DEEPFAKE **DETECTION**

Prepared By :

Omar Daniel Abou Assaf

Abstract

Deepfake detection is an essential artificial intelligence field focused on the development of mechanisms for identifying and controlling the spread of synthetic media – false audio, visual, or combined content produced employing advanced machine learning systems. The term is combined from “deep learning” and “fake” because it describes deep neural networks manipulated to create highly plausible still and video images and audio recordings. It is urgent to acquire a new perspective of methodologies for detecting the systems since the rapidly changing horizon of produced artificial contents demands the creation of strong and scalable detection models. An effective approach to solving complex challenges is to facilitate interdisciplinary research and collaboration to maintain the unity and trustworthiness of details and counter the spread of fallacious information.

Table Content

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Network
CSV	Comma-Separated Values
ReLU	Rectified Linear Unit
Adam	Adaptive Moment Estimation
GAN	Generative Adversarial Network
RMSP	Root Mean Square Propagation

Keywords

Dataset, Classification, Machine learning, Neural network, Deepfake.

LIST OF FIGUERS

LIST OF TABLES

Introduction:

The training of generative neural network designs, such as Autoencoders or GANs [1] [2], and deep learning constitute the foundation for the development of the deepfake approaches. for the intended good, which was to expand the dataset size for scientists studying machine learning and deep learning, but shortly after it was developed, a lot of people started using this approach for illegal activities, like: using Deepfake [3] to create media featuring any political figure and disseminate false information through it, commit financial fraud, create media by manipulating people's faces, and so forth.

Deepfake detection models have been trained to counteract misuse. in our project. We have specifically focused on detecting faces, as they pose the greatest risk due to their potential for forgery and identity impersonation, to achieve that we used CNN, among the methods proposed for Deepfake detection, convolutional neural networks (CNNs) have been a popular choice as they have a special ability to extract features from the image. We also used pretrained models such as Xception, Inception-ResNet-v2.

Dataset Preprocessing and visualization:

This dataset combines 70,000 actual facial photos drawn from the Flickr collection curated by Nvidia, and an equal number of synthetic facial images collected from the 1 million FAKE faces dataset generated by StyleGAN, donated by Bojan. The dataset unites both actual and synthetic data, each reduced to 256 pixels, and rigorously divided into training, validation, and test sets for ease of analysis. Additionally, associated CSV files have been rigorously produced to assist seamless data manipulation and interpretation. This merged dataset serves as a great resource for researchers and practitioners interested in deepfake detection and related studies, delivering a complete repository of different facial imagery for algorithmic training and evaluation [4].

Exploratory Data Analysis:

The proportion of data in each subset is shown graphically in a pie chart, with the Train subset holding the biggest part (71.4%). Validation and Test, the two remaining subsets, each make up 14.3%. Figure 1 distribution of images.

Each of the three vertical bars denotes a different dataset category:

- Train: This category has up to 100,000 photos, the most of any other category.
- Validation: There are about 20,000 photos in the validation collection.
- Test: There are about 20,000 photos in the testing collection as well. Figure 2 distribution of images in labels

The dataset is complemented by three CSV files: Train, Validation, and Test. These files contain standardized data representations for each subset, enabling easy data handling during model training, validation, and testing. They ease preprocessing, structuring, and integration with the ImageDataGenerator module, allowing for complete analysis and interpretation of model findings. Figure 3

