FALSE IMAGE DETECTION WITH ELA AND DEEP LEARNING

Agus Gunawan, Holy Lovenia, Adrian Hartarto Pramudita

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

ABSTRACT

Images are often manipulated with the intent and purpose of benefiting one party. Whereas, 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 image forgery, a large amount of image data is needed, and a model that can process every ***pixel*** in the image. 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 a ***Convolutional Neural Network*** (CNN) architecture that utilizes ***Error Level Analysis*** (ELA), image fraud detection can reach 91.83% and convergence is only 9 ***epochs.***

Keywords— image fraud detection, convolutional neural network, error level analysis, deep learning, big data

1. BACKGROUND

According to ***the 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 gain economic benefits, whether it's through increasing the number of news clicks or making news that shouldn't benefit one party [1].

In addition, fake news can also affect stock prices, which can benefit those who release the news. Another reason is to gain support or bring down other parties socially or politically [2].

Based on statistics courtesy of the Telematics Society Indonesia (MASTEL) in 2017, the types of fake news that were most often received were socio-political, SARA (ethnicity, religion, and race), health, food and drink, financial fraud, and science and technology. As many as 84.5% of all respondents stated that they felt disturbed by the existence of fake news, and more than 70% agreed that fake news disrupted community harmony and hindered development.

Apart from writing, around 40% of respondents stated that the spread of fake news was often accompanied by pictures. Images are used by humans to reproduce reality, and are often used as evidence for news, publications, or facts.

Fake news that has supporting images tends to be accepted and trusted by the public.

In general, humans are easier to remember the form of pictures than writing. According to ***the Social Science Research Network,*** as many as 65% of humans are people who enjoy learning through visuals. In ***marketing*** and ***visual science,*** it is stated that images have a very big influence on an article.
People are more likely to respond when pictures are present 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 spreading information.

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

This technology requires a lot of image data, and each image has many constituent ***pixels .*** With ordinary machine learning, this technology will be difficult to develop. So, ***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 fraud detection has two main objectives as follows.

1. Propose a new method using ***deep learning*** to classify images as original images and modified images with a simpler architecture, so that computational costs can be reduced 2. Involve the use of ELA in machine learning as an effort to increase efficiency

There is some impetus behind these two main goals. As is well known, there have been several studies in the past that also aim to detect image fraud [10, 11]. However, most of these researches require considerable computational costs (can be seen from the number of ***epochs*** and ***layers*** required), so that the flexibility of the proposed method is reduced and it is difficult to apply in everyday life due to computational costs. However, there is a need for image fraud 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 fraud 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 fraud detection can be used for the following things.

1. Increasing convenience in obtaining information that is in accordance with facts 2. The public
gets consideration in determining whether an image is genuine or fake

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

4. LIMITATIONS

There are several limitations that apply to image fraud detection data mining, namely the raw data must be in the form of an image with ***lossy compression*** (for example .jpg), nor is it ***a computer generated image*** (CGI).

5. METHOD

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

5.1. *Error Level Analysis (ELA)*

Error Level Analysis is a technique used to detect image manipulation by re-storing images at a certain quality level and calculating the comparison between the compression levels [4]. In general, this technique is performed on images that have a ***lossy*** format ***(lossy compression).*** The image type used in this data mining is JPEG.

In JPEG images, compression is done independently for every 8x8 pixels in the image. If an image is not manipulated, every 8x8 pixels in the image must have the same ***error*** rate [6].

5.2. *Convolutional Neural Network (CNN)*

CNN is a type of feedforward- based ***network ,*** where the flow of information is only one way, namely from input to output. Although there are several types of CNN architectures, in general, CNNs have several ***convolutional layers*** and ***pooling layers.*** Then, it is followed by one or more ***fully connected layers.*** In image classification, the input to CNN is in the form of an image, so that each ***pixel*** can be processed [5].

In short, ***the convolutional layer*** is used as a feature extractor that studies the representation of these features from the images that are input to the CNN.
Meanwhile, the pooling layer is responsible for reducing the spatial resolution of feature maps. Generally, before ***the fully connected layer,*** there are several ***convolutional*** and ***pooling layers*** that function to extract a more abstract feature representation.
After that, ***the fully connected layer*** will interpret these features and perform functions that require ***high-level reasoning.*** Classification at the end of CNN will use the ***softmax*** function [5].

