Facial expression recognition application using CNNs

Peagno Eleonora
1237035

Peron Giovanni
1237783

Rossi Daniel
1211017

Abstract

The facial expression recognition is a challenging task in machine learning field, and there is an active research on this topic. Being able to make a machine understand the human emotions is a fascinating goal. The purpose of this report is to exploit all the topics studied during the Vision and Cognitive Services course and others like Machine Learning and Deep Learning in order to implement a system able to predict human emotions. We will describe how we realized this system specifying all the steps performed, from the first CNN we tried to the final model we obtained. We will illustrate all the procedure we used trying to achieve better results. Our target was not very high since this is a challenging task as we said, however with our final model we arrived to a predition accuracy of ... that is better than the human accuracy. Finally using this model we reached our goals implementing a nice application.

1. Introduction

Facial expressions recognition (FER) is an interesting and a challenging problem in machine learning field. It is also a task that can be applied in many important applications. Facial expressions have an important role in every human interaction so having a machine able to recognize and understand human expressions automatically can be very useful in many existing and novel fields [5].
One of these fields is behaviomedics that are systems which exploit automatic analysis of affective and social signals to aid diagnosis, monitoring and treating medical condition that alter behavior. Facial expression recognition can be also use in data analytics field for example to understand emotions of people that are looking at ads or political debate and make statistics related to people's preferences. Another application field for Facial expression recognition is human-computer interaction, understanding human emotions would make the attitude of systems like vocal assistants or robots much closer to the way that humans interact with each other. Recognizing expressions could also be useful to improve the identification of micro facial expressions which can be used in deceit detection applications. Due to

all these possible applications, facial expressions recognition is widely studied also because recognizing human expressions in natural condition environment is a very challenging task. With this project we aim to build a facial expressions classifier able to reach and overtake the human accuracy on this task. The main idea was focusing mostly on study different types of model in order to understand a good way to achieve valid outcomes. For this reason all preprocessing techniques that can be applied to the input data for improving classification results were not be consider.

The chosen dataset is the FER2013 it was selected after some researches, above some examples of images of this dataset are reported. FER2013 was the desired dataset, the most suited for our purpose. Training a model with a dataset like that was the challenge with the right level of difficulty we wanted. Moreover FER2013 provides a very large set of examples well differentiated in terms of person age, face pose, illumination and other realistic conditions. So using this dataset we realize an ensemble model using many types of Convolutional Neural Networks (CNNs), in order to achieve a test accuracy greater than 65.5%, that is the human accuracy measured on FER 2013 dataset [1]. Our target was reached using the final model described in the next sections, which achieve test accuracy of 71.52% on FER2013.

2. Related Work

The first step for reaching our targets it has been a research of all the scientific papers related to the facial expres-

sion recognition. We found many different works related to the topic all reviewed in *"Deep Facial Expression Recognition: A Survey"* by S. Li and W. Deng [4], thank to this paper we could find a lot of different works about the same kind of classification we were looking for. We searched for all the works reported that were using the FER2013 dataset, in order to understand a way to realize a model aligned with the best results achieved in the last years. We found six different papers related to the FER2013 dataset, they were classified by accuracy reached and type of neural networks used. After inspecting all these works we chose to follow the paper that achieved the best accuracy, moreover it used an ensemble method and we were interested in understanding better this approach. So we decided to follow the methods used in *"Facial Expression Recognition using Convolutional Neural Networks: State of the Art"* by C. Pramerdorfer and M. Kampel [6], that consists on an ensemble method made up eight different CNNs of three different types, the accuracy reported for this method was 75.2%. This paper has been enticing for us, because in it are explained the power and the bottlenecks of the CNNs models, moreover the final part of that paper was destined to explain how is possible to overcome the problems of CNNs. We found also other papers that we used as support to realize many different types of Convolution Neural Networks:

- "Very Deep Convolutional Networks for Large-Scale Image Recognition" by K. Simonyan and A. Zisserman [7];

- "Going Deeper with Convolutions" by C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich [8];

- "Deep Residual Learning for Image Recognition" by K. He, X. Zhang, S. Ren, and J. Sun [2].

