Medicon Engineering Themes
Volume 8 Issue 2 February 2025
Article Type: Research Article
ISSN: 2834-7218

Image Manipulation Detection using Augmentation and Convolutional Neural Networks

Annant Maheshwari*, Rishi Jain, Ritika Mahapatra, Saagar Palakuru and Anand Kumar M

Dept. of Information Technology, National Institute of Technology Karnataka

***Corresponding Author:** Annant Maheshwari, Dept. of Information Technology, National Institute of Technology Karnataka.

Received: January 08, 2025; **Published:** February 04, 2025

Abstract

Image tampering is now simpler than ever thanks to the explosion of digital photos and the creation of easy image modification tools. As a result, if the situation is not handled properly, major problems may arise. Many computer vision and deep learning strategies have been put out over the years to address the problem. Having said that, people can easily recognise the photographs that were used in that research. This begs the key question of how CNNs might do on more difficult samples. In this paper, we build a complex CNN network and use various machine learning algorithms to classify the images and compare the accuracies obtained by them. Its performance is also compared on two different datasets. Additionally, we assess the impact of various hyperparameters and a data augmentation strategy on classification performance. This leads to a conclusion that performance can be considerably impacted by dataset difficulty.

Keywords: Image Manipulation; CNNs; Data Augmentation; Patch Extraction

Introduction

The process of modifying or manipulating digital photographs using different software tools and methodologies is known as image manipulation. The availability of more tools and software for picture editing, as well as the ease with which photographs may be shared online, has made image alteration more accessible than ever before. It is employed in a number of industries, including forensic science, fashion, advertising, and art. However, it can also be employed maliciously, such as when photographs are edited for use in deceit, fraud, or political propaganda.

Following are the different types of image manipulation techniques that exist:

- *Copy-move*: This method entails copying a section of an image and pasting it onto a different section of the same image. This can be done to duplicate or conceal a person or item inside of a picture. Comparisons between two or more parts of an image can be used to spot copy-move manipulation.
- *Splicing*: In order to form a single composite image, two or more images are combined during splicing. This can be used to change the appearance of individuals or objects in an image. By examining the image's colour and texture differences, as well as spotting discrepancies in the lighting and shadows of the combined items, splicing can be identified.
- *Retouching*: Retouching is a technique used to enhance or modify an image by removing or adding elements, smoothing skin, or adjusting colours. Retouching can be used to improve the appearance of an image, or to misrepresent information in a way that is not accurate. Retouching can be detected by analysing the image's sharpness and texture.
- *Morphing*: By using a sequence of intermediary steps, morphing entails changing one image into another. Movie visual effects or animations can be produced using this. By examining the image's pixel changes over time, morphing can be detected.

- *Enhancement*: By modifying an image's brightness, contrast, sharpness, or other visual attributes, one can enhance the quality of the image. Enhancement can be done manually or automatically using image processing software. By examining the visual characteristics of the image, such as contrast and sharpness, to spot artificial or exaggerated alterations, enhancement can be identified.

We will be focusing on splicing and copy-move image forgery recognition as those are the most common forms of image manipulation. We will be using various techniques to hone our model specifically on the patches which have been manipulated by using the masks of the image. We will also by employing various strategies to augment our dataset. Another key change from the norm is that we will be using an ML model after the CNN layers to classify our data instead of the usual approach of using a final dense layer within the CNN architecture with a softmax or sigmoid function.

Literature Survey

The authors in [1] demonstrate an effective CNN model for detecting copy-move image forgeries, which occurs when a portion of an image is copied, moved, and repositioned to hide or duplicate information. The proposed model uses convolutional layers and pooling layers to extract features from the input image and achieve higher accuracy, precision, and recall rates compared to other state-of-the-art methods. The model is evaluated on a new dataset called CMFD2022, containing 4000 authentic and 4000 forged images. The proposed model can be used for forensic analysis, copyright protection, and tamper detection in digital images.

The authors in [2] propose a new architecture for detecting different types of techniques of image forgery. The proposed approach involves pre-processing the input image, extracting features using a convolutional neural network, and then classifying the image as forged or genuine using a fully connected neural network. The authors have evaluated the proposed method on a publicly available dataset and achieved promising results with high accuracy and low false positive rate. Overall, the paper presents a promising approach for detecting image forgeries using deep neural networks.

