



# Facial Emotion Recognition

Using Deep learning (CNN)



# Contents

Facial Emotion Recognition-Project workflow :

<b><u>Dataset</u></b>	Kaggle - <a href="#">FER-2013</a>
<b><u>Data Cleaning &amp; Analysis</u></b>	Selecting appropriate Image extension, Calculating no.of Images per Emotion
<b><u>Custom CNN Model</u></b>	Creating a Custom CNN for Image Classification & Plotting Performance Metrics
<b><u>Using Image Augmentation</u></b>	Using various augmentation technique for increasing data diversity, Improve model generalization, Reduce Overfitting
<b><u>VGGNet Transfer Learning</u></b>	Using VGG Network Architecture for better Performance Metrics
<b><u>ResNet50 Transfer Learning</u></b>	Using ResNet50 for string feature extraction, efficient training and better performance



**01**

**Dataset**

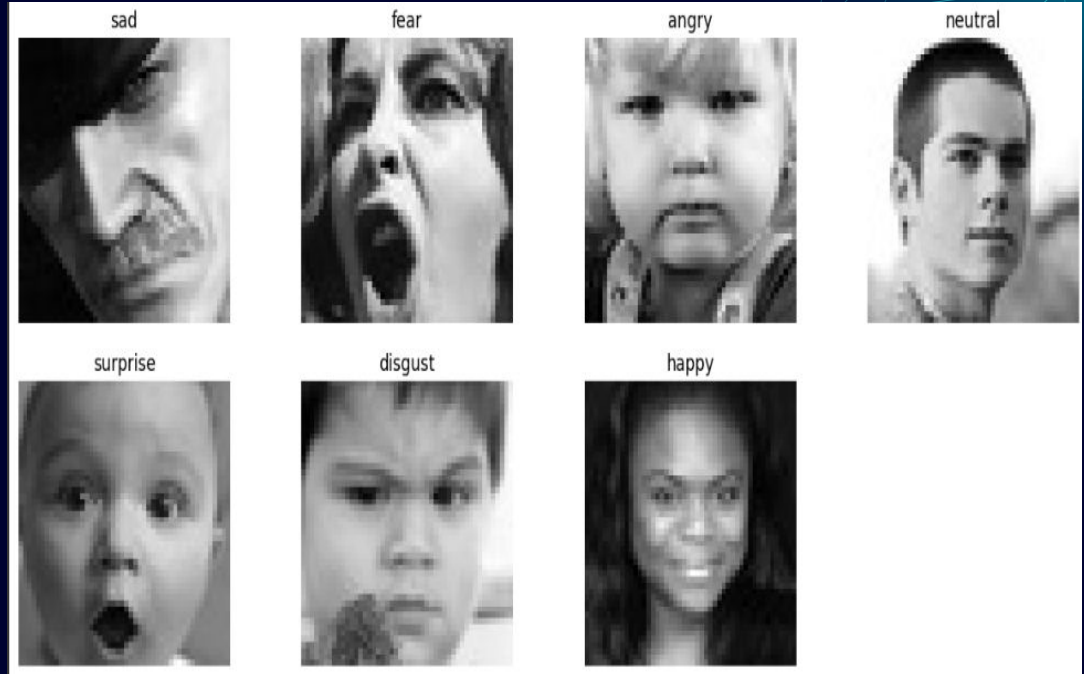


# Dataset



## Kaggle-FER

The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centred and occupies about the same amount of space in each image. The task is to categorize each face based on the emotion shown in the facial expression into one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral)





