Transfer Learning with intelligent training data selection for prediction of Alzheimer's Disease

Luca Polenta 1794787

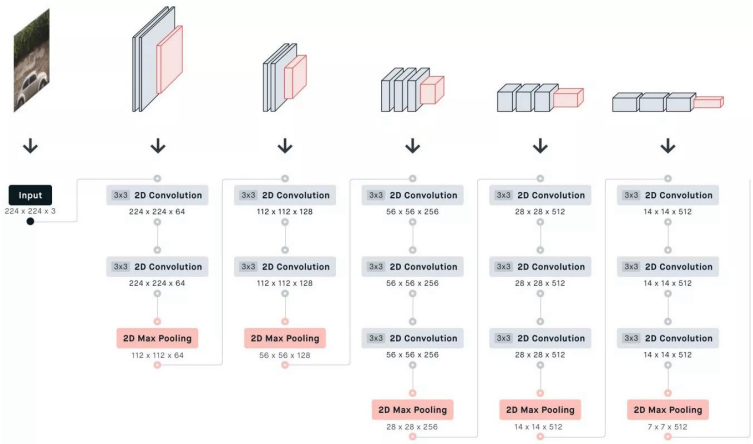
February 11, 2021

Abstract

In this project various implementations based on Transfer Learning have been developed to predict Alzheimer's disease. In addition, a series of features have been implemented into the code in order to improve the results. This project was developed by starting and expanding the paper cited in the references.

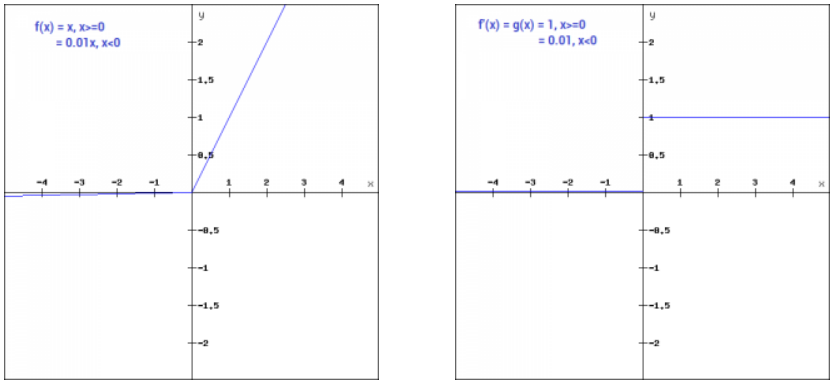
1 Preliminaries and Tools

In the field of machine learning there are two main ways in order to create a neural network: it is possible to code it from scratch or it is possible to use the so-called "transfer learning". This practice allows to reuse most of the parameters (weights) of a neural network already trained previously on a similar problem. The transfer learning methodology leads to a huge increase in performance as the first part of the network, which is usually responsible for extracting the features of the problem, starts from weights that are already partly close to the optimum of the problem and then it is only necessary to modify them slightly to adapt them to the specific problem under consideration. One of the most famous pre-trained neural network used to classify images is VGG16. This network has been designed to receive images that have a size of 224x224x3, while the current dataset is made up of images of size 150x150x3. An example of the first part of the network is portrayed in the next image:



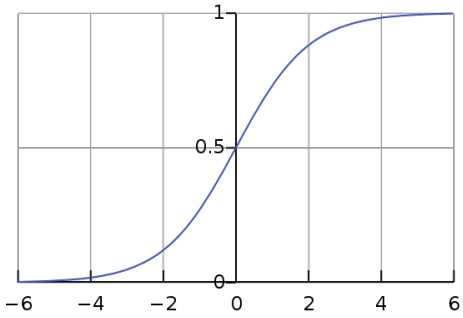
The network can be divided into 5 main blocks, the particularity of which are the very small convolutional filters. The size of the convolutionals is 3×3, the smallest to capture the notion of right/left, above/below, and center. Padding and stride are calibrated in such a way that the spatial resolution remains constant after the filter is applied. Then this network has one last section of layers fully connected, but in the current project they have not been included because they have been coded by hand. The activation functions used in these new layers are the "LeakyReLU". Its main feature is that it never completely deactivates the nodes, letting that even those useless to achieve the goal are trained so that they can make a positive contribution, if possible, in later steps of the training. It was selected as it led to the best results experimentally. Its mathematical formulation is the following:

$$f(x) = \begin{cases} 0.001 * x & \text{when } x \leq 0 \\ x & \text{when } x > 0 \end{cases} \qquad f'(x) = \begin{cases} 0.001 & \text{when } x \leq 0 \\ 1 & \text{when } x > 0 \end{cases}$$