The paper [3] addresses the problem of detecting image splicing tampering, which is a common technique used to manipulate digital images. The proposed method combines deep learning and attention mechanisms to automatically detect splicing tampering in images. The method is used on two datasets, CASIA v2.0 and Columbia, which contain various types of splicing tampering. The proposed method has potential applications in forensics, security, and journalism for detecting image manipulation.

The authors in paper [4] start by providing an introduction to image forgery detection and its importance in the current digital world. They then present a detailed survey of recent research work in this field, including the types of image forgeries, the challenges in detecting them, and the various deep learning techniques used for detection. The paper concludes with a discussion on the future directions of research in image forgery detection using deep learning, highlighting the need for the development of more robust and efficient methods to handle different types of image forgeries and the challenges associated with them.

Datasets Used

We have used both CASIA v2.0 and N16 Datasets which are widely used datasets for image manipulation detection and have compared the performance and accuracy for both the datasets.

CASIA v2.0 dataset contains 7,000 untampered and 5,000 manipulated images with a resolution of 512×512 pixels. The manipulated images are have had copymove, splicing, and retouching performed on them. This dataset also provides ground truth information indicating the types of manipulation applied to each image.

NC16 dataset contains images of size 256×256 pixels and it has 1400 authentic images and 600 manipulated images. The images are manipulated using five different types of operations, namely copy-move, splicing, removal, resizing, and recoloring. This dataset also includes the original images from which the manipulated images are created, allowing for the creation of paired authentic-manipulated image sets.

We have also augmented the dataset using image rotations. Usually for image manipulation detection, rotating the image gives us a lot more info on whether the image is actually manipulated or not. Hence, after we extract the manipulated patches from the image, we perform rotations on these patches to increase the size of our dataset and our results show that this augmentation very clearly benefits the accuracy of the model.

Methodology

The running of the entire program has been split into 3 main parts. These include:

- Preprocessing and augmentation.
- Feature extraction and feature fusion.
- Classification using machine learning.

Each of these steps have been explained in detail below.

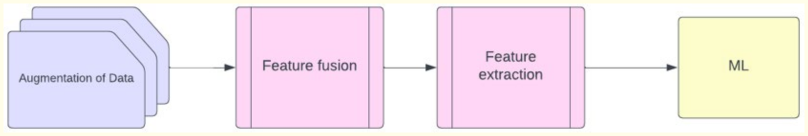

Figure 1: Flow chart of the process for the program.

Preprocessing and Augmentation

This part of the code is used to extract image patches from a dataset of tampered and authentic images, specifically from the CASIA 2.0 dataset. The code extracts 2 patches for the tampered image used for training a machine learning model to detect tampered images while the authentic image patches are used as negative examples.

The extraction process involves dividing each image into non-overlapping patches of size 128x128 pixels with a stride of 8 pixels. For each patch, the code applies 4 rotations at 90-degree intervals to augment the dataset. The patches are saved to disk in separate directories for tampered and authentic images.

To extract the authentic image patches, the code uses the names of the tampered and authentic images to match them. It first reads the authentic images and then extracts patches from them in the same way as for tampered images.

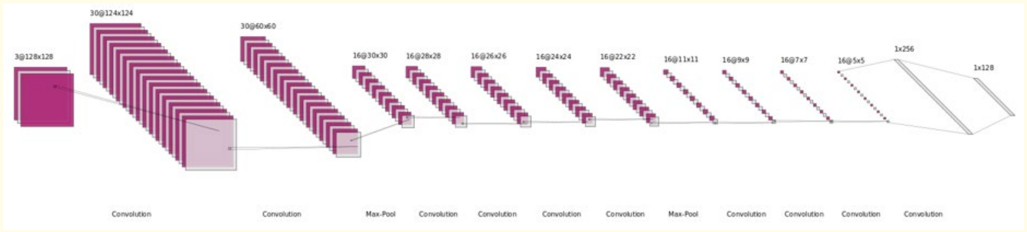

Figure 2: CNN Model.

