Neural Networks Project

A.Y. 2020/21

Professors:
Aurelio Uncini
Simone Scardapane

Students:
Jacopo Rossi 1801667
Giacomo Venneri 1810169

Contents

1 Introduction 2
1.1 Paper . . . 2

2 Dataset 3
2.1 Preprocessing . . . 3
2.2 Data generation . . . 3
2.2.1 Data augmentation . . . 4

3 DACL Model 5
3.1 Model overview . . . 5
3.2 Implementation details . . . 6
3.2.1 Workflow . . . 6
3.2.2 Model details . . . 6
3.2.3 Model architecture . . . 7
3.2.4 Training and validating the model . . . 7

4 Experiments and Results 9
4.1 Experiment 0 . . . 9
4.2 Model variations . . . 10
4.2.1 Experiment 1 . . . 10
4.2.2 Experiment 2 . . . 10
4.2.3 Experiment 3 . . . 10
4.2.4 Experiment 4 . . . 11
4.3 Confusion Matrix and Avarage accuracy . . . 11
4.4 Recognition results . . . 12
4.4.1 Softmax model . . . 12
4.4.2 Center Loss Model . . . 12
4.5 Attention weights visualization . . . 13
4.6 Classification samples . . . 14

5 Conclusions 16

Chapter 1

Introduction

Facial expression recognition (FER) is an essential ability for good interpersonal relations. Today, it is one of the most important subjects of study in the field of human development, psychology and computer science. Indeed, the recognition of emotions has a main role in the interpretation of social behaviour, of emotional intelligence and empathy.
There are also a lot of applications of FER in business, automotive, media advertising, video games, education and human and computer interaction; recently the enormous diffusion of cameras and biometric sensors has played a relevant role in the development of this type of technology.

Analysing more in details the computer vision area, it's since decades that researchers are developing methods for automated emotion recognition and today there are different techniques that can be exploited. It is a very interesting and challenging task, since sometimes also people have problems in recognizing emotions and there is great variability in the categorization.

1.1 Paper

In this document, we are going to describe the implementation of the paper *"Facial Expression Recognition in the Wild via Deep Attentive Center Loss"* [1] based on the work made by Amir Hossein Farzaneh, Xiaojun Qi.
Making an overview about this, it's possible to observe that using Convolutional Neural Networks (CNNs) is a non-trivial task, since there are relevant intra-class variations and inter-class similarities. Approaches based on the use of center loss and softmax loss are very diffuse in this field, however they might include some irrelevant features that are going to affect the generalization of the algorithm. So In the paper, it is proposed a *Deep Attentive Center Loss (DACL)* method that allows estimating the attention weights in such a way to take advantage only of the most important feature to achieve intra-class compactness and inter-class separation.

Chapter 2

Dataset

The original model described in the paper was trained and tested on the **RAF-DB** and **AffectNet** dataset. However, these datasets, like the majority used in this field, are suitable to the use only to researchers and professors, for this reason, we try to find a different dataset that makes possible to test face expression recognition.
Indeed, in our implementation, the *DACL* model was trained and tested on the **FER2013** dataset. **FER2013** contains 35685 facial images of 48x48 pixel grayscale. Images are associated with a number that is referred to the category they represent. For our experiments, the training set consists of 28709 examples and the test set is made of 3589 images. There is also the validation set that consist of another 3589 examples and that is used to test all the experiments. The categorized expressions identified are: happy, sad, surprise, anger, fear, disgust and neutrals.

2.1 Preprocessing

The dataset is organised in folders, there are 3 different directories that are used respectively to identify the training set, the test set and the validation set. Every one of these folders is divided into 7 subfolders, where each of them is used to store the images of a specific facial expression category. As we can observe from the dimension of the folders in the dataset, FER2013 is unbalanced. For example, analyzing the training set folder it's possible to notice that there are only 436 images of disgusted people, while there are about 7215 images that can be used to learn information about happy people.

2.2 Data generation

The training, testing and validation sets assume a different role in the implementation of the model. The training set is very important during the learning phase of the model, while the testing and the validation set are used to estimate if the model is learning correctly the information. The techniques that we have used to load the images are based on the use of the functionality "ImageFolder" which is a generic data loader that allows us to load the images divided by category.

2.2.1 Data augmentation

In this phase, in order to make the model more robust and to avoid the overfitting on the data, we applied some augmentation techniques on the images of testing and training dataset.
The most relevant are:

- *Random Horizontal Flip*: Horizontally flip the given image randomly with a given probability.
- *Random Five Crop*: The image is cropped into four corners and central crop, after that it is used only one of the cropped images, which is chosen randomly.
- *Center Crop*: Crops the given image at the center. If the image size is smaller than the output size along any edge, the image is padded with 0 and then center-cropped.
- We shuffle our data during each epoch, to avoid being "stuck" in batches that are not representative of the overall dataset.

