

# Analysing Facial Expressions for Emotion Recognition using Machine Learning Algorithms

Gonalo Matos

*T3picos de Aprendizagem Autom3tica 2020/2021*

*Departamento de Eletr3nica, Telecomunica3es e Inform3tica*

*Universidade de Aveiro*

Aveiro, Portugal

gmatos.ferreira@ua.pt

Margarida Martins

*T3picos de Aprendizagem Autom3tica 2020/2021*

*Departamento de Eletr3nica, Telecomunica3es e Inform3tica*

*Universidade de Aveiro*

Aveiro, Portugal

margarida.martins@ua.pt

**Abstract**—Analysing facial expressions is often used in emotion recognition. The aim of this paper is to analyse the performance of different machine learning models, that classify emotions through facial images taken in the wild. For this purpose 8400 images were used from the FER-2013 dataset [1]. Using Jupyter notebooks, two different machine learning algorithms were tested, support vector machine and neural networks. The best model achieved an accuracy of 38,8% classifying 6 different emotions.

**Index Terms**—emotions recognition, machine learning, neural networks, facial expressions, keras, SVM, python, jupyter notebooks



Fig. 1. Sample images from the dataset

## I. INTRODUCTION

The final decade of the 20<sup>st</sup> century brought the concept of Affective computing. The purpose of this field is to give the machines the capability of interpreting emotions and react accordingly. From car safety where the drivers emotional state is being monitored to video game testing there are numerous applications for this field. Therefore we found this theme very interesting and appealing to work with. We decided to work with images opposed to sound or video because they are lighter, meaning that the model would be trained faster.

In this project we will compare the performance of two different classic machine learning algorithms: neural networks and support vector machines.

## II. DATA PREPOSSESSING AND VISUALIZATION

The dataset used consisted in 32298 images with a training set of 28709 examples and a test set of 3589 examples. The examples were classified into seven different emotions: anger, disgust, fear, happy, neutral, sad and surprise. Due to a low number of examples (less than 500) disgust emotion was discarded.

The examples were already prepossessed. All images were in gray scale with a size of 48\*48 pixels, as we can see in "Fig 1". Furthermore the faces in the photos occupied the same amount of space and were centered.

## III. IMPLEMENTED MODELS

A seemingly simple classification problem gave us 2304 (48\*48\*1) non-linearly separable features per example to work with, a number too high for logistic regression. We had to consider more complex approaches.

### A. Computer vision

This scientific field has been studying ways to create computational models of human visual system in order to mimic its behaviour and automate it [2], having already proposed several of them. In our lessons we've learned two and we are going to explore them in this paper: Neural Networks (NN) and Support Vector Machine (SVM).

### B. Data splitting

Due to the large number of samples in the dataset we were advised by our teacher to train the network only with 1000 examples per emotion. We also created development and testing sets with 200 images each.

The development images were retrieved from the test set because the original division did not have one and development and test examples should come from the same source, so as to improve accuracy values. The original test set had more than enough examples so no images had to be repeated.

To analyse the relation of the accuracy variation with the number of emotions we used to train our model, we decided to analyse not only the 6 emotions together, but also subgroups with a smaller number of elements:

- Angry, Fear, Neutral and Surprise;
- Angry, Happy and Neutral;
- Fear, Happy and Sad;
- Angry, Fear and Surprise;
- Angry and Fear;
- Happy and Sad.

#### IV. NEURAL NETWORK

In classes we have studied code with the mathematic calculus behind this model. However, the approach followed was too simple and did not fit our problem, so, even though we have tried to change it for our context, we didn't manage to do it.

We were then advised by our teacher to use a library and have chosen to try an implementation with Keras [3], a high-level deep learning library that simplifies the process of creating and training models. It is part of TensorFlow API for Python.

##### A. Parameters definition

When defining the parameters for our model, we only had one established: the number of samples per emotion, which was 1000. We also discovered that Keras had a default batch size of 32 and a learning rate of 0.01.

In order to reduce the number of iterations needed per emotion to complete one epoch ( $1000/32=32$ ) and as consequence the time consumed for training our model, we decided to increase the batch size to 128 ( $1000/128=8$ ), reducing the iterations number in 75%. We have also decided to train our model for other 2 learning rates: 0.001 and 0.1.

