Face Detection (from images) - comparison between SVM and Neural Network

João Vasconcelos, Nº Mec: 88808
DETI, University of Aveiro[1]
j.vasconcelos99@ua.pt

Vasco Ramos, Nº Mec: 88931
DETI, University of Aveiro[1]
vascoalramos@ua.pt

***Abstract*—Face detection and recognition have become more and more important in the fields of security and information access. In this paper, we compare two methods of face detection from images: Support Vector Machines and Neural Networks, both based on Histogram of Oriented Gradients (HOG) features. To achieve satisfactory results that we could assert as realistic we used K-Fold Cross Validation with randomly selected parts of the train dataset and found the best hyper-parameters so we could minimize the problem of overfitting.**

***Index Terms*—Face Detection, Image Processing, Computer Vision, Data Augmentation, HOG, SVM, Neural Networks.**

I. State of the art review

Detection of face and non face images is a classical problem in computer vision forming the first step to all the facial analysis methods like face recognition, face alignment and face modeling. For this reason, building a system that detects faces in images can be very useful and it could be implemented in cctv systems or even in lip reading systems.

Although it's a hard problem, over the years proposals of face detection applications have been made, being frequent the usage of SVM as can be found in [1], [2] and NN such as in [3], [4] (Neural Networks are also frequently used in facial recognition as stated in [4]). Most recently there have also been some studies to analyse which of these models are better options depending on the use-case scenario, as in [5], where is conducted an in-depth analysis of the advantages and disadvantages of using SVM and NN applications to perform face detection.

In addition, it is not only important to choose which model to use, but also which strategy to process the images, because the full processing of an image would be too expensive and unrealistic. Fortunately, there is also a lot of research in this field and some alternatives such as DCT and HOG, that are both explored in detail in [6], or even Skin Segmentation as detailed in [7].

At last, given the small dataset that we had, we also did some research about the advantages of using data augmentation to expand our universe of examples and found a recent work done in 2019 where it is detailed how data augmentation can improve the accuracy and performance of models based on NN and CNN, details can be found in [8].

[1] This work was done for the course of Topics of Machine Learning at the University of Aveiro, under the supervision of the course's head teacher, Pétia Georgieva.

II. Introduction

This report has the goal of exposing and explaining our project developed for the course of Topics of Machine Learning which focuses on Face Detection from images.

Throughout this paper, we present a comparison between an SVM-based approach and an NN-based approach, both taking advantage of data augmentation and HOG as the feature extraction method and, in section VII, we compare the models we developed with the models detailed in some of the research work we showcased in section I.

III. Data description, visualization and statistical analysis

We used a dataset given by the teacher containing two distinct folders: one with face image examples and the other with non-face image examples. The images are in gray scale and weren't normalized as can be seen in figures 1 and 2.

Fig. 1. Overview of the images with faces in the dataset.

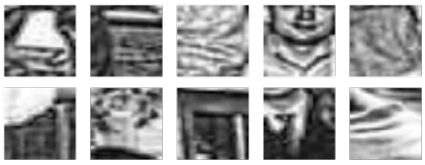

Fig. 2. Overview of the images without faces in the dataset.

This dataset contains 124 images, 69 of which are face images and 55 are non-face images, with a window of pixels of 27×18. We can conclude that the dataset is a little skewed, having more face images than non-face images as it is represented in figure 3.

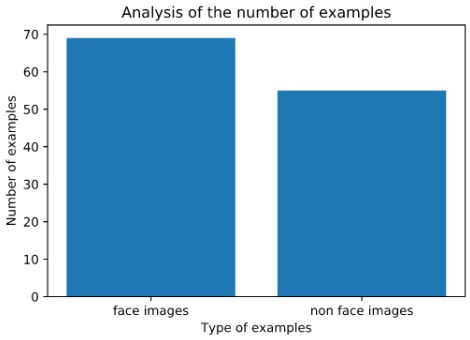

Fig. 3. Analysis of the number of examples for each type of image.

IV. Data preprocessing

A. Resizing and Normalization

Before applying the classification models we preprocessed our image data. We started off by resizing each image to a window of pixels of 64×64, forming a square. Although all the images had the same size before this step, we decided to resize them to a base size that would form a square because it's usually standard procedure in face detection problems.

