Deep Learning Assignment #1

André Matos
MIEI — 55358
NOVA School of Science and Technology
Caparica
afr.matos@campus.fct.unl.pt

João Palma
MIEI — 55414
NOVA School of Science and Technology
Caparica
author2@campus.fct.unl.pt

Ruben Belo
MIEI — 55967
NOVA School of Science and Technology
Caparica
rc.belo@campus.fct.unl.pt

1 Section 1

In the multilayer perceptron we used a network with 4 layers, one of those of input, and another one of output with a softmax activation function. For this MLP we tested multiple configurations with different number of layers and neurons, but it didnt achieve any aceptable result.

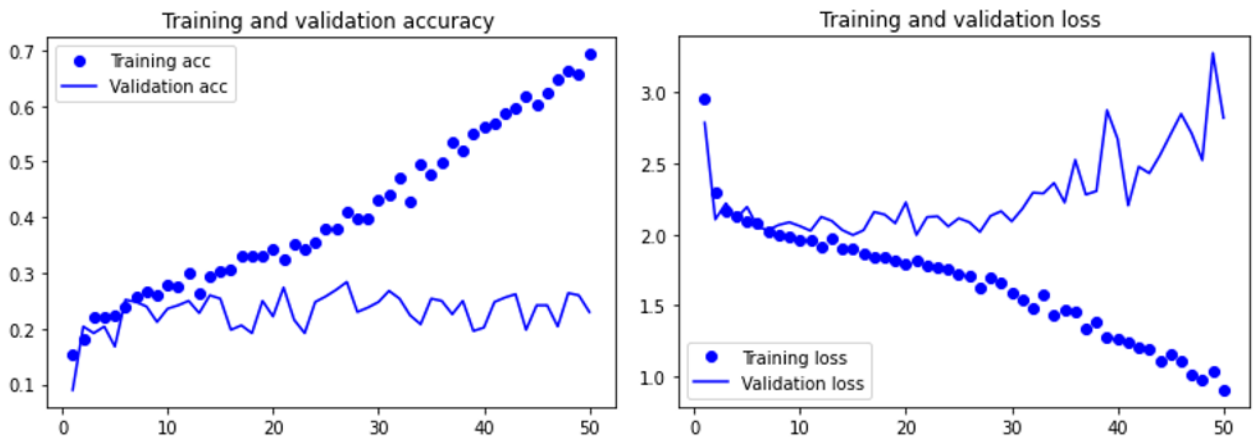


Figure 1: Loss and accuracy when training for the best parameters.

As we can see, using an MLP, the results are not acceptable, especially comparing to the next ones, using CNNs.

2 Section 2 - CNN implementation for classes

Regarding this task, we decided to test 2 CNNs and see which one provided better results.

For the first CNN(CNN 1), we used the same structure presented in the theoretical classes for the CIFAR10 example, and after some tests we arrived to the conclusion that we should reduce the number of the neurons from 64 to 32, and also we added some layers with Batch Normalization and a softmax activation function.

For the second CNN(CNN 2), we decided to go with a different approach. While in the first, for the convolutional layers, we started with 256 and we were reducing, in this second one we start with 16 and we go up until we reach 128, with some Batch Normalization layers, and for the hidden layers, we start with 512 units until we reach 32 units, where after we have an output layer with softmax as activation function. Regarding the loss, after some research, we saw that the appropriate one for this dataset would be Categorical CrossEntropy, because this one is ideal for multiclass classification problems.

