International Journal of Advances in Applied Sciences (IJAAS)
Vol. 13, No. 1, March 2024, pp. 93~105
ISSN: 2252-8814, DOI: 10.11591/ijaas.v13.i1.pp93-105

Detecting facial image forgeries with transfer learning techniques

Nishigandha N. Zanje[1], Anupkumar M. Bongale[1], Deepak Dharrao[2]
[1]Department of Artificial Intelligence and Machine Learning, Symbiosis Institute of Technology, Symbiosis International (Deemed University), Lavale, India
[2]Department of Computer Science and Engineering, Symbiosis Institute of Technology, Symbiosis International (Deemed University), Lavale, India

Article Info

Article history:

Received Jun 26, 2023
Revised Jul 15, 2023
Accepted Dec 22, 2023

Keywords:

Convolution neural network
DenseNet201
Image forgery
Transfer learning
VGG-19

ABSTRACT

Digital images have become ubiquitous in our daily lives, appearing on our smartphone screens and online websites. They are widely used in numerous industries, including media, forensic and criminal investigations, medicine, and more. The ease of access to consumer photo editing tools has made it simple to manipulate images. However, such altered images pose a serious risk in fields where image authenticity is crucial, making it challenging to confirm the reliability of digital images. Digital image fraud involves altering an image's meaning without leaving any obvious signs. In this study, we present three convolutional neural network-based transfer learning techniques "CNN" classification of facial image forgeries, using VGG-19, InceptionV3, and DenseNet201. Among these methods, DenseNet201 achieved the highest accuracy of 99%, followed by InceptionV3 at 94% and VGG-19 at 84%.

Corresponding Author:

Anupkumar M. Bongale
Department of Artificial Intelligence and Machine Learning, Symbiosis Institute of Technology Symbiosis International (Deemed University)
Lavale, 412115, Maharashtra, India
Email: anupkumar.bongale@sitpune.edu.in

1. INTRODUCTION

Digital images have evolved as the primary source of information and the quickest method of conveying it in the digital age we live in. Digital pictures are widely employed in various fields, including the military, diagnosis in the health domain, art, and photography. Consequently, forensics of digital images has come to be a rapidly increasing need in society as a whole, and it is crucial to acquire authentic images. However, because of the accessibility of inexpensive hardware and software tools and computers, digital picture alteration is very simple and leaves no discernible signs of artifice, making it difficult for humans to detect such modifications. As a result, digital images' authenticity and integrity are compromised, and image modification can be used to conceal crucial traces or transmit incorrect information. Therefore, to ensure the images' integrity, it is necessary to identify any modifications made to them [1]. Deepfakes possess the potential to bring creative or productive effects in various fields, such as movies, virtual reality, video games, and photography and recreation activities. For instance, they can enable realistic video translation of foreign films, historical personalities may be simulated for educational purposes, and allow virtual clothes try-on while shopping [2], [3]. The emergence of "deep fakes," which are manipulated materials created using user-friendly deep learning technologies like automatic encoders (AE) or generating adversarial networks (GAN), has resulted in a significant increase in phony multimedia [4].

Due to the quality of manipulated videos and their user-friendly applications, deepfakes have gained popularity among a wide range of users, from professionals to novices, with varying levels of computer skills. These applications primarily rely on deep learning techniques, known for their ability to represent data that is complicated and multidimensional. One specific type of a deep network with this capability is the deep autoencoder network, which has been extensively used for the compression of images and reduction of dimensionality [5]–[7]. Figure 1 showcases several well-known deep fakes that are circulating online. While these fakes were created for entertainment and feature prominent individuals in improbable scenarios, they are still easily identifiable.

Figure 1. Spindle tool clamping system

2. RELATED WORK

Various methods can be used to collect a diverse and substantial collection of broadcasted images, including screen capture, scanning printed images, or rephotographing displayed or printed images. The resulting dataset consists of approximately 14,500 retransmission images collected using different devices, such as 180 recapture cameras, 234 monitors, 173 scanners, and 282 printers. It is important to keep reference numbers to ensure the preservation of inherent camera qualities in the original JPEG files when using file-based forensic tools to identify any modifications made to the images [8].

Despite considerable success in various applications, face identification systems are still vulnerable to face spoofing attacks [9], [10] including face video replay assaults. To address this issue, the authors proposed a "twin streaming convolutional neural network (TSCNN)" that operates on two complementing spaces: the red, green, and blue (RGB) area, which is the original imaging space and the multi-scale retinex (MSR) space, which is an illumination-invariant space. A two-stream convolution network utilizing CNN classifiers, MobileNet and ResNet 18, from RGB and MSR, is suggested to identify real faces from fraudulent faces [11].

Punnappurath and Brown [12] presented a method for a document recapture detection technique that utilizes a Siamese network to take out distinguishing characteristics from a document capture image. This approach combines picture forensic methods and metric learning and employs multiple experimental protocols and network architectures. By using ResNeXt101 as the backbone of the suggested network, they achieved an attack presentation classification error rate (APCER) of fewer than 5% and a bona fide presentation classification error rate (BPCER) of 5.56% at the 5% BPCER decision threshold.

In this article, we explore the potential impact of the National Institution of Standardization and Framework for Cybersecurity (NIST) in establishing suitable cybersecurity standards, as well as the increasing responsibility for cybersecurity [13]. To enhance information exchange and create plans to lessen cyber hazards, critical infrastructure cybersecurity is being promoted, including public cooperation with infrastructure owners and management.

Chorowski *et al*. [14] conducted a comprehensive study in 2020 on deep learning-based methods for detecting fake images, with openly accessible findings and information on the datasets used in the study. The study focused on a detection-only task, where the detection was performed at the image level, with the output being either a manipulated or legitimate image. The localization task marked off the tampered area in the

fake image as the detection was done at the pixel level. Shackleford *et al*. [15] proposed a method that used principal component analysis (PCA) in their work. However, due to the linear nature of PCA, this approach is unable to detect fused edges, which may result in the loss of some nonlinear features. To overcome this limitation, a new approach based on locally linear embedding (LLE) was developed, which can detect both copy-move regions and fused edges.