There were still two parameters to be defined that were crucial to avoid overfitting: the regularization hyper parameter and the number and size of hidden layers. Because their variation and relation can vary according to the context of the problem, we decided to train our models for several of them (that we found in literature) and then decide which combination is the best. The values we defined were:

- Hyper parameter: 0, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, 3, 10;
- Hidden layers:
  - 1 layer with 20 neurons;
  - 1 layer with 144 neurons;
  - 2 layers with 144 neurons each;
  - 4 layers with 300, 200, 100 and 50 neurons;
  - 4 layers with 300, 100, 100 and 100 neurons.

Finally we had to define the number of epochs. We started by training our model with 5000, but soon understood that the value of the cost function starts to converge to a minimum after the 1000 and so we tested for 1500. This value did not reveal as good as we expected, as the validation cost accuracy is stable since the first 200 epochs but the training cost function starts getting close to 1 after those, which indicates overfitting. In "Fig 2" we can see some metrics that illustrate this. Finally we decided to go with 200 epochs.

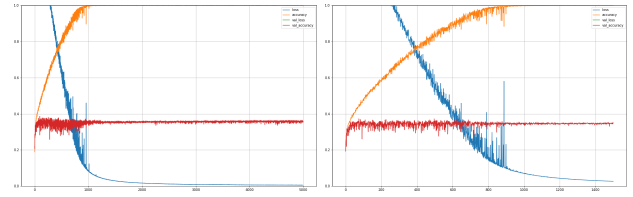


Fig. 2. Training metrics for batch size of 128, hyper parameter of 0.001 and 1 hidden layer with 144 neurons with epochs of size 5000 (a) and 1500 (b)

##### B. Training

To train our models we have built a Keras sequential model with Hidden (Dense) layers, according to the values defined in the parameters and an Output (Dense) layer, with the same number of neurons as the number of emotions being classified.

For training purposes we have also used:

1) *Activation functions*: To facilitate the parameter optimization, we implemented Relu activation function on the hidden layers and Softmax for the final layer, as it is widely used for classification with multiple labels.

2) *Cost function*: The cost function we used is the sparse categorical crossentropy loss function, also widely used in classification problems.

3) *Optimizer*: To optimize the model against the cost function and ensure it converges to an optimal solution, we have used stochastic gradient descent.

##### C. Results

The first thing we analysed after training our models was the test set accuracy prediction for the different learning rate/hyper parameter/hidden layers combinations. The best accuracy was achieved with  $\alpha=0.01$ ,  $\lambda=0.1$  and 4 hidden layers with 300, 100, 100 and 100 neurons.

In "Fig 3" and "Fig 4" we can see the accuracy for the combination of the hidden layers with the hyper parameter and the learning rate, in both cases, with the other fixed to the one that achieved the best accuracy ( $\alpha=0.01$ ,  $\lambda=0.1$ ).

With the hyper parameter variation there are other 4 combinations that returned an accuracy less than 1% below (1 without regularization) but with the alpha variation there are none (the closest is distanced by 1.17%).

	144	144 / 144	20	300 / 100 / 100 / 100	300 / 200 / 100 / 50
0	0.3142	0.3500	0.3100	0.3383	0.3658
0.001	0.3325	0.3625	0.3450	0.3575	0.3625
0.003	0.3267	0.2992	0.3408	0.3417	0.3500
0.01	0.3258	0.3408	0.3458	0.3308	0.3275
0.03	0.3367	0.3558	0.3117	0.2892	0.3375
0.1	0.3492	0.3575	0.3367	0.3692	0.3433
0.3	0.3642	0.2925	0.3483	0.3183	0.3450
1	0.3367	0.3100	0.3242	0.2675	0.3567
3	0.3533	0.3258	0.3450	0.3475	0.3358
10	0.2908	0.3542	0.3142	0.3325	0.3117

Fig. 3. Accuracy for hyper parameter (rows) and hidden layer (columns) variation with  $\alpha=0.01$

	144	144 / 144	20	300 / 100 / 100 / 100	300 / 200 / 100 / 50
0.001	0.3325	0.3242	0.3250	0.3067	0.3175
0.01	0.3492	0.3575	0.3367	0.3692	0.3433
0.1	0.3192	0.2758	0.2733	0.3100	0.1667

Fig. 4. Accuracy for learning rate (rows) and hidden layer (columns) variation with  $\lambda=0.1$

For the model with the best accuracy, "Fig 5" represents the accuracy and loss function evolution during the training process. The development set has a smoother evolution, while the training set increases the accuracy and decreases the loss function. This increase in the training set accuracy and the distance to the development suggests overfitting.

