Deep Learning-based approach for Image Forgery Detection

Thuy Nguyen-Chinh, Thien Do-Tieu, Qui Nguyen-Van, Thuong Le-Tien

***Abstract*—As there are a huge range of powerful tools to edit images now, the need for verifying the authentication of images is more urgent than ever. While forgery methods are increasingly more subtle that even human vision seems hard to recognize these manipulations, conventional algorithms, which try to detect tampering traces, often pre-define assumptions that limit the scope of problem. Therefore, such methods are unable to encounter forgery methods in general applications. In this paper, we propose a framework that utilizes Deep Learning techniques to detect tampered images. Concretely, the MobileNetV2 network in [20] is modified so that it can be consistent to the task of image forgery detection. We argue that by remaining spatial dimension of early layers, the model is likely to learn rich features in these layers, and then following layers are to abstract these rich features for making a decision whether an image is forged. Besides, we also conduct a comprehensive experiment to prove those arguments. Experimental results show that the architecture-modified network achieves a remarkable accuracy of 95.15%, which surpasses others relying on the original architecture by a large margin up to 12.09%. Code and trained models are available at *https://github.com/AntiAegis/Image-Forgery-using-Deep-Learning*.**

***Index Terms*—Image Forensics, Image Forgery Detection, Deep Learning, Convolutional Neural Network**

I. Introduction

In the era of digital technology, the amount of multimedia grows considerably. In particular, image is the most popular kind of use due to its visibility. To serve purpose of human, there are a number of editor applications, which are used to modify content of images in order to match users' interest. Nonetheless, besides positive aspects, manipulating detail of images may lead to some negative effects because the edited image no longer reflects the real content, which is required in some serious fields like court, legislation, intelligence, and publication. Therefore, Image Forgery Detection is to detect manipulated images so as to prevent harmful impacts coming when such images used naively.

Typically, there are two kinds of forgery methods, namely copy-move and splicing. In the former method, the whole process is done in a unique image. Concretely, a region inside the image is selected and pasted into a different position of the same image. Meanwhile, with the latter method of manipulation, an object is copied from an image (source) and then pasted into a different image (destination). Besides, there may be some transformations (scaling, rotation, flip, and distortion) applied to the object before pasting it into the image. This work is to change the original object in both shape and size so that it is tough to trace back the source of editing. Moreover, to increase the nature of the forged image, people also perform a post processing into the image. For example, the boundary of the object is smoothed by using median filters, which try to erase the discontinuity caused by the editing.

A. Copy-move forgery

To detect the copy-move method of manipulation, researchers [1]–[3] normally exploit reliable matching techniques, for instance, SIFT [4] and SURF [5]. The reason for this adaptation is the ability of transition-, scale-, and rotation- invariance of these matching methods. Therefore, although the forged objects are different from the origins, they are still detected. However, tiny objects or objects that contain less pattern structure detail, these kinds of techniques do not perform as accurate as expected. This can be explained by a fact that SIFT and SURF extract features based on pattern information of objects in the image, before carrying out a matching operation. Thus, the matching stage is likely to miss duplicates that contain less pattern structure detail.

Besides, there is a branch of approach that divides an image into overlapping blocks, then a transform (Discrete Cosine Transform (DCT) [6]–[8], Discrete Wavelet Transform (DWT) [9], and Zernike moment [10]) is applied to these blocks to extract relevant features, which are inputs of a matching stage. More clearly, in the matching stage, feature vectors are compared as different pairs. A pair is considered as matched if its similarity score is lower than a predefined threshold. The similarity is a metric that reveals how similar two feature vectors are represented in a space vector. Typically, the metric is the well-known Euclidean distance.

B. Splicing forgery

In real world, the splicing forgery is more common as well as complex than the copy-move method, because spliced objects come from different sources. In [11], Zhang *et al.* proposed a classification model to detect such spliced objects. For more clear, moment-based feature extracted using DWT and DFT, and Image Quality Metrics were used as a feature of size 208 to train a fully connected neural network. Then, the neural network was to classify patches of a given image into two categories, namely positive and negative. However, in [12], authors introduced an application of saliency detection

Thuy Nguyen-Chinh, Thien Do-Tieu, and Qui Nguyen-Van are senior students in the Department of Electrical and Electronics Engineering, University of Technology, National University of Ho Chi Minh city, Vietnam (e-mail: {thuy.ng.ch, dotieuthien9997, nvqui97}@gmail.com).

