



iJRASET

International Journal For Research in
Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 Issue: III Month of publication: March 2025

DOI: <https://doi.org/10.22214/ijraset.2025.67285>

www.ijraset.com

Call:  08813907089

E-mail ID: ijraset@gmail.com

Digital Image Forgery Detection Using Convolutional Neural Network

Kandukuri Nirosha¹, Dr. D. Vivekananda Reddy²

Sri Venkateswara College Of Engineering, India

Abstract: Digital images are a main source of shared information in social media. Digital image forgery has become a growing concern with the advancement of image editing tools, leading to the spread of misleading and manipulated content. Detecting such forgeries is crucial for ensuring the authenticity and reliability of digital images. Various digital image forgery detection techniques are tied to detecting only one type of forgery, such as image splicing or copy-move it is not applied in real life. To enhance digital image forgery detection using deep learning techniques via transfer learning is uncover two types of image forgery at the same time.

This study proposes a deep learning-based approach utilizing Convolutional Neural Networks (CNNs) to detect forged images effectively. The proposed model is trained to identify common types of image forgeries, including copy-move, splicing, and retouching. By leveraging CNNs, the model extracts spatial and texture-based features to distinguish forged regions from authentic ones.

The dataset used for training and evaluation consists of both real and manipulated images, ensuring robust performance across different types of forgeries.

Experimental results demonstrate that the proposed CNN-based model achieves high accuracy in forgery detection compared to traditional image processing methods. This research highlights the potential of deep learning techniques in forensic image analysis and contributes to the advancement of automated forgery detection systems.

I. INTRODUCTION

Digital image forgery is alteration or manipulation of a digital image to mislead viewers. Common techniques include copy-move, splicing and retouching. Manual methods struggle with complex and sophisticated forgery techniques. Traditional forgery detection dependence on manual feature extraction, limited capability to handle complex manipulations, Time-Consuming and less scalable for real-world application.

CNN's automatically extract and learn features from image data. High accuracy in identifying subtle inconsistencies. Scalability and robustness across different forgery types. Adaptability to different datasets and forgery styles.

CNN's are a type of deep learning model particularly well-suited for image-related tasks. They automatically learn features from raw data, making them highly effective in detecting subtle inconsistencies caused by forgery. Input Layer: Accepts the input image, typically represented as a matrix of pixel values.

The image is processed in its raw form or after re-sizing for uniformity. Conventional Layers: Detect features like edges, textures, or patterns in the image. Operation as filter (kernel) slides over the image matrix. At each position, the dot product of the filter and the corresponding image segment is calculated, producing a feature map. Feature maps highlight the presence of specific patterns in different regions of the image.

Activation Function: Non-linear functions are applied to introduce non-linearity. Helps the network learn complex patterns. Pooling

Layers: Reduce the spatial dimensions of feature maps to make computation efficient and retain essential information. Types are max pooling and average pooling. Reduces complexity, prevents overfitting, and highlights dominant features. Fully Connected

Layers: Aggregate the learned features and make predictions. The flattened feature maps are fed into one or more dense layers. The output represents the final classification or regression result.

Output Layer: The final layer provides the result based on the problem type are classification and regression produce outputs a continuous value.

Training the CNN: Loss function is measures the error between predicted and actual outputs. Backpropagation is to adjusts weights using gradients to minimize the loss function. Optimization Techniques help improve learning efficiency.

II. LITERATURE REVIEW

S.No.	Author	Year	Title	Outcomes
1	Ashgan H. Khalil, Atef Z.Ghalwash, Hala Abdel-Elsayed, I. Gouda, Salsama2 and Haitham A. Ghalwash	2023	Enhancing Digital Image Forgery Detection Using Transfer Learning ASH	The technique with the pre-trained model MobileNetV2 has the highest detection accuracy rate (around 95%) with fewer training parameters, leading to faster training time.
2	M. A. Elaskily, M. H. Alkinani, A. Sedik, M.M. Dessouky	2023	Deep learning based algorithm (ConvLSTM) for copy move forgery detection	Datasets have been combined to build new datasets for all purposes of generalization testing and coping with an over-fitting problem.
3	A. Mohassin, K. Farida	2021	Digital image forgery detection approaches	Different types of image forgeries and their corresponding detection has been provided and dedicated towards pixel-based methods of passive detection
4	K. D. Kadam, S. Ahirrao, K.Kotecha	2021	Multiple image splicing dataset (MISD)	A study of existing datasets used for the detection of image splicing reveals that they are limited to only image splicing and do not contain multiple spliced images.

