# Emotional Recognition Using Convolutional Neural Networks

Zaris Ovidiu Neamtiu

January 2023

## 1 Introduction

### 1.1 Context

This project attempts to provide a usable piece of software that uses a trained Convolutional Neural Network which performs emotional recognition on the end user based on facial expressions, and then displays the results back to the user.

### 1.2 Motivation

Thought and Emotional Recognition is one of the next steps in integrating programmed machines in our daily lives. Accurate emotional reading performed by devices can enable them to provide a much more personalised service, tending to users in exactly the way that they need in a seamless manner.

### 1.3 Objectives

The main objective of this project is to bring forth a possible solution to the need of emotional integration between the end user and a piece of software.

## 2 State of the art

Literature provides us with the following paper: Facial Emotion Recognition: State of the Art Performance on FER2013 by Yousif Khaireddin and Zhuofa Chen. They use the VGGNet architecture, fine-tune its parameters and experiment with various optimization methods.

### 2.1 Dataset, Preprocessing, and Augmentation

One of the optimization methods used is data augmentation. This augmentation includes rescaling the images up to $\pm 20\%$ of its original scale, horizontally and vertically shifting the image by up to $\pm 20\%$ of its size, and rotating it up to $\pm$ 10 degrees. Each of the techniques is applied randomly and with a probability of 50%. After this, the image is then tencropped to a size of $40 \times 40$, and random portions of each of the crops are erased with a probability of 50%. Each crop is then normalized by dividing each pixel by 255.

### 2.2 Training and Inference

They run all experiments for 300 epochs optimizing the cross-entropy loss. They vary the optimizer used as well as the learning rate schedulers and maintain other parameters constant.

They also use a fixed momentum of 0.9 and a weight decay of 0.0001. All experiments are run with gradient scaling to prevent gradient underflow. The models are evaluated using validation accuracy and tested using standard ten-crop averaging.

### 2.3 Results

Figure 3 shows the validation and testing accuracies attained by their models. All runs here use the best performing optimizer (also tested by them), SGD with Nesterov momentum. The first thing to note is that Reducing Learning Rate on Plateau (RLRP) performs best. It achieves a validation accuracy of 73.59% and a testing accuracy of 73.06%. This is already surpassing the previous single-network state-of-the-art performance (to their best knowledge).

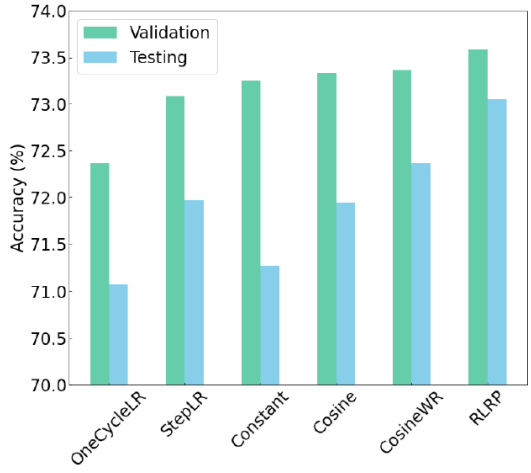



*Figure 3 VGGNet performance using different LR schedulers. The green bars show the final validation accuracies and the blue ones show the corresponding testing accuracies.*

## 3 Proposed solution

### 3.1 Structure of the network

Since this is an image classification issue, the deep-learning model chosen is a CNN. The chosen library for creating the model is the Keras library from Python.

The types of layers comprising the model are the following:

1.) Conv2D: The number of filters used ranges from 32 to 128, and the activation function is ReLU. The ReLU function is computationally efficient as it only requires a simple threshold operation, and it comes with the advandage of solving the vanishing gradient problem.

2.) MaxPooling2D: Pool size used is (2, 2). This layer is used in order to reduce the spatial dimensions of the input, which in turn reduces the number of parameters in the model and the amound of computation requred.

3.) Dropout: Used in order to prevent overfitting. There are three layers of this type used in the model, the first two with a rate of 0.25, and the last one with a rate of 0.5.

4.) Flatten: Used to flatten the output of the previous layers, so that it can be fed into a dense layer.

5.) Dense: There are two layers of this type in the model. The first one, with 1024 neurons and a ReLU activation function, enables the model to learn more complex representations of the input image and also to reduce the dimensionality of the input, in order to make it more manageable for the final output layer. The second and last one, with seven neurons, is used as the output layer. Since the model preforms image classifcation, the activation function used is "softmax". This enables the last layer to provide probabilities that an input photo belongs to one of seven

classes.

I used the Adam optimizer, as it is computationally efficient and an overall robust optimizer that can handle multiple situations, such as sparse gradients. The learning rate is set to 0.0001. Because we are in a multi-class classification situation and the softmax function is used, the chosen loss function is categorical cross-entropy. It is used to compute the difference or "loss" between the predicted class probabilities and the true class labels.

