PROFESSIONAL TRAINING REPORT

at

Sathyabama Institute of Science and Technology

(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

DUBAGUNTLA VENU LAKSHMI SAI
(Reg. No – 42110321)

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

CATEGORY- 1 UNIVERSITY BY UGC

Accredited "A++" by NAAC I Approved by AICTE

JEPPIAAR NAGAR, RAJIV GANDHI SALAI CHENNAI - 600119

OCTOBER - 2024

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Category - I University by UGC
Accredited "A++" by NAAC | Approved by AICTE
www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Professional Training-1 Report is the bonafide work of **DUBAGUNTLA VENU LAKSHMI SAI (42110321)** who carried out the Project entitled **"DEEPFAKE DETECTION USING MACHINE LEARNING"** under my supervision from June 2024 to October 2024.

Internal Guide

Dr. G. NAGARAJAN, M.E., Ph.D.,

Head of the Department

Dr. L. LAKSHMANAN, M.E., Ph.D.,

Submitted for Interdisciplinary Viva Voce Examination held on ___________

Internal Examiner **External Examiner**

DECLARATION

I, **DUBAGUNTLA VENU LAKSHMI SAI (Reg. No-42110321),** hereby declare that the Professional Training-1 Report entitled **"DEEPFAKE DETECTION USING MACHINE LEARNING"** done by me under the guidance of **Dr. G.NAGARAJAN, M.E., Ph.D.,** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE:

PLACE: Chennai **SIGNATURE OF THE CANDIDATE**

ACKNOWLEDGEMENT

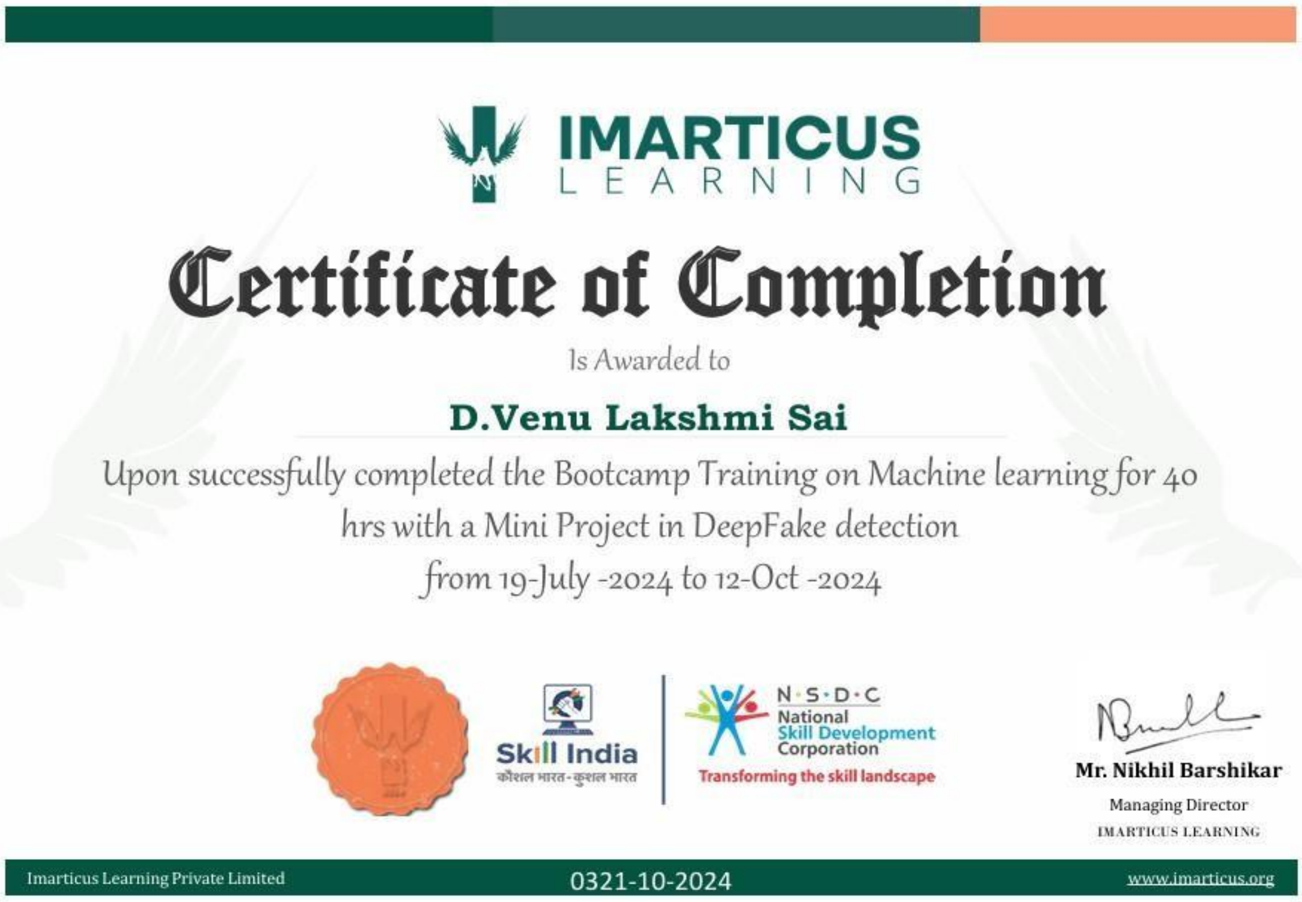
I am pleased to acknowledge my sincere thanks to **BOARD OF MANAGEMENT** of **Sathyabama Institute of Science and Technology** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph. D.**, **Dean**, School of Computing, and **Dr. L. Lakshmanan, M.E., Ph.D., Head of the Department** of Computer Science and Engineering for providing me with necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. G. NAGARAJAN, B.E., M.E., Ph.D.**, for his valuable guidance, suggestions, and constant encouragement paved the way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATE



IMARTICUS LEARNING

Certificate of Completion

Is Awarded to

D.Venu Lakshmi Sai

Upon successfully completed the Bootcamp Training on Machine learning for 40 hrs with a Mini Project in DeepFake detection from 19-July -2024 to 12-Oct -2024

Skill India
कौशल भारत-कुशल भारत

N·S·D·C
National Skill Development Corporation
Transforming the skill landscape

[illegible]

Mr. Nikhil Barshikar
Managing Director
IMARTICUS LEARNING

Imarticus Learning Private Limited

0321-10-2024

www.imarticus.org

ABSTRACT

This project addresses the growing challenge of identifying deepfake images, leveraging advanced machine learning and computer vision techniques. Deepfakes have emerged as a major concern in the digital age, with applications ranging from entertainment to malicious misinformation. In response to this, the project focuses on building a robust image classification system using Convolutional Neural Networks (CNN) to differentiate between authentic (real) and manipulated (fake) images. The system is implemented using the TensorFlow and Keras frameworks, which provide a flexible and scalable environment for deep learning model development. The project workflow includes several critical stages, beginning with data acquisition and pre-processing. The dataset used for training and testing the model consists of a diverse collection of real and fake facial images, stored in different categories. The images are pre-processed using the OpenCV library, ensuring consistency in dimensions and quality, which are key factors in enhancing model performance. Once the data is prepared, it is split into training and testing sets using the train_test_split function from the Scikit-learn library to ensure the model is evaluated on unseen data. The CNN architecture is composed of multiple convolutional and max-pooling layers, which progressively extract higher-level features from the input images. These layers are followed by fully connected layers and a softmax output layer for binary classification, indicating whether an image is real or fake. To optimize the learning process, the Adam optimizer is employed, as it combines the advantages of both the Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp), allowing for efficient learning with minimal tuning. The binary cross-entropy loss function is chosen to evaluate the performance during the training process, as it is well-suited for binary classification problems. Moreover, various hyperparameters, such as learning rate and batch size, are fine-tuned to maximize accuracy and minimize overfitting. The project also places a strong emphasis on performance evaluation. The trained model is assessed using accuracy metrics provided by Scikit-learn, and further analysis is conducted using confusion matrices to gauge the model's precision, recall, and F1 score. Data augmentation is to increase the model's robustness against variations in input images.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW

Image and video forgery are posing a threat to the society in today's world. People can artificially create any audio or video clip. Artificial intelligence, mainly machine learning, manipulates images and videos in such a way that they are often visually indistinguishable from real ones . There are some prevalent techniques which are widely used to manipulate images/videos. Some are computer graphic based (e.g. Photoshop, GIMP, and Canva) and the rest are content changing. Deepfake, a deep learning-based method, is a serious contender among the content changing video falsification techniques. The term ―deepfake‖ originates from the words ―deep learning‖ and ―fake‖. Use of deep learning networks (DNN) has made the process of creating convincing fake images and videos increasingly easier and faster. It is a technique in which a video or image of a person is manipulated by the image of another person using deep learning methods

In today's life, social network/media play a significant role. They can affect someone's mental health and social status, though it shouldn't be that way. Mobile or mini cameras let people take pictures or videos anywhere, anytime. Commercial photo editing tools allow anyone to create fake images/videos So, amid multimedia forgery, we need some counter measures to protect our privacy and identity, especially in social media where a person is vulnerable In social media, when images or videos are uploaded, they get compressed and resized. So the techniques applicable to uncompressed videos might not work for highly compressed videos. In this paper, we are proposing a novel method to detect deepfake videos in compressed social media.

Deepfakes, generated using artificial intelligence and deep learning techniques, are hyper-realistic synthetic media—typically videos or images—where a person's likeness is replaced with someone else's. While initially considered novel, deepfakes have rapidly become a significant societal concern due to their potential for malicious use, such as creating false information, spreading fake news, identity theft, and even damaging personal reputations. This has led to a pressing need for effective detection systems.

To further improve the system's efficacy, Transfer Learning techniques, employing pre-trained models like VGG16, ResNet, or EfficientNet, can be applied. These models, having been trained on large datasets, provide a strong foundation for deepfake detection, requiring only fine-tuning to adapt to domain-specific tasks. The incorporation of video-based detection methods could also be an extension, where temporal inconsistencies between frames are analyzed.

