Detect human emotion through the images

**Adelene Ng, Wong Cheng Kwan, Perumal Srinivas**

# Introduction

The objective of this project is to develop a deep learning model to detect the human facial emotion through images.

Some of the use cases [2] that are directly applicable to this space are:

* Detecting the onset of drowsiness in drivers and alerting the driver to stop for a break, for example.
* We can also use this in a candidate-interviewer interaction session to capture a candidate’s mood and to access their personality traits.
* Facial emotion recognition can also be used in market research that observes a user’s reaction whilst interacting with a brand or product. Visual cues are more objective than verbal methods.
* It can also be used for testing video games by analysing a live feed of the user and detecting his/her facial emotions. This will help the game content producer understand which emotions are experienced at what points in the game

# Problem Framing

We want the machine learning model to be able to recognize different facial emotions. Given an image or video footage of people, it is able to predict the 7 different facial emotion expressions: angry, disgust, fear, happy, neutral, sad and surprise.

If we solve this using classical machine learning, we will have to define facial features such as eye, nose and mouth and let the system identify which features are more important in detecting a particular emotion; whereas deep learning automatically finds out the features which are important.

In short, classical vision systems involve a human telling a machine what should be there versus a deep learning algorithm which automatically extracts the features of what is there.

We will apply Convolutional Neural Networks (CNNs) for images and videos. This works much better compared to classic neural networks because the convolutional layers in CNNs layers take advantage of inherent properties of images. It takes advantage of local spatial coherence of images. It is able to significantly reduce the number of operations needed to process an image by using convolution on patches of adjacent pixels. There are also pooling layers, which allow the image to be downscaled, thus reducing image size. In classical neural networks, image vectors cannot be downscaled because there is no coherence between an input and the one next to it.

# Datasets

For this project, we will use the Facial Expression Recognition - 2013 dataset. This can be downloaded from [Kaggle](https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data).

The data consists of **48x48** pixel grayscale images of faces. The faces have been automatically registered so that the face is approximately 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 (see Table 1)

|  |  |
| --- | --- |
| Emotion Identifier | Emotion |
| 0 | Angry |
| 1 | Disgust |
| 2 | Fear |
| 3 | Happy |
| 4 | Sad |
| 5 | Surprise |
| 6 | Neutral |

Table 1 – Facial Emotion Categories

The dataset is presented as two CSV files: train and test.

“train.csv” contains two columns, "emotion" and "pixels". The "emotion" column contains a numeric code ranging from 0 to 6, inclusive, for the sentiment that is present in the image. The "pixels" column contains a string surrounded in quotes for each image. The contents of this string a space-separated pixel values in row major order.

test.csv contains only the "pixels" column and the objective here is to predict the emotion column.

The training set consists of **28,709 labelled images**, and **7178 images** in the **test set**.

There are also other publicly available datasets for facial emotion detection. The other top seven datasets [1] available for this problem space are listed below:

1. AffectNet
2. Ascertain
3. Dreamer
4. Extended Cohn-Kanade Dataset
5. Emotic
6. Google Facial Expression Comparison Dataset
7. K-EmoCon

Unlike the FER-2013 dataset which can be easily downloaded from Kaggle, some of these datasets require special request to download, with a waiting period because of too many requests.

Another dataset that has been suggested is by Jonathan Oheix. This can also be downloaded from [Kaggle](https://www.kaggle.com/jonathanoheix/face-expression-recognition-dataset). These are provided as jpeg grey scale images and split into train and validation folders. Under each folder, the images are organized and stored under their respective tagged folders: angry, disgust, fear, happy, neutral, sad and surprise respectively. Each jpeg image is **48 x 48** pixels. The training set consists of a total of **28,821** images and the validation set contains **7086** images

# Machine Learning Approaches

We plan to use the following deep learning architectures for this project:

* VGGNet16
* MobileNetV2
* DenseNet121

Once we get the base model built, we can then run different experiments and fine tune the models to the desired accuracy. The best performing model will then be used for final deployment.

We will not be using transfer learning in this project as this does not allow us to fine tune the earlier layers.

# Experiments and Evaluation Metrics

This project is considered a multi-class classification problem, since we are trying to predict **seven** different facial emotions. We will use the following performance metrics to help in our model evaluation. These are:

* Accuracy: This value tells us how accurately our model performs in predicting the different facial expressions.
* Confusion Metric: Using this metric can help us visualize which classes are more influential over others or towards which class the model is more predisposed. This will give us a clear representation of the model prediction result.
* Multi-Class Log Loss: We will use categorical cross entropy as our loss function since this is a multiclass classification problem. It is well suited for this class of problems since one example can be considered to belong to a specific category with probability 1, and to other categories with probability 0. The goal here is to reduce the multi-class log loss/cross-entropy loss.

# Performance tuning and optimization

These are some of the parameters we can tune to improve the CNN model performance:

* Number of epochs
* Vary the batch size
* Learning rate
* Dropout layer
* Use different optimizers

# Deployment

Once we have determined the best model, this saved model can then be deployed to a web server. Inference happens when we call a RESTful API which is written to access the model and return a result classifying the emotion of an image that is uploaded to the site. We can also do real time detection of emotions on a video stream.

# Reference

1. [Top 8 Datasets Available for Emotion Detection One Must Know (analyticsindiamag.com)](https://analyticsindiamag.com/top-8-datasets-available-for-emotion-detection/)
2. https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data
3. [Facial Emotion Detection using AI: Use-Cases | by Shashank Gupta | Towards Data Science](https://towardsdatascience.com/facial-emotion-detection-using-ai-use-cases-248b932200d6)
4. <https://www.dynam.ai/what-is-computer-vision-technology/>
5. [Why are convolutional neural networks better than other neural networks in processing data such as images and video? - Quora](https://www.quora.com/Why-are-convolutional-neural-networks-better-than-other-neural-networks-in-processing-data-such-as-images-and-video)
6. <https://towardsdatascience.com/increase-the-accuracy-of-your-cnn-by-following-these-5-tips-i-learned-from-the-kaggle-community-27227ad39554>
7. <https://towardsdatascience.com/the-quest-of-higher-accuracy-for-cnn-models-42df5d731faf>
8. <https://jonathan-hui.medium.com/improve-deep-learning-models-performance-network-tuning-part-6-29bf90df6d2d>
9. <https://towardsdatascience.com/increase-the-accuracy-of-your-cnn-by-following-these-5-tips-i-learned-from-the-kaggle-community-27227ad39554>
10. <https://medium.com/mlearning-ai/7-best-techniques-to-improve-the-accuracy-of-cnn-w-o-overfitting-6db06467182f>
11. <https://www.analyticsvidhya.com/blog/2021/01/building-a-cnn-model-with-95-accuracy/>
12. <https://stackoverflow.com/questions/62077360/how-do-i-augment-images-using-keras-imagedatagenerator>
13. <https://medium.com/featurepreneur/data-augmentation-using-keras-preprocessing-layers-6cdc7d49328e>