Efficient Detection of Manipulated Images

Nisarg Trivedi
University of Michigan
Ann Arbor, MI
nisargtr@umich.edu

Chenwu Liao
University of Michigan
Ann Arbor, MI
liaocw@umich.edu

Matt Wilmes
University of Michigan
Ann Arbor, MI
mwilmes@umich.edu

Abstract

Manipulation and alteration of images are becoming more prevalent due to easy availability of modern-day image editing software. It is possible to create a well-crafted manipulated image with minimal efforts that can deceive a human brain and lead to false interpretation. Considering the greater challenges posed by efficient image manipulating systems, we aim to devise intelligent neural network mechanism that can well-distinguish between genuine and manipulated images.

In this paper, we approach this challenge from two different perspectives and opt for an integrated model that successfully determines whether an image is real or fake. The first approach is to use a deep neural network as a binary classifier trained on vast amounts of data which takes input images and produces a real versus fake prediction. The second approach is to determine self-consistency between patches of an image with a goal of identifying inconsistent image-patches / splices. At the end, we combine both trained models and work towards making a unified prediction from the outputs of both of these networks. This method can potentially boost the network accuracy and make it sensitive to finer manipulations, such as insertion of splices or abnormal zooming. We describe several important experiments with our network and note our observations for improving the network prediction.

1. Introduction

Due to the accessibility of present-day image editing tools such as Adobe Photoshop®, manipulating images that can pass as authentic has become much easier [17]. The Internet and social media platforms have fueled the spread of these altered images [9, 12]. These malicious images may potentially cause great social and economic harm. For example, in a study of real and fake images posted on Twitter during Hurricane Sandy, it was found that the fake images were shared almost twice as much as real ones. Such photos slowed down the response to this disaster [5, 12].

Kumar and Shah [12] make an important distinction between *misinformation* and *disinformation*–the latter indicating an intent of misleading and deceiving the viewer. The threat of the spread of disinformation continues to increase and fetches great attention from forensic scientists, law enforcement agencies and the government. As studied in [9], the current methods for detection of manipulated images are slow and less efficient compared to the spread of disinformation.

In our solution to this challenge we choose two main aspects to focus on. The first one is quickly detecting these manipulated images before they spread. To process images as quickly as possible, it is important for us to come up with a solution that is minimally complex with significant accuracy. The second aspect is the wide range of methods used to manipulate images such as cut & move, splicing, object duplication etc. [1]. To account for this, we make sure to train on a diverse data set from multiple sources with significant variance in image features such as camera angle, lighting, different object categories etc.

To achieve these goals, we designed and implemented our own network architecture. The basic architecture consists of a binary classifier that determines whether an image is real or fake. We also implemented our own patch-consistency estimator to aid this binary classifier in its task. A significant feature of our approach is that both the networks are trained independently and their predictions are combined during testing to make a final determination. This novel network architecture has the potential to perform significantly better than the binary classifier can alone. Furthermore, the model gives consistent accuracy when handling a diverse data set of images and processes large data sets with high throughput.

2. Related Work

2.1. Supervised Learning Methods for Detection

We find several supervised learning approaches in the literature for the detection of manipulated images using neural networks. Earlier attempts were focused on leveraging *a*

priori knowledge, but more recent methods have started using labeled data [10, 14] in an attempt to enlarge the scope of possible image manipulations. Labels for an image can be generated at varying degrees. At a high level, one can label the entire image with a specific class. This can either be a binary classification, such as real or fake, or there can be more than two classifications, such as labels representing the method(s) used to manipulate an image [3]. More granular degrees of labeling can be on a pixel-by-pixel basis, as is done by Salloum *et al*. [14]. In this method, each pixel is given a binary classification such that a mask is produced indicating the spliced region in an altered image. A model can then learn to predict its own mask for test images [10, 14].

Although this latter method produces a higher level of detail, binary masks cannot be easily generalized to other forms of manipulation and require more complex networks. Because our goal was to design a simpler architecture that could be generalized to many forms of manipulation, we decided to label each image in our dataset as either real or fake on a higher level.

