IMAGE FAKE DETECTION WITH ELA AND DEEP LEARNING

Agus Gunawan, Holy Lovenia, Adrian Hartarto Pramudita

Technical Information
School of Electrical and Informatics Engineering
Bandung Institute of Technology

ABSTRACT

Images are often manipulated with the intent and purpose to benefit one party. In fact, images are often considered as evidence of a fact or reality, therefore, fake news or any form of publication that uses images that have been manipulated in such a way has greater capability and potential to mislead. To detect such image forgery, large amounts of image data are needed, and a model that can process each of these images is required *pixels* in the picture. In addition, efficiency and flexibility in data training are also needed to support its use in everyday life. The concept of big data and deep learning is the right solution for this problem. Therefore, with architecture *Convolutional Neural Networks (*CNN) which utilizes *Error Level Analysis (* ELA), image forgery detection can reach 91.83% and convergence only with 9 *epoch.*

Keywords- *image forgery detection,* convolutional neural network, error level analysis, deep learning, big data

1. BACKGROUND

according to *EU High Level Expert Group (*2018), fake news is defined as disinformation, which is any form of inaccurate, false or misleading information that is presented, promoted or designed. Behind the fake news, there are several reasons for the publication. One of them is to get an economic advantage, whether it's through increasing the number of news clicks or making news that is not supposed to benefit either party [1].

In addition, fake news can also affect stock prices, which can benefit those who release the news. The other reason is to gain support or bring down the other party

social or political [2]. Based on

stats belonging Public Telematics Indonesia (MASTEL) in 2017, the types of fake news that are most often received are socio-political, SARA (ethnicity, religion, and race), health, food and beverage, financial fraud, and science and technology. As many as 84.5% of all respondents stated that they feel disturbed by the existence of fake news, and more than 70% agreed that fake news disturbs community harmony and hinders development.

Apart from writing, around 40% of respondents stated that the spread of fake news is also often accompanied by pictures. Images are used by humans to reproduce reality, and are often used as evidence of news, publications, or facts. Fake news that has a supporting image tends to be accepted and trusted by the public.

In general, humans are easier to remember the form of pictures than writing. according to *Social Science Research Network,* as many as 65% of people are people who like to learn through visuals. in science *marketing* and *visual,* It is mentioned that the image has a very big influence on an article. People are more likely to respond when there are pictures than just text. According to an infographic with the theme of the influence of images in the world of marketing, images can increase the number of respondents for an article by up to 94% [8]. Therefore, an image is a strong element in disseminating information.

To determine an image is genuine or fake, it is very difficult to see with the naked eye, special techniques and certain accuracy are needed in order to know for sure an image is an original image or has been modified. For ordinary people, this may be difficult to do. For this reason, image forgery detection technology needs to be developed, so that it can be used as a means to assist people in determining the authenticity of an image.

This technology requires a lot of image data, and each image has a lot *pixels* its constituents. With ordinary machine learning, this technology would be difficult to develop. So that, *big data* and *deep learning* is the right solution to solve this image forgery detection problem.

2. PURPOSE

Data mining in the form of image forgery detection has two main objectives as follows.

1. Propose a new method using *deep learning* to classify images as original images and images that have been modified with a simpler architecture, so that computational costs can be reduced

2. Involving the use of ELA in machine learning as an effort to increase efficiency

There are several impetus behind these two main goals. As is well known, there have been several past studies which also aimed to detect image falsification [10, 11]. However, most of these researches require considerable computational costs (as can be seen from the number of *epoch* and *layers* required), so that the flexibility of the proposed method is reduced and difficult to apply in everyday life due to computational costs. However, there is a need for image forgery detection methods to be able to adapt to the addition of original image data and modifications over time.

Therefore, in this paper, a method for detecting image forgery is proposed which is relatively more efficient and has an increase in scalability that is directly proportional to the increase in data.

3. BENEFITS

Data mining in the form of image forgery detection can be used for the following things.

1. Increase convenience in getting information that is in accordance with the facts
2. The public gets consideration in determining whether the image is real or fake

With a reference for the public to find out whether an image is genuine or not, of course it will reduce the anxiety that exists due to fake images.

