Project Report

Image Forgery Detection using ELA and CNN

Objective

The objective of this project is to develop a Convolutional Neural Network (CNN) model for detecting image forgeries, differentiating between real and manipulated images. The project leverages Error Level Analysis (ELA) as a preprocessing step to enhance the discriminative features in the images.

Methodology

1. Data Collection:

   • Utilize the CASIA dataset, containing authentic and tampered images(spliced and copy-moved images, as well as images affected by post-processing operations such as filtering and blurring), for training and evaluation.

   • Use ELA to preprocess images, enhancing the visual artifacts of manipulation.

2. ELA Preprocessing:

   • Implement the ELA algorithm to highlight error levels in the images.

   • Adjust the ELA parameters such as quality and scaling for optimal results.

   • Integrate ELA preprocessing into the data pipeline.

3. Model Architecture:

   • Design a CNN architecture for image forgery detection.

   • The CNN should take ELA-enhanced images as input.

   • Include convolutional layers, pooling layers, dense layers, and dropout layers for regularization.

4. Data Augmentation:

   • Apply data augmentation techniques to artificially increase the size of the training dataset.

   • Augmentation may include rotation, flipping, and zooming.

5. Model Training:

- Train the CNN using the preprocessed images.
- Use a binary classification setup with labels (real/fake).
- Monitor training progress with validation data and implement early stopping.

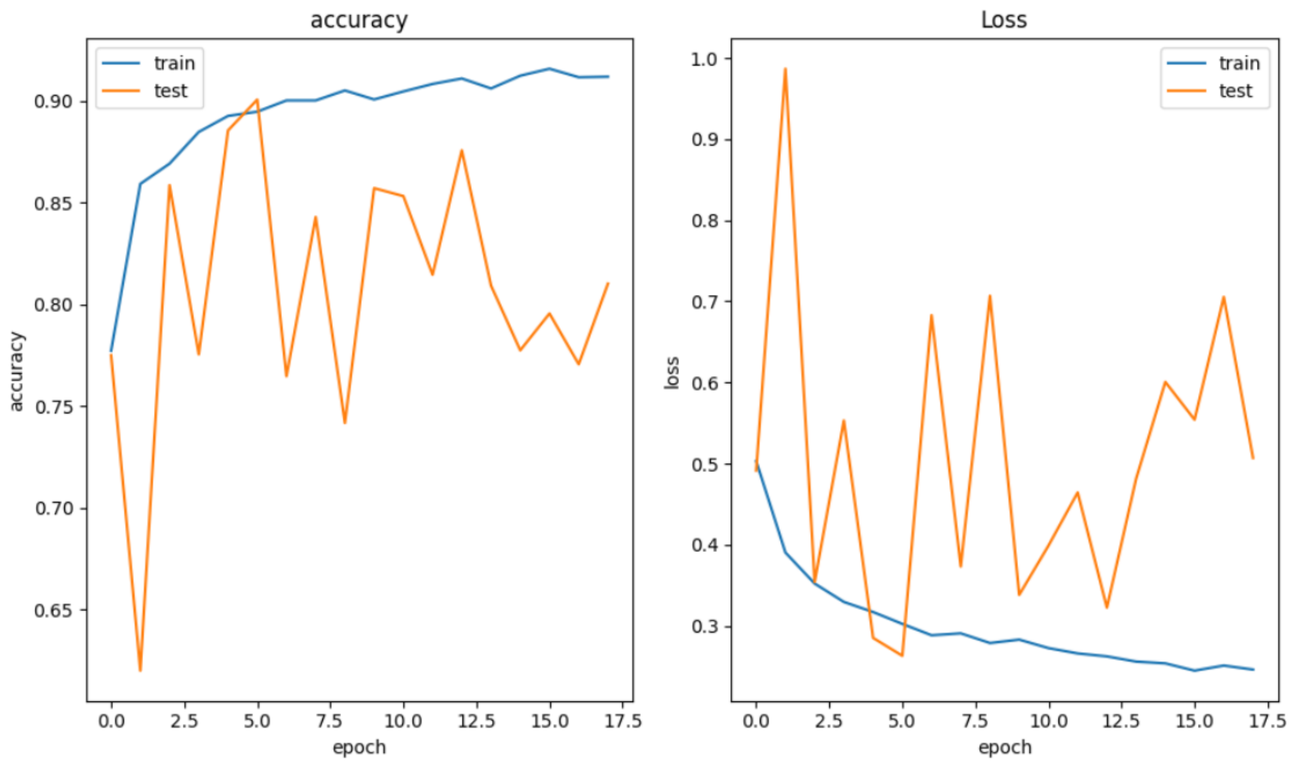
6. Model Evaluation:
    - Evaluate the model's performance on a separate validation set.
    - Visualize confusion matrices to understand model behavior.
7. Hyperparameter Tuning:
    - Experiment with hyperparameter tuning to optimize model performance.
    - Adjust learning rates, batch sizes, and network architecture.
8. Interpretability and Visualization:
    - Visualize feature maps and activations to understand what the model learns.
    - Use visualization tools to identify areas of interest in manipulated images.