6. DESIGN AND IMPLEMENTATION

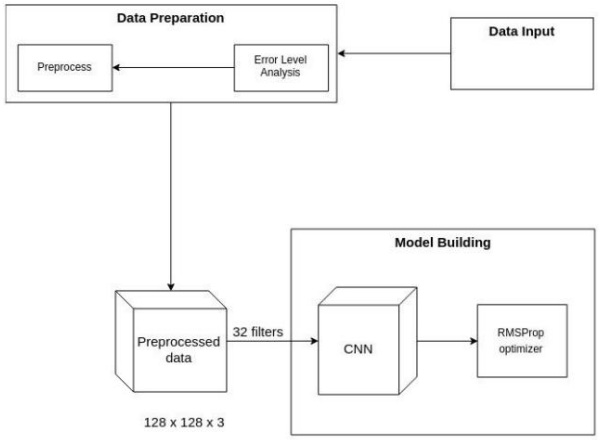


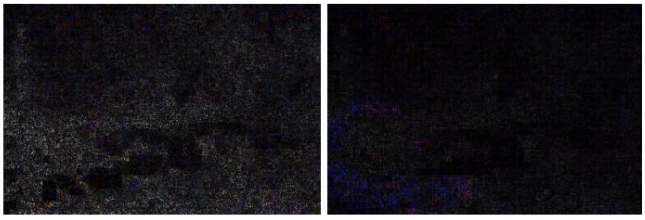
Figure 1. CNN architecture in outline

In general, architectural design is divided into two major parts, namely ***data preparation*** and ***model building.*** In the initial stage, input data consisting of images in ".jpg" format, with the following details: 1771 images with ***tampered*** labels and 2940 images with ***real*** labels [3], are entered into the ***data preparation stage. The data preparation*** stage is the stage where each image which is input data is converted first to an image resulting from ***the Error Level Analysis.*** Then, the ELA image will ***be resized*** to an image with a size of 128 x 128.

a) b)

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

Converting raw data to images resulting from ELA is a method used to increase the training efficiency of the CNN model. This efficiency can be achieved because the resulting ELA images contain information that is not as redundant as the original image. The features generated by ELA images have been focused on the part of the image that has an ***error*** level above the limit. In addition, ***the pixels*** in the ELA image tend to have colors that are similar or in contrast to the adjacent ***pixels ,*** so that the training of the CNN model 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 normalize, so that CNN converges more quickly (reaching the global minimum of the ***loss*** value belonging to the 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*** to become ***a categorical value.*** After that, the distribution of training data and validation data was carried out using a division of 80% for training data and 20% for validation data.

The next step is to use training data and validation data to train ***deep learning*** models using CNN. The optimization applied during training is ***RMSProp optimizer,*** which is one of the ***adaptive learning rate methods.***

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

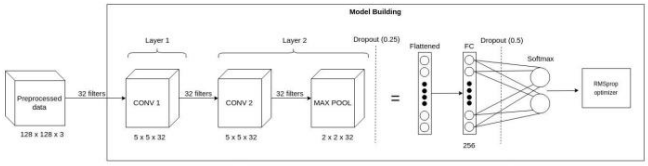


Figure 4. The architecture of the CNN model development

In the ***deep learning*** model used, the first layer of CNN consists of ***a convolutional layer*** with a kernel size of 5x5 and a total of 32 filters. The second layer of CNN consists of ***a convolutional layer*** with a kernel size of 5x5 and a total of 32 filters, and a ***Max Pooling layer*** with a size of 2x2. The two ***convolutional layers*** used use ***the glorot uniform kernel initializer,*** and the ReLU activation function to make the neurons in the ***convolutional layer*** make a selection so that they can receive useful signals from the input data [9].

After that, the ***MaxPooling*** layer is added ***a dropout*** of 0.25 to prevent ***overfitting.*** Next layer

is a ***fully connected layer*** with 256 ***neurons*** and the ReLU activation function. After ***the layer is fully connected, a dropout*** of 0.5 will be added to prevent ***overfitting.*** The ***output*** layer used has a ***softmax activation function.***

In the architecture used, only two ***convolutional layers*** are needed, because the results resulting from the conversion process into ELA images 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%. The image of the accuracy curve and ***loss*** curve can be seen in the image below.

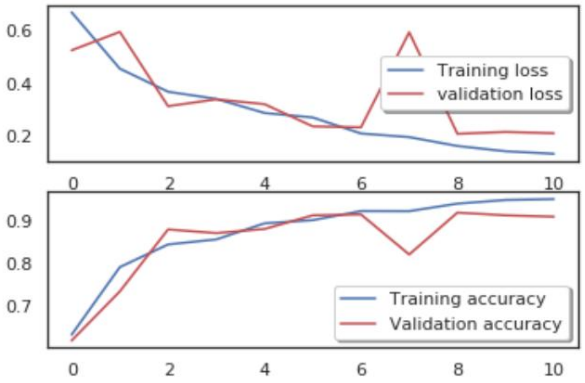


Figure 5. Accuracy curve and loss curve for training data and validation data

It can be seen in the image above that the best accuracy is obtained at the 9th ***epoch .*** The ***validation loss*** value after the 9th ***epoch*** starts to flat and eventually increases, which is a sign of ***overfitting.*** A good method of identifying the number of ***epochs*** to use during training is ***early stopping.*** With this method, training will be stopped when the validation accuracy value starts to decrease or the ***validation loss*** value starts to increase.