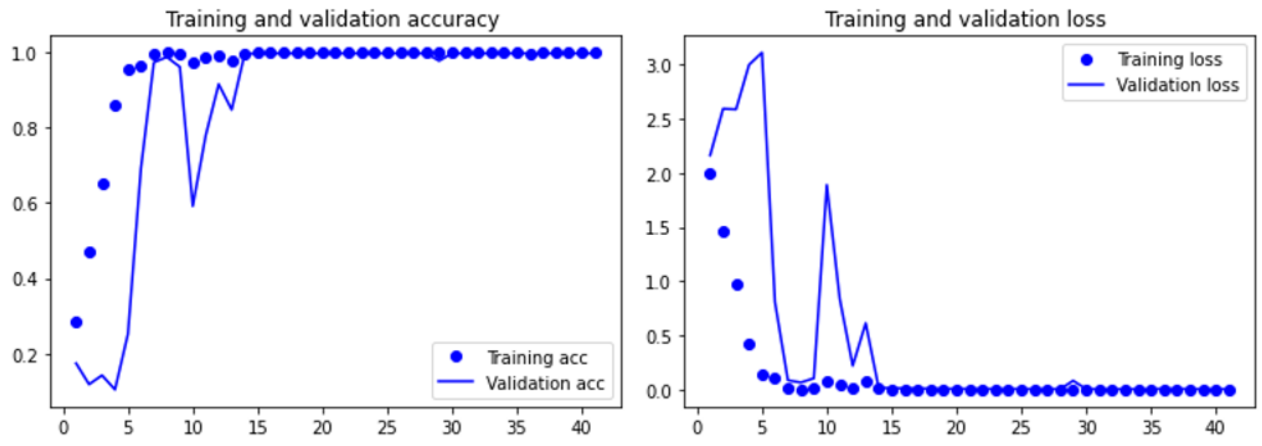
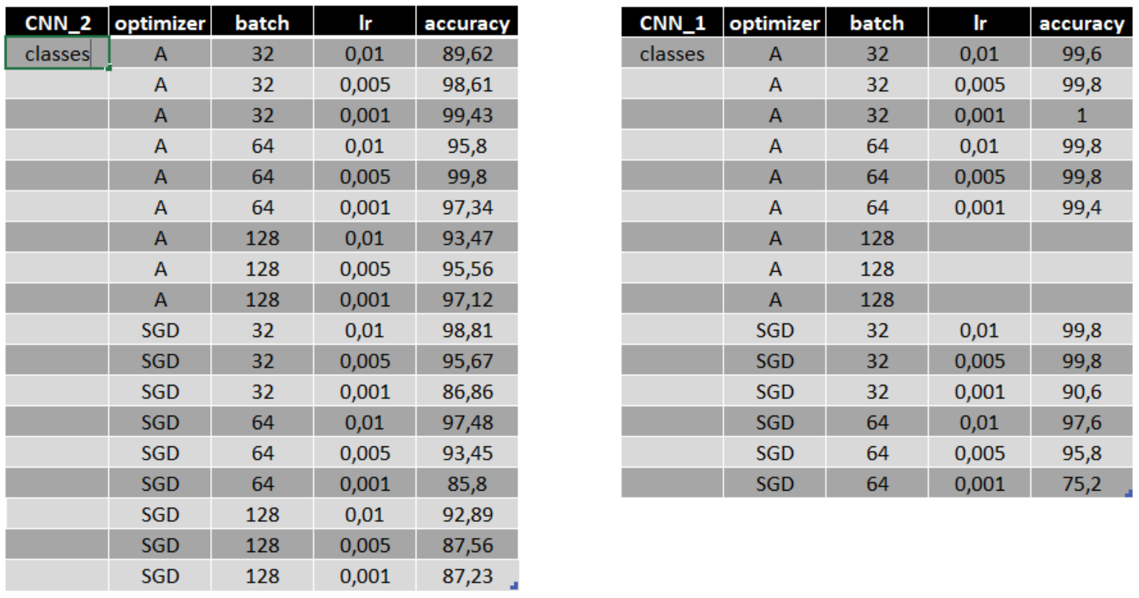


Figure 2: Loss and accuracy when training for the best parameters.



CNN_2	optimizer	batch	lr	accuracy
classes	A	32	0,01	89,62
	A	32	0,005	98,61
	A	32	0,001	99,43
	A	64	0,01	95,8
	A	64	0,005	99,8
	A	64	0,001	97,34
	A	128	0,01	93,47
	A	128	0,005	95,56
	A	128	0,001	97,12
	SGD	32	0,01	98,81
	SGD	32	0,005	95,67
	SGD	32	0,001	86,86
	SGD	64	0,01	97,48
	SGD	64	0,005	93,45
	SGD	64	0,001	85,8
	SGD	128	0,01	92,89
	SGD	128	0,005	87,56
	SGD	128	0,001	87,23

CNN_1	optimizer	batch	lr	accuracy
classes	A	32	0,01	99,6
	A	32	0,005	99,8
	A	32	0,001	1
	A	64	0,01	99,8
	A	64	0,005	99,8
	A	64	0,001	99,4
	A	128		
	A	128		
	A	128		
	SGD	32	0,01	99,8
	SGD	32	0,005	99,8
	SGD	32	0,001	90,6
	SGD	64	0,01	97,6
	SGD	64	0,005	95,8
	SGD	64	0,001	75,2

Figure 3: Accuracy on test for each parameter tested.

Even though it's not seen in these plots, we trained this model with 100 epochs, but since we have Early Stopping it only reaches 40 epochs. We would also like to refer that we used ReduceLrOnPlateau. These both funcionalities helped us, since we saw some improvements on the results when using these.

As we can see, we obtained near perfect results when using the CNN1, since it was the only one reaching an accuracy of 1, when using ADAM optimizer, a batch of 32 and a learning rate of 0.001. The other CNN didn't achieve this result. Also we can see that there was no overfitting since the validation values were close to the training values.