Instead, the last Dense layer consists of a single node and it has the Sigmoid activation function in order to map the last logits between 0 and 1, thus performing binary classification. Its mathematical formulation is the following:

$$f(s_i) = \frac{1}{1+e^{-s_i}}$$



In addition, the "Binary Cross-Entropy" is used for the computation of the loss. It is often called "Sigmoid Cross-Entropy Loss" as it consists of a Sigmoid activation function added to the Cross-Entropy loss. The respective mathematical formulation is the following:

$$Cross - Entropy: \quad CE = -t_1 * log(f(s_1)) - (1 - t_1) * log(1 - f(s_1))$$

In the end, both "Adam" and "AdaGrad" were evaluated as the network optimizer. They minimize the cost function, identifying during training the optimized values to be assigned to weights related to each node. "Adam" was chosen because it fully combines the Momentum optimizer, characterized by a fast and excellent minimum search, and the RMSProp optimizer, which prevents excessive oscillations during such research. The corresponding mathematical formulations are the following:

$$
\begin{aligned}
\textit{For each value in } \omega_{\mathrm{j}}: \quad \omega_{\mathrm{t+1}} &= \omega_{\mathrm{t}} + \Delta\omega_{\mathrm{t}} \\
\Delta\omega_{\mathrm{t}} &= -\eta * \frac{\upsilon_{\mathrm{t}}}{\sqrt{s_{\mathrm{t}} + \epsilon}} * g_{\mathrm{t}} \\
\upsilon_{\mathrm{t}} &= \beta_1 * \upsilon_{\mathrm{t\text{-}1}} - (1 - \beta_1) * g_{\mathrm{t}} \\
s_{\mathrm{t}} &= \beta_2 * s_{\mathrm{t\text{-}1}} - (1 - \beta_2) * g^2{}_{\mathrm{t}}
\end{aligned}
$$

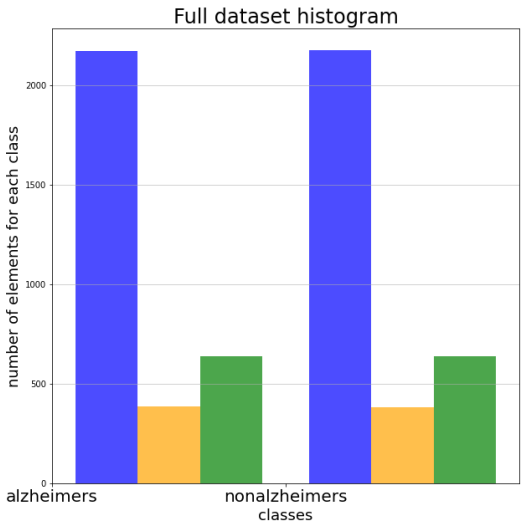
Where: η is the initial learning rate; υ_{t} is the exponential average gradient along ω_{j}; $\mathrm{s_t}$ is the quadratic exponential mean of the gradient along ω_{j}; $\mathrm{g_t}$ is the t-time gradient of ω_{j}; β_1 and β_2 are hyperparameters that allow to adjust the depence of Adam from the Momentum and RMSProp's approach; ϵ is a very small constant to avoid having divisions for zero.
Instead, "AdaGrad" was choosen because it uses an independent learning rate for each parameter. Intuitively, this method uses higher learning rates for parameters with a more sparse distribution, and slower for parameters with a denser distribution. The learning rate η is multiplied by a vector G, whose elements represent the scale factors for each parameter. This vector G is the diagonal of the matrix obtained as follows:

$$G = \sum_{\tau=1}^{t} g_\tau g_\tau^{\mathsf{T}}$$

where $g_\tau = \nabla Q_i(w)$ is the gradient at the iteration τ .

