Facial Expression Recognition using Zoning

AML - Advanced Machine Learning Final Project
Instructor: Fabio Galasso

Zain Ullah, Giovanni Giunta,
Mehrzad Jafari Ranjbar, Marco Muscas

Winter semester - 2021

1 Introduction

With this final project we took the opportunity to study in detail the field of "Human Emotion Recognition", and to try implementing a fully functional model on our own. Of remarkable importance for the development of this project were the studies and research papers from Wang et al. Two-pathway attention network for real-time facial expression recognition [1], related to the multi-layer feature data compression (that will be aimed at reducing the amount of parameters in our ensemble model), and also from Zhang et al. Joint face detection and alignment using multitask cascaded convolutional networks [2]. In particular, this last paper is the basis for the MTCNN python library that we used for the data pre-processing part of our project. These and more resources are indicated in the "References".

The ultimate goal of our project is to put into practice the knowledge acquired from all these sources. Along our journey, we aim especially at getting a deeper understanding of Computer Vision techniques, in particular with respect to Facial Expression Recognition Systems for Human Emotion Detection. After becoming familiar with libraries such as MTCNN, we also took the opportunity to push our research even further, by implementing ourselves, from scratch, an algorithm capable of adding an additional key point on pictures with human faces.

This new key point is able to locate with precision an additional facial zone: the forehead. Due to the number of models in use, another challenge we aimed at overcoming is the stacking of the so-called "bottleneck" features, a thematic that will be explored and explained more in detail in the subsequent chapters of this report.

2 Dataset

For our model we decided to use two well known datasets of images: FER2013 and CK+.

The first dataset, FER2013 is from a paper from Goodfellow et al. [3], written after a workshop/challenge in the Facial Expression Recognition domain. It consists of approximately 30'000 images divided in 7 classes. The 7 classes from FER2013: Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral.

The other dataset, CK+ is from another paper from Lucey et al. [4]. It is an extension of an older dataset from the year 2000, namely the CK. The problem domain is also the same as ours, facial expression recognition.

3 Preprocessing

The final task for our model is to be able to correctly predict, up to a certain level of accuracy, what is the emotion expressed by each face in the FER2013 / CK+ dataset, and to achieve our goal we are going to make use of a technique called **Zoning**. At first, we started by implementing the **MTCNN library**, thanks to which we were able to calculate the key points needed to locate the four main facial zones composing the faces in each image, that are:

1. Left Eye (1 key point)
2. Right Eye (1 key point)
3. Nose (1 key point)
4. Mouth (2 key points)

It is important to note that 2 of those key points are used to locate a single facial zone, that is the mouth. For a better understanding of this implementation, it is possible to refer to the image below, generated upon a few sample images from the joint dataset.

