PROJECT TITLE :

CNN Image Classifier

A Comprehensive Guide to Build a Convolutional Neural Network(CNN) for Image Classification

Presented BY : Patricia Stanley

Intern ID : MST03-0053

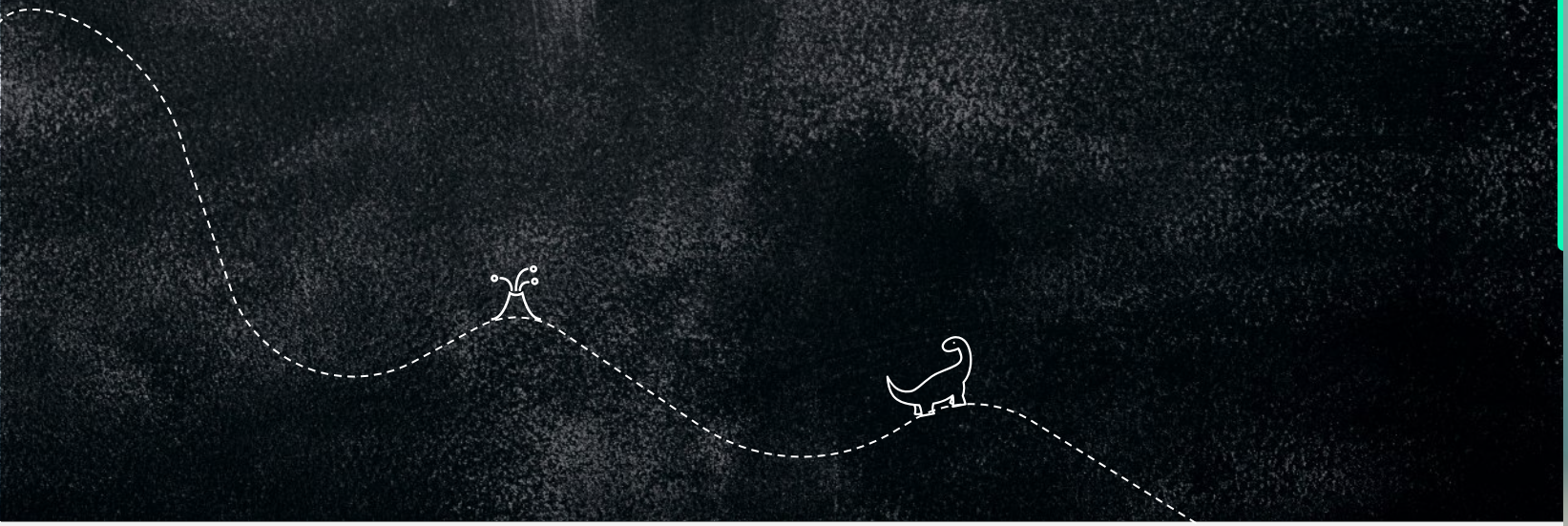
Guided BY : Urooj Khan

Project Overview:

Objective: Classify the images into “Happy” and “Sad” Categories using a Convolutional Neural Network(CNN).

