POLITECNICO
DI TORINO

Report Homework 2
Course: Machine learning and deep learning

Davide Bussone

16 May 2020

1 Data preparation

The dataset examined is "Caltech-101 dataset". It contains pictures of single objects belonging to a specific category as shown in figure 1.

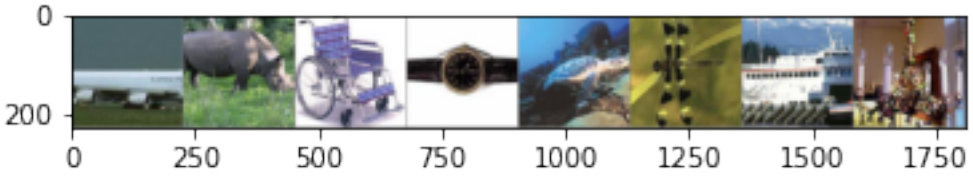

Figure 1: Examples of images to classify.

As the dataset's name suggests, categories are 101, without considering the "Background-Google", removed at the beginning of the case study.
Classes are not equally distributed, in fact there are some categories over-represented, as you can see in picture 2. This aspect can lead to achieve a weaker accuracy in the model.
Through the exploitation of a custom dataset, both the training and the test set were loaded and vectorized. Validation set was created by splitting the train dataset with a split size equal to 0.5.

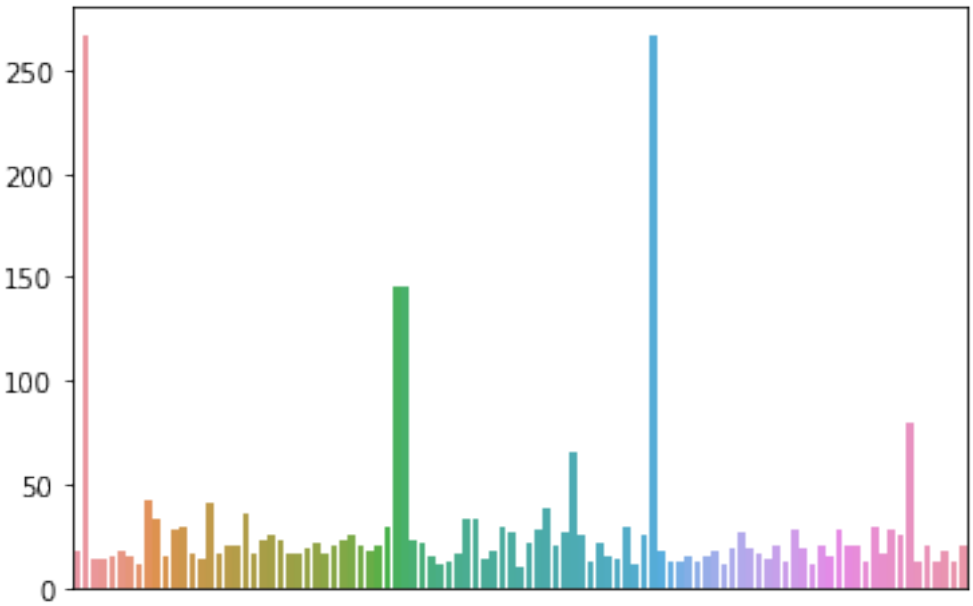

Figure 2: Classes distribution.

2 Train from scratch

The aim of the assignment is to train a convolutional neural network able to classify correctly as much images as possible. First of all, the network used is Alexnet. It contains five convolutional layers, each one followed by max pooling layers, and three fully connected layers. In addition, ReLU non-linearity is applied after all the layers. Alexnet has few layers, of course a net model with more layers and hidden units will increase the capability of representing more complicated functions.
In this section, the net is training by scratch, so hyperparameters were established manually, following different criteria.

Learning rate :
Perhaps, it is the most important parameter to tune. It manages to control the effective capacity of the model in a more complicated way compared to other hyperparameters. If its value is too large, gradient descent will increase rather than decrease the training error; If its value is too small, training will be slower and there will be a risk that training becomes permanently stuck with a high error. In table 1 is reported the optimization of this hyperparameter.