2.2. Few Shot Learning

Our study finds that many networks attempting to detect fake images are trained on a specific type of manipulation such as splicing [10, 14], CNN-based generation [18] or warping by Photoshop [17]. However, Cozzolino *et al*. [3] note that the accuracy of these networks can fall to nearly 50% (random chance) when tested on images altered by different manipulation methods.

Human-beings have an impressive ability to detect alterations / anomalies by learning from few examples. This concept, called *few shot learning*, can be applied to neural networks to help them identify classes for which they have few examples [2, 13]. *Feature extractors* are specifically trained to pick out low-level details in images that go beyond simple visual features [13]. A model can be trained to use these features to make inferences about new images it sees in terms of whether an image has been manipulated, even if the model has seen limited examples of that type of alteration. Chen *et al*. [2] demonstrate that ResNet-18 with an input size of 224×224 followed by a classifier works well for the task of few shot learning. Although this method is generally utilized for multi-class objectives, Cozzolino *et al*. [3] prove this strategy works for binary classifiers as well, which is why we chose to implement it.

2.3. Image Consistency

In their paper, Huh *et al*. [10] successfully train a network to predict the spliced mask of an image without ever training on manipulated images. Their main approach to this task is an *EXIF-consistency model*. All images have EXIF tags that contain metadata about an image such as date, camera model, brightness value, etc. The EXIF-consistency model takes in a pair of 128×128 image patches and passes them through identical ResNet-50 *Siamese networks*. The outputs are concatenated and passed into a series of fully-connected layers, ending in a vector of 83 binary classifications. These classifications are a determination by the network as to which of these 83 categories (80 EXIF features and 3 augmentation operations) are identical between the two patches. Using this EXIF-consistency model, each patch of the image is exhaustively compared with all other patches in the image to form a *response map* that indicates the pair-wise consistency. This is done for each patch, and these response maps are merged into a *consistency map*, which shows the consistent and inconsistent regions of the image.

Huh *et al*. also demonstrate a similar model without EXIF features, termed as an *Image-consistency model*. It gets outperformed by their EXIF-consistency model in most of the tests, however, they acknowledge that this model could potentially produce similar accuracy if given more time and data to train.

To aid our binary classifier with an Image-consistency model of our own, we draw inspiration from this approach by Huh *et al*. To address some of the shortcomings of their approach, we make several changes: 1) instead of passing image patches into our Siamese network, we pass patches from the features map generated by our feature extractor, 2) we train with fake images as well in order to better assist our network with identifying features indicative of inconsistent pairs, 3) we choose fixed patch locations instead of random locations to easily deal with overlap between patches.

3. Our Method

We segmented our project into four phases. In phase-1, we devised efficient image pre-processing and data-loading techniques. This included quick loading of around 12,000 images from different data-folders on cloud storage, their batch-wise segregation, normalization, shuffling and creation of data sets and data-loaders. This, along with useful data-augmentation techniques is described in section **4.1**. In phase-2, we successfully implemented our own deep neural network performing *Binary Classification* into real or fake images. The phase-3 involved implementing from scratch the algorithms for image *Patch-extraction and Consistency Estimation* through *Contrastive Loss on a Siamese Network*. In phase-4, we combined the independently trained binary classifier and image self-consistency estimator to collectively determine manipulation in an image.

3.1. Binary Classifier

The *Binary Classifier* takes images as an input and produces a real versus fake prediction. The network consists of a *Feature Extractor* and two fully connected layers. We use

Layer (Type)	Output Shape	Param #
CBR-1	[-1, 64, 112, 112]	9,536
MaxPool2d-1	[-1, 64, 56, 56]	0
CBR-2,3,4,5	[-1, 64, 56, 56]	147,968
CBR-6,7,8,9,10	[-1, 128, 28, 28]	525,568
CBR-11,12,13,14,15	[-1, 256, 14, 14]	2,099,712
CBR-16,17,18,19,20	[-1, 512, 7, 7]	8,393,728
AdaptiveAvgPool2d	[-1, 512, 1, 1]	0
FC-1 + LeakyReLU	[-1, 128]	65,664
FC-2	[-1, 1]	129
Total Trainable Parameters		11,232,769