Ps: When traning with a batch size of 128 or more, in some networks the program would crash.

3 Section 3 - CNN implementation for labels

For this task, we made the choice to test with both CNN's used before, and add another one, with more hidden layers starting in 512 units and ending in 32 units, but with the difference that in all of these 3, for the output, we use a sigmoid activation function instead of a softmax. We wanted to experiment with this models because, after getting ideal results with the first CNN, we wanted to know what happened if we changed it. Since this problem is a multi label classification problem, the right loss to be used is the Binary Cross Entropy, because it can exists two types of classification active at the same time.

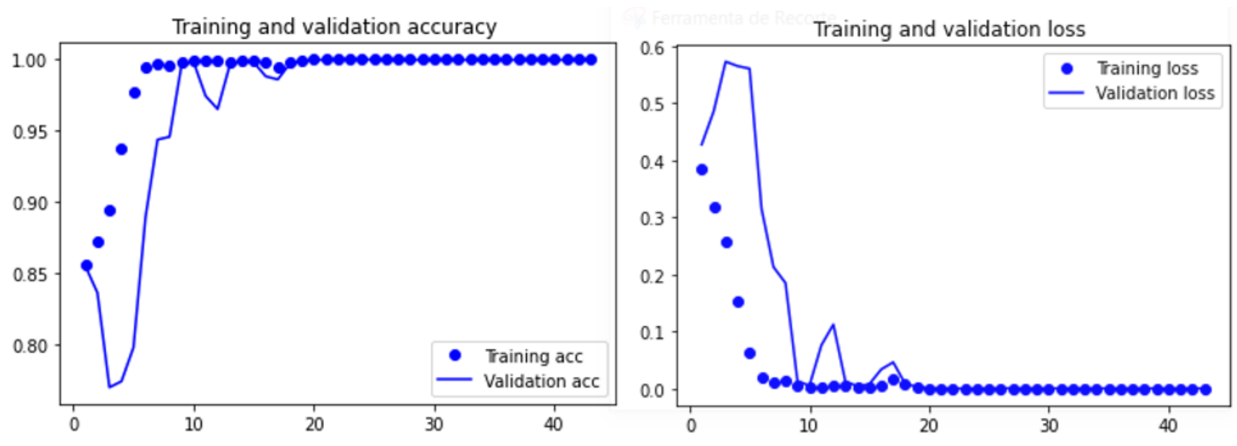
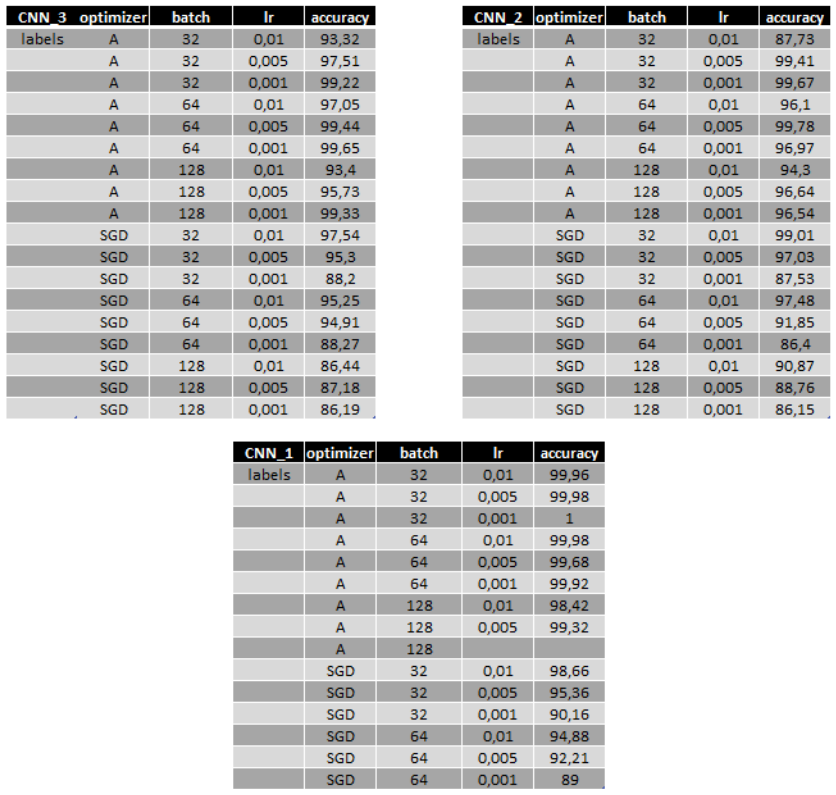


Figure 4: Loss and accuracy when training for the best parameters.