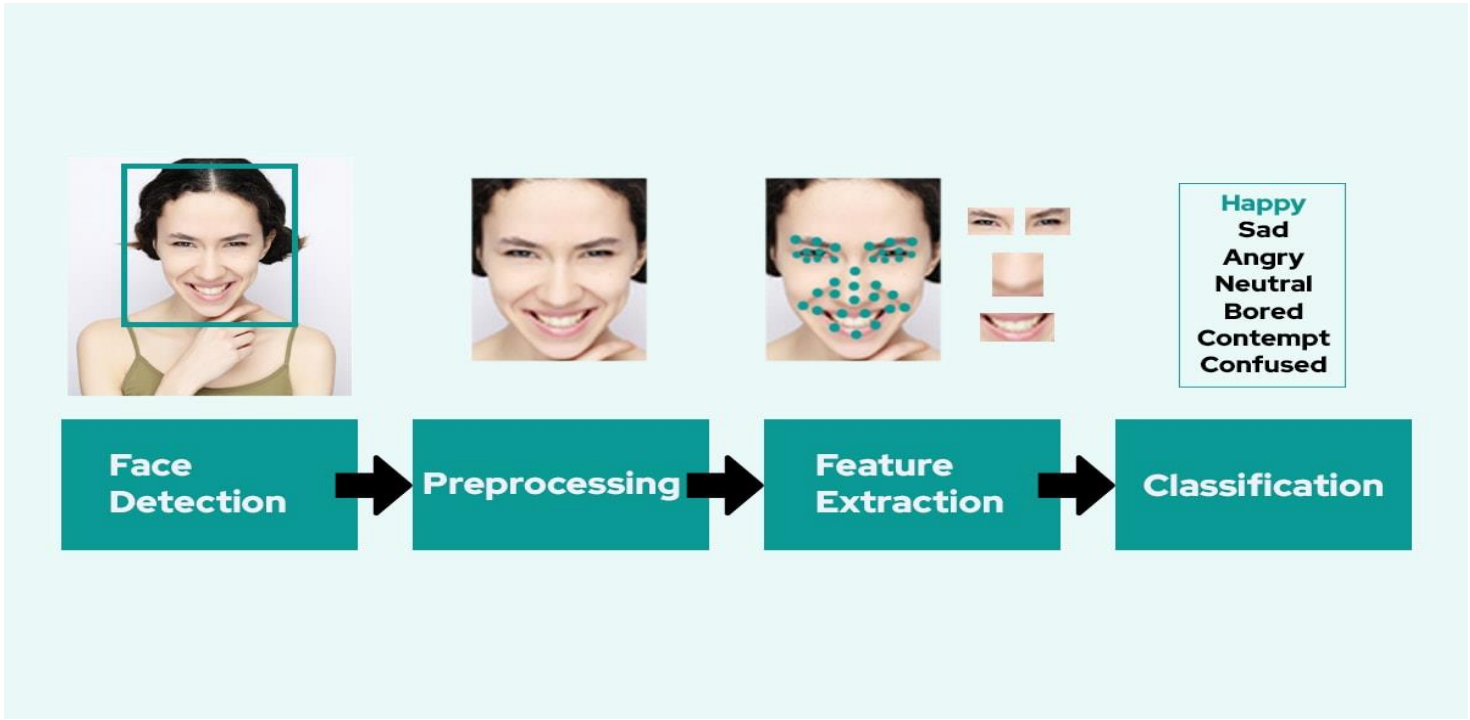
Data Source: Images are stored in a directory, stored into subdirectories based on labels.

Tools Used: TensorFlow, OpenCV, Matplotlib, NumPy



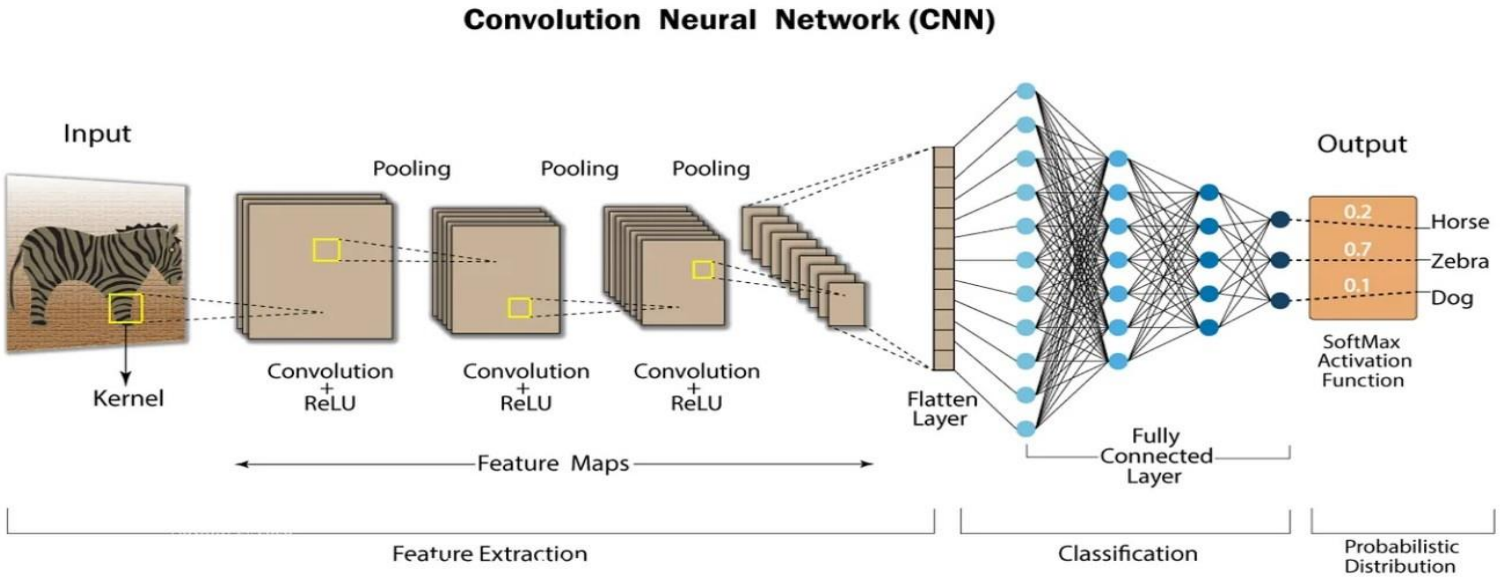
What is Image Classification?

