Image Tampering Detection using Error Level Analysis and Deep Learning

Geetha D
Senior Engineer, Software SBU
Bharat Electronics Ltd
Jalahalli,,Bangalore 560013
Email: geethad@bel.co.in

Vani Patil
Manager, Software SBU
Bharat Electronics Ltd
Jalahalli,,Bangalore 560013
Email: vanipatil@bel.co.in

***Abstract* – Images are becoming increasingly ubiquitous in everyday life and are used in various paramount fields like medical science, journalism, sports, criminal investigation where authenticity of image is of vital importance. Tampering, forging images and identifying such images are promising research areas in this digital era. We propose methodologies to identify such photo images using neural network which also recognizes the regions of the image that have been manipulated and reveals the segments of the original image.**

Index Terms –Tampered Images, Forged Images, Error Level Analysis, Multilayer Perception Network, Deep Neural Network.

I. Introduction

Image has remarkable role in various areas such as medical imaging, sports, surveillance systems, forensic investigation, criminal investigation, intelligence system, legal services, insurance claim, and journalism. Due to the availability of sophisticated tools for image manipulation, it is very easy to tamper the image by anyone. A huge number of people have become victims of image forgery. Social media is a great platform to socialize, share and spread knowledge but if caution is not exercised, it can mislead people. So, to put an end to this, the images should be categorized as real or fake accurately.

Digital images offer many attributes for tamper detection algorithm - the color and brightness of individual pixels as well as an image's resolution and format. These properties allow for analysis and comparison between the fundamentals of digital forgeries in an effort to develop an algorithm for detecting image tampering. This paper focuses on images saved in the JPEG format. Therefore, a research work on basis of compression scheme is discussed to determine what information can be gathered about a digital forgery saved in this format.

A. Image Format Analysis

Images can be stored in a variety of formats. Some, such as RAW, only contain pixel data, while other formats contain a wealth of information. In many cases, such as JPEG, GIF, PNG, and TIFF, the format can be as informative as the image. Changes to the image yield changes to the file format. Although this paper focuses on the JPEG format, the methods can be applied to other complex formats including GIF, PNG, and TIFF. JPEG files contain a well-defined feature set. Changes to the image will modify the feature set. Thus, if the features indicate manipulation then the manipulation can be identified. The feature set for JPEG includes Meta data, quantization tables for image compression, lossy data compression, and subdivided image processing using 8x8 pixel cells.

B. Error Level Analysis

Error Level Analysis (ELA) is a forensic technique on the image to analyze images through different levels of compression. This technique permits identifying areas within an image that are at different compression levels. With JPEG images, the entire picture should be at roughly the same level. If a section of the image is at a significantly different error level, then it likely indicates a digital modification.

JPEG is a lossy format, but the amount of error introduced by each resave is not line. Any modification to the picture will alter the image such that stable areas (no additional error) become unstable. Fig.1 shows a modified image using Photoshop. The modified picture was based on the first 75% resave. Books on the shelf were duplicated and a toy dinosaur was added to the shelf. The 95% ELA identifies the changes since they are areas that are no longer at their minimal error level. Additional areas of the picture show slightly more volatility because Photoshop merged information from multiple layers, effectively modifying many of the pixels.

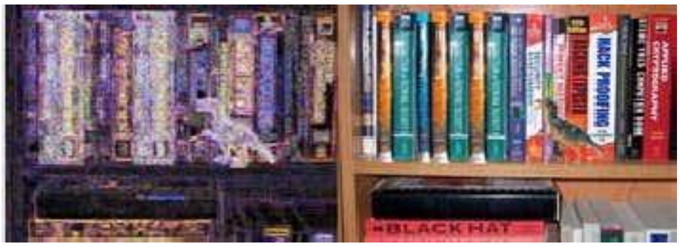


Fig.1. Error level analysed image on the left and fake image on the right

C. Machine Learning

The process of machine learning is similar to that of data mining. Both systems search through data to look for patterns. However, instead of extracting data for human comprehension as is the case in data mining applications machine learning uses that data to detect patterns in data and adjust program actions accordingly. Machine learning algorithms are often categorized as being supervised or unsupervised. Supervised algorithms can apply what has been learned in the past to new data. Unsupervised algorithms can draw inferences from datasets.

In this experiment, a new system that can distinguish forgery image and original images with deep learning has been implemented. Deep learning methods applied in the introduction of false images and the original image is Convolutional Neural Network (CNN). CNN is the development of a multilayer perceptron (MLP) designed to process two-dimensional data. We choose CNN in this research because CNN does not require image processing before processing by a neural network. We are doing this experiment to help the community in distinguishing real images and forgery images that are widely circulated in social media. Besides testing against CNN obtain better results from experiments that have been conducted previously.

