Fake Currency Detection

PROJECT EXHIBITION – 01

SUPERVISED BY: DR SUBASH CHANDRA BOSE I REVIEWED BY: DR NILAMADHAB MISHRA , DR SHIV SHANKAR PRASAD SHUKLA

TEAM MEMBERS

Shubhankar Pandey (23BAI11053)
Meera Singh (23BAI11317)
Tithi Semwal (23BAI10920)
Jaison Jai John (23BAI11378)
Anshuman Pandey(23BAI10208)

PROBLEM STATEMENT

Money drives economic activities like manufacturing, circulation and consumption, making it essential in today's society.

However, counterfeit currency is a growing threat, harming everyday people and shrinking the economy.

Counterfeit notes, especially in Rs. 500 and Rs. 1,000 denominations, are hard to detect, affecting transactions at every level. Globally, the issue is a concern.

In India, fraud cost banks Rs. 16,789 crores in one year, and counterfeit notes have surged, with a significant rise reported in Rs. 500 (101.9%) and Rs. 2,000 (54.6%) notes, according to the RBI's 2021–22 report.

OUR AGENDA

- **Improve Detection Efficiency:** By leveraging advanced image processing techniques to quickly identify counterfeit currency with high accuracy.
- **Support Real-Time Verification:** By analyzing key features such as security threads, watermarks, and note patterns for instant detection of fake notes.
- **Enhance User Accessibility:** By providing a user-friendly interface for seamless implementation in banks, businesses, and daily transactions.

NOVELTY

The **Fake Indian Currency Detection** system introduces a novel approach by leveraging advanced deep learning and computer vision techniques to analyze intricate features of currency notes, such as watermarks, security threads, and micro-lettering, beyond traditional methods. The system offers **real-time detection capabilities**, ensuring immediate identification of counterfeit notes with high accuracy. This innovative solution not only improves operational efficiency but also strengthens the fight against counterfeit currency circulation.

LITERATURE REVIEW

Author	Title	Description
Devid Kumar, Surendra Singh Chauhan	A STUDY ON INDIAN FAKE CURRENCY DETECTION	The detection of **fake Indian currency** has progressed significantly with the integration of **image processing** and **machine learning** techniques. Traditional manual methods, though widely used, are time-consuming and error-prone. Advanced approaches such as **feature extraction**, edge detection, and classification algorithms like **Support Vector Machines (SVM)** and **Convolutional Neural Networks (CNN)** analyze security features like watermarks, security threads, and micro-lettering. Real-time mobile applications using computer vision provide accessibility and accuracy, helping individuals and businesses detect counterfeit notes efficiently.
Aneena Babu, Vineetha Shankar P	Fake Indian Currency Detection	This paper enhances Counterfeit currency detection employs image processing techniques, machine learning, and hybrid models to identify fake notes effectively. Approaches like ORB, SSIM, and deep learning enhance accuracy. Challenges include dataset limitations, real-time processing, and evolving counterfeiting methods. Future research emphasizes advanced algorithms, blockchain integration, and global collaboration to combat counterfeiting effectively.

PROJECT TIMELINE

WEEK 01- 04

Project Planning
And literature Review
And
Dataset Collection

WEEK 05-08

Dataset
Preprocessing
And model
development
(Software Level)

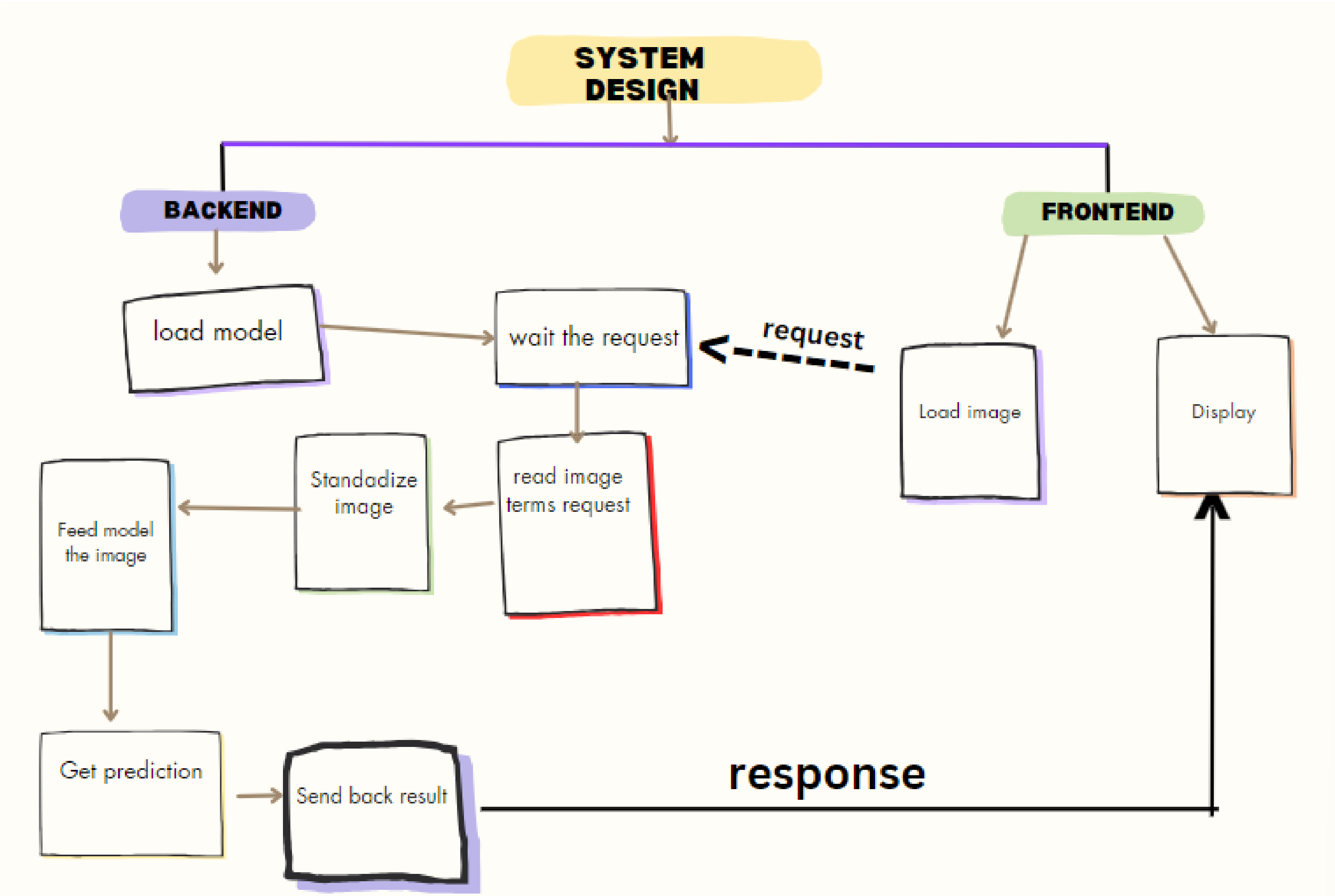
WEEK 09-12

Model Evaluation and
improvement and
Implementing ESP 32
CAMERA

WHAT IS COUNTERFEITING?