Learning rate	Loss	Accuracy
e^{-2}	3.85	20.88
e^{-3}	4.58	9.19

Table 1: Learning rate optimization

Number of epochs :
The choice was to fix the number of epochs to 20, an interval reasonable both to see effects in terms of loss and accuracy on the training-validation set and in terms of time execution.

Step size :
It indicates after how many epochs the learning rate will decrease. Table 2 represents its research.

Step size	Loss	Accuracy
5	4.26	9.37
10	3.78	23.44
15	3.25	26.31

Table 2: Step size research

Weight decay :
A coefficient which, if decreased, frees the model parameters to become larger, improving the capacity. In board 3 is shown the tuning of this hyperparameter.

Weight decay	Loss	Accuracy
$5 * 10^{-4}$	3.78	23.44
$5 * 10^{-5}$	3.85	20.88
$5 * 10^{-6}$	3.71	20.23

Table 3: Weight decay tuning

Momentum :
The optimizer used was SGD (stochastic gradient descent). Often, learning with it is very slow; method of momentum is designed on purpose to accelerate the process of learning. This method introduces a variable that represents the velocity, defined as the direction and speed at which the parameters move through parameters' space. α is momentum's parameter and it is set to 0.9. This means that the SGD's maximum velocity is multiplied by 10, following the relation (1).
In addition, Nesterov momentum was applied on the model. So, the gradient will be evaluated after the current velocity is applied.

$$v = \frac{1}{1-\alpha} \tag{1}$$

Gamma :
A multiplier of the learning rate step down. In table 4 is reported the investigation related to this hyperparameter.

Gamma	Loss	Accuracy
0.1	3.25	26.31
0.2	3.31	25.86
0.3	3.23	27.28

Table 4: Gamma analysis

Each experiment was realized by changing manually one hyperparameter, keeping the others with the same value.The goal is to catch the contribute brought by each analysed hyperparameter on the loss and accuracy of the model, during 20 epochs. Actually, changing manually the parameters is a sub-optimal solution, because maybe some possible combinations or values are excluded. On the other side, the application of research algorithm, such as grid search or random search, was infeasible, because both of computational efficiency and of time execution.
The best set found is applied on the test set (loss and accuracy are portrayed in graph 3). Board 5 reportes the score obtained.

Learning rate	Weight decay	Step-policy	Gamma	Test-Accuracy
e^{-2}	$5 * 10^{-4}$	15	0.3	25.30