Table 1: Details of the Binary Classifier

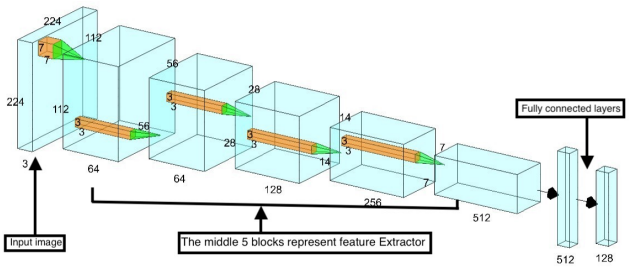

Figure 1: Binary Classifier Structure

ResNet-18 pre-trained on ImageNet [4] for feature extraction as seen in **Figure 1**. It contains multiple Convolution-BatchNorm-ReLU (CBR) blocks with residual connections. These skip connections, along with batch-normalization, solve the problem of vanishing and exploding gradients and enables us to train very deep neural architectures [7].

Since our task is different from the ImageNet Classification, we remove the last layer of ResNet-18 and append a couple of linear classifiers that output a binary score (of real vs. fake) at the end. Once our images are pre-processed, we pass them into the feature extractor in batches of 512 or 256 images at a time which generates a feature map for each image in the batch. This is then passed into our image classifier (linear layers) giving a single output score for each image. Finally, we use a binary cross-entropy loss to map the network score with true labels (0 for real images and 1 for fake images).

The learning objective for our network is detection of the manipulated images, which is significantly different from the ImageNet classification into 1000 categories. So we experimented with 3 different approaches of integrating the pre-trained ResNet-18 in our task – 1. freezing all pre-trained parameters, 2. fine-tuning only the last few blocks of ResNet-18, and 3. fine-tuning entire ResNet-18. The highest classification accuracy was achieved while fine-tuning the entire feature extractor, affirming our conjecture. The results of these experiments are explained in section **4.3.1** and can be seen in **Table 4**.

Layer (Type)	Output Shape	Param #
CBLR-1	[-1, 128, 14, 14]	74,112
MaxPool2d-1	[-1, 128, 7, 7]	0
CBLR-2	[-1, 256, 7, 7]	295,680
MaxPool2d-2	[-1, 256, 4, 4]	0
CBLR-3	[-1, 512, 4, 4]	1,181,184
MaxPool2d-3	[-1, 512, 2, 2]	0
AdaptiveAvgPool2d	[-1, 512, 1 1]	0
Total Trainable Parameters		1,550,976

Table 2: Details of the Self-consistency Estimator

3.2. Image Self-consistency Estimator

3.2.1 Network Architecture

This network determines consistency between two image patches. Each image is passed through the first five layers of ResNet-18 to get a feature map of 64 channels and 56×56 spatial dimensions. Then, the image patches are extracted in a structured manner, as described in section **3.2.2**, and resized to 28×28 spatial dimensions.

We created a Siamese network structure inspired from the work by Huh *et al*. [10] wherein two patches of a pair are passed through the same network. The network shares the same weights for both patches and is expected to produce different outputs for patches from different images and similar outputs for patches from the same image. The former is defined as an inconsistent patch-pair and the latter as a consistent patch-pair. The Siamese network consists of three groups of 3×3 Convolution-BatchNorm-Leaky ReLU (CBLR) blocks, each followed by one max pooling layer. After this, there is an adaptive average pooling layer and a linear layer producing an 8-element vector for each image. To initialize the weights of this model, we used *Kaiming Initialization* [6].

