MASTER IN DATA SCIENCE

JOSE ANTONIO VILLEGAS ZELAYA

A PRECISION COMPARATIVE ANALYSIS OF DIFFERENT MACHINE LEARNING AND DEEP LEARNING MODELS FOR THE PREDICTION OF FACIAL EXPRESSION

Support vector machines have proven to be very robust for classifications and generalizations in many cases. In the same way, K-nearest neighbors (KNN) is also a model that is used as a classifier in many areas such as Financial, medical, agribusiness, etc. In this paper we will use and compare these two models to try to predict emotion through a dataset composed of images of people's faces and their facial expressions.

INTRODUCTION

Facial expression is one of the most important forms of non-verbal communication, it is a way in which humans exchange information that tells us about the mood of our peers. In the field of Computer Vision there are many applications for the recognition of facial expression, "FER", for its acronym in English. As examples we have: Man/Machine Interaction, Biometrics, Customer Satisfaction, Smart Medicine, etc.

However high intra -class variation and low inter -class variation make FER a very challenging problem in the field of Computer Vision (Saeed , et al. 2018). Normally, this based on social interactions, facial expressions can be classified into: happiness, sadness, anger, fright, surprise, disgust and finally we can also add the "neutral" expression.

Most of the recent studies focus on solving this problem using CNNs (Convolutional Neural Networks). Many of the studies focus on algorithms to extract better attributes. Example: Gabor filter or an RBF network . (Karkra . 2016).

In this work we use a selection of three Machine Learning models and a couple of Deep Learning Network models to predict facial expression using the FER2013 Dataset , available on Kaggle . We made special emphasis on the use of VGG16 for attribute extraction, and finally VGG16 with Data Augmentation for attribute extraction and prediction. We will compare results at the end.

RELATED WORK

VGG is a Convolutional Neural Network proposed by K. Simonyan and A. Zisserman in their publication belonging to the University of Oxford; “ Very Deep Convolutional Networks for Large-Scale picture Recognition ”. His model achieves 92.7% accuracy on ImageNet , which is a dataset of more than 14 million images belonging to 1000 classes. It is an upgrade to another network called AlexNet .

Regarding FER, we researched some works like the one by Usman et al, that proposes a three-step solution. In the first stage, a Viola-Jones method is used to detect faces in the images, in step two, HoG attributes (Histogram of Oriented Grandients) of the identified areas of interest, at the same time as autocoding and dimensionality reduction through PCA (Principal Component Analysis). Finally, SVMs Support Vector Machines are used for the Classification process. A similar process is used by N. Kauser and J.Sharma with the difference for the attribute extraction stage, a Local Binary Pattern (LBP) is used.

Data Augmentation has already come a long way in Machine Learning. Initial uses experimented with the concept of label-preserving transformations, to simulate larger training datasets to reduce the effects of overfitting and improve generalizability (Bernhard , et al. 1996). For example, Simard et al translate or rotate images randomly assuming that the image labels will not change with such small perturbations.

METHODOLOGY

We use the Python programming language through colab.research notebooks and Anaconda jupyer notebooks. For the implementation of the models we use the Scikit Learn libraries and its different classifiers, support vector machines, ensembles; in addition to its preprocessors and data splitters. For the convolutional neural network we use keras from Tensor Flow . We also import the pretrained model from VGG16 via Tensor Flow . As we already mentioned, we work with the FER2013 dataset downloaded from kaggle in two versions. One Divided into two folders, training and Tested respectively, which contain the 48x48 size jpg files. And the other in csv format

ACTIVITIES AND RESULTS

1. Dataset.- The dataset that we will use is that of FER 2013. It is a dataset that has 28709 instances classified into 7 labels that are: Angry , Disgust , Fear , Happy , Sad , Surprise and Neutral, it is divided into Training Data and Training Data. testing. This dataset presents some challenges. Initially we can see that it is unbalanced with more classifications directed at " happy ". Another very typical complication of images is high dimensionality .