Barad and Goswami [16] developed a passive-blind detection method to detect copy-move forgery. Their approach involved dividing the image into blocks of equal size and applying improved singular value decomposition using block-matching procedures to obtain a reduced-dimension representation of all the image blocks. The blocks were then lexicographically sorted and a matching step was applied to identify duplicate blocks based on their feature vectors. A correlation coefficient threshold was set, and a decision was made based on whether the threshold was reached, indicating a forged or unforged region. The proposed algorithm showed strong detection capability and anti-noise capability.

Mirsky and Lee [17] presented a method based on "scale-invariant feature transform (SIFT)" to detect image alterations. The authors described the state-of-the-art SIFT-point matching method and compared it with their SIFT-based approach, which includes keypoint clustering, cluster matching, and texture analysis. Instead of matching individual points, the approach aims to identify clusters of points that belong to the same object, resulting in better detection results. Finally, the method analyzes the textures of the matched area and compares them to validate the results, eliminating false positives.

In the past, local binary patterns (LBP) methods were commonly used to defend against replay attacks. However, Simonyan and Zisserman [18] presented a method based on local speed patterns that achieved higher accuracy. Given the arrival of deep learning, replay attack detection has significantly improved. Szegedy *et al*. [19] used a pre-trained CNN and support vector machine to classify features. Chingovska *et al*. [20] optimized the high-performance CNN architecture filters. Yu *et al*. [21] developed their own CNN that uses nonlinear diffusion based on an additive operator splitting scheme. Kim *et al*. [22] utilized a pre-trained CNN and the entire image instead of just the extracted face region.

In their research, Yang *et al*. [23] conducted several studies demonstrating the vulnerability of a face verification system to prevent attacks using masks, evaluated an anti-spoofing method proposed by Menotti *et al*. [24] also in Alotaibi and Mahmood work's [25], and explored a reflectance-based approach for detecting 3D mask attacks inspired by Ito *et al*. and Kose and Dugelay work [26], [27]. Kose and Dugelay [28] presented a score-level fusion strategy for identifying numerous attack types and later suggested an anti-spoofing solution based on dynamic texture, which outperformed the original LBP [29].

Although very deep convolutional networks (VGGNet) [30] have a simple architecture, it has a significant drawback: the network requires a substantial amount of computation for evaluation. On the other hand, GoogLeNet's Inception architecture [31] was designed specifically to perform well under tight memory and computational constraints. To achieve this, GoogleNet utilized about 7 million parameters, resulting in a 9x reduction compared to its predecessor, AlexNet, which used 60 million parameters. Moreover, VGGNet employed roughly three times more parameters than AlexNet.

Through the detailed literature review, the current state of research with proposed work is identified and the current research challenges and limitations are explored. It is observed that there is a need for more accurate and efficient techniques for detecting facial image forgeries. Based on the information provided in the related work, some potential gaps in the facial image forgeries area could include the limited effectiveness of current anti-spoofing techniques in detecting sophisticated facial image forgeries, the need for more robust and efficient algorithms that can handle real-world scenarios and the potential for transfer learning to increase accuracy and efficiency of facial image forgery detection.

3. DATASET SETUP FOR EXPERIMENTAL INVESTIGATION

3.1. Dataset description

The dataset consists of real and fake human face photographs, where the fake facial images were created by skilled Photoshop experts. It comprises professionally produced, high-quality facial images that have been photoshopped. These images are composites of different faces, which have been separated based on the eyes, mouth, nose, or the entire face. There are 2,041 images in the dataset, where 47% are fake face images and 53% are real face images. For training, 1,632 images were used and for testing 409 images were used.

3.2. Dataset pre-processing using Gabor filter

A few frequently used image pre-processing techniques are the first step in preparing the image for use in a machine learning model resizing, which entails altering the image's size to a standard size that suits the model. This is crucial if the images are of varying sizes or if the model requires images to be of a particular size. The second step is grayscale conversion, which involves converting the color of the original

image into a grayscale image. This may be required if the model cannot handle color information or if color information is irrelevant to the task at hand. Finally, normalization involves scaling the image's pixel values to a common range. This step is necessary if the pixel values are not centered on zero or if the range of values is not uniform across all images.

The Gabor filter, named after its inventor Dennis Gabor in the 1940s while seeking to improve electron microscope resolution, is an effect employed in image processing to detect edges and other features. It works by convolving an image with a complex sinusoidal function that is defined by frequency and orientation. The frequency of the function determines the scale of the features that the filter detects, while orientation specifies the direction of the features. Applying a Gabor filter to an image at different frequencies and orientations enables the detection of a broad range of image features as shown in Figure 2. Because they can capture both local and global aspects of an image, Gabor filters are frequently utilized in image analysis tasks like object detection and recognition. They are particularly useful for analyzing images with patterns or textures, as they record the spatial content of the image. Image processing is a crucial stage in the machine learning process because it guarantees that images are suitable for use in the model and ensures that the model can derive the necessary features from the images. Figure 2(a) shows some of the sample real and fake images in the dataset. After applying the Gabor filter level 1 and level 2, the images are transformed as shown in Figures 2(b) and 2(c) respectively.

(a) (b) (c)

Figure 2. Gabor filter usage (a) sample real and fake image, (b) Gabor filter level 1 real and fake image, and (c) Gabor filter level 2 real and fake image

4. PRETRAINED MODELS

4.1. VGG-19

The VGG-19 is a convolutional neural network (CNN) with 16 layers of convolution and 3 fully linked layers total of 19 layers as shown in Table 1. The input layer takes an image of the size 224×224×3 (RGB). The convolutional layers have varying filter sizes with stride 1 and padding 1, and the maximum layer pooling stride 2 and a 2×2 window. The output sizes for the "FC-fully connected layers" are 4,096, with the last layer having 1,000 neurons for the ImageNet dataset's 1,000 output classes. The network has a total of approximately 143 million parameters, making it one of the larger CNNs at the time of its development.

