**Facial Expression Recognition**

This project implements a facial expression recognition model using the EfficientNet architecture. The model is trained on a dataset of facial images to classify them into seven basic emotions: angry, disgust, fear, happy, neutral, sad, and surprise.

1. **Dataset**

The dataset used for training and validation is the Facial Expression Recognition Dataset, available on **Kaggle:** <https://www.kaggle.com/jonathanoheix/face-expression-recognition-dataset>

1. **Model**

The model utilizes a pre-trained EfficientNet-b0 from the timm library, with the final classification layer modified to output scores for the seven emotion classes.

1. **Training**

The model is trained using the Adam optimizer and Cross-Entropy Loss. Data augmentation techniques such as random horizontal flips and random rotations are applied during training to improve robustness. The training process tracks both training and validation loss and accuracy, saving the model weights that achieve the best validation loss.

1. **Inference**

After training, the best performing model weights are loaded for inference. The model can then predict the emotion expressed in a given facial image. A utility function is provided to visualize the input image and the predicted class probabilities.

**Project Structure**

* **Facial-Expression-Dataset/:** Directory containing the training and validation image data.
* **best-weights.pt:** Saved weights of the best performing model during training.
* **Notebook cells:**

Installation of required libraries (albumentations, timm, opencv-contrib-python) and dataset cloning.

Configuration of training parameters (learning rate, batch size, epochs, device).

Loading and preprocessing of the dataset using torchvision.datasets.ImageFolder and custom augmentations.

Creation of data loaders for training and validation batches.

Definition of the FaceModel class using EfficientNet.

Implementation of training and evaluation functions.

Training loop with optimizer and model saving based on validation loss.

Inference code to load the best weights and predict emotions on new images, including a visualization function.

Requirements

Python 3.6+

PyTorch

torchvision

timm

albumentations

opencv-contrib-python

numpy

matplotlib

tqdm

How to Run

Clone the repository.

Install the required libraries using pip.

Download the dataset and place it in the Facial-Expression-Dataset directory, with 'train' and 'validation' subdirectories containing the respective emotion classes.

Run the notebook cells sequentially to train the model and perform inference.