Image Forgery Classification : Tampering Detection

Final Project Report

Team ID - 17

AVISH SANTHOSH
ee20btech11007@iith.ac.in
DEVULAPALLI SAI PRACHODHAN
ee20btech11013@iith.ac.in
FASAL MOHAMED T
ee20btech11016@iith.ac.in
NANDYALA VISHWARAM REDDY
ee20btech11059@iith.ac.in

Abstract

Images represent an effective and natural communication medium for humans due to their immediacy and the ease with which image content can be understood. The widespread availability of image editing software tools makes it easier to alter image content or create new images. As a result, the possibility of tampering and counterfeiting visual content is no longer restricted to experts. This situation underscores the need for methods to verify the truthfulness of images and assess their quality. Answering these queries is relatively easy when the original image is known. However, in practical cases, almost no information can be assumed to be known a priori about the original image, making our task difficult. This paper explores all possible methods that can be applied to detect tampering forgery and its subtypes, such as Copy-move, splicing, and part-removal.

Index Terms – Deep learning, DCT, Block Matching, InceptionV3, ResNet, VGG16, Support Vector Machines(SVM), Linear Kernel, Random Forest Classifier, Feature Extraction, Gabor filters, Quantization.

0. Aim

Building off our previous report, we continue to examine various methods to detect copy-move forgery. We tried a DCT-based traditional method, following which we study various Deep Learning models

1. Introduction

As seen in previous reports, we have classified the passive image forgery detection techniques into **Traditional** and **Deep learning** based methods. Our general outline to detect copy-move forgery as discussed in the preliminary project report is as shown in the below figure.

According to [3], we have divided copy-move forgery detection techniques divided into three:

- **Block-based techniques**: In block-based methods, an image is split into overlapping blocks and then feature vectors are computed corresponding to each block. These vectors are matched to identify tampered images taking into account some empirical parameters.
- **Key-point based techniques**: In key-point-based methods, key-points are extracted and matched to detect image forgeries.
- **Deep Learning methods**: Instead of traditional feature extraction using block-based or key-point based we use Deep Learning models to detect features following which we train classifiers to detect image forgeries.

This report will cover a block-based method using the Discrete Cosine Transform (DCT), rounding off our analysis of traditional methods. We then proceed with forgery detection using deep learning models.

2. Build up on previous report

2.1. Our plan

After the submission of our MTR, we decided to additionally begin work on studying deep learning methods

apart from the traditional methods we had covered earlier. The four of us divided the work among us such that three of us focused on detecting using one deep learning model architecture for three different classifiers, while the fourth member focussed on the DCT-based approach. As part of that we read some additional papers that are focused on the above methods and finally ended with [14] and [16] to reproduce results.

2.2. Dataset

As said in the previous reports we have used this [9] to obtain our datasets. The dataset used can be downloaded here [17] (Small-image category database). We used the same dataset for both the DCT-based and the deep learning methods, creating a suitable split for training and validation in the latter.

2.3. Update on Key-point based Method

Using the same algorithm and improving the key-point to spatial coordinates conversion the running time has somewhat improved over the previous iteration but the accuracy remains the same. We tried using another implementation of SIFT [15] but this lead to a worse run-time since the number of features obtained were significantly higher leading to greater computation costs; it produced only a slightly better accuracy. Here are our key-takeaways:-

- The SIFT implementation from OpenCV the nfeatures parameter can be explicitly mentioned which sometimes catches the features from the forged part and sometimes does not.
- The other SIFT implementation is too intensive; it finds way too many features leading to more memory consumption and causing MLE.
- We think the SIFT method should be modified or aided with some other function so as to help us decide which key-points are relevant to our use.

So, the best-case scenario would be to find an optimal range/method such that it can find the features which can match both the original and the tampered part. But the problem is that this function would heavily be dependent on the type of image. In conclusion to make this key-point based method successful one should implement a SIFT algorithm from scratch so that it is tailored to this use case.