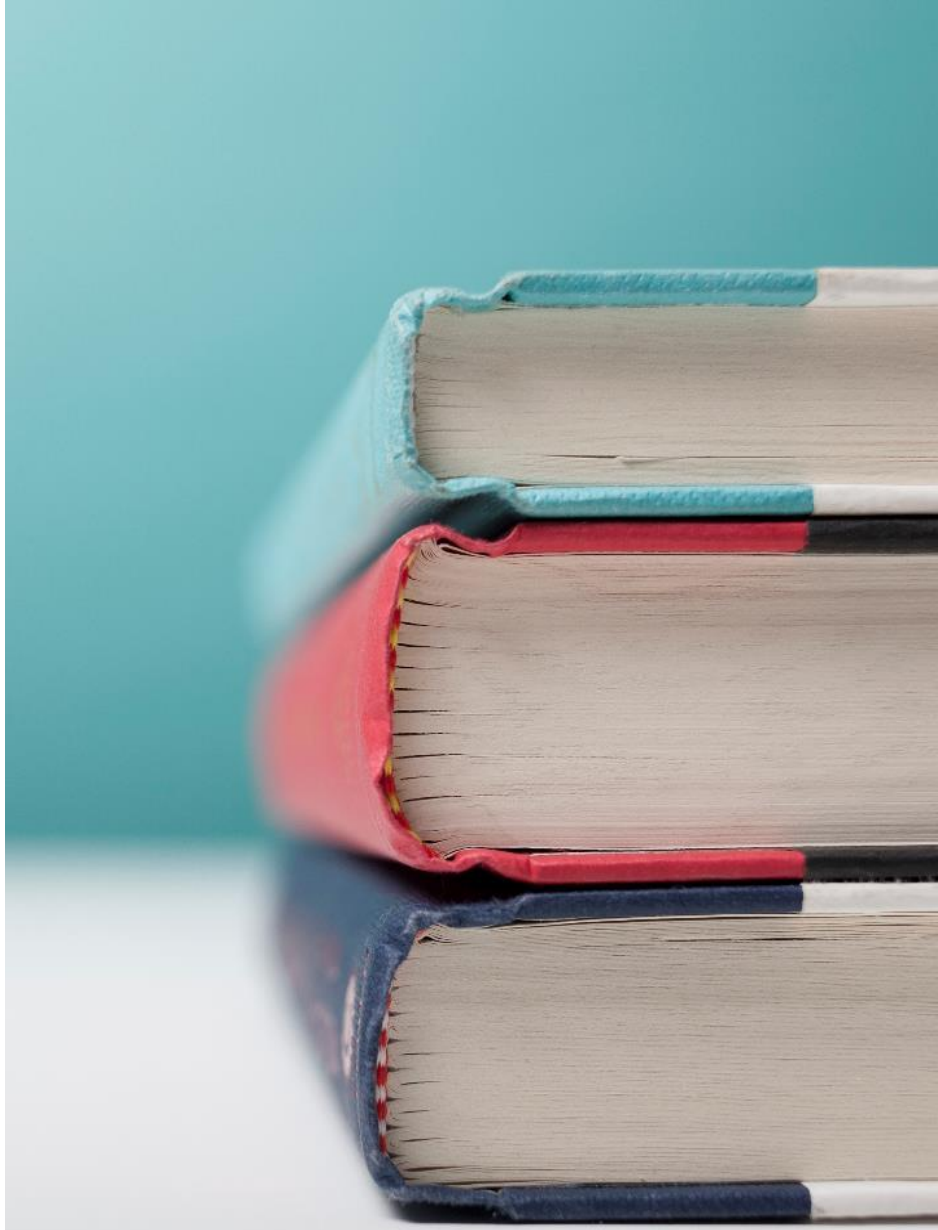
Image classification is a task in computer vision where the goal is to categorize an image into one of several predefined classes or categories. The process involves using machine learning algorithms, typically deep learning models like convolutional neural networks (CNNs), to analyze the features of an image and make predictions about its class label. This technology is widely used in various applications such as facial recognition, autonomous driving, medical diagnostics, and more.