Table 1. The VGG-19 architecture

Layer	Name	Output size
Input	Input	224×224×3
Convolutional	Conv1-64	224×224×64
Convolutional	Conv2-64	224×224×64
Max pooling	Pool1	112×112×64
Convolutional	Conv3-128	112×112×128
Convolutional	Conv4-128	112×112×128
Max pooling	Pool2	56×56×128
Convolutional	Conv5-256	56×56×256
Convolutional	Conv6-256	56×56×256
Convolutional	Conv7-256	56×56×256
Convolutional	Conv8-256	56×56×256
Max pooling	Pool3	28×28×256
Convolutional	Conv9-512	28×28×512
Convolutional	Conv10-512	28×28×512
Convolutional	Conv11-512	28×28×512
Convolutional	Conv12-512	28×28×512
Max pooling	Pool4	14×14×512
Fully connected	FC1	4,096
Fully connected	FC2	4,096
Fully connected	FC3	1,000

VGG-19 parameters and model setup are shown in Figure 3. The pre-trained model is initialized with weights from the "imagenet" dataset and the top layer that is completely “Fully” connected at the top is excluded using the "include_top=False" argument. The input tensor shape is set to (128, 128, 3), with the feature maps having a maximum layer of pooling applied. The model is then set to non-trainable and a new “fully connected - FC layer” with 512 units and activation of rectified linear units (ReLU) is added to the output of the VGG-19 base. Finally, a sigmoid activation function is applied to 2 units belonging to the output layer for binary classification. The model is compiled using binary classification using the "Adam optimizer" and "binary_cross-entropy" loss function. The metric for "accuracy" is used to evaluate using the model for training. The model instance is then summarized using the "summary()" method to display the architecture, how many parameters are trainable and untrainable, and the output shape of each layer.

```
vgg_model=tf.keras.applications.VGG19(include_top=False,
                                      weights="imagenet",
                                      input_tensor=None,
                                      input_shape=(128,128,3),
                                      pooling="max",classes=1000)
vgg_model.trainable = Falses
inputs = vgg_model.input
m = tf.keras.layers.Dense(512, activation='relu')    (vgg_model.output)
outputs = tf.keras.layers.Dense(2, activation='sigmoid')(m)
vgg_model = tf.keras.Model(inputs=inputs, outputs=outputs)
vgg_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
vgg_model.summary()
```

Figure 3. VGG-19 parameters and model setting based on Keras library

4.2. InceptionV3

InceptionV3 is a model that is a “CNN” used for tasks that classify images. The model has a total of 48 layers, including, Inception parts, a layer for dropouts, a fully connected layer, max-pooling layers, and a softmax activation layer as shown in Table 2. The input image size is 224×224×3 and the model produces a probability distribution as its output with over 1,000 classes. The model uses multiple convolutional layers with different kernel sizes in each Inception module to capture features at different scales, and the quantity of filters increases when we explore the network deeper to learn more complex features. The global average pooling layer is employed to list the attributes across spatial locations and the completely connected layer, and then use the softmax activation layer for classification.

Table 2. The InceptionV3 architecture

Layer	Name	Output Size
Convolutional layer	Conv2d_1a_3×3	224×224×3
Convolutional layer	Conv2d_2a_3×3	112×112×32
Convolutional layer	Conv2d_2b_3×3	112×112×64
Max pooling layer	Maxpool_3a_2×2	56×56×64
Convolutional layer	Conv2d_3b_1×1	56×56×80
Convolutional layer	Conv2d_4a_3×3	56×56×192
Max pooling layer	Maxpool_5a_3×3	28×28×192
Inception module 1	Mixed_5b	28×28×256
Inception module 2	Mixed_5c	28×28×288
Inception module 3	Mixed_5d	28×28×288
Inception module 4	Mixed_6a	14×14×768
Inception module 5	Mixed_6b	14×14×768
Inception module 6	Mixed_6c	14×14×768
Inception module 7	Mixed_6d	14×14×768
Inception module 8	Mixed_6e	14×14×768
Inception module 9	Mixed_7a	7×7×1,280
Inception module 10	Mixed_7b	7×7×2,048
Inception module 11	Mixed_7c	7×7×2,048
Global average pooling layer	Globalaveragepool2d_1a	1×1×2,048
Dropout layer	Dropout_1b	1×1×2,048
Fully connected layer	Logits	1×1×1,000
Softmax activation layer	Softmax	1×1×1,000

Figure 4 shows the code snippet defines a model which is a “CNN model” based on the InceptionV3 architecture utilizing the Keras application programming interface (API) for TensorFlow. The InceptionV3 model is used with pre-trained weights on the ImageNet dataset by setting weights="imagenet". Include_top is configured with a False value indicating that the final layer, (entirely connected) of the model will be removed, allowing for the addition of custom layers. The input_shape parameter specifies the expected input image dimensions of 128×128 pixels with three color channels. The pooling parameter is set to "max", indicating that the maximum activation value of each filter will be selected during the pooling process. The inception_model.trainable = False statement freezes the layers which have been already trained, of the InceptionV3 model so that their weights cannot be updated during training. The frozen model is used as a feature extractor to get features out of the input image.