2.4. Update on Gabor Filters based Method

We have tried to improve the performance of the algorithm discussed in MTR using Gabor filters. Here are our changes that we have made:

- Previous implementation of Rotation invariant filters is not correct and we have read the paper [7] proposed in the reference paper [13] for the correct implementation of the same. we updated our code to increase the size of the filters.
- We tried changing the parameters like Distance threshold (D) and N_f, N_d and observed that with the other combinations, the accuracy suffers greatly.Finally we are continuing with the same parameters with change in the implementation of Gabor filters.

3. DCT-Based Method

3.1. General Overview

One of the main challenges associated with these methods is the significant amount of time required to detect matching blocks. Additionally, determining the appropriate thresholds for such algorithms can be a complex and time-consuming process, often requiring trial-and-error or expert knowledge. These thresholds are also influenced by the texture and other characteristics of the image being analysed, further complicating the detection process.

3.1.1 Idea

The proposed optimisations for a block-matching algorithm is to reduce the number of instances, the dimension of the feature vector, and improve the block matching algorithm. To achieve this, the authors suggest using DCT to represent the image in the frequency domain. This is because DCT can capture most of the intensity distribution details with fewer coefficients, which helps to reduce the feature vector dimension. The authors have demonstrated that this method improves the execution time without compromising the algorithm's robustness against post-processing operations such as noise addition and JPEG compression.

3.1.2 DCT

In DCT coding, each component of the image is subdivided into blocks of b x b pixels. A 2D DCT is applied to each block of data to obtain an b x b array of coefficients. If I[x,y] represents the image pixel values in a block, then the DCT coefficients D[u,v] are computed for each block of the image data as follows:

$$D[u,v] = \frac{2}{b}C[u]C[v]\sum_{x=0}^{b-1}\sum_{y=0}^{b-1} I[x,y]\cos\left(\frac{\pi u(2x+1)}{2b}\right)\cos\left(\frac{\pi v(2y+1)}{2b}\right) \quad (1)$$

where $0 \leq u, v \leq b-1$ and $C[u] = \sqrt{\frac{1}{2}}$ if $u = 0$ and 1 if $1 \leq u \leq b-1$.

An alternative dimensionality reduction technique is

the PCA, which however does not produce the same robustness as the DCT, simply because the DCT procedure is designed with great correlation to the human visual perception.

4. Algorithm and Analysis

4.1. Explanation of Algorithm

The proposed algorithm works as follows:

1. Convert the input image to grayscale, if it is not already in that format.
2. Define a fixed-sized window of dimensions b x b.
3. Slide the window by one pixel from the upper left corner of the image to the bottom right corner, overlapping each time, and extract the block of pixels covered by the window.
4. Apply a 2D Discrete Cosine Transform (DCT) to the block, quantize the coefficients, and order them in zigzag order to create a row vector of length p x b x b.
5. Repeat steps 3 and 4 for all possible window positions in the image to obtain a matrix A of dimensions (m-b+1) x (n-b+1) x p x b x b, where m and n are the dimensions of the image.
6. Sort the vectors in each row of A lexicographically.
7. For each row ai in A, compare it with its neighbouring rows a_j (where j¿i) such that the first q quantized DCT coefficients of a_i and a_j are the same.
8. If the distance between the patches from which a_i and a_j originated is greater than the block size b, calculate the shift vector s = $(i_1 - j_1, i_2 - j_2)$, where (i_1, j_1) and (i_2, j_2) are the original block coordinates.
9. Increment the count for s.
10. The shift vector with the highest count is taken as the threshold frequency, which represents the presence of copy-move regions in the forged image. The threshold frequency should be set appropriately for multiple copy-move parts, and it should be greater than b x b to represent significant duplication.