III. IMPLEMENTATION

A. Modules Description

1) Dataset

In the first module of Digital Image Forgery Detection, we developed the system to get the input dataset. Data collection process is the first real step towards the real development of a machine learning model, collecting data. This is a critical step that will cascade in how good the model will be, the more and better data that we get, the better our model will perform. There are several techniques to collect the data, like web scraping, manual interventions. Our dataset is placed in the project and it's located in the model folder. The dataset is referred from the popular standard dataset repository kaggle where all the researchers refer it. The dataset consists of 12,615 Digital Image Forgery images. The following is the URL for the dataset referred from kaggle.

<https://www.kaggl.com/datasets/jayaprakashpondy/casia2-dataset>

2) Importing the necessary libraries

We will be using Python language for this. First we will import the necessary libraries such as keras for building the main model, sklearn for splitting the training and test data, PIL for converting the images into array of numbers and other libraries such as pandas, numpy, matplotlib and tensorflow.

3) Retrieving the Images

In this module we will retrieve the images from the dataset and convert them into a format that can be used for training and testing the model. This involves reading the images, resizing them, and normalizing the pixel values. We will retrieve the images and their labels. Then resize the images to (200,200) as all images should have same size for recognition. Then convert the images into numpy array.

4) ELA Image Analysis

Error level analysis is one technique for knowing images that have been manipulated by storing images at a certain quality level and then calculating the difference from the compression level. When JPEG was first saved, then it will compress the image the first time, most editing software like adobe photoshop, gimp, and adobe lightroom support JPEG compressing operation. If the image is rescheduled using image editing software, then compressed again. So it shows that the original image when the first image is taken using a digital camera has been compressed twice, first use the camera and the second is editing software. When viewed with the naked eye the image looks the same, but by using this method it will look the difference between a forgery image with the original image. Calculation for the average difference of the quantization table Y (luminance) and CrCb (Chrominance). The digital camera does not optimize the image for a specified camera quality level (high, medium, low, etc.). Original images from digital cameras should have high ELA values. Each subsequent resave will decrease the potential error rate. Original images from photography have high ELA values shown through white on the ELA image. When the image is resaved, using ordinary human vision does not show a significant degree of difference, but ELA shows the dominant black and dark colors. If this image is resaved again it will decrease the image quality. If the original image is then modified, ELA will show the modified area has a color with a higher ELA level. The describes how the output of ELA on the condition of the image.

5) Splitting the Dataset

In this module, the image dataset will be divided into training and testing sets. Split the dataset into Train and Test. 80% train data and 20% test data. This will be done to train the model on a subset of the data, validate the model's performance, and test the model on unseen data to evaluate its accuracy. Split the dataset into train and test. 80% train data and 20% test data.

6) Building the Model

For building the we will use sequential model from keras library. Then we will add the layers to make convolutional neural network. In the first 2 Conv2D layers we have used 32 filters and the kernel size is (5,5).

In the MaxPool2D layer we have kept pool size (2,2) which means it will select the maximum value of every 2 x 2 area of the image. By doing this dimensions of the image will reduce by factor of 2. In dropout layer we have kept dropout rate = 0.25 that means 25% of neurons are removed randomly.

We apply these 3 layers again with some change in parameters. Then we apply flatten layer to convert 2-D data to 1-D vector. This layer is followed by dense layer, dropout layer and dense layer again. The last dense layer outputs 2 nodes as the brain tumour or not. This layer uses the softmax activation function which gives probability value and predicts which of the 2 options has the highest probability.