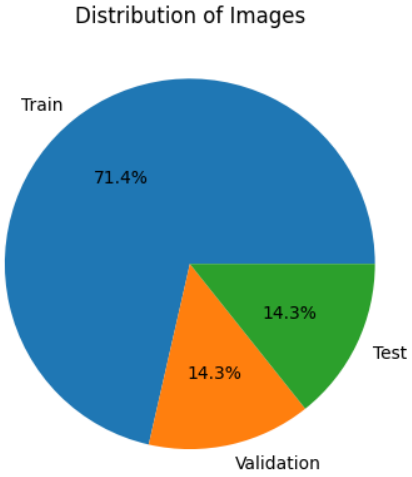


Figure 1: distribution of images

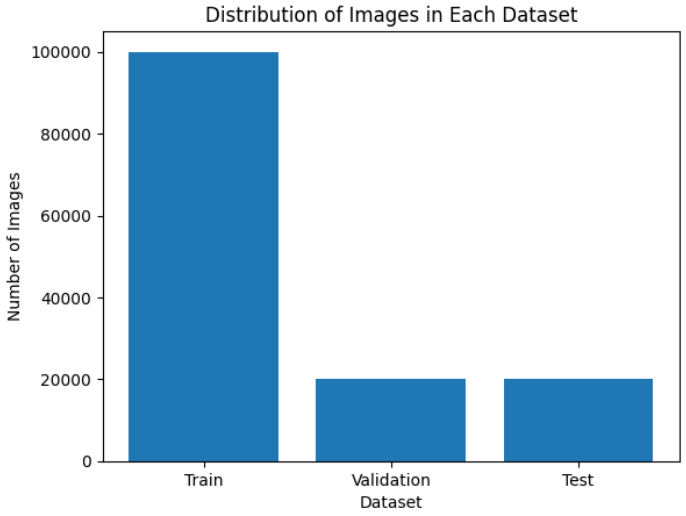


Figure 2: distribution of images in labels

original_path	id	label	label_str	path
100000 unique values	0 – 70.0k	0 – 1	2 unique values	100000 unique values
/kaggle/input/flickrfaceshq-dataset-nvidia-part-7/images1024x1024-20191222T221133Z-035/images1024x10...	31355	1	real	train/real/31355.jpg
/kaggle/input/flickrfaceshq-dataset-nvidia-part-1/images1024x1024-20191222T221133Z-004/images1024x10...	02884	1	real	train/real/02884.jpg
/kaggle/input/flickrfaceshq-dataset-nvidia-part-4/images1024x1024-20191222T221133Z-019/images1024x10...	33988	1	real	train/real/33988.jpg

Figure 3: Describe the CSV file

Figures 4 and 5 show a selection of the dataset's photos. Figure 4 is a fake sample, whereas Figure 5 is a real sample. The kind of data used to train and assess the deepfake detection model is demonstrated by these examples.

Figure 4: Fake sample

Figure 5: Real sample

Preprocessing:

ImageDataGenerator The ImageDataGenerator module plays an essential role in the preparation pipeline for the deepfake detection presented in this paper given the study's context. This module, a potent part of the TensorFlow framework, allows picture data to be appropriately managed and augmented, which increases their diversity and durability, eventually leading to an improved model. The ImageDataGenerator does so by applying several changes to the input photos to enrich the dataset with variant copies of the original pictures, as previously explained. The changes in question include rescaling, rotation, flipping, shear, zoom, and shift, among others. By applying such changes occasionally, the ImageDataGenerator produces a dataset that is more representative of the original data, hence reducing the risk of overfitting and allowing the model to better generalize unknown data [5].

Resolution rescaling is one of the most important functions of ImageDataGenerator, as it scales the pixel values the images share to some defined scale, usually [0, 1]. This weights the pixel intensities shared by all the images scale down to some set scale for equality during the training of the model. As a result, all images share pixel intensities, mainly to avoid change, which can change dramatically due to changes in light or camera settings. Furthermore, the ImageDataGenerator facilitates the production of batches of enhanced images on-the-fly during

model training. This dynamic production of augmented batches guarantees that the model receives a diversified set of training examples in each iteration, so exposing the model to a wider range of data instances and enhancing its capacity to learn significant patterns from the data. Furthermore, the ImageDataGenerator easily interfaces with structured data abstractions, for example, dataframes with image paths and their respective labels, which facilitates efficient data access and utilization in model training and validation. As a result, the ImageDataGenerator searches and preprocess photographs in batches from the dataset and provides to the model during training. As a result, the workload of retrieving and preparing data on-the-go is greatly decreased.

Overall, the ImageDataGenerator module performs a key function in preparing and supplementing picture data for deepfake detection model training. By meticulously applying adjustments and generating enhanced batches of photographs, the ImageDataGenerator enhances the diversity, quality, and usefulness of the dataset, ultimately aiding to the building of more durable and effective deep learning models for deepfake detection.

Model:

CNN:

CNNs are a class of deep neural networks designed to analyze structured grid-like input, primarily images. They are made up of layers of convolutional and pooling procedures, followed by fully connected layers for classification. CNNs are recognized for their abilities to automatically uncover hierarchical patterns from raw pixel data, making them excellent for classification applications such as image recognition. However, while LeNet, AlexNet, and VGG16 are standard CNN models, new CNN [6] architecture could be designed to fulfill the objectives of deepfake detection.

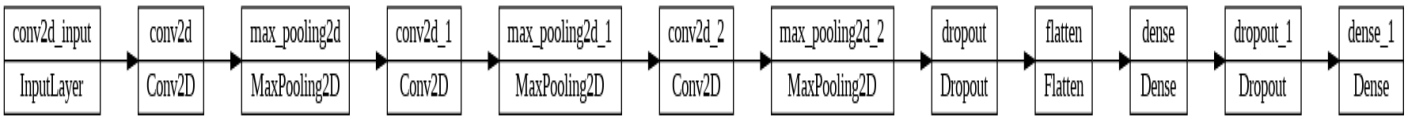


Figure 6: CNN architecture

```
model_cnn = Sequential()

model_cnn.add(Conv2D(32,kernel_size=(3, 3), activation='relu', input_shape=(256, 256, 3)))
model_cnn.add(MaxPooling2D((2, 2)))
model_cnn.add(Conv2D(64, (3, 3), activation='relu'))
model_cnn.add(MaxPooling2D((2, 2)))
model_cnn.add(Conv2D(64, (3, 3), activation='relu'))
model_cnn.add(MaxPooling2D((2, 2)))
model_cnn.add(Dropout(0.2))
model_cnn.add(Flatten())
model_cnn.add(Dense(64, activation='relu'))
model_cnn.add(Dropout(0.2))
model_cnn.add(Dense(1, activation='sigmoid'))
```