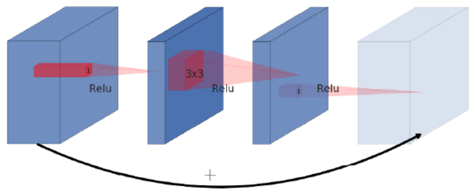
Thuong Le-Tien is with the Department of Electrical and Electronics Engineering, University of Technology, National University of Ho Chi Minh city, Vietnam (e-mail: thuongle@hcmut.edu.vn).

as a support for image splicing detection. They treated salient objects as unmodified ones, so by extracting a saliency map, they could figure out tampered regions. In a different branch, Pan *et al.* in [13] exploited a framework that made use of noise model to detect forged regions. Concretely, the input image was converted to the DCT domain. After that, a noise model was constructed to estimate noise variance in non-overlapping patches of the image. Finally, a clustering stage using K-mean algorithm was designed to discriminate tampered and original patches. Furthermore, Fridrich and Kodovský in [14] continued exploring the use of noise in steganalysis of digital images. They described a novel method to detect steganography in digital images. In which, noise residuals of the input image is computed by using linear and nonlinear high-pass filter. Then, a numerous number of models were built by joint distributions of these noise residuals. These models were aggregated to form a rich model for detecting steganography.

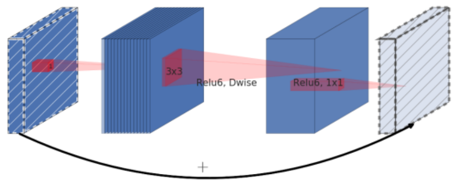
C. *Deep-learning-based methods*

Because conventional methods often assume some constraints limiting the problem scope that they are solving, such these methods can not be applied in general situations. In [15], Goh and Thing designed a hybrid framework to detect tampered images. Concretely, a list of methods was considered to extract feature vectors, including DCT, Markov Random Process, DWT, Edge Statistics, and Correlation filters. All of these features were aggregated and an evolutionary Memetic algorithm was exploited to select an optimal set of features. Furthermore, the authors continued applying the Genetic algorithm to optimize an ensemble of classifiers to make a final decision. Although the result was promising, the method still relied on hand-crafted features and the ensemble of multiple classifiers led to complex computation.

Due to the development of hardware (e.g., Graphic Processing Unit (GPU), Tensor Processing Unit (TPU)) as well as breakthroughs in Computer Vision, there are increasing works applying Deep Learning to detect forged images. Zhang *et al.* [16] used the Daubechies Wavelet Transform to extract feature vector of size 450 for overlapping blocks inside an image. Afterward, these feature vectors were fed into a Fully Connected Neural Network in order to classify which block was tampered. Also, Rao *et al.* [17] proposed a model containing a Convolutional Neural Network (CNN) with 10 layers, a SVM classifier, and a feature fusion to detect forgery in image level. They compared the efficiency between Xavier and SRM [11] initialization methods and found that the latter helped their model trained faster and better than the former. Instead of using CNN, authors in [18] exploited resampling feature and deep learning. In which, they proposed two techniques. In the first method, resampling features of overlapping patches are extracted, and then these features are transformed using Radon transformation. After that, a deep classifier accompanied with a Gaussian model create a heatmap of the image. Finally, a Random Walker algorithm uses the heatmap to localize forged region. In the second method, resampling features are still utilized, yet classified by a Long Short-Term Memory (LSTM) network to localize forged region.



(a) Residual bottleneck



(b) Linear bottleneck

Fig. 1: While the Residual bottleneck (a) squeezes at the middle for reducing the computational cost, the Linear bottleneck (b) expands its neck to capture features elaborately. Each of linear bottleneck is indicated by factors, namely expansion t, channels c, and stride s. Picture is taken from [20].

II. Proposed method

In this paper, we introduce a Deep Learning technique for detecting tampered images in both copy-move and splicing cases. First, the input image is divided into overlapping patches. These patches are extracted features (logits) by using MobilenetV2 [20]. Finally, a post processing stage is to assemble extracted features to conclude whether the input image is forged. The reason for using MobilenetV2 is to inherit its light-weight computation and fast speed. Nonetheless, instead of direct use, we argue that down-sampling by setting stride of 2 in early layers may lead to loose detail information of patches. Therefore, we modify the MobilenetV2 architecture to remain the spatial dimension in low-level layers. As a result, our model obtains a reasonable accuracy of 95.15% on the CASIA2 dataset [19].

Our work is organized as follows. In section III, the methodology is described, followed by experiments in section IV. Finally, we conclude our work in the last section V.

III. Methodology

A. *Background*

In this section, we review the network MobileNet-V2 [20], which is the background of our model. The reason that we choose this architecture is because of its light-weight configuration for implementing a robust deep convolutional neural network on limited hard devices. Both the number of parameters and add-multiply calculations are relatively small.

Linear Bottleneck The network is built up on basis blocks, called linear bottleneck, which is illustrated in Fig. 1. The linear bottleneck is inspired by the residual bottleneck used in ResNet [21]. In the residual bottleneck, there are four convolutional layers, in which, the block is squeezed at the middle by using 1x1 convolutions. Because between the two intermediate layers, a 3x3 convolution is applied for capturing large-view features, the computational cost is expensive. Therefore, by reducing the number of intermediate channels, the complexity in computation will be released. In contrast, linear bottleneck expands at its neck so as to capture features elaborately. However, to deal with the number of calculations, the depthwise convolutions (see below) to sympathize computations without any sacrifice in effectiveness.