II. APPROACH

1. Feature engineering: Error Level Analysis (ELA) and machine learning with deep learning techniques in the form of Convolutional Neural Network (CNN).
2. Instead of using the array/ image matrix from raw image data of pristine and tampered images, we make use of ELA in the learning engine in an effort to increase efficiency of Convolutional Neural Network (CNN).
3. This is much better than using just image matrix obtained from raw images as our input and the results obtained were also much better.

A. Convolutional Neural Network

CNN has a high ability in large-scale image classification. CNN consists of three layers: convolutional layer, pooling layers, and fully connection layers. A Convolutional layer and pooling layer is the most important layer on CNN. Convolutional layer is used for extract feature by combining the image area with many filters. Pooling layer reduces the size of the output map of the convolution layer and prevents overfitting. Through these two layers the number of neurons, parameters, and connections is much less than there is a CNN model. This makes CNN more efficient compared with BP networks with similar layers. The final formula of the single output image channel of the convolution layer as:

$$conv(I,K)_{xy} = \sigma\left(b + \sum_{i=1}^{h} \sum_{j=1}^{w} \sum_{k=1}^{d} K_{ijk} \cdot I_{x+i-1,y+j-1,k}\right)$$

III. ARCHITECTURE

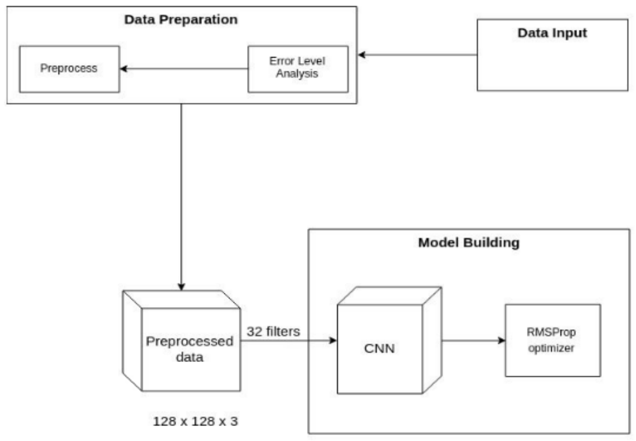


Fig 2: CNN architecture in outline

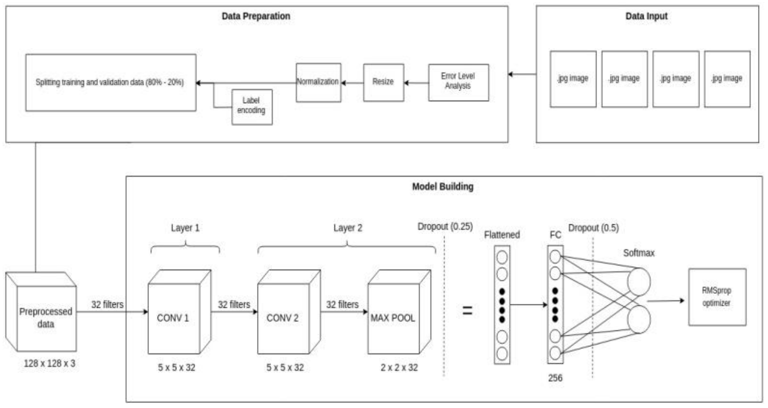


Fig 3: CNN Model building architecture

IV. TRAINING & TESTING

For training, the CASIAV2 splicing Dataset provided by CASIA (Chinese Academy Of Sciences Institute Of Automation) is used. CASIA provides spliced and copy-moved images of various objects. The tampered regions are carefully selected and some post processing like filtering and blurring is also applied. For testing we shall first evaluate using CASIA's validation set and also test with another standard dataset that was given in the IEEE IFS-TC Image Forensics Challenge to evaluate the model.