4.1.1 Analysis of the Algorithm

The proposed method removes limitations of popular block matching algorithms by modifying the structure of the matching algorithm. The following improvements are noteworthy:

One optimization involves only comparing the first q coefficients of the feature vector array in the matching step, as these coefficients represent the major intensity distribution of the block and are highly correlated for copy moved regions. This reduces the number of vector comparisons required and improves the algorithm's efficiency.

Additionally, the proposed method eliminates the need for manually setting certain thresholds. For example, the threshold for the minimum distance between matching blocks (N_d) is set to the size of the block (b) to avoid erroneously tagging overlapping blocks as similar. The threshold count for valid shifts (N_f) is also difficult to set prior to applying the algorithm to an arbitrary image, but it can be logically set to the size of the block ($b \times b$) in the event of a perfect match between two image patches. These modifications improve the robustness and efficiency of the algorithm.

4.2. Results

In an ideal case, the code should be made to run for O(n^2) time complexity in step 7 of the algorithm, where n is the number of vectors obtained after flattening; however, since it is too time-consuming, we have decided to compare only the next 10 vectors in the vector matrix. We have set the parameters p = 0.1 and q = 3, number of images = 40; it's quite possible that we get better results for larger values of p and q. We have tested our results on dataset [17] and we got **Accuracy of 60**%.

The authors have mentioned that this algorithm does not work well with noisy and compressed images, which abound in the dataset chosen. This is one of the major drawbacks for less accuracy.

As observed here and in the MTR, the traditional methods are very slow and give less accuracy; we thus make the jump to deep learning methods to work on the forgery detection.

5. Deep Learning Methods

Deep learning methods have shown great success in detecting copy-move forgery in digital images. By using artificial neural networks, such as convolutional neural networks (CNNs), deep learning methods can learn to recognize patterns and features in image data that are indicative of copy-move forgeries. This is achieved by training the network on a large dataset of labeled images that include both authentic and forged images. Once trained, the network can accurately classify new, unlabeled images as authentic or forged by analyzing their features and patterns. Deep learning methods have achieved state-of-the-art performance in copy-move forgery detection and are a valuable tool in digital image forensics.

5.1. Methodology

As part of this report, we are using the below steps for detection:

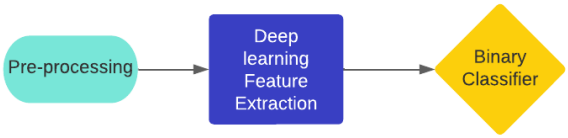

Figure 1. General flowchart of deep learning methods

- As we have very less time and less data available to train a Deep learning model from scratch for feature extraction, we have used pre-trained models available in tensorflow with the help of blogs like [5] and [12].
- The above fraction of feature vectors are passed through ML classifier where classifier gets trained.
- The remaining portion from the above feature vectors are used for validation. Using scikit-learn [1] we calculate the accuracy score which is our metric (as mentioned previously).

5.2. Feature Extraction

As said earlier, we picked three pre-trained models as per their popularity in the research community, namely, InceptionV3, ResNet50 and VGG16. Brief explanation of the above are given below:

5.2.1 VGG16

VGG16 (Visual Geometry Group) was firstly proposed in the paper [11] by Karen Simonyan Andrew Zisserman. It is an image recognition deep learning network architecture based on Convolutional Neural Networks (CNNs) and has been trained on a subset of the ImageNet dataset which consists of a collection of 14 million images belonging to 22,000 different categories.

High level overview of proposed architecture

- VGG16 has 21 layers in which 13 are convolutional layers, 5 Max Pooling layers and three dense layers. But it has 16 weighted layers i.e. 16 out of above layers having parameters that have to be learnt.
- It takes input size of image as 224x224x3 so we have used tf.keras.preprocessing.image to preprocess the image and the given RGB image irrespective of size converted to image of size 224x224x3 using bilinear interpolation.
- The VGG network consists of a series of convolutional layers, each followed by a non-linear activation function, such as the Rectified Linear Unit (ReLU), and then max pooling layers. The convolutional layers use small 3x3 filters with a stride of 1 pixel and same padding to preserve the spatial dimensions of the input.
- The architecture is characterized by its simplicity and uniformity, with all convolutional layers having the same filter size and the same number of filters.