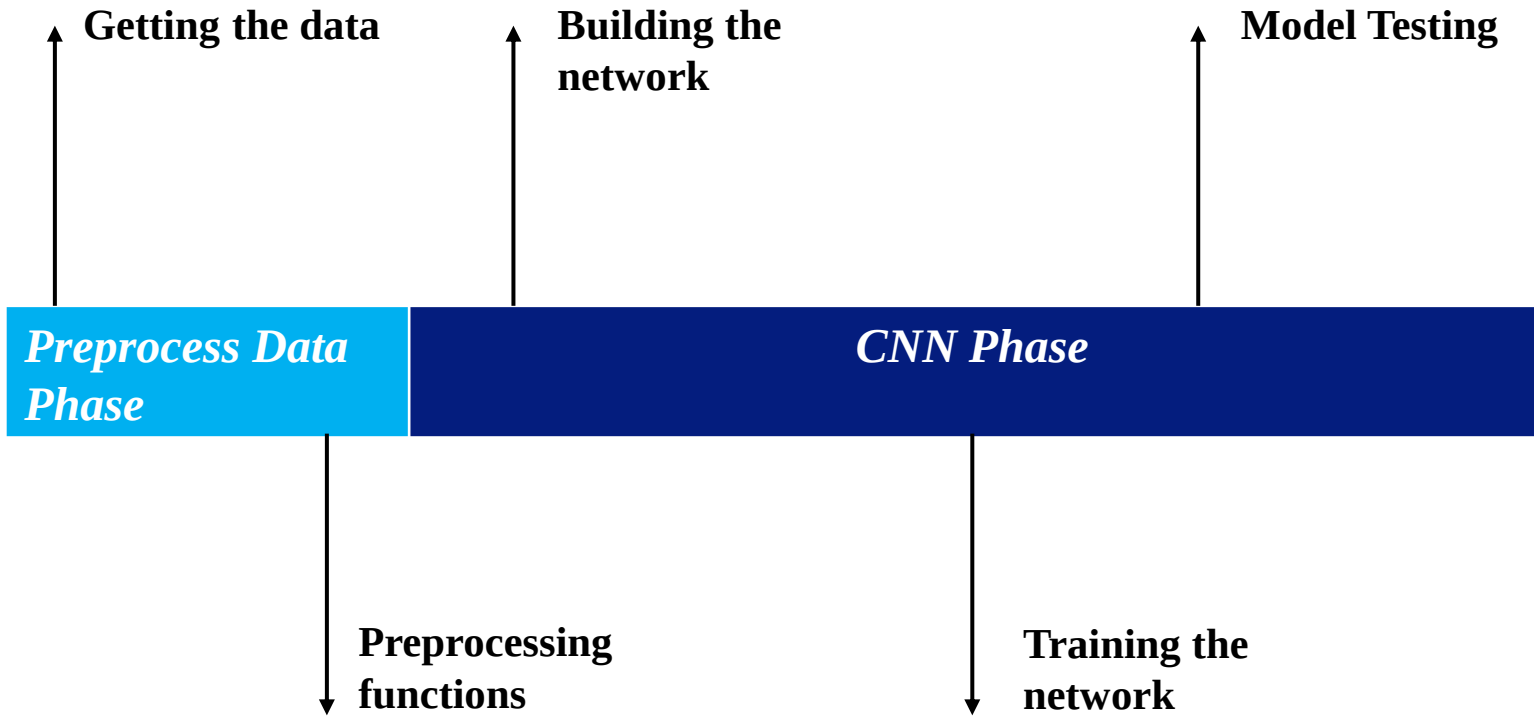
What are Convolutional Neural Network (CNN) ?