Compile and fit:

```
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
```

```
history = model.fit(train_data_generator ,epochs = 7,validation_data= (validation_data_generator))
```

Xception:

Xception is an architecture developed by François Chollet in 2017 based on the concept of depthwise separable convolutions. The rationale of this methodology is to capture spatial and channel-wise relationships separately. In turn, it allows developing more effective feature extractors while keeping the model's parameter count low. Due to its efficiency, Xception has become the state-of-the-art-like solution in various computer vision workloads. Additionally, its modular nature and cost-efficient architecture make it the optimal solution for workloads with limited computational resources, such as deepfake detection on edge or mobile devices [7].
Figure 7

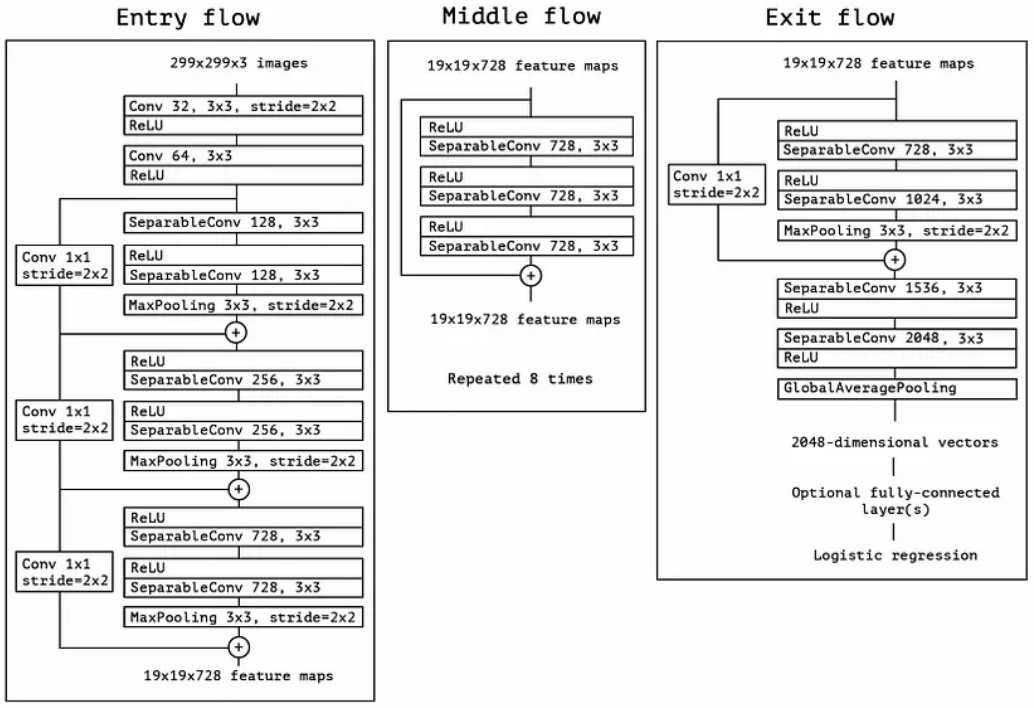


Figure 7: Xception architecture [8]

```
base_model = Xception(weights='imagenet',include_top=False,input_shape=(256, 256, 3))

for layer in base_model.layers:
    layer.trainable = False

Xception_model = Sequential()
Xception_model.add(base_model)
Xception_model.add(Flatten())
Xception_model.add(BatchNormalization())
Xception_model.add(Dense(256, activation='relu'))
Xception_model.add(Dropout(0.5))
Xception_model.add(Dense(128, activation='relu'))
Xception_model.add(Dropout(0.5))
Xception_model.add(BatchNormalization())
Xception_model.add(Dense(1, activation='sigmoid'))

Xception_model.summary()
```

Compile and fit:

```
Xception_model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
history = Xception_model.fit(train_data_generator ,epochs = 10,validation_data= (validation_data_generator))
```

Inception-ResNet-v2:

Inception-ResNet-v2 is a convolutional neural network blueprint that extends the Inception architecture to accommodate remnant connections. The leftover connections replace the filter concatenation stage of the Inception architecture. The system is optimized to construct and maximize the learning process and output in photo categorization operations. These are the crucial details for the Inception-ResNet-v2. Architecture The Inception-ResNet-v2 is developed by merging components from Inception and ResNet design protocols. The idea is to use a combination of multi-scale convolutions and remarkable linkages to catch more intricate features. Depth The depth of the Inception-ResNet-v2 is 164. This is a truly lengthy and wide network that will fit more complicated patterns in images [9]. Figure 8