B. CNN Convolutional Neural Networks

1) Padding

We have seen that convolving an input of 6 X 6 dimension with a 3 X 3 filter results in 4 X 4 output. We can generalize it and say that if the input is n X n and the filter size is f X f, then the output size will be (n-f+1) X (n-f+1):

- Input: n X n
- Filter size: f X f
- Output: (n-f+1) X (n-f+1)

There are primarily two disadvantages here:

1. Every time we apply a convolutional operation, the size of the image shrinks
2. Pixels present in the corner of the image are used only a few number of times during convolution as compared to the central pixels. Hence, we do not focus too much on the corners since that can lead to information loss

To overcome these issues, we can pad the image with an additional border, i.e., we add one pixel all around the edges. This means that the input will be an 8 X 8 matrix (instead of a 6 X 6 matrix). Applying convolution of 3 X 3 on it will result in a 6 X 6 matrix which is the original shape of the image. This is where padding comes to the fore:

- Input: $n \times n$
- Padding: p
- Filter size: $f \times f$
- Output: $(n+2p-f+1) \times (n+2p-f+1)$

There are two common choices for padding:

1. **Valid:** It means no padding. If we are using valid padding, the output will be $(n-f+1) \times (n-f+1)$
2. **Same:** Here, we apply padding so that the output size is the same as the input size, i.e.,

$$n+2p-f+1 = n$$

$$\text{So, } p = (f-1)/2$$

We now know how to use padded convolution. This way we don't lose a lot of information and the image does not shrink either. Next, we will look at how to implement strided convolutions.

2) Strided Convolutions

Suppose we choose a stride of 2. So, while convoluting through the image, we will take two steps – both in the horizontal and vertical directions separately. The dimensions for stride s will be:

- Input: $n \times n$
- Padding: p
- Stride: s
- Filter size: $f \times f$
- Output: $[(n+2p-f)/s+1] \times [(n+2p-f)/s+1]$

Stride helps to reduce the size of the image, a particularly useful feature.

Convolutions Over Volume

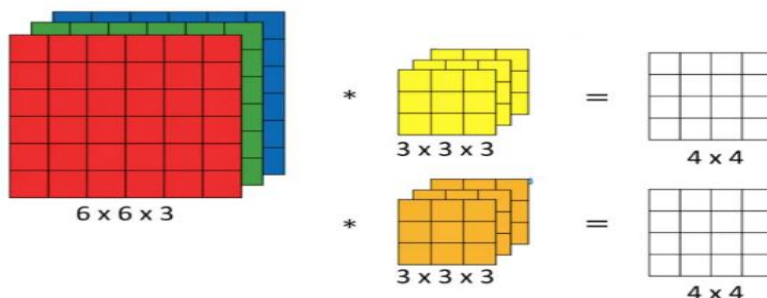
Suppose, instead of a 2-D image, we have a 3-D input image of shape $6 \times 6 \times 3$. How will we apply convolution on this image? We will use a $3 \times 3 \times 3$ filter instead of a 3×3 filter. Let's look at an example:

- Input: $6 \times 6 \times 3$
- Filter: $3 \times 3 \times 3$

The dimensions above represent the height, width and channels in the input and filter. Keep in mind that the number of channels in the input and filter should be same. This will result in an output of 4×4 . Let's understand it visually:

Since there are three channels in the input, the filter will consequently also have three channels. After convolution, the output shape is a 4×4 matrix. So, the first element of the output is the sum of the element-wise product of the first 27 values from the input (9 values from each channel) and the 27 values from the filter. After that we convolve over the entire image.

Instead of using just a single filter, we can use multiple filters as well. How do we do that? Let's say the first filter will detect vertical edges and the second filter will detect horizontal edges from the image. If we use multiple filters, the output dimension will change. So, instead of having a 4×4 output as in the above example, we would have a $4 \times 4 \times 2$ output (if we have used 2 filters):



Generalized dimensions can be given as:

- Input: $n \times n \times n_c$
- Filter: $f \times f \times n_c$
- Padding: p
- Stride: s
- Output: $[(n+2p-f)/s+1] \times [(n+2p-f)/s+1] \times n_c'$

Here, n_c is the number of channels in the input and filter, while n_c' is the number of filters.