Patches are passed into the network as a 128×28×28 pair (x_1, x_2) and then subsequently separated inside the Siamese network, which produces feature embeddings $f(x_1)$ and $f(x_2)$. We use a distance function $d(x_1, x_2)$ (1) that takes the absolute difference of the two feature embeddings. The result is then passed into a fully connected layer producing a 2-element output vector z.

$$d(x_1, x_2) = |f(x_1) - f(x_2)| \tag{1}$$

3.2.2 Patch Extraction from Images

The patch-extraction is efficiently carried out at large scale with variable spatial dimensions however the choice of particular patches from an image to pass into the model proved

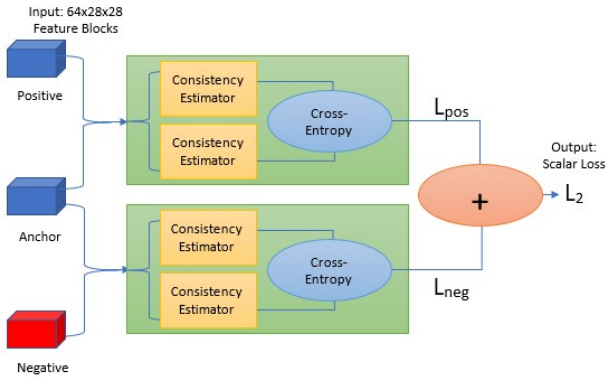

Figure 2: Processing positive and negative patch-pairs through the Siamese Network.

to be a challenging task. The number of possible pairs / triplets to extract grows cubically with the number of examples [16] and it's infeasible to process them all. For example, 100K to 1M identities are used for training Siamese networks that detect face similarity.

The first stage of Image Self-consistency Estimator involves extracting patches of various spatial dimensions from feature map, and then resizing them each to 64×28×28. Our experiments show that extracting fixed-size patches from each part of an image boosts the estimator's accuracy in predicting consistency, in accordance with [10]. Structured patch extraction outperforms random, multi-scale patch-extraction, especially with constrained computational resources. The feature map obtained from truncated ResNet-18 is of size 64×56×56. We divide this feature-map into four 64×28×28 patches and then take a fifth patch from center of the spatial grid.

A consistent patch-pair consists of two patches from the same image and an inconsistent patch-pair has two patches from different images. These pairs are made only from real images because we don't have masks for fake images indicating where consistent and inconsistent patches are located.

3.2.3 Formulating Consistency Objective through Contrastive Loss

Schroff *et al*. [15] describe contrastive training using image triplets (x^a, x^p, x^n). x^a is an anchor patch from the given image, x^p is a positive patch from the same image and x^n is a negative patch from a different image. Two pairs are formed: a positive pair (x^a, x^p) and a negative pair (x^a, x^n).

It is important to note that the above triplets have patches extracted only from real images, therefore, the network might simply learn visual similarities such as colour histograms [11] without performing deeper investigation of whether a patch came from real or fake image. To overcome this, we extract a fixed patch-pair from both real and fake images and add a cross-entropy loss. These patches are taken from the top-left and bottom-right 64×40×40 corners of the feature-map of an image with an overlap of 64×28×28 pixels. Pairs from a real and fake images are labeled consistent and inconsistent, respectively. Larger patches with overlapping regions are used to cover potentially manipulated parts of an image.

We want to know *how* consistent or inconsistent the patches are, so cross-entropy loss seems to be a good choice. Recall that z is a 2-element output vector produced by passing the absolute distance $d(x_1, x_2)$ (1) through the final linear layer of our network. The Siamese network is trained such that when the first element of z is greater, the patches are consistent. The opposite is true when the second element is greater. We take softmax of z and compute consistent loss (2) and inconsistent loss (3). The goal is to make $Softmax(z) \approx \hat{y}$ where $\hat{y}$ represents the true label.

$$L_{pos} = -z[0] + log(e^{z[0]} + e^{z[1]}) \tag{2}$$

$$L_{neg} = -z[1] + log(e^{z[0]} + e^{z[1]}) \tag{3}$$

Thus, for a pair of patches, we define the cross-entropy loss L_1 as:

$$L_1 = \begin{cases} L_{pos} & \text{for consistent pair } (x^a, x^p) \\ L_{neg} & \text{for inconsistent pair } (x^a, x^n) \end{cases} \tag{4}$$