2. To deal with the high dimensionality, we will do a PCA Principal Component Analysis to reduce it. The PCA can obtain a set of components that allow us to map the data to another space, where the components can be ordered according to their importance. This way we can project the data into just a few principal components. In the attached Python code we see that we reduce the number of attributes by a factor close to 10. Speciflcy, from 2304 attributes to 267 attributes.

3. SVMs . SVMs (Support Vector Machines) can be used for multiclass classification like the one we need for this FER facial expression recognition problem. In the attached Python code we can use SVC from the Scikit library Learn . We fit the model to our training data processed with the PCA. Initially we tested the SCV model with a linear kernel . We obtain a score of 0.37 for the validation data. Then we fit the same model, but this time with a " rbf " kernel which stands for Radial Basis. Function. We see a great improvement in the score which is now 0.43.
4. KNN.- We imported the KNeighbor Classifier from the Scikit Neighbors library Learn . This time we do not use the training data processed by the PCA. We get a score of 0.38 for the validation data.

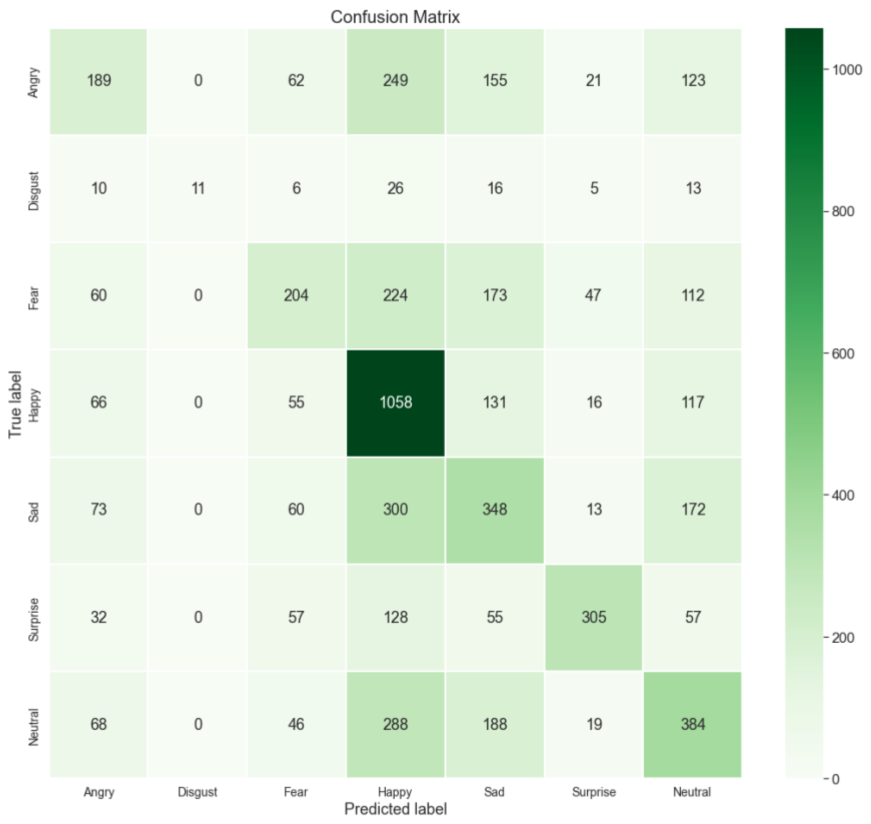
5. Random Forest.- The Random classifier Forest is imported from scikit 's ensemble library Learn . In the same way we adjust our training data and measure the score in the validation data. This gives us 0.27

6. **CNN.** We import the Sequential from the keras.models library to make our layer sequence. We imported Conv2D, BatchNormalization , MaxPool2D, Dropout , Flatten , Dense from the Keras layer library and Tensor Flow . Finally we import callbacks , the Adam optimizer and the ImageDataGenerator Image preprocessor . We build the 4-layer sequence of our neural network and compile with the Adam optimizer. We fit our model to our training and validation data with 100 epochs and 64 batches . The score on the validation data gives us 0.64, much better than the scores of the previous models.