3) One Layer of a Convolutional Network

Once we get an output after convolving over the entire image using a filter, we add a bias term to those outputs and finally apply an activation function to generate activations. *This is one layer of a convolutional network.* Recall that the equation for one forward pass is given by:

$$z^{[1]} = w^{[1]} * a^{[0]} + b^{[1]}$$

$$a^{[1]} = g(z^{[1]})$$

In our case, input ($6 \times 6 \times 3$) is $a^{[0]}$ and filters ($3 \times 3 \times 3$) are the weights $w^{[1]}$. These activations from layer 1 act as the input for layer 2, and so on. Clearly, the number of parameters in case of convolutional neural networks is independent of the size of the image. It essentially depends on the filter size. Suppose we have 10 filters, each of shape $3 \times 3 \times 3$. What will be the number of parameters in that layer? Let's try to solve this:

- Number of parameters for each filter = $3 \times 3 \times 3 = 27$
- There will be a bias term for each filter, so total parameters per filter = 28
- As there are 10 filters, the total parameters for that layer = $28 \times 10 = 280$

No matter how big the image is, the parameters only depend on the filter size. Awesome, isn't it? Let's have a look at the summary of notations for a convolution layer:

- $f^{[l]}$ = filter size
- $p^{[l]}$ = padding
- $s^{[l]}$ = stride
- $n_{[c]}^{[l]}$ = number of filters

Let's combine all the concepts we have learned so far and look at a convolutional network example.

Simple Convolutional Network Example

We'll take things up a notch now. Let's look at how a convolution neural network with convolutional and pooling layer works. Suppose we have an input of shape $32 \times 32 \times 3$:

We take an input image (size = $39 \times 39 \times 3$ in our case), convolve it with 10 filters of size 3×3 , and take the stride as 1 and no padding. This will give us an output of $37 \times 37 \times 10$. We convolve this output further and get an output of $7 \times 7 \times 40$ as shown above. Finally, we take all these numbers ($7 \times 7 \times 40 = 1960$), unroll them into a large vector, and pass them to a classifier that will make predictions. This is a microcosm of how a convolutional network works.

There are a number of hyperparameters that we can tweak while building a convolutional network. These include the number of filters, size of filters, stride to be used, padding, etc. We will look at each of these in detail later in this article. Just keep in mind that as we go deeper into the network, the size of the image shrinks whereas the number of channels usually increases.

In a convolutional network (ConvNet), there are basically three types of layers:

1. Convolution layer
2. Pooling layer
3. Fully connected layer

Let's understand the pooling layer in the next section.

4) Pooling Layers

Pooling layers are generally used to reduce the size of the inputs and hence speed up the computation. Consider a 4 X 4 matrix as shown below:

1	3	2	1
2	9	1	1
1	3	2	3
5	6	1	2

Applying max pooling on this matrix will result in a 2 X 2 output:

1	3	2	1
2	9	1	1
1	3	2	3
5	6	1	2

→

9	2
6	3

For every consecutive 2 X 2 block, we take the max number. Here, we have applied a filter of size 2 and a stride of 2. These are the hyperparameters for the pooling layer. Apart from max pooling, we can also apply average pooling where, instead of taking the max of the numbers, we take their average. In summary, the hyperparameters for a pooling layer are:

1. Filter size
2. Stride
3. Max or average pooling

If the input of the pooling layer is $n_h \times n_w \times n_c$, then the output will be $\{(n_h - f) / s + 1\} \times \{(n_w - f) / s + 1\} \times n_c$.

C. CNN Example

We'll take things up a notch now. Let's look at how a convolution neural network with convolutional and pooling layer works. Suppose we have an input of shape 32 X 32 X 3:

There are a combination of convolution and pooling layers at the beginning, a few fully connected layers at the end and finally a softmax classifier to classify the input into various categories. There are a lot of hyperparameters in this network which we have to specify as well.

Generally, we take the set of hyperparameters which have been used in proven research and they end up doing well. As seen in the above example, the height and width of the input shrinks as we go deeper into the network (from 32 X 32 to 5 X 5) and the number of channels increases (from 3 to 10).

All of these concepts and techniques bring up a very fundamental question – why convolutions? Why not something else?