CNN_3	optimizer	batch	lr	accuracy
labels	A	32	0,01	93,32
	A	32	0,005	97,51
	A	32	0,001	99,22
	A	64	0,01	97,05
	A	64	0,005	99,44
	A	64	0,001	99,65
	A	128	0,01	93,4
	A	128	0,005	95,73
	A	128	0,001	99,33
	SGD	32	0,01	97,54
	SGD	32	0,005	95,3
	SGD	32	0,001	88,2
	SGD	64	0,01	95,25
	SGD	64	0,005	94,91
	SGD	64	0,001	88,27
	SGD	128	0,01	86,44
	SGD	128	0,005	87,18
	SGD	128	0,001	86,19

CNN_2	optimizer	batch	lr	accuracy
labels	A	32	0,01	87,73
	A	32	0,005	99,41
	A	32	0,001	99,67
	A	64	0,01	96,1
	A	64	0,005	99,78
	A	64	0,001	96,97
	A	128	0,01	94,3
	A	128	0,005	96,64
	A	128	0,001	96,54
	SGD	32	0,01	99,01
	SGD	32	0,005	97,03
	SGD	32	0,001	87,53
	SGD	64	0,01	97,48
	SGD	64	0,005	91,85
	SGD	64	0,001	86,4
	SGD	128	0,01	90,87
	SGD	128	0,005	88,76
	SGD	128	0,001	86,15

CNN_1	optimizer	batch	lr	accuracy
labels	A	32	0,01	99,96
	A	32	0,005	99,98
	A	32	0,001	1
	A	64	0,01	99,98
	A	64	0,005	99,68
	A	64	0,001	99,92
	A	128	0,01	98,42
	A	128	0,005	99,32
	A	128		
	SGD	32	0,01	98,66
	SGD	32	0,005	95,36
	SGD	32	0,001	90,16
	SGD	64	0,01	94,88
	SGD	64	0,005	92,21
	SGD	64	0,001	89

Figure 5: Accuracy on test for each parameter tested.

The conclusion of this section is no different than the last one. We used those same functionalities, and even though we created an additional CNN, the first one was the only one reaching an accuracy of 1 in the test, with the exact same parameters.

4 Section 4 - Pretrained Deep Networks

In this task, we tested the multiple pretrained networks provided by Keras, those being: ResNet50, MobileNet, VGG16, VGG19, MobileNetV2, Xception,ResNet152V2 and NASNetMobile, for both problems. Since this pretrained networks works as feature extraction, we then added multiple layers, with some Batch Normalization and Dropout, and the adequate activate function in the output for each problem. This also applies for the loss function.

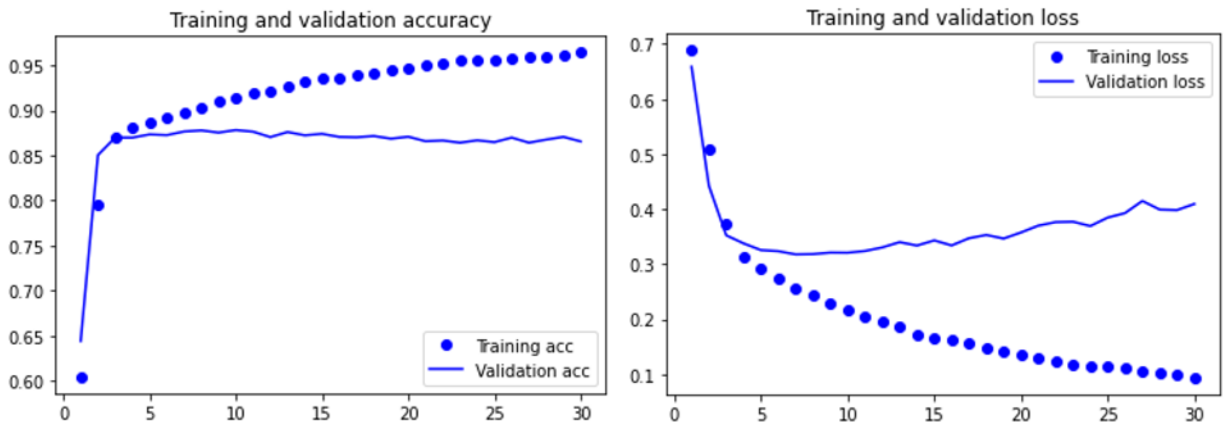


Figure 6: Loss and accuracy when training for the best parameters.

In this one, we noticed that it took way more time training this model, even when using smaller pretrained networks, and it didn't reach any results as good as the others ones, for both problems. These plots are for the labels problem, since it is the more complicated. Using the classes dataset, the results were better but, again, not as good as the others used in the previous sections.