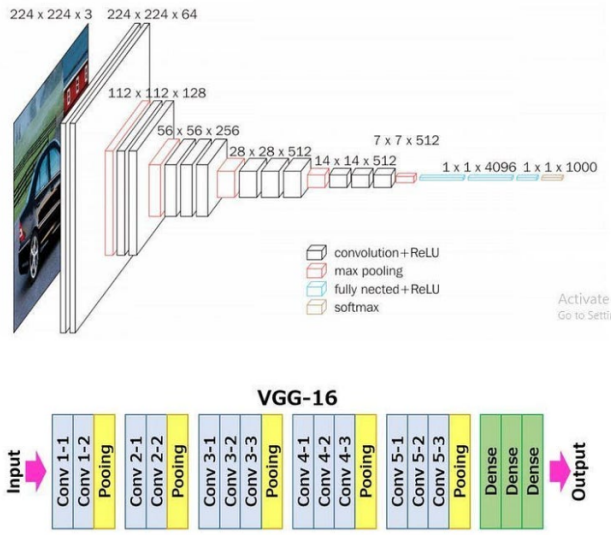

Figure 2. Image showing architecture of VGG16 from [6]

5.2.2 ResNet50

Residual Network also called ResNet was proposed by Microsoft research team in 2015 [10]. It was a CNN architecture to solve the issue of vanishing/exploding gradient by introducing a concept called “Residual Blocks”. With ResNets, the gradients can flow directly through the skip connections backwards from later layers to initial filters. The pre-trained version of it was trained on a subset of ImageNet dataset consisting of more than million images.

High level overview of proposed architecture

- One of the major positives of this network from VGG, is Skip connections which will be responsible for solving vanishing/exploding gradient even though we are increasing number of layers.
- According to [10], the above is the architecture overview along with the skip connections.
- It has 5 classes of convolutional layers which accounts for 49 layers along with softmax layer at the end resulting in 50 layers.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 3. Proposed ResNet50 architecture from [10]

5.2.3 InceptionV3

InceptionV3 was proposed in [4] by Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens. It is a convolutional neural network architecture with many added improvements like factorized 7 x 7 convolutions, Efficient Grid Size Reduction and the use of an auxiliary classifier along with the use of batch normalization for layers in the sidehead.

High level overview of proposed architecture

type	patch size/stride or remarks	input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

Figure 4. Proposed InceptionV3 architecture from [4]

- It is a pre-trained convolutional neural network that is 42 layers.
- Instead of deep convolutional layers, it uses multiple filters of different sizes which results in reduced overfitting of the data. This Inception model consists of multiple Inception modules and each module is made up of four parallel layers (1×1 convolution, 3×3 convolution, 5×5 convolution, 3×3 max pooling).
- According to [4] the proposed architecture is given left.

All the above pre-trained models are obtained from [2]. We have also applied preprocessing from tensorflow.keras to make our image data compatible with the corresponding neural network architectures.

5.3. Classification

After feature extraction, we have to classify the image as forged or real. We have used Machine learning Classifiers and trained them to detect forgery. As we have limited data roughly around 10,000 images in our dataset we haven't opted for Deep learning based classifiers as they have many parameters to find and thus require more data to get accurate results. We used **Random Forest Classifiers** and **Support Vector Classifiers (SVC)** to obtain our results.

We have picked SVC because Support Vector Algorithms generally perform well when less data and more features available in comparision with other classifiers. We also used Random Forests in case if SVC underfits data then Random Forests will generally fit well.

So we are finally ended up with performance analysis of three different feature extraction architectures along with two classifiers making 6 possible combinations.