As shown in **Figure 2**, we combine (2) and (3) to produce a single loss function (5):

$$L_2 = L_{pos} + L_{neg} \tag{5}$$

Additional binary cross entropy loss is calculated for the patch-pairs extracted from *either* real or fake images. To enforce the network to learn real vs. fake detection more than just the patch-similarity, we multiply this last loss by a factor of 2.

$$L_{comb} = L_2 + 2L_1 \tag{6}$$

3.3. Inference using Models 3.2 & 3.3

Once both models are trained independently, we integrate them to jointly make a decision about manipulation in an image. To integrate the Consistency Estimator during testing, we extracted five 64×28×28 patches from a given test image similar to training and generate exhaustive pairs using all patch combinations. Each pair is passed through the Consistency Estimator and those outputs are used to form an *Inconsistency Matrix* M. $M[i, j]$ indicates the inconsistency score between $patch_i$ and $patch_j$ of a single image. For a given M, all values above a certain threshold

m_{thresh} are summed together to give the matrix an overall inconsistency score M_{score} (7). A second threshold, s_{thresh}, is used such that if $M_{score} > s_{thresh}$, the image is classified as inconsistent (a label of 1). Otherwise, it is classified as self-consistent (a label of 0). We call this classification E_{pred} (8).

$$M_{score} = \sum_{i=0,j=0}^{4,4} (M_{ij} > m_{thresh}) \tag{7}$$

$$E_{pred} = \begin{cases} 1 & \text{if } M_{score} > s_{thresh} \\ 0 & otherwise \end{cases} \tag{8}$$

Sigmoid at the end of Binary Classifier produces a score between 0 and 1. We use this value $0 \leq B_{pred} \leq 1$ as a confidence score of the model in its prediction of manipulated images. We apply a weighted sum to B_{pred} and E_{pred} and set a threshold, c_{thresh}, for the final prediction of real (class 0) or fake (class 1) (9).

$$class = \begin{cases} 1 & \text{if } \frac{2}{3}B_{pred} + \frac{1}{3}E_{pred} > c_{thresh} \\ 0 & otherwise \end{cases} \tag{9}$$

4. Experiments

4.1. Dataset and Augmentation

Our dataset consists of a balanced set of real and fake images from three separate sources: Professor David Fouhey's dataset[1] (1,591 real and 2,000 fake images), the Flickr30k dataset (2,716 real images) [19], and the PS-Battles dataset (1,593 real and 3,600 fake images) [8]. This forms a well-balanced dataset containing 5,900 real images and 5,900 fake images. All images are resized to 3×224×224 dimensions and normalized using mean and standard deviation values obtained from 14 million images of the ImageNet. Several image-augmentation techniques are applied in order to generalize the network predictions to a broader range of images. We found that the techniques such as Gaussian blurring, random rotation, horizontal flipping, arbitrary cropping and slight zooming improve the validation and testing accuracy.

4.2. Evaluation

During training we plot the training loss and validation accuracy. Ideally, the training loss should keep on decreasing and validation accuracy should keep on increasing. In a practical scenario we observe spikes in both these plots due to momentum added by optimizers such as Adam and sometimes due to nonconvex loss constraints. Nevertheless, we continue our training until both training loss and validation accuracy curves flatten. At this point, we apply

[1]We thank Prof. Fouhey for this dataset and Prof. Owens for making it available.

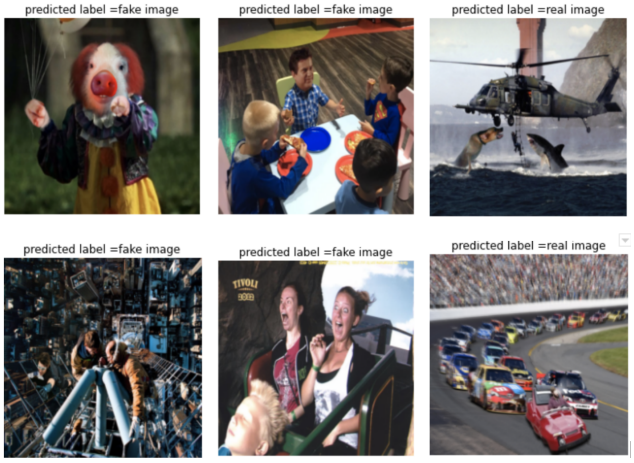

Figure 3: Successful predictions (columns 1 & 2) and wrong predictions (column 3) for fake images.