Counterfeiting is the act of creating fake or imitation items with the intent to deceive others into believing they are genuine. Common examples include counterfeit currency, goods, documents, and trademarks. Counterfeiting undermines economic stability, intellectual property rights, and public trust, often causing financial and reputational harm.

ARCHITECTURE DESIGN



ABOUT OUR DATASET

- This Dataset consists of Indian Currency image which is having two category i.e. real and fake. The image have been augmented by rotation and sharpening to produce significant amounts of images.
- The dataset contains total 5426 images, with each image having an average resolution 500 X 500 pixels. These images are in JPEG And PNG format. The dataset divided into three subset:
- Training set – 3797 images (Real and Fake)
- Testing set – 1085 images (Real and Fake)
- Validation Set – 544 images (Real and fake)
- Dataset is categorized into 2 Classes- Real and Fake

SOME PICTURES FROM DATASET (REAL)

SOME PICTURES FROM DATASET (FAKE)

IMAGE PREPROCESSING

rescale=1.0 / 255,
rotation_range=20,
width_shift_range=0.15,
height_shift_range=0.15,
shear_range=0.2,
zoom_range=0.2,
horizontal_flip=True,
brightness_range=[0.9, 1.1]

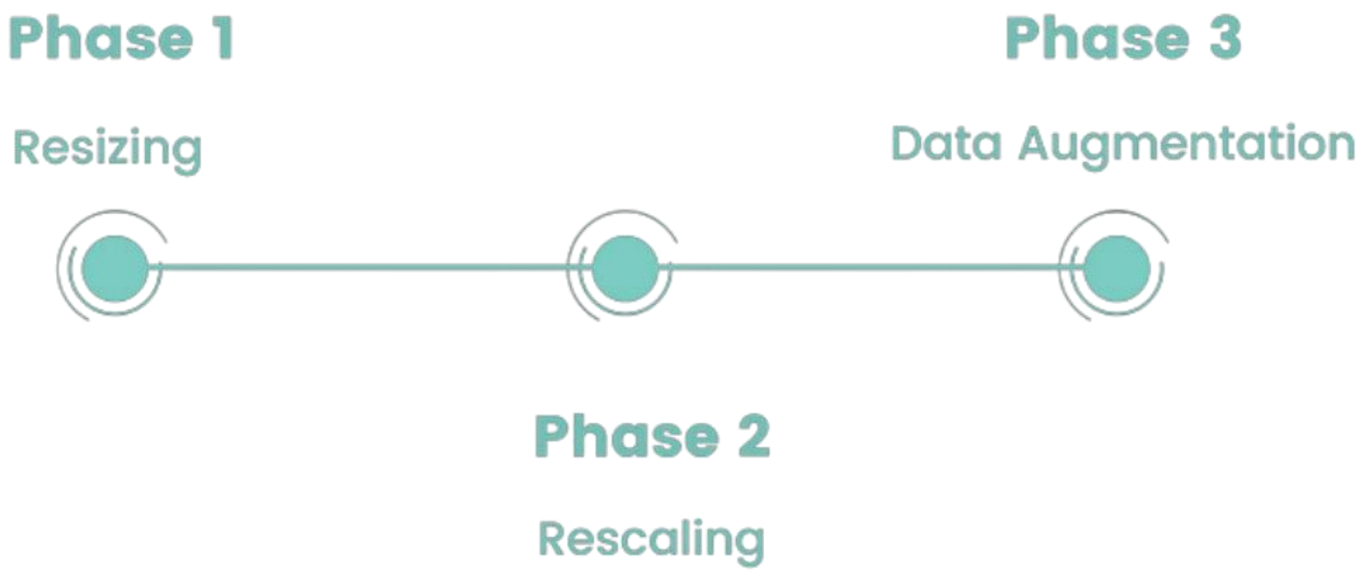
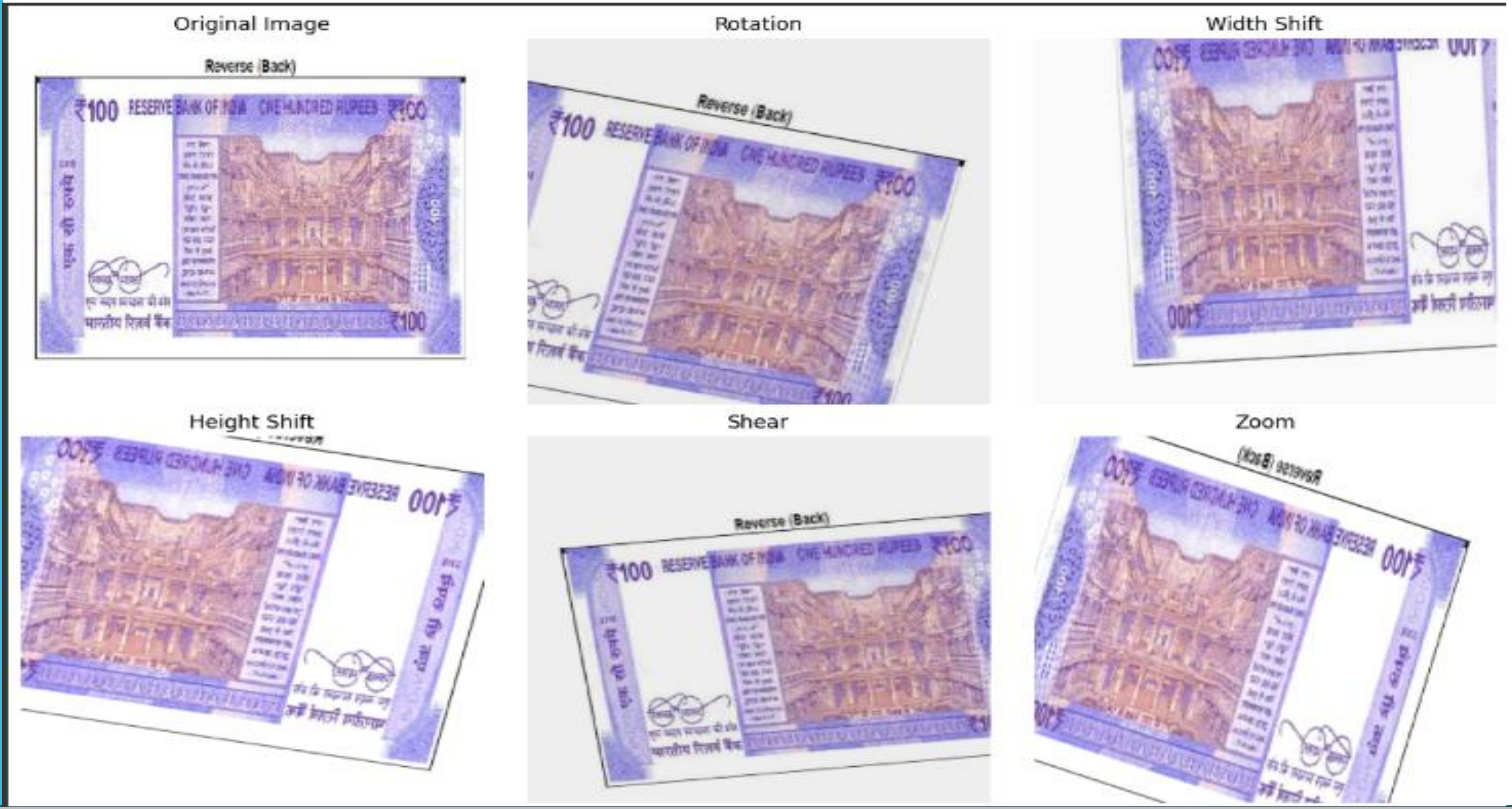
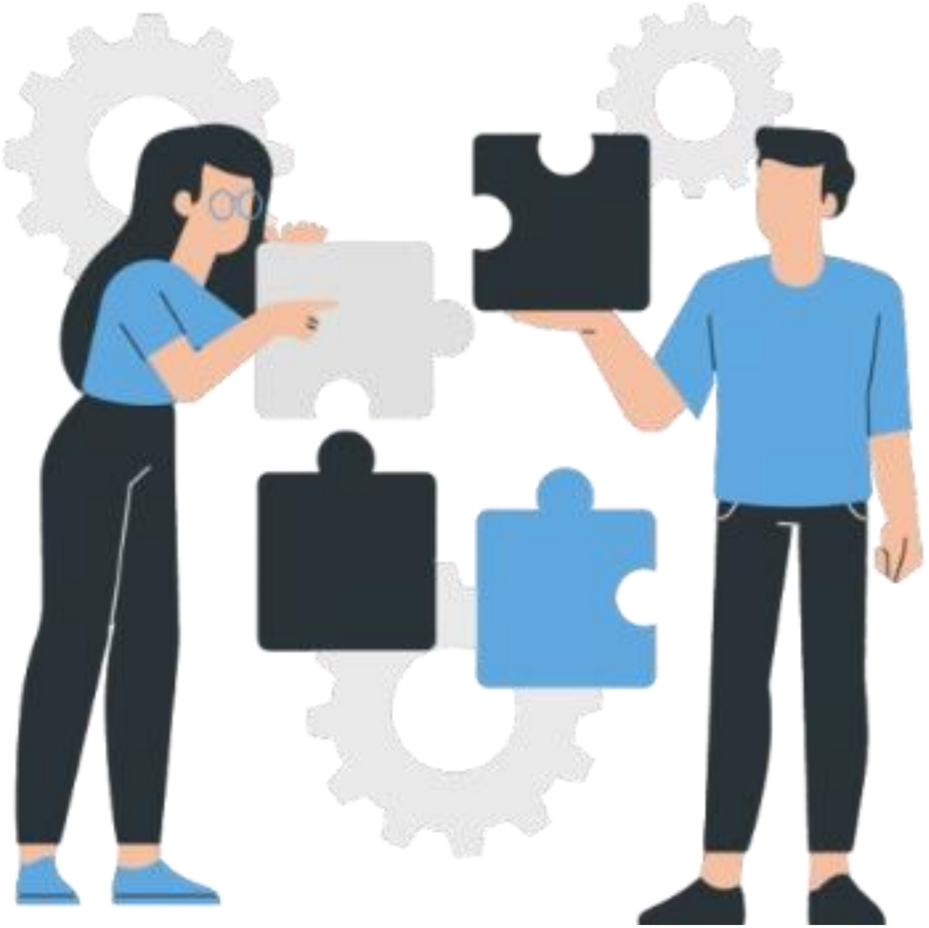