4. LIMITATIONS

There are several limitations that apply to this image forgery detection data mining, namely the raw data must be an image with *lossy compression (*for example . jpg), is also not a *computer generated image* (CGI).

5. METHOD

There are two main methods used in this data mining, namely Error Level Analysis (ELA) and machine learning with deep learning techniques in the form of Convolutional Neural Network (CNN).

5.1. *Error Level Analysis (ELA)*

Error Level Analysis is one of the techniques used to detect image manipulation by re-saving the image at a certain quality level and calculating the ratio between the compression levels [4]. In general, this technique is performed on images that have a format *lossy (lossy compression).* The image type used in this data mining is JPEG. In JPEG images, compression is performed independently for every 8x8 pixels in the image. If an image is not manipulated,

level *error* every 8x8 pixels in the image must have the same [6].

5.2. *Convolutional Neural Network (CNN)*

CNN is type *network* based on *feedforward,* the flow of information is only one direction, namely from input to output. While there are several types of CNN architectures, in general, CNNs have several *convolutional layers* and *pool layer.* Then, followed by one or more *fully connected layer.* In image classification, the input to CNN is in the form of images, so that each *pixels* can be processed [5].

In brief, *convolutional layers* used as a feature extractor that learns the representation of these features from the image that is input to the CNN. Meanwhile, the pooling layer is tasked with reducing the spatial resolution of feature maps. Generally, before *fully connected layer,* there are piles of several *convolutional* and *pool layer* which serves to extract a more abstract feature representation. After that, *fully connected layer* will interpret these features and perform the required functions *high-level reasoning.* Classification at the end of CNN will use the function *softmax [*5].

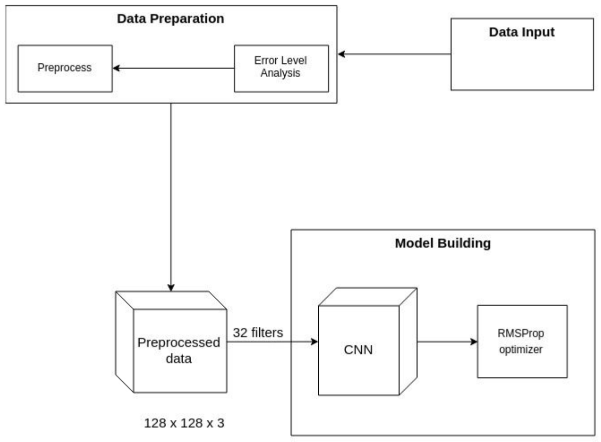
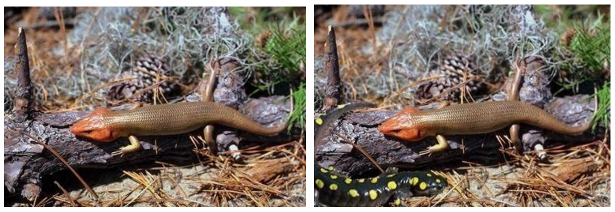


Figure 1. CNN architecture in outline

In general, architectural design is divided into two major parts, namely *data preparation* and *model building.* At the initial stage, the input data consists of images with the format ".jpg", with the following details: 1771 images with labels *tampered* and 2940 images with labels *real* [3], entered into stage *data preparation.* Stage *data preparation* is the stage where each image which is the input data is first converted into a result image *Error Level Analysis.* Then, the ELA image will be *resize* be a picture with

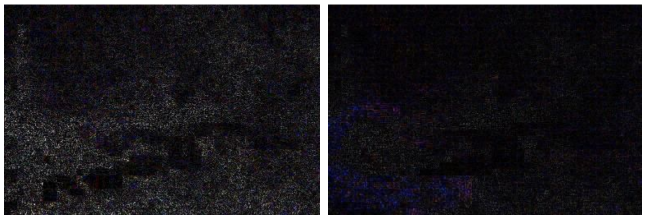
size 128 x 128.



a) **b)**

Figure 2. a) An example of an original drawing of a lizard and b) an example modified image

