where they experimented with numerous architectures and finally achieve a model which had high accuracy interesting. Dong-Hyun Kim and Hae-Yeoun Lee [2] also used Convolutional Neural Networks to detect Image Manipulation and achieved a 95% accuracy. Peng Zhou, Xintong Han, Vlad I. Morariu and Larry S. Davis [8]

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

proposed a two-stream Faster R-CNN network and trained it end-to-end to detect the tampered regions in a manipulated image.

Furthermore, Minyoung Huh, Andrew Liu, Andrew Owens and Alexei Efros [7] proposed a model that learns to detect visual manipulations from unlabeled data through self-supervision. They compared three image processing techniques namely CFA (Color Filter Array), JPEG DCT and NOI (Noise Variance).

On the same dataset that our work is based on, Wei Wang, Jing Dong and Tieniu Tan [3][6] developed an image splicing detection method based on gray level co-occurrence matrix (GLCM) of thresholded edge image of image chroma. They utilized SVMs (Support Vector Machines) as a classification algorithm and used GCLMs of edge images as features in their approach. They were able to achieve 96% accuracy on the CASIA v2 dataset, albeit with considerable fearture engineering.

However, based on the literature available, we noticed that popular CNN architectures like Alexnet, Resnet, Densenet, Inceptionnet and VGG16 have not been used to implement fake image detectors yet.

3 Dataset and Features

We have used the CASIA v2 dataset[9] which has 7200 real images and 5331 altered (manipulated) images. Each altered image is either a single authentic image modified using professional image editing tools, or a combination of two authentic images.

We used a 80-10-10 split for the training, dev and test set respectively. Therefore, our training set had 10024 examples, while dev and test set had 1253 and 1254 examples respectively.

We preprocessed all images and resized them to a size that the different architectures we experimented with accept, to make subsequent training and evaluation runs faster and easier. Therefore the resolution of the input images depends on the CNN architecture that we used. Given below are the input resolutions we used with respect to various CNN architectures:

- Inception-net: 299×299
- Densenet: 224×224
- Resnet: 224×224
- VGG16: 224×224
- Alexnet: 224×224

An example of an authentic image and a corresponding fake image is shown below.

(a) Authentic Image

(b) Fake Image

Figure 1: Example of an authentic and fake image

3.1 Dataset pre-processing

We broadly used two approaches feeding the input image to the models.

1. We fed the input image as is (only reducing the resolution with respect to a particular model)
2. We applied Error Level Analysis to the image before feeding it to the model

ELA (Error Level Analysis) is a technique that identifies areas within an image that are at different compression levels and can be applied to images with lossy compression. For a JPEG image, the

entire picture should be at the same compression level. If a section of the image were to be at a significantly different level, it likely indicates a digital modification[3]. Using this technique, we pre-processed our dataset of JPEG images and then fed the images into the CNNs. This pre-processing step was vital as it aided the models and improved their performance greatly.

Figure 2 shows an authentic and fake image after undergoing Error Level Analysis:

(a) Authentic Image (b) Modified Image (c) Close up of modified section

Figure 2: Error Level Analysis on Images

4 Methods

After pre-processing the images, we trained and evaluated the models of varying architectures on our dataset to gain a sense of which models performed better. Hyper-parameter tuning helped push the performance of the best model by a small margin.

4.1 Loss Formulation

As this is a binary classification task at hand, we chose Cross Entropy Loss to optimize on. While Cross Entropy Loss penalizes both types of prediction errors (false negatives and false positives), it especially penalizes predictions where the model was confident but incorrect.

$CrossEntropyLoss = \sum(-y * log(p) - (1-y) * log(1-p))$

4.2 Network Architectures

Custom CNN is a simple shallow CNN we tested with two convolutional layers, Relu activation and Max pooling. This was used to understand how well a shallow CNN performs at detecting fake images.

AlexNet[11] was the first widely popular CNN architecture that beat traditional image classification methods. It is composed of five convolutional layers followed by three fully connected layers. It used the ReLu activation function instead of the then standard Tanh and Sigmoid for non-linearities.

VGG16[12] improved over Alexnet by replacing the larger kernels with smaller 3x3 ones, and stacking to create 11 or more layers. This network showed that the depth of the network was critical for good performance.

Inception[13] devised inception modules and a bottleneck layer that address the computational requirements needed by VGG16. It used the idea that an efficient deep CNN architecture should only have sparse connections between activations.