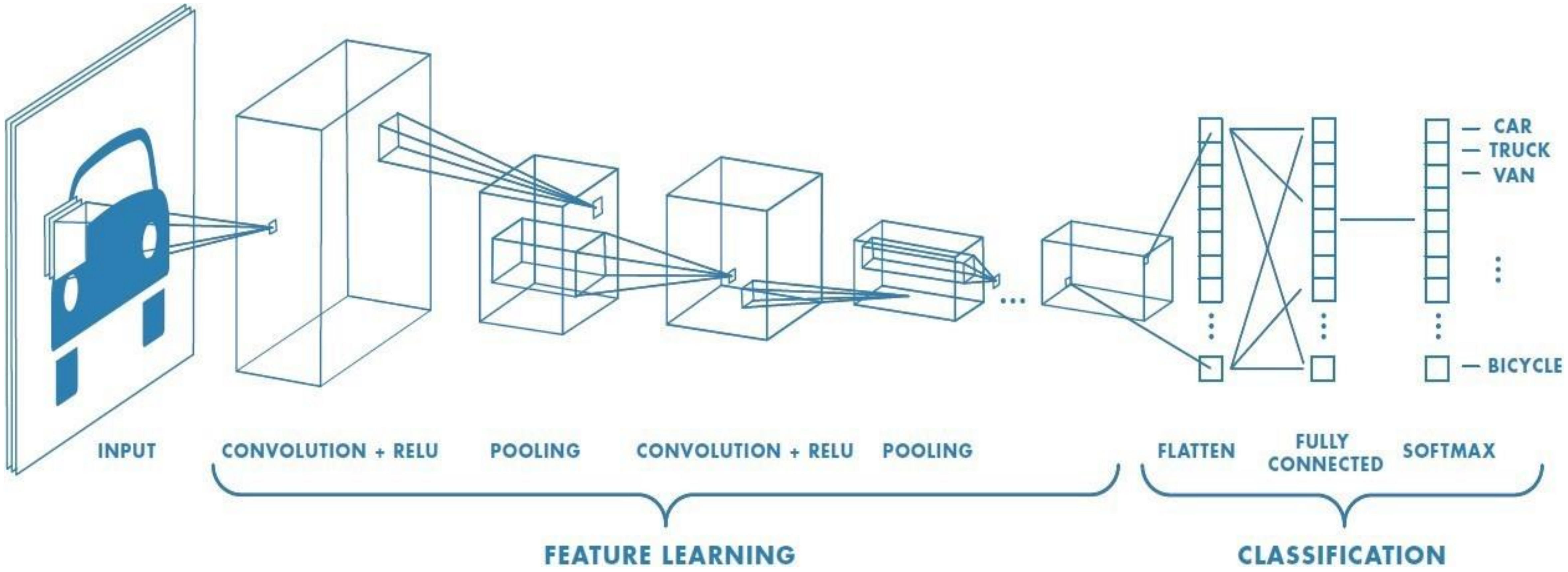
IMAGE PREPROCESSING





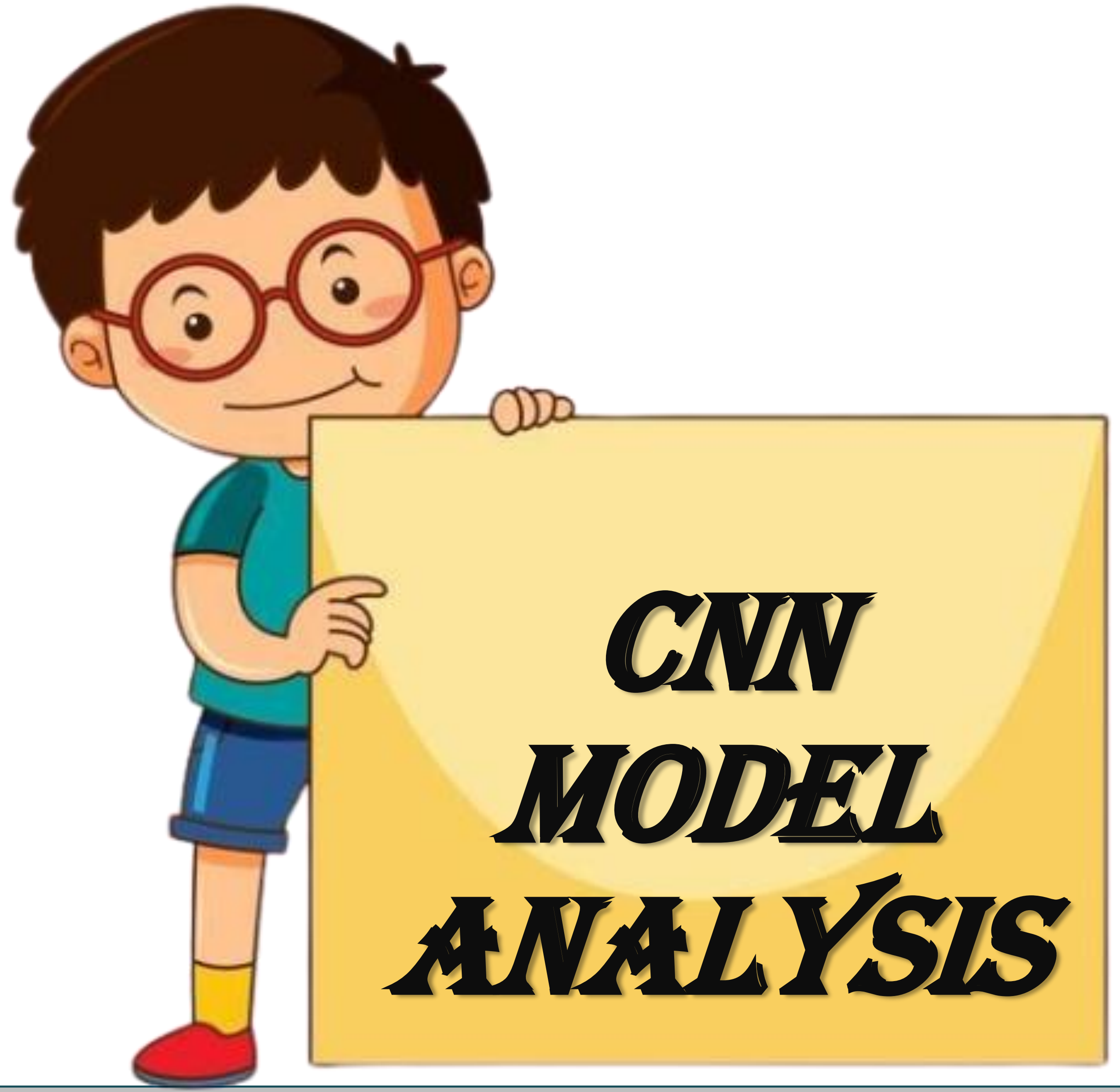
MODEL BUILDING

OUR MODEL – CNN MODEL(XCEPTION MODEL)



```python
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.applications import Xception

# Load the pre-trained Xception model
base_model = Xception(weights='imagenet', include_top=False, input_shape=(299, 299, 3))

# Freeze the layers of the base model
base_model.trainable = False