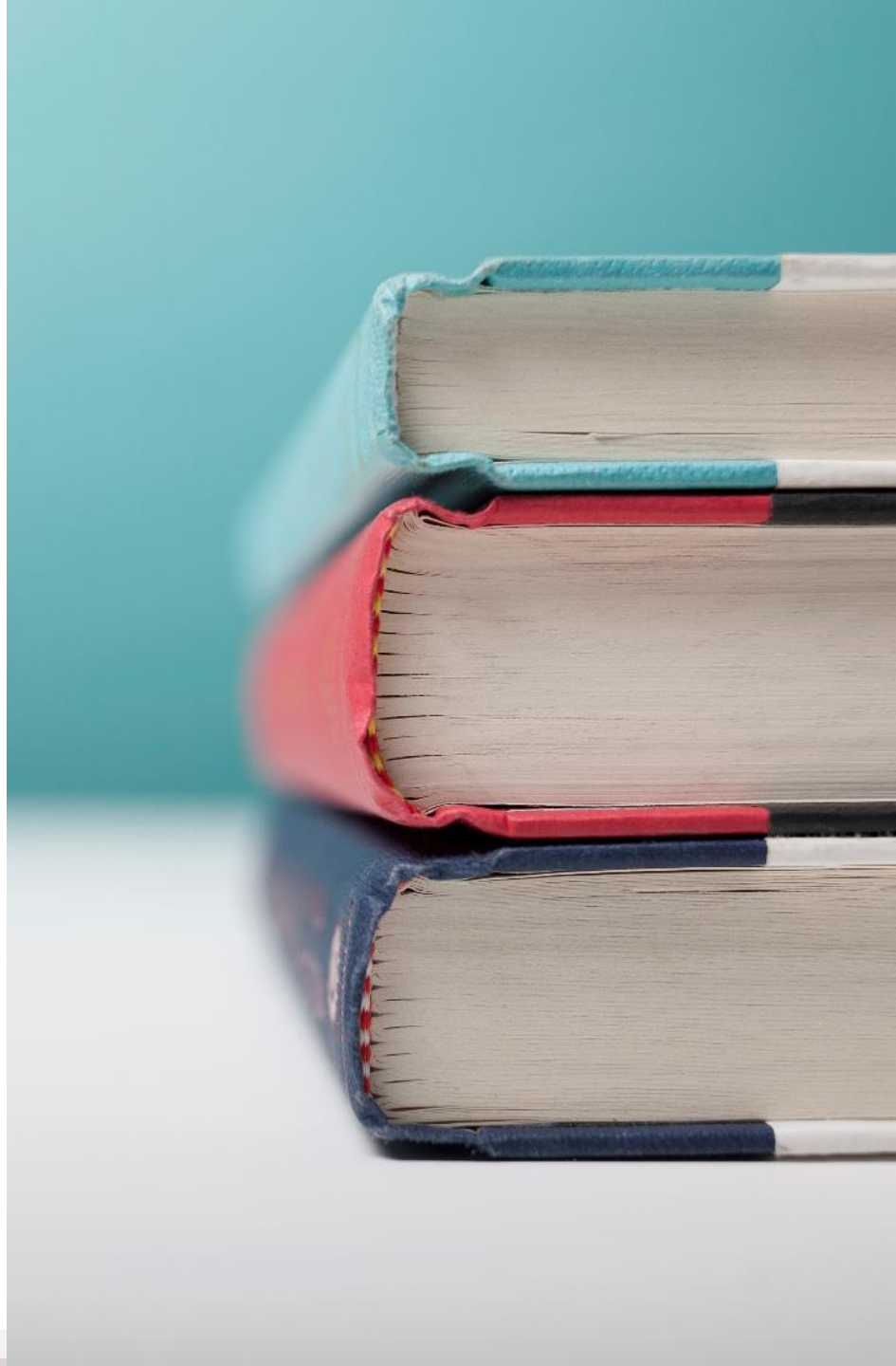
In deep learning, a convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyze visual imagery. The cnn architecture uses a special technique called Convolution instead of relying solely on matrix multiplications like traditional neural networks. Convolutional networks use a process called convolution, which combines two functions to show how one changes the shape of the other.