Next, a unique classifier is introduced to the model. Having 512 units, the dense layer and ReLU layer of activation are added on top of the InceptionV3 model's output. The final output of the model is created by passing the result of the custom layer through a second Dense layer with a pair of units and a sigmoid activation function. Finally, the customized InceptionV3 model is built with the inputs and outputs specified for the custom layers using the tf.keras.Model method. To classify photos into two groups, the model that results may be trained through backpropagation with the appropriate loss function and optimizer.

```
inception_model=tf.keras.applications.InceptionV3(include_top=False,
                                    weights="imagenet",
                                    input_tensor=None,
                                    input_shape=(128,128,3),
                                    pooling="max",
                                    classes=1000)
inception_model.trainable = False

Adding dense layers
inputs = inception_model.input
m=tf.keras.layers.Dense(512,activation='relu')
          (inception_model.output)
outputs = tf.keras.layers.Dense(2, activation='sigmoid')(m)
inception_model=tf.keras.Model(inputs=inputs, outputs=outputs)
```

Figure 4. InceptionV3 parameters and model setting based on Keras library

4.3. DenseNet201

DenseNet-201 is a “deep CNN” employed for image classification tasks and has 201 layers. It takes as input a 224×224×3 RGB image and outputs a probability distribution over 1,000 possible classes as shown in Table 3. The network begins with an input layer, which accepts an image as input followed by two convolutional layers and the maximum pooling layer to shrink the feature maps' spatial dimensionality. The output of the initial convolutional layer is passed through four dense blocks, each of which contains several convolutional layers that are densely interconnected. Each layer within a dense block gets the input map of features from all preceding layers and transmits its own set of feature maps to all succeeding layers. The network can efficiently learn complicated representations thanks to the dense connection, which also helps to solve the vanishing gradient problem.

The transitional layer lowers the dimension of the maps of features and regulates how many feature maps are passed on to the following dense block. An activation layer for ReLU, a batch normalization layer, a max pooling layer, and a convolutional layer make up the transition layers. A universal averaging layer that averages the feature maps across all spatial dimensions follows the last dense block to provide a fixed-sized vector of features. After going through three fully interconnected layers, this feature vector is finally generated as a probability distribution over all 1,000 classes utilizing the softmax activation function.

The Python snippet shown in Figure 5 defines a DenseNet201 model, an already trained using a deep CNN to classify images, using the TensorFlow Keras library. The model is configured to exclude the top layer and use the ImageNet dataset's pre-trained weights. The source shape is set to 128×128 pixels with 3 color channels. The resulting maps of features of the layers of convolution are subjected to the universal

maximum pooling operation to minimize the spatial dimensions. The final dense layer's activation function is sigmoid and the total quantity of output class values is set to 2. Over the pre-trained layers, a fresh dense layer containing 512 units and activation of ReLU is added to adapt the model for a particular classification job. Finally, the model is compiled and set to be non-trainable, which means the weights of the pre-trained layers will not be updated during training. This code creates a new classification model by fine-tuning the DenseNet201 architecture on a different dataset with two output classes.

Table 3. The DenseNet201 architecture

Layer	Name	Output size
Input layer		224×224×3
Convolutional layer	Conv1-64	112×112×64
Convolutional layer	Conv2-64	112×112×64
Max pooling layer	Pool1	56×56×64
Dense block 1	Dense-b1	56×56×256
Transition layer 1	Trans-l1	28×28×256
Dense block 2	Dense-b2	28×28×512
Transition layer 2	Trans-l2	14×14×512
Dense block 3	Dense-b3	14×14×1024
Transition layer 3	Trans-l3	7×7×1024
Dense block 4	Dense-b4	7×7×1920
Global average pooling layer	GA-Pool	1,920
Fully connected layer	FC1	1,000
Softmax activation layer	Softmax	1,000
Input layer		224×224×3
Convolutional layer	Conv1-64	112×112×64
Convolutional layer	Conv2-64	112×112×64
Max pooling layer	Pool1	56×56×64

4.4. Proposed system

The present study employs three transfer learning frameworks based on CNN to classify facial image forgery. The dataset is collected and pre-processed prior to dividing it into test and training sets. Data used for training is utilized to teach the VGG-19, InceptionV3, and DenseNet201 models as shown in Figure 5. Subsequently, the trained models are tested with the test data and the resulting output is evaluated using various metrics such as accuracy and loss graphs, classification reports, and confusion matrix.

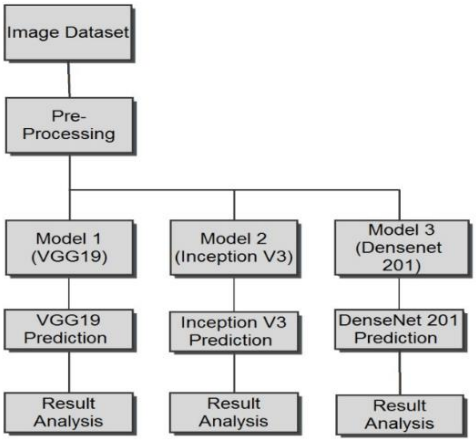

Figure 5. Proposed system architecture

5. RESULTS AND DISCUSSION

5.1. Accuracy plot graph

Figure 6 plots the accuracy of the model on the y-axis against the quantity of training epochs (iterations) in the x-axis of VGG-19. This model went through 20 epochs of training, with the first epoch's training accuracy being 56% and the validation accuracy being 63%. Each epoch saw a slight improvement in accuracy and the last epoch had an accuracy in training of 76% and an accuracy for validation of 81%.

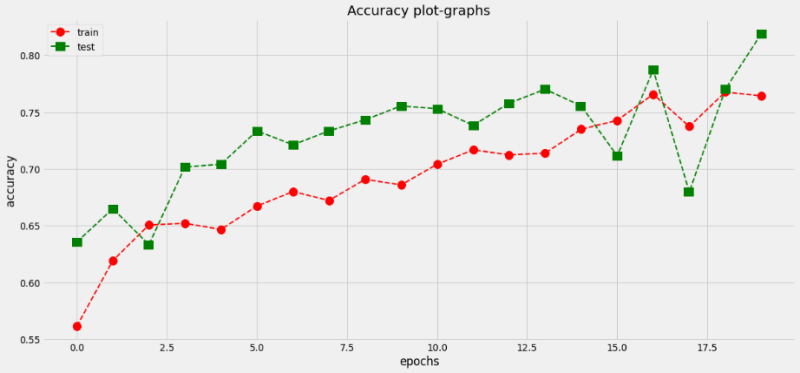

Figure 6. VGG-19 accuracy plot-graph

Figure 7 shows the accuracy of the InceptionV3 model. The training model underwent 20 epochs, in the first epoch, there was a training accuracy of 57% and the validation accuracy of 67% was achieved with each subsequent epoch, the accuracy gradually increased, culminating in the final epoch where the model achieved 89% training accuracy and 94% validation accuracy. Figure 8 shows the accuracy of the DenseNet201 model. The training model underwent 20 epochs, with the initial epoch yielding a training accuracy of 55% and a validation accuracy of 59%. The accuracy gradually increased with each subsequent epoch, ultimately reaching a training accuracy of 94% and a validation accuracy of 99% in the final epoch.