Each of the precedent techniques related to the cropped of the image is adopted choosing a cropped size of 200x200. Operations used during the data augmentation are offered by the *PyTorch* framework.

Chapter 3

DACL Model

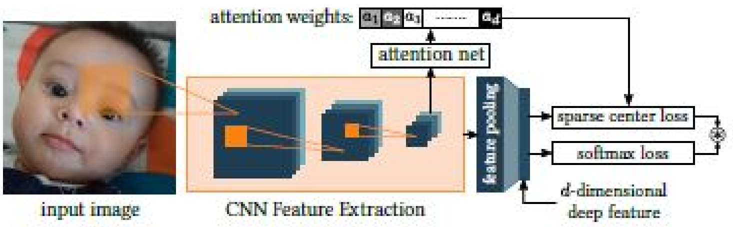
3.1 Model overview

To perform the FER task, our *DACL* model takes as input an image and gives as output one of the seven classes from which the image can belong. The structure of the model can be divided into two sections: the CNN and the *Attention net.*

Is well known that Convolutional Neural Network (CNN) methods are frequently used in the field of Face Expression Recognition because they have outperformed other techniques in extracting deep features representation of the images. So we start creating the model following the paper's instructions and building a CNN to extract spatial features of the images, finally these features will be converted into a single deep feature vector using a pooling layer. In practice, a softmax loss with the contribution of the *Sparse Center Loss* estimates a probability distribution over all classes in the final stage.

The widely used softmax loss is insufficient in extracting discriminative features from the images, which is a fundamental objective in FER tasks. For this reason, the paper's authors propose a Deep Metric Learning (*DML*) method named Deep Attentive Center Loss *(DACL).*

The second section of the model represents the implementation of this method. The last layer of the CNN is given to the *Attention net*, in order to produce some weights that will focus the objective function on the most relevant information. The objective function is the *Sparse Center Loss*, that is a variation of the *Center Loss* (usually used with softmax), but weighted using the output of the *Attention net.*
The *Attention net* is composed of two parts. At the beginning, we have the *Context Encoder Unit (CE-Unit)* which takes the spatial feature map from the CNN as input and generates a latent representation, that contains all the important information needed to represent our images. Then we have the *Multi-head Binary Classification* module that takes the latent representation and estimates the attention weights.
In the following section, we will see the implementation details of the *DACL* method.

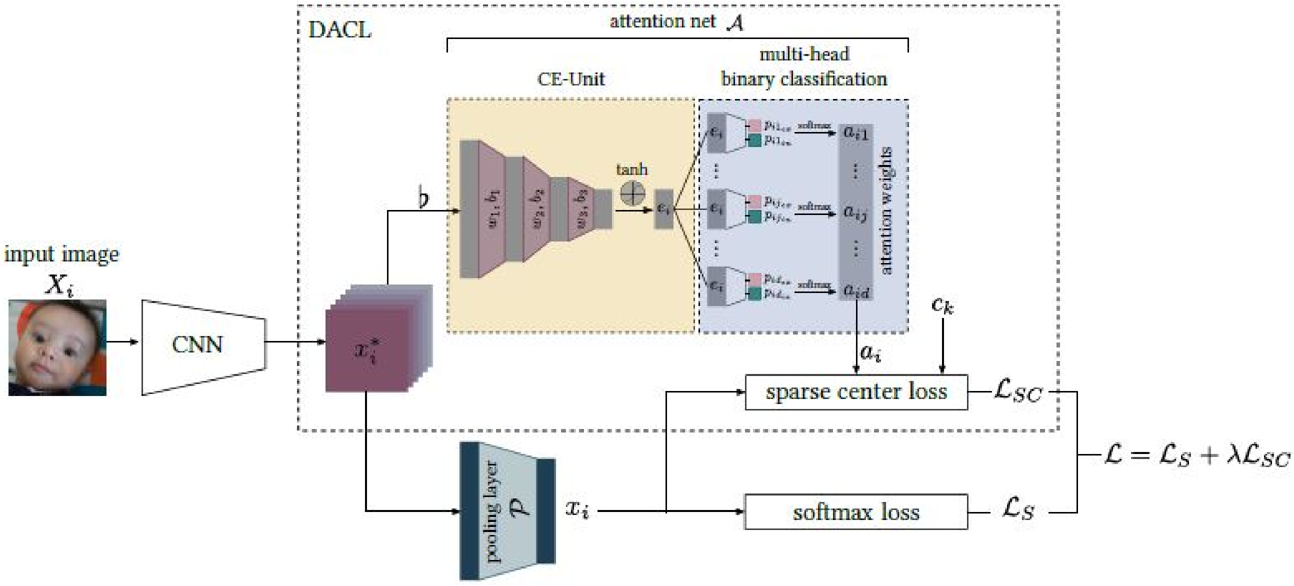

attention weights: α_1 α_2 α_3 ······ α_d
attention net
feature pooling
sparse center loss
softmax loss
d-dimensional deep feature
input image
CNN Feature Extraction