7. **VGG16.** Here we no longer need to make a sequence of layers because we downloaded the pre-trained model from keras.applications , with the training weights of Imagenet , we adjusted the dimensions of the input to the dimensions of our images. We define a function to extract attributes from the directory with the training and testing folders that contain the images in jpg format . In the function the images go through the Keras image preprocessor ImageDataGenerator and after being processed, the attributes are extracted through . predict of the vgg model . We get our training attributes with their respective tags and our validation attributes, also with their respective tags. We fit our data to a one-layer CNN to train with the extracted attributes . We obtain a score on the validation data of 0.45.

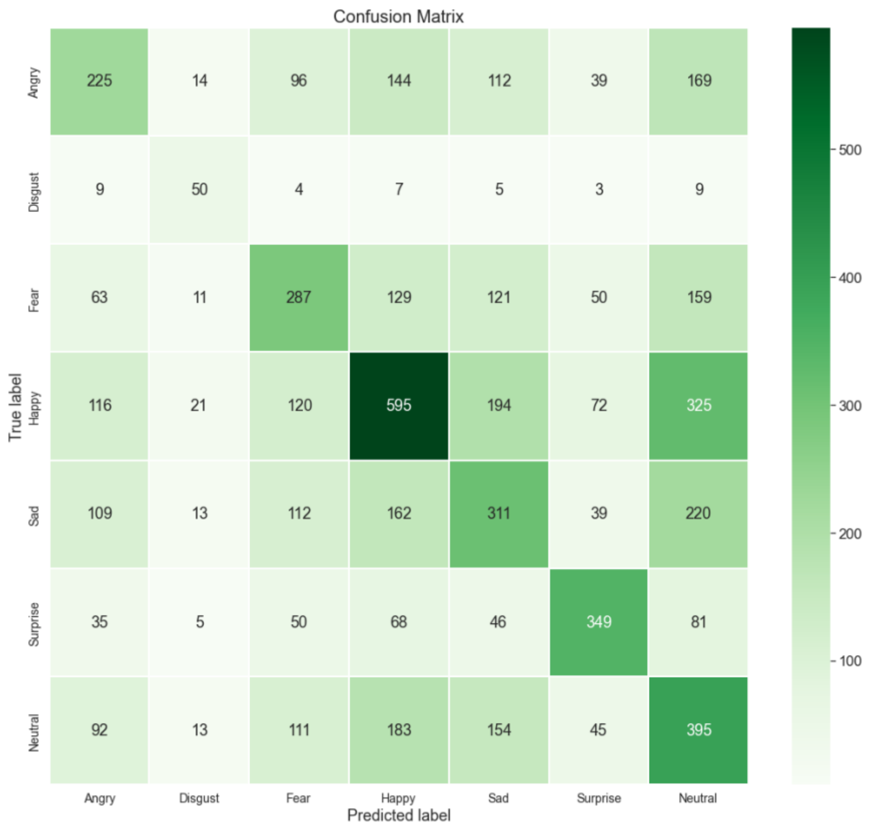
8. **VGG16 WITH DATA AUGMENTATION.** Once again we pass our training and validation images (jpg files) through the keras ImageDataGenerator , but this time we scale, rotate, width, height, zoom and rotate the image. We then generate a new model with a sequence that includes our pretrained VGG16 model and adjust the optimization and loss parameters. We fit our “boosted” data to this new model. We obtain a score on our validation data of 0.66, the best of all