By taking advantage of advanced machine learning techniques, such as Convolutional Neural Networks (CNNs), this approach aims to build a robust system capable of detecting deepfake images with high accuracy, thereby addressing the growing challenges of digital misinformation and media manipulation.

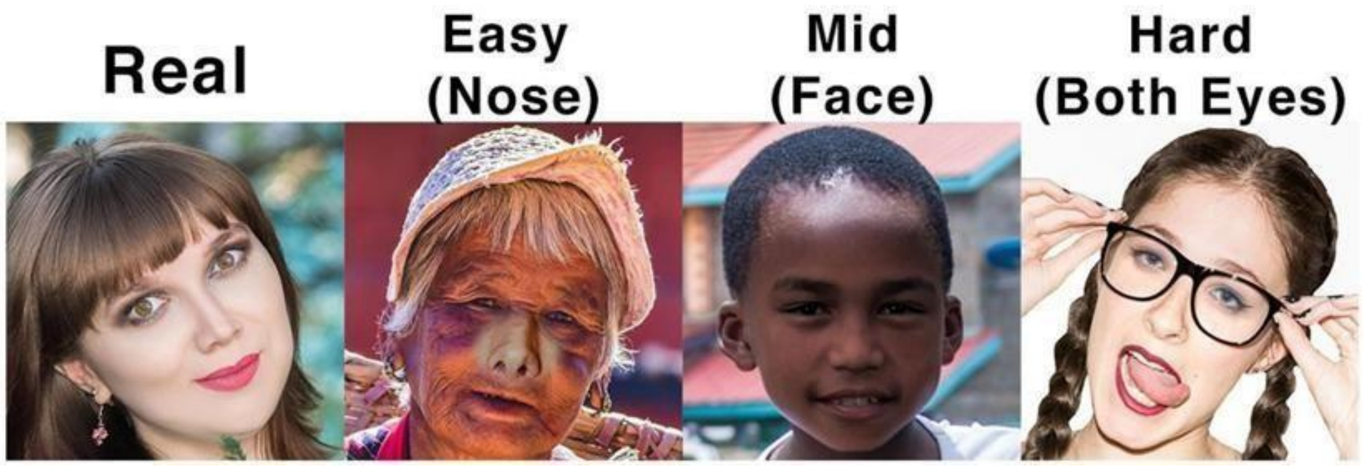


Fig 1: IMAGE SAMPLES SHOWING REAL AND FAKE IMAGES

CHAPTER 2
LITERATURE SURVEY

2.1 PRODUCT AVAILABILITIES

This highlights key studies and approaches that have leveraged TensorFlow for deepfake detection.

Convolutional Neural Networks (CNNs) with TensorFlow

Afchar et al. (2018) introduced MesoNet, a lightweight CNN architecture designed to detect deepfakes, specifically focusing on small, local visual artifacts left by deepfake generation processes. Implemented in TensorFlow, the model is effective in detecting manipulated faces by identifying texture inconsistencies

Transfer Learning using Pre-trained Models

TensorFlow's Keras API facilitated seamless integration of transfer learning by enabling the re-use of trained layers and modification of top layers to suit deepfake detection, highlighting TensorFlow's flexibility in handling complex models efficiently.

Generative Adversarial Networks (GANs) and TensorFlow

Wang et al. (2020) investigated using TensorFlow to train GANs not only to generate deepfakes but also to detect them. GANs have been widely adopted for generating realistic fake images and videos, and the study used TensorFlow’s high flexibility to build adversarial training loops. GAN-based detection models were trained to identify specific noise patterns and image artifacts that often remain in GAN-generated content

FaceForensics++ Dataset and TensorFlow

Rossler et al. (2019), in their work on the FaceForensics++ dataset, leveraged TensorFlow to build state-of-the-art models for detecting manipulated faces. The dataset includes a wide variety of manipulated videos, including deepfakes, and the models trained with TensorFlow demonstrated high detection accuracy. TensorFlow's efficient model deployment and optimization capabilities helped in scaling the training process over a large dataset, ensuring that the detection model could generalize well to unseen data.

Autoencoders for Deepfake Detection

Mirsky and Lee (2021) explored the use of autoencoders for detecting deepfakes in TensorFlow. Autoencoders, typically used for anomaly detection, were trained on real images and tasked with reconstructing input images. When presented with deepfakes, the autoencoders struggled to accurately reconstruct the fake images, revealing clear discrepancies between the input and output. TensorFlow's dynamic computational graph and efficient model training mechanisms enabled the autoencoders to process large-scale image datasets efficiently.

CHAPTER 3
ANALYSIS

3.1 OBJECTIVE OF THE PRODUCT

The Study field will be a distorted grouping. This area is the mechanism by which minor imperfections are found in doctored videos and exposes the false representation of originality. The creation of a doctored image/video needs information to be examined in order to reveal the dishonest, particularly facial expression variables. It is defined as an in-depth image/video study to find minor imperfections such as boundary points, background incoherence, double eyebrows or irregular twitch of the eye

The motive force behind this research is to recognize these distorted media, which is technically demanding and which is rapidly evolving. Many engineering companies have come together to unite a great deal of dataset. Competitions and actively include data sets to counter deepfakes

Deepfake videos are now so popular that multiple political parties utilise this tool to produce faked images of the leader of their opposing party to propagate hate against them. Fake political videos telling or doing things that have never happened is a threat to election campaigns. These images are the primary source of false media controversies and propagate misleading news.