02

## Data Cleaning & Analysis

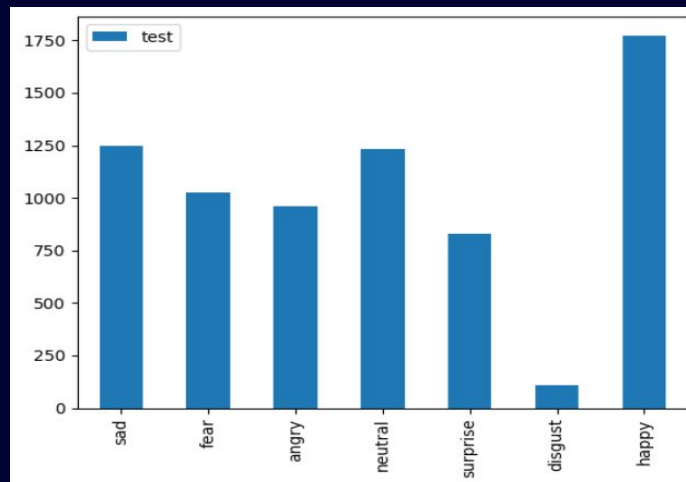
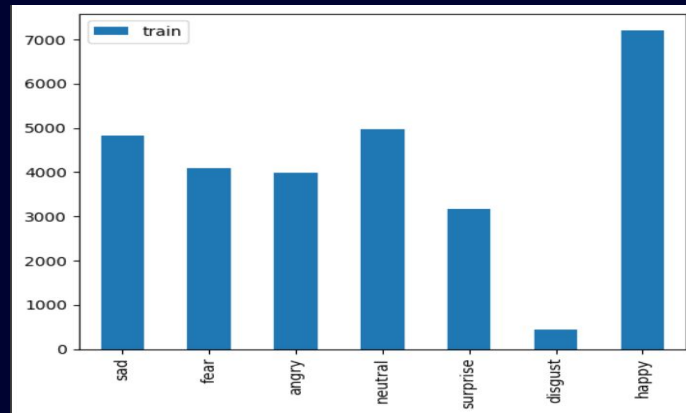
# Data Cleaning & Analysis

## Data Cleaning

Removing the images which does not have extensions like 'jpeg', 'jpg', 'png', and if the image belongs to these extensions reading it using cv2 library

## Data Analysis

Analysing each and every emotion data for training and testing, returning a pandas Dataframe containing type of emotion and its individual count.





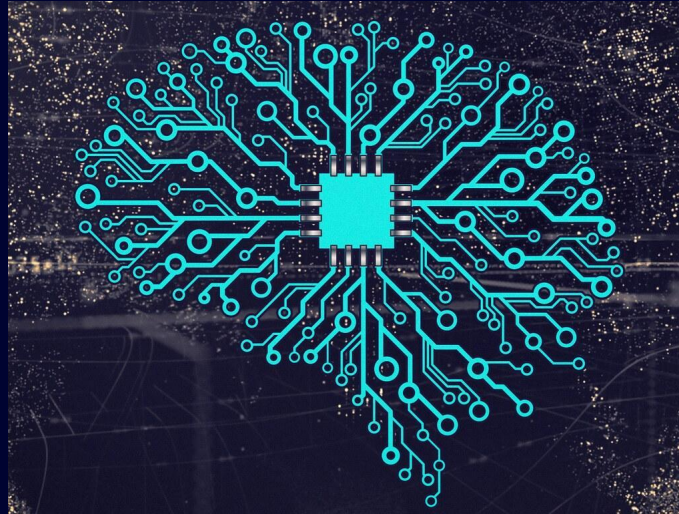
# Approach

## Custom CNN

The initial model was built using normalization, max-pooling, and ReLU activation functions, achieving 52.54% accuracy due to imbalanced data.

## Image Augmentation

To address the data imbalance, I applied techniques like rotation, flip, and zoom, leading to more diverse training data.  
Accuracy=57.93%