Converting raw data to ELA-generated images is a method used to increase the training efficiency of the CNN model. This efficiency can be achieved because the resulting ELA image contains information that is not as redundant as the original image. The features generated by the ELA image are already focused on the part of the image that has a level *error* above the limit. Besides that, *pixels* in ELA images tend to have colors that are similar to or even very contrasting with *pixels* nearby, so that CNN model training becomes more efficient.



a) **b)**

Figure 3. a) ELA image results from Figure 2a) and b) ELA image results from Figure 2b)

After that, the image size is changed. The next step is to normalize by dividing each RGB value by 255.0 to perform normalization, so that CNN converges faster (reaching the global minimum of the value). *loss* property of validation data) because the value of each RGB value only ranges between 0 and 1. The next step is to change the label on a data, where 1 represents *tampered* and 0 represents *real* Becomes *categorical value.* After that, the distribution of training data and validation data was carried out using the distribution of 80% for training data and 20% for validation data.

The next step is to use training data and validation data to conduct model training *deep learning* by using CNN. Optimizations applied during training are *RMSProp optimizer* , which is one method *adaptive learning rate.*

The complete architecture used in the *model building* can be seen in the image below or by using the link[] which is a complete architectural drawing.

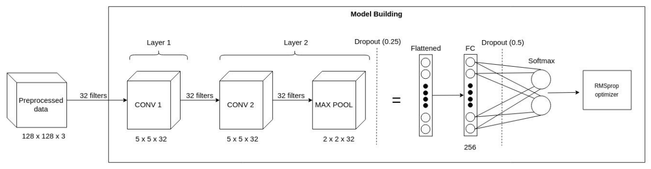


Figure 4. CNN model development architecture

On models *deep*
CNN consists of *learning* used the first layer *convolutional layers* kernel size of 5x5 and the number of filters is 32. The second layer of CNN consists of *convolutional layers* with a kernel size of 5x5 and a number of filters as many as 32, and *Max Pooling layer* with a size of 2x2. Second *convolutional layers* used using *kernel initializer glorot uniform,* and the ReLU activation function to make neurons present in *convolutional layers* make a selection so that it can receive a useful signal from the input data [9].

After that, layers *MaxPooling* added *dropout* as big as 0.25 to prevent *overfitting.* Next layer is *fully connected layer* with the number *neurons* as many as 256 and the ReLU activation function. After *fully connected layer,* will be added *dropout* by 0.5 to prevent *overfitting.* Layers *output* used has an activation function *softmax.*

In the architecture used, only two *convolutional layers* needed, because the results resulting from the conversion process into an ELA image can highlight important features to determine whether an image is original or has been properly modified.

7. ANALYSIS

The results obtained from the proposed method have a maximum accuracy of 91.83%. Accuracy curve drawing and curve *loss* can be seen in the image below.

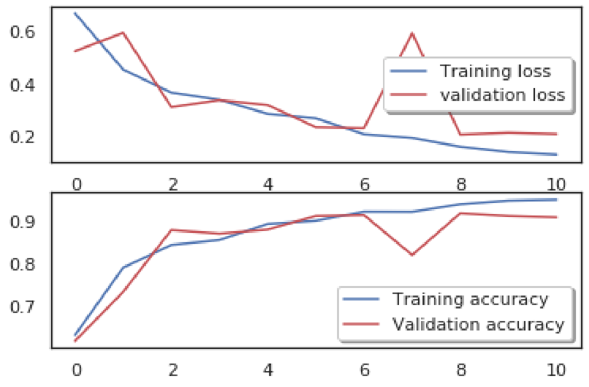


Figure 5. Accuracy curve and loss curve for training data and validation data

It can be seen in the image above that the best accuracy is obtained at *epoch* 9th. Mark *validation loss* after *epoch* 9th start flat
is a sign of *overfitting,* and finally increased, which. Identification method total *epoch* a good one to use during training is *early stops.* With this method, training will be saved

decrease or value *validation loss* watch when starts to increase. validation accuracy value

Amount *epoch* Less training is required to achieve convergence, because the use of the ELA converted image feature makes model training much more efficient, and normalization is performed on the RGB values for each *pixels* also accelerates the convergence of the CNN model.