3.2 Implementation details

3.2.1 Workflow

The code, that we use to implement and test the model, is written in *Pytorch*, an open-source machine learning framework, that offered us all the needed tools to develop the different modules of our project.
We have written code together, using a video conferencing app and sharing a *Google Colab* document on a *GitHub* repository. *Colab* is a platform that allows you to execute code directly in Cloud, using the computing power provided by Google (unfortunately for a limited time every day). We used this alternative because neither of us have the necessary hardware accelerators to train the models.
In implementing the code we followed the paper's instructions, changing some parameters to exploit the characteristics of our datasets.

3.2.2 Model details



The model backbone is based on the implementation of the standard Convolutional Neural Network, that is a ResNet18. As is typical in the ResNet architecture, we performed first an initial convolution and a max-pooling using 7×7 and 3×3 kernel sizes respectively. Then we added the convolutional layers and following the paper's instructions we matched the last convolutional feature map x_i^* with a size of 512×7×7, adding at the end a pooling layer that is the standard 2D average pooling.
After the CNN, we built the *Attention Net* starting from the *CE-Unit*. It is designed by stacking three fully-connected layers with 3,584, 512, and 64 out-features dimensions respectively. Hence, the output of the *CE-Unit*, the latent feature vector $\mathrm{e_i}$, is 64-dimensional. These layers are interjected with batch normalization and rectified linear units to capture non-linear relationships between layers. At the end, the authors put a tangent function to preserves both positive and negative activation values. Then as described in the paper, we attached a fully connected layer to implement the attention-head structure, that takes as dimension of the input the size of the last *CE-Unit*, and as output the size of the last feature map multiplied by two, having each head two outputs. Finally, we applied a softmax function to the attention-head block.

Now the attention heads can be used to calculate the *Sparse Center Loss* as follow:

$$L_{\mathrm{SC}} = \frac{1}{2m} \sum_{i=1}^{m} \sum_{j=1}^{d} a_{\mathrm{ij}} \odot \parallel x_{\mathrm{ij}} - c_{\mathrm{y_i j}} \parallel_2^2$$

3.2.3 Model architecture

Image shape	200×200×3

Layer	Output size	Parameters
Conv2d-1	[-1, 64, 100, 100]	9,408
BatchNorm2d-2	[-1, 64, 100, 100]	128
ReLU-3	[-1, 64, 100, 100]	0
MaxPool2d-4	[-1, 64, 50, 50]	0
ResNet18	[-1, 512, 7, 7]	11,166,976
Linear-67	[-1, 3584]	89,918,976
BatchNorm1d-68	[-1, 3584]	0
ReLU-69	[-1, 3584]	7,168
Linear-70	[-1, 512]	1,835,520
BatchNorm1d-71	[-1, 512]	1,024
ReLU-72	[-1, 512]	0
Linear-73	[-1, 64]	32,832
BatchNorm1d-74	[-1, 64]	128
Tanh-75	[-1, 64]	0
Linear-76	[-1, 1024]	66,560
AdaptiveAvgPool2d-77	[-1, 512, 1, 1]	0
Linear-78	[-1, 7]	3,591

Table 3.1: Architecture of the model

3.2.4 Training and validating the model

Even in these tasks, as in the rest of the paper, we tried to be coherent as much as possible with the indication of the authors.

Once that we have created the model, during training we have used as optimizer the standard the *Stochastic Gradient Descent* (*SGD*), with a momentum of 0.9 and a weight decay of 5×10^{-4}, using a learning rate of 0.01.

At the end of each training step, during each epoch, we performed an evaluation on a smaller and different dataset of 3589 images. The evaluation has different benefits, such as the possibility to understand early how much the model is learning, looking at the difference between the trend of the train accuracy/loss and the eval accuracy/loss. In addition, we have the possibility to save the learning phase of the model, in order to have some checkpoints in case of problems with *Colab*. Moreover, using an evaluation after each training, we collect useful data, that are exploited to plot the model's performances at the end of the training. However, this decreases the images given to the validation phase, used to built the Confusion Matrix. Even in this case the validation set has 3589 images.

During this task, the accuracy in each batch is calculated with respect to the number of correct predictions, obtained considering the ground zero values of the batch, all dived by the number of samples in that batch, so that we obtained the average results.