2 Selected Dataset and Pre-Processing

The dataset used is composed of two classes: images of brains with and without Alzheimer's. For each class there are 2560 samples available, of which 15% will be used to obtain the validation set. The latter set will be used during training to improve learning performance. There is also a second test folder which can be used at the end to understand how the training went. This folder is divided into two classes and there are 640 samples per class. Furthermore, it is also possible to view the histogram of the densities of each class and each subset (training, validation and test sets). The training set density is shown in blue, the validation set density in yellow and the test set density in green. In the end, given the scarcity of samples, augmentation has also been applied in some tests in order to check whether it can improve



the performance obtained. Considering that the images obtained from the slabs cannot vary much as patients are aligned in the machine, the augmentation performed on the dataset is very mild and concerns only the following values:

- shear_range: it applies angular distortion to slightly warp the image. It was set to 0.06;
- zoom_range: it zooms in on images. It could help focus the network on recognizing the central areas of the brain. It has been set to 0.06;
- horizontal_flip: it is set to true in order to create greater variability of samples;
- width_shift_range: it translates the image on the edges, causing the image itself to be cut horizontally. It was set to 0.06;
- height_shift_range: it moves the image vertically, causing the image itself to be cut on the edges. It was set to 0.06;
- fill_mode: it has been set to constant. This mode was the best because, once the previous distortions were performed, the outer areas are filled with black color such as the edges of the slabs.

Instead, the following augmentations have been disabled:

- vertical_flip: it is set to false because people are always subjected to the machine in the same way, so it is impossible to find brains reflected vertically;
- rotation_range: it has been disabled because there are no rotated plates. People are aligned to the slab machine

3 Method and Implementation

The main neural network under examination is "VGG16", as also recommended by the paper to which this project is inspired. Only the first part is taken from it so that it can be used to extract the most important features from the images. The second part has been designed ad hoc to guarantee maximum performance. The layer following this first part is a Flatten layer that allows to switch from the three-dimensional structure of the matrix in the previous layer to a one-dimensional vector that equally represents all the results obtained. Next there are two Dense layers of 512 and 256 respectively, interspersed with two Dropout Layers of 0.4 and 0.3 respectively. Dropout layers allow to completely disable nodes or some connections between them in such a way as to decrease network overfitting on training set. The activation functions are all LeakyReLU. Finally, the last layer is of type Dense and it is composed of only one node in order to

Model: "sequential"

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 4, 4, 512)	14714688
flatten (Flatten)	(None, 8192)	0
dense (Dense)	(None, 1024)	8389632
dropout (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 512)	524800
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 1)	513

Total params: 23,629,633
Trainable params: 23,629,633
Non-trainable params: 0

perform the binary classification. To do this it uses the Sigmoid activation function.
Then, in addition to the neural network mentioned above, it was decided to compare this network with a second pre-trained network. In this case, the "MobileNetv2" network was chosen. It fully combines efficiency and high accuracy in recognizing images. The network has been structured in a similar way to the previous one, but there are differences. After the Flatten layer that performs the same function mentioned above, there is a Dense layer of 1024 nodes followed by a Dropout layer set to 0.4 . Later there are two Dense layers of 512 and 64 nodes respectively. All of the Dense layers mentioned above have LeakyReLU as an activation function. At the end there is a Dense layer from a node with the sigmoid function as an activation function. The main difference from the previous one is that this network can download different weights with respect to the size of the input images. Supported sizes are 96x96, 128x128, 160x160, 192x192, and 224x224. If a supported size is not specified, weights of size 224x224 are downloaded. The images in the current dataset are 150x150, so they are resized to become 128x128 in size, thus downloading the most appropriate weights.