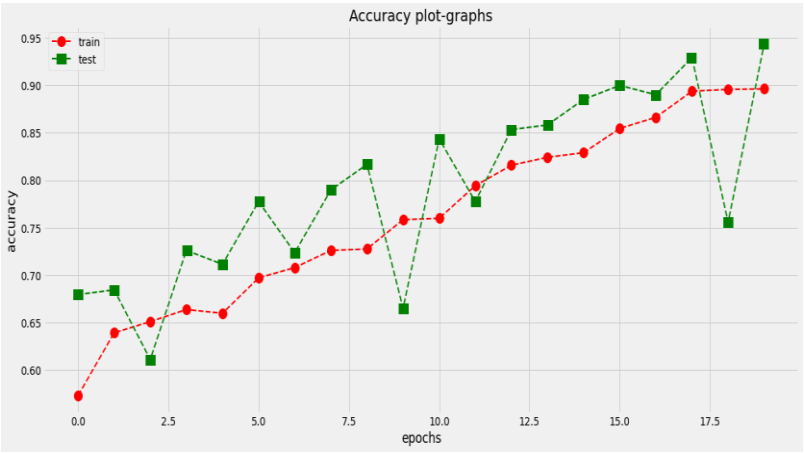

Figure 7. InceptionV3 accuracy plot

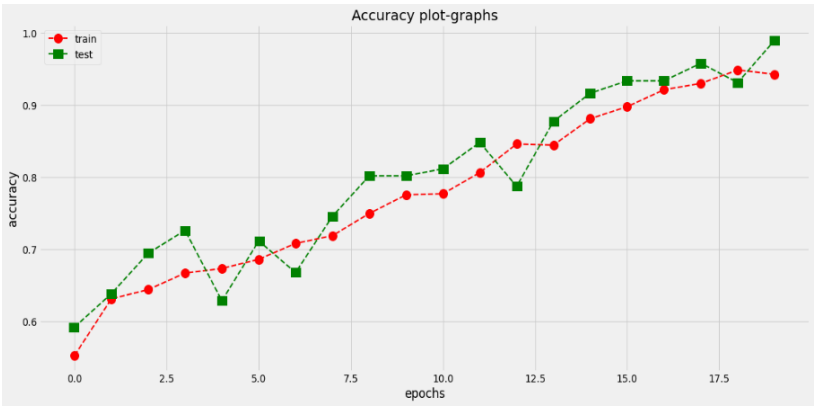

Figure 8. DenseNet201 accuracy plot

5.2. Loss plot graph

The loss plot shows the loss of the model in the y-axis against the number of training epochs (iterations) in the x-axis. Figure 9 shows that, in the first epoch, with 20 epochs in total, the training loss was 71% and the validating loss was 64% for the VGG-19. As each epoch progressed, both the training and validating loss gradually decreased. By the final epoch, the training loss had decreased to 47% and the validating loss had decreased to 43%. Similarly, in Figure 10 the training model underwent 20 epochs, in the first epoch, there was a training accuracy of 57% and the validation accuracy of 67% was achieved for InceptionV3. With each subsequent epoch, the accuracy gradually increased, culminating in the final epoch where the model achieved 89% training accuracy and 94% validation accuracy. In Figure 11 it is observed that the DenseNet201 training model underwent 20 epochs, in the first epoch, there was a training accuracy of 57% and the validation accuracy of 67% was achieved with each subsequent epoch, the accuracy gradually increased, culminating in the final epoch where the model achieved 89% training accuracy and 94% validation accuracy.

