Team :
SANJAITH - ED23B055
MANDEEP - ED23B053
KUNAL - ME23B177

Report:

**Approach 1:**
Just CNN with all images directly to create a Neural network. Just 3 layers .
0.35 => Acc

**Approach 2:**
Just CNN with few image processing (blurring) to reduce overfitting was done…(We try image hist which reduced Accuracy)...Now this time we increased number of feature to 512
We got a Acc. => 42%

**Approach 3:**
We tried with different CNN Models with different Activation Function and Optimizers and we included batch normalisation after each layer.We fixed with AdamW and ReLU and Softmaxx in the last layer. We also added Augmented Images in this stage…
We did it because of 2 reasons >increases data balance (to avoid over find so weight get balanced over all classes. We also tried many image processing techniques but we felt it reduced our Accuracy.
This time Accuracy => 50%.

**Approach 4: (*Transfer Learning*)**
After trying all approaches as given in resources we thought of giving VIT a try and it worked very good as we started getting 63 accuracy here are the steps i did for ViT and i used google pretrained model which is not trained on this dataset which is google/fit-base-patch 16-224-in21k.

**Approach 4.1: VGG Net**
We didn't get good Accuracy, we Got around 44% ,we got more Acc than this with CNNs (16Layers with 512 Features)

**Approach 4.2 : Vision Transformers**
*Tools and Libraries:*
*Hugging Face Transformers:* Used for model fine-tuning and preprocessing of images.
ViT (Vision Transformer): A pre-trained model for image classification.
Datasets & Evaluate: For dataset handling and metric calculations.
Pandas & NumPy: Data manipulation and processing.
Matplotlib & Seaborn: Visualisation of the results, particularly for confusion matrices.

1. Data Preprocessing:
Loaded the FER-2013 data from a CSV file where the image pixel data was stored as space-separated strings.
Converted the string pixel data into NumPy arrays and reshaped them into 48x48 images.

Each grayscale image was expanded into an RGB format (3 channels).
The dataset was split into training, validation, and test sets.
2. Feature Extraction:
The images were processed using the ViTFeatureExtractor, resizing the images to 224x224 (as required by the ViT model). This step standardised the input size for the Vision Transformer model.

3. Model Setup:
I tried other models but google one only performed well without any errors. The model was fine-tuned to classify images into the seven emotion categories.
4. Training:
The model was trained for 15 epochs with a learning rate of 1e-5. The evaluation and model saving strategies were based on the accuracy metric and AdamW optimiser was used and for loss CrossEntropyLoss.

5. Evaluation:
After training, the model was evaluated on the test dataset.
Confusion matrices were used to visualise the model's performance across the seven emotion categories.
6. Inference on New Data:
A new test dataset (without emotion labels) was provided.
The same preprocessing steps were applied to this dataset.
The model predicted the emotion for each image, and the results were saved in a CSV file.

We did many image processing (pre-processing to test and train images nothing worked out well)
*Batch Size 8(Train+Val) and LR = 2*e^-5 gave…*

*Flops :*
We tried VIT model image Augmentation… But that actually reduced our Accuracy to 58%
But We tried As much as possible.
Trouble with colab we got up to *69%* Accuracy with Train (Augmented dataset ) split in to 20% validation set. When we preprocessed images to extract Vit features it peaked the RAM memory and Runtime got disconnected…
With this multiple trials we got Score of 59%-63%

**Approach 5:**
We Added *Swin Transformers:*
It has a Good Accuracy than ViT becoz it uses

- ☐ *Hierarchical Structure*: Swin Transformers divide the image into smaller patches (like ViT), but they do it hierarchically. Initially, small patches are processed, and at each subsequent stage, the patch size is increased (merging adjacent patches) while maintaining a hierarchical feature representation.
- ☐ *Window-based Attention*
- ☐ *Flexible for Various Tasks*

So training with this we got 66% Accuracy.
*Mixup Data Augmentation:* The model was improved further to get more generalisation through the use of Mixup. Mixup is an algorithm that creates new training examples by mixing up two images and their corresponding labels. In the process, it smoothes the decision boundary of the model and thus prevents overfitting.
*Optimizer and Scheduler* During training, we utilised the *AdamW* optimizer since it enables adaptive learning rates and weight decay, two key factors against overfitting. To adjust the model for optimal predictions, a learning rate scheduler reduces the learning rate across the training loop in real-time

Acc. => 65%

https://github.com/sanjaithg/AI-Club-GUILD-CONTEST <=REPO