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenetv2_1.00_128 (Functi (None, 4, 4, 1280)        2257984
_________________________________________________________________
flatten_2 (Flatten)          (None, 20480)             0
_________________________________________________________________
dense_6 (Dense)              (None, 1024)              20972544
_________________________________________________________________
dropout_4 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_7 (Dense)              (None, 512)               524800
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_8 (Dense)              (None, 1)                 513
=================================================================
Total params: 23,755,841
Trainable params: 21,497,857
Non-trainable params: 2,257,984
_________________________________________________________________
```

In both networks, layers imported from pre-trained networks are set with trainable as True. This setting was adopted because it was recommended by the reference paper as it also allows the first part of the network to modify its parameters based on the input images, thus reporting better results. Furthermore, these statements will be tested in the next section.
In the end, early stopping was added to both networks. The original reference paper doesn't mention it, but training the networks too long on the same train images can provoke that the network overfits the training dataset thus losing the ability to recognize general images not present in the dataset. Early stopping controls the loss value in the prediction reported by the validation set at the end of each epoch. When this value tends to increase for too long and the value of the training loss tends to decrease, it means that the network is training to recognize more the features characteristics of the training set, instead making it more difficult to predict features from other sets. In order to ensure maximum performance on any image, it was decided to always keep this feature enabled.

4 Metrics

The metrics allow to evaluate the progress of training based on various factors. The metrics used and the data that they report are the following:

- **Precision:** this metric determines when the cost of False Positive is high. It is the ratio of correctly predicted positive observations to the total predicted positive observations. It can often lead to a forviant analysis as it is possible to obtain high precision values even when the network is not very optimal, so generally it is a useful tool to be compared to the others that will be carried out below. It is computed as follows:

$$\text{Precision} = \frac{\text{True Positive (TP)}}{\text{True Positive (TP)} + \text{False Positive (FP)}}$$

- **Recall:** it is the ratio of correctly predicted positive observations to the all observations in actual class. Recall is useful to clearly understand how high the cost of false negatives is. The formula is the following:

$$\text{Precision} = \frac{\text{True Positive (TP)}}{\text{True Positive (TP)} + \text{False Negative (FN)}}$$

- **F1-Score:** it is an overall measure of a model's accuracy that combines precision and recall. Having a good F1-Score means that you have few false positives and low false negatives, so you are classifying correctly without having trouble with noise. An F1-Score is perfect when it is equal to 1, while the model is a total failure when it is equal to 0. The respective formula is:

$$\text{F1-Score} = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} = \frac{\text{TP}}{\text{TP} + \frac{\text{FN} + \text{FP}}{2}}$$

5 Experiments and Results

In this test section, the solutions of the binaries classifications obtained from the two networks previously mentioned will be compared. Furthermore, these networks will be optimized and tested with various hyperparameters in order to find the combination that reports the highest, most accurate and reliable results. The tests can be divided as follows:

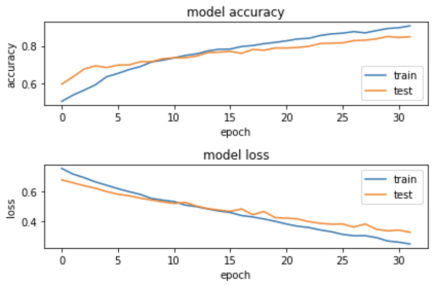
- Comparison of the results of the VGG16 network in the presence or not of augmentation when all the blocks of the pre-trained neural network are trainable;
- Comparison of the results obtained by the VGG16 pre-trained neural network when making some of its 5 blocks trainable and not;
- Comparison of results between the best VGG16 and MobileNetv2 with and without the augmentation;
- Comparison of results between the MobileNetv2 in the presence or not of augmentation and the best VGG16 by changing the optimizer.

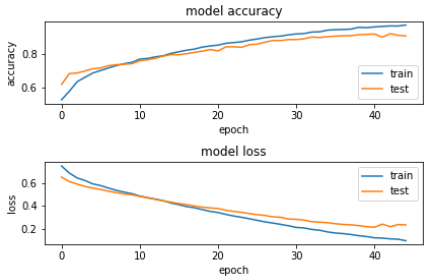
5.1 VGG16: augmentation test

This first comparison is focused on the differences in performance obtained with increase and not. The augmentation that is applied to the dataset is not excessive as we do not want to create an immense variability of it. The images available, in fact, are obtained from patients always positioned in the same way in a machine that does not give much freedom of movement. Therefore, we do not want to obtain variability, but we just want to help the network to study new situations similar to those already available in order to improve the results obtained. The performances obtained are the following:

WITHOUT AUGMENTATION	WITH AUGMENTATION

TRAINING PROGRESS



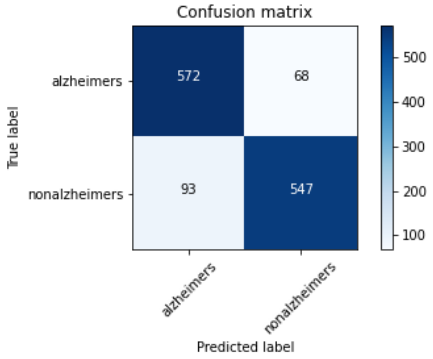


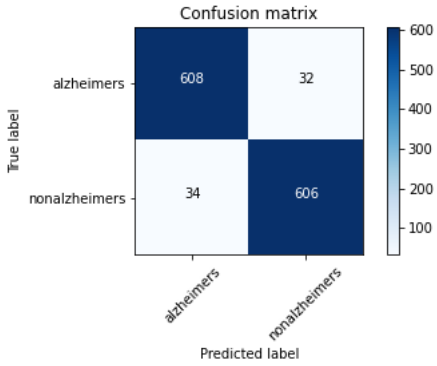
CLASSIFICATION REPORTS

	precision	recall	f1-score	support
alzheimers	0.86	0.89	0.88	640
nonalzheimers	0.89	0.85	0.87	640
accuracy			0.87	1280
macro avg	0.87	0.87	0.87	1280
weighted avg	0.87	0.87	0.87	1280

	precision	recall	f1-score	support
alzheimers	0.95	0.95	0.95	640
nonalzheimers	0.95	0.95	0.95	640
accuracy			0.95	1280
macro avg	0.95	0.95	0.95	1280
weighted avg	0.95	0.95	0.95	1280

CONFUSION MATRICES





As is evident, the training carried out using augmentation has led to better results without affecting the reliability of the results. In fact, there is not only an high value of the precision, but also recall and f1-score are at the same good level. This in fact means that the results obtained are also reliable, as can be seen from the related confusion matrix.
In addition, it is possible to note that even the early stopping was fundamental in order to avoid overfitting, as it finished the training earlier than the scheduled 75 epochs. In fact, the first training without augmentation ended at 32 epochs, while the one with augmentation ended at 45 epochs.
Finally, it is possible to notice how the trainings have had a good trend of continuous improvement without peaks of decrease. This factor is also encouraging, as it indicates good continuous training of the network without too many errors in the meantime.

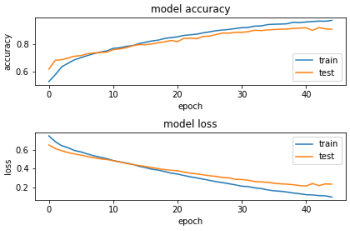
5.2 VGG16: tests on the trainability of weights in the blocks of the network

The tests in this section were carried out starting from a dataset to which it was applied augmentation, as it previously reported the best results. The tests carried out are 6: starting from the case in which all the blocks have all the weights that can be trained and then a new block is disabled at each test starting from the first to the last. The tests reported the following results:

TRAINABLE BLOCKS	CLASSIFICATION REPORTS	TRAINING PROGRESS

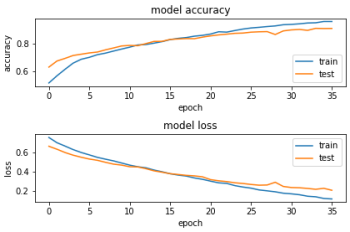
All blocks are trainable:

	precision	recall	f1-score	support
alzheimers	0.95	0.95	0.95	640
nonalzheimers	0.95	0.95	0.95	640
accuracy			0.95	1280
macro avg	0.95	0.95	0.95	1280
weighted avg	0.95	0.95	0.95	1280



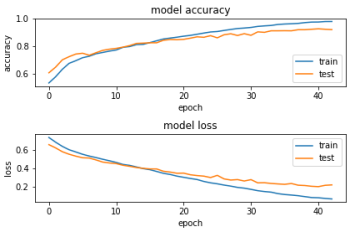
The first block is not trainable and the others are:

	precision	recall	f1-score	support
alzheimers	0.91	0.95	0.93	640
nonalzheimers	0.95	0.90	0.92	640
accuracy			0.93	1280
macro avg	0.93	0.93	0.93	1280
weighted avg	0.93	0.93	0.93	1280



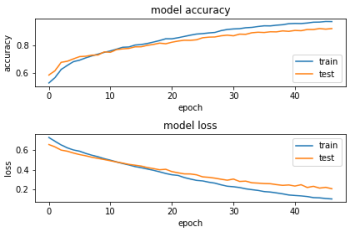
The first two blocks are not trainable and the others are:

	precision	recall	f1-score	support
alzheimers	0.95	0.94	0.94	640
nonalzheimers	0.94	0.95	0.94	640
accuracy			0.94	1280
macro avg	0.94	0.94	0.94	1280
weighted avg	0.94	0.94	0.94	1280



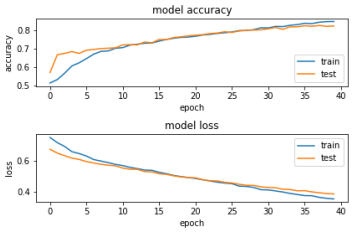
The first three blocks are not trainable and the others are:

	precision	recall	f1-score	support
alzheimers	0.93	0.96	0.94	640
nonalzheimers	0.95	0.93	0.94	640
accuracy			0.94	1280
macro avg	0.94	0.94	0.94	1280
weighted avg	0.94	0.94	0.94	1280



The first four blocks are not trainable and the last is trainable:

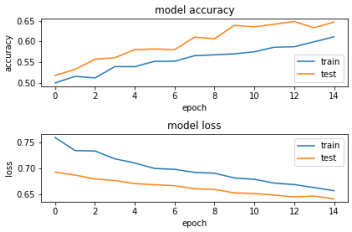
	precision	recall	f1-score	support
alzheimers	0.81	0.89	0.85	640
nonalzheimers	0.88	0.79	0.83	640
accuracy			0.84	1280
macro avg	0.84	0.84	0.84	1280
weighted avg	0.84	0.84	0.84	1280



TRAINABLE BLOCKS	CLASSIFICATION REPORTS	TRAINING PROGRESS

All blocks are not trainable:

	precision	recall	f1-score	support
alzheimers	0.67	0.58	0.63	640
nonalzheimers	0.63	0.72	0.67	640
accuracy			0.65	1280
macro avg	0.65	0.65	0.65	1280
weighted avg	0.65	0.65	0.65	1280

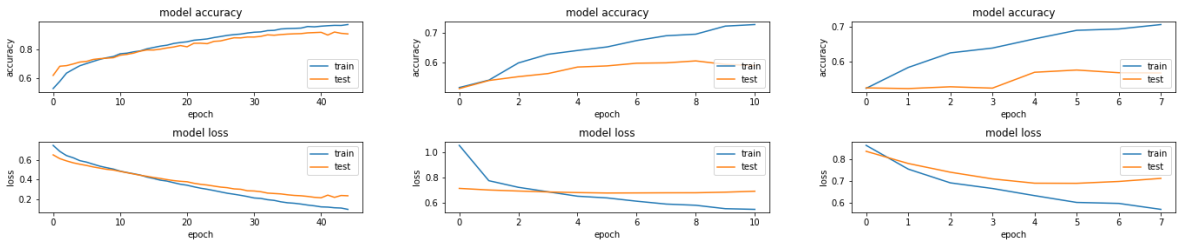


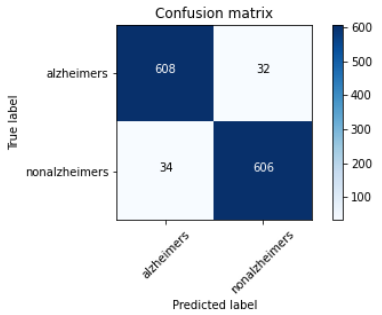
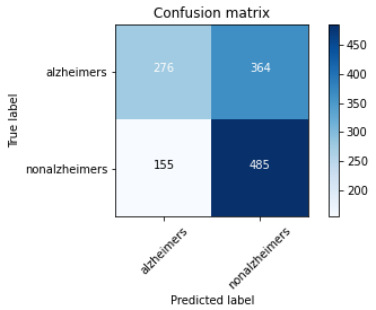
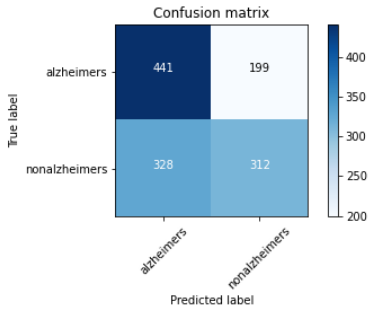
Looking closely at the results obtained, it is possible to denote how not training the weights of the first blocks has led to an increase in the number of epochs to train the neural network, but without having an increase in performance. This could mean that the neural network has more difficulty to learn how to recognize the images under consideration, moreover without getting better results. Only in one case better results have been achieved, but just for one class and, given the progress of all other similar cases, it could be a lucky initialization. In addition, by making more and more blocks not trainable, performance worsened and training began to take fewer epochs to finish.
Furthermore, even in these tests it is possible to see how early stopping has stopped the training in advance in order not to make the network overfitting the dataset under consideration. Therefore, the decision previously taken to insert it has once again proved to be an excellent feature.
Therefore, given the results obtained, it is possible to declare that the best case is the one in which all layers are kept trainingable.

5.3 Comparison between VGG16 and MobileNetv2

In this test the results of the most successful VGG16-based network were compared with two versions of the MobileNetv2 network: one version was trained with the dataset on which the augmentation had been applied and the other not. The results obtained are the following:

VGG16 WITH AUGMENTATION	MobileNetv2 WITHOUT AUGMENTATION	MobileNetv2 WITH AUGMENTATION