Resnet[5] was proposed to address the issue of vanishing gradients in very deep CNNs by using identity shortcut connections, parallel to the regular convolution layers. These shortcut connections can skip one or more layers, as shown in Figure 3. We used a 18-layer Resnet for our experiments.

Densenet[10] built on top of networks proposed earlier to address the vanishing gradients problem. Since all earlier netwokrs created shorter paths from earlier layers to later layers, Densenet simply connected every layer to all subsequent layers. This strengthened feature propagation, encouraged feature usage, and substantially reduced the number of parameters allowing for deeper networks.

Figure 3: A residual network compared to a regular deep network[4]

4.3 Training

We initialized these architectures with pre-trained weights and tested both training only the last layers, as well as training all available layers. We found that training all layers consistently outperformed training only the last layers, although it was computationally more expensive.

We utilized the Adam optimization algorithm[] to update the weights of the neural network. Adam is popular and been known to do well for CNNs.

4.4 Hyperparameter Tuning

The hyperparameters we tuned for our models were Mini-batch size, Number of Epochs, Learning Rate and L2 regularization rate. We used grid search and bayesian optimization to tune the parameters.

Grid Search simply tries every possible combination of hyperparameter values and returns the combination that gave the best dev set accuracy. We had a 256 cell grid with the following values.
Mini-batch size: 25, 50, 75, 100 *Number of Epochs*: 15, 35, 70, 100
Learning Rate: 0.00025, 0.0005, 0.00075, 0.001 *L2 Regularization*: 0, 0.005, 0.01, 0.05

Bayesian Model-based Optimization keeps track of past evaluation runs to form a model mapping hyperparameter values to a probability score on the objective function. We defined similar bounds as above and optimized for dev accuracy.

Both the methods of hyperparameter tuning gave similar results.

5 Results

Baseline Logistic Regression Model: We implemented a Logistic Regression model using a single neuron neural network as our baseline model. We achieved a dev set accuracy of 59% using this model.

CNNs: Of the numerous CNN architectures that we tested, we found that the Resnet architecture outperformed the others in terms of the dev set accuracy, with densenet coming in at a second. Apart from accuracy, we also looked at other metrics such as precision, recall and F1 Score.

We believe that the Resnet performed marginally better than the other architectures as it effectively takes care of the vanishing gradients issue while being a sufficiently deep (18-layer) network. It is also less computationally expensive compared to DenseNet and did a good job of generalizing to unseen data. We believe densenet performed well as a close second for similar reasons.

After finding the best performing CNN architecture, we performed hyperparameter tuning. The hyperparameters that we tuned were mini-batch size, learning rate and L2 regularization. We found that the optimal hyperparameters for the Resnet architecture with all weights unlocked were:

$LearningRate = 0.00025 \quad Mini - batchSize = 100 \quad L2Regularization = 0.005$

We used this model architecture and trained it using both original images (resized to 224x224), as well as ELA preprocessed images.

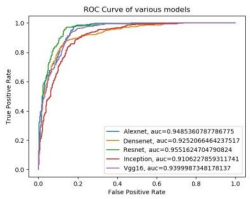
We observed a stark increase in accuracy when Error Level Analysis was used in the pre-processing step. We obtained an accuracy of 75% when the training was done on original images and an accuracy of 94% when the training was done on images that were preprocessed using Error Level Analysis. The varying compression levels in modified images clearly aided the model.

Metrics observed for our model:

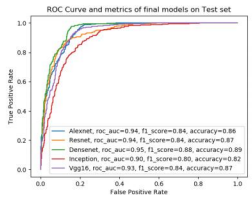
$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = 0.94 \quad Precision = \frac{TP}{TP+FP} = 0.9212$$

$$Recall = \frac{TP}{TP+FN} = 0.9355 \quad F1Score = \frac{2*Precision*Recall}{Precision+Recall} = 0.9283$$

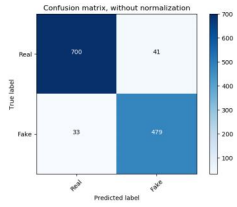
Figure 4 shows the ROC Curves and the best model's Dev set confusion matrix.



(a) ROC Curve (Dev Set)



(b) ROC Curve (Test Set)



(c) Confusion Matrix for best model

Figure 4: ROC Curves and Confusion Matrix

Table 1 shows the best models for each CNN architecture and their performance metrics.