# Build a custom model with Xception as the base
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(512, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(256, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(1, activation='softmax')  # Sigmoid for binary classification (real/fake)
])

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

# Print the model summary
model.summary()
```

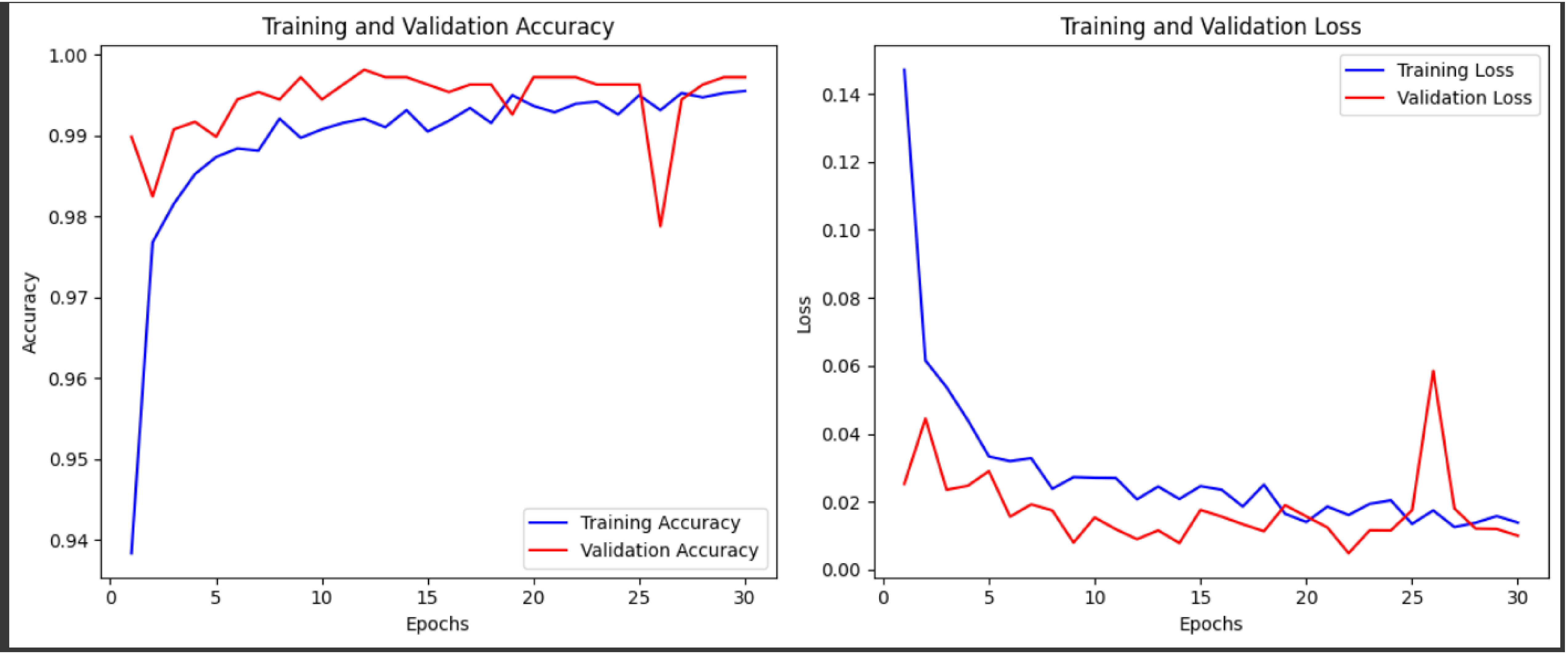
- ✓GlobalAveragePooling2D Layer
- ✓Dense Layer
- ✓Block_sepconv layer
- ✓Batch_normalization