Following we report a table that summarizes three of the works, executing facial expressions recognition task on the FER2013 dataset, we considered from [4].

Papers	Network type	Accuracy reached
Zhang et al. [9]	CNN Multitask Network	Test: 75.10
Kim et al. [3]	CNN Network Ensemble	Test: 73.73
Pramerdorfer et al. [6]	CNN Network Ensemble	Test: 75.2

3. Dataset

The FER2013 database was created by Pierre Luc Carrier and Aaron Courville. It was introduced during the ICML 2013 Challenges in Representation Learning, it is a large-scale and unconstrained database. This dataset was built collecting images in an automatic way using the Google image search API. In order to find useful faces images a research has been carried out combining a set of emotion related keywords with other words associated to gender, age or ethnicity. In this way about 600 strings were obtained and they were used to query the search API. Then all the images collected with this system have been cropped and resized to 48*48 pixels and they have been also converted to grayscale. We choose this dataset for many reasons, for example we find it is cited in many papers and we consider it well formed dataset for the reason that it contains images which have different illumination, subjects with different age, pose and expression intensity, moreover some images have also occlusions. In general the images contained in the FER2013 dataset represent a good sampling under realistic conditions. The most important reason that pushed us to choose it is the huge number of images that it provides. Precisely it contains 28709 training images, 3589 validation images and 3589, so in total 35887 images for each of them there is an associated class that represent seven different expression labels: Angry, Disgust, Fear, Happy, Sad, Surprised, Neutral. Below we report an histogram showing the distribution of all the FER2013's labels.

Distribution of emotions,(0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprised, 6=Neutral)
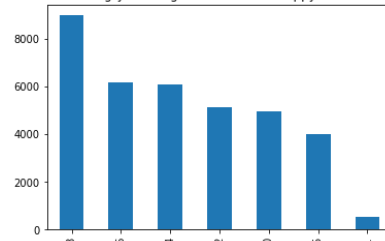
We tried to apply as few preprocessing techniques as possible because we want to focus on the model performance providing to it a dataset not too optimized. We do this choice in order to emphasize the power of the model built, in the sense we want a model able to reach interesting performance independently the goodness of the dataset provided to it. For this reasons we operate only some minimal preprocessing modification over the dataset, in order to prepare the data to be learned. First of all we notice that the class disgust is not so big with respect to the others, so we decide to merge it with the angry class. Following a graph showing the merged classes result.

Distribution of emotions,(0=Angry,1=Fear, 2=Happy, 3 = Sad, 4=Surprised, 5=Neutral)
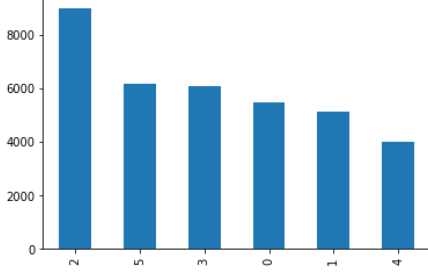
After this operation the classes are not so much leveled, in particular compared to happy class. In fact we have a lot of images for the happy class instead the ones for the surprised emotion are very few. To solve this issue we decide to replicate the 60% of the images of each class, excluding the happy class because as we said it already has enough images. In this way we make the images amount for each class more balanced and we reach the following distribution with a total amount of 52025 images.