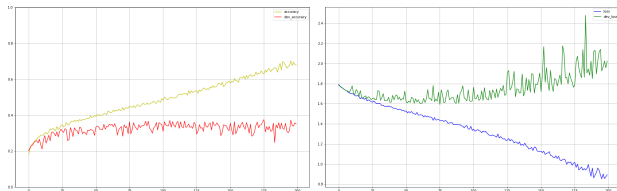


Fig. 5. Training and development sets accuracy (a) and loss (b) evolution for model with  $\alpha=0.01$ ,  $\lambda=0.1$  and 4 hidden layers with 300, 100, 100, 100 neurons

To understand better how our model was responding to overfitting with different parameters, we have decided to check the training evolution for models with the a higher hyper parameter and a different hidden layer combination.

With different hyper parameters we haven't observed the diversity of graphics we expected. "Fig. 6" shows the evolution of the training metrics for a model with the same hidden layers but with 10 instead of 0.1 as hyper parameter. The difference is almost imperceptible.

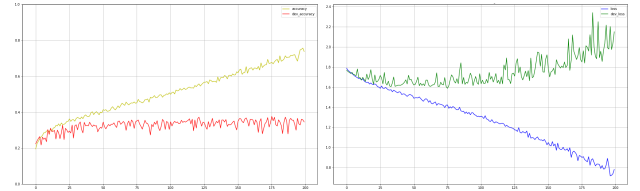


Fig. 6. Training and development sets accuracy (a) and loss (b) evolution for model with  $\alpha=0.01$ ,  $\lambda=10$  and 4 hidden layers with 300, 100, 100, 100 neurons

When we change the hidden layers the change is more evident. In "Fig. 7" it is shown the training metrics for a model with one hidden layer with only 20 neurons.

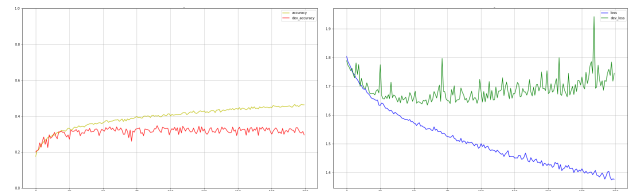


Fig. 7. Training and development sets accuracy (a) and loss (b) evolution for model with  $\alpha=0.01$ ,  $\lambda=0.1$  and 1 hidden layer with 20 neurons

Getting back to the analysis of our best model, the low accuracy that we announced before already suggested that we would have a low performance with the test dataset. In "Fig. 8" we can see that most of the positive classifications were false positives, having the surprise and happy emotions been the only ones with a greater rate of true positives.

	TP (%)	TN (%)	T (%)	FP (%)	FN (%)	F (%)	Total
angry	89 (7.42%)	776 (64.17%)	859 (71.58%)	230 (19.17%)	111 (9.25%)	341 (28.42%)	1200
fear	58 (4.83%)	847 (70.58%)	905 (75.42%)	153 (12.75%)	142 (11.83%)	295 (24.58%)	1200
happy	94 (7.83%)	885 (73.75%)	979 (81.58%)	115 (9.58%)	106 (8.83%)	221 (18.42%)	1200
neutral	82 (6.83%)	825 (68.75%)	907 (75.58%)	175 (14.58%)	118 (9.83%)	293 (24.42%)	1200
sad	9 (0.75%)	993 (82.75%)	1002 (83.50%)	7 (0.58%)	191 (15.92%)	198 (16.50%)	1200
surprise	111 (9.25%)	923 (76.92%)	1034 (86.17%)	77 (6.42%)	89 (7.42%)	166 (13.83%)	1200

Fig. 8. Test set performance for model with  $\alpha=0.01$ ,  $\lambda=0.1$  and 4 hidden layers with 300, 100, 100, 100 neurons

The confusion matrix ("Fig 9") also shows us the emotions that the model performed better with and the ones where it made more mistakes, with highlight for the high precision when classifying surprise and the opposite with angry, that was classified for other emotions more than for itself. The sad emotion was also classified a lot as other emotions, being classified correctly only 9 times!

	angry	fear	happy	neutral	sad	surprise
angry	89	28	25	45	1	12
fear	61	58	24	28	2	27
happy	47	15	94	27	1	16
neutral	49	31	21	82	2	15
sad	54	46	35	49	9	7
surprise	19	33	10	26	1	111

Fig. 9. Test set confusion matrix for model with  $\alpha=0.01$ ,  $\lambda=0.1$  and 4 hidden layers with 300, 100, 100, 100 neurons

## V. SUPPORT VECTOR MACHINE

Support Vector Machine is a supervised learning algorithm that is most commonly used in classification problems. For the development of the SVM model the python library scikit-learn was used [4], more specifically it's SVC class. The SVC class allows to create models for classification problems with multiple different kernels. The gaussian radial basis function

was the kernel choosed because the problem at hands is a nonlinear classification problem.