In order to expose the forgery in extremely detailed facial expression, such details to be investigated frame by frame in the production of a deepfake picture. The goal of this research is to create a deep learning model that is capable of recognizing deepfake images. A thorough analysis of deepfake video frames to identify slight imperfections in face head and the model will learn what features differentiate a real image from a deepfake.

3.2 REQUIREMENTS

For a product focused on deepfake detection using machine learning, there are several technical, hardware, and software requirements to consider.

3.2.1 HARDWARE REQUIREMENTS

Processor: Intel Core i5 or higher

RAM: 8 GB or more

Storage*:* 256 GB SSD or more

GPU: NVIDIA GeForce GTX 1050 or higher (for machine learning model training)

3.2.2 SOFTWARE REQUIREMENTS

Operating System: Windows 10 or higher, macOS, or Linux

Development Environment: Anaconda for Python, Jupyter Notebook

Python:

Version: Python 3.8 or later.

Purpose: The main purpose of using Python in deepfake detection with machine learning is to build systems that can automatically recognize manipulated or fake videos/images

Integrated Development Environment (IDE):

Recommended: Visual Studio Code, PyCharm, or Jupyter Notebook.

Purpose: To facilitate code development, debugging, and version control

Frameworks:

PyTorch:

Version: PyTorch 1.10 or later.

Purpose: The main purpose of using PyTorch is to make it easier to build and train machine learning models, especially neural networks

TensorFlow :

Version: TensorFlow 2.x.

Purpose: The purpose of this is to provide a powerful and flexible platform for building, training, and deploying machine learning models

Data Processing and Management

NumPy:

Purpose: For numerical operations and handling multi-dimensional arrays and matrices, essential for data processing tasks.

Pandas:

Purpose: For data manipulation and analysis, particularly for managing data logs, detection results, and integrating various datasets.

OpenCV:

Purpose: For image processing tasks,and applying post-processing algorithms to refine detection outputs.

CHAPTER 4
METHOD AND IMPLEMENTION

4.1 ALGORITHMS USED

In our deepfake detection project using machine learning with TensorFlow, the following algorithms and techniques are commonly used:

Convolutional Neural Networks (CNN)

CNNs are a primary method for image classification tasks. In your case, the CNN architecture is used to classify images as real or fake based on features extracted from the image.

Components like convolutional layers, pooling layers, and fully connected layers help the model learn spatial hierarchies of features from the images.

Binary Classification (Logistic Regression)

Since the task is to classify an image as either real or fake (deepfake), a binary classification layer (sigmoid activation function) is used in the final layer of the CNN to output a probability score

Adam Optimizer

The Adam (Adaptive Moment Estimation) optimizer is used for efficient gradient-based optimization during model training. Adam is known for combining the advantages of both the RMSProp and Stochastic Gradient Descent (SGD) algorithms.

Binary Cross-Entropy Loss Function

Binary cross-entropy is the loss function employed in binary classification tasks. It measures the difference between the predicted output (real/fake) and the actual target values.

Data Augmentation

Techniques such as rotation, flipping, zooming, and shifting are applied to the training data to improve generalization and make the model more robust to variations in input images.

Transfer Learning (Extensions)

Pre-trained models like VGG16 or ResNet are used to leverage learned features from large image datasets (e.g., ImageNet) to improve detection accuracy. You fine-tune these models for your specific deepfake detection task.

OpenCV for Preprocessing

OpenCV is used for tasks like image resizing, normalization, and other preprocessing techniques to prepare the dataset for the model.

Train-Test Split & Performance Metrics (Scikit-learn)

The dataset is divided into training and testing sets using Scikit-learn's train_test_split. Performance metrics such as accuracy, precision, recall, and F1 score are used to evaluate the model.

These techniques and algorithms plays a major role in deepfake detection and helps to detect the manipulations in images.

4.2 SOURCE CODE

Import libraries

```
import os

import cv2

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import tensorflow as tf

fromsklearn.model_selectionimporttrain_test_split
```

```
from tensorflow.keras.utils import to_categorical

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.losses import BinaryCrossentropy

from sklearn.metrics import accuracy_score
```

DIRECTORY PATH TO LOAD AND PREPROCESS THE IMAGES

```
data_dir = r'C:\Users\heman\OneDrive\Desktop\Deep Fake
Detection\data\real_and_fake_face'
```

LOAD AND PREPROCESS IMAGES

```
def load_dataset(data_dir):
```

LOADING THE DATASET

```
X, y = load_dataset(data_dir)
```

```
Accessing folder: C:\Users\heman\OneDrive\Desktop\Deep Fake
Detection\data\real_and_fake_face\training_real

Accessing folder: C:\Users\heman\OneDrive\Desktop\Deep Fake
Detection\data\real_and_fake_face\training_fake
```