CNN MODEL ANALYSIS

PROJECT EXHIBITION – 01

SUPERVISED BY: DR SUBASH CHANDRA BOSE I REVIEWED BY: DR NILAMADHAB MISHRA , DR SHIV SHANKAR PRASAD SHUKLA


Training and Validation Accuracy
1.00
0.99
0.98
0.97
0.96
0.95
0.94
Accuracy
0
5
10
15
20
25
30
Epochs
Training Accuracy
Validation Accuracy
Training and Validation Loss
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
Loss
Training Loss
Validation Loss

1

CONVERGENCE ACHIEVED

Both accuracy and loss stabilize by the end of the training, indicating that the model has reached an optimal state and further training may not improve performance significantly.

2

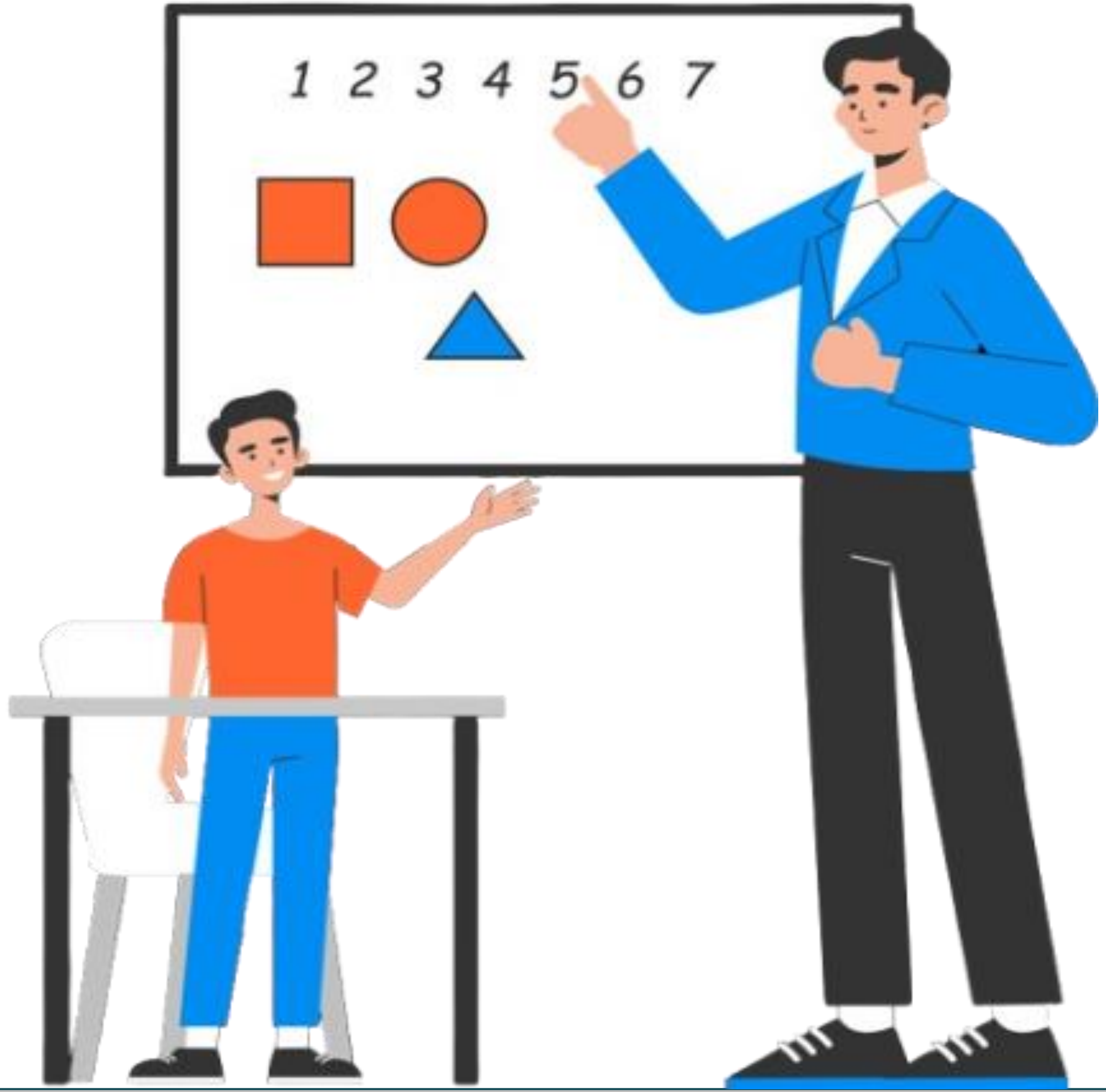
HIGH MODEL PERFORMANCE

The training and validation accuracy curves converge and reach close to 99%, indicating that the model performs very well on both training and validation data, with minimal performance gap

3

NO SIGNIFICANT OVERFITTING

The training and validation loss curves decrease steadily and remain close to each other throughout the training process, suggesting that the model generalizes well without significant overfitting.