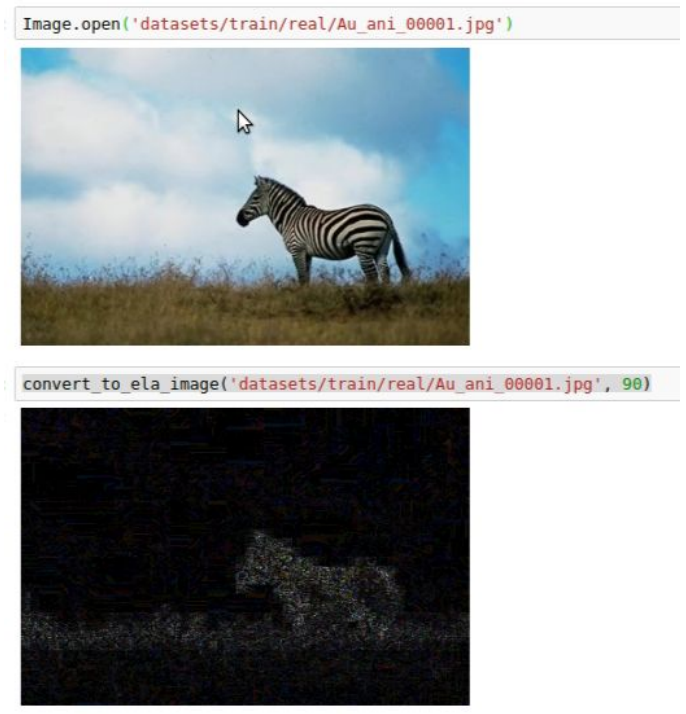
The accuracy results obtained by the model in classifying can be said to be high. This is an indication that the feature is an ELA gambar image successfully used to classify whether the image is an original image or has been modified.

8. CONCLUSION

In this study, there are several things that can be concluded from the results of machine learning using ELA and CNN.

1. CNN uses two *convolutional layers,* one *MaxPooling layer,* one *fully connected layer,* and one *output layer* with *softmax* can achieve 91.83% accuracy.
2. The use of ELA can increase efficiency and reduce computational costs of the training process. This can be seen from the reduction in the number of layers from the previous method [11] and the number of *epoch* required. In the proposed model, the number *epoch* all it takes to reach convergence is 9.

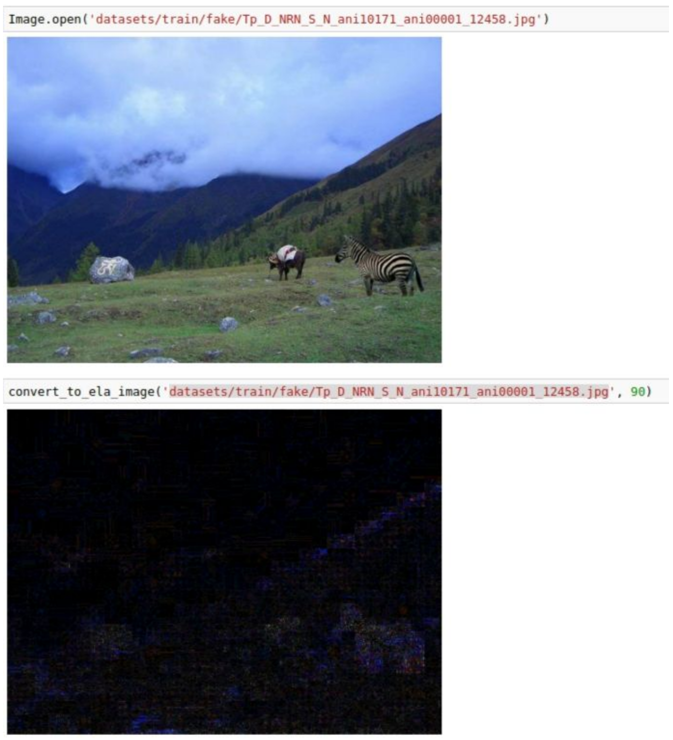
9. DOCUMENTATION



```
Image.open('datasets/train/real/Au_ani_00001.jpg')
```

```
convert_to_ela_image('datasets/train/real/Au_ani_00001.jpg', 90)
```