TRAINING PROGRESS		

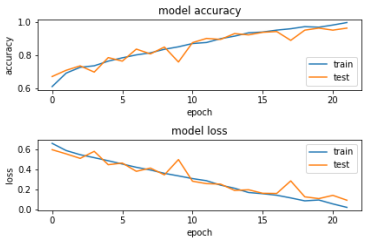
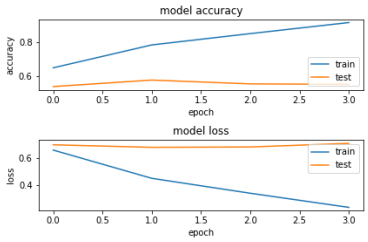
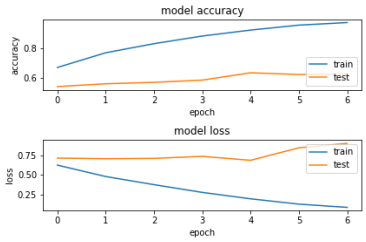
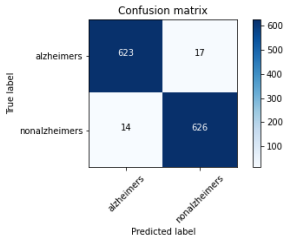
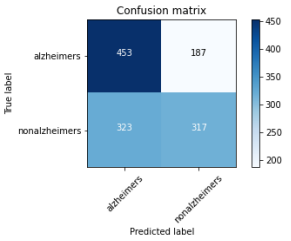
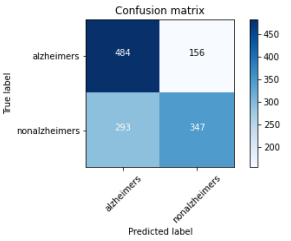


VGG16 WITH AUGMENTATION	MobileNetv2 WITHOUT AUGMENTATION	MobileNetv2 WITH AUGMENTATION
CLASSIFICATION REPORTS		
precision recall f1-score support alzheimers 0.95 0.95 0.95 640 nonalzheimers 0.95 0.95 0.95 640 accuracy 0.95 1280 macro avg 0.95 0.95 0.95 1280 weighted avg 0.95 0.95 0.95 1280	precision recall f1-score support alzheimers 0.64 0.43 0.52 640 nonalzheimers 0.57 0.76 0.65 640 accuracy 0.59 1280 macro avg 0.61 0.59 0.58 1280 weighted avg 0.61 0.59 0.58 1280	precision recall f1-score support alzheimers 0.57 0.69 0.63 640 nonalzheimers 0.61 0.49 0.54 640 accuracy 0.59 1280 macro avg 0.59 0.59 0.58 1280 weighted avg 0.59 0.59 0.58 1280
CONFUSION MATRICES		
		

The results show that the MobileNetv2 pre-trained neural networks don't achieve the same performance as the VGG16 pre-trained neural network. In the second test the augmentation was not applied to the dataset and the classification of images where Alzheimer is not present reported discrete results, while the classification of Alzheimer's reported bad results. In fact, the recall value is very low, so the number of correctly classified images compared to the total number that belong to that class is low, as can be seen from the confusion matrix. Instead, in the third test, augmentation is applied to the dataset and the image classification shows results that are the opposite with respect to the previous case. In this case, in fact, Alzheimer's images are recognized discreetly well, but the opposite are not.
Furthermore, even here it is possible to see how the early stopping prevented overfitting by stopping the training in advance with respect to the maximum number of iteration that was planned.

5.4 Comparing of the results by changing optimizers

A final test carried out involves the change of optimizer in order to try to improve the performance of the last 3 networks examined. In this test, the AdaGrad optimizer was used instead of the Adam optimizer. Previously, Adam had been set with a learning rate of 1e-6 as it was the value that reported the best results. Now AdaGrad does not necessarily require to specify an initial value of the learning rate and therefore it has been called without specifying any value. The results obtained are the following:

VGG16 WITH AUGMENTATION	MobileNetv2 WITHOUT AUGMENTATION	MobileNetv2 WITH AUGMENTATION
CLASSIFICATION REPORTS		
	precision recall f1-score support	
alzheimers 0.98 0.97 0.98 640	alzheimers 0.58 0.71 0.64 640	alzheimers 0.62 0.76 0.68 640
nonalzheimers 0.97 0.98 0.98 640	nonalzheimers 0.63 0.50 0.55 640	nonalzheimers 0.69 0.54 0.61 640
accuracy 0.98 1280	accuracy 0.60 1280	accuracy 0.65 1280
macro avg 0.98 0.98 0.98 1280	macro avg 0.61 0.60 0.60 1280	macro avg 0.66 0.65 0.65 1280
weighted avg 0.98 0.98 0.98 1280	weighted avg 0.61 0.60 0.60 1280	weighted avg 0.66 0.65 0.65 1280
TRAINING PROGRESS		
		
CONFUSION MATRICES		
		