During each batch, we present three different losses: softmax loss, *Sparse Center Loss* and total loss; still considering their average value. The softmax loss is obtained using the *CrossEntropyLoss* function implemented by the *Torch* library. While the *Sparse Center Loss*, is obtained as described in the previous section. Finally, the Total Loss, used to evaluate the model, is obtained combining the softmax and the *Sparse Cente Loss* values as follow:

$$L_{\text{tot}} = L_{\text{soft}} + \lambda \cdot L_{\text{SC}}$$

where λ, is a parameter set empirically by the paper to 0.01, to tune the influence of the *Attention net* on the model.

Chapter 4

Experiments and Results

In this chapter, we describe the experiments that we have performed using the implementation of the *DACL* model and we analyze the result obtained.

4.1 Experiment 0

Initially, we tried to perform a training with the **FER2013** dataset using the same parameter used by the paper for the **RAF-DB** and **Affect-Net** database. The images of the training set were modelled on-the-fly by extracting random crops of 224x224 pixels from the original input images (resized to 256x256 pixels) and the model was trained with 60 epochs, 128 batch size and an initial learning rate of 0.01 decayed by a factor of 10 every 20 epochs.

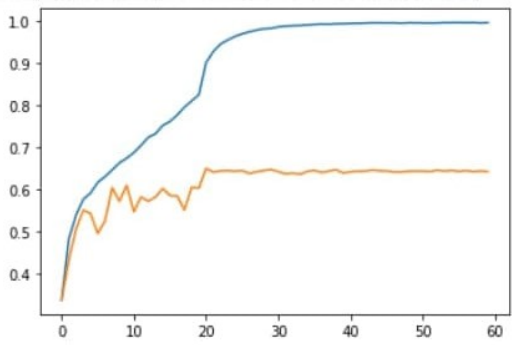


Figure 4.1: First experiment accuracy, 60 epochs.

The model was trained and tested without any problem. However, as it is possible to see from the Figure 4.1, there was big overfitting on the accuracy of the training set with respect to the accuracy of the test set and moreover the model start to assuming an asymptotic trend around epoch 20.
For this reason, we decided to train the model for 30 epochs, hypothesizing to have similar behaviour in the successive experiments.

4.2 Model variations

At this point we decided to make some variations in the implementation of the model, having the scope to obtain a better result.

4.2.1 Experiment 1

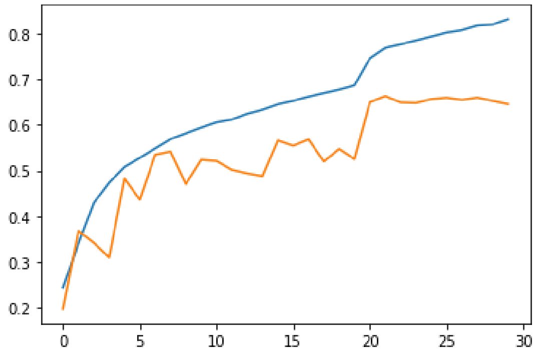


Figure 4.2: batch 128, crop size 200

In this experiment, we have achieved a better result. As first attempt, we have reduced the crop size of the images from 224x224 to 200x200 to reduce the overfitting.
We still continued to use the random five crop in the training set and the center crop in the test set, still using a batch of 128.

4.2.2 Experiment 2

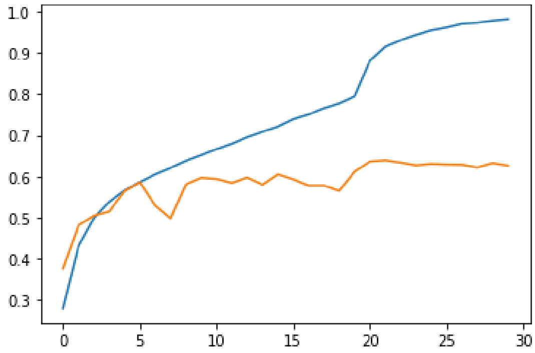


Figure 4.3: batch 64, crop size 224

The second experiment interested the batch size, that has been reduced to 64. In fact it's possible, in some cases, that having a bigger batch size could lead to a too poor generalization and so to a decreasing of the performance's accuracy.
Thee crop of the images still remained 224x224.
We can see that there is a bit reduction of overfitting respect to the Experiment 0, but the previous attempt is still better.