After this step we validated that each image was in gray-scale, converting from RGB to GRAY and making the problem less complex by having only one dimension to analyze instead of three. We also applied Adaptive Histogram Equalization in the images, so that contrast and luminosity effects were reduced, following the advice and procedure of the authors of the paper [2].

To finalize the preprocessing phase, we normalized each image by dividing the pixel values by 255 achieving consistency in all our images, with the pixel values ranging from zero to one.

B. Data Augmentation

As we stated in section III, the dataset we were given had a total of 124 images. This is an incredibly small dataset and we weren't able to achieve satisfactory results with it, so, we decided to expand the dataset with data augmentation, following some of the practices showcased by the research work explained in [8].

Considering the slight imbalance between the number of examples of face images and non-face images we adopted the following strategy:

- Each face image produces 3 new images through reflection and dislocation.
- Each non-face image produces 4 new images through reflection, dislocation and by inserting a random level of noise.

Thereby, ending up with a total of 551 images, 276 of which are face images and 275 are non-face images.

C. Histogram of oriented gradients

The Histogram of Oriented Gradients (HOG) is a feature descriptor used in computer vision, more specifically in the field of object detection. Instead of our algorithm analysing each image,pixel by pixel, the HOG will count occurrences of gradient orientation in localized portions of an image, improving it's performance. Our implementation was based on the example documented in [9], provided by the scikit-image documentation.

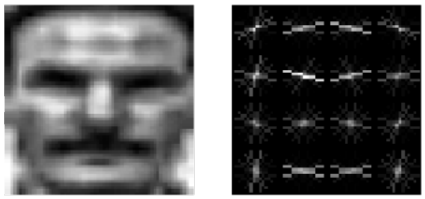

Fig. 4. HOG feature extraction

Figure 4 shows an example of an image before and after applying the Histogram of Oriented Gradients extraction.

V. Description of the applied machine learning algorithms

A. Support Vector Machine

Support Vector Machines (SVM) are highly used due to high empirical performance and somewhat-simple computational complexity, being binary classifiers that when allied with strategies, such as *one-vs-all*, can be expanded to become multi-class classifiers.

In our case, SVM was our first approach to the face detection problem, because the dataset, even after the process of data augmentation, is not very large and we are dealing with a binary classification problem, so we decided to start our initial approach with a simpler model like SVM.

Initially we divide our data in two subsets: training subset with 80% of the images and testing subset with 20% of the images.

In SVM there are two main hyper-parameters that can be tweaked to improve the performance of the model, C and Gamma. To get the best performance out of the model we made every combination possible for C and gamma, starting with one of the values present in table I.

TABLE I
SVM - Initial Configuration

Train Data	80%
Test Data	20%
Gamma	[100, 33, 10, 3, 1, 0.3, 0.1, 0.03]
C	[0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30]

To validate which combination of hyper-parameters enabled the best performance out of our model we used K-Fold Cross Validation where our training set was divided in $K = 5$ smaller sets, where 4 were used for training and 1 was used to validate the model. With this approach we didn't need to reduce the training dataset by creating a validation dataset and the results

did not depend on a particular random choice for the pair of (train, validation) datasets.

After validating every combination, as we can see in figure 5, the values of C and gamma where our model performed the best are: C = 30 and Gamma = 1.

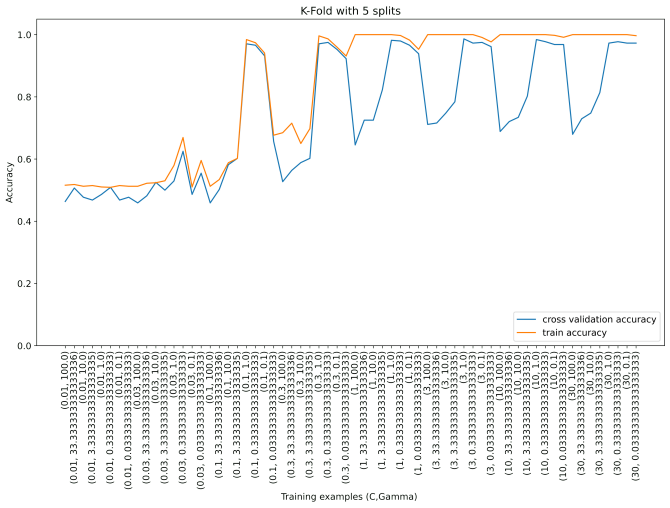

Fig. 5. SVM - Cross-Validation using multiple values for C and Gamma

Based on the Cross Validation data, we created an SVM model with a Radial Basis Function (RBF) kernel, and with the hyper-parameters C = 30 and Gamma = 1, that returns 1 when the image is a face, and 0 otherwise, as we can see in table II.