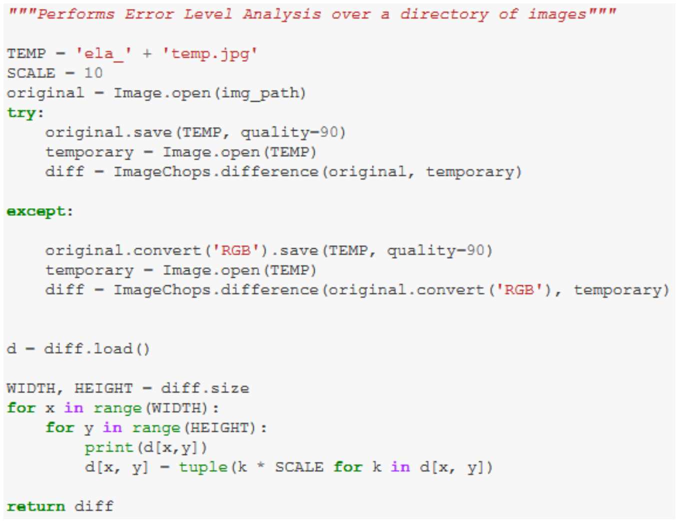
The conversion of raw data to the ELA result image is a method used to increase the training efficiency of the CNN model. This efficiency can be achieved because the results of the ELA image contain information that is not as excessive as the original image. The features produced by the ELA image are focused on the part of the image that has a level error above the limit.

Other than that, the pixels ELA images tend to have colors that are similar to or in sharp contrast to the pixels nearby, making CNN model training more efficient. After that, the image size changes. The next step is to normalize by dividing each RGB value by the number 255.0 to normalize, so that CNN converges faster (reaches the global minimum value The name of this article is: loss belongs to 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 training data and validation data were divided using the 80% division for training data and 20% for validation data.

It can be inferred that when an image is altered, the compression ratio of the specific portion changes with respect to other parts. Fig 1 shows the data preparation and outline of the CNN architecture. The next step is to use training data and validation data to conduct model training deep learning by using CNN as shown in Fig 2.

Preparing the Dataset:
1. Each picture which is the input data is first converted into an Error Level Analysis image. Then, the ELA image will be re-sized to size 128 x 128.
2. Normalize by dividing each RGB value with the number 255.0 to do normalization, so that CNN converges faster.
In the architecture used, only two convolutional layer is needed, because the results generated from the conversion process into an ELA image can highlight important features for knowing whether an image is original or has been modified properly.

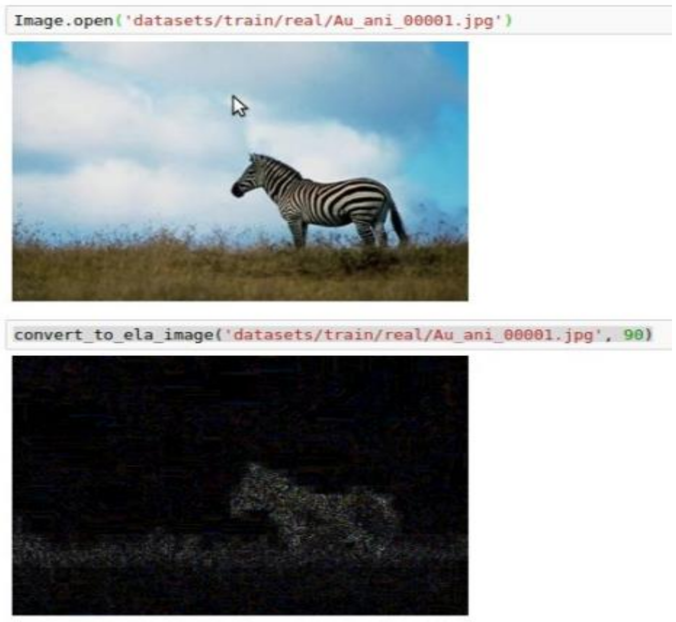
The steps to perform ELA over images can be summarized as:
1. Read in image as JPEG
2. Write image as JPEG with Quality lower level (ex. 90)
3. Read in compressed image (decompress)
4. Take absolute value of the difference between the decompressed image in step 3 and the original image in step 1.



```
"""Performs Error Level Analysis over a directory of images"""

TEMP = 'ela_' + 'temp.jpg'
SCALE = 10
original = Image.open(img_path)
try:
    original.save(TEMP, quality=90)
    temporary = Image.open(TEMP)
    diff = ImageChops.difference(original, temporary)

except:

    original.convert('RGB').save(TEMP, quality=90)
    temporary = Image.open(TEMP)
    diff = ImageChops.difference(original.convert('RGB'), temporary)


d = diff.load()

WIDTH, HEIGHT = diff.size
for x in range(WIDTH):
    for y in range(HEIGHT):
        print(d[x,y])
        d[x, y] = tuple(k * SCALE for k in d[x, y])

return diff
```

