# Emotion-Recognition

## Installation Instructions

Trained and tested on python 3.6

```
pip install -r requirements.txt
```

Put the unzipped kaggle data in the `data/` folder. The fer2013.csv file must be at the path `data/fer2013/fer2013/fer2013.csv`. The path to the csv can also be manually specified ( Look at Running the code instructions)

Alternatively, on linux

```
bash scripts/download_data.sh
```

can be run to download and store the data in the needed location

## Running the code!

The code for training and testing has been split into train.py and test.py

Several utilities and supporting code used in these files have been provided in the following structure

```
-train.py
-test.py
-models/
    -blocks.py # IMPORTANT: Definition of the block based model definition used
    -model.py # IMPORTANT: Model definition based on the block structure

-config/ # IMPORTANT: Folder containing the json config of all the models.

-utils/ # Folder containing utilies modules
    -utils.py # Utility scripts for preprocessing, training and eval
    -metrics.py # Contains Metrics class to store different metrics while training and testing
    -backprop.py # Contains code to generate saliency maps using vanilla backpropagation
    -viz.py # Contains code to visualize activations, filters and other visualization utilities.
-data/ # Folder containing the downloaded data.
-results/ # Folder containing best model.
-scripts/ # Bash scripts to run expts
-tests/ # Additional code to run tests and generate model graph viz.
```

## Training Code

To run the training code with default options:

```
python train.py
```

Detailed usage:

```
usage: train.py [-h] [--data_path DATA_PATH] [--augment AUGMENT]
                [--model_config MODEL_CONFIG] [--epochs EPOCHS]
                [--batch_size BATCH_SIZE] [--train_split TRAIN_SPLIT]
                [--balanced_loss BALANCED_LOSS] [--loss LOSS] [--wandb WANDB]

optional arguments:
  -h, --help            show this help message and exit
  --data_path DATA_PATH
                        Path to the full dataset
  --augment AUGMENT     Enable data augmentation
  --model_config MODEL_CONFIG
                        Path to the model configuration json
  --epochs EPOCHS       Number of epochs to train
  --batch_size BATCH_SIZE
                        Batch size
  --train_split TRAIN_SPLIT
                        Train-valid split
  --balanced_loss BALANCED_LOSS
                        if True, weights losses according to class instances
  --loss LOSS           Type of loss to be used
  --wandb WANDB         Wandb integration
```

The `--data_path` can be changed as mentioned earlier.

## Testing Code

To run the testing code with default options:

```
python test.py
```

Detailed usage:

```
usage: test.py [-h] [--data_path DATA_PATH] [--model_config MODEL_CONFIG]
               [--load_model LOAD_MODEL] [--test_split TEST_SPLIT]

optional arguments:
  -h, --help            show this help message and exit
  --data_path DATA_PATH
                        Path to the full dataset
  --model_config MODEL_CONFIG
                        Path to the model configuration json
  --load_model LOAD_MODEL
                        Path to the model tar file to load
  --test_split TEST_SPLIT
                        Label of the test split to evaluate on
```

## Visualizations

Run

```
jupyter-notebook
```

Then open the `Visualizing_CNNs.ipynb` in the jupyter notebook environment.

A pdf of the notebook can also be found at `Visualizing_CNNs.pdf`

# Code walkthrough

The model definition structure can be found under `config/` folder. The benefit of the model definition structure is that it can be really easy to change the model structure while running experiments and no changes to code need to be made to do so. This ensures a very scalable approach to experimentation.

The json files

```
-config/
    -Baseline.json # Baseline configuration
    -kernelSize_exp/
        -53kernel.json
        -55kernel.json
        -73kernel.json
        -75kernel.json
    -nLayers_exp/
        -5blocks.json
        -6blocks.json
        -7blocks.json
    -normalization_exp/
        -BatchNorm.json
        -InstanceNorm.json
```

can be referred to see what parameters in terms of model architectures (Normalization, Number of layer, filter sizes and channels) are used. Detailed descriptions of how the json files work can be found in the files `models/blocks.py` and `models/model.py` .

Other parameters experimented with can be found as commandline arguments to the `train.py` script.

The experiments and results are synced with Weights and biases

The generated report can be found here https://app.wandb.ai/surajpai/FacialEmotionRecognition/reports/Facial-Emotion-Recognition--

VmlldzoxMTYyOTk