SVM Confusion Matrix with RBF kernel



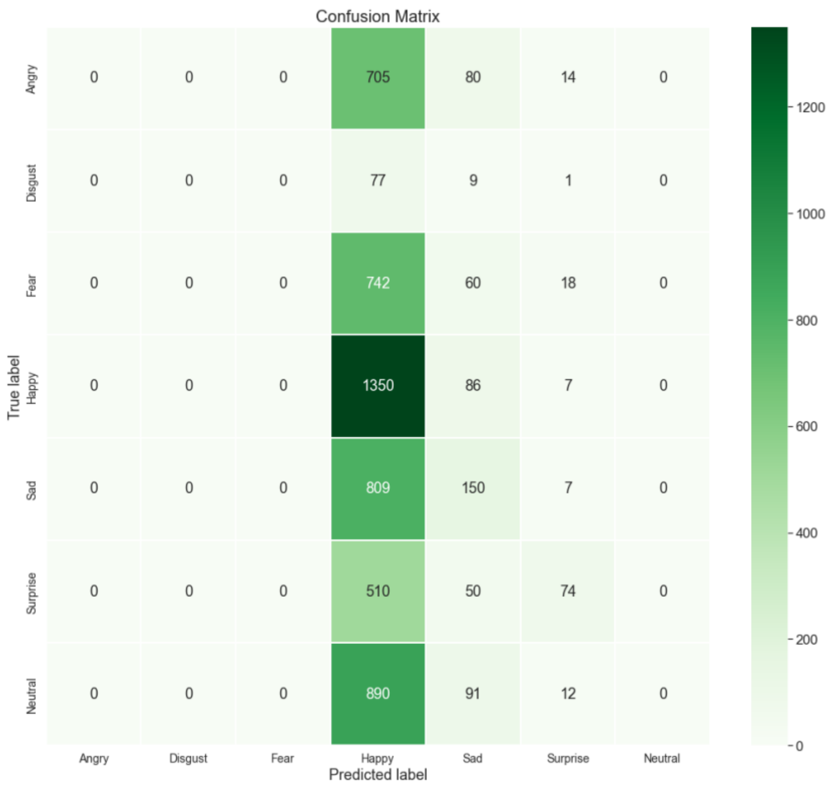
True label \ Predicted label	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
Angry	189	0	62	249	155	21	123
Disgust	10	11	6	26	16	5	13
Fear	60	0	204	224	173	47	112
Happy	66	0	55	1058	131	16	117
Sad	73	0	60	300	348	13	172
Surprise	32	0	57	128	55	305	57
Neutral	68	0	46	288	188	19	384

KNN Confusion Matrix



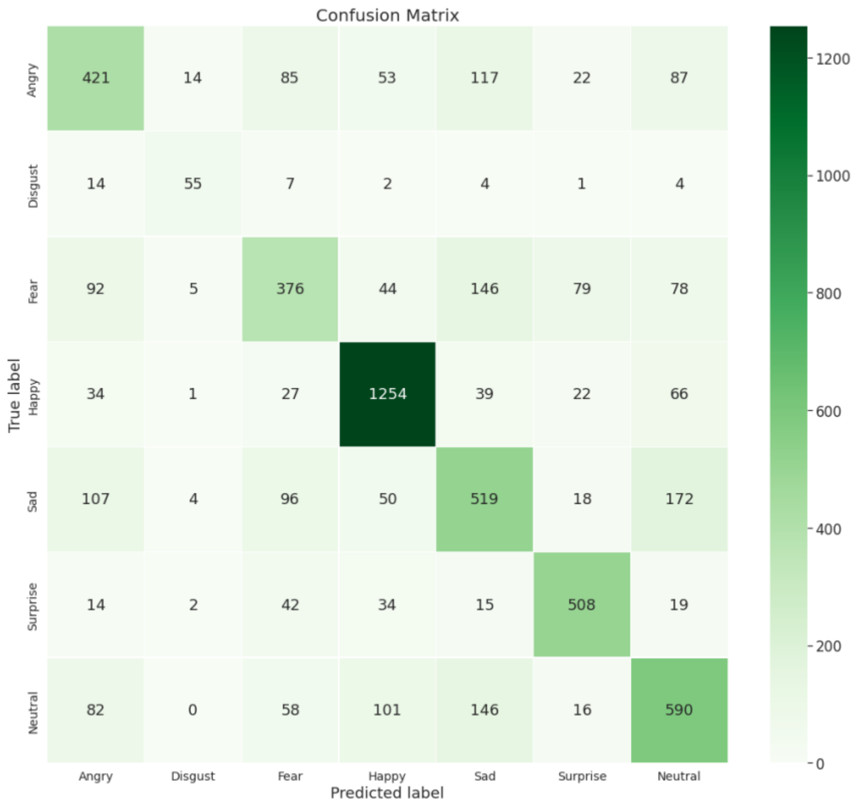
True label \ Predicted label	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
Angry	225	14	96	144	112	39	169
Disgust	9	50	4	7	5	3	9
Fear	63	11	287	129	121	50	159
Happy	116	21	120	595	194	72	325
Sad	109	13	112	162	311	39	220
Surprise	35	5	50	68	46	349	81
Neutral	92	13	111	183	154	45	395