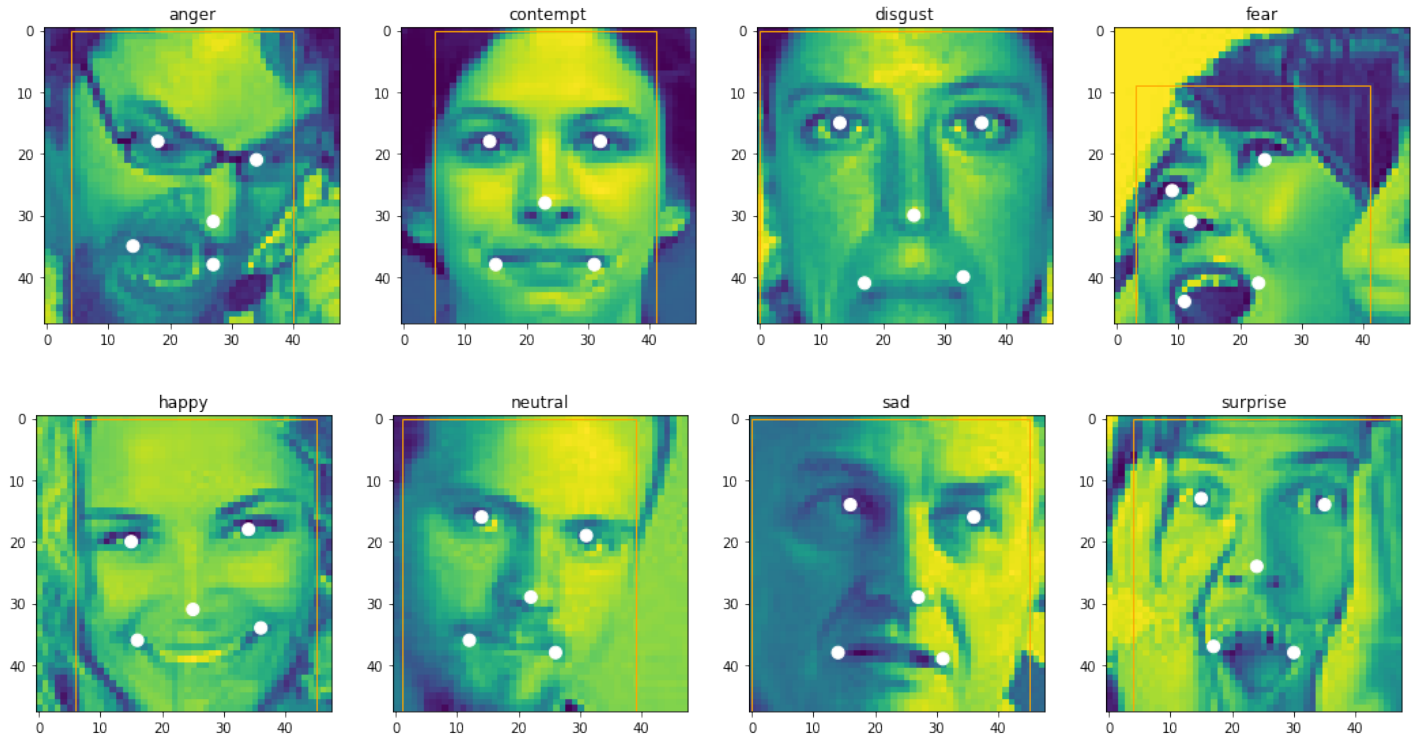


Figure 1: Localized keypoints for various facial expressions

To make our work more significant, we also developed a specific algorithm that, by making use of the position of both the left and the right eye, calculates the position of an additional key point that we use to locate a new facial zone: **the forehead**. With this new key point in place, we then proceeded with identifying all the facial zones that are composing each face (left, eye, right eye, nose, mouth, forehead) by calculating their own specific areas around each point. In the image below we provide the reader with an example.