Table 5: Best parameters on the test set

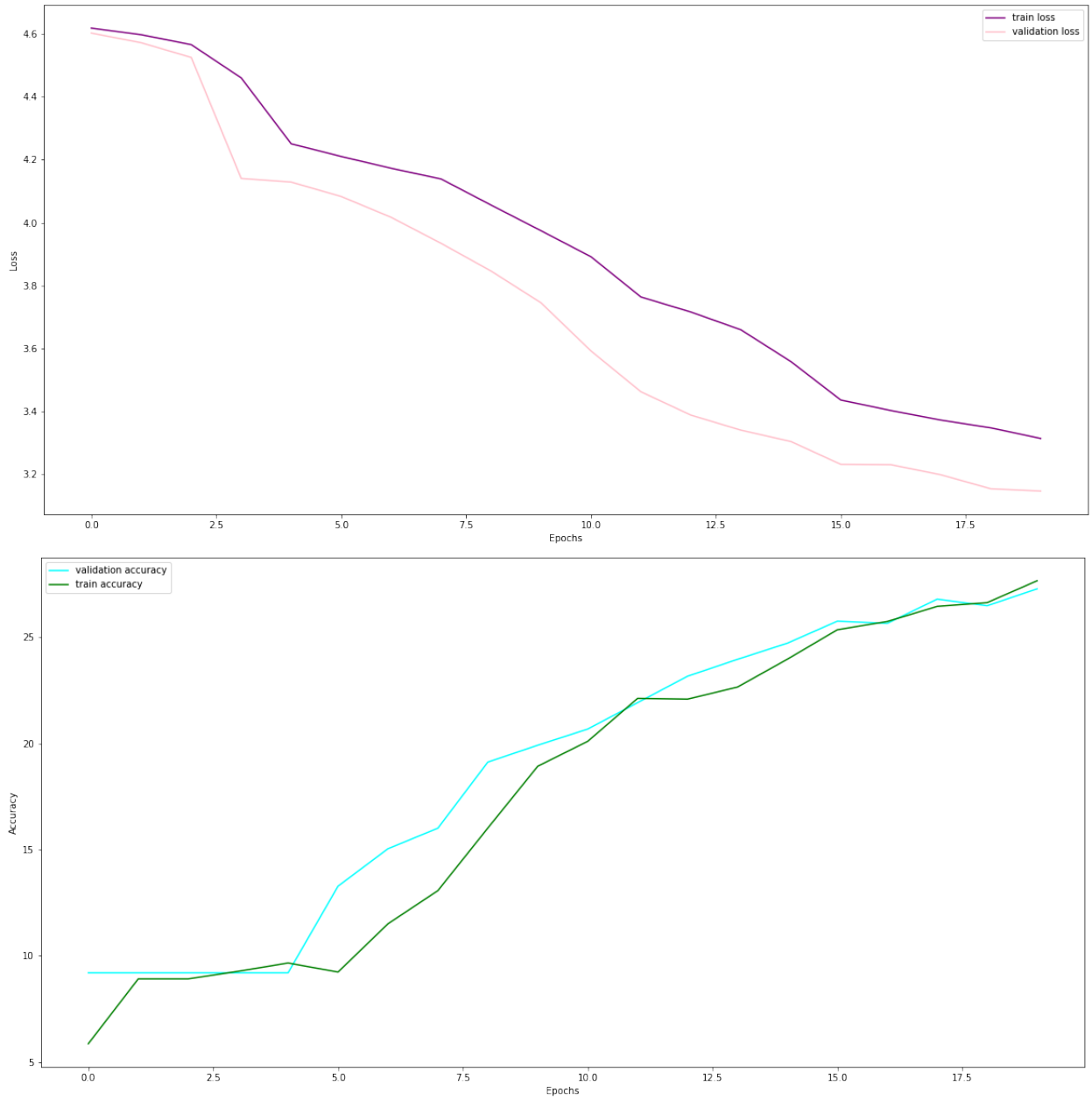

Figure 3: Loss and accuracy for the best set of hyperparameters.

3 Transfer learning

Transfer learning's method could be perceived as an improvement of learning in a new task, through the transfer of knowledge from a related task that has already been learned. To do that, it was used a pretrained model on a larger dataset: "ImageNet". This model was applied on the same

convolutional neural network, Alexnet.
Training, validation and test set were normalized by using the mean and standard deviation of ImageNet. The number of epochs was modified, because in transfer learning there are more data to train, as a consequence, a lower number of epochs is enough. For this reason, it was set to 15. The same hyperparameters of section 2 are optimized, the aim is to catch the improvement from previous experiments on both the validation and test set. Again the values are set manually. Tables 6, 7, 8, 9 show the results.

Learning rate	Loss	Accuracy
e^{-2}	0.11	82.26
e^{-3}	1.08	71.57
e^{-4}	3.22	30.77

Table 6: Learning rate optimization with transfer learning

Weight decay	Loss	Accuracy
$5 * 10^{-4}$	0.10	82.60
$5 * 10^{-5}$	0.11	82.26
$5 * 10^{-6}$	0.10	82.15

Table 7: Weight decay tuning with transfer learning

Step size	Loss	Accuracy
5	0.10	82.60
10	0.05	82.57

Table 8: Step size research with transfer learning

Gamma	Loss	Accuracy
0.1	0.05	82.57
0.2	0.04	82.43
0.3	0.05	82.19

Table 9: Gamma analysis with transfer learning

Then, the best set of hyperparameters (its loss and accuracy are displayed in lineplot 4) was applied on the test set. Board 10 shows the accuracy score.