Confusion Matrix Random Forest



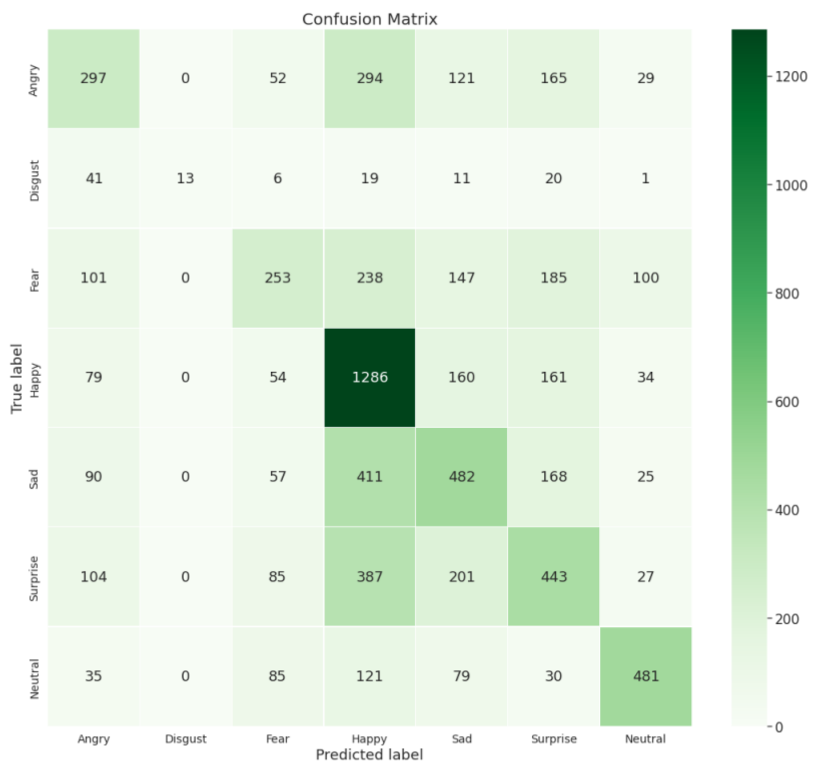
	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
Angry	0	0	0	705	80	14	0
Disgust	0	0	0	77	9	1	0
Fear	0	0	0	742	60	18	0
Happy	0	0	0	1350	86	7	0
Sad	0	0	0	809	150	7	0
Surprise	0	0	0	510	50	74	0
Neutral	0	0	0	890	91	12	0