Distribution of emotions,(0=Angry,1=Fear, 2=Happy, 3 = Sad, 4=Surprise, 5=Neutral)
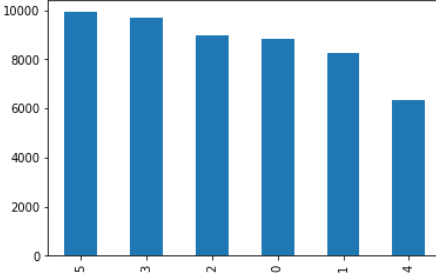
Finally we performed a normalization operation over the dataset in order to provide to the model a numeric common scale. After all these needed preprocessing steps we obtain a training set formed by 44847 images and a test set and a validation set each one formed by 3589 images.

An evaluation of other dataset has been done, we considered also CK+ (CohnKanade) dataset that is also commonly used for evaluating FER systems. The first point that brought us to discard CK+ were the huge amount of data provided by FER2013 we decided to have more images as possible in order to obtain a model that could well generalize, moreover the aim of reaching an interesting accuracy with FER2013 was more challenging. We decided to prefer FER2013 also because it appeared to us more easier to work with, in that FER2013 was provided in a csv file. A more important reason that led us to prefer FER2013 is the fact this last provides specified training, validation and test sets, CK+ instead does not, so in order to compare our results with other works it was the most appropriate one.

4. Method

The final model we build is an ensemble composed by different models from VGG, inception and resnet types, for each type we train different model in order to generalize prediction.

VGG

For realizing this kind of network we follow the structure VGG-B described in [7]. The VGG model is composed by several [CCPD] blocks and after flatting data some F layers. **VGG0** is composed by 4[CCPD], flatten layer and 3F layers with respectively 25,25,6 neurons.
VGG1 is composed by 4[CCBPD], flatten layer and 4F layers with respectively 512,256,128,6 neurons.
VGG2 is composed by 4[CBCBPD], flatten layer and 4F layers with respectively 512,256,128,6 neurons. VGG1 and VGG2 regularize the first C layer with L2 norm lambda fixed to 0.01.

Inception

Inception model is a type of network developed from Googlenet [8] with some modification according to paper[147], we remove the initial pooling layer because how the paper report [6] images are not so big. The network consist of several I blocks, flatten, dropout and 1F with 6 neurons. Every I take a parameter called n which represent the number of input neurons. We start with 64 neurons and we increment it of 32 after each I block. I block is composed as follow:

- 1C relu, (3/4)*n, K=(1,1), padding=same and L2 norm fixed on 0.002
- 2C relu, (1/2)*n, K=[(1,1),(3,3)], padding=same and L2 norm lambda fixed on 0.002
- 2C relu (1/8)*n, K=[(1,1), (5,5)], padding=same and L2 norm lambda fixed on 0.002
- PC relu (1/4)*n, K=(1,1), padding=same and L2 norm lambd fixed on 0.002

All these blocks are concatenate in a [1C,2C,2C,PC] layer and became the input of next I blocks.
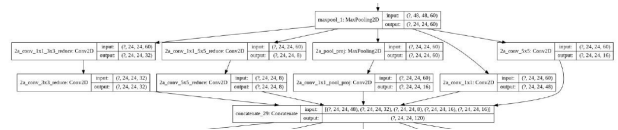

Figure 1. Representation of an inception block

ResNet

Resnet model is a network that use R blocks. Each R consists of B layer and then we have CFBC with strides=1 and the identity function. At some point we want to downsample the input we substitute to the identity function another C layer with strides=2.
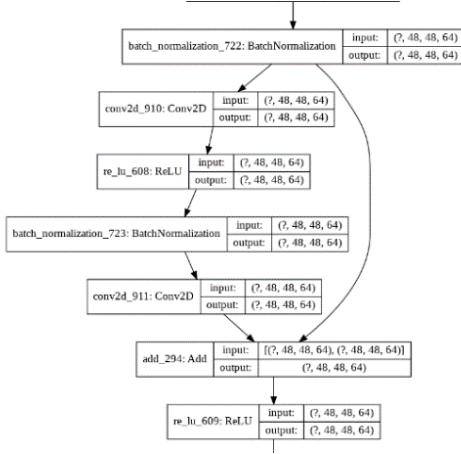
3

during training and in some models we use a regularizer to help the loss function with plateaus by reducing the learning rate if there is no improvement on validation set loss after 10 epochs.

