.

Tampered image detection on a budget

Christian Millsop
Department of Computer Science
University of California
cjmillsop@ucdavis.edu

Min Sung Kim
Department of Computer Science
University of California
mngkim@ucdavis.edu

Jinyoung Pung
Department of Computer Science
University of California
jpung@ucdavis.edu

Zahra Aminiranjbar
Department of Electrical Engineering
University of California Davis
zaminiranjbar@ucdavis.edu

Abstract

An image speaks a million words. One look at an image and we can write a book about it. On the other hand, images can leverage the spread of false information. Our initial goal was to improve performance on the SOTA in tampered image detection using the CASIA v2 dataset and Error Level Analysis. We found that a simple CNN model recently proposed on a related dataset outperformed all of our attempts with an accuracy of 89% and AUC of 96%. Using Knowledge Distillation we were able to shrink this model by 4,321x of its original size and maintain an accuracy and AUC of 86% and 92% respectively. In order to quantify the usability impact of shrinking the model, we deployed the baseline CNN and distilled CNN to an Android device with Tensorflow Lite and found that the inference time had been reduced by 4.83%.

1 Introduction

Fake images are becoming more common and can spread rapidly through social media, creating serious social issues, like defamation and political instigation through the distortion of facts. Such manipulated photos were shared through social media during the U.S. presidential election campaign, such as a Facebook political ad in which Democratic candidate Joe Biden was shown wearing an earphone and receiving coaching in real time during the first TV debate with President Donald Trump—this was a fake photo[14]. Similarly, in July 2020, a photo of President Trump in a car hugging Jeffrey Epstein, who had been accused of luring children into sexual abuse in the financial district, spread quickly through Twitter. However, the photo had been fabricated from an original of Donald Trump holding his daughter, Ivanka[9].

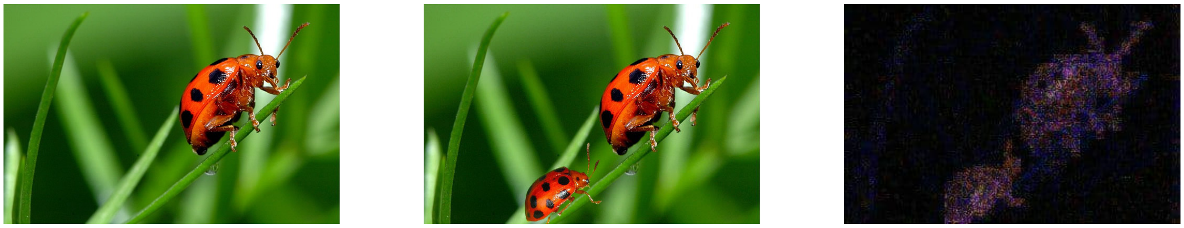
The problem of fake images lingers even after it has been determined that the images were forged, and memories and feelings cannot be erased, despite knowing that the images are fake [13] [10]. Unfortunately, most fake photographs are not easy to distinguish with the naked eye, and manipulation techniques are becoming increasingly sophisticated. More efforts are needed to prevent the malicious damage caused by fake images using equally sophisticated technology to detect them. If highly accurate fake image detection algorithms can be implemented on social media platforms, they could, in real time, detect manipulated images and prevent them from spreading.

Considering that the Internet is one of the major sources of information which is easily accessible with mobile devices, fake detection capability on the edge devices needs to be ensured. Although many studies could suggest useful fake detection models[17][11][18], deep neural network models require tons of parameters to be stored on the memory, and computational cost through feed-forward neural network is expensive.[15][4] Running inference on such heavy neural network models in the mobile environment can be demanding due to devices' limited resources.

For better performance on the edge devices, we compressed our models by knowledge distillation and quantization. Knowledge distillation is a model compression technique that builds simplified student models and let them learn from the teachers, heavy original models. On the other hand, quantization utilizes reduced precision variables to downsize models. We designed our own student models and trained them to learn from the original models. And then, we quantized both teacher and student models with the off-the-shelf solution, TensorFlow Lite, to conduct inference on android. In this way, we compared the accuracy and time-cost of models on a mobile device.

2 Dataset and Features

For this project, we have used the CASIA v2 dataset [8]. This dataset includes 7437 real images and 5123 tampered images (40.8% tampered). 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 70-15-15 split for the training, validation, and test set respectively. We preprocessed all the images and resized them to a size that is acceptable by the different architectures and also to make subsequent training and evaluation faster. An example of an authentic image and a corresponding tampered image is shown in Figure 1 below.