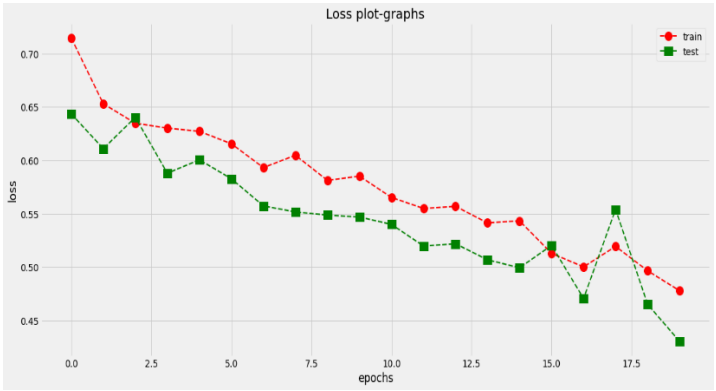

Figure 9. Loss graph for VGG-19

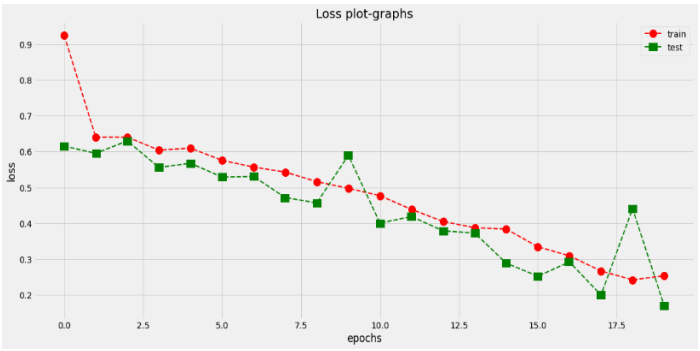

Figure 10. Loss graph for InceptionV3

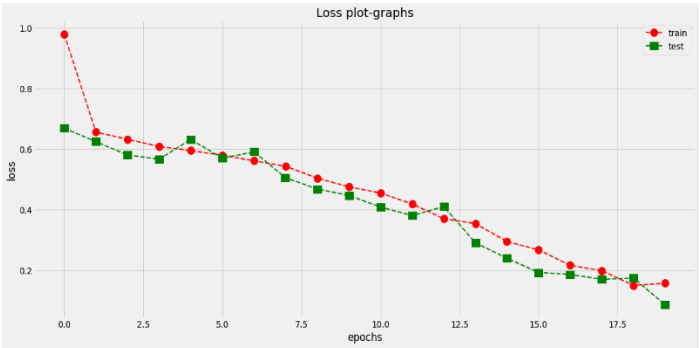

Figure 11. Loss graph for DenseNet201

5.3. Classification report

In deep learning, a classification report is an evaluation metric used to assess performance. It includes information on the model's accuracy, memory usage, F1-score, and support. This report is used to display the trained categorization. The classification report denotes "0" as a real face and "1" as a fake face. Out of a total of 409 input Luisa images for testing, 83 are real face images, and 81 are fake face images. The precision of real face image classification is 83% and that of fake face image classification is 81%. The recall rate for real face images is 83%, while for fake face images it is 81%. Additionally, the F1-score for real faces is 83%, and for fake faces it is 81%. All of these metrics are displayed in Table 4.

Table 5 shows the classification report for the InceptionV3 model. The training accuracy of the model is 40.9%, with 192 out of the total 409 input images classified as fake and 217 as real. The precision of the model for identifying fake face images is 94%, while for real face images, it is 95%. The recall rates for fake and real faces are both 94%. Additionally, the F1-scores for fake and real images are 95% and 94%, respectively. Out of the 409 input images provided for testing, 217 are real face images, and 192 are fake face images. The precision achieved for real face image classification is 100%, and for fake face image classification, it is 98%. The recall rate for real face images is 99% and for fake face images, it is also 99%. Additionally, the F1-score for real faces is 99% and the fake face is 99%. These performance metrics are visibly depicted in Table 6.

Table 4. Classification report of VGG-19 model

	Precision	Recall	F1-score	Support
Fake	0.81	0.81	0.81	192
Real	0.83	0.83	0.83	217
Accuracy			0.82	409
Macro avg.	0.82	0.82	0.82	409
Weighted avg.	0.82	0.82	0.82	409

Table 5. Classification report of InceptionV3 model

	Precision	Recall	F1-score	Support
Fake	0.94	0.94	0.94	192
Real	0.95	0.94	0.95	217
Accuracy			0.94	409
Macro avg.	0.94	0.94	0.94	409
Weighted avg.	0.94	0.94	0.94	409

Table 6. Classification report of DenseNet201 model

	Precision	Recall	F1-score	Support
Fake	0.81	0.81	0.81	192
Real	0.83	0.83	0.83	217
Accuracy			0.82	409
Macro avg.	0.82	0.82	0.82	409
Weighted avg.	0.82	0.82	0.82	409

5.4. Confusion matrix

The confusion matrix presented in Figure 11, Figure 11(a) depicts the performance of the VGG-19 model, which appears to be satisfactory based on the results. Out of the 409 images provided, this model achieved an accuracy of 81%. Figure 11(b) displays "the confusion matrix of the InceptionV3 model". Out of 217 real photographs, the model successfully classified 205 as real and 12 as phony. It properly detected every 192 fake test images. The model had an overall accuracy of 94%. Figure 11(c) displays the confusion matrix for DenseNet201. Among the 192 test images classified as fake, this model correctly predicted all of them. On the other hand, out of the 217 real images provided to the model, it classified 214 as real and 3 as fake. The model achieved an overall accuracy of 99%.

5.5. Comparative analysis

Table 7 shows the comparison of three different deep learning models (VGG-19, InceptionV3, and DenseNet201) for detecting whether a photograph is authentic or not. The "precision", "F1-score", "recall", and accuracy of each model for differentiating between actual and false photos are listed in the table. F1-score is a harmonic average of precision and recall; recall assesses the capacity to recognize real positives; accuracy reflects overall performance. Precision reflects the correctness of positive predictions. For VGG-19,

the precision for detecting real images is 83%, and for fake images is 81%, while recall is 83% for images, both real and phony and accuracy is 82%. InceptionV3 has a higher precision of 95% for real images and 94% for fake images and recall is 94% for both real and fake images, with an accuracy of 94%. DenseNet201 has the highest precision of 100% for real images and 98% for fake images and recall is 99% for both real and fake images, with an accuracy of 99%.