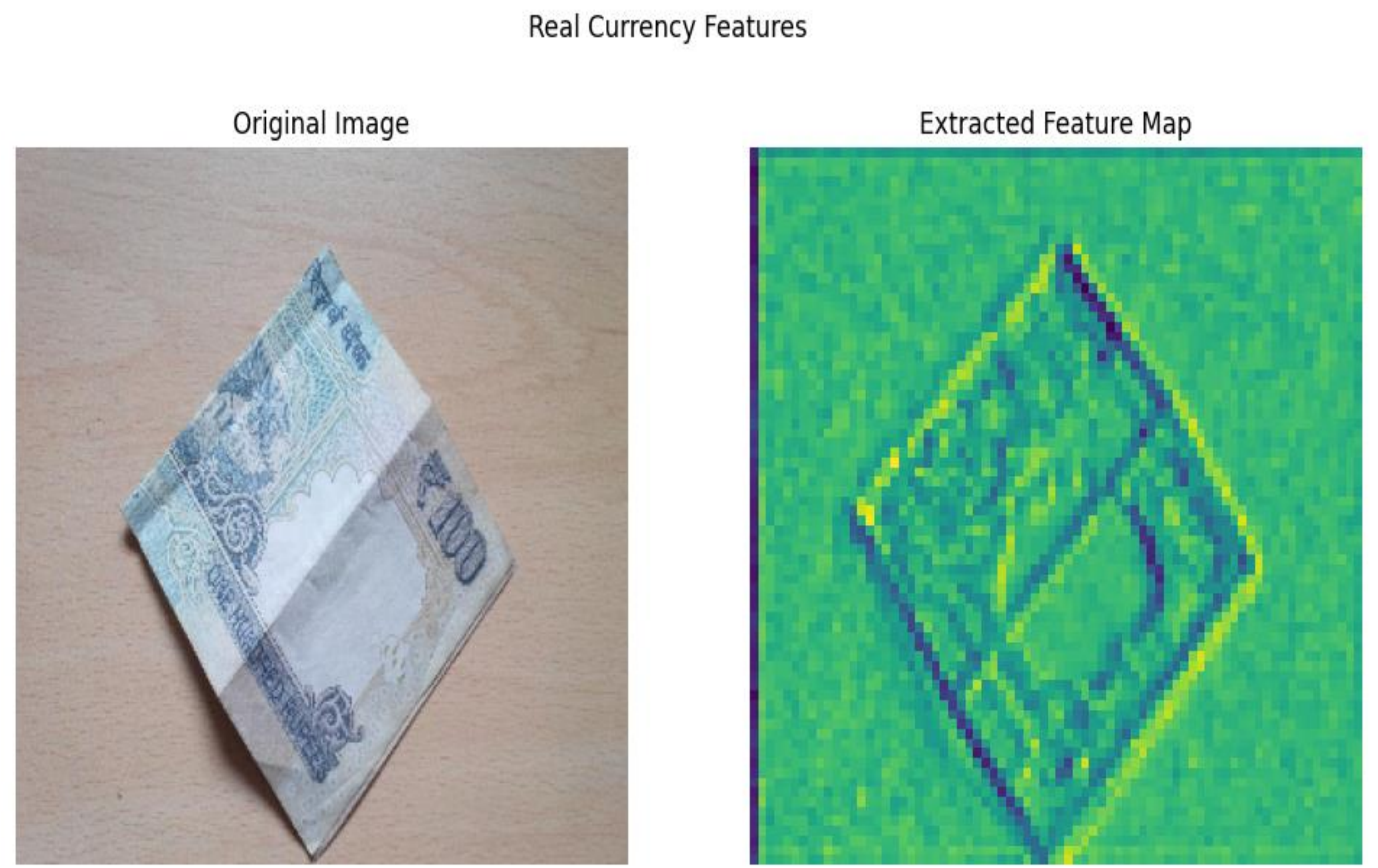
XCEPTION MODEL ANALYSIS

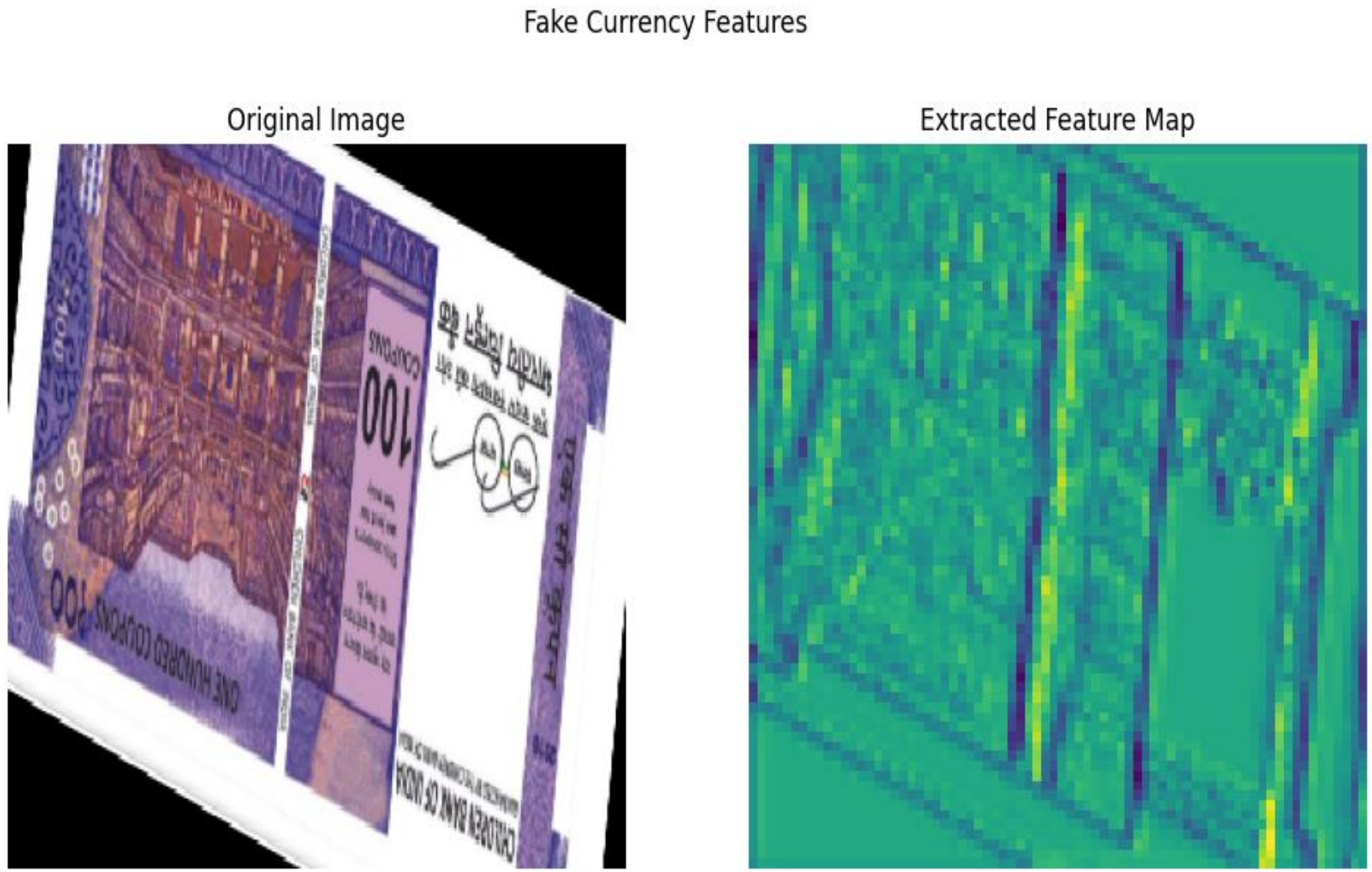
PROJECT EXHIBITION – 01

SUPERVISED BY: DR SUBASH CHANDRA BOSE | REVIEWED BY: DR NILAMADHAB MISHRA , DR SHIV SHANKAR PRASAD SHUKLA

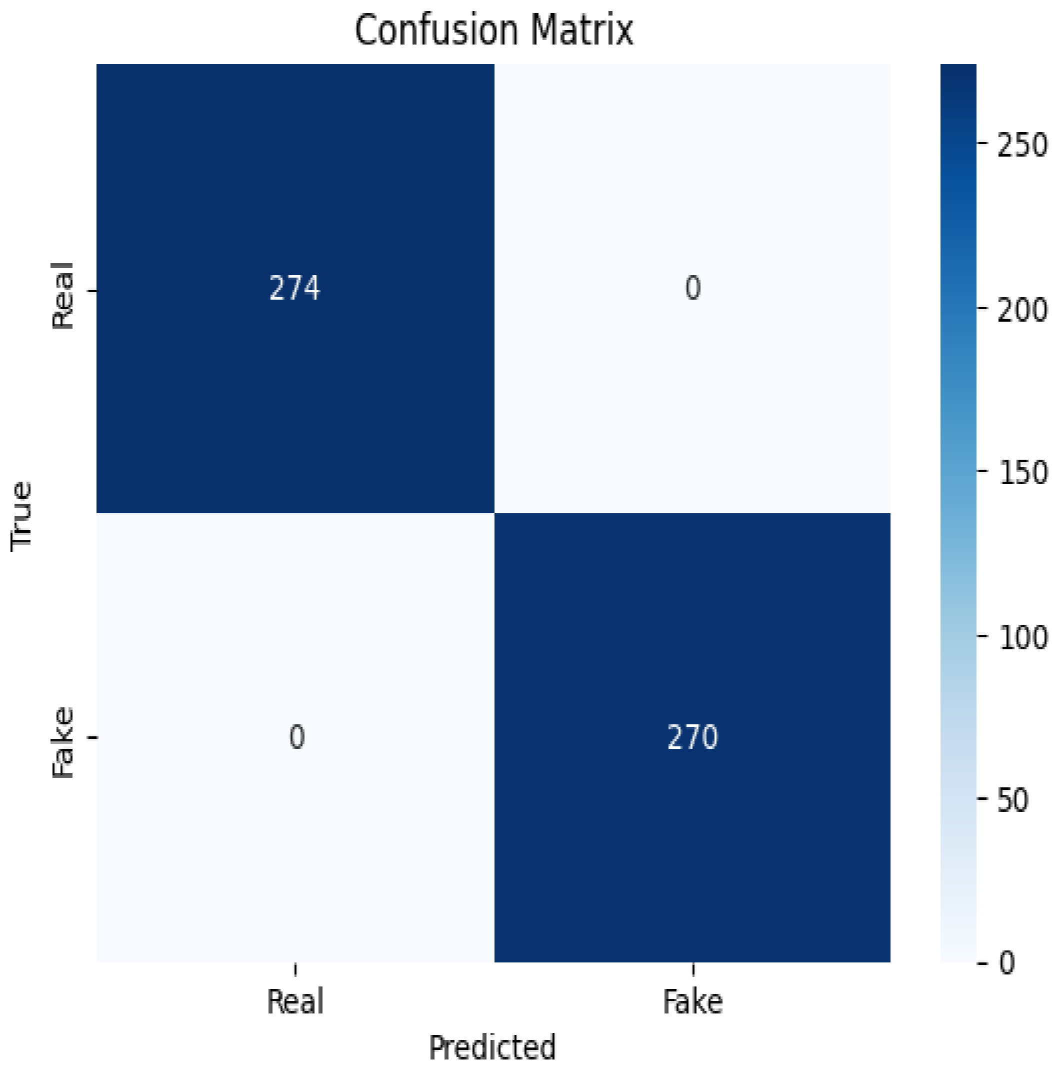
```
[ ] import numpy as np
    import matplotlib.pyplot as plt
    from tensorflow.keras.applications import Xception
    from tensorflow.keras.preprocessing import image
    from tensorflow.keras.models import Model
    import tensorflow as tf

    # Load the pre-trained Xception model
    base_model = Xception(weights='imagenet')

    # Select a valid layer you want to visualize (use 'block3_sepconv1' instead of 'block3_conv1')
    layer_name = 'block3_sepconv1'  # Updated to a valid layer name
    layer_output = base_model.get_layer(layer_name).output

    # Create a model that outputs the features from the selected layer
    feature_model = Model(inputs=base_model.input, outputs=layer_output)

    # Function to preprocess image and get features
    def preprocess_and_extract_features(image_path):
        # Load and resize the image to the size expected by Xception (299x299)
        img = image.load_img(image_path, target_size=(299, 299))
        img_array = image.img_to_array(img)  # Convert image to numpy array
        img_array = np.expand_dims(img_array, axis=0)  # Add batch dimension
        img_array = tf.keras.applications.xception.preprocess_input(img_array)  # Preprocess for Xception