Depthwise Convolutions Conventional convolutions are dense because filters are shared for both input and output channels. For example, sizes of input and output are $H\text{x}W\text{x}C_i$ and $H\text{x}W\text{x}C_o$, respectively, and the size of filter would be $K\text{x}K\text{x}C_i\text{x}C_o$ ($H\text{x}W$ is spatial dimension, C corresponds number of channels, and K is kernel size). To apply the filter on the input, it costs:

$$H.W.K.K.C_i.C_o \tag{1}$$

Nonetheless, the depthwise convolutions are designed to reduce the number of calculations with an equivalent result. Concretely, there are two stages, namely separate convolution and 1x1 convolution. First, each channel of the input is convolved by its individual filter of size $K\text{x}K$. Then, features are aggregated and passed through a 1x1 convolution of size $1\text{x}1\text{x}C_i\text{x}C_o$. Therefore, the total number of multiply operations is:

$$H.W.C_i.(K^2 + C_o) \tag{2}$$

which is smaller than its conventional version by a factor:

$$\frac{K^2.C_o}{K^2 + C_o} \tag{3}$$

B. MobileNetV2 modification

In [20], authors trained the network on the ImageNet dataset, in which, images were resized to 224 before passing through the network. The original architecture of the MobileNetV2 is shown in Table I. Particularly, factor s denotes the stride of the first convolutional layer in a bottleneck. The stride is to reduce the spatial dimension of feature tensors that helps the network synthesize higher features from previous layers. In the original version, the output tensor (right before the Global Average Pooling) has size smaller than the input image 32 times. However, in our dataset, patches, which are extracted from images, have size of 64 (in section III-C), the original network is no longer appropriate because of the significant subsampling of $1/32$. For more clear, low layers (layers near the input) are responsible for capturing low-level features like edges, corners, whilst, top layers (layers near the output) abstract high-level features like objects. Therefore, early layers need to be large enough so that low-level detail is clear and high-resolution for convolutional filters to capture.

Based on aforementioned arguments, we modify the original MobileNetV2 architecture in a way that dimension of early layers is relatively high-resolution. Thus, we convert stride factors of the $conv2d$ 3×3 layer and the second $bottleneck$ from 2 to 1 in order to maintain the spatial dimension of early layers. As a result, the output tensor is smaller than the input image by 8 times of size. In addition, we also replace the $conv2d$ 1×1 layer by a fully connected layer ($linear$ 1280×2) containing both weights and biases instead of only weights in the original form.

TABLE I. ORIGINAL MOBILENET-V2 ARCHITECTURE

Input	Layer	t	c	n	s
224^2x3	conv2d 3x3	-	32	1	2
112^2x32	bottleneck	1	16	1	1
112^2x16	bottleneck	6	24	2	2
56^2x24	bottleneck	6	32	3	2
28^2x32	bottleneck	6	64	4	2
14^2x64	bottleneck	6	96	3	1
14^2x96	bottleneck	6	160	3	2
7^2x160	bottleneck	6	320	1	1
7^2x320	conv2d 1x1	-	1280	1	1
7^2x1280	avgpool	-	-	1	-
1x1280	conv2d 1x1	-	-	-	-

TABLE II. MODIFIED MOBILENET-V2 ARCHITECTURE

Input	Layer	t	c	n	s
64^2x3	conv2d 3x3	-	32	1	**1**
64^2x32	bottleneck	1	16	1	1
64^2x16	bottleneck	6	24	2	**1**
64^2x24	bottleneck	6	32	3	2
32^2x32	bottleneck	6	64	4	2
16^2x64	bottleneck	6	96	3	1
64^2x96	bottleneck	6	160	3	2
8^2x160	bottleneck	6	320	1	1
8^2x320	conv2d 1x1	-	1280	1	1
8^2x1280	avgpool	-	-	1	-
1x1280	**linear 1280x2**	-	-	-	-

C. The whole framework

To classify if a give image is not authentic, a sliding window of size 64 is used to take out overlapping (stride of 32) from the image. After being converted to the YCrCb color channel, such patches are passed through the modified-MobileNetV2 network to score their authentication. Afterward, a post-processing stage is to take neighborhood information of scores into account to decide labels corresponding to patches. Assume that the patch p_0, which is scored $s(p_0)$ by the network, has k neighbors, denoted as $p_i (i = \overline{1, k})$. So, its reliability can be calculated as follows:

$$Reliability = \frac{1}{k+1} \sum_{i=0}^{k} s(p_i) \tag{4}$$

If the reliability exceeds a threshold $\alpha (0 \leq \alpha \leq 1)$, label for the patch p_0 is positive, unless, the label is negative. The whole framework is depicted in Fig. 2.