PREPARE THE DATASET FOR TRAINING, VALIDATION, AND TESTING

```
X = X.astype('float32') / 255.0

y = to_categorical(y, 2)

X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3,
random_state=42)

X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5,
random_state=42)
```

CNN MODEL

```
model = Sequential([

    Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)),

    MaxPooling2D(pool_size=(2, 2)),

    Flatten(),

    Dense(64, activation='relu'),

    Dense(2, activation='softmax')  # Two outputs for one-hot encoding

])



# Compile the model with categorical_crossentropy for one-hot encoded labels

model.compile(optimizer=Adam(), loss='categorical_crossentropy',
metrics=['accuracy'])

# Train the model with one-hot encoded labels
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 242ms/step - accuracy:
0.7249 - loss: 0.5637 - val_accuracy: 0.5556 - val_loss: 0.9190

Epoch 9/10

3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 205ms/step - accuracy:
0.9684 - loss: 0.1292 - val_accuracy: 0.6111 - val_loss: 0.9337

Epoch 10/10

3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 206ms/step - accuracy:
0.9528 - loss: 0.1540 - val_accuracy: 0.5556 - val_loss: 0.9996
```

ACCURACY

```
def plot_accuracy(history):

    """Plot training and validation accuracy."""

    plt.figure(figsize=(8, 6))

    plt.plot(history.history['accuracy'], label='Training Accuracy')

    plt.plot(history.history['val_accuracy'], label='Validation Accuracy')

    plt.title('Model Accuracy')

    plt.xlabel('Epoch')

    plt.ylabel('Accuracy')

    plt.legend(loc='lower right')

    plt.grid(True)

    plt.show()

# Assuming you have already trained your model and have a history object

plot_accuracy(history)
```

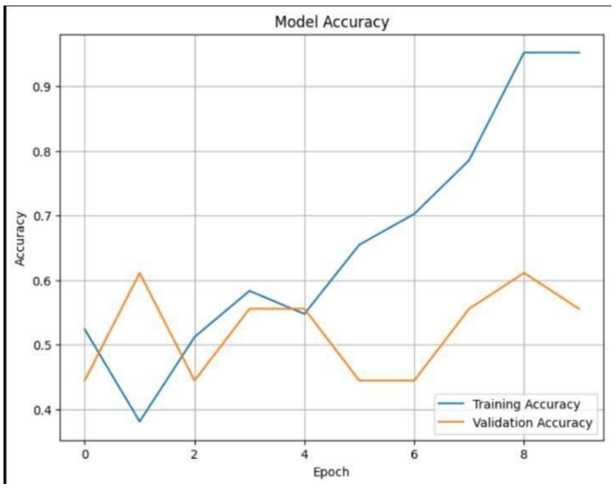


Fig 2: ACCURACY GRAPH

LOSS

```
def plot_loss(history):

"""Plot training and validation loss."""

plt.figure(figsize=(8, 6)) plt.plot(history.history['loss'],

label='Training Loss')

plt.plot(history.history['val_loss'], label='Validation Loss')

plt.title('Model Loss')

plt.xlabel('Epoch')

plt.ylabel('Loss')

plt.legend(loc='upper right')

plt.grid(True)

plt.show()

# Assuming you have already trained your model and have a history object

plot_loss(history)
```

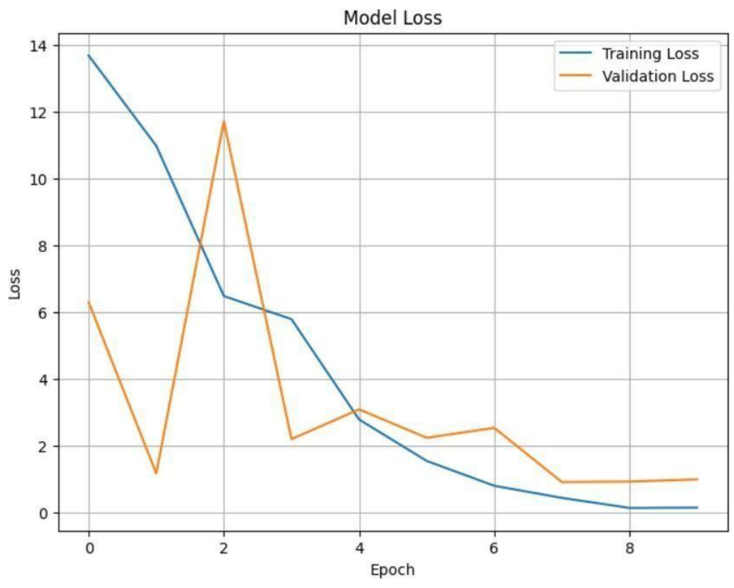


Fig 3: LOSS GRAPH

PRINT TRAINING, VALIDATION, TEST SAMPLES

```
print(f'Training samples: {X_train.shape[0]}')

print(f'Validation samples: {X_val.shape[0]}')

print(f'Test samples: {X_test.shape[0]}')
```

```
Training samples: 84

Validation samples: 18

Test samples: 19
```