## VGG16

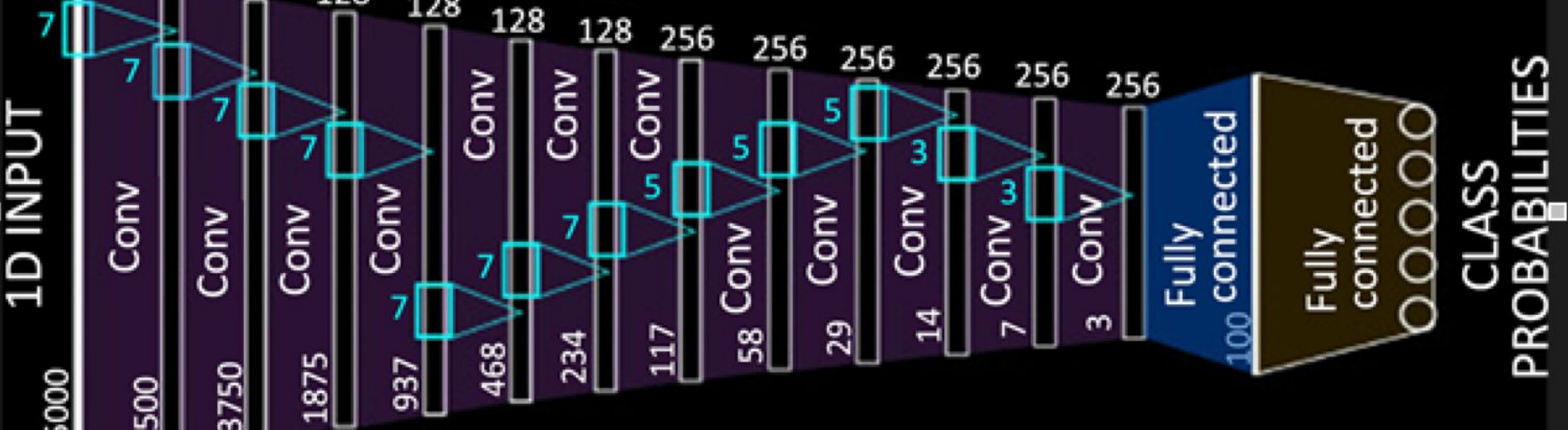
Implemented transfer learning by fine-tuning a VGG16-based model, resulting in significant performance improvements.  
With 63.53% accuracy

## ResNet50

I used the ResNet architecture, achieving a remarkable 97% accuracy for emotion detection.