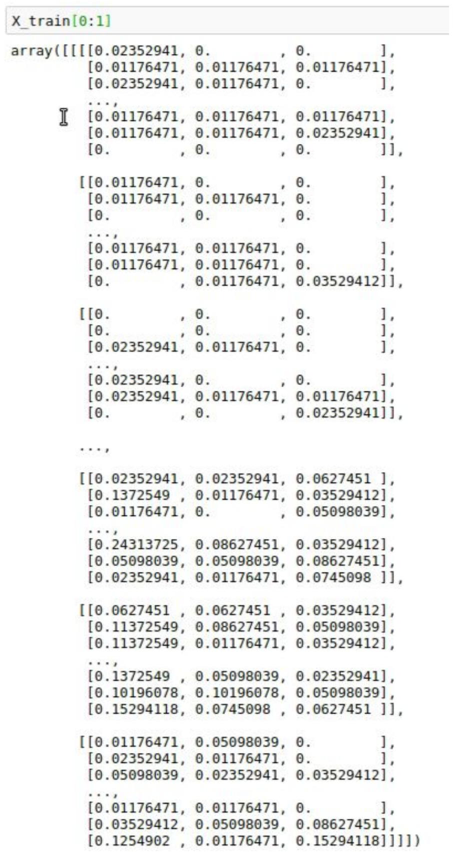
Figure 6. Convert the original image to an ELA result image



```
Image.open('datasets/train/fake/Tp_D_NRN_S_N_ani10171_ani00001_12458.jpg')
```

```
convert_to_ela_image('datasets/train/fake/Tp_D_NRN_S_N_ani10171_ani00001_12458.jpg', 90)
```

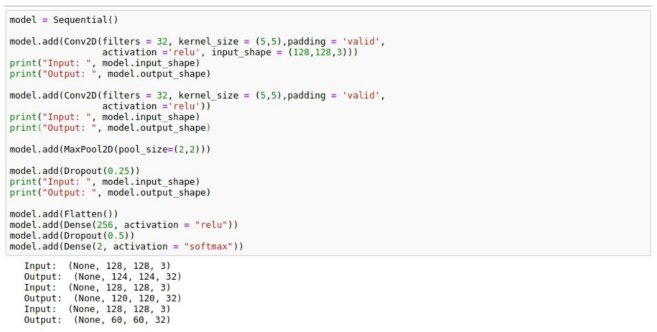
Figure 7. Image conversion *tampered* to the ELA result image



```
X_train[0:1]
```

```
array([[[[0.02352941, 0.        , 0.        ],
         [0.01176471, 0.01176471, 0.01176471],
         [0.02352941, 0.01176471, 0.        ],
         ...,
         [0.01176471, 0.01176471, 0.01176471],
         [0.01176471, 0.01176471, 0.02352941],
         [0.        , 0.        , 0.        ]],

        [[0.01176471, 0.        , 0.        ],
         [0.01176471, 0.01176471, 0.        ],
         [0.        , 0.        , 0.        ],
         ...,
         [0.01176471, 0.01176471, 0.        ],
         [0.01176471, 0.01176471, 0.        ],
         [0.        , 0.01176471, 0.03529412]],

        [[0.        , 0.        , 0.        ],
         [0.        , 0.        , 0.        ],
         [0.02352941, 0.01176471, 0.        ],
         ...,
         [0.02352941, 0.        , 0.        ],
         [0.02352941, 0.01176471, 0.01176471],
         [0.        , 0.        , 0.02352941]],

        ...,

        [[0.02352941, 0.02352941, 0.0627451 ],
         [0.1372549 , 0.01176471, 0.03529412],
         [0.01176471, 0.        , 0.05098039],
         ...,
         [0.24313725, 0.08627451, 0.03529412],
         [0.05098039, 0.05098039, 0.08627451],
         [0.02352941, 0.01176471, 0.0745098 ]],

        [[0.0627451 , 0.0627451 , 0.03529412],
         [0.11372549, 0.08627451, 0.05098039],
         [0.11372549, 0.01176471, 0.03529412],
         ...,
         [0.1372549 , 0.05098039, 0.02352941],
         [0.10196078, 0.10196078, 0.05098039],
         [0.15294118, 0.0745098 , 0.0627451 ]],

        [[0.01176471, 0.05098039, 0.        ],
         [0.02352941, 0.01176471, 0.        ],
         [0.05098039, 0.02352941, 0.03529412],
         ...,
         [0.01176471, 0.01176471, 0.        ],
         [0.03529412, 0.05098039, 0.08627451],
         [0.1254902 , 0.01176471, 0.15294118]]]])
```