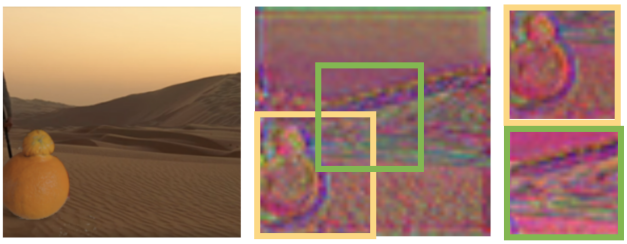

Figure 4: Fake image (left), patch-pair extraction from respective feature map (center), a consistent pair (right).

learning rate decay and observe further reduction in loss and increase in validation accuracy. We add regularization to prevent the network from overfitting on the training set and observe consistently good validation & test results.

Accuracy can concisely describe a model's performance in case of a balanced, shuffled dataset as here. It is the simplest and yet quite intuitive performance metric calculated as the ratio of true predictions to the total predictions. Let's use the following terminology: TP for True Positives, TN for True Negatives, FP for False Positives, FN for False Negatives. The accuracy is then calculated by (10). **Table 3** and **Table 4** concisely summarize performance of both the networks.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

4.3. Results

4.3.1 Binary Classifier Results

Table 4 summarizes results from our Binary Classifier. Both validation and testing accuracy increase as we move from 1. keeping all layers of ResNet-18 fixed (with pre-trained weights) to 2. fine-tuning the last few layers to 3. fine-tuning all layers. Since our learning objective is different from ImageNet classification, it is expected to see this

Method	Val	Test
Single Network, stacked patches	53.3%	50.9%
Siamese Network with triplets	84.1%	80.0%
Siamese Network with triplets + additional real vs. fake training	85.2%	82.0%

Table 3: Accuracy for Image Self-consistency Estimator.

Method	Val	Test
Classifier with Fixed ResNet	68.4%	61.4%
Tuning last few blocks of ResNet	69.6%	62.7%
Tuning entire Classifier	71.8%	64.2%
Tuning with Augmented Dataset	72.6%	64.9%
Integrating Siamese Estimator	73.8%	66.0%

Table 4: Real vs. fake prediction accuracy for Binary Classifier. Integration of both models is shown in the last row.

improvement in accuracy by tuning all layers of the network.

4.3.2 Image Self-consistency Estimator

The first method of evaluating the Consistency Estimator is to concatenate two image patches across their channel dimension and pass that tensor to a network at once (**Table 3**). This approach slows down the training as the model struggles to distinguish the difference between the first and second patches embedded within a single tensor. The second method deploys a Siamese network and passes two patches separately through the network as explained in section **3.2.3**. The network excels at predicting consistent and inconsistent pairs but produces slight improvement to accuracy when deployed to determine real vs. fake images. As cautioned by Huh *et al*. [10], the contrastive network easily picks up on simpler cues such as colour histograms, which is not useful when trying to determine self-consistency within an image. The third method enforces the model to simultaneously distinguish the difference between real and fake images during training, through combined loss L_{comb} in (6).

4.3.3 Combining the Models

Tuning the thresholds of m_{thresh}, s_{thresh} & c_{thresh} in Equations (7), (8), and (9) requires a very methodical approach. We start with tuning m_{thresh} and plotting the resulting M_{scores} of each image versus true prediction. In this way, we were able to find that $m_{thresh} = 15$ best separates the images. Once they are maximally separated, determining s_{thresh} is relatively simpler because we can visually inspect the graph for a best guess and then try different values for maximum prediction accuracy. This yields the best estimate of $s_{thresh} = 120$ and $c_{thresh} = 0.42$.

We use a weighted average of predictions made by both models as described by (9). The binary classifier makes the primary decision and the consistency estimator improves that prediction by a slight amount, as observed in **Table 4**.