Feature extraction and Feature Fusion

Feature extraction

Now feature extraction is performed for images using a convolutional neural network (CNN) model which we pre-train on our dataset. It has two functions to extract features from two different datasets one is called CASIA2 dataset and the other is called NC2016 dataset. The features are extracted from image patches using the pre-trained CNN model and written to a CSV file. The CSV file contains columns for image names, labels, and the extracted features. The extracted features are represented as a vector of 400 values.

The input to the CNN is a 3-channel image patch. Let's look at each layer in detail:

- *Convolutional Layer 1*: It has 3 input channels and 3 output channels. It uses a 5x5 kernel with a stride of 1 and no padding. The weights are initialized using Xavier initialization.
- *Convolutional Layer 2*: It has 3 input channels and 30 output channels. It uses a 5x5 kernel with a stride of 2 and no padding. The weights are initialized using the learnable Square Receptive Field Matrix (SRM) filters for the second convolutional layer.
- *Local Response Normalization*: After the second convolutional layer, a Local Response Normalization (LRN) layer is applied to the output of the last layer. The purpose of the LRN layer is to normalize the output and improve the performance of the network. The LRN layer has a size of 3.
- *Pooling Layer 1*: It performs max pooling using a 2x2 kernel and a stride of 2. This results in the reduction of the spatial dimensions by half.
- *Convolutional Layers 3-4*: The third and fourth convolutional layers have 30 input channels and 16 output channels each. They use a 3x3 kernel with a stride of 1 and no padding. The weights are initialized using Xavier initialization.
- *Convolutional Layers 5-8*: The fifth through eighth convolutional layers each have 16 input channels and 16 output channels. They use a 3x3 kernel with a stride of 1 and no padding. The weights are initialized using Xavier initialization. After the fifth and seventh convolutional layers, another LRN layer is applied.
- *Pooling Layer 2*: The second pooling layer is a max pooling layer with a kernel size of 2x2 and a stride of 2. It reduces the spatial dimensions of the output by a factor of 2.
- *Fully Connected Layer*: The output of the last convolutional layer is flattened and fed into a fully connected layer with 16x5x5=400 input neurons and 2 output neurons. The weights are initialized using Xavier initialization. In the training phase, a dropout layer with p=0.5 is applied before the fully connected layer.
- *Activation Functions*: In all the convolutional layers, the rectified linear unit (ReLU) activation function is used. The sigmoid function is used in the output layer during training to produce class probabilities.

Feature Fusion

Now that we have these 400 features from every patch obtained from an image, and there are 8 such patches for every image, we fuse these features into one by using either max or mean of the values for every feature of every patch. Now after this step, each image is represented as 400 dimensional vector and these are the final features that will be passed to various machine learning models to classify it into the two required classes.

Classification

This is the most crucial step in the working of this program as we use various machine learning algorithms for the final classification step instead of the usual CNN dense layer output. We take these 400 dimensional vectors obtained in the feature extraction step and pass them to the machine learning models for them to classify the data. The different models compared are:

- SVM.
- Gradient Boost.
- Naive Bayes.

- Random Forest.

We implement a pipeline for optimizing the hyperparameters of the classifier, classifying feature vectors using the optimized model, and analyzing the classification performance. The pipeline includes functions for hyperparameter optimization, classification, printing the confusion matrix, and finding misclassified samples.

The first function, optimizes the hyperparameters of the model by performing crossvalidation with the training data. It takes the feature vectors, labels, and a grid of possible hyperparameters as inputs, and returns the optimal hyperparameters. The second function, takes the feature vectors, labels, and optimal hyperparameters as inputs, and uses the optimized model to classify the feature vectors.

Finally, the misclassified samples are found using the optimized model. This pipeline provides a structured approach to training and evaluating a machine learning model, and can be easily modified for different models or datasets. By using cross-validation to optimize the hyperparameters, the pipeline ensures that the model is not overfitting to the training data, and can generalize well to new data. The confusion matrix provides a detailed analysis of the classification performance, including false positives, false negatives, true positives, and true negatives. The pipeline can be used to identify misclassified samples, which can be further analyzed to identify sources of error and improve the model.

Result Analysis

Accuracy over epochs graph is a plot of the performance of a machine learning model over time, where each epoch represents one complete iteration through the training data. The accuracy metric measures the percentage of correctly classified instances in the dataset. Fig 3. represents the accuracy during the CNN training for both the augmented and non augmented CASIA v2.0 Datasets. The data augmentation techniques used during training have led to an improvement in the model's performance and therefore the augmented CASIA v2.0 dataset has a better accuracy.