(a) Real Image (b) Tampered Image (c) ELA of Tampered Image

Figure 1: Example of an authentic and Tampered image

2.1 Data preprocessing

1. **Error Level Analysis (ELA)** is a filter that allows identifying areas within an image that are at different compression levels. With JPEG images, the entire picture should be at same compression

level.[12] If a section of the image is at a significantly different error level, then it likely indicates a digital modification. ELA highlights differences in the JPEG compression rate. Looking at the ELA modified figure (figure 1) you can identify the high and low contrast edges. Comparing the sharp edges of the bottom lady bug to the one on top identifies that there may have been digital alteration. We used ELA and pre-processed our dataset of JPEG images and then fed the images into the CNNs. This preprocessing increased our model efficiency greatly as it will be discussed in the result section.

2. **Contrast Maximizing** was used to normalize the image. A histogram of the image was calculated and remapped the image so that the darkest pixels were black (0) and the brightest pixels were white (255).

3 Methods

We started our experiment with developing models for tampered image detection and later used knowledge distillation and Tensorflow Lite to detect the tampered images on a budget. this section includes details of the methods we used and developed.

3.1 Network Architectures

1. **Baseline CNN** is a state of the art CNN model that we tested with two convolutional layers , Relu activation, Max pooling , dropout , a fully connected layer with Relu activation, another dropout and a final fully connected layer with Softmax activation. This model was used to understand how well a baseline CNN model performs at detecting tampered images.

2. **ResNet** was proposed instead of a deep neural network to address the issue of vanishing gradients by using identity shortcut connections, parallel to the regular convolution layers. [4] We used a 50-layer ResNet model with 176 layers and two fully connected layers with 'ImageNet' weights for our experiment.
   We tested numerous CNN architectures and found that the CNN baseline model outperforms others in terms of the test set accuracy. However, we observed a huge increase in accuracy when Error Level Analysis was used in the preprocessing steps for all the models. With our ResNet50 model, We obtained an accuracy of 75% when the training was done on original images and an accuracy of 90% when training was done on images that were preprocessed using Error Level Analysis.
   While training our ResNet model, we tested training only at the last layers as well as training more available layers. We found that the best accuracy is obtained when we first train the last two dense layers while freezing other layers. We then unfreeze a few last layers and continue the training using a smaller learning rate for our optimization algorithm. A larger learning rate would soon result in overfitting. After finding the best performing ResNet architecture, we performed hyperparameter tuning. The hyperparameters that we tuned were mini-batch size and learning rate. We found that the optimal mini-batch size and hyperparameters for the ResNet architecture were 0.0001 and 32, respectively.

3. **EfficientNet** is a CNN model that improves network accuracy and efficiency by scaling three factors: width, depth, and image resolution, which was first introduced in 2019[15], and reached state of the art accuracy in image classification tasks. As the number of parameters increases, from model EfficientNetB0 to model EfficientNetB7, the accuracy improves on the Imagenet data set. However, there was no further improvement from B0 to B7, even though EfficientNetB0 was much faster in this task. We used the weights pre-trained on ImageNet and decapitated the final dense layer which is used for 1000 ImageNet class predictions. Two dense layers are appended and used a softmax function for prediction of the two classes.

3.2 Knowledge Distillation

The practice of Knowledge Distillation (KD) is founded on the hypothesis that a model's classifier probability outputs contain useful information. The teacher model is the standard, full-sized classifier that we are trying reduce in size. We then construct a complementary student model that has a reduced number of layers and layer sizes. Our student model follows the same general architecture as the teach model, however there is no consensus in the literature that this should be a requirement[5]. After training the teacher model, we make predictions on the training dataset and then use that as an additional label for our student model to predict. The student has two outputs: (1) the standard authentic versus tampered classification and the other (2) is the knowledge distillation output.

The loss function used for KD, based on Cho et al. 2019[3], is as follows:

$$Loss_{total} = \alpha Loss_{CE} + (1-\alpha) Loss_{KD}$$

$$Loss_{KD} = -\tau^2 \sum^{i} (s_i^{t,\tau} log s_i^{s,\tau})$$

where $s^{t,\tau}$ is the softmax-temperature of the teacher and $s^{t,\tau}$ is the softmax-temperature of the student.