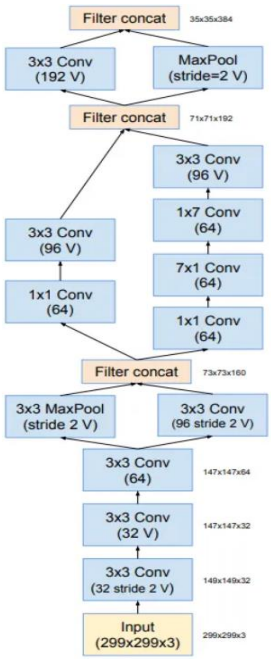


Figure 8: display the input part of Inception ResNet v2 [10]

```
base_model_2 = InceptionResNetV2(weights='imagenet', include_top=False, input_shape=(256, 256, 3))

for layer in base_model_2.layers:
    layer.trainable=False

x = base_model_2.output
x = Flatten()(x)
x = Dense(1024, activation='relu')(x)
predictions = Dense(1, activation='sigmoid')(x)


model = Model(inputs=base_model_2.input, outputs=predictions)
model.summary()
```

```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

```
history = model.fit(train_data_generator,epochs = 10,steps_per_epoch = len(train_data_generator),
    validation_data = validation_data_generator,validation_steps = len(validation_data_generator))
```

We trained the three models using:

ReLU:

ReLU activation function works by applying a simple mathematical operation to the input value. If the input value is greater than or equal to zero, the output is equal to the input value. If the input value is negative, the output is zero [12]. Mathematically, the ReLU function can be represented as:

$$f(x) = max(0, x)$$

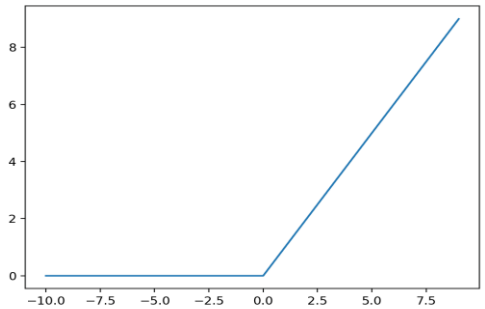


Figure 9: ReLU function

Sigmoid function:

is a mathematical function with a characteristic "S"-shaped curve or sigmoid curve. It transforms any value to a number between 0 and 1 [13].

Mathematical definition**:**

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

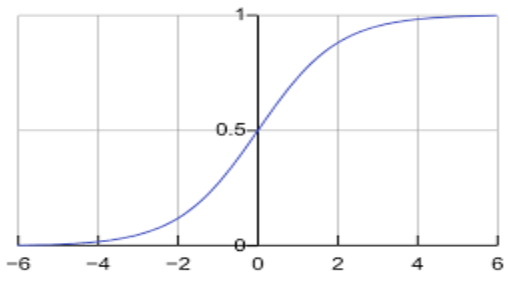


Figure 10: Sigmoid function

Binary Cross Entropy (or Log Loss):

is a commonly used loss function in binary classification problems. It measures the difference between two probability distributions, typically the predicted probabilities and the actual binary labels[14]

$$L(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} \left(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right)$$

Adam Optimizer:

is an algorithm for optimization technique for gradient descent. The method is really efficient when working with problems involving a lot of data or parameters. It requires less memory and is efficient. Intuitively, it is a combination of the 'gradient descent with momentum' algorithm and the 'RMSP' algorithm [15].

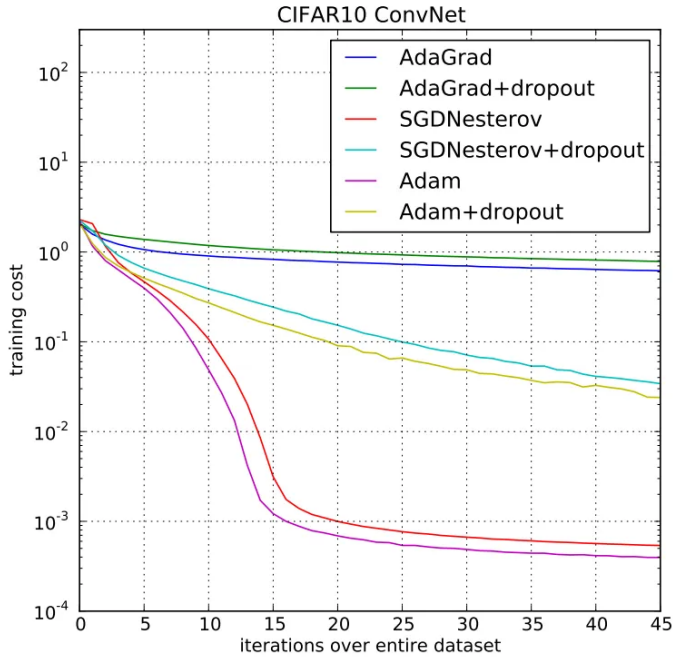


Figure 11: Comparison between the training cost of different optimizers [11]

Result

CNN model:

Model evaluation:

```
test_loss, test_accuracy = model_cnn.evaluate(test_data_generator)
print(f'Loss: {test_loss}, Accuracy: {test_accuracy}')
```

```
625/625 [==============================] - 71s 113ms/step - loss: 0.0773 - accuracy: 0.9767
Loss: 0.07729087769985199, Accuracy: 0.9767000079154968
```

confusion matrix and plot:

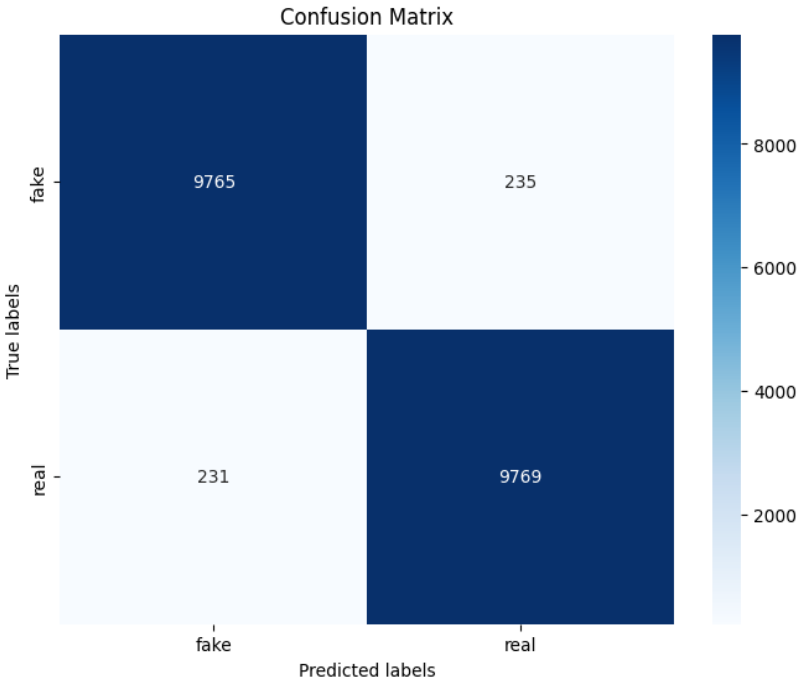


Figure 12: CNN Confusion matrix

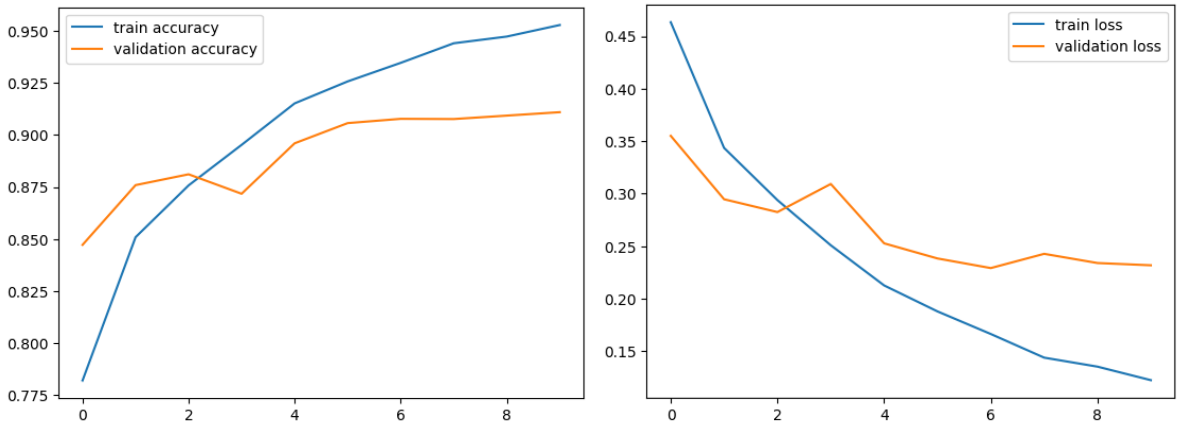


Figure 13: CNN plot (accuracy and loss)

Classification Report:	precision	recall	f1-score	support
fake	0.98	0.98	0.98	10000
real	0.98	0.98	0.98	10000
accuracy			0.98	20000
macro avg	0.98	0.98	0.98	20000
weighted avg	0.98	0.98	0.98	20000

Figure 14: Report for CNN model

Xception model:

Model evaluation:

```
test_loss, test_accuracy = Xception_model.evaluate(test_data_generator)
print(f'Loss: {test_loss}, Accuracy: {test_accuracy}')
```