### 3.2 Data set

The data set used is the FER-2013 data set. The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centred and occupies about the same amount of space in each image. The training set consists of 28,709 examples and the public test set consists of 3,589 examples. There are 7 classes, each corresponding to a feeling (anger, disgust, fear, happiness, neutral, sadness and surprise).

The haar cascade method is used in order to detect faces in each frame of a supplied video or coming straight from the webcam feed.

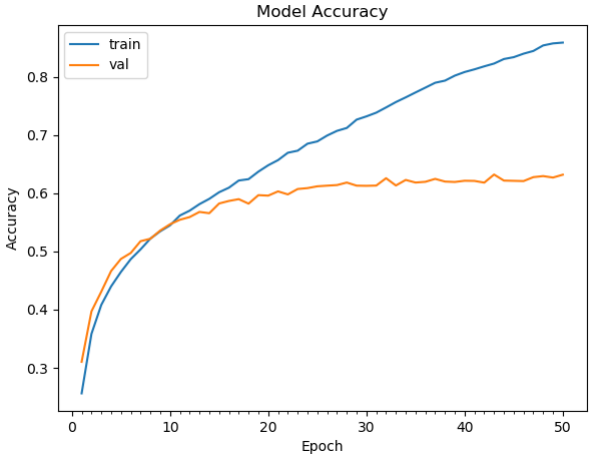
### 3.3 Data augmentation

Adding to the already present model, I have implemented data augmentation by passing arguments to the ImageDataGenerator used to extract the data from the datasets. The augmentation comes in the form of random sample rotation with a rotation range of 0.1 degrees, and random horizontal flipping of the sample.

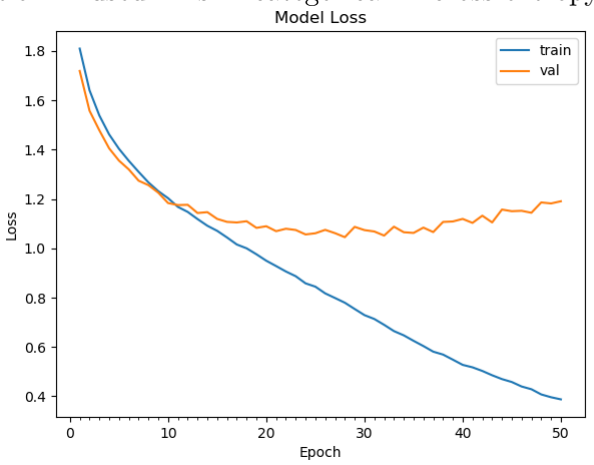
## 4 Results

The model performance can be observed on the following plots.

First, we have accuracy. Being the metric that describes the fraction of predictions that the model got right, accuracy is ideal for estimating the overall efficiency of the model.



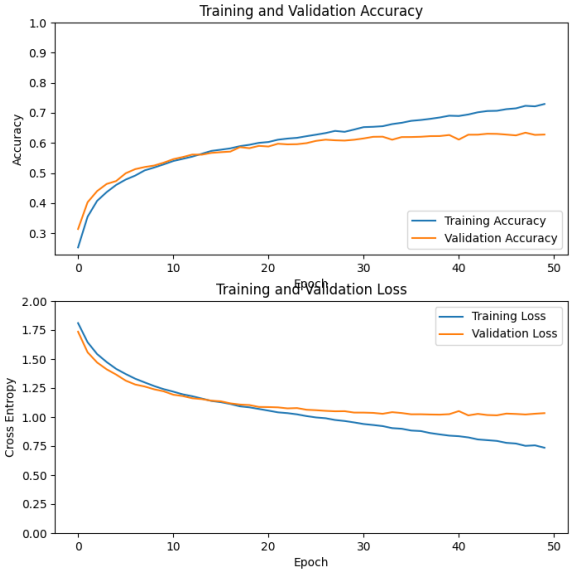

Second, there is loss. As mentioned beforehand, the loss function used is categorical cross-entropy.




This implementation by default detects emotions on all faces in the webcam feed, or from a supplied pre-recorded video. With a simple

4-layer CNN, the test accuracy reached 63.2%, achieved in 50 epochs of training (without data augmentation).

After data augmentation, the accuracy is marginally better, at 63.4%. Also, the progression of the loss seems to be smoother.




# 5 Comparisons with other models

## 5.1 Introduction

I have chosen MobileNetV2 as a model to compare with the previous one. This decision stems from the fact that MobileNetV2 is trained to perform image classification tasks, which comes close to recognizing feelings on a human figure.

## 5.2 Model description

MobileNetV2 is a deep neural network architecture that has multiple layers, including convolutional, pooling, and fully connected layers. The exact number of layers can vary depending on the specific configuration of the network. However, in the original paper "MobileNetV2: Inverted Residuals and Linear Bottlenecks" by Google, the model has around 57 convolutional layers and 5 pooling layers, and two fully connected layers. It also has around 35 million parameters.