CNN Confusion Matrix

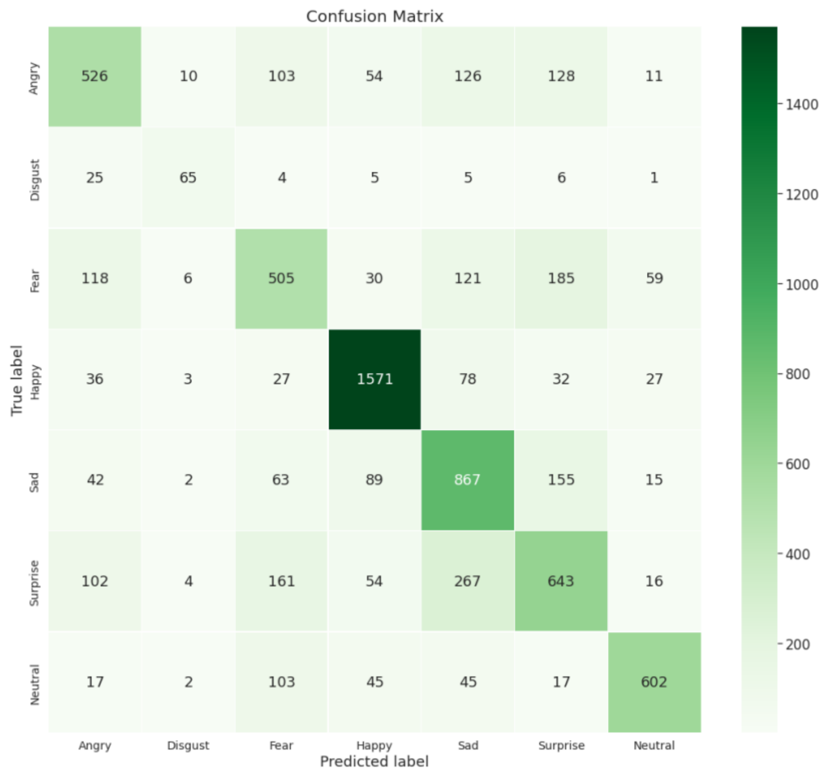


	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
Angry	421	14	85	53	117	22	87
Disgust	14	55	7	2	4	1	4
Fear	92	5	376	44	146	79	78
Happy	34	1	27	1254	39	22	66
Sad	107	4	96	50	519	18	172
Surprise	14	2	42	34	15	508	19
Neutral	82	0	58	101	146	16	590