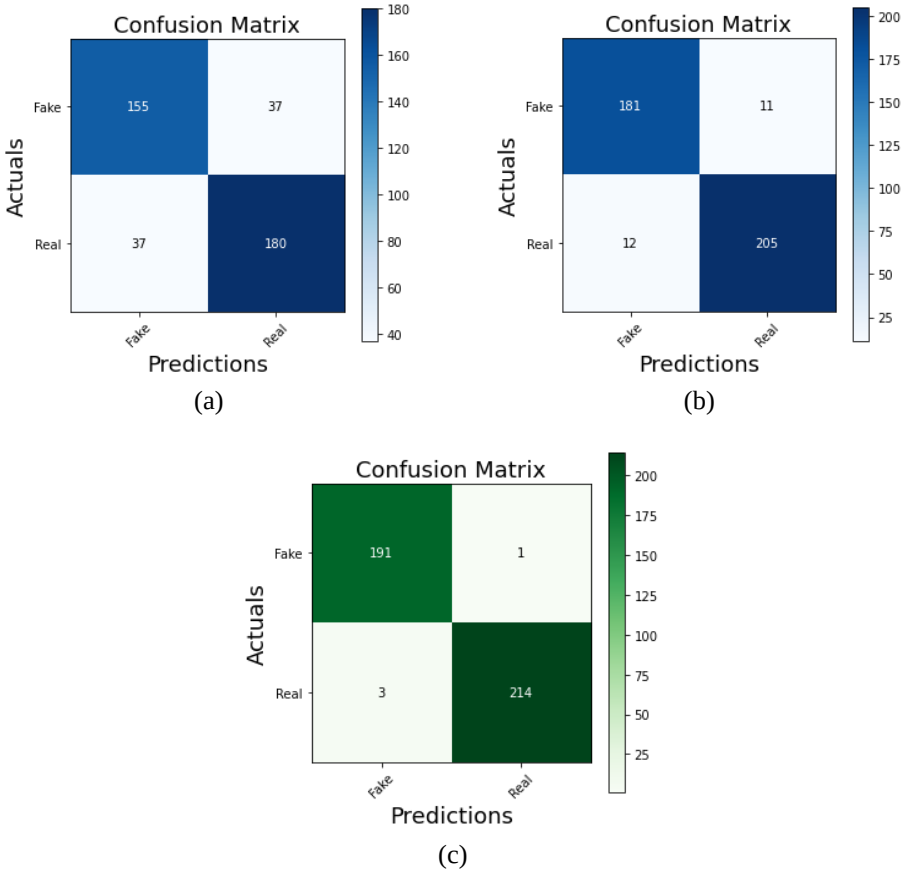

Figure 11. Confusion matrix of (a) VGG-19, (b) InceptionV3, and (c) DenseNet201

Table 7. Comparison of the models

		Precision	Recall	F1-score	Support
VGG-19	Real	83	83	83	82
	Fake	81	81	81	
InceptionV3	Real	95	95	94	94
	Fake	95	95	94	
DenseNet201	Real	100	99	99	99
	Fake	98	99	99	

6. CONCLUSION

In this paper, the deep learning algorithms have shown potential for the categorization of face fake images. These algorithms have been applied to picture data and have exhibited good results both accurate and efficient. The comparison of VGG-19, InceptionV3, and DenseNet201 deep learning models for detecting real and fake images shows that all models perform well, with DenseNet201 demonstrating the highest precision, recall, and accuracy. These findings suggest that DenseNet201 is a promising model for image classification tasks and could be useful in applications like detecting fake images. InceptionV3 and DenseNet201 exhibit higher precision scores and lower false-positive rates. DenseNet201 has the highest recall and accuracy scores, indicating better detection of true positive cases. Further research is needed, but these results advance deep learning and image classification, with potential implications for various industries.

REFERENCES

[1] T. T. Nguyen *et al.*, "Deep learning for deepfakes creation and detection: a survey," *Computer Vision and Image Understanding*, vol. 223, p. 103525, Oct. 2022, doi: 10.1016/j.cviu.2022.103525.

[2] H. Chen, G. Hu, Z. Lei, Y. Chen, N. M. Robertson, and S. Z. Li, "Attention-based two-stream convolutional networks for face spoofing detection," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 578–593, 2020, doi: 10.1109/TIFS.2019.2922241.

[3] L. Verdoliva, "Media forensics and deepfakes: an overview," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 5, pp. 910–932, Aug. 2020, doi: 10.1109/JSTSP.2020.3002101.

[4] C. Chen, S. Zhang, F. Lan, and J. Huang, "Domain generalization for document authentication against practical recapturing attacks," Jan. 2021, [Online]. Available: http://arxiv.org/abs/2101.01404

[5] L. Kang and X. Cheng, "Copy-move forgery detection in digital image," in *2010 3rd International Congress on Image and Signal Processing*, IEEE, Oct. 2010, pp. 2419–2421. doi: 10.1109/CISP.2010.5648249.

[6] E. Ardizzone, A. Bruno, and G. Mazzola, "Detecting multiple copies in tampered images," in *2010 IEEE International Conference on Image Processing*, IEEE, Sep. 2010, pp. 2117–2120. doi: 10.1109/ICIP.2010.5652490.

[7] A. Kaur and R. Sharma, "Copy-move forgery detection using DCT and SIFT," *International Journal of Computer Applications*, vol. 70, no. 7, pp. 30–34, May 2013, doi: 10.5120/11977-7847.

[8] S. Agarwal, W. Fan, and H. Farid, "A diverse large-scale dataset for evaluating rebroadcast attacks," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, Apr. 2018, pp. 1997–2001. doi: 10.1109/ICASSP.2018.8462205.

[9] T. de F. Pereira, A. Anjos, J. M. De Martino, and S. Marcel, "Can face anti-spoofing countermeasures work in a real world scenario?," in *2013 International Conference on Biometrics (ICB)*, IEEE, Jun. 2013, pp. 1–8. doi: 10.1109/ICB.2013.6612981.

[10] T. de F. Pereira *et al.*, "Face liveness detection using dynamic texture," *EURASIP Journal on Image and Video Processing*, vol. 2014, no. 1, p. 2, Dec. 2014, doi: 10.1186/1687-5281-2014-2.

[11] Z. Junhong, "Detection of copy-move forgery based on one improved LLE method," in *2010 2nd International Conference on Advanced Computer Control*, IEEE, 2010, pp. 547–550. doi: 10.1109/ICACC.2010.5486861.

[12] A. Punnappurath and M. S. Brown, "Learning raw image reconstruction-aware deep image compressors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 4, pp. 1013–1019, Apr. 2020, doi: 10.1109/TPAMI.2019.2903062.

[13] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Energy compaction-based image compression using convolutional AutoEncoder," *IEEE Transactions on Multimedia*, vol. 22, no. 4, pp. 860–873, Apr. 2020, doi: 10.1109/TMM.2019.2938345.