1) Apply the model and plot the graphs for accuracy and loss

Once the model is built, it will be applied to the validation set to evaluate its accuracy and loss. The accuracy and loss will be plotted as a function of the number of epochs to visualize the performance of the model. We will compile the model and apply it using fit function. The batch size will be 10. Then we will plot the graphs for accuracy and loss. We got average training accuracy of 98%.

2) Accuracy on Test Set

After training and evaluating the model on the validation set, the accuracy of the model will be assessed on the test set. The accuracy on the test set will be an important metric for evaluating the model's performance. We got an accuracy of 92% on test set.

3) Saving the Trained Model

Once you're confident enough to take your trained and tested model into the production-ready environment, the first step is to save it into a .h5 or .pkl file using a library like pickle .

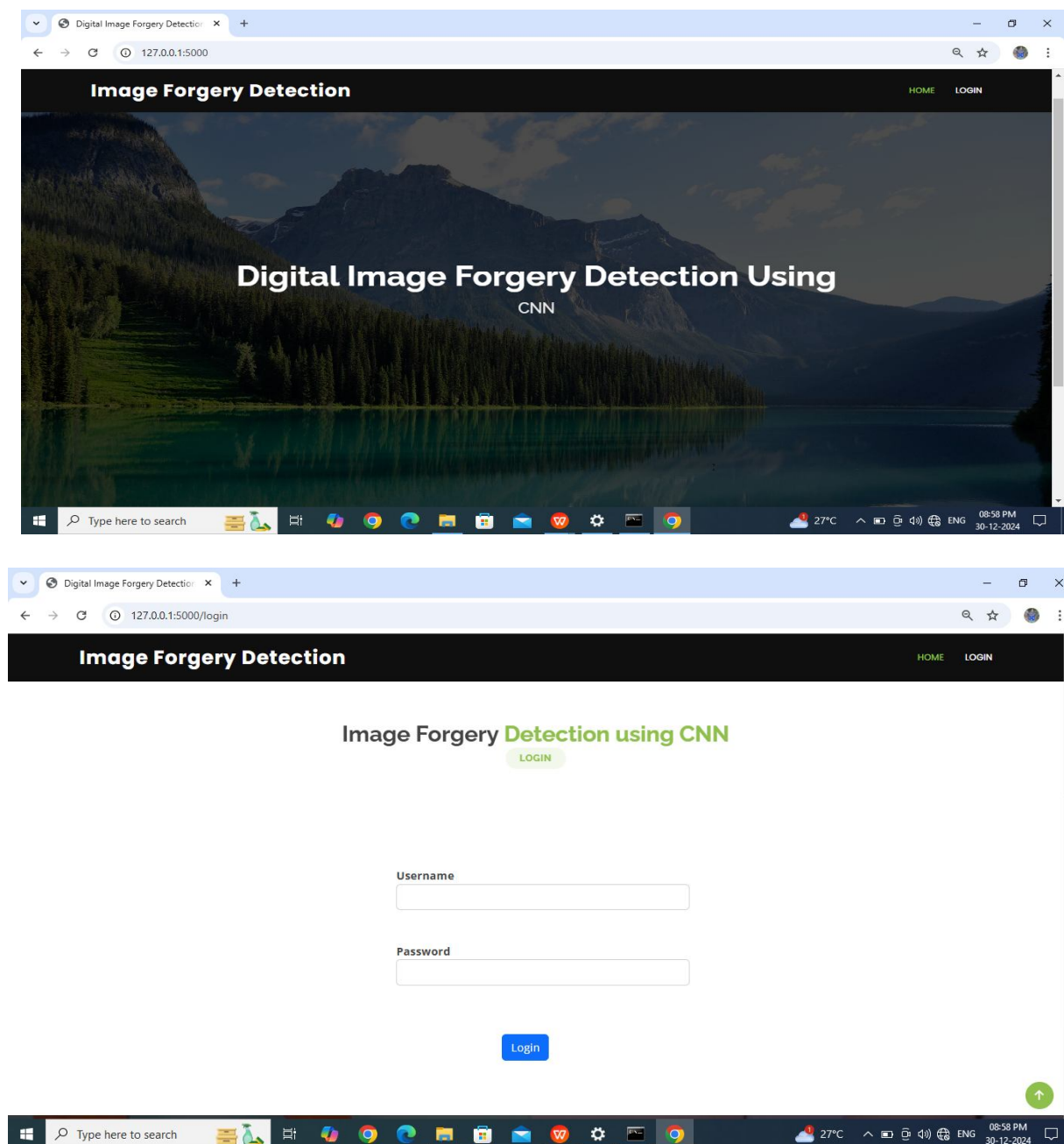
Make sure you have pickle installed in your environment.