The softmax-temperature is a modified softmax with smoothing:

$$s^{,\tau} = \frac{\exp \frac{z_j}{\tau}}{\sum \exp \frac{z_k}{\tau}}$$

The α is the loss weight coefficient. It is used to control the contribution of KD versus the typical Cross Entropy loss. For most experiments we used $\alpha = 0.9$. In a single experiment we varied the impact to 0.8 and noted a significant negative effect. The τ is the temperature of smoothing. Our baseline value for this is 4, however we also tested [1,10,20] to explore the effect.

Beyond the formulaic hyperparameters for KD, the method by which the teacher probabilities are obtained and how long the student is taught also impact the outcome of KD. In our experiments we extracted teacher probabilities after 5 of the total of 20 training epochs. The rationale behind this is that a underfit teacher tends to generalize better to the student. We did not explore this hyperparameter. The student teaching duration was explored in our experiments. To do this we used the $Loss_{total}$ as described above for a set number of epochs, and then removed the KD output and reverted the Loss function back to cross entropy for the remainder of the training epochs.

In addition to the teacher and student models, we also trained a student-sized model from scratch as a control to understand whether KD improved student training.

This methodology closely follows the methodology provided by Cho et al. Hyperparameter sweeps were performed in triplicate using fixed initialization conditions. Weights and Biases was used to keep track of hyperparameter sweeps[1]. The final distilled model for the baseline CNN was separately verified with three-fold cross-validation and three randomizations of initialization conditions.

3.3 Quantization

Quantization is the trade-off between precision and memory occupation. It converts heavy, precise representation of numbers to light, approximate representation. Quantizing neural network models is beneficial in the edge computing environment, considering limited memory capacity and low computing power. Reduced format not only saves memory but also boosts computation by efficient use of cache and register.

We implemented TFLite to embed our models to a mobile device and observed how they works in the actual edge computing environment. TFLite is a commercial off-the-shelf solution for model compression that utilizes quantization. It converts 32-bit floating number representations in regular tensorflow models to unsigned 8-bit integers by scaling the value into 0 to 255 range[16].

4 Results

4.1 Model Evaluation

Our results for the three model architectures that we tested are below. All models performed well using the ELA data. The primary difference in performance between these models was the AUC score. The CASIA v2 dataset is slightly imbalanced with 40.8% of samples being tampered.

Two of the models, baseline CNN and EfficientNetB0, were distilled. Most of our distillation effort was focused on the baseline CNN model, which we were able to reduce in size significantly.

		CNN	ResNet50	EfficientNetB0
Teacher	Accuracy	89	89	89
	AUC	96	95	92
	parameters	33,583,266	23,792,713	20,106,661
Distilled	Accuracy	86	-	77
	AUC	92	-	70
	parameters	7,772	-	3,538,984

Table 1: Model performance

4.2 Model Budget

Teacher and student models are saved in .h5 format and converted to .tflite format. We measured the size of models before and after model compression process.

	Teacher	Teacher TFLite	Distilled	Distilled TFLite
CNN	345.98	115.316	0.127	0.033
ResNet	95.051	92.570	-	-
Eff	109.455	36.154	41.110	13.553

Table 2: Model size (MB)

The elapsed time of a single inference for each model is measured on an Android device, Samsung Galaxy Note8. We conducted 100 inferences on converted TFLite models for both authentic and tampered images respectively, and calculated mean elapsed time.

	Teacher TFLite	Distilled TFLite
CNN	181.455	8.77
ResNet	247.66	-
Eff	73.75	40.705

Table 3: Time-cost per inference (ms)

4.3 Knowledge Distillation

We didn't notice a large effect from KD until the CNN network had been shrunk to a shadow of its former size. The student-sized control model was surprisingly capable even with a very small number of layers and neurons. Once we were able to degrade the performance of the control model such that we saw at least a 10% drop in accuracy compared to the baseline model, we were then able to investigate the usefulness of KD.

CNN Size	Dense Size	Parameters	Student	Control
			% Change Accuracy / AUC	% Change Accuracy / AUC
4	2	31,064	-2.2 / -1.9	-1.9 / -4.1
2	2	15,536	- 2.4 / -4.1	-2.0 / -4.4
1	4	15,466	-4.2 / -4.6	-10.8 / -10.0
1	**2**	**7,772**	**-3.8 / -4.5**	**-12.7 / -11.8**
1	1	3,925	-4.9 / -5.6	-34.0 / -21.1

Table 4: Best case KD results at varying model sizes. Smaller numbers are better. Final model in bold.

