Aprendizagem Automática Avançada

TP8 - Image Classification

João Romão - 55760, Pedro França - 55848

Introduction

The purpose of this exercise is to compare different configurations of a training model always using the same image dataset. We developed 2 functions, *model_1* and *model_2*, which both do the tutorial process (preprocess, train, and result display). The difference between these two functions is the addition of the augmentation and dropout preprocess present in model 2, which are absent in model 1. Additionally, model 1 used 10 epochs for training, while model 2 used 50, so we could observe if any overfitting occurred for a higher number of epochs in this model.

We first trained the models with the strategy used in the tutorial. We did this with the augmentation data and dropout preprocess (model 2) and without it as well (model 1). We call these trained models the reference ones and the goal is to later compare them with the models with altered parameterization. To summarize the most relevant characteristics of the models for this exercise, the different parameters are initially set (reference) as follows: batch size = 32, dropout rate (model 2 only) = 20%, loss function = SparseCategoricalCrossentropy, and batch normalization is absent (see Fig 1 for the results of these models).

Model 1 **Model 2**

Fig 1. Reference models.

Next we will summarize the parameters we will be changing on the four parts of this exercise, as well as a brief description of the influence of these parameters in the model. We will discuss the results at the end of the exercise.

1. Mean squared error as a loss function

On the standard models we were using SparseCategoricalCrossentropy loss function that computes the cross entropy loss between the labels and predictions of our model. The cross entropy loss function is used when we have labels to classify. The goal is to compare the reference results with those where the trained models use the mean squared error (MSE) as a loss function. The mean squared error tells us how close a computed regression is relatively to the set of points. It does this by taking the distances from the points to the regression line (these distances are the "errors") and squaring them. The squaring operation is necessary to remove any negative signs, which would “cancel” the positive values, and also gives more weight to the differences.

2. Dropout of 40%

As mentioned in the tutorial, a technique to reduce overfitting is to introduce Dropout to the network, which is a form of regularization. This technique forces the exclusion of a certain percentage of outputs of a layer (by setting the activation to zero) , resulting in randomly ignored neurons during the training phase. The standard model uses a 20% dropout value to reduce the overfit problem, and we changed this value to 40%, thus increasing the number of dropped out units from the applied layer. Since we are comparing the influence of changing the dropout value, we will only evaluate the results using the model 2 function.

3. Batch size of 8

As we discussed in a previous exercise in this course (TP4) , batch size is the number of samples that will be propagated through the network in one forward and backward pass. The higher the batch size, the more memory space we will need. Our standard model had a batch size of 32, and we are reducing it to 8 to compare the results.

4. Batch normalization

Batch normalization is a method that normalizes the input to a layer, re-centering and re-scaling it. This standardization approach usually has a stabilization effect on the learning process, reducing the number of training epochs required to train the deep network. In the exercise, we introduced batch normalization before Conv2D and Dense layers, with the exception of the first Conv2D layer, since its input has already been rescaled.

Exercise Results

1. Mean squared error as a loss function

Model 1

Model 2

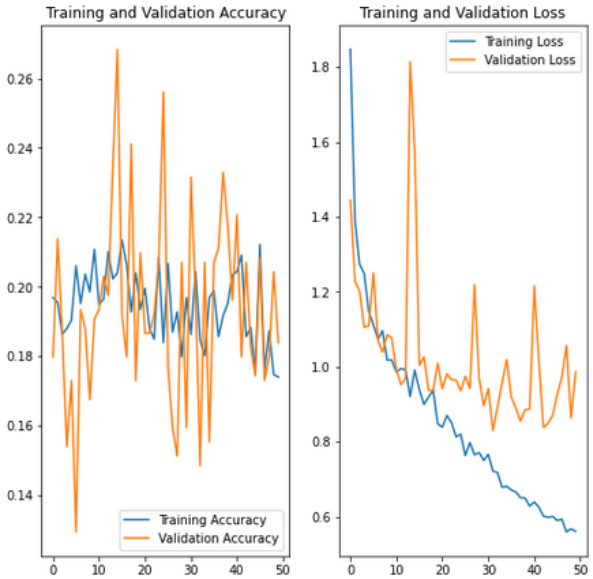

Fig 2. Models using mean squared error as loss function.

2. Dropout of 40%

Model 2

Fig 3. Models with a dropout of 40%.