DISPLAY TRAINING, VALIDATION, TEST SAMPLES

```
def plot_sample_images(images, labels, title, n=5):

plt.figure(figsize=(15, 3))

plt.suptitle(title)for i

in range(n):

ax = plt.subplot(1, n, i + 1)

# Convert BGR to RGB for display
```

```
plt.imshow(cv2.cvtColor(images[i], cv2.COLOR_BGR2RGB))
# If labels are one-hot encoded, get the index of the max value
if isinstance(labels[i], np.ndarray):
    label = np.argmax(labels[i])
else:
label = labels[i]


plt.title('Real' if label == 0 else 'Fake')
plt.axis("off")
plt.show()
# Display a few training images
print("Training Samples:")
plot_sample_images(X_train, y_train, "Training
Samples")# Display a few validation images
print("Validation Samples:")
plot_sample_images(X_val, y_val, "Validation
Samples")# Display a few test images
print("Test Samples:")
plot_sample_images(X_test, y_test, "Test
Samples")TRAINING SAMPLES
```

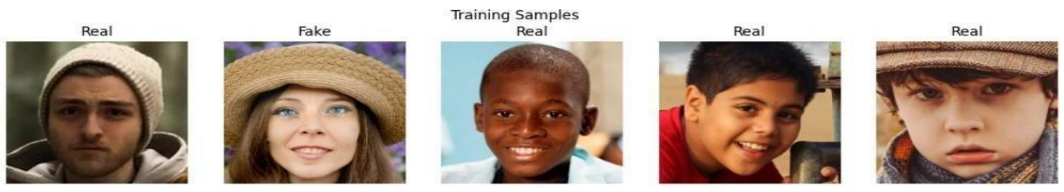


Fig 4: TRAINING SAMPLE IMAGES

VALIDATION SAMPLES

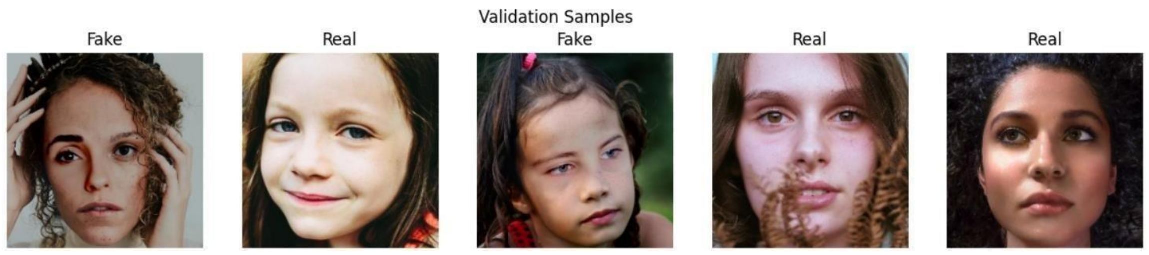
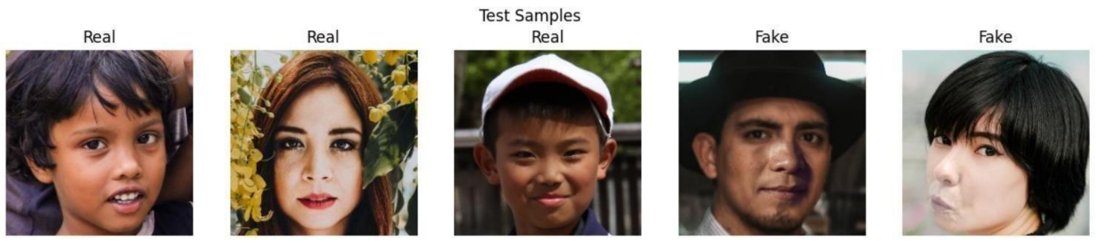


Fig 5: VALIDATION SAMPLE IMAGES



TEST SAMPLES

Fig 6: TEST SAMPLE IMAGES

COUNT NO OF REAL AND FAKE IMAGES

```
 def count_images(real_dir, fake_dir):

  """Count the number of images in real and fake directories."""

  counts = {'real': 0, 'fake': 0}

  if os.path.exists(real_dir):

      counts['real'] = len(os.listdir(real_dir))

  else:

      print(f"Real directory not found:

  {real_dir}") if os.path.exists(fake_dir):

      counts['fake'] = len(os.listdir(fake_dir))

  else:
```

```
print(f"Fake directory not found: {fake_dir}"
return counts #

Update paths

real_dir = r'C:\Users\heman\OneDrive\Desktop\Deep Fake
Detection\data\real_and_fake_face\training_real'

fake_dir = r'C:\Users\heman\OneDrive\Desktop\Deep Fake
Detection\data\real_and_fake_face\training_fake'

class_counts = count_images(real_dir, fake_dir)

print(f"Number of Real Images: {class_counts['real']}")

print(f"Number of Fake Images: {class_counts['fake']}")
```

```
Number of Real Images: 60

Number of Fake Images: 61
```