```
625/625 [==============================] - 105s 168ms/step - loss: 0.2341 - accuracy: 0.9135
Loss: 0.2341022491455078, Accuracy: 0.9135000109672546
```

confusion matrix and plot:

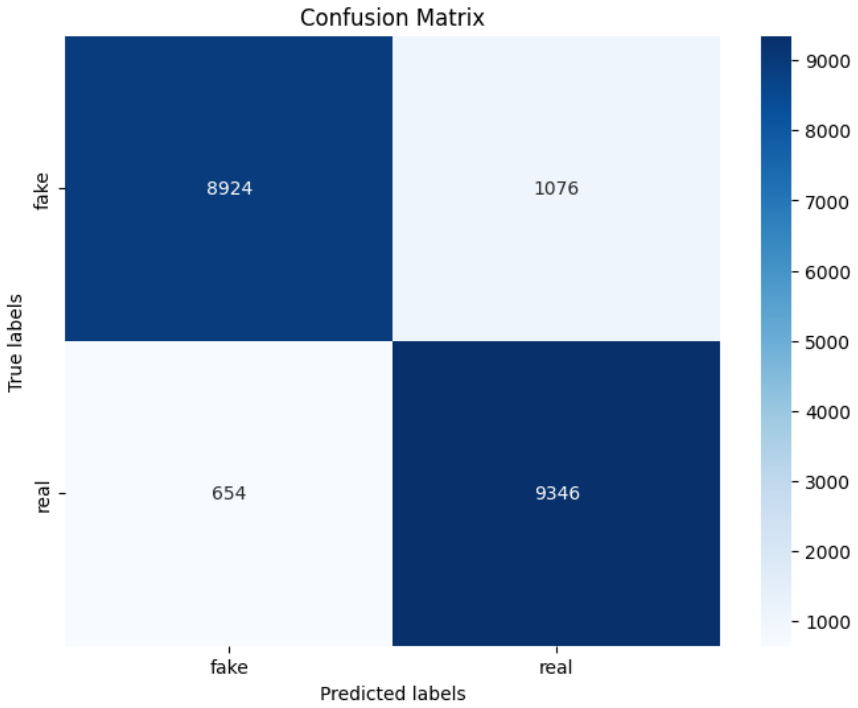


Figure 15: Xception Confusion matrix

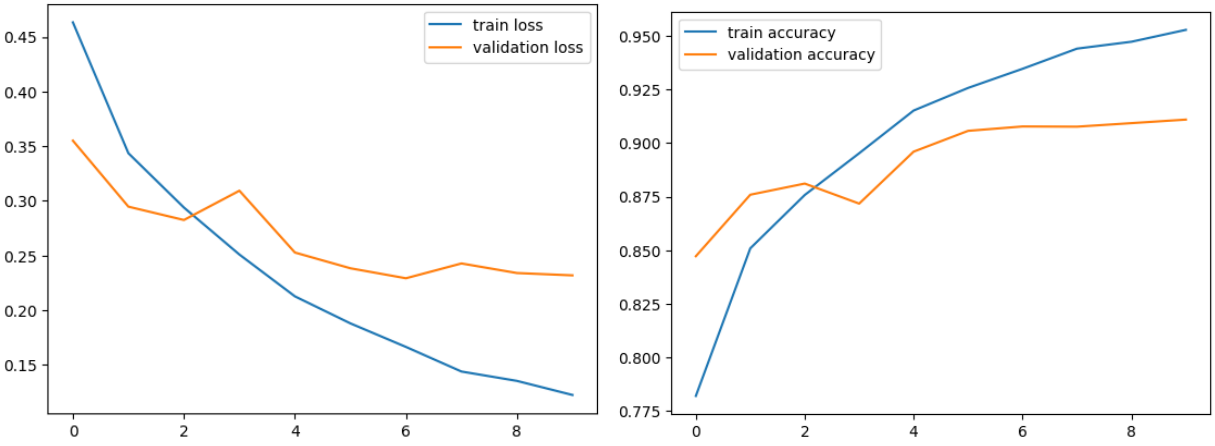


Figure 16: Xception plot (accuracy and loss)

```
Classification Report:
              precision    recall  f1-score   support

        fake       0.93      0.89      0.91     10000
        real       0.90      0.93      0.92     10000

    accuracy                           0.91     20000
   macro avg       0.91      0.91      0.91     20000
weighted avg       0.91      0.91      0.91     20000
```

Figure 17: Report for Xception model

InceptionResNetV2 model:

Model evaluation:

```
test_loss, test_accuracy = model.evaluate(test_data_generator)
print(f'Loss: {test_loss}, Accuracy: {test_accuracy}')
```