[14] J. Chorowski, R. J. Weiss, S. Bengio, and A. van den Oord, "Unsupervised speech representation learning using WaveNet autoencoders," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 12, pp. 2041–2053, Dec. 2019, doi: 10.1109/TASLP.2019.2938863.

[15] S. Shackleford, A. Proia, B. Martell, and A. Craig, "Toward a global cybersecurity standard of care? exploring the implications of the 2014 NIST cybersecurity framework on shaping reasonable national and international cybersecurity practices," *Texas International Law Journal*, vol. 50, no. 2, pp. 303–353, 2015.

[16] Z. J. Barad and M. M. Goswami, "Image forgery detection using deep learning: a survey," in *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, IEEE, Mar. 2020, pp. 571–576. doi: 10.1109/ICACCS48705.2020.9074408.

[17] Y. Mirsky and W. Lee, "The creation and detection of deepfakes," *ACM Computing Surveys*, vol. 54, no. 1, pp. 1–41, Jan. 2022, doi: 10.1145/3425780.

[18] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," Sep. 2014, [Online]. Available: http://arxiv.org/abs/1409.1556

[19] C. Szegedy *et al.*, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2015, pp. 1–9. doi: 10.1109/CVPR.2015.7298594.

[20] I. Chingovska, A. Anjos, and S. Marcel, "On the effectiveness of local binary patterns in face anti-spoofing," *Proceedings of the International Conference of the Biometrics Special Interest Group, BIOSIG 2012*, 2012.

[21] Z. Yu, Y. Qin, X. Li, C. Zhao, Z. Lei, and G. Zhao, "Deep learning for face anti-spoofing: a survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–22, 2022, doi: 10.1109/TPAMI.2022.3215850.

[22] W. Kim, S. Suh, and J.-J. Han, "Face liveness detection from a single image via diffusion speed model," *IEEE Transactions on Image Processing*, vol. 24, no. 8, pp. 2456–2465, Aug. 2015, doi: 10.1109/TIP.2015.2422574.

[23] J. Yang, Z. Lei, and S. Z. Li, "Learn convolutional neural network for face anti-spoofing," Aug. 2014, [Online]. Available: http://arxiv.org/abs/1408.5601

[24] D. Menotti *et al.*, "Deep representations for iris, face, and fingerprint spoofing detection," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 4, pp. 864–879, Apr. 2015, doi: 10.1109/TIFS.2015.2398817.

[25] A. Alotaibi and A. Mahmood, "Deep face liveness detection based on nonlinear diffusion using convolution neural network," *Signal, Image and Video Processing*, vol. 11, no. 4, pp. 713–720, May 2017, doi: 10.1007/s11760-016-1014-2.

[26] K. Ito, T. Okano, and T. Aoki, "Recent advances in biometrie security: A case study of liveness detection in face recognition," in *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, IEEE, Dec. 2017, pp. 220–227. doi: 10.1109/APSIPA.2017.8282031.

[27] N. Kose and J.-L. Dugelay, "On the vulnerability of face recognition systems to spoofing mask attacks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, May 2013, pp. 2357–2361. doi: 10.1109/ICASSP.2013.6638076.

[28] N. Kose and J.-L. Dugelay, "Reflectance analysis based countermeasure technique to detect face mask attacks," in *2013 18th International Conference on Digital Signal Processing (DSP)*, IEEE, Jul. 2013, pp. 1–6. doi: 10.1109/ICDSP.2013.6622704.

[29] J. Maatta, A. Hadid, and M. Pietikainen, "Face spoofing detection from single images using micro-texture analysis," in *2011 International Joint Conference on Biometrics (IJCB)*, IEEE, Oct. 2011, pp. 1–7. doi: 10.1109/IJCB.2011.6117510.

[30] N. Kose and J.-L. Dugelay, "Countermeasure for the protection of face recognition systems against mask attacks," in *2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, IEEE, Apr. 2013, pp. 1–6. doi: 10.1109/FG.2013.6553761.

[31] X. Tan, Y. Li, J. Liu, and L. Jiang, "Face liveness detection from a single image with sparse low rank bilinear discriminative model," in *Computer Vision – ECCV 2010. ECCV 2010. Lecture Notes in Computer Science*, Berlin: Springer, 2010, pp. 504–517. doi: 10.1007/978-3-642-15567-3_37.

BIOGRAPHIES OF AUTHORS

Nishigandha Zanje is an M.Tech. student who specializes in artificial intelligence and machine learning. She is also employed by Search Company as a data engineer. Skilled in cloud, deep learning, and AI/ML. Strong education professional with a bachelor's degree focused in B.E. (Computer Science) from Dr. D.Y. Patil Institute of Technology, Pune. She can be contacted at email: zanje123n@gmail.com.

Anupkumar M. Bongale received his Ph.D. degree from Visvesvaraya Technological University (VTU), Belgaum, Karnataka, India. He is currently working as an associate professor with the Department of Artificial Intelligence and Machine Learning, Symbiosis Institute of Technology, Symbiosis International (Deemed University), Lavale, Pune, Maharashtra, India. He has filed a patent and has published book chapters. He also published several research articles in reputed international journals and conferences. He is a senior member of IEEE. His research interests include wireless sensor networks, machine learning, optimization techniques, and swarm intelligence. He can be contacted at email: anupkumar.bongale@sitpune.edu.in.

Dr. Deepak Dharrao currently working as an assistant professor in the department of CSE-IT at Symbiosis Institute of Technology, Lavale, Symbiosis International University. He has 15 years of experience in research and academics. He has authored or co-authored over 30 technical articles in refereed journals and conference proceedings. He is a member of IEEE, ACM, and a life member of ISTE professional organizations. His research interests include the areas of image and real-time video processing, data science, artificial intelligence, and machine learning. He can be contacted at email: deepak.dharrao@sitpune.edu.in.