Phases:


Getting the data
Building the network
Model Testing
Preprocess Data Phase
CNN Phase
Preprocessing functions
Training the network



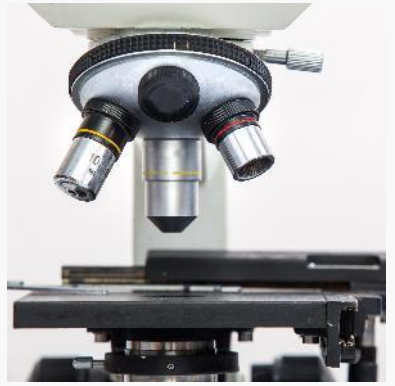
Install Dependencies and Import Libraries:

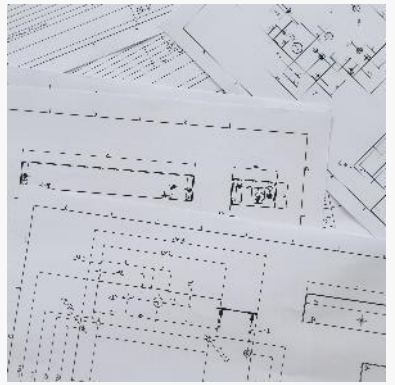
•**Dependencies**:

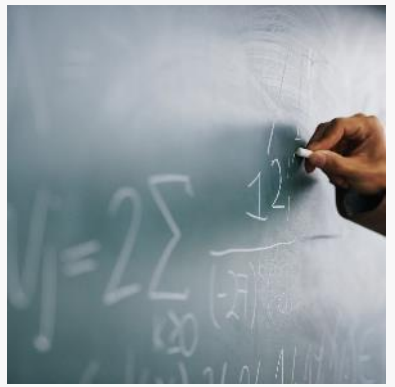
•TensorFlow & TensorFlow-GPU
•OpenCV-Python
•Matplotlib

Libraries:

•**numpy** for numerical operations
•**matplotlib.pyplot** for plotting
•**tensorflow** for deep learning
•**os** for file path operations
•**cv2** for image processing
•**warnings** to ignore warnings








About The Dataset:

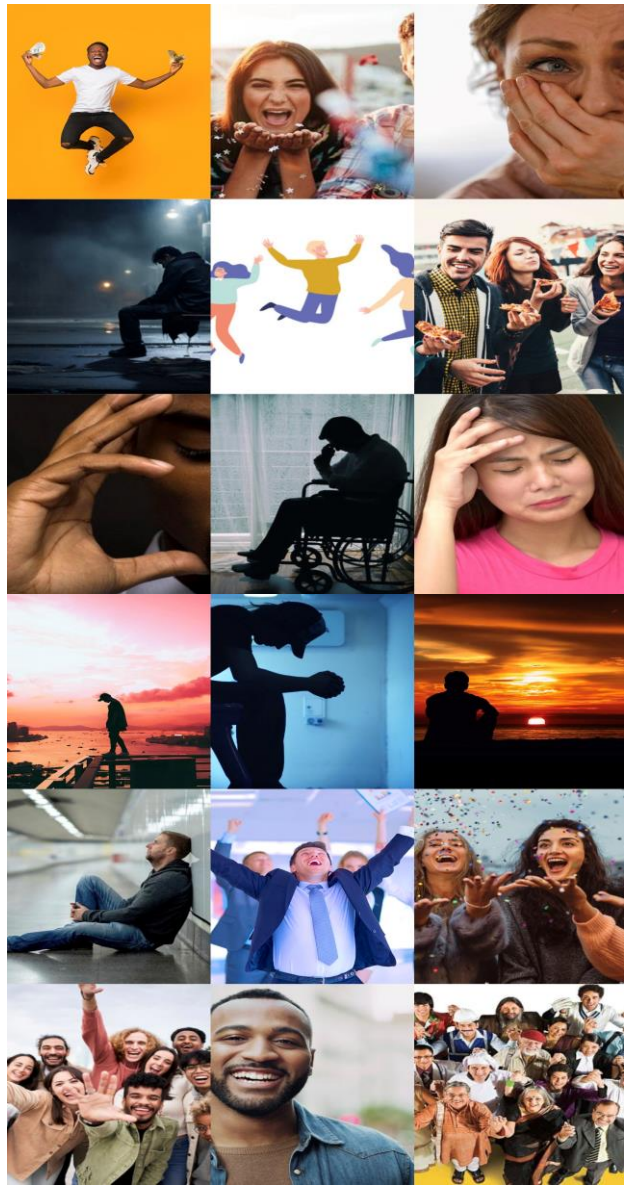
The dataset used in this project consists of images categorized into two classes: 'Happy' and 'Sad'. These images are organized in a directory structure that is suitable for loading and processing using TensorFlow's image dataset utilities. Below is a detailed description of the dataset:

1. Directory Structure

- **Root Directory**: Contains two subdirectories, one for each class ('Happy' and 'Sad').