PRINT RESULTS

```
# Define loss function

loss_function = BinaryCrossentropy()

# Predict on test data

predictions = model.predict(X_test)

true_labels = np.argmax(y_test, axis=1) if y_test.ndim > 1 else y_test

# Convert predictions to binary labels

predicted_labels = np.argmax(predictions, axis=1) if predictions.shape[1] > 1 else
(predictions > 0.5).astype(int)

# Calculate loss for each image

losses = [loss_function(tf.convert_to_tensor(y_true, dtype=tf.float32),

tf.convert_to_tensor(y_pred, dtype=tf.float32)).numpy()

for y_true, y_pred in zip(y_test, predictions)]#

Calculate overall accuracy

accuracy = accuracy_score(true_labels, predicted_labels)
```

```
# Calculate accuracy percentage for each image

accuracy_percentages = [100 if true == pred else 0 for true, pred in
zip(true_labels, predicted_labels)]

print(f"Overall Accuracy: {accuracy * 100:.2f}%")

print(f"Average Loss: {np.mean(losses):.4f}")

# Detailed results

 results = pd.DataFrame({

'Image': range(len(losses)),

'True Label': true_labels,

'Predicted Label': predicted_labels,'Loss':

losses,

'Accuracy (%)': accuracy_percentages

})

# Display the first few results

print(results.head())
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step

Overall Accuracy: 52.63%

Average Loss: 1.1142

Image  True Label  Predicted Label      Loss  Accuracy (%)

0      0           0                0  0.099059        100

1      1           0                1  0.747106          0

2      2           0                0  0.141974        100

3      3           1                0  1.830892          0

4      4           1                0  3.641300          0
```

CHAPTER 5
RESULTS & DISCUSSION

5.1 PROPOSED SYSTEM ARCHITECTURE

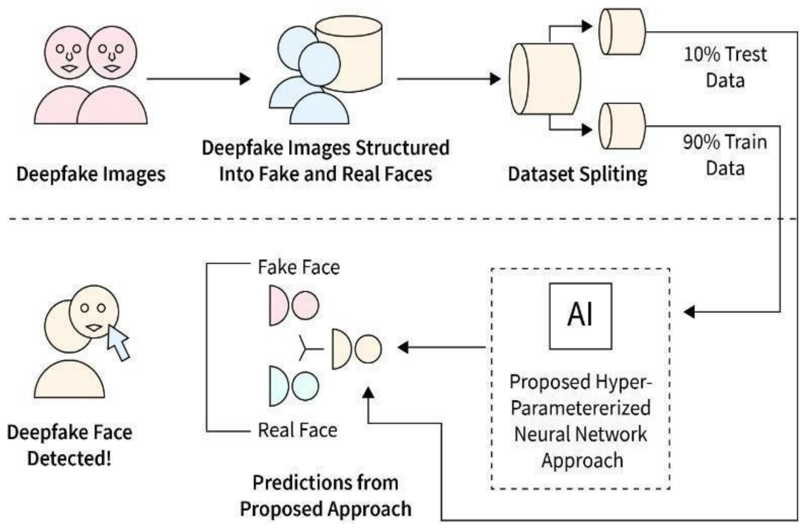

Deepfake Images
Deepfake Images Structured Into Fake and Real Faces
Dataset Spliting
10% Trest Data
90% Train Data
Fake Face
Real Face
AI
Proposed Hyper-Parametererized Neural Network Approach
Deepfake Face Detected!
Predictions from Proposed Approach