Learning rate	Weight decay	Step-policy	Gamma	Test-Accuracy
e^{-2}	$5 * 10^{-6}$	10	0.2	83.30

Table 10: Best parameters on the test set

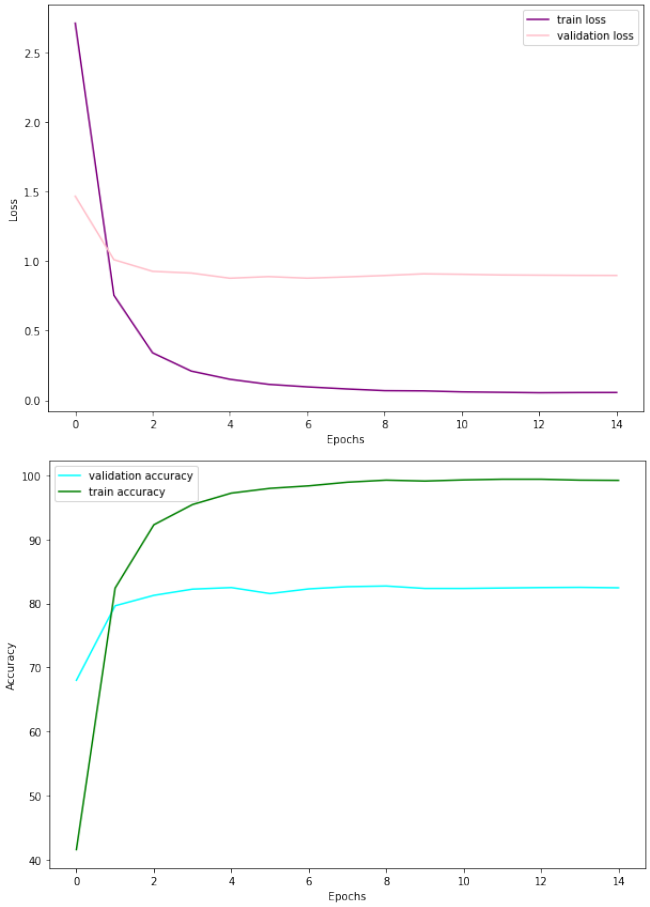

Figure 4: Loss and accuracy for the best set of hyperparameters.

4 Freezing layers

When you have a small dataset and a pretrained model with a very similar task compared to the case study, it could be useful to freeze part of the network. You might gain in terms of time and, sometimes, in terms of avoiding overfitting. Exploiting the best set of hyperparameters found in section 3, convolutional layers were frozen. Freezing consist of keeping the layers in their original form and then use their outputs to feed the unfrozen layers (in this case fully connected layers).
In picture 5 are plotted the loss and accuracy. Actually, the results were very close to the ones with all unfrozen layers, but the time of execution was inferior. Moreover, from the application on the test set, the accuracy increased as you can infer from table 11.

Learning rate	Weight decay	Step-policy	Gamma	Test-Accuracy
e^{-2}	$5 * 10^{-6}$	10	0.2	84.10

Table 11: Best parameters on the test set with frozen convolutional layers

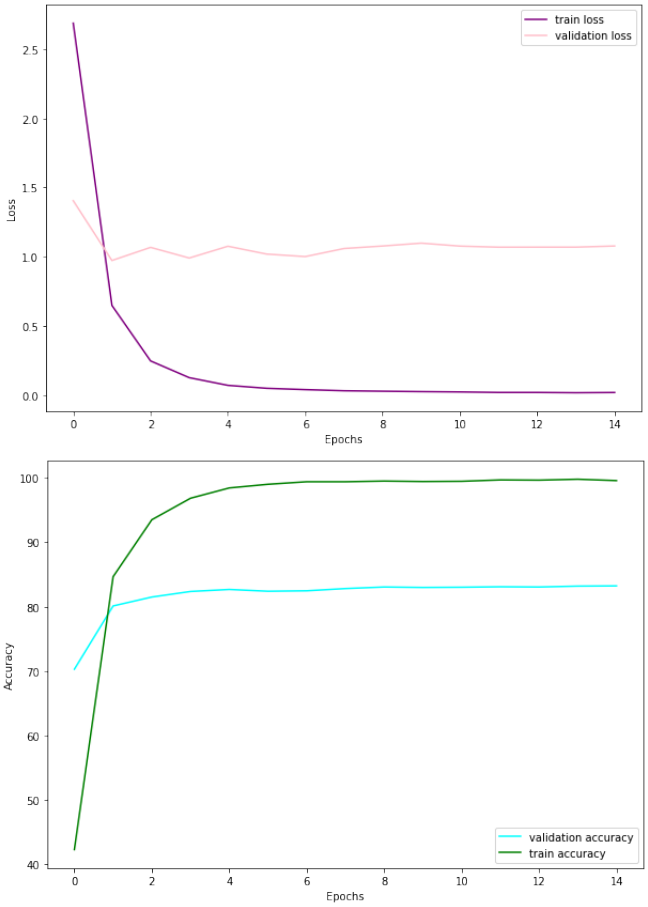

Figure 5: Loss and accuracy with frozen convolutional layers.