Fig 4. represents the accuracy during the CNN training for both the augmented and non augmented N16 Datasets. The data augmentation techniques used during training have led to an improvement in the model's performance and therefore the augmented represents the cross-entropy loss per epoch during the CNN training for the two datasets for both the augmented and non augmented CASIA v2.0 Datasets. The data augmentation techniques used during training have led to an improvement in the model's performance and therefore the augmented CASIA v2.0 dataset has a lower loss.

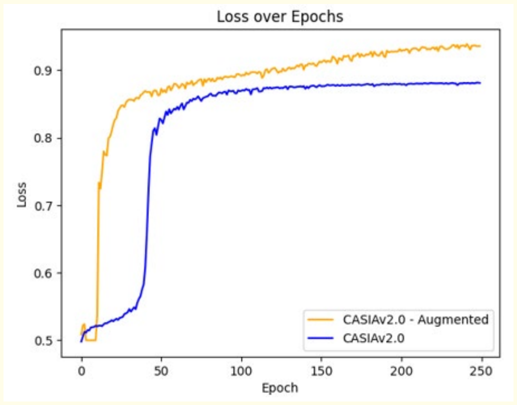

Figure 3: The accuracy per epoch during the CNN training for the CASIA v 2.0 Dataset N16 has a better accuracy.

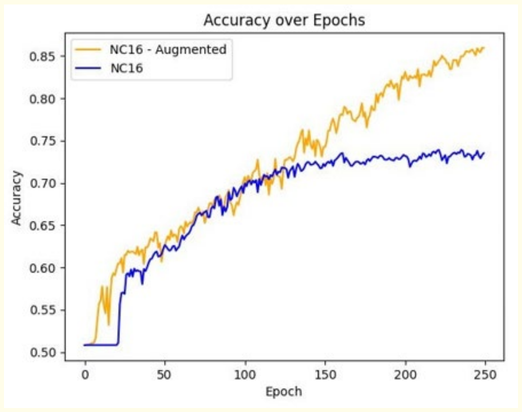

Figure 4: The accuracy per epoch during the CNN training for the N16 Dataset Fig 5.

Fig 6. represents the cross-entropy loss per epoch during the CNN training for the two datasets for both the augmented and non augmented NC16 Datasets. The data augmentation techniques used during training have led to an improvement in the model's performance and therefore the augmented NC16 Dataset has a lower loss.

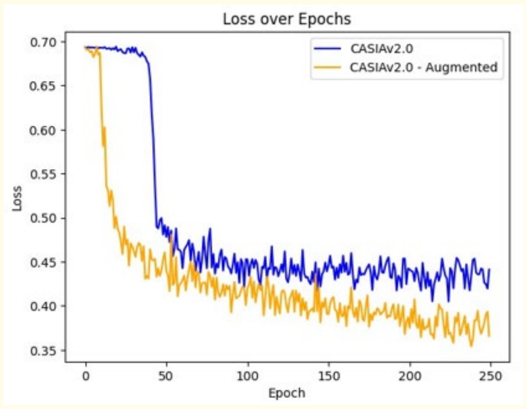

Figure 5: The cross-entropy loss per epoch during the CNN training for the CASIA v 2.0 Dataset.

Table 1 and table 2 provide a comparison of the accuracy values for non-augmented and augmented dataset for four different classification models: Random Forest, Support Vector Machine (SVM), Gradient Boosting, and Naive Bayes, for CASIAv2 and NC16 respectively. The accuracy values indicate the percentage of correctly classified instances in the test data, while the loss values indicate how well the model fits the training data.

Based on this table, we can see that Random Forest and SVM perform similarly well, achieving accuracy values of 96.86 and 96.76 respectively. Gradient Boosting and Naive Bayes have slightly lower accuracy values of 95.43 and 96.1 respectively.

The accuracies we got while using a dense layer with a sigmoid activation function for the last layer of the CNN during the training phase gave us an output of 93.28% Table 3 provides a comparison between the accuracy we obtained with our best classifier with each of the dataset and the accuracies for current state of the art image manipulation techniques.