Fig 7: SYSTEM ARCHITECTURE

5.2 SNAPSHOTS

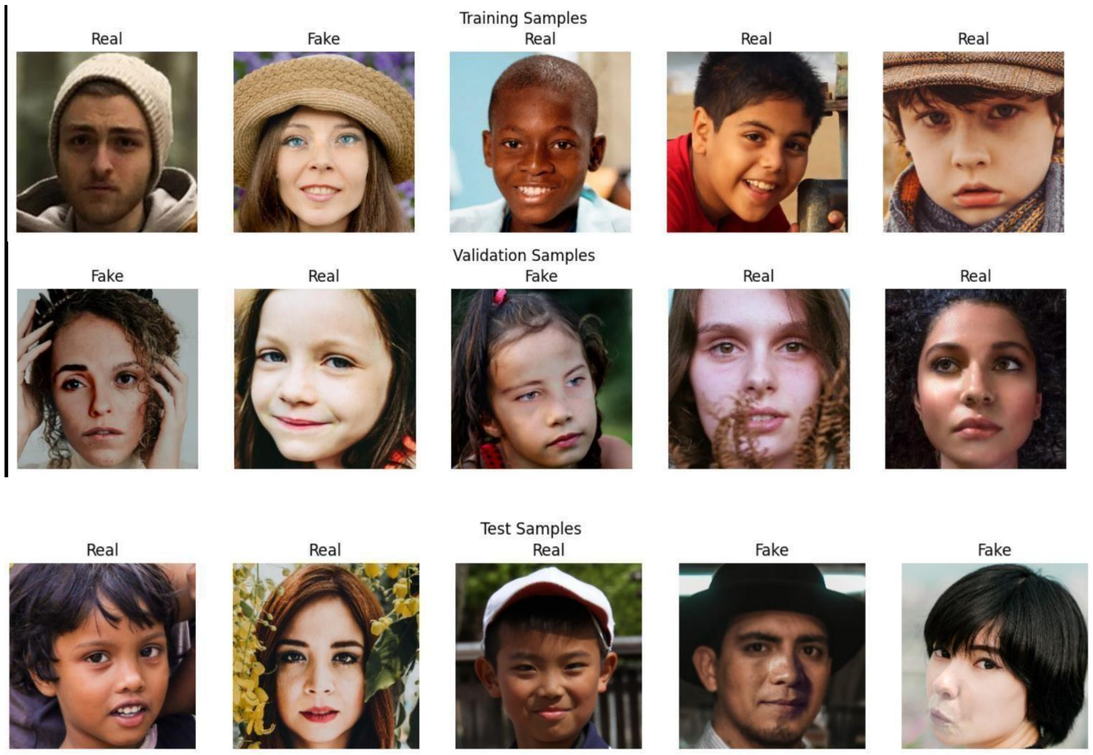


Fig 8: VARIOUS SAMPLE IMAGES

CHAPTER 6

CONCLUSION

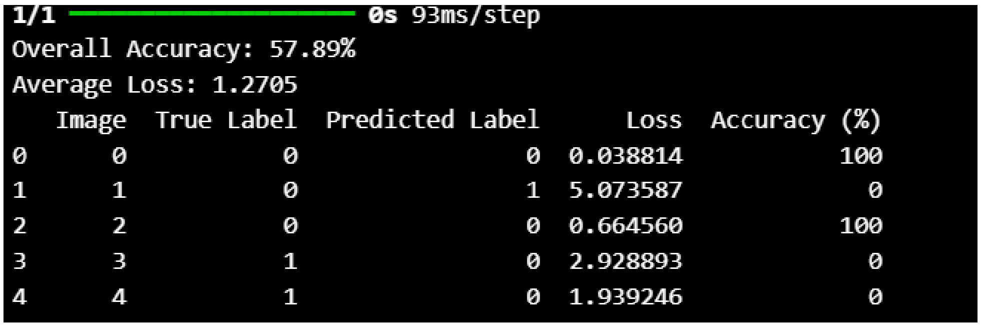
6.1 CONCLUSION

This Self-Supervised Learning Representation presents various state-of-the-art methods for detecting Deepfake published in 112 studies from the beginning of 2018 to the end of 2020. We present basic techniques and discuss different detection models' efficacy in this work. We summarize the overall study as follows:
The deep learning-based methods are widely used in detectingDeepfake.

In the experiments, the FF++ dataset occupies the largest proportion.

The deep learning (mainly CNN) models hold a significant percentage of all the model

The most widely used performance metric is detection accuracy.



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step
Overall Accuracy: 57.89%
Average Loss: 1.2705
   Image  True Label  Predicted Label      Loss  Accuracy (%)
0      0           0                0  0.038814           100
1      1           0                1  5.073587             0
2      2           0                0  0.664560           100
3      3           1                0  2.928893             0
4      4           1                0  1.939246             0
```

Fig 9: RESULTS FROM PROPOSED MODEL

The overall accuracy is **52.63%** and average loss is 1.1142%

Deepfake detection for images using machine learning with TensorFlow leverages powerful models like CNNs to detect subtle inconsistencies typical of deepfakes. TensorFlow's scalability, combined with techniques like transfer learning and data augmentation, enhances model performance. Overall, TensorFlow is a robust platform for developing accurate and scalable deepfake detection systems

REFERENCES

Andreas Rossler, Davide Cozzolino, LuisaVerdoliva, ChristianRiess, Justus Thies, Matthias Nießner, "FaceForensics++:Learning to Detect Manipulated Facial Images"2019 IEEE/CVF International Conference on Computer Vision (ICCV).

Md Shohel Rana, Mohammad Nur Nobi,Beddhu Murali,Andrew H. Sung ,Deepfake Detection: A Systematic Literature Review IEEE Access 10:1-1.
TensorFlow, December 2020, [online] Available: https://www.tensorflow.org/.

Brian Dolhansky, Russ Howes, Ben Pflaum, Nicole Baram and Cristian. Ferrer,The Deepfake Detection Challenge(DFDC)Preview Dataset, 2019.

Faceswap: Deepfakes software for all, February 2021, [online] Available: https://github.com/deepfakes/faceswap.

Gaojian Wang, Qian Jiang, Xin Jin, Xiaohui Cui, ―FFR FD: Effective and FastDetection of DeepFakes Based on Feature Point Defects‖, 2020.

Siwei Lyu, ―DEEPFAKE DETECTION: CURRENT CHALLENGES AND NEXT STEPS‖, IEEE International Conference on Multimedia & Expo Workshops (ICMEW) (2020).

D. Guera and E. J. Delp, ―Deepfake video detection using recurrent ¨ neural networks,‖ in 2018 15th IEEE international conference on advanced video and signal based surveillance (AVSS). IEEE, 2018, pp. 1– 6.

Maksutov, A. A., Morozov, V. O., Lavrenov, A. A. and Smirnov, A. S. (2020). Methods of deepfake detection based on machine learning, pp. 408-411.

Nguyen, T. T., Nguyen, C. M., Nguyen, D. T., Nguyen, D. T. and Nahavandi, S. (2019). Deep learning for deepfakes creation and detection: A survey