After the training phase we obtain about eleven models. After a selection, that we explain in experiments section, we find out that the best combination of models consist of 9 models, we have 5 VGG, 1 inception and 3 ResNet models. We combine them and create two different ensemble, the first one make the average and the second the weighted average based on accuracy of the model to each class.

Figure 2. Representation of a residual block without downsample

Figure 3. Representation of a residual block with downsample

Name	0 (%)	1 (%)	2 (%)	3 (%)	4 (%)	5 (%)	Test set (%)
VGG_v0	53.53	49.19	76.87	51.60	81.92	57.00	63.58
VGG_v1	55.64	39.31	80.00	59.26	82.65	62.60	64.45
VGG_v2	60.80	31.85	80.33	61.86	76.86	61.12	65.78
VGG_v3	65.77	36.08	82.01	42.72	81.20	76.93	66.06
VGG_v4	60.03	40.72	81.56	47.93	79.03	72.98	67.18
Inception_v0	58.69	41.73	78.88	56.20	75.90	55.51	63.61
ResNet_v2	64.62	34.47	79.88	39.05	77.83	69.85	61.80
ResNet_v3	56.78	41.53	80.89	49.92	82.89	61.94	63.17
ResNet_v4	63.28	39.91	83.35	51.30	75.90	50.08	63.53

Table 2. 0=Angry,1=Fear,2=Happy,3=Sad,4=Surprised,5=Neutral

Ensemble method

Ensemble method weighted

final application webcam.

5. Experiments

Our experiments started from an unique convolution neural network with a VGG like structure. This first network was trained with Adam optimizer and it reached a test accuracy of 60.27%. So we started from this first network and we search for best methods in order to increase our accuracy. Like we said in the previous sections the final model we obtained is an ensemble method composed by nine convolution neural networks of three different types, that are VGG, Inception and Resnet. We started following the model described in [6], implementing CNNs architecture. The initial modification to the first CNN has been the addition of batch normalization layers after every convolutional layer. We also added some dropout layers useful to control the overfitting problem. Then we tried to use an SGD optimizer with initial learning rate of 0.1,a weight decay of 0.0001, momentum of 0.9, batch size of 128. We trained every network for 300 epochs. Moreover with added a learning rate regularizer in order to half the learning rate if the validation accuracy did not improve for 10 epochs. With this optimizer set as said previous, we did not reach very interesting results due to the fact that sometimes the validation accuracy remained stuck on too low value, despite the learning rate regularizer. For this reason we switched to use Adam optimizer, which being an adaptive learning

In the following table we summarize all the models realized, each one structured as descripted.

Name	Architecture	Depth	Params
VGG_v0	4[CCP] FF	10	4.8m
VGG_v1	4[CCBP] FFF	11	5.9m
VGG_v2	4[CCBP] FFF	11	5.9m
VGG_v3	4[CCBP] FFF	11	5.9m
VGG_v4	CCBP 3[CBCBP] FFF	11	5.9m
Inception_v0	CIPIIPIIPIIPF	16	4.8m
ResNet_v2	C 2[R]5[R]5[R]2[R]PF	30	15.6m
ResNet_v3	C 3[R]4[R]6[R]3[R]PF	34	21.3m
ResNet_v4	C 3[R]4[R]6[R]3[R]PF	34	21.3m

Table 1. C, P, B, I, and F stands for convolutional, pooling, batch normalization, inception and fully connected layers respectively. 3[R] means group of three residual blocks. 4[CCP] means group of four CCP blocks. Final layer and dropout have been omitted.