4.2.3 Experiment 3

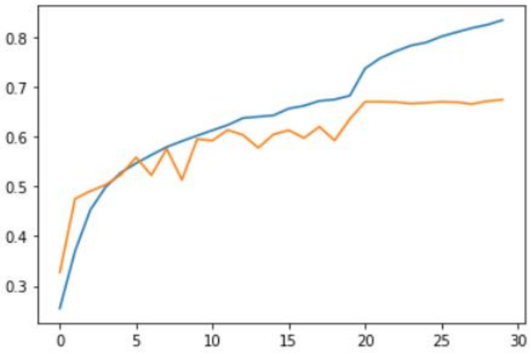


Figure 4.4: batch 64, crop size 200

With this configuration we achieved the best result, using both a smaller batch (64) and crop size of the images (200x200). How we can see we have poor overfitting and a decent accuracy of 0.674 on the evaluation set.
As suggested in the paper we also decided to train the model with a learning rate of 0,01 decayed by a factor of 5 every five epochs. This is a different configuration, since in the precedent models we adopted a learning rate decay of 0,01 decayed of a factor of 10 every 20. However we didn't achive a reasonable better result.

4.2.4 Experiment 4

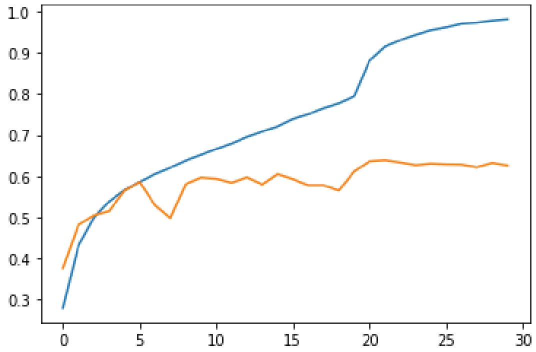
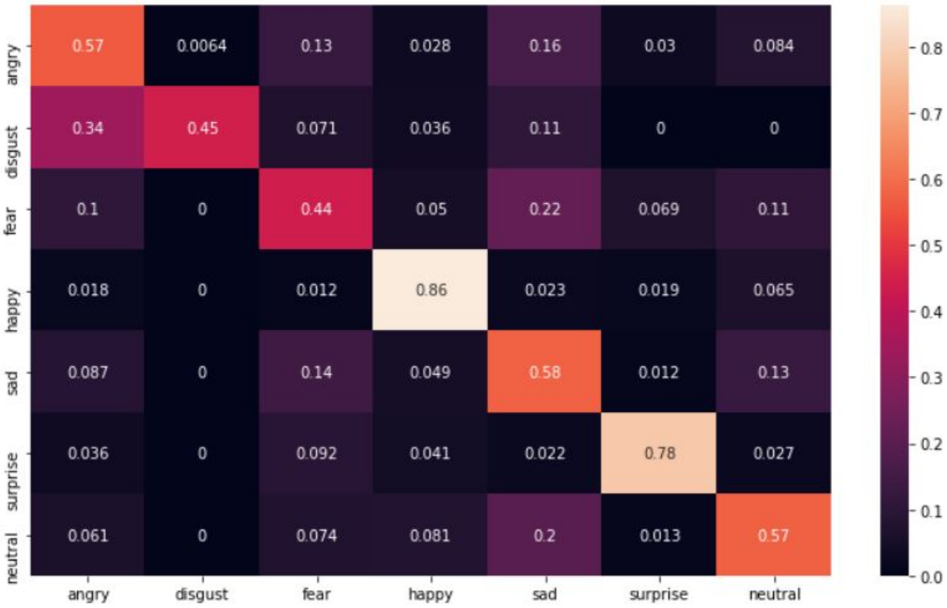


Figure 4.5: batch 64, crop size 200, pretrained with msceleb.

In this implementation, we exploited a pre-trained model. Indeed, since FER's domain is similar to the Face Recognition task, we adopted a pretrained ResNet18 on the **MS-CELEB-1M** dataset composed of over 10 million images of face related to about 100,000 people. The batch size used is 64 and the other parameters of the model are the same used in the precedent experiments. We can notice that in this case, the model learned everything faster, but after the first part that has a relevant increase, we have an asymptotic and stable behaviour.

4.3 Confusion Matrix and Avarage accuracy

Observing the model implementation we have to underline that, unfortunately, the dataset used is imbalanced so, to calculate the value of the accuracy reached by the model we report the average accuracy, which is the mean of the diagonal values in the confusion matrix. This is the same behaviour adopted by paper's authors, since even their datasets are unbalanced too, a common condition in FER.
The Confusion Matrix was built using the validation set that has been offered by the dataset. So, after the training of the model, we have validated it and we have plotted the confusion matrix. Taking into consideration all the variations analysed in the precedent steps, we have discovered that the Experiment 3, presents the best performance.
Its confusion matrix is shown below and its average accuracy is 0.61.



	angry	disgust	fear	happy	sad	surprise	neutral
angry	0.57	0.0064	0.13	0.028	0.16	0.03	0.084
disgust	0.34	0.45	0.071	0.036	0.11	0	0
fear	0.1	0	0.44	0.05	0.22	0.069	0.11
happy	0.018	0	0.012	0.86	0.023	0.019	0.065
sad	0.087	0	0.14	0.049	0.58	0.012	0.13
surprise	0.036	0	0.092	0.041	0.022	0.78	0.027
neutral	0.061	0	0.074	0.081	0.2	0.013	0.57

Figure 4.6: Confusion Matrix, Experiment 3.