```
625/625 [==============================] - 142s 227ms/step - loss: 0.2554 - accuracy: 0.9269
Loss: 0.25540289282798767, Accuracy: 0.9269499778747559
```

confusion matrix and plot:

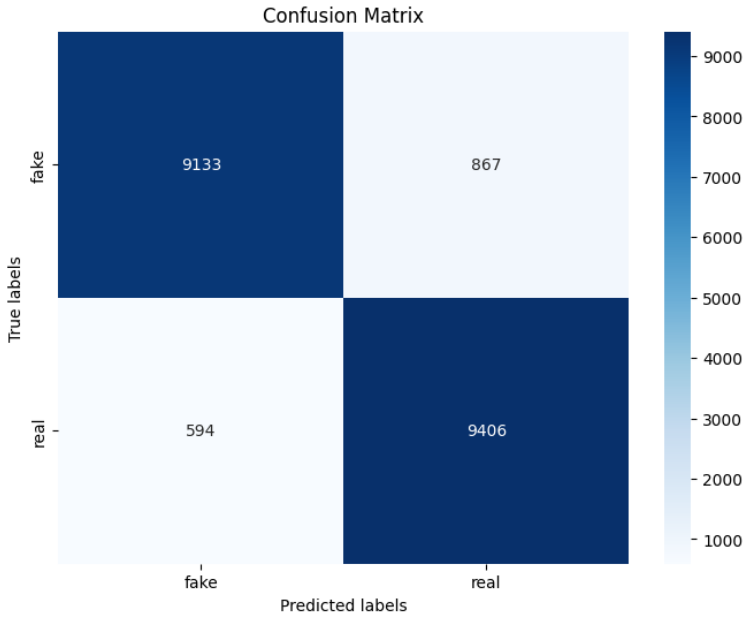


Figure 18: InceptionResNetV2 Confusion matrix

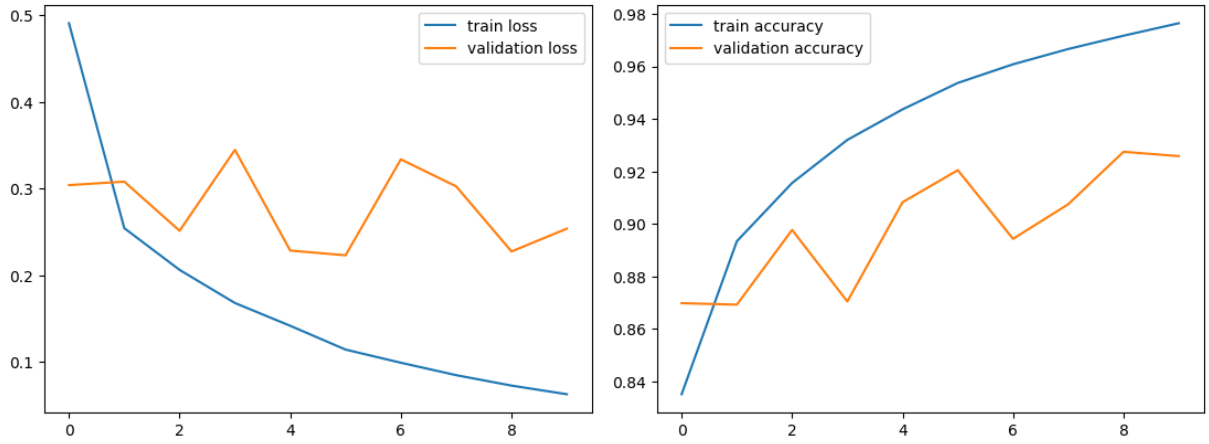


Figure 19: InceptionResNetV2 plot (accuracy and loss)

```
Classification Report:
              precision    recall  f1-score   support

        fake       0.94      0.91      0.93     10000
        real       0.92      0.94      0.93     10000