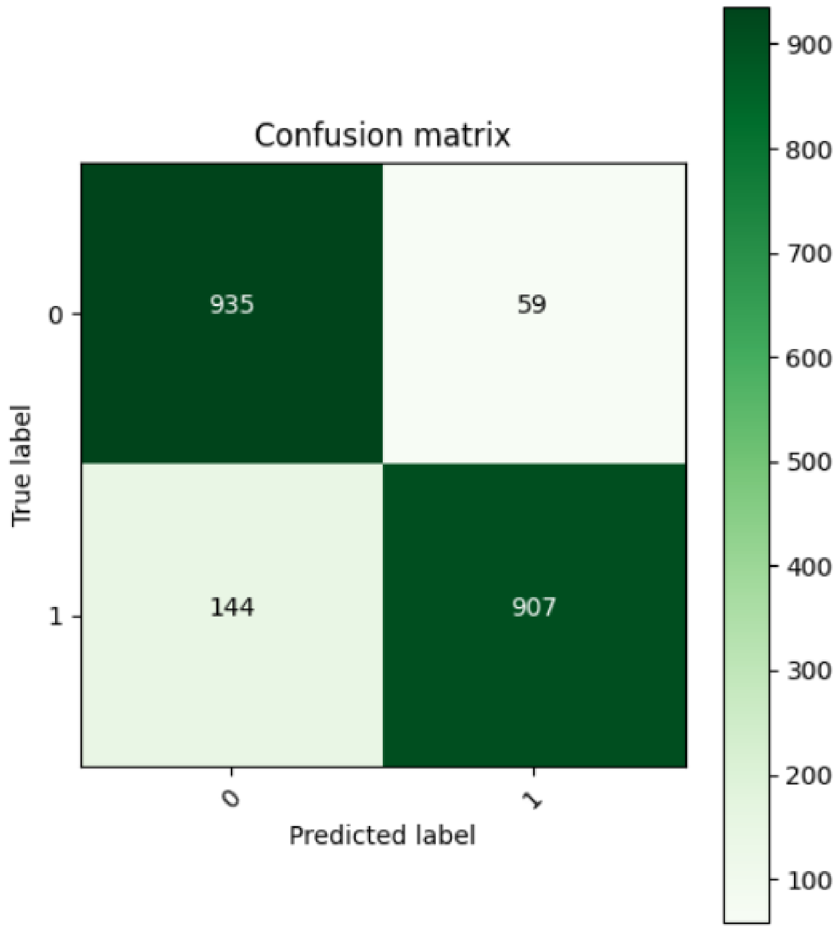
Model Architecture

```
model = Sequential()
model.add(Conv2D(filters=64, kernel_size=(5,5), activation='relu', input_shape=(128,128,3)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(filters=64, kernel_size=(5,5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(units=128, activation='relu'))
model.add(Dropout(0.3))
model.add(BatchNormalization())
model.add(Dense(units=64, activation='relu'))
model.add(Dropout(0.5))
model.add(BatchNormalization())
model.add(Dense(units=1, activation='sigmoid'))
```

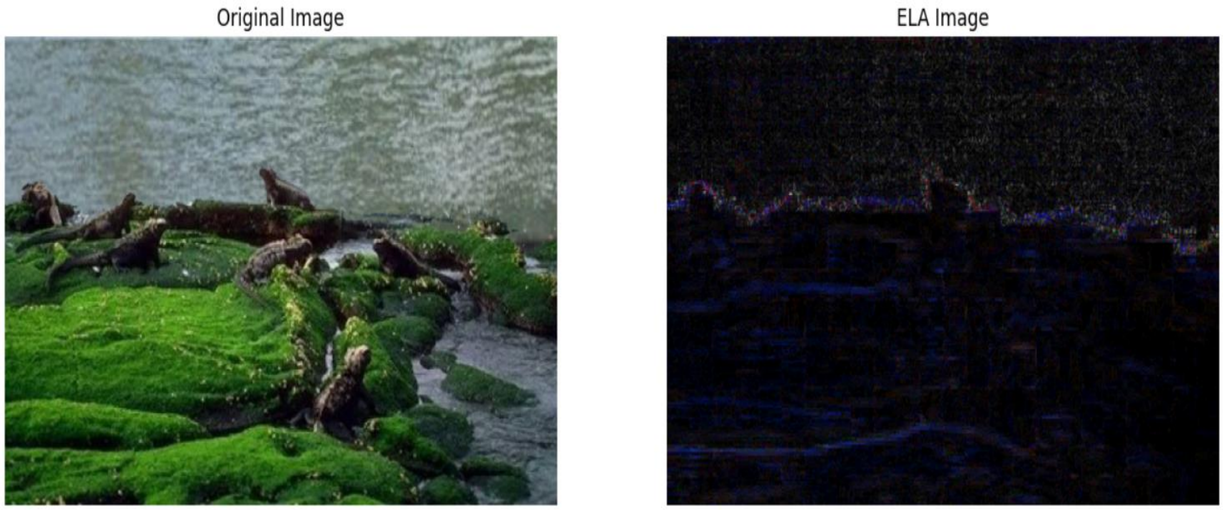
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 124, 124, 64)      4864