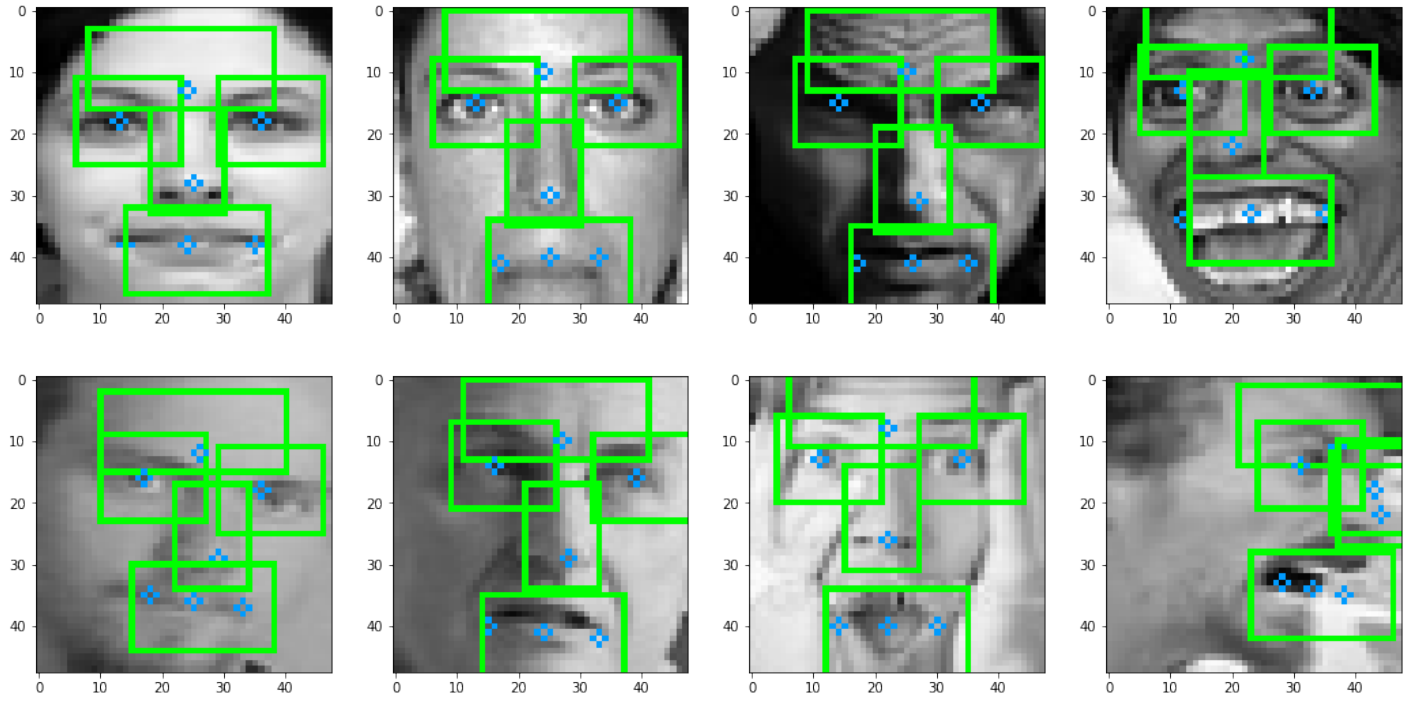


Figure 2: Localized bounding boxes and key points of features

The code we developed to accomplish this task has shown particularly satisfying results on a vast number of images, beyond the ones represented in the figure above. For more information, please refer to the attached file `AML-Preprocessing.ipynb`.

3.1 Feature Extraction

Selecting the areas belonging to each facial zone is going to be crucial for us, as each zone will contribute to correctly predicting the emotion represented in each image. For this reason, we cropped each zone out of the original image, as well as the entire face, that is kept in its entirety. Each zone is then considered as an "extracted feature", and is going to participate in the learning process: for this reason, each zone (feature) is saved as a file into its own dataset.

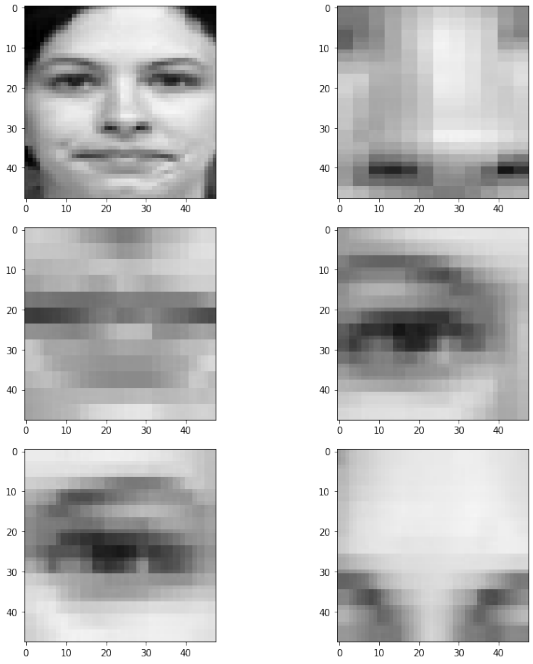


Figure 3: Individual features, extracted and separated

We now have 6 separate datasets, one for each facial feature and an additional for the whole face:

1. Left eye
2. Right eye
3. Nose
4. Mouth
5. Forehead
6. Cropped face

As a side note, the split we used for the final version of our model was **80%** for training and **20%** for testing.