V. ANALYSIS & DOCUMENTATION

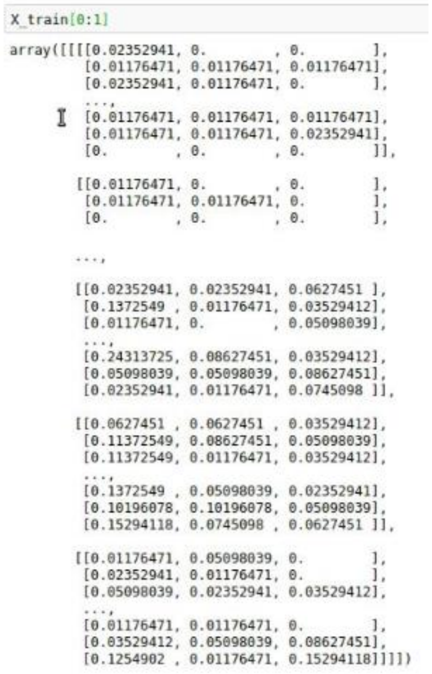
CNN uses two convolutional layer, one MaxPooling layer, one fully connected layer, and one output layer softmax. It can achieve 91.83% accuracy. Loss can be seen in the Fig 9. The use of ELA can increase efficiency and reduce the computational costs of the training process. This can be seen from the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



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

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

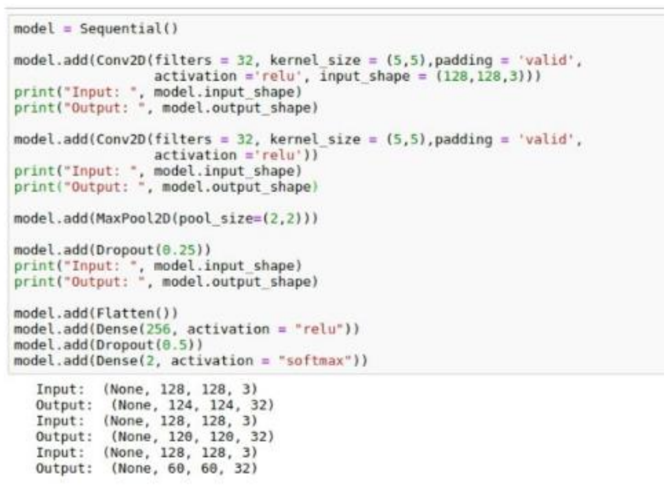
Fig 4. Converting the original image into an ELA result image



```
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.        ],


        ...,

        [[0.02352941, 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]]]])
```

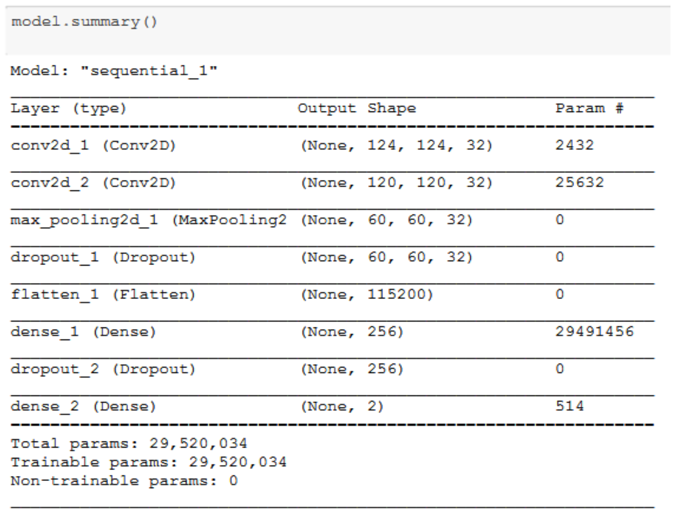
Fig 5. Output training data from data preparation



```
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)
```

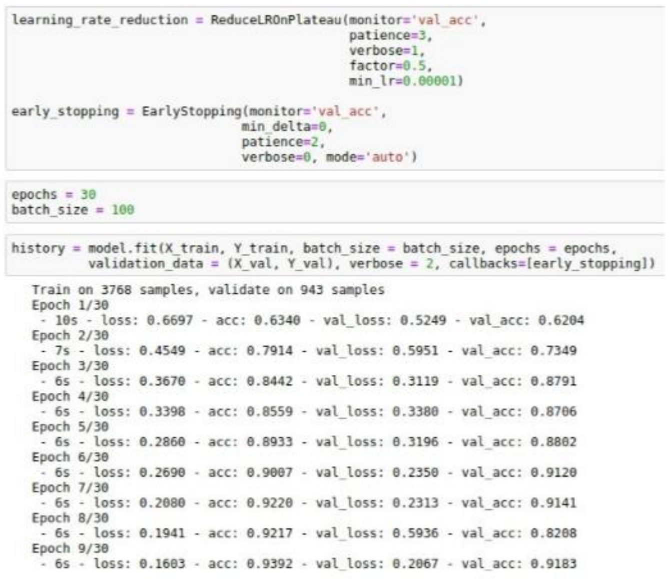
Fig 6. Model Building



```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
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
_________________________________________________________________
```