IV. Experiment

A. Metrics

To evaluate the model, some typical metrics are used. The classification output belongs to 4 possible cases, including True Positive (TP), True Negative (TN), False Positive (FP),

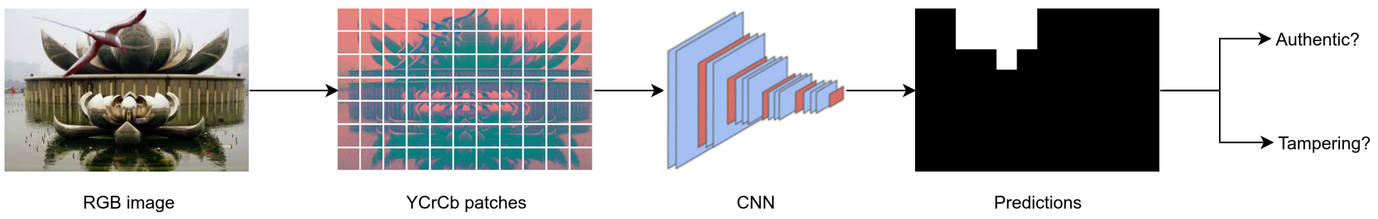


Fig. 2: The whole framework. An RGB image, firstly, is divided into overlapping patches. Then, RGB patches are converted to the YCrCb color channel, before being scored by a network. Lastly, a post-processing stage is designed to refine predictions of the network and make a final conclusion on the authentication of the image.

and False Negative (FN). Metrics are computed from the 4 factors as follows.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

$$Fscore = \frac{2 * Precision * Recall}{Precision + Recall} \tag{8}$$

Among these metrics, Accuracy reveals how accurate the model performs. This is a primary metric that people usually use for evaluating the general performance of a model. However, in the case of anomaly detection or imbalanced-class problem. Accuracy is not enough to conclude the strength of model because class which has the number of samples is likely to dominate class which has fewer one. Therefore, Precision and Recall are additionally utilized in order to overcome the shortcoming of Accuracy in these kinds of problem. In particular, Precision depicts the percentage of exactly positive samples over samples predicted as positive, meanwhile, Recall is to indicate how many positive samples predicted as positive. Depend on each specific application, there is a trade-off between Precision and Recall. To represent for this trade-off, people also exploit Fscore as a single value instead of two metrics. All of the four metrics, the higher these values, the better model performance. Also, note that such metrics are in the range from 0 to 1.

B. Data preparation

To train and evaluate our model, we use dataset CASIA2 in [19]. According to the authors, this dataset contains more than 12000 images, in which there are about 7500 authentic images and about 5000 tampered images. Particularly, tampered images are manipulated from authentic images by different kinds of methods (e.g., copy-move, splicing, affine transform, blurring tampering edges). Although the dataset does not provide ground truth for indicating which pixels are forged in tampered images, we can infer this by subtracting a tampered image and its corresponding authentic one based on the information of tampering method used to create the tampered image within its file name. Then the subtracting result is filtered and thresholded to get the forgery map. Fig. 3 shows some forgery maps obtained from their authentic and tampered images. Henceforth, we call these forgery maps as ground truth.

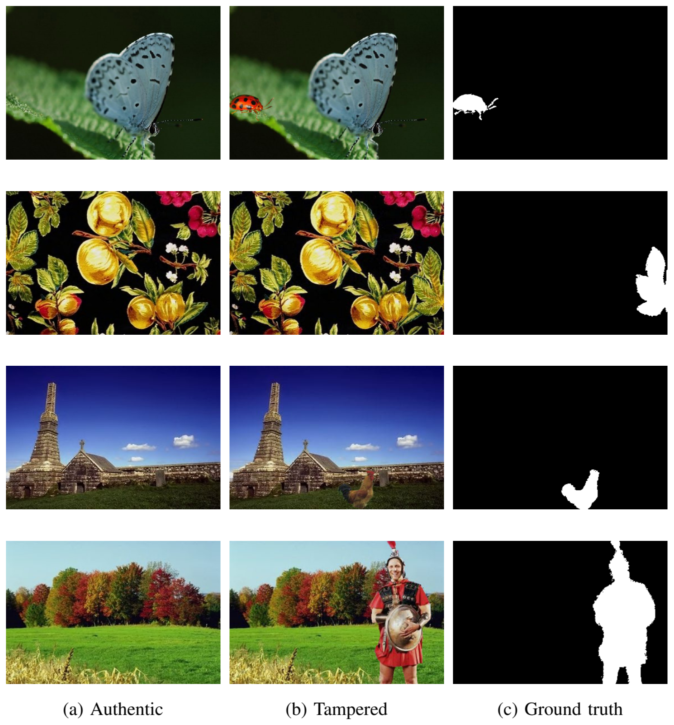


Fig. 3: Columns from left to right are authentic, tampered, and ground truth images. From information within file names of tampered images, we can trace their authentic ones. By performing subtraction, filtering, and thresholding, we can obtain ground truths. In ground truths, the white regions indicate forged objects, while the black regions are original.