5.4. Results

We have tested each of the above combination on CO-FOMOD dataset [17]. For testing, We have done the below steps:

- First we have extracted feature vectors for all images in that dataset and stored in array. Now we have splitted the obtained array into testing set and training set in some fraction (0.2 in our case).
- We trained our classifiers on training set and then tested on test set giving us **"validation accuracy"**

Feature Extractor	SVC	Random Forests
VGG16	98.5%	96.70%
ResNet50	98.8%	96.25%
InceptionV3	52.45%	71.65%

Table 1. Table consisting of validation accuracy for different classifier and feature extractor combination on [17].

We have considered SVC with **linear kernel** and all the parameters for both classifiers are set to default. We weren't left with much time to fine tune the hyperparameters; if we were to tune them then we might have achieved higher accuracies.

6. DeepLearning vs Traditional Observations

In the duration of this project we have explored both the traditional methods(as classified at the beginning) and the more modern Deeplearning models for Copy Move Forgery Detection. The DL model method's accuracy is much better than that of traditional methods (atleast as compared to our implementation of traditional methods). This higher accuracy might be due to the fact that neural networks generally combine more techniques(like edge detections/convolutions etc) and the huge training data makes neural nets a definitely better choice in this scenario.

Method	Accuracy
ResNet50 with SVC	98.8%
SIFT key-points based method	62%
Gabor filters based method	60%
DCT based method	60%

Table 2. Traditional vs Deep Learning methods on [17].

Note: We have tested SIFT based method on [8].

7. Individual Contributions

- Avish Santhosh worked on InceptionV3 based approach.
- Devulapalli Sai Prachodhan worked on ResNet50 and Gabor filters based approaches.
- Fasal Mohamed worked on DCT based approach.
- Vishwaram Reddy worked on VGG16 and SIFT key-points based methods.

References

[1] Documentation of "scikit-learn". 4

[2] Documentation of "tensorflow". 5

[3] Soumen Ba Anuja Dixit. A fast technique to detect copy-move image forgery with reflection and non-affine transformation attacks. 1

[4] Sergey Ioffe Jonathon Shlens Christian Szegedy, Vincent Vanhoucke. "rethinking the inception architecture for computer vision". 5

[5] franky. Blog on medium titled as "using keras' pre-trained models for feature extraction in image clustering". 4

[6] Rohini G. Blog on medium titled as "everything you need to know about vgg16". 4

[7] Ju Han and Kai-Kuang Ma. "rotation-invariant and scale-invariant gabor features for texture image retrieval," in elsevier, image and vision computing 25, pp. 1474–1481, 2007. 2

[8] Roberto Caldelli Alberto Del Bimbo-Giuseppe Serra Irene Amerini, Lamberto Ballan. A sift-based forensic method for copy-move attack detection and transformation recovery. 6

[9] Chao Zhang Jingjing Chen Yu-Gang Jiang Larry S. Davis Junke Wang, Zhenxin Li. Fighting malicious media data: A survey on tampering detection and deepfake detection. 2

[10] Shaoqing Ren Jian Sun Kaiming He, Xiangyu Zhang. Deep residual learning for image recognition. 4, 5

[11] Andrew Zisserman Karen Simonyan. Very deep convolutional networks for large-scale recognition. 4

[12] mrgrhn. Blog on ai.plainenglish.io titled as "vggnet with tensorflow (transfer learning with vgg16 included)". 4

[13] Manju Manuel Raichel Philip Yohannan. Detection of copy-move forgery based on gabor filter. 2

[14] Nandan Kanvinde Pandharinath Ghonge Raunak Joshi, Abhishek Gupta. Forged image detection using sota image classification deep learning methods for image forensics with error level analysis. 2

[15] rmislam. SIFT Implementation from Github. 2

[16] Shaktidev Mukherjee Sunil Kumar, Jagannath Desai. "a fast dct based method for copy move forgery detection". 2

[17] Grgic S. Grgic M. Tralic D., Zupancic I. "comofod - new database for copy-move forgery detection", in proc. 55th international symposium elmar-2013, pp. 49-54, september 2013. 2, 3, 6