Fig 7. Model Summary



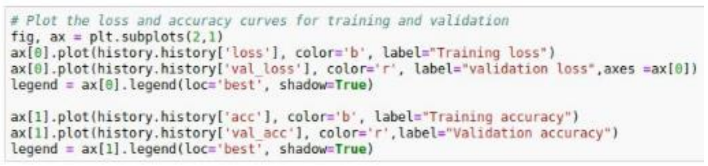
```
learning_rate_reduction = ReduceLROnPlateau(monitor='val_acc',
                                            patience=3,
                                            verbose=1,
                                            factor=0.5,
                                            min_lr=0.00001)

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
```

Fig 8. Training data



```
# 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)
```

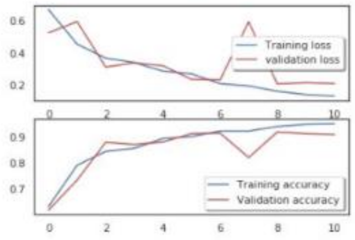
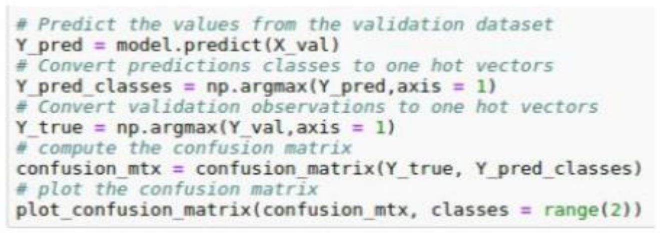


Fig 9. Training and validation accuracy



```
# 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))
```

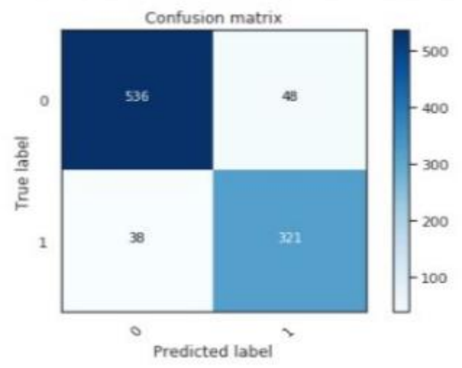


Fig 10. Confusion Matrix of data validation.
(**1** symbolizes tampered, **0** represents the original image)

VI. CONCLUSION

Despite that proper application can allow experts to easily discover image modification (including scaling, cropping and resave operations), ELA analysis depends on the quality of the image. Working on a picture resulting from numerous resave operations isn't effective. If an image is resaved numerous times, then it may have a minimum error level, where more resaves don't alter the image. ELA will return a black image, and no modifications may be detected.

The technique is very effective at discovering alterations introduced with tools like Photoshop or Gimp. By just saving a picture with these applications, users introduce a higher error level potential in the image.

The downside is that these tools could be the cause of unintentional modification. Considered in the analysis of any picture, ELA is just an algorithm to analyze the images. Despite that it's very efficient under specific conditions, it's suggested to integrate it with other forensics tools to provide valid results.

References

[1] A picture's worth, Digital Image Analysis and Forensics, N Krawetz - 2007 Ph D, Hacker Factor Solutions
[2] http://en.wikipedia.org/wiki/Joint_Photographic_Experts_Group
[3] Hashmi MF, Anand V, Keskar AG. Copy-move Image Forgery Detection Using an Efficient and Robust Method Combining Un-decimated Wavelet Transform and Scale Invariant Feature Transform. AASRI Procedia. 2014; 9: 84–91.
[4] http://fotoforensics.com/
[5] Patel HC, Patel MM. Forgery Frame Detection From The Video Using Error Level Analysis. IJAERD. 2015; (6): 242–247.
[6] CASIA Image Tampering Detection Evaluation Database
[7] https://resources.infosecinstitute.com/error-level-analysis-detect-image-manipulation/
[8] http://imagej.net/Welcome
ImageJ is an open source image processing program designed for scientific multidimensional images.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73
[8] http://forensics.idealtest.org/ CASIA v2.0
CASIA V2.0 is with larger size and with more realistic and challenged fake images by using post-processing of tampered regions. It contains 7491 authentic and 5123 tampered color images.
[9] https://www.github.com/afsalashyana/FakeImageDetection
GitHub repositor for fake image detector desktop application