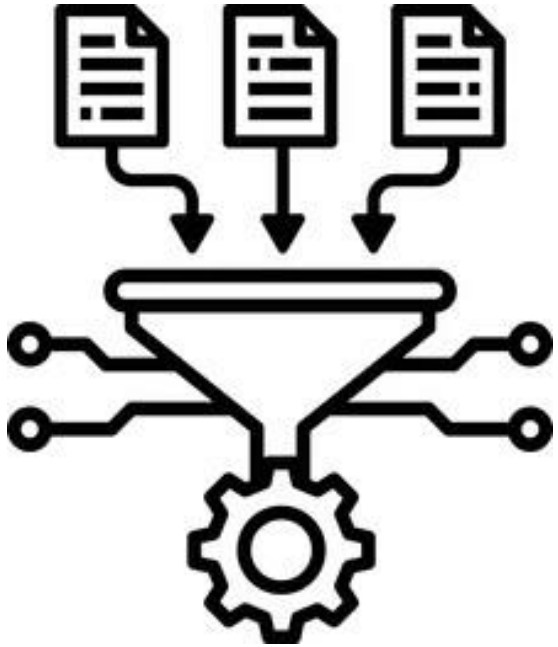
2. Image Characteristics

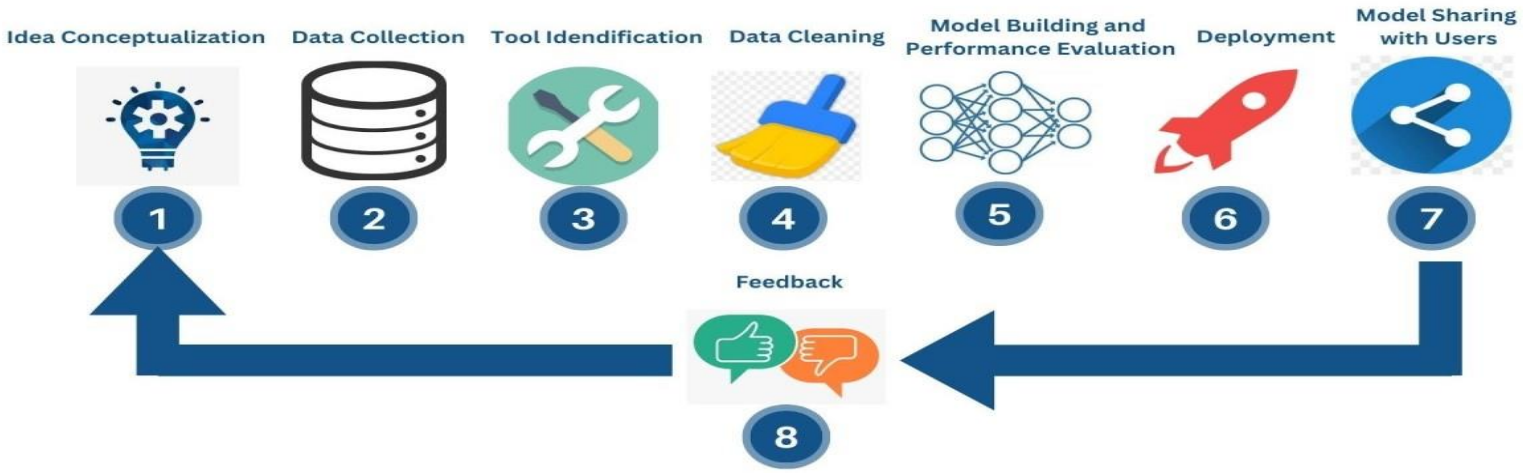
- **Classes**:
  - **Happy**: Images of individuals with happy facial expressions.
  - **Sad**: Images of individuals with sad facial expressions.
- **File Format**: Images are typically in JPEG format (.jpg).
- **Dimensions**: Images vary in size and resolution but are standardized to 256x256 pixels during preprocessing.



Preparing the Data:

- The image dataset is loaded using TensorFlow's image_dataset_from_directory function. This function automatically labels the images based on their directory names.
- To inspect the data, we convert it into an iterator and retrieve a batch of images and their labels. The images are visualized to understand the data distribution.
- Images are scaled to the [0, 1] range to facilitate better convergence during training. This is done by dividing the pixel values by 255.