4 Bottleneck feature stacking

The bottleneck feature stacking is a concept we got from the article by Wang et al. [1] and that we plan to implement so to improve the overall performance of our classifier.

To explain what it consists of, we would like to recap a bit our situation.
So far, we have split our data into 6 different datasets, one for each facial feature. From each image of a feature (though ultimately extracted from the same image of a face), we have trained a separate classifier. This classifier is VGG-16, fine-tuned to suit our needs.

Our needs were the extraction of the *feature maps* from the last pooling layer - these are the **bottleneck** features.

Eventually we are left with 6 separate feature maps, each of size $K \times K \times 512$, that can be directly stacked to get a $K \times K \times 3072$ vector. For a better visualization of this process, here is an image representing it:

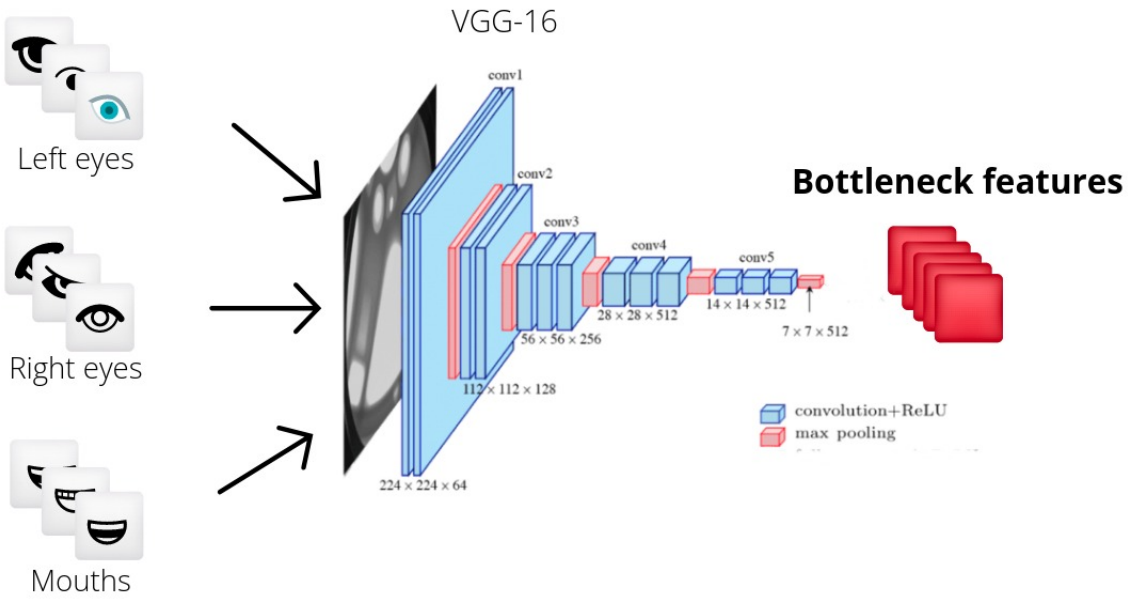


Figure 4: Bottleneck feature stacking process

5 Classification

For the second part of our model, as we have mentioned in Section 4, we now have a dataset made of the stacked feature maps (bottleneck features). After this process, we have set up another classifier that is ready to classify such bottleneck features.

It consists of a series of fully connected layers that ends with a final `Dense` layer of **7 classes**, that is designed to produce a vector containing the probabilities for the input to belong to a certain class.