Optimization Each model was training with 150 epochs and we use Adam optimizer and early stopping with a patience of 10 epochs, we use checkpoints to save best weights

method solved our problem. Then we decided to reduce the number of epochs for which we trained the models to 150 for reasons of time needed for the training process. Using the tensorflow callback ReduceLROnPlateau[1] we added another system to control the learning rate. With this method we help the loss function to get rid when the gradient descent is stuck in plateaus regions. What is done is reducing the learning rate parameter multiplying it for a defined factor if there is no improvement on the validation loss after a certain amount of epochs specified with the patience parameter. In order to save time and to optimize the accuracy of our models we decided to use the early stopping technique, trying different patience setting, in this way we were stopping the training when the model were not able to improve his validation loss for 10 epochs. We also use a checkpointer callback[2] order to save the best model reached during the training process.

In addition to the dropout we tried to use also an L2 regularizer in the VGG type networks. Combining all this settings we managed to get improved results. Practically trying may different combination of modification including the change of filters used in convolution layers and the addition of some more dense layer before the output layer. Like we reported above we achieved a 67.18% of test accuracy with a VGG neural network, that is our best result for a single network model. So, after many experiments we obtained eleven different models from which we selected these following models:

- VGG_v0 with 63.58%
- VGG_v1 with 64.45%
- VGG_v2 with 65.78%
- VGG_v3 with 66.06%
- VGG_v4 with 67.18%
- Inception_v0 with 63.61%
- ResNet_v2 with 61.80%
- ResNet_v3 with 63.17%
- ResNet_v4 with 63.53%

Initially we used all the eleven models to build the ensemble model. The ensemble model was built using a bagging technique, so retrieving the predictions made from all the single models and combining in a unique final prediction. The results of every models are combined making the average of predictions of each models. In this way, that is putting together all the models we obtained a test accuracy of 67.73%. At this point we tried to improve our performance weighting the results of each models in a way to give an importance related to the accuracy reached by ever model on every expression predicted. So we computed a weighted average of

---

the results provided by the single models and in this way we improve our accuracy to 67.79%. In order to compute this average we calculated a vector of weights to apply on the predictions of every model. To compute the weights we calculated the recall on the validation set for every prediction value returned by every model and we normalize it dividing by the sum of the weights computed. The computation of the weights can be represented as following

$$\sum_{m=1,...,n} \frac{r_m}{\sum_{i=1,...,6} r_m[i]}$$

where r_m is a vector containing the recalls value for the model m Having the weights normalized means that we have a level of the importance for every predicted class of every models. The weighted average has been calculated as follows

$$\sum_{m=0,...,n} y_m \cdot w_m$$

$\sum_{m=0,...,n} y_m \cdot w_m$ where in y_m we have the predictions of the model m and in w_m we have the computed wheigths for the model m.

Final accuracy reached with ensamble application webcamm

6. Conclusion

We overtakes the accuracy reached from three papers which describe models that perform the same task with the same dataset used by us cited in [4]

References

[1] Ian J. Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will 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 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. 2013.

[2] K. He, X. Zhang, haoqing Ren, , and J. Sun. Deep residual learning for image recognition. 2015.

[3] B.-K. Kim, S.-Y. Dong, J. Roh G. Kim, and S.-Y. Lee. Fusing aligned and non-aligned face information for automatic affect recognition in the wild: A deep learning approach. 2016.

[4] S. Li and W. Deng. Deep facial expression recognition: A survey. *IEEE Transactions on Affective Computing*, 2020.

[5] M. V. B. Martinez. Advances, challenges, and opportunities in automatic facial expression recognition. *Advances in Face Detection and Facial Image Analysis*, 2016.

[6] C. Pramerdorfer and M. Kampel. Facial expression recognition using convolutional neural networks: State of the art. 2016.

[7] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. 2014.

[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, , and A. Rabinovich. Going deeper with convolutions. 2015.

[9] C. C. Loy Z. Zhang, P. Luo and X. Tang. Learning social relation traits from face images. 2015.