ID2223 - Scalable Machine Learning and Deep Learning

- Project report -

PAOLO TETA RALFS ZANGIS

teta | zangis @kth.se

January 6, 2022

1 Problem description

It is not easy to make software understand and classify real world phenomena, especially when dealing with images as input. In this project, we aim to develop a neural network that would be capable of classifying human emotions. As a final result, we want to be able to determine if a person is experiencing one of the following emotions: anger, fear, sadness, neutral (no emotion), happiness, surprise or disgust.

2 Tools

We are using several tools to implement and solve the problem.

- Language: Python
- Data processing: Python Pandas
- Dataset: CSV file
- Prediction: TensorFlow
- Coding platform: Google Colab and Jupyter Notebook

3 Dataset

We are going to use a dataset for facial expressions which contains in total 35887 labeled images. The data is available in CSV format, accessible following the provided link: https://www.kaggle.com/deadskull7/fer2013. We intend on using image pixels as features, having emotion as output label.

4 Methodology and algorithm

Our idea was to use a CNN (Convolutional Neural Network) for the classification problem, as its neurons are arranged in 3 dimensions: width, height and depth. The dataset itself is divided into subsets used for training, validation and testing. The final goal is to implement a real-time emotion recognition using camera and augmenting the expression label to the video.

4.1 Neural network structure

Figure 1 shows the structure of the CNN with all the implemented layers. As input layer we use 2D Convolution Layer with 16 filters and we specify as input shape the scaled size of each image, which is 48×48 pixels. Then, we introduce the max pooling layer. Repeating the convolution and pooling three consecutive times with increasing number of filters. After this, we flatten and add a dense layer with 512 neurons. Here, we put the dropout with a rate of 0.2 to avoid overfitting during the training process. In the end, the output layer has only 7 neurons (*labels_num*) and uses the *softmax* activation function. For all the previous layers of the network, we always use the *ELU* activation function to avoid the dead *ReLU* problem due to the vanishing of the gradient. This comes with a longer execution time, but a better accuracy of the result.

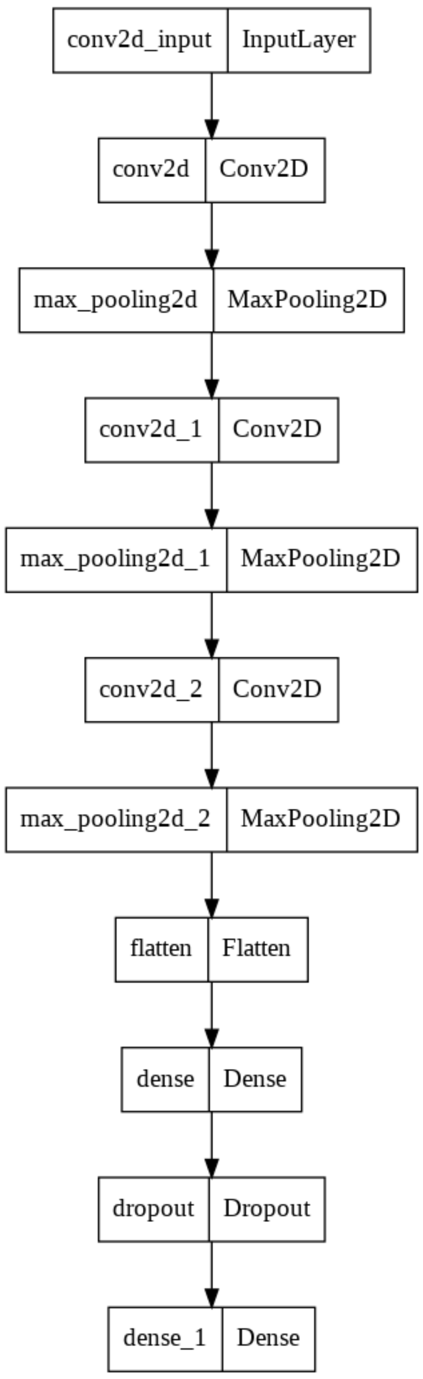
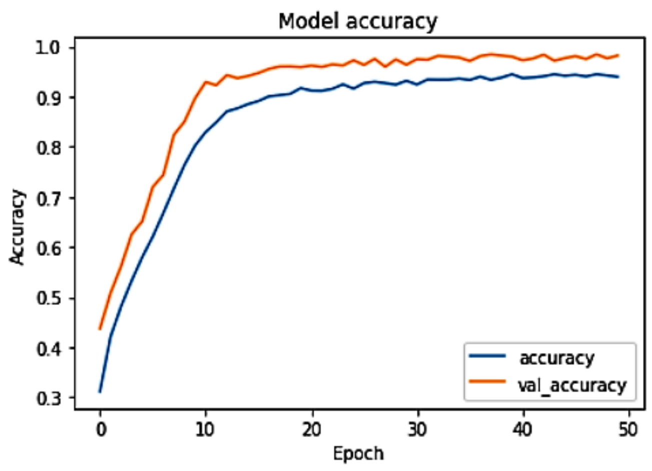