Here is another image illustrating this part:

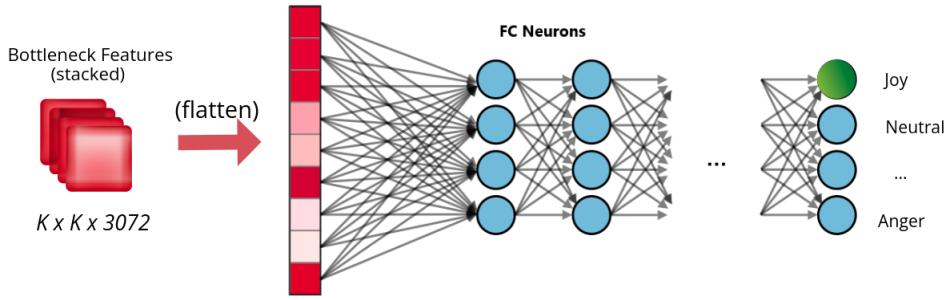


Figure 5: Fully connected neural network for the emotions

As a side note, in the hidden layers part of the network we also added different dropout layers and tried different activation functions, such as the **Leaky-ReLU**.

6 Experimental Results

6.1 Data Augmentation

Since the CK+ dataset is quite reduced in size, we decided that a bit of data augmentation would not hurt. Of course, we've only done that to the training part of our dataset, the validation was left intact.

The augmentations we applied increased the number of samples in the training split, while training thanks to the `ImageDataGenerator`. More specifically, we applied:

1. Image rotation of ± 20 degrees.
2. Rescaling (required for normalization), factor of $1/255$.
3. Shear range (distorting the image along an axis)
4. Cropping
5. Horizontal flipping

6.2 Initial Results

The first model we wanted to implement was a bit simpler. Instead of using 6 separate VGG-16, we tried fine-tuning the last 3 layers of a single VGG-16, then we performed the bottleneck feature stacking (from Section 4).

We initially achieved an accuracy of around 60%. Quite disappointing, but expected. The model was not nearly ready enough.

With our second attempt we started developing the model we said we would implement, in other words, having the 6 separate "fine-tuned" VGG-16 which provided us the feature maps, and another neural network (this time trained from scratch) consisting of fully connected layers (with dropout enabled in between them). Here are the results we got from the training:

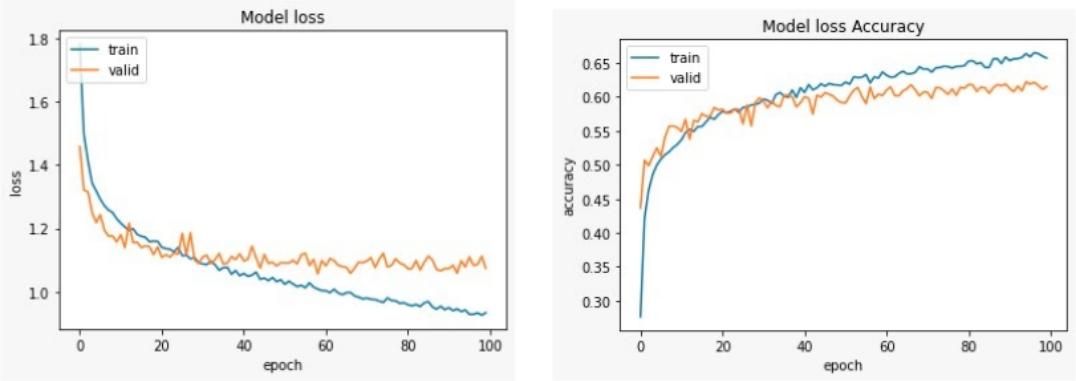


Figure 6: Accuracy-Loss curves for training and validation, CK+ & Fer2013 datasets

Such an outcome is strange indeed. We only have around 65% of validation accuracy, and as a side note, only 25% of validation accuracy when training each VGG-16 (which was concatenated to a few `Dense` layers to obtain our 7 classes probabilities).

Therefore, we investigated a bit more, this time considering each dataset separately.