        # Extract features using the feature model
        features = feature_model.predict(img_array)
        return features, img

    # Load and process the real and fake currency images
    real_image_path = r"C:\Users\Shubhankar\OneDrive\Desktop\organized_data\testing\Real\100__60.jpg"
    fake_image_path = r"C:\Users\Shubhankar\OneDrive\Desktop\organized_data\testing\fake\lp (34).png"

    real_features, real_img = preprocess_and_extract_features(real_image_path)
    fake_features, fake_img = preprocess_and_extract_features(fake_image_path)
```

```python
# Function to display the original image and extracted features
def display_features(original_img, features, title=""):
    plt.figure(figsize=(12, 6))

    # Display original image
    plt.subplot(1, 2, 1)
    plt.imshow(original_img)
    plt.title("Original Image")
    plt.axis('off')

    # Display one of the feature maps from the extracted features
    # We choose the first feature map in the output for visualization
    feature_map = features[0, :, :, 0]  # Select the first feature map from the layer output
    plt.subplot(1, 2, 2)
    plt.imshow(feature_map, cmap='viridis')
    plt.title("Extracted Feature Map")
    plt.axis('off')

    plt.suptitle(title)
    plt.show()

# Display features for both real and fake images
display_features(real_img, real_features, title="Real Currency Features")
display_features(fake_img, fake_features, title="Fake Currency Features")
```