conv2d_input | InputLayer
conv2d | Conv2D
max_pooling2d | MaxPooling2D
conv2d_1 | Conv2D
max_pooling2d_1 | MaxPooling2D
conv2d_2 | Conv2D
max_pooling2d_2 | MaxPooling2D
flatten | Flatten
dense | Dense
dropout | Dropout
dense_1 | Dense

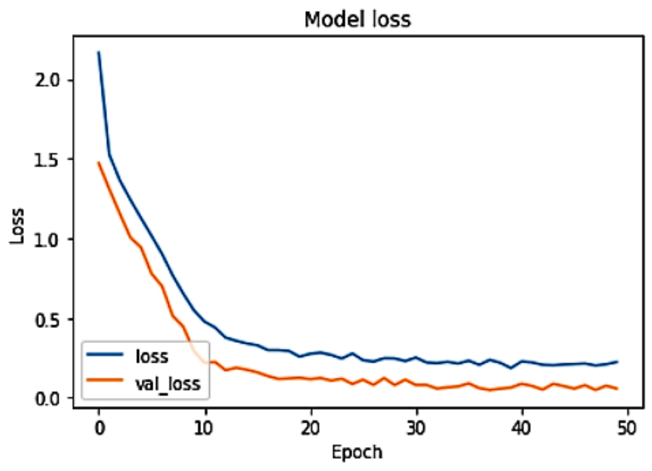
Figure 1: CNN structure

5 Results

After training the model, we use the testing subset of the dataset to test the model and perform the inference process. As shown in Figure 2a, after 50 epochs we obtain a high level of accuracy for both the training and validation subsets, 94% and 98% respectively. This result is confirmed by the values of the loss function in Figure 2b. Looking at the graphs, after the first 10 epochs the model accuracy and loss values are stabilised and decrease less rapidly. Throughout the testing, we tried many configurations changing the number of total epochs, the dropout rate value and the number of neurons for each layer. By using the dropout as regularization method, we were able to avoid overfitting and tuning the other parameters to achieve the best result with high accuracy.



(a) Model accuracy



(b) Model loss

Figure 2: Graphs results

6 Conclusions

Looking at the results, we can conclude that our model achieves a high level of accuracy followed by a very low value of the loss function. Considering the prediction task, we use the last subset of the provided dataset, the unseen data, to test the trained model. Again, achieving a good result, more than 98% for model accuracy and 0.0581 for the loss function. Thus, by using a new set of unseen face images the model is able to select the right emotion with respect to the true label in almost all cases. We also build the confusion matrix (see the Colab notebook) to have a complete overview of the final result in terms of true and predicted emotion label.
For the final goal, which was the real-time emotion recognition capability, we achieved results that can predict emotions expressed by the person to a high degree of accuracy (according to authors beliefs). However, in future to make this product more appealing for real world usage, it would require additional changes: dataset for training could be of a higher image quality and camera should automatically zoom in on face (since model only saw face).

7 How to run the code?

The project repository can be found on GitHub following this link: `https://github.com/bubriks/ID2223/tree/main/Project`.
We have implemented two scripts for this project. The first one, *project-CNN.ipynb* can be executed on Google Colab to speed up the execution for the training process. It will load the dataset and perform some preliminary operations on it. Then, it will create the model with all the layers, train it on 50 epochs and in the end perform the evaluation on the test set. After the training process, we have saved the trained model which can be used for the real-time emotion recognition. The second script, *project_use_model.ipynb* will load the saved model and using the built-in webcam of your computer it will perform the real-time detection of the image and the classification of the related emotion. This second file should be executed locally, such as by using Jupyter Notebook, just remember to check the saved model path in order to be able to load it correctly.