Next, let's import the module and dump the model into .pkl file.

4) Digital Image Forgery Detection Using Convolutional Neural Network Approach

Digital image forgery detection using conventional neural networks (CNNs) involves leveraging deep learning techniques to identify and classify altered or manipulated images. First, we should collect the dataset then we pre-process the data with Error Level Analysis (ELA). Once we done that, we should split the data into train and test the training data will be feed through inside the Convolutional Neural Network for train the model. Once it's getting trained, we use that trained model for evaluate the result for our test data. Advantages are Process large datasets quickly and efficiency, automatic features extraction, reducing the number of parameters and computational cost, leading to high detection accuracy, autonomously improve detection capabilities over time through continuous learning and adaptation.

IV. RESULTS





Digital Image Forgery Detection x +

127.0.0.1:5000/login

Image Forgery Detection HOME LOGIN

Image Forgery Detection using CNN

LOGIN

Username

admin

Password

....

Login

↑

Type here to search

Digital Image Forgery Detection x +

127.0.0.1:5000/index


Image Forgery Detection HOME LOGIN PREVIEW

Digital Image Forgery Detection using CNN Detection

PREVIEW

Upload Image:

Choose File Tp_D_CRN_...070_11413.jpg

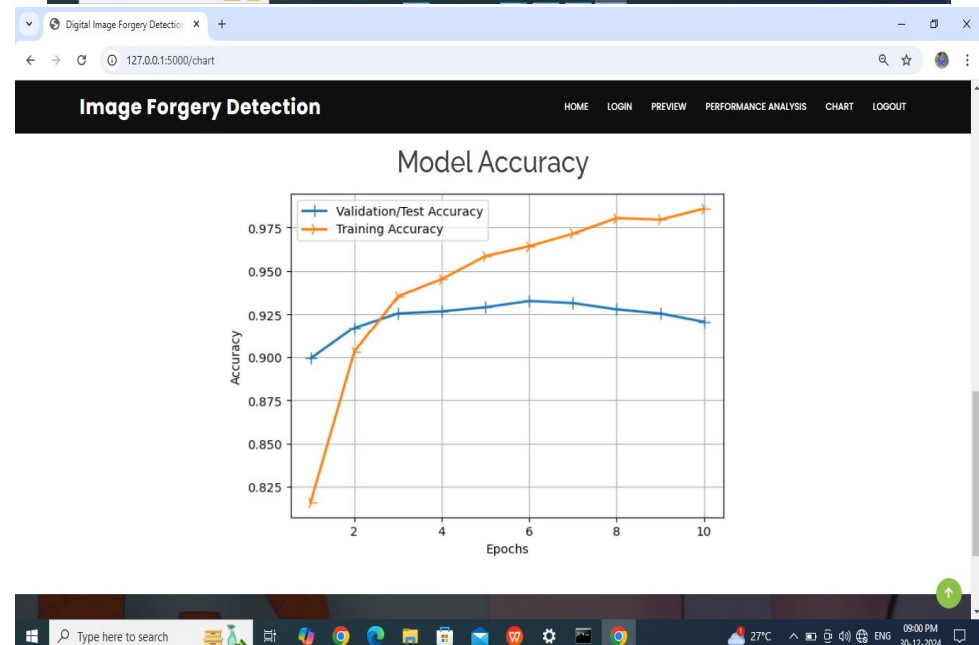
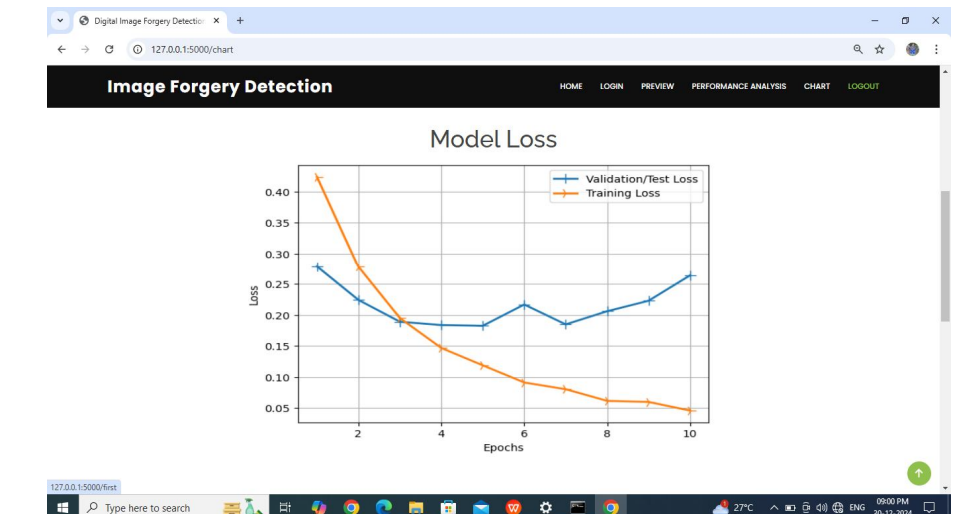
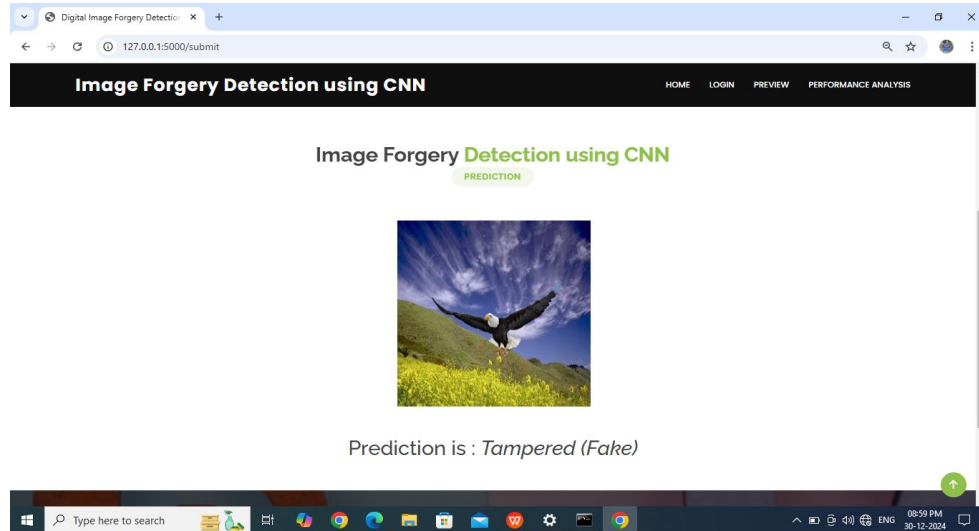