After inferring the ground truth, we prepare data for training as well as testing. As input of our model has size 64x64, we generate a set of image patches of size 64 that are cropped from images. With authentic images, patches are randomly taken inside images regardless where to crop. Nevertheless, tampered patches are cropped at specific positions inside images. Concretely, we use a window that slides over the ground truth. The sliding is carried out with kernel size of 64 and stride of 8. If a window has the percentage of forged region lies in a range from 0.4 to 0.6, that window is used for cropping tampered patches.

Finally, we attain a dataset with 307914 authentic and 331365 tampered patches, belonging to 6238 authentic and 3888 tampered images. This dataset is used for training and validating the model, particularly, the training set occupies

TABLE III. DATASET SUMMARY

Dataset	Authentic	Tampered
Training	277051 patches	298155 patches
Validating	30863 patches	33210 patches
Testing	1253 images	766 images

90%, leaving 10% for the validating set. Besides, the rest with 1253 authentic and 766 tampered images is for testing. Table III summaries the dataset.

C. Training

In the training phase, the model learns to discriminate two distributions of the positive and negative classes by using prepared dataset in section IV-B. There are totally 277051 negative and 298155 positive patches for training the model. All of those patches are of size 64. To avoid a possibility that the model is significantly energetic enough to overfit to the training set, an augmentation strategy is applied so that training images are slightly transformed over epochs. Specifically, random flip in both horizontal and vertical sides, and rotation in range from -30 to 30 degree are used to linearly transform images. Then, these images are converted to the YCrCb color channel, before being scaled to unit range. Subsequently, a normalization is carried out:

$$I_{norm}^{(c)} = \frac{I^{(c)} - m^{(c)}}{d^{(c)}} \tag{9}$$

where I is the image, $c \in \{Y, Cr, Cb\}$ denotes the color channel, $m^{(c)} \in [0.485, 0.456, 0.406]$ and $d^{(c)} \in [0.229, 0.224, 0.225]$ are mean and standard deviation values, in turn, corresponding to color channels. The two vectors are inferred from the training dataset. Moreover, to reflect the ability to generalize of the model, a more validation dataset is prepared, containing 30863 authentic and 33210 tampered samples. Images in this dataset are not to train parameters of the model, instead they are used for determining hyper-parameters such as number of epochs, learning rate, and weight decay. These images are not transformed by aforementioned augmentation methods.

To train the network, we use the SGD optimizer, accompanying with nesterov and momentum of 0.9. The learning rate is set at $1e^{-3}$, while the weight decay is deactivated at the beginning. This configuration is kept until the 15th epoch. Afterwards, in the 16th and 17th epochs, learning rate is reduced to $1e^{-4}$ and the weight decay is activated with value of $1e^{-5}$. In addition, CrossEntropy loss is used along side with SDG for optimizing parameters of the network:

$$L = -\frac{1}{C} \sum_{i=0}^{C-1} \Big(y_i ln(\hat{y_i}) + (1 - y_i) ln(1 - \hat{y_i}) \Big) \tag{10}$$

where C is the number of classes, y_i and $\hat{y}_i$ are the ground truth and prediction of the corresponding class. Before training the model, parameters in the network are initialized appropriate values so that the training phase can converge quickly. Concretely, all biases are set to zero. Meanwhile, weights w are assigned as normal distributions $\mathcal{N}$, excepting for batch normalization layers:

$$w_{conv} \sim \mathcal{N}\left(0, \sqrt{\frac{2}{n}}\right) \tag{11}$$

$$w_{linear} \sim \mathcal{N}(0, 0.01) \tag{12}$$

$$w_{batchnorm} = 1 \tag{13}$$

where $n = K^2 * C_i$, in which, K is the kernel size, and C_i is the number of input channels.

Fig. 4 illustrates the training process with loss and accuracy of the training and validating samples. The model was trained with 17 epochs, however, it underwent a fast converge within the first five epochs. This proves that the initialization is relevant to bootstrap the learning progress faster. Here we used a PC with Intel Corei7@2.8GHz CPU, 8GiB RAM, and NVIDIA 1050 GPU to train the network. It took around 100 minutes per an epoch with batch size of 8 patches.