6.3 Our model using the Fer2013 dataset

The initial results we have are a bit disorienting. We kept the 6 VGG-16s for our feature maps (fine-tuned again, of course) and then the other neural network made of `Dense` and `Dropout` layers. Here are the results:

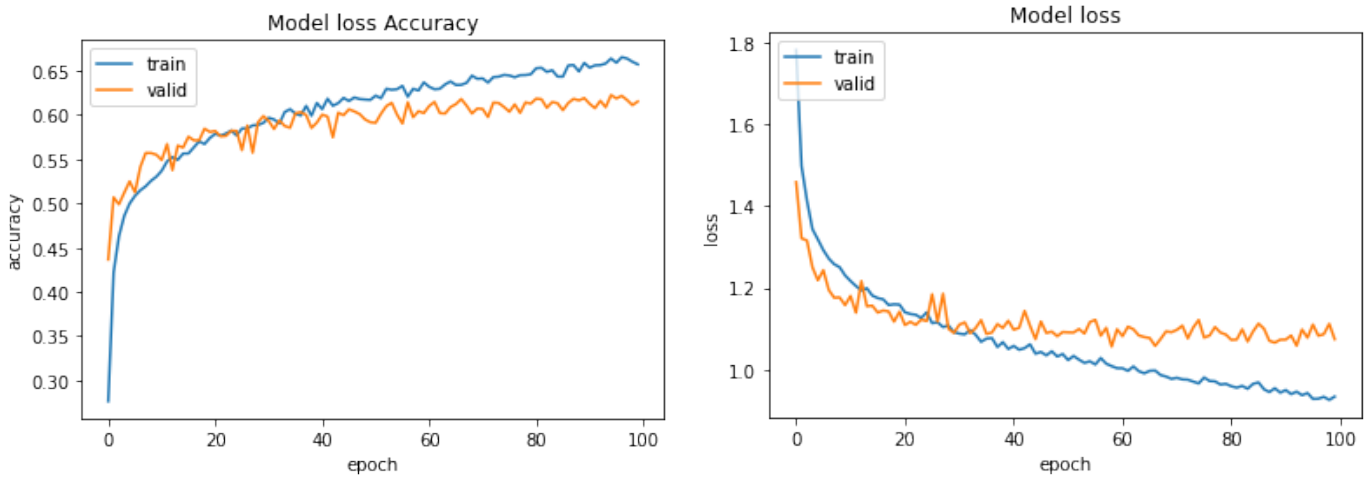


Figure 7: Accuracy-Loss curves for training and validation, CK+ only

It seems that, though more comprehensive, the FER2013 does not bring the results we expected to get. Surely not those we got from the next section.

6.4 Model using the CK+ dataset

In the given figure below the results we have are the loss and accuracy for the dataset **CK+** using all features bottleneck combined like **Eyes, Nose, Left Eye, Right Eyes, Mouth, Forehead and full Cropped face**, using the above mentioned model. To recap, each VGG-16 has been fine-tuned and the "fully connected part", trained from scratch.