The results obtained in the cases of the MobileNetv2 pre-trained network did not lead to any significant improvements. In fact, each of the classification reports show accurateness, recalls and f1-scores very similar to the previous case. The biggest difference is in the case of the MobileNetv2 network where the dataset had not been augmented. In fact, in it now the class of images belonging to Alzheimer's is recognized correctly, but the other suffered of a drastic decrease in performance. Instead, the MobileNetv2 network where the dataset had been augmented reported results in line with the previous ones: they are slightly better, but not enough to consider them an improvement. Moreover, for both configurations based on MobileNetv2 the training course ended very soon. This behavior seems to be a good thing, as it achieves the same performance as Adam which required many more epochs, but in reality it could mean that the training ran into one of the optimizer issues. In fact, one of them is the tendency of the method to cancel the learning rate during the iterations, leading to a learning arrest, since the terms of the sum are positive and the denominator value is strictly increasing as a function of the iterations. Therefore, given this possibility, it is not possible to decree this configuration as the best for networks based on MobileNetv2.
Instead, it is interesting to notice the case of the VGG16 network whose dataset had been augmentated. First of all, it can be seen that the training performance was more unstable

than when Adam optimizer is used: in fact there have been peaks where network performance has decreased during training, but these decreases have been recovered quickly and have not compromised the results obtained. Indeed, in this case the network reported even better results than in the previous case: the accuracy and the recall levels increased by a couple of points, but above all the F1-Score level reached almost perfect score. The F1-Score is computed using the harmonic mean of precision and recall, and it is the most reliable metric to understand the progress of training. Given the high scores, it can be confirmed that replacing the Adam optimizer with the AdaGrad optimizer has brought additional benefits to the network that had already behaved very well previously.

6 Conclusion

In conclusion, it is possible to denote how the pre-trained VGG16 network was the best in binary classification of Alzheimer's images. The combination of hyperparameters that led to the best result is the one in which the dataset has been augmented, all the layers of the model are trainable and the optimizer of the model is "AdaGrad".

References

[1] Naimul Mefraz Khan, Marcia Hon, Nabila Abraham, *Transfer Learning with intelligent training data selection for prediction of Alzheimer's Disease*, June 2019, available at: https://paperswithcode.com/paper/transfer-learning-with-intelligent-training

[2] tensorflow.org, *Transfer learning and fine-tuning*, available at: https://www.tensorflow.org/tutorials/images/transfer_learning

[3] keras.io, *Documentation of Keras*, available at: https://keras.io/api

[4] Gupta and Dishashree, *Fundamentals of deep learning-activation functions and when to use them*, 2017, available at: https://www.analyticsvidhya.com/blog/2017/10/fundamentals-deep-learning-activation-functions-when-to-use-them

[5] Gómez, Raúl, *Understanding categorical cross-entropy loss, binary cross-entropy loss, softmax loss, logistic loss, focal loss and all those confusing names*, 2018, available at: https://gombru.github.io/2018/05/23/cross_entropy_loss/

[6] Roan Gylberth, *An Introduction to AdaGrad*, 2018, available at: https://medium.com/konvergen/an-introduction-to-adagrad-f130ae871827

[7] Akshay L. Chandra, *Learning Parameters, Part 5: AdaGrad, RMSProp, and Adam*, 2019, available at: https://towardsdatascience.com/learning-parameters-part-5-65a2f3583f7d