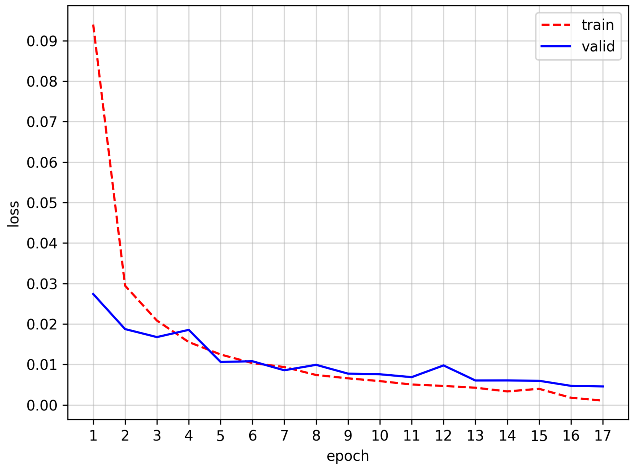
D. Testing

In this section, we evaluate the trained model on a set of 1253 authentic and 766 tampered images, which is equal to a one-sixth of the whole images in the CASIA2 database. To indicate the authentication of an image, the algorithm fist uses a window of size 64, striding over the image with stride of 32 to take out patches of size 64. After that, these patches are fed into the trained network where they are predicted their scores. Finally, the post-processing stage is to refine scores of patches by utilizing neighborhood information. If all patches in the image are negative, the image will be indicated as origin, while if there exists at least one positive patch, the image will be marked as tampered. Table IV shows the testing result where the corresponding model is denoted as MBN2-mod-YCrCb, which achieves a remarkable accuracy of 95.15%.

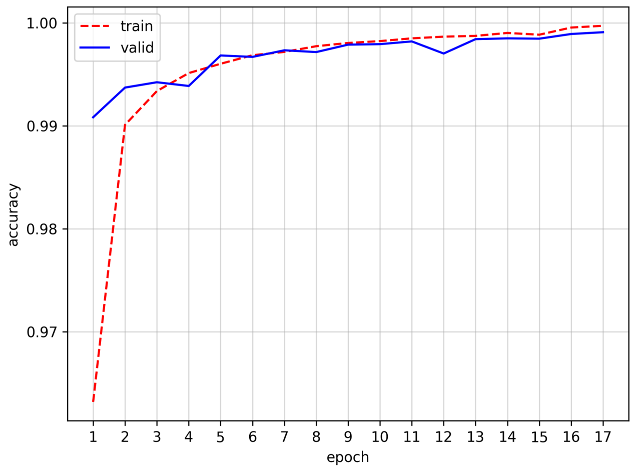
E. Comparison

In this section, we conduct a comprehensive evaluation on model configurations to show which factor improve the final performance of the model. To figure out this, we define six configurations accompanied with the MobileNetV2, denoted as MBN2, as the core. There are two color channels to be considered, namely RGB and YCrCb. The RGB color space is highly familiar to human's vision that almost images are represented in this format. Meanwhile, as mentioned in [22] and [23] that YCrCb is more sensitive to tampering operation than RGB so that we take into account to examine the possibility to boost the performance of model by using YCrCb color channel. Furthermore, three MobileNetV2 architectures are taken into account for comparing. The first architecture is MobileNetV2 trained from scratch, the second one is MobileNetV2 initialized with pre-trained weights from ImageNet, and the last one is modified MobileNetV2 trained from scratch.

The result of comparison between models is revealed in Table IV. As can be seen, models trained with RGB images



(a) Loss



(b) Accuracy

Fig. 4: Loss and accuracy during the training phase. Time for training an epoch is about 100 minutes on a PC with Intel Corei7@2.8GHz CPU, 8GiB RAM, and NVIDIA GTX 1050 GPU. Note that these metrics are calculated as patch level.

TABLE IV. PERFORMANCE COMPARISON

Model	Accuracy	Precision	Recall	F-score
MBN2-RGB	78.50%	88.51%	66.21%	75.75%
MBN2-YCrCb	80.14%	64.10%	79.58%	71.01%
MBN2-pre-RGB	74.69%	58.75%	69.77%	63.78
MBN2-pre-YCrCb	83.06%	73.50%	80.20%	76.70%
MBN2-mod-RGB	91.08%	96.48%	82.85%	89.14%
MBN2-mod-YCrCb	**95.15%**	**97.26%**	**90.63%**	**93.83%**

perform worse than ones trained with YCrCb images. This can be explained that in the RGB format, subtle tampering detail is carefully obscured when people manipulate images by software so that it is fairly harsh to recognize these operations in RGB channel. Nevertheless, in YCrCb, images are transformed to a new color space, which may expose more tampering information. Additionally, when converting an RGB image to the YCrCb, tampering edges are likely to be preserved, especially in chroma channels (e.g., Cr and Cb). This leads to the outperform of the YCrCb over the RGB color channel in this comparison.

TABLE V. COMPUTATION COMPARISON

Model	Number of parameters	Model size (MiB)	Inference time (ms)
MBN2-RGB	2,226,434	21.03	5.5
MBN2-YCrCb	2,226,434	21.03	5.5
MBN2-pre-RGB	2,226,434	21.03	5.5
MBN2-pre-YCrCb	2,226,434	21.03	5.5
MBN2-mod-RGB	2,286,674	118.25	6.1
MBN2-mod-YCrCb	2,286,674	118.25	6.1