4.4 Recognition results

After choosing a model as a reference, we reproduced other experiments described in the paper. We studied the impact of the *Attention Net* on the model, firstly by measuring its performance by training the model with only softmax as loss function and then using the classic center loss function, instead of our *Sparse Center Loss*.
We tested these variants on the Experiment 3 model configuration.

4.4.1 Softmax model

In this implementation of the model, we considered only the softmax loss without giving importance to the sparse center loss and the other part of the model. Since in our model the loss can be computed with the formula: $L_{\text{tot}} = L_{\text{soft}} + \lambda \cdot L_{\text{SC}}$. It's possible to consider the softmax loss a special case of the formula where the λ value is set to 0.
How we can see its average accuracy of 0.59, is worst with respect to the original model.

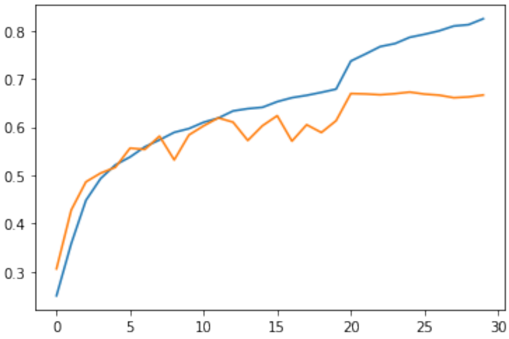


Figure 4.7: softmax accuracy

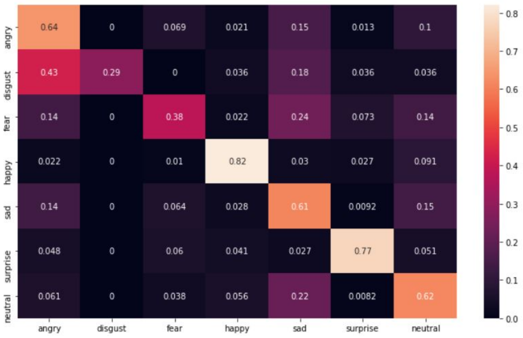


Figure 4.8: softmax confusion matrix

4.4.2 Center Loss Model

The center loss model learns a center for deep features of each class and penalizes the distances between the deep features and their corresponding class centers [2]. Our DACL model is inspired by the center loss, but it is reformulated thanks to the introduction of the attention-weights. Indeed generally we can reduce the *Sparse Center Loss* function to a center loss function simply by having all the attention weight $a_{\text{i}1} = ... = a_{\text{id}}$ with the same value. Center Loss is jointly supervised with softmax loss to compose the final loss.
Even in this case with an average accuracy of 0.60, this configuration does not out performed the *DACL* method.

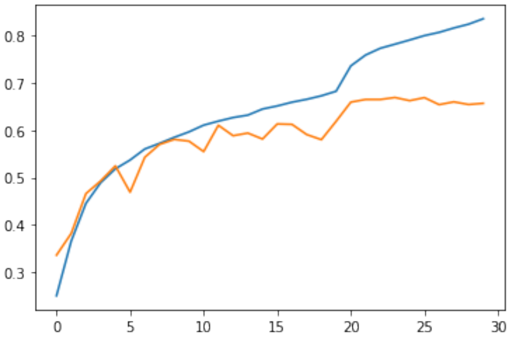


Figure 4.9: Accuracy center loss

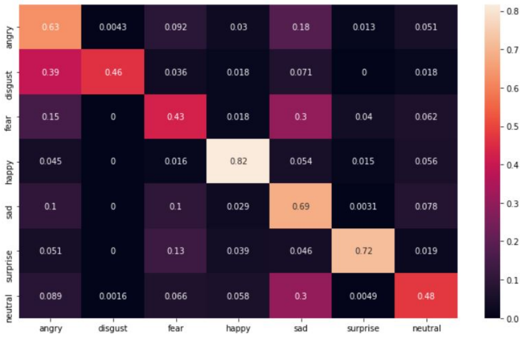


Figure 4.10: Center loss Confusion Matrix

4.5 Attention weights visualization

To show the correctness of our work, as done in the paper, we show the inter-class similarities of the attention weights. To visualize this result, we have plotted the 512-dimensional attention weights for images that belong to the same class, showing how these weights present some similar patterns.
Generally, it's possible to compare attention-weights of images of different classes, to emphasize the learnt inter-separation.