    accuracy                           0.93     20000
   macro avg       0.93      0.93      0.93     20000
weighted avg       0.93      0.93      0.93     20000
```

Figure 20: Report for Xception model

Discussion:

Model	Training accuracy	Training loss	Validation accuracy	Validation loss	Training accuracy	Training loss
CNN	**0.9815**	**0.0548**	**0.9788**	**0.0714**	**0.9767**	**0.0772**
Xception	0.9528	0.1224	0.9110	0.2319	0.9135	0.2341
InceptionResNetV2	0.9765	0.0629	0.9258	0.2540	0.9269	0.2554

Table 1: Comparison of models' performance

As a result of the analyzed confusion matrix as well as the generated plots, a distinct trend is evident: the Convolutional Neural Network has exhibited markedly better performance than

others. This trend is especially noticeable from the vital metrics' standpoint such as that of accuracies, losses, and the confusion matrix, and Through experiments, CNN showed good performance in classification Moreover, the continuous lower values of loss indicate low error, further indicating the CNN model's capability to optimize its prediction patterns.

Compare

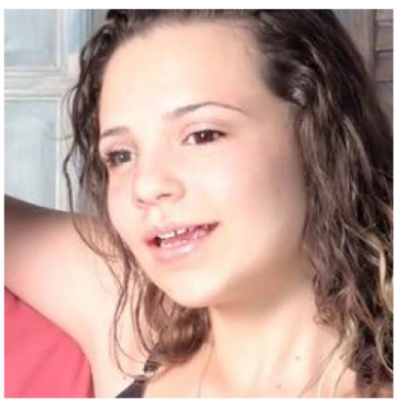
Compare between my model CNN and other paper using CNN [16]:

	Validation Accuracy	Training Accuracy	Precision	Recall	F1 Score
CNN	0.9788	0.9815	0.98	0.98	0.98
DeepFake Image Detection using CNN	0.9141	0.9375	0.937516	0.9564	0.9468

Table 2: Compare between model CNN and paper

Testing:

These are some of the samples. We have conducted several trials on a set of images derived from another dataset, where the model has classified fake images as fake and real images as real.



1/1 [==============================] - 0s 43ms/step
Predicted label: fake

1/1 [==============================] - 0s 50ms/step
Predicted label: fake

1/1 [==============================] - 0s 21ms/step
Predicted label: fake

Conclusion:

The topic of deepfake detection continues to be driven by the desire for accuracy and efficiency. The overview of this difficult topic underlines the importance of a multilayer strategy that integrates multiple diverse detection methodologies, from neural network analysis to face biometrics. Nonetheless, it is vital to remember that deepfake technology will continue to change swiftly, presenting restrictions for detection tactics. Moreover, the considerations of false positives and the risks of misuse suggest that guaranteeing the openness and accountability of detection tools is of major importance.

In conclusion, our high validation accuracy of 97.88% and training accuracy of 98.15% demonstrate the strength of our detection model in distinguishing real media from deepfake. In addition, achieving a precision, recall, and F1 of scores of 98% on all modules further establishes the high confidence in our model identification and classification of any manipulated content. Overall, creating fertile ground for researchers, industry players, and policymakers to collaborate promises a more effort in building stronger defenses against the ever-rising invasion of deepfakes. Such contributions are essential in keeping the integrity of our digital conversation safe from the many negative ripples that come with the emergence of such content.

Reference:

[1] Ian J Jean, Mehdi, Bing, & David. (2014, June 10). Generative Adversarial Networks. In https://arxiv.org/abs/1406.2661.

[2] .Francesco Marra, Diego Gragnaniello, Davide Cozzolino,Luisa Verdoliva, “Detection of GANGenerated Fake Images over Social Networks”, IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), 10-12 April 2018.

[3] ThanhThi Nguyen, Cuong M. Nguyen, Dung Tien Nguyen, DucThanh Nguyen, SaeidNahavandi, “Deep Learning for Deepfakes Creation and Detection”, arXiv:1909.11573 [cs.CV], 25 Sep 2019.

[4]*140k Real and Fake Faces*. (2020, February 4). Kaggle. Retrieved May 3, 2024, from https://www.kaggle.com/datasets/xhlulu/140k-real-and-fake-faces

[5] *ImageDataGenerator*. (2024, May 27). TensorFlow. Retrieved May 3, 2024, from https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

[6] Karen Simonya[n and Andrew Zisserman, “Very Deep Convolutional Networks for Large – Scale Image Recognition”, ICLR 2015, arXiv:1409.1556v6 [cs.CV] 10 Apr 2015

[7] A. V and P. T. Joy, "Deepfake Detection Using XceptionNet," 2023 IEEE International Conference on Recent Advances in Systems Science and Engineering (RASSE), Kerala, India, 2023.

[8] Sarkar, A. (2023, May 19). *Xception: Implementing from scratch using Tensorflow*. Medium. https://towardsdatascience.com/xception-from-scratch-using-tensorflow-even-better-than-inception-940fb231ced9

[9] *Papers with Code - Inception-ResNet-v2 Explained*. (n.d.). https://paperswithcode.com/method/inception-resnet-v2

[10] Raj, B. (2020, July 31). *A Simple Guide to the Versions of the Inception Network*. Medium. https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202

[11] Brownlee, J. (2020, August 20). *A Gentle Introduction to the Rectified Linear Unit (ReLU)*. MachineLearningMastery.com. https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

[12] Saeed, M. (2021, August 17). *A Gentle Introduction to Sigmoid Function*. MachineLearningMastery.com. https://machinelearningmastery.com/a-gentle-introduction-to-sigmoid-function/

[13] Zhang, & R. Sabuncu. (2018, May 20). Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels. In *https://arxiv.org/abs/1805.07836*.

[14] Marti, K. (2005, January 1). *Stochastic Optimization Methods*. Springer Science & Business Media. http://books.google.ie/books?id=eDf3tjjdUJoC&printsec=frontcover&dq=Adam:+A+Method+for+Stochastic+Optimization&hl=&cd=1&source=gbs_api

[15] Alabdullatef, L. (2024, January 4). *Complete Guide to Adam Optimization - Layan Alabdullatef - Medium*. Medium. https://medium.com/@LayanSA/complete-guide-to-adam-optimization-1e5f29532c3d

[16] Salpekar. (2020). *DeepFake Image Detection*. Retrieved May 3, 2024, from http://cs230.stanford.edu/projects_spring_2020/reports/38857501.pdf