Regarding architectures, the modified MobileNetV2 version is the best, followed by ImageNet-pretrained MobileNetV2, and the rest is MobileNetV2 trained from the scratch. Firstly, it is quite straightforward that the model trained from scratch underperforms the model initialized with ImageNet-pretrained weights. This is due to well-trained filters on the large-scale dataset of images so that these filters can adapt to our dataset effortlessly. Secondly, although on ImageNet dataset, the MobileNetV2 is trained with RGB images, convolutional filters are learned to capture features from low-level such as edges, corners, to high-level like objects. Hence, when finetuning it with YCrCb images, filters, which are self-energetic, tend to be updated an insignificant amount to be compatible to YCrCb format. As the result, ImageNet-pretrained MobileNetV2 on YCrCb is noticeably more robust than the one on RGB for nearly 10% in accuracy. Lastly, introduced modifications in the original version of MobileNetV2 is demonstrated as relevant as expected. By remaining the spatial dimension in early layers, the model can learn a rich source of features in these layers while subsequent layers are responsible for abstracting rich features learned from predecessors. This leads to the best performance of this kind of architecture among the others.

In regard to Table V, we additionally analyze the computation aspect of the models. There are three factors for comparing, including number of parameters, model size, and inference time. Firstly, number of parameters of the six models are quite similar, however, the two modified architectures have more parameters than the rest due to the replacement of the conv2d1x1 layer by the linear1280x2 layer. Secondly, because features in early layers are maintained their size, MBN2-mod-RGB and MBN2-mod-YCrCb take an amount of memory in GPU that is significantly greater than the others (more than 5 times). Lastly, when running inference (forward pass), all models take a small time around 5-6 ms. Note that the computation comparison is performed using an input of size 64 and batch size 1.

V. Conclusion

In this paper, we propose a model which can detect forged images in both copy-move and splicing forgery methods. Our model utilizes the fast and low-cost MobileNetV2 architecture to classify patches inside input images whether a patch is forged. However, instead of directly using the original architecture, we modify the network so that spatial resolution of early layers remained unchanged, which help the model learn rich features from the input, before abstracting these rich features in top layers. Furthermore, I also conduct a comprehensive experiment to compare model configurations to demonstrate the effectiveness of our modified version of the MobileNetV2. The final testing result show that the performance of the modified network is reasonable with accuracy of 95.15% on the CASIA2 dataset. In the future, we will consider Semantic Segmentation techniques to localize tampered regions inside the image.

Acknowledge

This work is funded by the University of Technology, National University of Ho Chi Minh city, Vietnam, under grant number "SVKSTN-2018-DDT04".

References

[1] X. Pan and S. Lyu, *"Region duplication detection using image feature matching"*, IEEE Transactions on Information Forensics and Security, vol. 5, no.4, ISSN: 1556-6013, pp. 857-867, 2010.

[2] I. Amerini, L. Ballan, R. Caldelli, A. Del Bimbo and G.Serra, *"A sift-based forensic method for copy–move attack detection and transformation recovery"*, IEEE Transactions on Information Forensics and Security, vol. 6, no. 3, ISSN: 1556-6013, pp. 1099-1110, 2011.

[3] P. Kakar, N. Sudha, *"Exposing postprocessed copy-paste forgeries through transform-invariant feature"*, IEEE Transactions on Information Forensics and Security, vol. 7, no. 3, ISSN: 1556-6013, pp. 1018-1028, June 2012.

[4] D.G. Lowe, *"Object recognition from local scale-invariant features"*, Proceedings of the 7th IEEE International Conference on Computer Vision, Kerkyra, Greece, Sept. 1999.

[5] H. Bay, A. Ess, T. Tuytelaars, L. V. Gool, *"Speeded-Up Robust Features (SURF)"*, Proceedings of the 9th European conference on Computer Vision, Graz, Austria, May 2006.

[6] Z. Lin, J. He, X. Tang, K. Tang, *"Fast, automatic and fine-grained tampered JPEG image detection via DCT coefficient analysis"*, Pattern Recognition, vol. 42, no. 11, ISSN: 0031-3203, pp. 2492-2501, January 2009.

[7] W. Wang, J. Dong, T. Tan, *"Exploring DCT coefficient quantization effects for local tampering detection"*, IEEE Transactions on Information Forensics and Security, vol. 9, no. 10, ISSN: 1556-6013, pp. 1653-1666, October 2014.

[8] L. Chen, T. Hsu, *"Detecting recompression of JPEG images via periodicity analysis of compression artifacts for tampering detection"*, IEEE Transactions on Information Forensics and Security, vol. 6, no. 2, ISSN: 1556-6013, pp. 396-406, June 2011.

[9] M. Bashar, K. Noda, N. Ohnishi, K. Mori, *"Exploring Duplicated Regions in Natural Images"*, IEEE Transactions on Image Processing, ISSN: 1057-7149, March 25, 2010.

[10] T.L. Tien, T.H. Ngoc, T.H. Kha, M. Luong, *"Zernike Moment Based Approach for Detecting Duplicated Image Regions by a Modified Method to Reduce Geometrical and Numerical Errors"*, Book Chapter as Lecture Note in Computer Science (LNCS) by Springer, SEPA-ICCSA 2015, Banff, Canada, June 2015.