A total of 31 KD experiments were performed on the baseline model. We found that knowledge distillation hyperparameters either had limited effect or were essential to the KD process. Table 4 presents some of the small student models that we created and the percent change in performance relative to the teacher model. Only the results at optimal parameters are included. Interestingly, the control model did out-perform the student model in some cases.

5 Discussion

5.1 Model Evaluation

Our initial goal was to develop models for tampered image detection. We achieved our initial goal through researching and reproducing the SOTA the art model as well as developing ResNet and EfficientNet models with accuracy close to the SOTA.In Table 1, the teacher model demonstrates our success in achieving high accuracy as well as AUC for detecting tampered images.

5.2 Model budget

After achieving high accuracy tampered image detection goal we took a step further to make our complex model compatible with edge devices. However, most of these models are too expensive computationally to run on devices like mobile phones. Further research brought us upon knowledge distillation which refers to the idea of model compression by teaching a smaller network, step by step, exactly what to do using a bigger already-trained network. Therefore using our bigger already trained networks we developed our knowledge distillation model. It is important that this model can sustain the accuracy achieved by the bigger model(teacher model). Table 1. The distilled section includes our results for this step. We were able to successfully train a student model while keeping the accuracy close to the teacher model. Table 2 also demonstrates the size of models before and after model compression process.
Finally, we put all the pieces together and implemented Tensorflow Lite to embed our models to a mobile device. We were successful in using our model on a mobile phone that has a much smaller memory as well as lower computational power than our laptop CPU. Table 3 includes the time cost per inference of the teacher model as well as the distilled model. The results clearly indicated that our distilled TFLite model runs much faster on an edge device.
Therefore we were able to develop complex models to identify tampered images. We successfully shrank our model down using knowledge distillation and ran our models on an edge device. On the edge device, our distilled models performed faster than the teacher model while maintaining most of the teacher's accuracy.

5.3 Knowledge Distillation

Our final distilled model is illustrated in Figure 2. It exceeded our expectations in terms of model size with respect to performance. We verified the model performance with 3-fold cross-validation and reviewed teacher and control performance, but could not identify any obvious reasons for the surprising result. Inclusion of the control model proved an important design decision since it allowed us to measure the treatment effect of KD versus the incidental effects of a smaller model.

The distilled model's architecture is very similar to the teacher model. The distilled model omits a secondary CNN layer immediately after the first and shrinks the size of the first CNN and first Dense layer from 32 to 1 and from 256 to 2 respectively. With such a limited neural network, we consider whether the model is fitting to discrete patterns in the data rather than learning to recognize general patterns that would indicate the presence of tampering.

The KD hyperparameter sweep provided insight into the KD process and in general agreed with the prior work of Cho et al.

One perspective on the hyperparameters is that they control how strongly the student optimizes for the teacher probabilities versus the class labels. In a sense, by applying KD we are relocating the starting point for gradient descent. Once we stop KD training, we return to optimizing the cross entropy loss over the error surface. Perhaps when the KD training loss is too dominant, then we overshoot optimal minima and are left in remote areas of the error surface. We observed that as the network size decreased, the sensitivity to KD hyperparameters increased. At low levels they either had no effect or improved the model results. At high levels they tended to be catastrophic and prevented the model from ever converging.

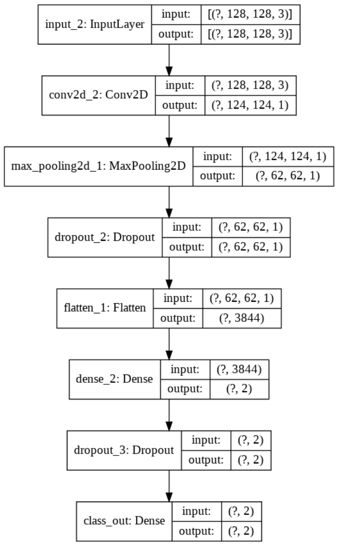


Figure 2: Distilled Architecture

The temperature hyperparameter, τ had limited effect. At large values (ie. 10,20) it reduced the performance of the student model slightly. We saw no effect at a value of 1. In this case the teacher probabilities would be unsmoothed and the loss function reduces to something similar to the binary cross entropy except against continuous labels rather than integers.