## 5.3 Dataset

MobileNetV2 was trained on the ImageNet dataset, which is a large dataset of labeled images that is commonly used for training and evaluating computer vision models. The dataset contains over 14 million images belonging to 1000 different classes, such as animals, vehicles, and objects. Each image is labeled with one of the 1000 class labels and the dataset is split into a training set of about 1.4 million images and a validation set of about 50,000 images.

## 5.4 Transfer learning

In order to perform transfer learning, some layers needed to be readjusted. First, MobileNetV2 was imported without the top layer, since my classification problem is different than the one it was programmed to solve initially. Replacing this layer, I added multiple layers of my own.

1.) RandomFlip, preprocessing layer set to horizontal for data augmentation.

2.) RandomRotation, preprocessing layer set to 0.2 degrees, also for data augmentation.

3.) GlobalAveragePooling2D layer in order to average over the 5x5 spatial locations and convert the features to a single 1280 element vector per sample.

4.) Dropout layer set to 0.2 dropout rate, in order to prevent overfitting.

5.) Dense layer with 7 units and softmax activation function in order to normalize the output of the neural network to a probability distribution over my 7 classes, each one representing a feeling.

The optimizer used is the Adam optimizer, paired with a learning rate of 0.0001 and the categorical cross-entropy loss function.

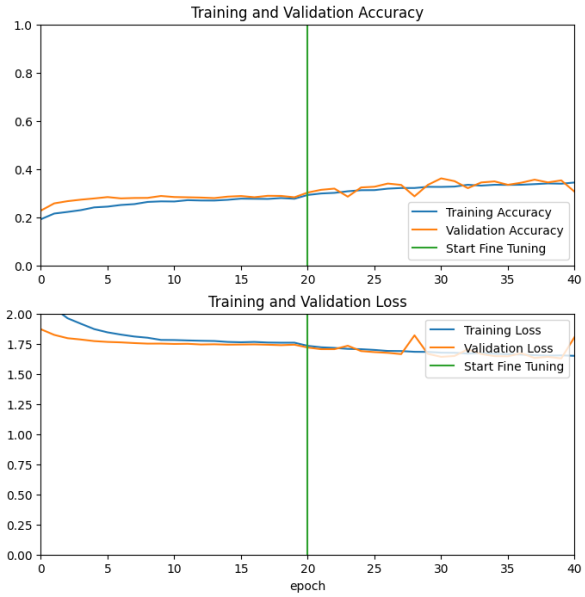

### 5.5 Results before fine tuning

I trained the newly created layers (the rest of the model was frozen, as the transfer learning technique entails) for 100 epochs, but the results are not convincing. The achieved accuracy over 100 epochs of training is 30%, thus contributing to a great loss of 1.7.


### 5.6 Results after fine tuning

The first step of the fine tuning process is unfreezing the base model and setting the bottom layers to be untrainable. Afterwars, for the changes to take effect, recompiling the model is necessary with a much slower learning rate, in order to prevent overfitting. The chosen learning rate for recompilation (0.00001) is 10 times less than the one previously used.

After recompiling, I continue training the model for 20 more epochs. I trained to convergence earlier, and this step increases the accuracy by a few percentages. I am now at approximately 35%.




Green line marks the beginning of fine tuning.

# 6 Conclusions

In the end, the proposed model can somewhat read human emotions by interpreting facial expressions. Of course, the accuracy of the results depends strongly on the quality of the video input it gets, and is also closely tied to how the analyzed person looks based on ethnicity and other genetic physical traits. As far as the "seamless integration between end user and software based on emotions" goes, there are definitely improvements to be made before considering using this solution in any sort of software that flows based on the emotions of the end user.

## 6.1 Extension ideas

A fun idea to consider when thinking about how such a model as this one could be used is to integrate it in video games. Players would be delighted to hear in-game characters switch dialogue lines, or even witness a switch in the entire story line of the game, based on the facial expression of the player at a certain moment.

Another, more serious application could be training a model that looks for fluctuations in emotions that are indicative of mental illnesses. Such a model would still need a specialized official closely monitoring the situation, as diagnosing mental illnesses is a rather ambiguous field, but AI software can be of great assistance.

# 7 References

1.) Github - petercunha - Emotion
2.) Github - akmadan - Emotion Detection CNN
3.) Github - Tomislav-Troha - Facial-Emotional-Recognition-Using-Keras
4.) Real-Time CNN for Emotion and Gender Classification by Octavio Arriaga, Paul G. Ploger and Matias Valdenegro
5.) Github - datamagic2020 - Emotion detection with CNN
6.) Github - atulapra - Emotion detection
7.) Facial Emotion Recognition: State of the Art Performance on FER2013 by Yousif Khaireddin and Zhuofa Chen
8.) Very Deep Convolutional Networks for Large-Scale Image Recognition by Karen Simonyan & Andrew Zisserman
9.) Tensorflow transfer learning tutorial
10.) "MobileNetV2: Inverted Residuals and Linear Bottlenecks by Google"