In addition, also fully connected layers were frozen, meanwhile convolutional layers were brought to their original state. So, the network was trained only on them.
In image 6 are portrayed the loss and the scores. The values computed on the test set are still very close to previous cases, as board 12 demonstrates.

Learning rate	Weight decay	Step-policy	Gamma	Test-Accuracy
e^{-2}	$5 * 10^{-6}$	10	0.2	84.65

Table 12: Best parameters on the test set with frozen fully connected layers

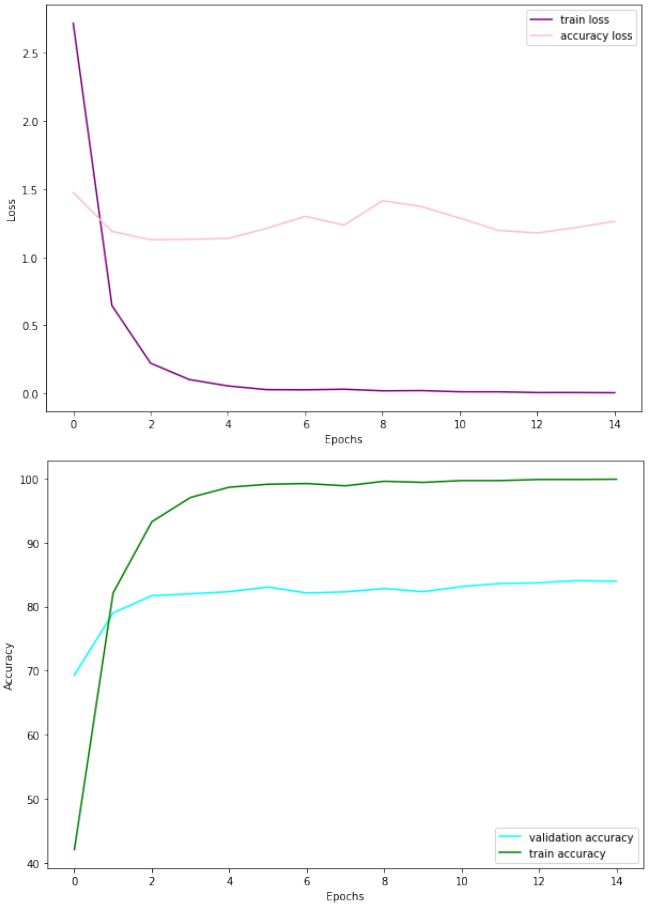

Figure 6: Loss and accuracy with frozen fully connected layers.

5 Data augmentation

Data augmentation is based on the fact that it is easy to improve the generalization of a classifier by increasing the size of the training set, exploiting the additional extra copies of the training examples which have been modified with transformations that do not change the task. Moreover, it is a technique addressed also to avoid overfitting. Three different variants of transformations are proposed. All of them were applied only on the training set, using Alexnet with ImageNet pretrained model and always with the same set of hyperparameters.

Color jitter :
The idea was to change randomly parameters like brightness, contrast, saturation and hue. All these features were increased, especially contrast and saturation. Contrast, because in photography helps to catch more details, useful in object recognition; saturation, because colours more saturated might facilitate the recognition of a specific category. In addition, also horizontal and vertical flips were applied. In figure 7, loss and accuracy are represented: the results get a bit worse in terms of accuracy on the test set compared to transfer learning without data augmentation, in board 13 you can catch it. On the other side, overfitting is reduced. This transformation slows down the time of execution of the fitting, in fact it lasts more than ten minutes.