TABLE II
SVM - FINAL CONFIGURATION

Train Data	80%
Test Data	20%
Kernel	Radial Basis Function (RBF)
Gamma	1
C	30

Finally, to ensure that the model didn't end up in a local minimum, and that our model was actually learning we run the all process 50 times and calculated the mean values to both train and test phases.

B. Neural Network

Neural Network (NN) is a classifier that tries to replicate the structure of the human thought.

Being the purpose of the paper to compare the performance of SVM vs NN, our second approach was to implement a Neural Network with a Multi-layer Perceptron Classifier [10].

Just as in our SVM flow, we initially divide our data in two subsets:

- Training subset: 80%
- Testing subset: 20%.

Similar to what was done in SVM, we used HOG as our feature extractor. Hence, the specific implementation we used has 3 layers: $InputLayer \rightarrow HiddenLayer \rightarrow OutputLayer$, with the follow specification:

- Input Layer: 128 neurons (the respective number of features extracted by HOG);
- Hidden Layer: 10 neurons (as default value, the optimized number of neurons will be specified ahead);
- Output Layer: 1 neuron.

Furthermore, we used *RELU* as the activation function, the loss function we used was *Log-Loss Function*, using the *LBFGS* weight optimizer (we decide to use *LBFGS*, because, according to Scikit-Learn documentation [10], it can converge faster and perform better, given smaller datasets, which is our case), and has the default learning rate, we used $\alpha = 0.001$, which will be optimized further ahead. To summarize we can see the initial configuration of our Neural Network in table III.

TABLE III
NN - INITIAL CONFIGURATION

Train Data	80%
Test Data	20%
Input Layer	128 neurons
Hidden Layer	[10, 20, 30, 40, 50, 60, 70] neurons, $default = 10$
Output Layer	1 neuron
Activation Function	*RELU*
Loss Function	*Log-Loss Function*, via *LBFGS* weight optimizer
Learning Rate (α)	[0.001, 0.01, 0.05, 0.1, 0.5, 1, 10], $default = 0.001$
Number of Iterations	[100, 200, 300, ..., 900, 1000], $default = 100$

As done in the previous section, the SVM aproach V-A, we optimized the hyper-parameters of the NN model to enable its best performance. Just like in SVM, we did this by applying K-Fold Cross Validation, with $K = 5$. The hyper-parameters we decided to vary, and consequently optimize, were as follows:

- Hidden Layer size (the number of neurons in the Hidden Layer);
- Learning rate (α);
- Number of iterations.

After validating every combination, as we can see in figure 6, our model performed the best when:

- $HiddenLayerSize = 60$ neurons;
- $\alpha = 1$;
- $NumberOfIterations = 700$.

Fig. 6. NN - Cross-Validation using multiple-value combinations

Based on the Cross Validation data, we created a Neural Network with a Multi-layer Perceptron Classifier, and a final configuration that can be seen in table IV.

TABLE IV
NN - FINAL CONFIGURATION

Train Data	80%
Test Data	20%
Input Layer	128 neurons
Hidden Layer	60 neurons
Output Layer	1 neuron
Activation Function	*RELU*
Loss Function	*Log-Loss Function*, via *LBFGS*
Learning Rate (α)	$\alpha = 1$
Number of Iterations	700

Finally, and as we did in our SVM approach, to ensure that the model didn't end up in a local minimum, and that our model was actually learning we run the all process 50 times and calculated the mean values to both train and test phases.

VI. PRESENTATION AND DISCUSSION OF RESULTS

As we briefly explained in section III, the number of initial face examples is slightly greater than the number of non-face examples, so, we decided not to use accuracy as performance metric, because it becomes misleading. Instead, we used the combination of **F1 Score**, **Accuracy**, **Precision** and **Recall** to compare both models.