# 03

## Custom CNN



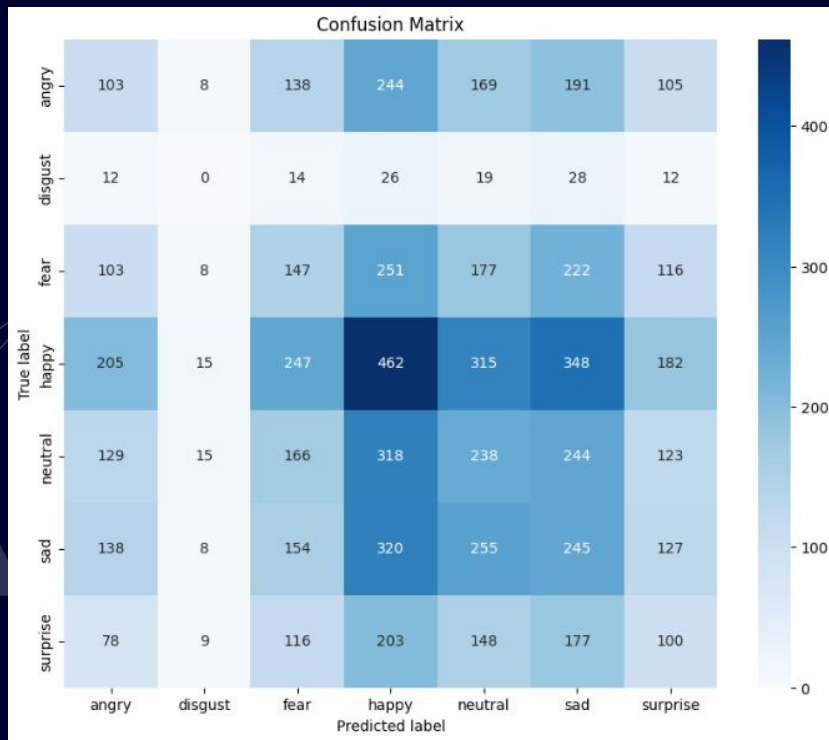


# Custom CNN

## Model Components

- **Input Shape:** (img\_width, img\_height, 1), representing grayscale images.
- **Convolutional Blocks:** Three blocks of Conv2D layers with ReLU activations.
- **Block 1:** Two Conv2D layers (32, 64 filters), BatchNormalization, MaxPooling2D, and Dropout.
- **Block 2:** Two Conv2D layers (128, 256 filters) with L2 regularization, BatchNormalization, MaxPooling2D, and Dropout.
- **Block 3:** Two Conv2D layers (512 filters each), BatchNormalization, MaxPooling2D, and Dropout.
- **Fully Connected Layers:**
- **Flatten** to convert the convolutional output into a 1D array.
- A **Dense** layer with 1024 units, **Dropout**, and ReLU activation.
- Final **Dense** layer with **Softmax** activation for class probabilities.
- **Regularization and Activation Functions**
- **Regularization:** L2 regularization is used in several Conv2D layers to reduce overfitting.
- **Dropout:** Applied after each convolutional block and before the final dense layer to prevent overfitting.
- **Activation Functions:** ReLU in intermediate layers, Softmax in the final layer.

```
359/359 [=====] - 9s 24ms/step - loss: 0.6928 - accuracy: 0.9148
113/113 [=====] - 3s 26ms/step - loss: 1.4386 - accuracy: 0.6254
final train accuracy = 91.48 , validation accuracy = 62.54
```





**04**

# **Image Augmentation**



# Image Augmentation

- **Purpose:** Data augmentation creates variability in training data, reducing overfitting applying transformations.
- **Augmentation Techniques:**
  - **rotation\_range=40:** Random rotation of images within a 40-degree range.
  - **width\_shift\_range=0.2, height\_shift\_range=0.2:** Random horizontal and vertical shifts.
  - **shear\_range=0.2:** Random shearing (like slanting).
  - **zoom\_range=0.2:** Random zooming in and out.
  - **horizontal\_flip:** Random horizontal flipping of images.
- **Rescaling:**
  - All pixel values are rescaled to a range of [0, 1] using **rescale=1./255**.
- **Validation Split:**
  - **validation\_split=0.2:** Splits 20% of the data for validation purposes.