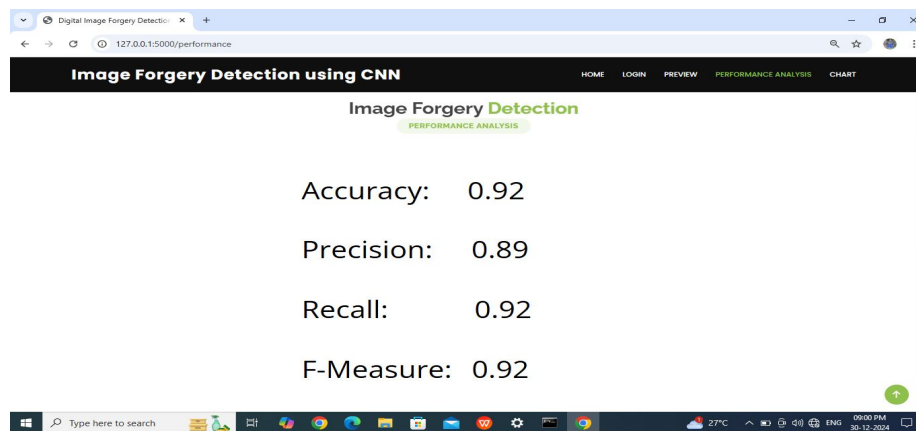
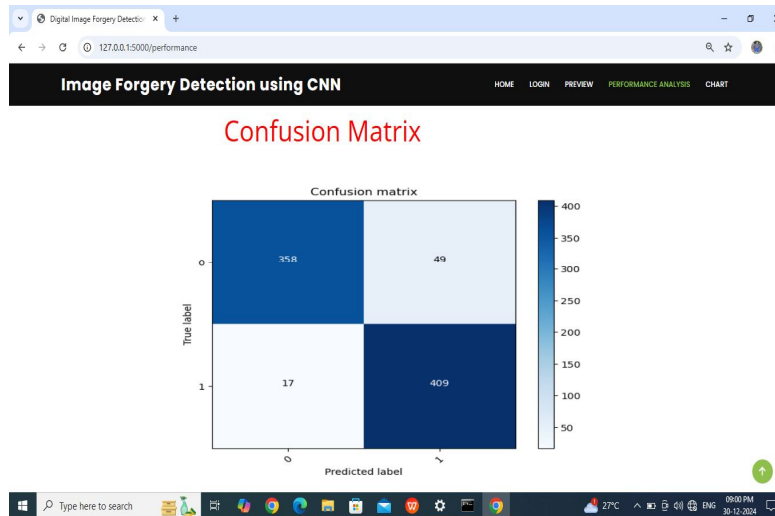
Submit

↑

Type here to search

27°C 09:01 PM 30-12-2024





V. CONCLUSION

The effectiveness of CNNs in automatically learning intricate patterns and features associated with forged regions, achieving high accuracy and robustness across diverse datasets. It eliminates the need for manual feature extraction, making it more adaptable to various forgery types and resolutions. The potential of CNN-based systems as a reliable and scalable solution for digital image forensics, with significant applications in media authenticity verification and cybersecurity.

REFERENCES

- [1] Bayar, B., & Stamm, M. C. (2016). A deep learning approach to universal image manipulation detection using a new convolutional layer. ACM Workshop on Information Hiding and Multimedia Security, 5-10.
- [2] Rao, Y., & Ni, J. (2016). A deep learning approach to detection of splicing and copy-move forgeries in images. IEEE International Workshop on Information Forensics and Security (WIFS), 1-6.
- [3] Zhou, P., Han, X., Morariu, V. I., & Davis, L. S. (2018). Learning rich features for image manipulation detection. IEEE/CVF Conference on Computer Vision and Pattern Recognition, 1053-1061.
- [4] Bappy, J. H., Roy-Chowdhury, A. K., Bunk, J., Nataraj, L., & Manjunath, B. S. (2017). Exploiting spatial structure for localizing manipulated image regions. IEEE International Conference on Computer Vision (ICCV), 4970-4979.
- [5] Huh, M., Liu, A., Owens, A., & Efros, A. A. (2018). Fighting fake news: Image splice detection via learned self-consistency. European Conference on Computer Vision (ECCV), 101-117.
- [6] Verdoliva, L., Poggi, G., & Cozzolino, D. (2020). A feature-based approach for image tampering detection and localization. IEEE Transactions on Information Forensics and Security, 15, 1231-1245.
- [7] Bi, Q., Liu, X., & Wen, Y. (2019). Video forgery detection based on convolutional neural networks and long short-term memory. Journal of Visual Communication and Image Representation, 58, 567-575.
- [8] Zhuang, P., Ni, J., & Zhao, Y. (2019). Image forgery detection via convolutional neural networks and multi-view feature learning. IEEE Transactions on Information Forensics and Security, 14(10), 2517-2532.



10.22214/IJRASET



45.98



IMPACT FACTOR:
7.129



IMPACT FACTOR:
7.429



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24*7 Support on Whatsapp)