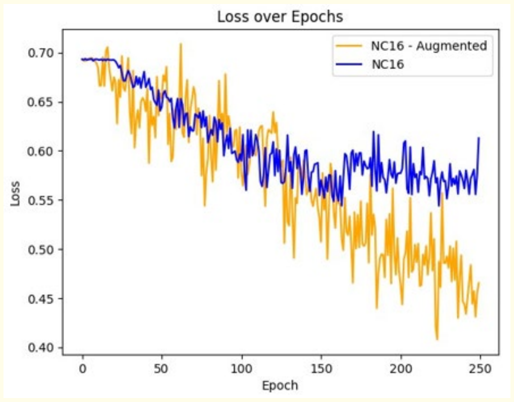

Figure 6: The cross-entropy loss per epoch during the CNN training for the N16 Dataset.

Model	***CASIA***	***Augmented CASIA***
Random Forest	92.23%	96.86%
SVM	91.78%	96.76%
Gradient Boosting	91.5%	95.43%
Naive Bayes	90.95%	96.1%

Table 1: Accuracies for CASIAv2.

Model	***NC16***	***Augmented NC16***
Random Forest	90.56%	94.23%
SVM	83.52%	84.89%
Gradient Boosting	82.78%	84.43%
Naive Bayes	81.98%	83.52%

Table 2: Accuracies for NC16.

Dataset	***State of the art***	***Our Accuracy***
CASIAv2	96.61% [7]	96.86%
NC2016	93.81% [9]	94.23%

Table 3: Accuracy comparison with state-of-the-art.

Conclusion and Future Scope

In conclusion, we have implemented an approach for image classification that uses various techniques like dataset augmentation which is done by rotating the images of the dataset by different angles to better capture the variations within the images. We also extract patches from the image instead of using the entire image as this allows the CNN to learn based on only the patches of the image that have been manipulated hence making sure it learns the most useful features.

We then implement the CNN model by borrowing from various previous research papers that have already established good models for image forgery detection. Then, we extract 400 dimensional vectors from the CNN and use various ML models to classify this data into manipulated images or not.

We have obtained an accuracy higher than the current state of the art techniques using CNN for feature extraction and a random forest model for classification. This could be improved further by experimenting with a different CNN architecture and using other models for classification. Pretrained models trained on the ImageNet dataset [10] could also be considered.

References

1. KM Hosny., et al. “An Efficient CNN Model to Detect Copy-Move Image Forgery”. in IEEE Access 10 (2022): 48622-48632.
2. A Singh and J Singh. “Image forgery detection using Deep Neural Network”. 2021 8th International Conference on Signal Processing and Integrated Networks (SPIN), Noida, India (2021): 504-509.
3. C Wang, Y Li and G Wu. “Image Splicing Tamper Detection Based on Deep Learning and Attention Mechanism”. 2021 IEEE 6th International Conference on Signal and Image Processing (ICSIP), Nanjing, China (2021): 267-271.
4. ZJ Barad and MM Goswami. “Image Forgery Detection using Deep Learning: A Survey”. 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India (2020): 571-576.
5. S Dhivya, B Sudhakar and K Devarajan. “2-level DWT based copy move forgery detection with surf features”. Proc. 3rd Int. Conf. Commun. Electron. Syst. (ICCES) (2018): 800-805.
6. B Wen., et al. “Coverage a novel database for copy-move forgery detection”. 2016 IEEE International Conference on Image Processing (ICIP) (2016): 161-165.
7. Ashutosh Pandey, Pritee Parwekar and Ashish Kumar Chakraverti. “A Review on Deep Learning Techniques for Image Forgery Detection”. 2022 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS) (2022): 890-895.
8. J Dong, W Wang and T Tan. “CASIA Image Tampering Detection Evaluation Database”. 2013 IEEE China Summit and International Conference on Signal and Information Processing, Beijing, China (2013): 422-426.
9. Hussain Muhammad., et al. “Evaluation of Image Forgery Detection Using Multi-Scale Weber Local Descriptors”. International Journal on Artificial Intelligence Tools 24 (2015).
10. J Deng., et al. “ImageNet: A large-scale hierarchical image database”. 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA (2009): 248-255.

Volume 8 Issue 2 February 2025