To clarify these concepts:

- **Precision** - the fraction of correctly classified positive examples from all classified as positive.
- **Recall** - actual positive rate of all positive examples, that is, the fraction of correctly classified examples.
- **F1 Score** - weighted average of Precision and Recall.

Furthermore, these concepts have mathematical representations, as follows:

- $Precision = \frac{TP}{TP+FP}$
- $Recall = \frac{TP}{TP+FN}$
- $F1Score = 2 * \frac{Recall*Precision}{Recall+Precision}$

A. SVM

After optimizing our SVM classifier, we tested it with our testing dataset. As we can see in figure 5 previously showed, the results of both train and cross-validation phases were good, as we could find a combination of parameters where the accuracy of the model in both phases had the value of 1.

Hence, in the table V we can clearly see what values of **F1 Score**, **Accuracy**, **Precision** and **Recall** our model was able to achieve.

TABLE V
SVM RESULTS

	Train & Cross-Validation	**Test**
F1 Score	1	0.985
Accuracy	1	0.985
Precision	1	0.985
Recall	1	0.986

As we can see, and reiterating the behaviour of our SVM classifier on both testing and cross-validation phases, we can see that all four metrics have the value of 1. In the testing phase, the results were not perfect, that is, the values are a little bit lower than 1, however, were quite good, as we were able to achieve a success rate approximately close to 0.985. It is necessary to remember that these results are not related to just one execution, but to 50 executions of the entire flow. This means that the values presented in table V, represent the average of each metric for the 50 repetitions of the entire flow (dataset splitting, training, cross-validation and testing).

Furthermore, to help us understand if the classifier was systematically incorrectly classifying one of the classes, we decided to produce the confusion matrix that can be seen in figure 7.

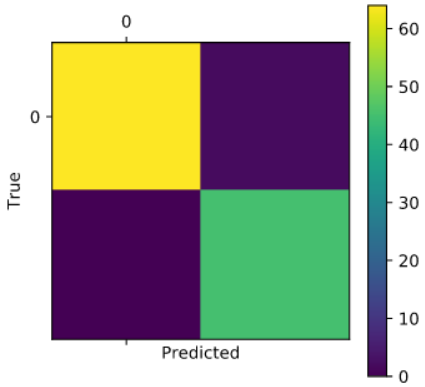

Fig. 7. SVM - Confusion Matrix

As we analyse the confusion matrix related to the results on the testing dataset, in figure 7, we can see that the model it's not systematically miss-classifying some of the classes, which means that our data augmentation of the dataset was able to regularize the learning process of our model. We can also see that in each class we have less than 5 examples miss-classified, which we see as positive outcome.

B. NN

To the Neural Network classifier, the adopted flow was similar to the one detailed above, that is, we optimized our NN model and then tested it on the testing dataset.

The table VI summarizes the result values the NN model was able to achieve.

TABLE VI
NN RESULTS

	Train & Cross-Validation	Test
F1 Score	1	0.973
Accuracy	1	0.973
Precision	1	0.974
Recall	1	0.974

As can be seen in table VI, as in our SVM model, for training and cross-validation phases, the model achieve the value of 1 to every metric, but for the testing phase, our NN model performed worse than our SVM model, achieving the average values of 0.97 to each evaluation metric.

Moreover, we also produced the confusion matrix to this model, as can be seen in figure 8. As in the SVM model, our NN classifier it's not systematically miss-classifying the examples, which confirms the conclusion we stated above about the data augmentation process. As in our SVM model, we can see that the miss-classified examples were also on a reduced number, although a little higher than in the SVM model. With this, we were able to conclude that neither of the models had systematic miss-classification problems and that our NN model was not able to generalize its predictions as much as the SVM classifier.