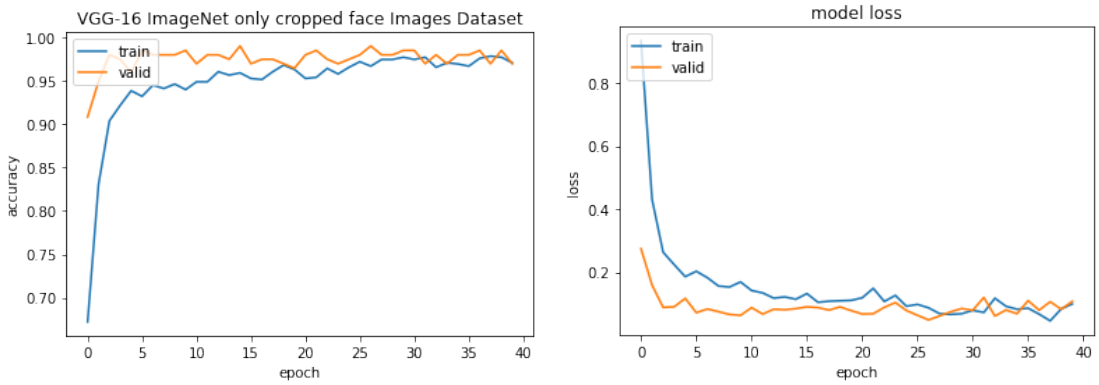


Figure 8: Accuracy-Loss curves for training and validation, CK+ only

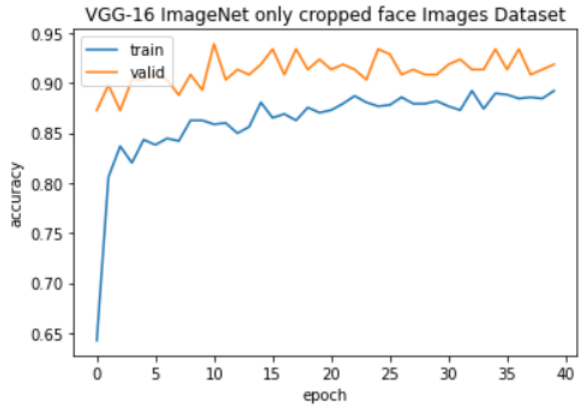
After **40 epochs** we got **0.08** loss and **98.47%** validation accuracy. Which is great, and also surprising! The convergence of the loss seems to be very fast, while the validation accuracy is consistent with the training accuracy (we overfitting like we were seeing before).

6.5 Comparison with another baseline

Another test we wanted to perform was more comparative: did we actually achieve something better than the baseline by using zoning, instead of just throw images of faces to a CNN?

Let us consider a generic VGG-16 as our baseline for a chosen model.

We built a model to do this comparison. Here we provide the plots for training/validation loss curves, and of course for the loss curves that does not require any preprocessing and just takes images of faces as input:




```
[INFO] accuracy: 91.84%
[INFO] Loss: 0.19902536273002625
```

Figure 9: Metrics of the VGG-16 model using cropped face images

As indicated by the caption of Figure **??** we also used VGG-16 (fine-tuned) as our starting point. Interestingly enough, the results seem quite acceptable, getting us a **91.8%** on the validation set.
But if we truly want to go with the best we have, we should consider the model that uses zoning, that got us that nice **98.4% accuracy** on the validation set.

Keep in mind, these results were achieved when only considering the CK+ dataset.

7 Conclusion

As stated before, the main idea for this project was trying to build a better model than what we would usually be able to build easily using libraries such as *Keras*.

By taking inspiration from the work of Wang et al. [1] and more, we can now clearly state that the application of zoning in the feature extraction process does improve the final performance of our model. The full datasets would need to be explored a bit more in depth as the discrepancy in the accuracy still brings some doubts.

To wrap it up, the results we found were really promising, when considering the CK+ only, a 98.4% accuracy!

References

[1] Lining Wang, Zheng He, Bin Meng, Kai Liu, Qingyu Dou, and Xiaomin Yang. Two-pathway attention network for real-time facial expression recognition. *J. Real Time Image Process.*, 18(4):1173–1182, 2021. doi:10.1007/s11554-021-01123-w.

[2] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multi-task cascaded convolutional networks. *CoRR*, abs/1604.02878, 2016. URL: http://arxiv.org/abs/1604.02878, arXiv:1604.02878.

[3] Ian J. Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron C. Courville, Mehdi Mirza, Benjamin Hamner, William Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, Yingbo Zhou, Chetan Ramaiah, Fangxiang Feng, Ruifan Li, Xiaojie Wang, Dimitris Athanasakis, John Shawe-Taylor, Maxim Milakov, John Park, Radu Tudor Ionescu, Marius Popescu, Cristian Grozea, James Bergstra, Jingjing Xie, Lukasz Romaszko, Bing Xu, Zhang Chuang, and Yoshua Bengio. Challenges in representation learning: A report on three machine learning contests. *Neural Networks*, 64:59–63, 2015. doi:10.1016/j.neunet.2014.09.005.

[4] Patrick Lucey, Jeffrey F. Cohn, Takeo Kanade, Jason M. Saragih, Zara Ambadar, and Iain A. Matthews. The extended cohn-kanade dataset (CK+): A complete dataset for action unit and emotion-specified expression. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2010, San Francisco, CA, USA, 13-18 June, 2010*, pages 94–101. IEEE Computer Society, 2010. doi:10.1109/CVPRW.2010.5543262.