4.4. Further Work and Potential Extensions

This well-structured, well-designed model allowed us to perform multiple experiments with the learning objective, network architecture and test-time combination. There are several open avenues of research that can potentially result in more efficient and accurate predictions by the model. We have used images from multiple datasets that offer wide variation in the objects being captured, mean and variance of colours, lighting and shadowing conditions and image manipulation techniques. Domain adaptation techniques can be implemented to get an adaptive network that can perform equally well on different datasets. A greater number of images with enhanced computational resources can certainly help the model to generalize to a broader range of image manipulations.

Our two models are trained independently and combined directly at the inference stage. We propose that fine-tuning the combined model during training would generate better results. The patch-consistency estimator is a Siamese network being trained using contrastive loss. Such networks learn to identify similarities between a pair of inputs very quickly, therefore it is important to find and select hard triplets during training for which the L_{pos} is greater than L_{neg} and the network is still forced to learn detailed features of the input patches, before declaring them consistent or inconsistent.

It is important to note that this network has very simple architecture while offering great potential for accurate prediction in almost real time. Complete inference including fresh patch-extraction and prediction from both the models does not take more than a few milliseconds for each image. It is therefore encouraging to work in above directions to leverage its full potential.

5. Conclusion

Two different network architectures have been proposed and successfully implemented for efficient detection of manipulated images. We have achieved 73.8% validation accuracy and 66.0% test accuracy over a large image dataset consisting of a wide variety of image features and image manipulation techniques (copy-&-move, object duplication, splice insertion, image composition). Leveraging its modular design, we have conducted several experiments on the network for training, testing and integration of both models and proposed a few solutions that can lead to significant improvement in accuracy while performing real-time inference.

References

[1] Maikol Castro, Dora M Ballesteros, and Diego Renza. A dataset of 1050-tampered color and grayscale images (CG-1050). *Data in brief*, 28:104864, 2020.

[2] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

[3] Davide Cozzolino, Justus Thies, Andreas Rössler, Christian Riess, Matthias Nießner, and Luisa Verdoliva. Forensictransfer: Weakly-supervised domain adaptation for forgery detection. *arXiv preprint arXiv:1812.02510*, 2018.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. IEEE, 2009.

[5] Aditi Gupta, Hemank Lamba, Ponnurangam Kumaraguru, and Anupam Joshi. Faking sandy: characterizing and identifying fake images on twitter during hurricane sandy. In *Proceedings of the 22nd international conference on World Wide Web*, pages 729–736, 2013.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] Silvan Heller, Luca Rossetto, and Heiko Schuldt. The PS-Battles Dataset – an Image Collection for Image Manipulation Detection. *CoRR*, abs/1804.04866, 2018.

[9] Austin Hounsel, Jordan Holland, Ben Kaiser, Kevin Borgolte, Nick Feamster, and Jonathan Mayer. Supporting early and scalable discovery of disinformation websites. *arXiv preprint arXiv:2003.07684*, 2020.

[10] Minyoung Huh, Andrew Liu, Andrew Owens, and Alexei A Efros. Fighting fake news: Image splice detection via learned self-consistency. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 101–117, 2018.

[11] Phillip Isola, Daniel Zoran, Dilip Krishnan, and Edward H Adelson. Learning visual groups from co-occurrences in space and time. *arXiv preprint arXiv:1511.06811*, 2015.

[12] Srijan Kumar and Neil Shah. False information on web and social media: A survey. *arXiv preprint arXiv:1804.08559*, 2018.

[13] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the annual meeting of the cognitive science society*, volume 33, 2011.

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

[15] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[16] Chong Wang, Xue Zhang, and Xipeng Lan. How to train triplet networks with 100k identities? In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 1907–1915, 2017.

[17] Sheng-Yu Wang, Oliver Wang, Andrew Owens, Richard Zhang, and Alexei A Efros. Detecting photoshopped faces by scripting photoshop. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10072–10081, 2019.

[18] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. CNN-generated images are surprisingly easy to spot... for now. *arXiv preprint arXiv:1912.11035*, 2019.

[19] Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014.