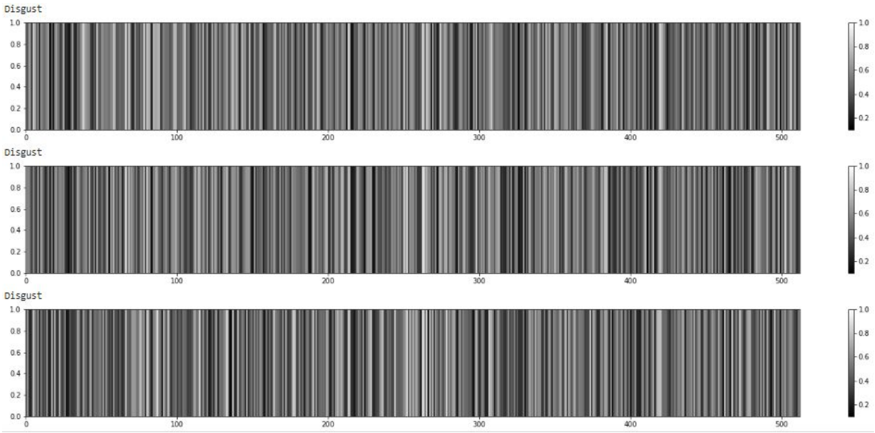


Figure 4.11: Disgust attention weights

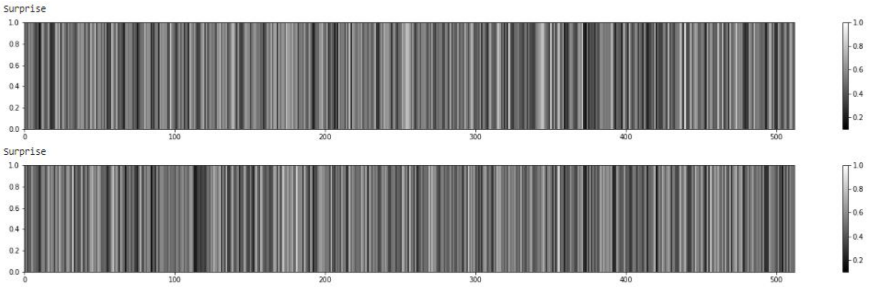


Figure 4.12: Surprise attention weights

Observing the images, we can notice that all the three attention-weights associated with the "disgust" class present some similar patterns such as the white line after the 400^{th} weight or the other white zone between the 200^{th} and the 300^{th} position. In the case of the two "surprise" attention weights, we can see in both the images a similar zone before the 400^{th} positions and another similar behaviour at the beginning.