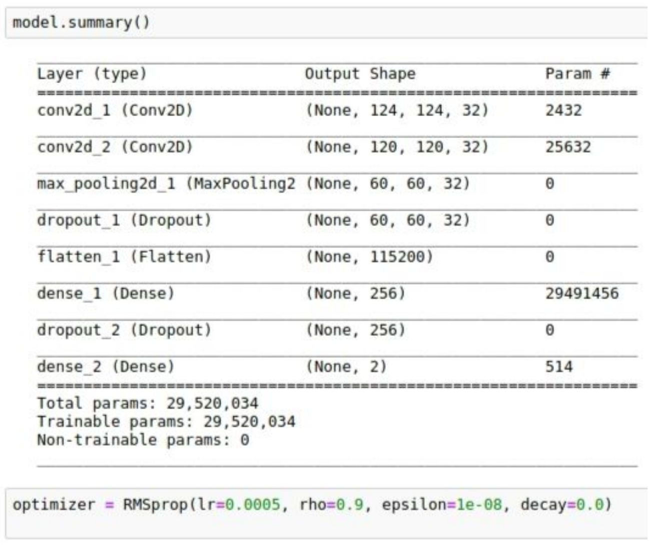
Figure 8. Training data output from the module *data preparation*



```
model = Sequential()

model.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'valid',
                 activation ='relu', input_shape = (128,128,3)))
print("Input: ", model.input_shape)
print("Output: ", model.output_shape)

model.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'valid',
                 activation ='relu'))
print("Input: ", model.input_shape)
print("Output: ", model.output_shape)

model.add(MaxPool2D(pool_size=(2,2)))

model.add(Dropout(0.25))
print("Input: ", model.input_shape)
print("Output: ", model.output_shape)

model.add(Flatten())
model.add(Dense(256, activation = "relu"))
model.add(Dropout(0.5))
model.add(Dense(2, activation = "softmax"))
```

```
Input:  (None, 128, 128, 3)
Output:  (None, 124, 124, 32)
Input:  (None, 128, 128, 3)
Output:  (None, 120, 120, 32)
Input:  (None, 128, 128, 3)
Output:  (None, 60, 60, 32)
```

Figure 9. Module *model building*

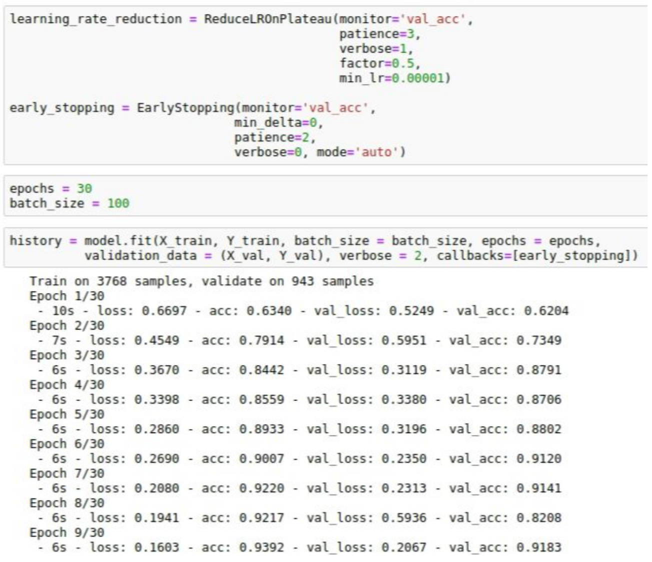


```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 124, 124, 32)      2432
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 120, 120, 32)      25632
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 60, 60, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 60, 60, 32)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 115200)            0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               29491456
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 514
=================================================================
Total params: 29,520,034
Trainable params: 29,520,034
Non-trainable params: 0
_________________________________________________________________
```

```
optimizer = RMSprop(lr=0.0005, rho=0.9, epsilon=1e-08, decay=0.0)
```