[11] Z. Zhang, G. Wang, Y. Bian, Z. Yu, *"A novel model for splicing detection"*, 2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA), Changsha, China, September 2010.

[12] O. Muratov, D.T.D Nguyen, G. Boato, F.G.B.D. Natale, *"Saliency detection as a support for image forensics"*, 2012 5th International Symposium on Communications, Control and Signal Processing, Rome, Italy, June 2012.

[13] X. Pan, X. Zhang, S. Lyu, *"Exposing image forgery with blind noise estimation"*, MMSec 11 Proceedings of the thirteenth ACM multimedia workshop on Multimedia and security, New York, USA, September 2011.

[14] J. Fridrich, and J. Kodovský, *"Rich models for steganalysis of digital images"*, IEEE Transactions on Information Forensics and Security, vol. 7, no. 3, pp. 868-882, June 2012.

[15] J. Goh, V.L.L. Thing, *"A hybrid evolutionary algorithm for feature and ensemble selection in image tampering detection"*, International Journal of Electronic Security and Digital Forensics, ISSN: 1751-911X, vol. 7, no.1, pp. 76-104, March 2015.

[16] Y. Zhang, J. Goh, L. Win, V. Thing, *"Image Region Forgery Detection: A Deep Learning Approach"*, Proceedings of the Singapore Cyber-Security Conference, Singapore, 2016, ISBN: 978-1-61499-616-3.

[17] Rao Yuan, Ni Jiangqun, *"A deep learning approach to detection of splicing and copy-move forgeries in images"*, IEEE International Workshop on Information Forensics and Security (WIFS), Abu Dhabi-United Arab Emirates, 2016, ISBN: 978-1-5090-1139-1.

[18] J. Bunk, J.H. Bappy, T.M. Mohammed, L. Nataraj, A. Flenner, B.S. Manjunath, S. Chandrasekaran, A.K.R. Chowdhury, and L. Peterson, *"Detection and Localization of Image Forgeries using Resampling Features and Deep Learning"*, IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, USA, July 2017.

[19] J. Dong and W. Wang, *"Casia tampering detection dataset"*, 2011.

[20] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, *"MobileNetV2: Inverted Residuals and Linear Bottlenecks"*, https://arxiv.org/abs/1801.04381, January 2018.

[21] K. He, X. Zhang, S. Ren, J. Sun, *"Deep Residual Learning for Image Recognition"*, https://arxiv.org/abs/1512.03385, December 2015.

[22] W. Wang, J. Dong, T. Tan, *"Effective image splicing detection based on image chroma"*, 2009 16th IEEE International Conference on Image Processing (ICIP), Cairo, Egypt, November 2009.

[23] P. Sutthiwan, Y.-Q. Shi, J. Dong, T. Tan, T.-T. Ng, *"New developments in color image tampering detection"*, Proceedings of 2010 IEEE International Symposium on Circuits and Systems, Paris, France, June 2010.

Thuy Nguyen-Chinh was born in Dong Nai, Vietnam. At the present, he is a senior undergraduate student in the Honor class of Electronics and Telecommunications, Electrical and Electronics Engineering Department, HoChiMinh city University of Technology (HCMUT), Vietnam. His research interests relate to apply machine learning and deep learning techniques to solve problems in signal processing, computer vision, and embedded system, particularly in image forensics, biometric recognition, and autonomous robotics.

Thien Do-Tieu is a senior student in the Honor class of Electronics and Telecommunications, Electrical and Electronics Engineering Department, HoChiMinh city University of Technology (HCMUT), Vietnam. He does research on Machine Learning and Deep Learning in Computer Vision, especially Image Forensics, Face recognition and Autonomous cars.

Qui Nguyen-Van is a senior student at HoChiMinh city University of Technology (HCMUT), Vietnam. His major is Electronics and Telecommunications, Electrical and Electronics Engineering Department. He does research on Machine Learning and Deep Learning in Computer Vision, Image Processing.

Thuong Le-Tien (MIEEE-96) was born in Saigon, HoChiMinh City, Vietnam. He received the Bachelor and Master Degrees in Electronics-Engineering from HoChiMinh City Uni. of Technology (HCMUT), Vietnam, then the Ph.D. in Telecommunications from the Uni. of Tasmania, Australia. Since May 1981 he has been with the EEEng. Department at the HCMUT. He spent 3 years in the Federal Republic of Germany as a visiting scholar at the Ruhr Uni. from 1989-1992. He served as Deputy Department Head for many years and had been the Telecommunications Department Head from 1998 until 2002. He had also appointed for the second position as the Director of Center for Overseas Studies since 1998 up to May 2010. His areas of specialization include: Communication Systems, Signal Processing and Electronic Circuits. He has published more than 170 research articles and the teaching materials for university students related to Electronic Circuits 1 and 2, Digital Signal Processing and Wavelets, Antenna and Wave Propagation, Communication Systems. Currently he is a full professor at the HCMUT.