 max_pooling2d_6 (MaxPoolin  (None, 62, 62, 64)        0
 g2D)

 conv2d_7 (Conv2D)           (None, 58, 58, 64)        102464

 max_pooling2d_7 (MaxPoolin  (None, 29, 29, 64)        0
 g2D)

 flatten_3 (Flatten)         (None, 53824)             0

 dense_9 (Dense)             (None, 128)               6889600

 dropout_6 (Dropout)         (None, 128)               0

 batch_normalization_6 (Bat  (None, 128)               512
 chNormalization)

 dense_10 (Dense)            (None, 64)                8256

 dropout_7 (Dropout)         (None, 64)                0

 batch_normalization_7 (Bat  (None, 64)                256
 chNormalization)

 dense_11 (Dense)            (None, 1)                 65

=================================================================
Total params: 7006017 (26.73 MB)
Trainable params: 7005633 (26.72 MB)
Non-trainable params: 384 (1.50 KB)
```

Result


accuracy
train
test
accuracy
epoch
Loss
train
test
loss
epoch

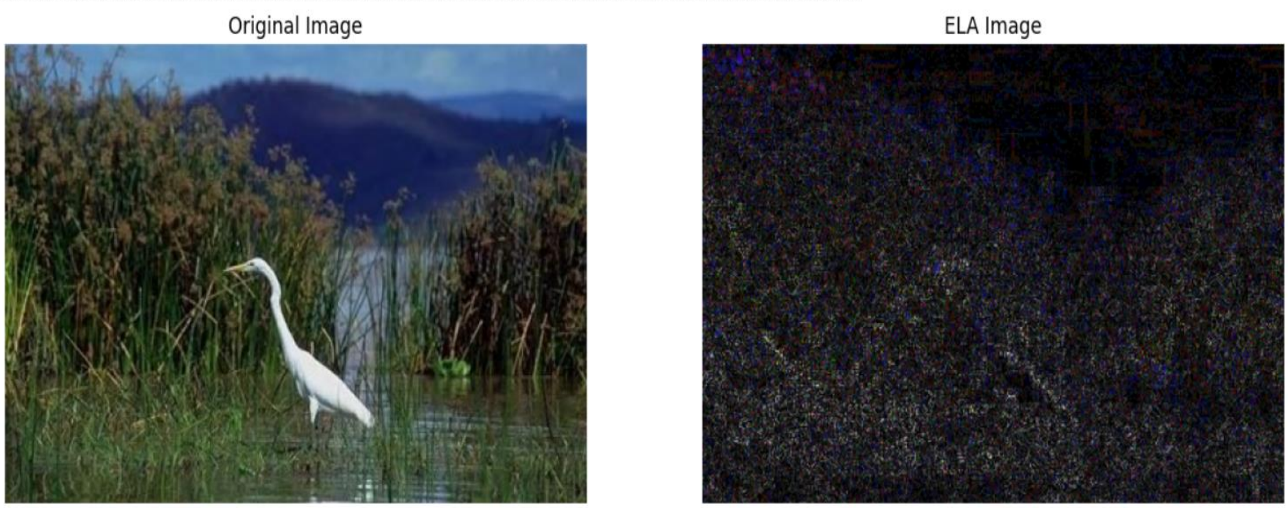

Confusion matrix
True label
Predicted label
0
1
0
1
935
59
144
907
900
800
700
600
500
400
300
200
100

```
1/1 [==============================] - 0s 79ms/step
Prediction: Forged
Confidence:  85.22%
----------------------------------------------------------------------------------------------------
```



True = Forged image

```
1/1 [==============================] - 0s 19ms/step
Prediction: Authentic
Confidence: 99.84%
----------------------------------------------------------------------------------------------------
```



True = Authentic image