3. Batch size of 8

Model 1

Model 2

Fig 4. Models with a batch size of 8.

4. Batch normalization

Model 1

Model 2

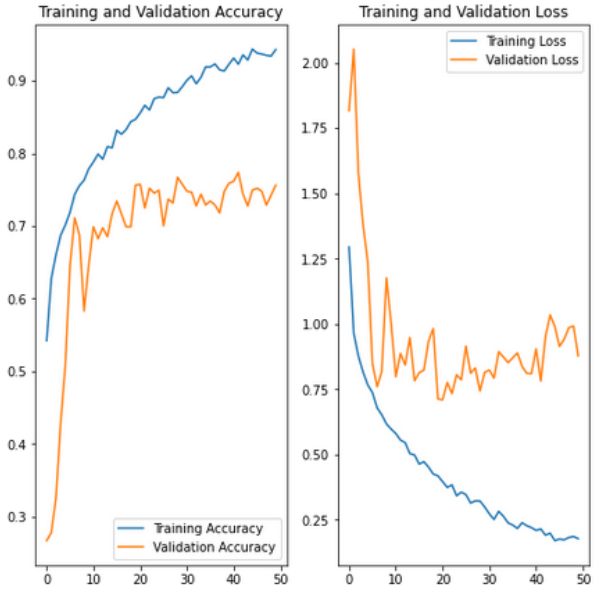

Fig 5. Models with batch normalization.

Discussion

Overall, each training epoch for the different models was complete in about 12-14 s/step for all models and exercises, except for exercise 3, where a reduction in batch size caused the increase of the time to about 15-17 s/step for both models. As for epochs until overfitting, considering only **model 1**, we observed that it occurred after 2 epochs for the reference scenario and exercise 3. Overfit was verified from the first epoch for exercise 4, and the results for exercise 1 were inconclusive, as using MSE as a loss function resulted in underfitting of the model. We also observed overfitting for **model 2**, which occurred around the 9th epoch for the reference and all exercises except for exercise 1, which continued to display underfitting. Overall, the inclusion of data augmentation and dropout in model 2 allowed for higher accuracies at the end of training, although overfitting was still verified (exception for model 1, as discussed).

We also compared the influence of changing the different parameters in the overall performance of the models in the exercise scenario with respect to the reference scenario. For **exercise 1** we observed a great degree of underfitting in both models, which did not display an overall monotonic evolution of the accuracy, but a rather erratic alteration in these values for both models, reaching low accuracy values for both training and validation data. Loss presented higher values than the reference, even when using 50 epochs for model 2. Overall we conclude that MSE is not an appropriate loss function for deep learning image classification algorithms (it is more appropriate for regressions), and it is usually better to use cross-entropy error to evaluate the quality of these types of neural networks.

For **exercise 2** we changed the dropout rate to 0.4, so we can only compare it with model 2 of the reference scenario. We could still observe overfitting starting around the 9^{th} epoch. The major alteration caused by the increased dropout rate was the reduction of the increase in loss after the early stopping point (i.e., apparent minimum of the validation loss curve) which occurred around epoch 15, with a loss of approximately 0.7 for both scenarios. In the reference scenario it continued to increase after this point, reaching a loss greater than 1.0 at the end of the 50^{th} epoch, while for exercise 2 it reached a loss of around 0.8.

For **exercise 3,** decreasing the batch size for either model caused no significant observable differences in accuracy, when compared to the reference. Overfitting occurred around epoch 2 for model 1 and epoch 10 for model 2 in both scenarios. As for the loss, we see that after the early stopping point, the loss increased more than in the reference, reaching values around 2.5 and 1.4 at the end of the training epochs for the validation loss curves of models 1 and 2, respectively. For comparison, the reference scenario had respective values of approximately 2 and 1.1.

Finally, for **exercise 4**, we observe overfitting starting at the first epoch in model 1 and the loss is also always higher for the validation set. Unlike previous cases, the accuracy and loss curves for each set are always far apart during the training of the model. In model 2 we can see that the accuracy for the validation set lagged behind the accuracy of the training set until epoch 9. However, after this epoch overfitting occurs and the scenario is similar to the reference model. Overall we can say that batch normalization decreased the performance of the reference model, which we did not expect to be the case since this strategy is normally used to increase model performance and allow faster

convergence. If we consider that convergence occurs around the early stopping point for the loss curve of the validation set, the results did not improve with this technique.