Model architecture	Accuracy on raw	Accuracy on ELA	Precision on ELA	Recall on ELA	F1 Score on ELA
AlexNet	67%	90%	83%	94%	88%
VGG16	68%	88%	82%	89%	86%
Inception	64%	91%	86%	93%	89%
ResNet	75%	94%	92%	93%	93%
DenseNet	67%	89%	94%	95%	89%

Table 1: Comparison of various CNN Models

Evaluating our tuned ResNet model on the test set (pre-processing using ELA) gave us an accuracy of 88%.

Error analysis: Looking at the examples that this model got wrong, we found there is no particular type of image that it was prone to misclassify. The errors were equally distributed across the various categories (nature, art etc). One interesting aspect about the images is that they tended to be smaller of size around 250 x 350 before they were resized whereas typical dataset images were of size around 800px. This suggests that there may not have been enough resolution of information for the altercation in the image to be detected. In addition, some errors were apparent from the limitations of the ELA pre-processing. For a few images, altercations were not detected and so an a diff image of all black essentially was generated which ended up being classified as real when it was fake.

6 Conclusion/Future Work

We found that the Resnet CNN architecture outperformed the other architectures. We also observed that performing ELA (Error Level Analysis) on the images before feeding them to the CNN helped increase the prediction accuracy considerably. We were able to increase our accuracy from 59% (our baseline model Logistic Regression) to 94% using a ResNet.

Next Step: Trying ensemble techniques like Bagging and Boosting have shown great promise in tackling variance issues in models and generalizing well to unseen data. Therefore, creating numerous models and leveraging ensemble techniques can further help to develop a more robust model and also increase the classification accuracy.

Next Step: An End to end learning approach. Our best model relied on hand engineered features. An area for future research is training an end-to-end model that performs equally as well on this task using raw inputs. This would also be beneficial to other image types where or cases error level analysis is not as effective. Finally, we want to build defenses against adversarial examples/attacks.

7 Contributions

All the team members contributed equally to the project.

References

[1] Mo, Huaxiao & Chen, Bolin & Luo, Weiqi (2018) Fake Faces Identification via Convolutional Neural Network, *Proceedings of the 6th ACM Workshop on Information Hiding and Multimedia Security*, pp. 43–47. New York, NY, USA: ACM.

[2] Kim, Dong-Hyun & Lee, Hae-Yeoun (2017) Image Manipulation Detection using Convolutional Neural Network *International Journal of Applied Engineering Research*, pp. 11640–11646. : Research India Publications.

[3] J. Dong & T. Tan (2009) Effective image splicing detection based on image chroma *2009 16th IEEE International Conference on Image Processing (ICIP)*, pp. 1257–1260. : IEEE.

[4] Jordan, Jeremy., Common Architectures In Convolutional Neural Networks., 2019, https://www.jeremyjordan.me/convnet-architectures/. Accessed 10 Mar 2019.

[5] He, Kaiming, et al., Deep residual learning for image recognition., *Proceedings of the IEEE conference on computer vision and pattern recognition*., 2016.

[6] J. Dong, W. Wang & T. Tan (2013) CASIA Image Tampering Detection Evaluation Database., *2013 IEEE China Summit and International Conference on Signal and Information Processing*,pp. 422-426 , Beijing.

[7] Huh M., Liu A., Owens A., & Efros A.A. (2018) Fighting Fake News: Image Splice Detection via Learned Self-Consistency. In: Ferrari V., Hebert M., Sminchisescu C., Weiss Y. (eds) Computer Vision – ECCV 2018. *Lecture Notes in Computer Science, vol 11215*., Springer, Cham

[8] Zhou, P., Han, X., Morariu, V.I., & Davis, L.S. (2018). Learning Rich Features for Image Manipulation Detection.,*2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*,pp. 1053-1061.

[9] Sovathana, Phat., CASIA Dataset., https://www.kaggle.com/sophatvathana/casia-dataset., Accessed 15 Jan 2019.

[10] Huang, Gao, et al., Densely connected convolutional networks., *Proceedings of the IEEE conference on computer vision and pattern recognition*., 2017.

[11] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton., Imagenet classification with deep convolutional neural networks., *Advances in neural information processing systems*., 2012.

[12] Simonyan, Karen, and Andrew Zisserman., Very deep convolutional networks for large-scale image recognition., arXiv preprint arXiv:1409.1556 (2014).

[13] Szegedy, Christian, et al., Going deeper with convolutions., *Proceedings of the IEEE conference on computer vision and pattern recognition*., 2015.

[14] Kosaraju, Muindi, Lundia., 'Deepfake Detector Code Repository'. Github, 2019, https://github.com/jervisfm/cs230-project. Accessed 20 Mar 2019.