Original Image



Augmented Image 1



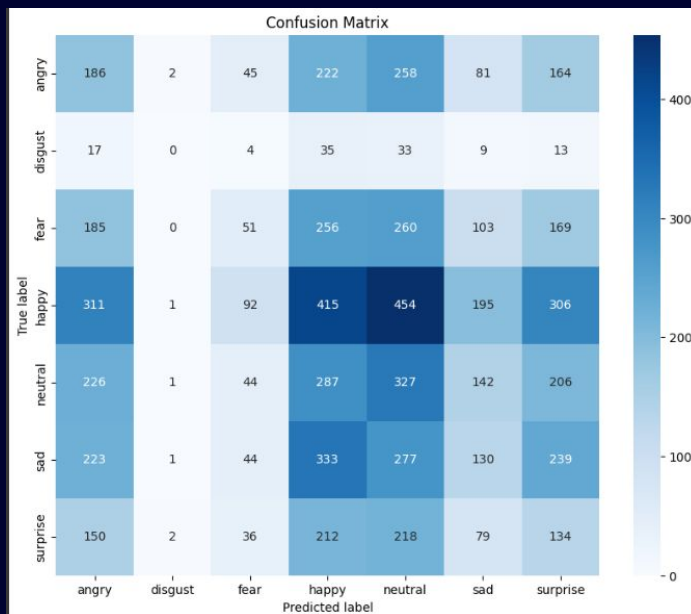
Augmented Image 2



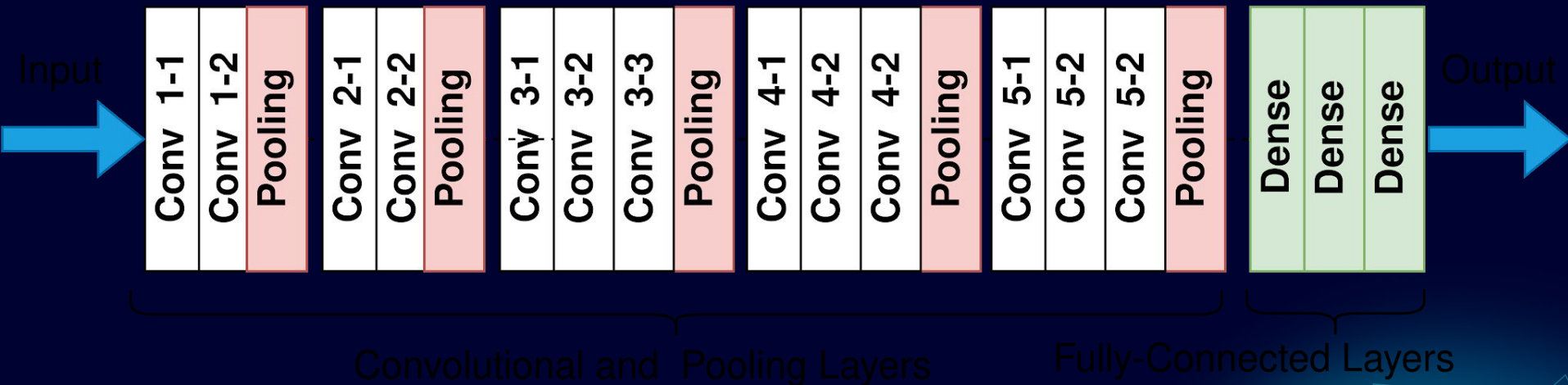
Augmented Image 3



Augmented Image 4



## VGG16 Model Architecture



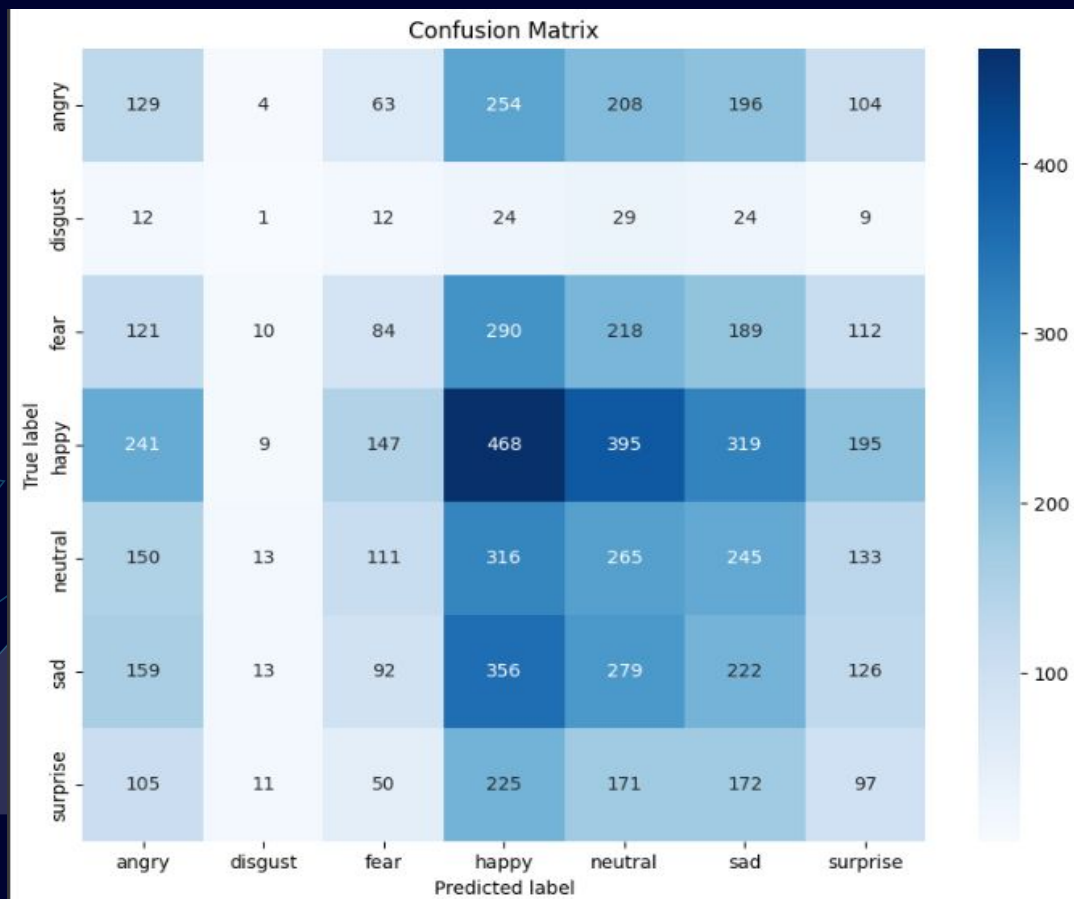
05

# VGG16 Network

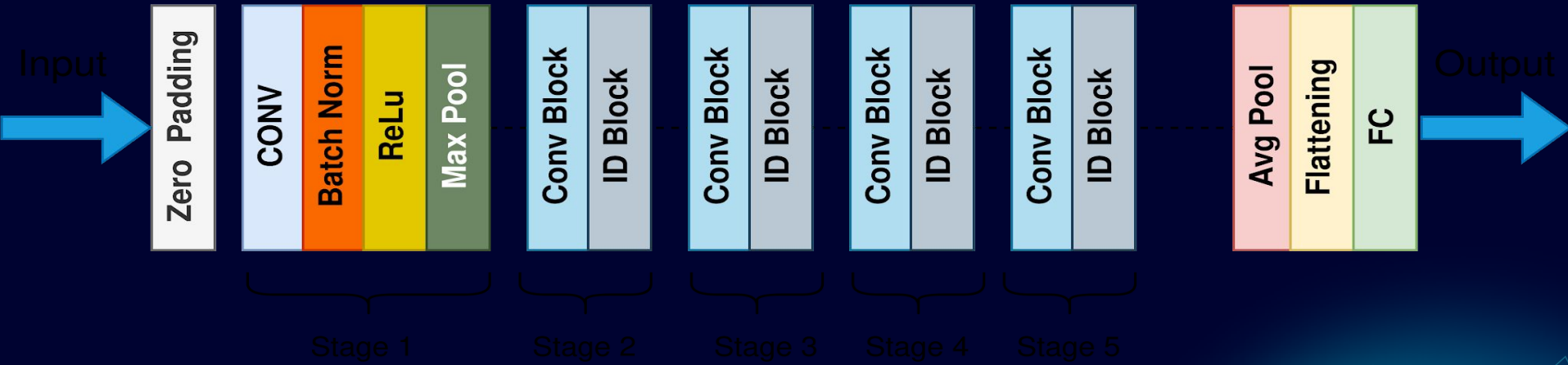


# VGG16 Network

- **VGG16 Model:**
- **VGG16(input\_shape=(224, 224, 3), include\_top=False, weights='imagenet')**: Loads the VGG16 architecture with a (224, 224, 3) input shape for RGB images.
- **Excluding Top Layers: include\_top=False** removes the fully connected layers, allowing custom top layers.
- **Pretrained Weights: weights='imagenet'** uses weights from training on the ImageNet dataset.
- Freezing Layers
- Setting Layers as Non-Trainable:
- **The code sets all but the last three VGG16 layers to be non-trainable**, meaning their weights are not updated during training.
- Purpose: Freezing early layers enables efficient feature extraction and focuses training on the last (custom) layers. It also reduces overfitting and computational overhead.
- Layer Iteration:
- **for layer in vgg.layers[:-3]**: This code line identifies the layers to freeze, excluding the last three from freezing.



## ResNet50 Model Architecture



06

**ResNet50  
Network**


# ResNet50 Network




- **ResNet50V2 as Base Model:**
- **ResNet50V2(include\_top=False, weights='imagenet', input\_shape=(224, 224, 3)):** Loads a pretrained ResNet50V2 model without the top (fully connected) layers. Uses ImageNet weights for feature extraction.
- **Adding Custom Layers:**
- **Sequential([base\_model, BatchNormalization(), GlobalAveragePooling2D()]):** Appends a new set of layers to the base model, including batch normalization and global average pooling.
- **Fully Connected Layers:**
- **Dense(512, activation='relu'):** A dense layer with 512 units and ReLU activation.
- **Dropout(0.1):** Applies dropout to reduce overfitting.
- Subsequent dense layers with 256 and 128 units, with dropout in between.
- The final dense layer has **7 units** (the number of classes) and a **softmax activation** for classification.

## Emotion Detection

Upload an image and see the predicted emotion.

Image







Clear

Submit

output

angry


Flag




Use via API  · Built with Gradio 

## Emotion Detection

Upload an image and see the predicted emotion.

Image







Clear

Submit

output

surprise

Flag

Use via API  · Built with Gradio 

# Thanks !

Soham Deshpande (BT21ECE083)  
Github-[Link to Project](#)