### A. Parameters definition

On the rbf kernel two parameters affect the models accuracy, gamma and C. Gamma values determine the importance of training data fitting. A high gamma value can cause overfitting while a low value might not capture the complexity of the data and behave similar to a linear model. The C parameter is responsible for determining the error margin accepted. An higher C value will accept a smaller margin whereas an lower value will allow an higher margin. Usually a lower gamma value compensates an higher C value and vice versa.

In order to define the optimal gamma and C parameter several combinations where tested using the following values: 0.001, 0.01, 0.1, 1, 10, 100.

To further increase parameter optimisation a second combination of gamma and C parameters was tested using a narrow interval based on the previous results, for C the following values were used: 3, 7, 10, 30, 50, with gamma values being: 0.005, 0.01, 0.05

### B. Training

For each parameter combination the model was trained with the training data and the accuracy calculated based on the prediction using the development data. The same process was repeated with adjusted gamma and C values. Afterwards, using the parameter combination with the best accuracy the model is trained again this time with both the training and the development data. The final confusion matrix is calculated using the test data.

### C. Results

In "Fig 10" and "Fig 11" bellow we can analyse the accuracy results for the different combinations of C and gamma parameters. With an accuracy of 39,7% the best parameter combination is a C value of 3 and a gamma value of 0.01.

	0.001	0.01	0.1	1	10	100
0.001	0.254	0.268	0.242	0.239	0.252	0.200
0.01	0.254	0.268	0.242	0.239	0.252	0.200
0.1	0.268	0.301	0.242	0.239	0.252	0.200
1	0.353	0.387	0.263	0.203	0.199	0.197
10	0.370	0.396	0.268	0.203	0.199	0.197
100	0.358	0.391	0.268	0.203	0.199	0.197

Fig. 10. Development accuracy for multiple C/gamma values I

	0.005	0.01	0.05
3	0.392	0.397	0.341
7	0.392	0.394	0.341
10	0.373	0.396	0.341
30	0.373	0.391	0.341
50	0.378	0.391	0.341

Fig. 11. Development accuracy for multiple C/gamma values II

With gamma and C set the model is trained again with the development and training sets. In "Fig 12" bellow we can observe the performance of the model with the testing set. The model accuracy was 38,8% with recall and precision values also being 38,8% this is due to the fact that the data is well balanced, i.e. all emotions have the same number of examples, which means that in this case accuracy is a good evaluation metric.

	TP (%)	TN (%)	T (%)	FP (%)	FN (%)	F (%)	Total
angry	55 (4.58%)	861 (71.75%)	916 (76.33%)	139 (11.58%)	145 (12.08%)	284 (23.67%)	1200
fear	55 (4.58%)	861 (71.75%)	916 (76.33%)	139 (11.58%)	145 (12.08%)	284 (23.67%)	1200
happy	96 (8.00%)	895 (74.58%)	991 (82.58%)	105 (8.75%)	104 (8.67%)	209 (17.42%)	1200
neutral	72 (6.00%)	863 (71.92%)	935 (77.92%)	137 (11.42%)	128 (10.67%)	265 (22.08%)	1200
sad	64 (5.33%)	872 (72.67%)	936 (78.00%)	128 (10.67%)	136 (11.33%)	264 (22.00%)	1200
surprise	124 (10.33%)	934 (76.17%)	1038 (86.50%)	86 (7.17%)	76 (6.33%)	162 (13.50%)	1200

Fig. 12. SVM Performance

The confusion matrix, "Fig 13" bellow, allows us to comprehend which emotions have better performance and which emotions the model most often mistakes for another. The emotion which the model recognizes best is surprise with a precision of 59%. On the contrary the model only has a precision of 28% for emotions fear and angry. The model greatly confuses a sad emotion with a neutral one with 42 sad emotions classified as sad ones. On the other hand the model can quite clearly distinguish between sad and surprise emotions, when the emotion is surprise the model only classifies the image as sad on 3,5% of the cases.

	angry	fear	happy	neutral	sad	surprise
angry	55	31	34	32	34	14
fear	34	55	21	22	32	36
happy	24	26	96	20	23	11
neutral	34	31	18	72	32	13
sad	31	28	23	42	64	12
surprise	16	23	9	21	7	124