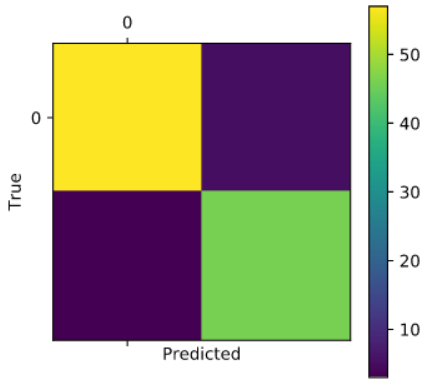

Fig. 8. NN - Confusion Matrix

The purpose of this paper was always to compare the two models against each other. However, we always thought the NN model would be better than the SVM model, even if by a little range. After these results, we came to the conclusion that this could be related with a number of causes:

- The dataset size that was not that big, and can harm the learning process of the NN model, that ends up needing more data than the SVM model.
- The feature extractor function: both models used the same feature extraction function (HOG) as a mean to achieve common ground and a true state of comparison, and that could indeed be one of the reasons, since other feature extractors, such as *Fast Gabor Filtering*, produce better results on Neural Network models.

These possibilities will be further detailed in section VII.

C. External Test Examples

Finally, after testing our two models with the testing dataset, we decided to test both models with external images that weren't part of the original dataset. Thus, we slightly modified how we processed the images and its predictions so we could test an example with multiple faces and a more complex context to assert how our models would handle that test case.

As we can see in figures 9 and 10, both models successfully identified all the existent faces in the images and both models produced instances of false positive results. We can also see that the NN model produced more false positives (it produced three false positives in the testing image) than the SVM model, that only produced one. This does not come as a surprise, because we already saw that our SVM model produces better results than our NN model.

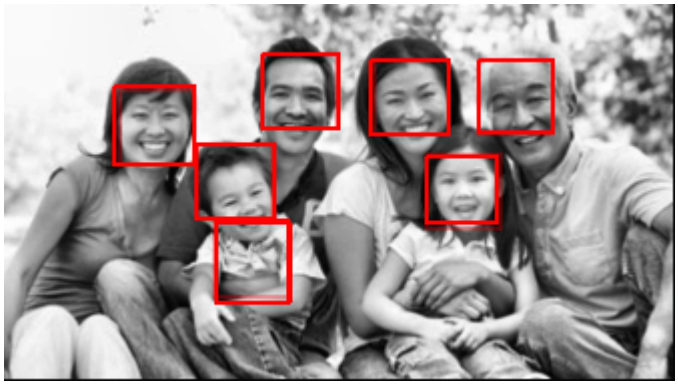

Fig. 9. SVM - Family External Image Test Result

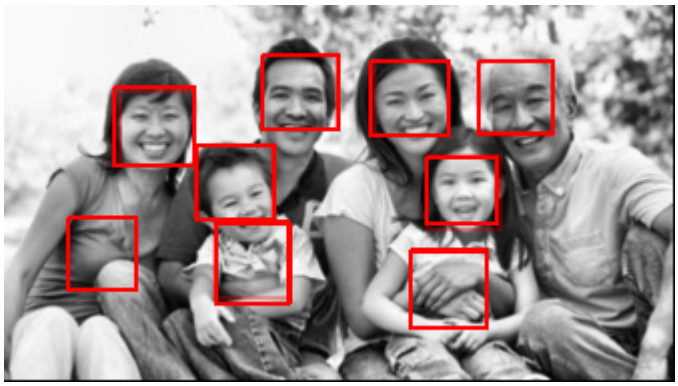

Fig. 10. NN - Family External Image Test Result

With this external test image, we were asserted that both models were able to correctly detect all existent faces, and that both had problems with false positives, which was expected as none of our models produced result scores of approximately 1 in the testing phase.

VII. NOVELTY AND CONTRIBUTIONS

In view of the analyzed papers and research work, although not all are using the same data (since we didn't find any

papers using the same dataset we used), we consider to have not made any novelty or contributions regarding our work, as we kept very close to what has been done in the context of face detection. However, considering the topics we will discuss next, we consider to have owned some positive impact regarding the success rate of the face detection system.

The next analysis will be made taking into consideration that none of the found research work used the same dataset we did, so most of our comparisons are not totally representative of what may be the reality, but are an indication of what we were able to achieve in comparison to other's work, given our dataset.

A. Support Vector Machine (SVM)

Regarding the SVM classifier, in the work present in [1] the authors stated an accuracy of 95%, in the paper in [2] the authors stated an accuracy of 97.1% and in the study in [7] the authors stated an accuracy of 94.7%. Considering that we were able to achieve an F1 Score, Precision, Recall an Accuracy of approximately 98.5%, we can conclude that our SVM classifier is better and more accurate than the ones in these papers. We can also say that, being our model 1.4% more effective than the best model in the reseach found, we were able to give a little contribution to this field providing a more effective and accurate SVM-based solution to problem of face detection.