Build the Model:

A Sequential model is built using TensorFlow's Keras API. The architecture includes three convolutional layers followed by max pooling layers, a flattening layer, and two dense layers. The output layer uses a sigmoid activation function to predict the binary class (Happy or Sad).

Training the Model:

- Data Splitting: The dataset is split into training, validation, and test sets in a 70-20-10 ratio. This ensures that the model can be trained and validated on separate data to evaluate its performance effectively.
- The model is trained for 20 epochs with the Adam optimizer and binary cross-entropy loss. A TensorBoard callback is used for monitoring the training process.

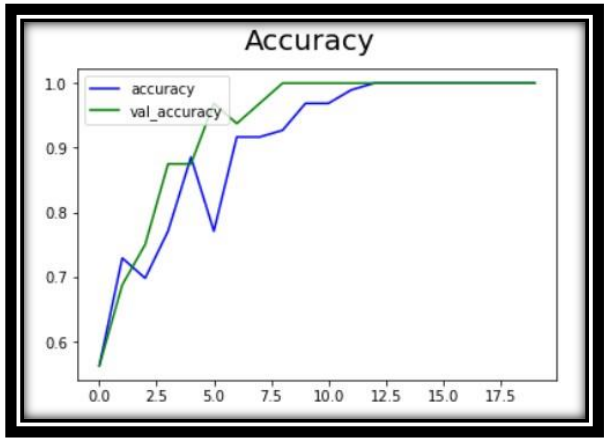
Epochs: An epoch refers to one complete pass of the entire training dataset through the model. During each epoch, the model iterates over the data, processes it in batches, and updates the model weights to reduce the loss.

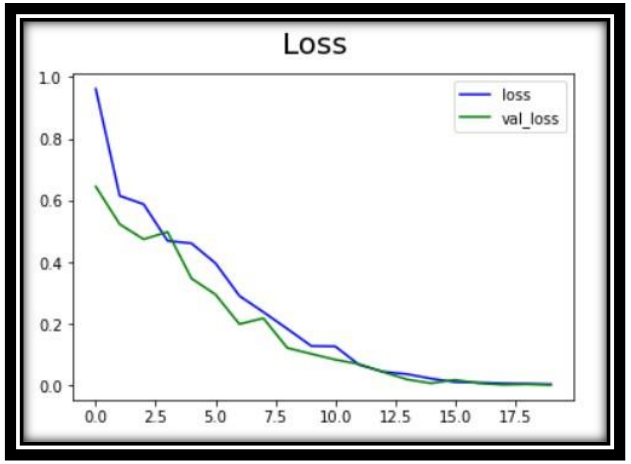
Adam Optimizer: The Adam optimizer, short for "Adaptive Moment Estimation," is an iterative optimization algorithm used to minimize the loss function during the training of neural networks.

Binary cross-entropy loss: Binary cross-entropy (also known as log loss) is a loss function used for binary classification tasks, where the target variable has only two possible values (e.g., 0 or 1, 'happy' or 'sad').

Tensorboard: TensorBoard is a tool for providing the measurements and visualizations needed during the machine learning workflow.

- The training and validation loss and accuracy are plotted to visualize the model's performance over epochs.





Model Evaluation and Prediction :

The model's performance on the test set is evaluated using precision, recall, and accuracy metrics.

Precision: The quality of a positive prediction made by the model.

Recall: A metric that measures how well a classifier or predictor can recognize positive cases in a dataset.

Accuracy metrics: A metric used in machine learning to measure how well a model performs its intended task by evaluating the correctness of its predictions

Result and Predictions:

After developing and training the Convolutional Neural Network (CNN) model for image classification, the following results were obtained:

•The model was trained for 20 epochs with a training dataset and validated using a validation dataset.

•The training process showed a gradual decrease in loss and an increase in accuracy, indicating effective learning by the model.

•The model was tested on new images, such as 'sadtest.jpg'. The preprocessing steps resized the image to the required input shape, and the model correctly classified the image with high confidence.

Conclusion

Summary:

- Successfully built and trained a CNN for image classification
- Achieved satisfactory performance metrics
- Model can classify images as 'Happy' or 'Sad’

Future Work:

- Experiment with different architectures
- Enhance dataset with more images and categories
- Deploy model for real-time image classification

Thank You