Figure 10. Summary of the model



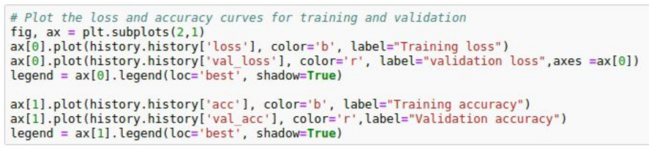
```
learning_rate_reduction = ReduceLROnPlateau(monitor='val_acc',
                                            patience=3,
                                            verbose=1,
                                            factor=0.5,
                                            min_lr=0.00001)

early_stopping = EarlyStopping(monitor='val_acc',
                              min_delta=0,
                              patience=2,
                              verbose=0, mode='auto')
```

```
epochs = 30
batch_size = 100
```

```
history = model.fit(X_train, Y_train, batch_size = batch_size, epochs = epochs,
          validation_data = (X_val, Y_val), verbose = 2, callbacks=[early_stopping])
```

```
Train on 3768 samples, validate on 943 samples
Epoch 1/30
 - 10s - loss: 0.6697 - acc: 0.6340 - val_loss: 0.5249 - val_acc: 0.6204
Epoch 2/30
 - 7s - loss: 0.4549 - acc: 0.7914 - val_loss: 0.5951 - val_acc: 0.7349
Epoch 3/30
 - 6s - loss: 0.3670 - acc: 0.8442 - val_loss: 0.3119 - val_acc: 0.8791
Epoch 4/30
 - 6s - loss: 0.3398 - acc: 0.8559 - val_loss: 0.3380 - val_acc: 0.8706
Epoch 5/30
 - 6s - loss: 0.2860 - acc: 0.8933 - val_loss: 0.3196 - val_acc: 0.8802
Epoch 6/30
 - 6s - loss: 0.2690 - acc: 0.9007 - val_loss: 0.2350 - val_acc: 0.9120
Epoch 7/30
 - 6s - loss: 0.2080 - acc: 0.9220 - val_loss: 0.2313 - val_acc: 0.9141
Epoch 8/30
 - 6s - loss: 0.1941 - acc: 0.9217 - val_loss: 0.5936 - val_acc: 0.8208
Epoch 9/30
 - 6s - loss: 0.1603 - acc: 0.9392 - val_loss: 0.2067 - val_acc: 0.9183
```

Figure 11. Data training



```
# Plot the loss and accuracy curves for training and validation
fig, ax = plt.subplots(2,1)
ax[0].plot(history.history['loss'], color='b', label="Training loss")
ax[0].plot(history.history['val_loss'], color='r', label="validation loss",axes =ax[0])
legend = ax[0].legend(loc='best', shadow=True)

ax[1].plot(history.history['acc'], color='b', label="Training accuracy")
ax[1].plot(history.history['val_acc'], color='r',label="Validation accuracy")
legend = ax[1].legend(loc='best', shadow=True)
```

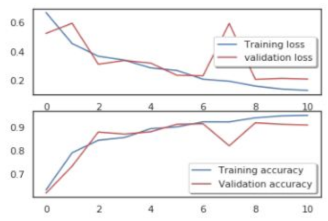
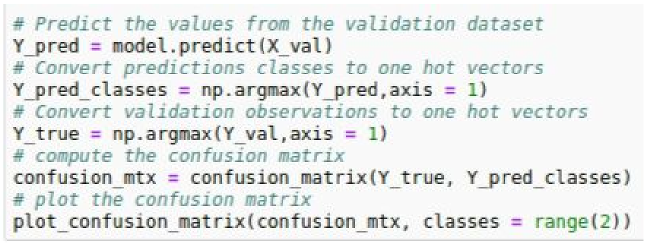