Fig. 13. SVM Confusion matrix true value/predicted value

## VI. PERFORMANCE COMPARATION BETWEEN MODELS

The two models presented have a similar accuracy value, 38,8% for the SVM model and 36,9% for the Neural Network model. However the number of images classified with a certain emotion is significantly different between the models as we can see in "Table I". The NN model number of classifications per emotion is much more heterogeneous comparing with the SVM model, it over classifies images with anger emotion (119 more than the supposed value of 200), while massively under classifying sad images, only 16.

	anger	fear	happy	neutral	sad	surprise
Neural Network	319	211	209	257	16	188
SVM	194	194	201	209	192	210

TABLE I  
NUMBER OF IMAGES CLASSIFIED BY EMOTION

Looking at the models precision for each emotion represented in "Table II", we can observe that the precisions are similar in the two models, with surprise being the most precise in both models. The probability of the models classifying an image as surprise and being right is approximately twice the probability of being correct when predicting anger, fear emotions. We can also infer that the surprise emotion is the only one in which the number of true positives is greater than the number of false positives with the exception of sadness in the NN model. The precision is not an accurate metric for the sad emotion in the NN model because the number of images classified as sad is very low which gives precision rates much larger than accuracy rates (4,5%)!

	anger	fear	happy	neutral	sad	surprise
Neural Network	28%	30%	45%	32%	56%	59%
SVM	28%	28%	48%	34%	33%	59%

TABLE II  
PRECISION BY EMOTION

As described in the beginning of this paper, we have trained our models for different emotion combinations in order to understand its impact in the accuracy. For NN we have trained them for all the hyper parameters and hidden layers combinations with the other parameters static (200 epochs, batch size of 128, 0.01 learning rate). Similarly, for SVM we have changed the the gamma and c values.

Finally, to understand how well each model can perform for different emotions, we have selected the maximum accuracy per emotion combination. "Fig. 14" shows the results of this research.

Despite that the SVM offers, as mentioned before, a higher accuracy for the 6 emotions and that the global analysis shows that the accuracy for both models is very similar, when it comes to models with less emotions, NN obtains higher values.

Generally, we can also conclude that both models perform better when trained for distinguishing a smaller number of emotions, specially when they have particular expressions that make them easier to distinguish even by a human.

	fear angry	happy sad	fear angry surprise	fear sad happy	happy angry neutral	fear angry surprise neutral	angry fear happy neutral sad surprise
NN	0.63	0.73	0.55	0.56	0.55	0.46	0.37
SVM	0.59	0.71	0.54	0.54	0.53	0.45	0.39

Fig. 14. Accuracy on neural network and support vector machine for different emotion combinations

## VII. CONCLUSIONS

As it is in the real world we can conclude that the facial expression of some emotions such as surprise and happiness are more clearly distinct from other emotions and therefore more easily recognized by machine learning algorithms. Given the results obtained we can infer that with the dataset used classical machine learning algorithms do not perform well and deep learning algorithms provide better results, such as convolutional neural networks [5] which can achieve an accuracy of 64% for 7 emotions.

However SVM algorithms can perform well when some features are extracted from the images [6], focusing on the eyes, nose and mouth removes a lot of prejudicial noise and can achieve an accuracy of 87.9%. Contrary to ours if the dataset images are more standardized, with faces with similar physical characteristics and always in the same position the SVM algorithms can achieve high accuracy rates.

## VIII. WORK LOAD

Each student worked 50% of the project.

## ACKNOWLEDGMENT

The authors would like to thank professor Petia Georgieva, regent of TAA course, for her support and assistance throughout the project.

## REFERENCES

- [1] FER-2013 dataset <https://www.kaggle.com/msambare/fer2013>
- [2] T. S. Huang, "Computer Vision: Evolution and Promise" <http://cds.cern.ch/record/400313/files/p21.pdf>
- [3] Keras website <https://keras.io/>
- [4] Scikit-learn website <https://scikit-learn.org/stable/index.html>
- [5] Shima Alizadeh, Azar Fazel, "Convolutional Neural Networks for Facial Expression Recognition" 2016 <https://arxiv.org/pdf/1704.06756.pdf>
- [6] E.M.Bouhabba, A.A.Shafie and R.Akmeliawati, "Support vector machine for face emotion detection on real time basis", 2011 <https://ieeexplore.ieee.org/document/5937159>