VGG16 CONFUSION MATRIX

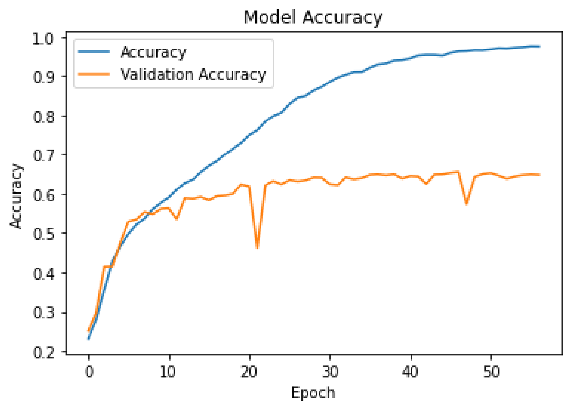


CONFUSION MATRIX VGG16 DATA AUGMENTATION

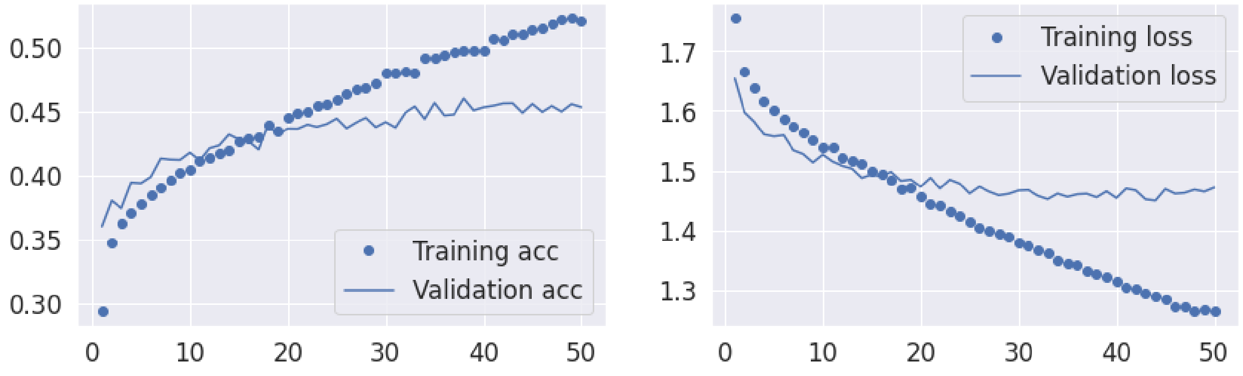


PRECISION AND LOSS CHARTS

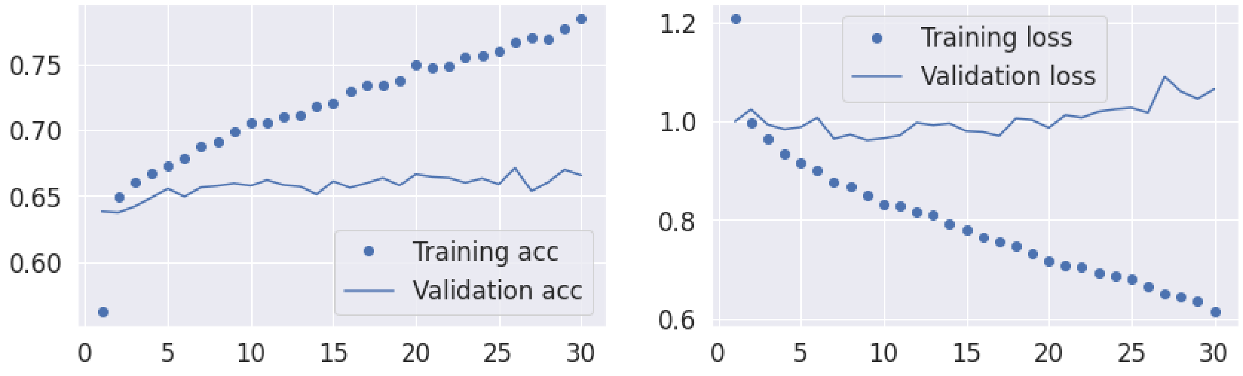
CNN



VGG16



VGG16 DATA AUGMENTATION



CONCLUSIONS AND FUTURE WORK

We conclude that "Data Augmentation " has been key to increasing the prediction accuracy of our pre-trained VGG16 model adjusted to the images extracted from FER20123. However, we could intuit that if we applied the image pre-processing known as Data Augmentation to the first CNN, perhaps the precision would have increased considerably. In the same way , with the first models, a pre-processing could have been done with techniques such as HOG (Histogram oriented Gradient) at the same stage as the Principal Component Analysis, perhaps with this also the accuracy scores on the validation data would go up considerably, especially with Support Vector Machines SVMs . We will leave these experiments for the near future.

REFERENCES

Bernhard Sc̈holkopf, Chris Burges, and Vladimir Vapnik. Incorporating invariances in support vector learning machines.
In ICANN. Springer, 1996

Patrice Simard, Yann LeCun, and John S Denker. Efficient pattern recognition using a new transformation distance. In NeurIPS, 1993.

Saeed, S., Baber, J., Bakhtyar, M., Ullah, I., Sheikh, N., Dad, I., & Sanjrani, AA (2018). Empirical evaluation of svm for facial expression recognition. *International Journal of Advanced Computer Science and Applications* , *9* (11).

Shan, C., & Braspenning, R. (2010). Recognizing facial expressions automatically from video. In *Handbook of ambient intelligence and smart environments* (pp. 479-509). Springer, Boston, MA.

Khan, SA, Hussain, A., & Usman, M. (2016). Facial expression recognition on real world face images using intelligent techniques: A survey. *Optik* , *127* (15), 6195-6203.

Dataset . https://www.kaggle.com/datasets/nicolejyt/facialexpressionrecognition