B. Neural Network (NN)

Regarding the NN classifier, in the work present in [3] the authors stated an accuracy of 91.43% and in the study in [8] the authors stated an accuracy of 96.24%. Considering that were able to achieve an F1 Score, Precision, Recall an Accuracy of approximately 97.3%, although not a perfect result we were mostly satisfied with this outcome as we based some of our approach [3] and were able to outperform it. The approach described in the work present in [8] was based on a CNN and it was done with a much larger dataset than the one we used, so, we cannot compare our results with theirs.

Finally, based on some papers and online notebooks, we think that as future work to optimize and improve our NN-based application we could benefit from the usage of Fast Gabor Filtering as our feature extractor. Another possible improvement would be to use a combination between a skin-color segmentation algorithm (to choose candidate areas with the use of other masks on top of the skin-color segmentation) with the use of a candidate area re-scaling algorithm, to make possible to detect faces with sizes different than 27×18, as detailed in [7].

VIII. Conclusion

All in all, we are satisfied with the models we were able to develop. We also realized that, in some notion, we managed to contribute to improve the current state of the art of an SVM-based application on face detection and, even though, or NN-based classifier did not achieve what we initially expected we also managed to present an increment in accuracy and effectiveness comparing to the research found on this topic.

Finally, we can also conclude that our SVM-based application with HOG as a feature extractor is pretty viable to be used in real world scenarios where we need a model that is able to quickly detect faces, even if some times produces some false positive results.

Work Done By Each Student

Due to the restrictions of Covid-19 virus, we were not able to always develop the project together (in the same place or close to each other).

Therefore, João Vasconcelos was responsible for the implementation of Feature Extraction (HOG) and the Neural Network and Vasco Ramos was responsible for the topic of Data Augmentation and Support Vector Machines.

Overall, we consider the work was split equally between both members.

To summarize:

- João Vasconcelos (50%) - Feature Extraction (HOG), Train/test NN and Papper.
- Vasco Ramos (50%) - Data Augmentation, Train/test SVM and Papper.

References

[1] C. Shavers, R. Li, and G. Lebby, "An svm-based approach to face detection," in *2006 Proceeding of the Thirty-Eighth Southeastern Symposium on System Theory*, pp. 362–366, 2006.

[2] E. Osuna, R. Freund, and F. Girosit, "Training support vector machines: an application to face detection," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 130–136, 1997.

[3] H. F. Hashem, "Adaptive technique for human face detection using hsv color space and neural networks," in *2009 National Radio Science Conference*, pp. 1–7, 2009.

[4] V. Bhandiwad and B. Tekwani, "Face recognition and detection using neural networks," in *2017 International Conference on Trends in Electronics and Informatics (ICEI)*, pp. 879–882, 2017.

[5] K. Dang and S. Sharma, "Review and comparison of face detection algorithms," in *2017 7th International Conference on Cloud Computing, Data Science Engineering - Confluence*, pp. 629–633, 2017.

[6] B. Nassih, N. Hmina, and A. Amine, "Face classification under different kernel function compared to knn classifier," in *2016 13th International Conference on Computer Graphics, Imaging and Visualization (CGiV)*, pp. 232–236, 2016.

[7] J. Ruan and J. Yin, "Face detection based on facial features and linear support vector machines," in *2009 International Conference on Communication Software and Networks*, pp. 371–375, 2009.

[8] T. U. Ahmed, S. Hossain, M. S. Hossain, R. ul Islam, and K. Andersson, "Facial expression recognition using convolutional neural network with data augmentation," in *2019 Joint 8th International Conference on Informatics, Electronics Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision Pattern Recognition (icIVPR)*, pp. 336–341, 2019.

[9] Scikit Image contributors, "Histogram of Oriented Gradients." https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_hog.html#sphx-glr-auto-examples-features-detection-plot-hog-py. Accessed: 2020-06-10.

[10] Scikit contributors, "Multi-layer Perceptron." https://scikit-learn.org/stable/modules/neural_networks_supervised.html. Accessed: 2020-06-12.