The number of training ***epochs*** required is small to achieve convergence, because the use of the converted image feature of ELA makes model training much more efficient, and the normalization performed on the RGB values for each ***pixel*** also accelerates the convergence of the CNN model.

The results of the accuracy obtained by the model in carrying out the classification can be said to be relatively high. This is an indication that the feature is an ELA image successfully used to classify whether the image is the original image or has undergone modification.

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*** which can achieve 91.83% accuracy.
2. The use of ELA can increase efficiency and reduce the computational cost 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 ***epochs*** required. In the proposed model, the number of ***epochs*** needed to reach convergence is only 9.

9. DOCUMENTATION

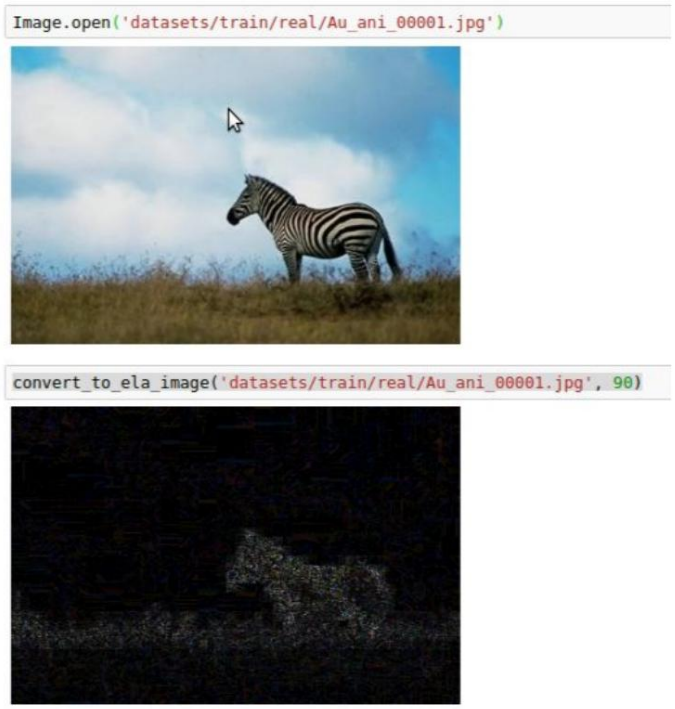


Figure 6. Conversion of the original image into an ELA result image

```
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. Convert *tampered* image to 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 *data preparation* module

```
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. *Model building* module

```
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)
```

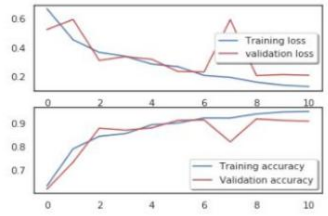


Figure 12. *Loss* curve **for training data and validation data**

```
# 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))
```

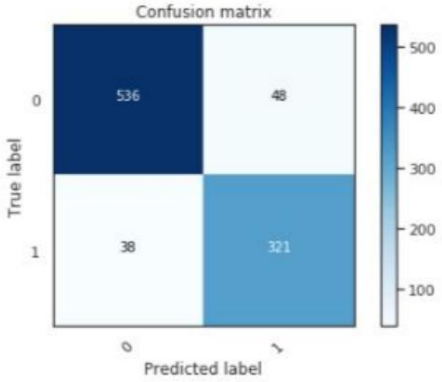


Figure 13. *Confusion matrix* **of validation data (1 represents *tampered,* 0 represents the original image)**

10. THANK YOU

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

11. REFERENCES

[1] Özgöbek, Özlem, 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 Science. "CASIA Image Tempering Detection Evaluation Database (CAISA TIDE) V2.0.*** Diambil dari http://forensics.idealtest.org

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

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

[6] Gunawan, Teddy Surya, Hanafiah, S. A. M., Kartiwi, M., Ismail, N., Za'bah, N. F., Nordin, A. N., ***"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,*** Juli 2017, hlm. 131-137.

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

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

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

[9] V. Nair and G. E. Hinton. ***"Rectified linear units improve restricted boltzmann machines," Proceedings of the 27th International Conference on Machine Learning,*** 21-24 Juni 2010, hlm. 807-814.

[10] Villan, M. Afsal, Kuruvilla, K., Paul, J., Elias, E. P., ***"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).***