4.6 Classification samples

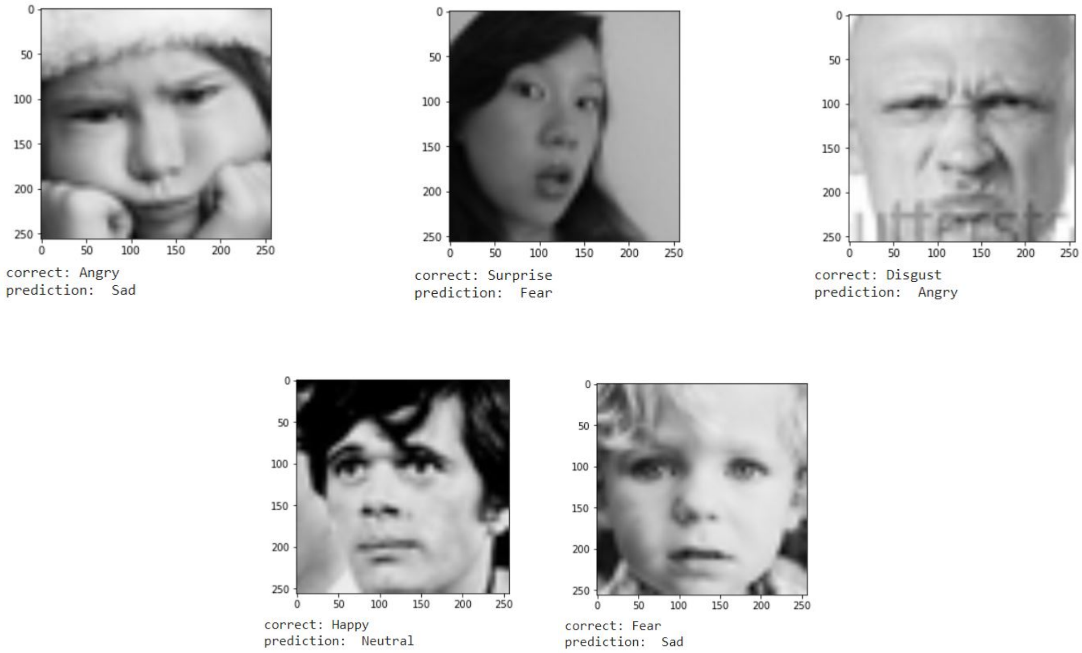


Figure 4.13: wrong classification

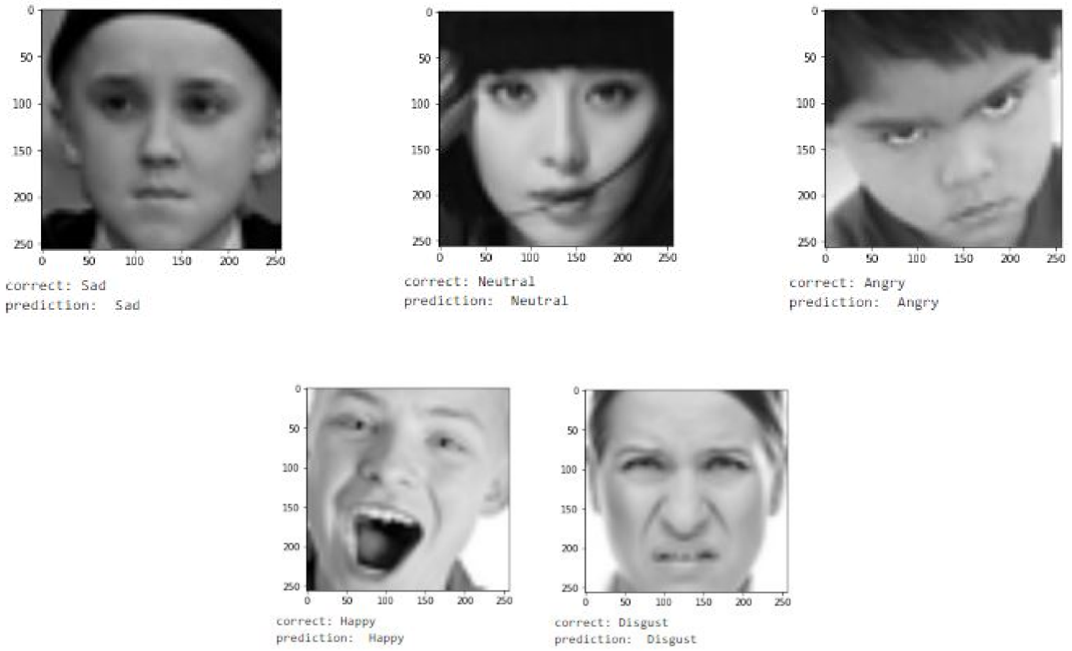


Figure 4.14: correct classification

These are some examples of images given as input to our network. Now, we can understand how difficult is the FER task, in fact for some of them there are problems even for humans to categorize with certainty the associated emotion. The most frequent misclassification, in our model, happens between "angry" and "disgust" classes, in fact even if they are different emotions, the shape of the face assumed by the people can be very similar (we can see this problem in the third image of the figure 4.13). Instead, other issues born when emotion are

very correlated. As example: the emotions of "surprise", can be felt together with "happy" or "fear" at the same moment and distinguish one of them can be hard.
This can easily lead to misclassification in the dataset's label, and so make the task of learning harder.

Chapter 5

Conclusions

In this project we have implemented a Deep Attentive Center Loss (*DACL*) for facial expression recognition. We have built the model following the indications described in the paper [1] and we have presented some variations. We have shown that the *DACL* implementation is better than using implementation based only on softmax loss or on classic center loss.
However, our result are not directly comparable to the ones described in the paper, in fact the dataset we have used is different (**FER2013**) from **Affect-net** and **RAF-DB** that are not opened to everyone.
In future works, we would like to test our model on a different dataset, to test its ability of classification on different kinds of images, and we are interested in finding a balanced and larger dataset to reach the state of the art results.

Bibliography

[1] Amir Hossein Farzaneh, Xiaojun Qi .
Facial Expression Recognition in the Wild via Deep Attentive Center Loss.
URL: https://openaccess.thecvf.com/content/WACV2021/papers/Farzaneh_Facial_Expression_Recognition_in_the_Wild_via_Deep_Attentive_Center_WACV_2021_paper.pdf

[2] Yandong Wen, Kaipeng Zhang, Zhifeng Li, Yu Qiao
A Discriminative Feature Learning Approach for Deep Face Recognition.
URL: https://kpzhang93.github.io/papers/eccv2016.pdf