Real Currency Features
Original Image
Extracted Feature Map


Fake Currency Features
Original Image
Extracted Feature Map

CONFUSION MATRIX ANALYSIS



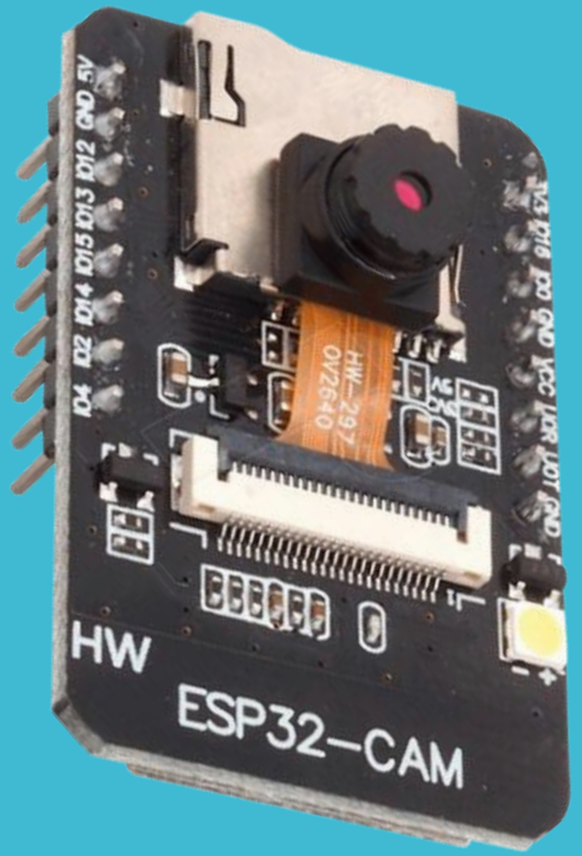
- The confusion matrix presented indicates perfect model performance for binary classification (Real vs. Fake). Here's the detailed analysis:

1. **True Positives (Real correctly classified): 274**
    1. The model successfully identified all 274 "Real" samples.
2. **True Negatives (Fake correctly classified): 270**
    1. All 270 "Fake" samples were correctly predicted.
3. **False Positives (Incorrectly classified as Fake): 0**
    1. No Real samples were misclassified as Fake.
4. **False Negatives (Incorrectly classified as Real): 0**
    1. No Fake samples were misclassified as Real.

- **Key Observations:**
- The confusion matrix demonstrates **100% accuracy** with no misclassifications.
- Both classes, Real and Fake, were perfectly identified by the model.
- This exceptional performance suggests the model has learned the distinguishing features between Real and Fake data very effectively.

PROJECT EXHIBITION – 01

SUPERVISED BY: DR SUBASH CHANDRA BOSE | REVIEWED BY: DR NILAMADHAB MISHRA , DR SHIV SHANKAR PRASAD SHUKLA

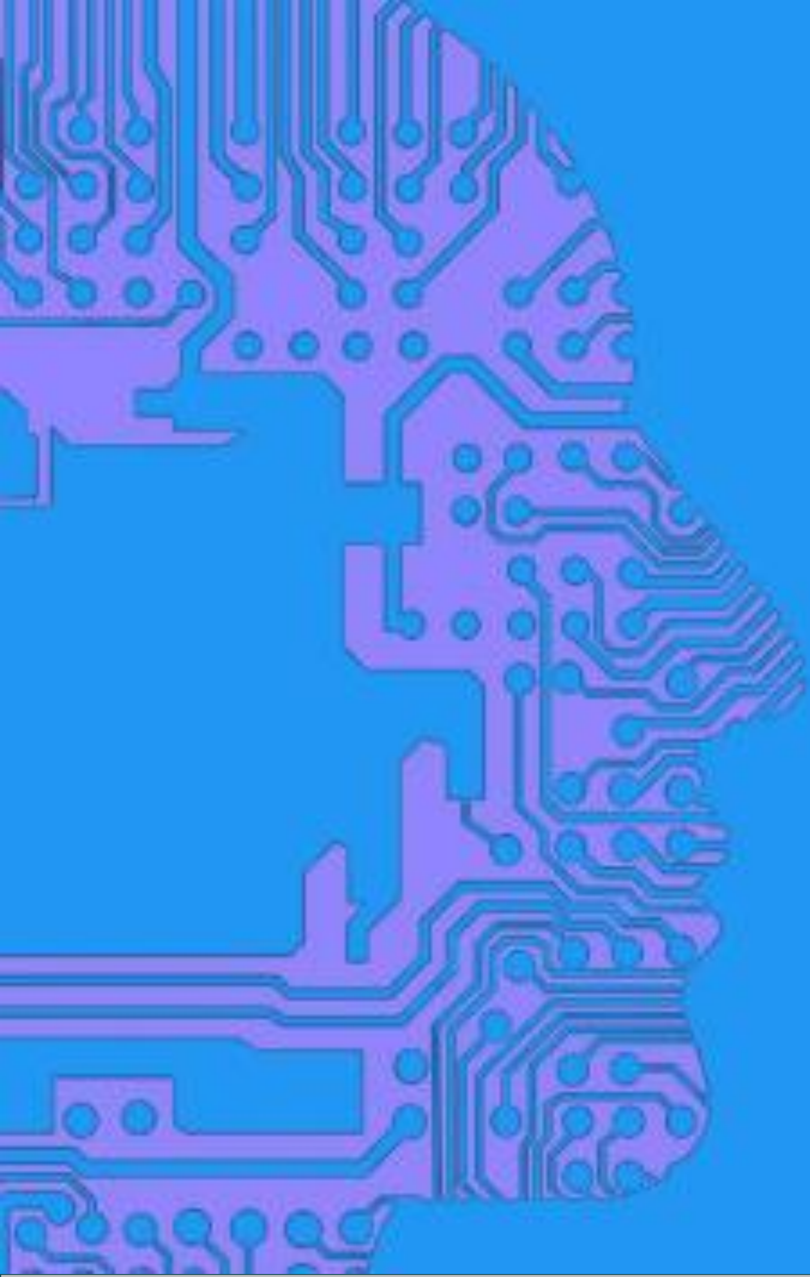


The ESP32-CAM can be used in projects aimed at detecting fake Indian currency by leveraging its camera capabilities for real-time image capture and processing. By integrating the ESP32-CAM with a machine learning model, such as the Xception model used in the Fake Indian Currency Detection project, the camera can capture high-resolution images of currency notes. These images can then be processed to detect authenticity by comparing features like security marks, patterns, and watermarks. The ESP32-CAM's Wi-Fi connectivity allows the processed results to be sent to a server or mobile device for verification, aiding in counterfeit currency detection in real time.

Start(Web)

Capture Image using ESP32 Camera

Predict (Real or Fake)




LIBRARIES USED
pandas
OS Module
TensorFlow
OpenCV
NumPy
matplotlib
Keras

REFERENCE

Google Scholar

MENDELEY DATA


THANK
YOU