The most interesting hyperparameter was the number of epochs of KD that our student was trained with. As the model got smaller, we reduced the training epochs to 1 and still saw an improvement in the distilled model versus the control model. At high values the model would not converge. This hyperparameter was very sensitive to network size.

5.4 Future Work

Based upon our analysis, we believe that there are several avenues to explore applicability of our results and to improve on our approach to tampered image detection.

The size and performance of the distilled model are very surprising to us. With this level of efficacy, we would expect KD to be more widely used. Additionally the original CNN model was 4,321x larger. We wonder if the authors of the CNN had explored any smaller models and how they came to their final architecture. Presumably with many fewer weights in our distilled model, it will have limited ability to generalize to new data. Our current proposal for investigating generalization includes additional cross-validation, mutations of the CASIA v2 dataset, new datasets, and modified ELA preprocesing to ensure that no information is leaking during preprocessing.

We are also interested in improving tampered image detection by investigating other image preprocessing techniques. In this case, we would use both the ELA data and the investigational data as inputs to the same model.

6 Author contributions

Christian Millsop : I developed the Knowledge Distillation process, applied it through experimentation on the CNN baseline model, and wrote the corresponding sections about it.
Min Sung Kim : I developed ELA Function and Image preprocessing. And I developed binary classifier using EffienctNet and its distillation model, and wrote the corresponding sections about it.
Jinyoung Pung : I quantized both original and distilled models, observed the performance of the converted models in a mobile device, and wrote the related sections in the report.
Zahra Aminiranjbar: I reproduced the CNN baseline model and the ELA, developed Resnet model and wrote the related sections as well as parts of our discussion in the report.

References

[1] Lukas Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020. URL: https://www.wandb.com/.

[2] Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

[3] Jang Hyun Cho and Bharath Hariharan. "On the Efficacy of Knowledge Distillation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019.

[4] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[5] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. *Distilling the Knowledge in a Neural Network*. 2015. arXiv: 1503.02531 [stat.ML].

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[7] Yann A LeCun et al. "Efficient backprop". In: *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48.

[8] namptpham. *'CASIA 2 . Groundtruth'*. https://github.com/namtpham/casia2groundtruth. 2013.

[9] RETERS. *Fact check: Photo of Trump embracing Epstein is an edited picture of Trump and his daughter Ivanka*. https://reut.rs/2I8tlrM. 2020.

[10] Dario LM Sacchi, Franca Agnoli, and Elizabeth F Loftus. "Changing history: Doctored photographs affect memory for past public events". In: *Applied Cognitive Psychology: The Official Journal of the Society for Applied Research in Memory and Cognition* 21.8 (2007), pp. 1005–1022.

[11] Ronald Salloum, Yuzhuo Ren, and C-C Jay Kuo. “Image splicing localization using a multi-task fully convolutional network (MFCN)”. In: *Journal of Visual Communication and Image Representation* 51 (2018), pp. 201–209.

[12] Sankalap. *Image manipulation detection using Deep Learning*. `https://medium.com/datadriveninvestor/image-manipulation-detection-using-deep-learning-dedcb7a84d06`. 2020.

[13] Cuihua Shen et al. “Fake images: The effects of source, intermediary, and digital media literacy on contextual assessment of image credibility online”. In: *New media & society* 21.2 (2019), pp. 438–463.

[14] Andrew Solender. *Trump Ads Feature Biden Photo Edited To Include AirPod, Asking ‘Who’s In Joe’s Ear?’* `https://www.forbes.com/sites/andrewsolender/2020/10/01/trump-ads-feature-biden-photo-edited-to-include-airpod-asking-whos-in-joes-ear/?sh=47cfce32fd01`. 2020.

[15] Mingxing Tan and Quoc V Le. “Efficientnet: Rethinking model scaling for convolutional neural networks”. In: *arXiv preprint arXiv:1905.11946* (2019).

[16] Tensorflow Official Website. *Model optimization*. `https://www.tensorflow.org/lite/performance/model_optimization?hl=en`. 2020.

[17] Yue Wu, Wael Abd-Almageed, and Prem Natarajan. “Image copy-move forgery detection via an end-to-end deep neural network”. In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2018, pp. 1907–1915.

[18] Peng Zhou et al. “Learning rich features for image manipulation detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1053–1061.