Machine learning project

Image forgery detection

Maged Ahmed

Andrew Tharwat

Ziad Tamer

Under the supervision of:

Dr. Ghada Khoriba

Contents

**No table of contents entries found.**

# Problem Statement:

Capturing moments from our lives has become crucial in our time. Digital images are the prominent kind of images nowadays thanks mainly to digital cameras and smartphones.

Image forgery is an art dedicated to tampering with images in order to deceive the human eye and arrive at mostly malicious purposes. And therefore, detecting said forged images is important. Our aim with this project is to capture forged images with a reasonable accuracy using machine learning and neural networks techniques.

# Related work:

1. B. Bayar and M. C. Stamm, “A deep learning approach to universal image manipulation detection using a new convolutional layer”

In this paper, the authors proposed a new forgery detection method using 2 CNNs, with one of them constrained to only learn error filters. This allowed the model to automatically learn how to detect multiple image manipulations without relying on pre-selected features or any preprocessing.

Diagram, engineering drawing

Description automatically generated

This method resulted in an average 99.10% accuracy.

Table

Description automatically generated

1. Y. Zhang, J. Goh, L. L. Win, and V. L. Thing, “Image region forgery detection: A deep learning approach.”

In this paper, the authors propose a deep learning method to detecting tampered regions in an image. They first converted the image into YCrCb color space, then segmented it into 32x32, then applied a 3 Level 2D Daubechies Wavelet to each YCrCb component of the patches. Then obtained the standard deviation, mean, and the sum for each of the approximation, horizontal, vertical and diagonal coefficients to obtain 90 features. They then use a SAE model to learn the features. This method resulted in an accuracy of 91.09%

Table

Description automatically generated

1. R. Salloum, Y. Ren, and C.-C. J. Kuo, “Image splicing localization using a multi-task fully convolutional network (mfcn)”

In this paper, the authors propose a method to detect and localize image splicing forgery using a MFCN. They first extract the surface and edge probability maps, and then threshold them to yield the binary system output mask. They used stochastic gradient descent to train the model. Below are the f1 scores of the author’s methods on popular image forgery datasets.

Table

Description automatically generated

References:

1. B. Bayar and M. C. Stamm, “A deep learning approach to universal image manipulation detection using a new convolutional layer,” in Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security, pp. 5–10, ACM, 2016.
2. Y. Zhang, J. Goh, L. L. Win, and V. L. Thing, “Image region forgery detection: A deep learning approach.,” in SG-CRC, pp. 1–11, 2016.
3. R. Salloum, Y. Ren, and C.-C. J. Kuo, “Image splicing localization using a multi-task fully convolutional network (mfcn),” Journal of Visual Communication and Image Representation, vol. 51, pp. 201–209, 2018.

# Models Architecture:

## The first CNN model:

Convolution Layer:

This layer uses filters of increasing size from 32 in the first layer to 64 in the second and third layers. It use a kernel of size 3 x 3. It takes the image input in the shape 150x 150, and has activation function ReLu.

Pooling layer:

This layer uses max pooling with pool size 2 x 2.

Flatten layer:

This layer flattens the input into a one dimensional array.

Dense layer:

In this model we added 128 dense layers with activation function ReLu.

Dropout:

This attempts to prevent overfitting by randomly turning off some neurons during training.

Non-Linearity Layer:

Uses the sigmoid function.

Loss function:

Binary cross entropy.

Optimizer:

The CNN uses the Adam optimizer.

## The second CNN model:

Convolution Layer:

This layer uses filters of decresing size from 128 in the first layer to 64 in the second layer and 32 in the third layer. It use a kernel too is decreasing in size from 7x7 to 3x3. It takes the image input in the shape 512x512, and has activation function ReLu.

Pooling layer:

This layer uses max pooling with pool size 2 x 2.

Flatten layer:

This layer flattens the input into a one dimensional array.

Dense layer:

In this model we added 128 dense layers with activation function ReLu.

Dropout:

This attempts to prevent overfitting by randomly turning off some neurons during training.

Non-Linearity Layer:

Uses the sigmoid function.

Loss function:

Binary cross entropy.

Optimizer:

The CNN uses the Adam optimizer.

## 3-The first machine learning model (Mandeep)

After extracting features using LBP and DWT the features are passed to the model.

The training data is split in a 80/20 method.

This model uses a support vector classifier with Regularization parameter = 100, gamma = 0.01

and the rbf kernel.

## 4-The second machine learning model (Shilpa)

After extracting features using LBP and DCT the features are passed to the model.

The training data is split in a 80/20 method.

First there has to be hyperparameter tuning using GridSeachCV

Then the model uses a support vector classifier with Regularization parameter = 100, gamma =

0.01 and the rbf kernel.

# Evaluation results

## The first CNN model results with 15 million trainable parameters:

It gave 90% accuracy in the test after running 100 epochs and 150 steps per epoch with the built in validation data split. However when evaluating the model on new images it only gave 65% accuracy.

Confusion matrix:

[1228 78]

[ 802 403]

precision recall f1-score support

0 0.60 0.94 0.74 1306

1 0.84 0.33 0.48 1205

accuracy 0.65 2511

macro avg 0.72 0.64 0.61 2511

weighted avg 0.72 0.65 0.61 2511

## the second CNN model results:

It gave 80% accuracy in the test after running 100 epochs and 150 steps per epoch with the built in validation data split. However when evaluating the model on new images it only gave 60% accuracy.

Confusion matrix:

[[915 391]

[606 599]]

precision recall f1-score support

0 0.60 0.70 0.65 1306

1 0.61 0.50 0.55 1205

accuracy 0.60 2511

macro avg 0.60 0.60 0.60 2511

weighted avg 0.60 0.60 0.60 2511

## first ML model (Mandeep)

this model preformed the worse, by predicting almost all images as authentic it gives 52% accuracy, however when doing the train test split it gave 94% accuracy.

[[1306 0]

[1205 0]]

precision recall f1-score support

0 0.52 1.00 0.68 1306

1 0.00 0.00 0.00 1205

accuracy 0.52 2511

macro avg 0.26 0.50 0.34 2511

weighted avg 0.27 0.52 0.36 2511

[[4831 186]

[ 636 7375]]

precision recall f1-score support

0 0.88 0.96 0.92 5017

1 0.98 0.92 0.95 8011

accuracy 0.94 13028

macro avg 0.93 0.94 0.93 13028

weighted avg 0.94 0.94 0.94 13028

## the second ML model (Shilpa):

this model preformed slightly better than the other ML model giving 60% accuracy when testing new images and 95% accuracy with the train test split.

[[1282 24]

[ 969 236]]

precision recall f1-score support

0 0.57 0.98 0.72 1306

1 0.91 0.20 0.32 1205

accuracy 0.60 2511

macro avg 0.74 0.59 0.52 2511

weighted avg 0.73 0.60 0.53 2511

[[4756 261]

[ 363 7648]]

precision recall f1-score support

0 0.93 0.95 0.94 5017

1 0.97 0.95 0.96 8011

accuracy 0.95 13028

macro avg 0.95 0.95 0.95 13028

weighted avg 0.95 0.95 0.95 13028

in conclusion we can clearly see that deep learning models outperforms traditional machine learning with CNN being the best of them.