Figure 12. Curve *loss* for training data and validation data



```
# Predict the values from the validation dataset
Y_pred = model.predict(X_val)
# Convert predictions classes to one hot vectors
Y_pred_classes = np.argmax(Y_pred,axis = 1)
# Convert validation observations to one hot vectors
Y_true = np.argmax(Y_val,axis = 1)
# compute the confusion matrix
confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)
# plot the confusion matrix
plot_confusion_matrix(confusion_mtx, classes = range(2))
```

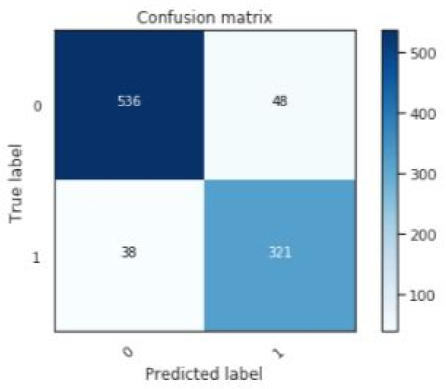


Figure 13. *Confusion matrix* of validation data (1 represents *tampered*, 0 represents the original image)

10. THANK YOU

The author's thanks for the use of the dataset *CASIA Image Tempering Detection Evaluation Database (CAISA TIDE) V2.0* addressed to *National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Science, Corel Image Database* and the photographers.

11. REFERENCE

[1] zgöbek, zlem, J. A. Gulla, *"Towards an Understanding of Fake News", Norwegian Big Data Symposium*(2017).

[2] Kshetri, Nir, J. Voas, *"The Economics of 'Fake News'", IT Pro (November/December 2017), IEEE Computer Society.*

[3] *Chinese Academy of Sciences. "CASIA Image Tempering Detection Evaluation Database (CAISA TIDE) V2.0.* Taken from http://forensics.idealtest.org

[4] N. Krawetz, "*A pictures worth digital image analysis and forensics," Black Hat Briefings,* p. 1-31, 2007.

[5] Rawat, Waseem, Z. Wang, *"Deep Convolutional Neura Networks for Image Classification: A Comprehensive Review" l , Neural Computation 29*(2017), p. 2352-2449.

[6] Gunawan, Teddy Surya, Hanafiah, SAM, Kartiwi, M., Ismail, N., Za'bah, NF, Nordin, AN, *"Development of Photo Forensics Algorithm by Detecting Photoshop Manipulation Using Error Level Analysis", Indonesian Journal of Electrical Engineering and Computer Science, Vol. 7, No. 1,* July 2017, p. 131-137.

[6] *Photo Forensics: Detect Photoshop Manipulation with Error Level Analysis,* September 2018. Taken from https:// resources.infosecinstitute.com/error-level-analysis-detect-i mage-manipulation/#gref

[7] Edelman, *"2018 Edelman Trust Barometer Global Report",* taken from https://cms.edelman.com/sites/default/files/2018-01/2018%20Edelman%20Trust%20Barometer%20Global%20Report.pdf

[8] Bulls, Jeff. *"6 Powerful Reasons Why you Should include Images in your Marketing",* taken from https://www.jeffbullas.com/6-powerful-reasons-why-you-should-in include-images-in-your-marketing-infographic/

[9] V. Nair and GE Hinton. *"Rectified linear units improve restricted boltzmann machines," Proceedings of the 27th International Conference o n Machin e-Learning g,* 21-24 June 2010, p. 807-814.

[10] Villan, M. Afsal, Kuruvilla, K., Paul, J., Elias, EP, *"Fake Image Detection Using Machine Learning", International Journal of Computer Science and Information Technology & Security (IJCSITS), Vol. 7, No. 2,* 2017.

[11] Rao, Yuan, Ni, J., *"A Deep Learning Approach to Detection of Splicing and Copy-Move Forgeries in Images", 2016 IEEE International Workshop on Information Forensics and Security (WIFS).*