Learning rate	Weight decay	Step-policy	Gamma	Test-Accuracy
e^{-2}	$5 * 10^{-6}$	10	0.2	82.03

Table 13: Best parameters of Alexnet with the first variant of data augmentation

Gray scale :
This transformation represents a conversion of a single image to a gray scale. The number of channels in input are fixed to 3, because input images are codified in RGB. As you can infer from picture 8, both loss and accuracy have a behaviour very closed to the one with only transfer learning on ImageNet. But the execution's speed, in training and in validating, increased. By applying the model on the test set, it is possible to denote from board 14 that accuracy decreases: the score stops to 81.71.

Learning rate	Weight decay	Step-policy	Gamma	Test-Accuracy
e^{-2}	$5 * 10^{-6}$	10	0.2	81.71

Table 14: Best parameters of Alexnet with the second variant of data augmentation

Geometric transformations :
Classifiers may benefit from random flips, both horizontal and vertical. In graph 9, loss and accuracy are reported. As you can infer, on the validation set, loss and accuracy were weaker compared to the ones computed with both the previous transformations. Despite that, table 15 attests that on the test set the results were better. Maybe, information on the grayscale or on color jitter's parameters do not help too much to generalize the model. In fact they are more accurated on the validation set rather than the test set.

Learning rate	Weight decay	Step-policy	Gamma	Test-Accuracy
e^{-2}	$5 * 10^{-6}$	10	0.2	83.40

Table 15: Best parameters of Alexnet with the third variant of data augmentation

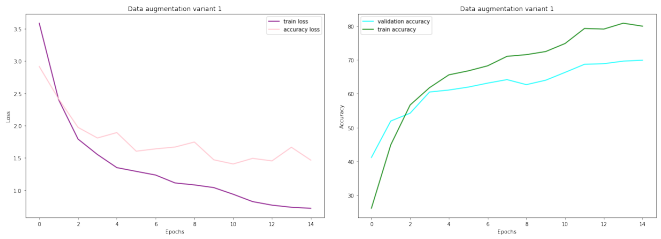

Figure 7: Loss and accuracy for the first transformation.

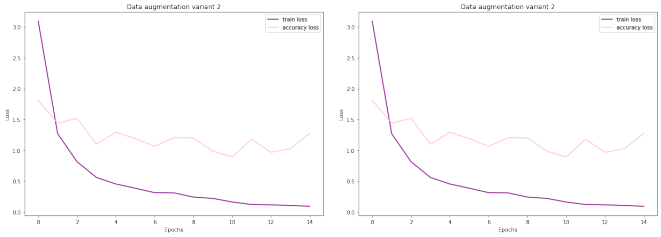

Figure 8: Loss and accuracy for the second transformation.

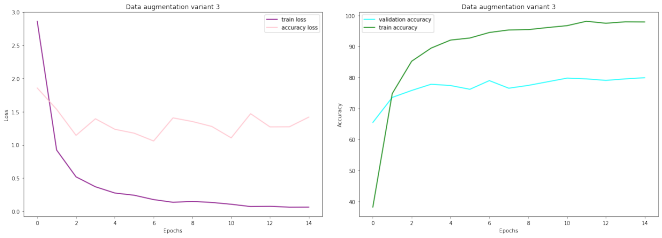

Figure 9: Loss and accuracy for the third transformation.

6 Beyond Alexnet

Resnet34 :
As said before, Alexnet is a small convolutional neural network. So, the experiment of transfer learning is repeated on bigger nets using best parameters of Alexnet. The first one is Resnet34. Generally, the family of Resnets let solve the the famous known vanishing gradient. In fact when the network is too deep, the gradients from where the loss function is calculated easily shrink to zero after several applications of the chain rule, this aspect could be infer from figure 10. ResNet34 's architecture consists on one convolution and pooling step followed by 4 layers which behave in a very similar way to the first one. Each layer follows the same pattern. It performs a 3x3 convolution

with a fixed feature map dimension [64, 128, 256, 512] , bypassing the input every 2 convolutions. Moreover, the width and height dimensions remain constant during the entire layer. Table 16 shows training details: the optimal values found in 3 were considered. The results on the test increased as you can infer from board 16, despite the reduction of the batch size to 64, for problems related to Cuda's memory.

Learning rate	Weight decay	Step-policy	Gamma	Test-Accuracy
e^{-2}	$5 * 10^{-6}$	10	0.2	88.32

Table 16: Best parameters of Alexnet tested on ResNet34

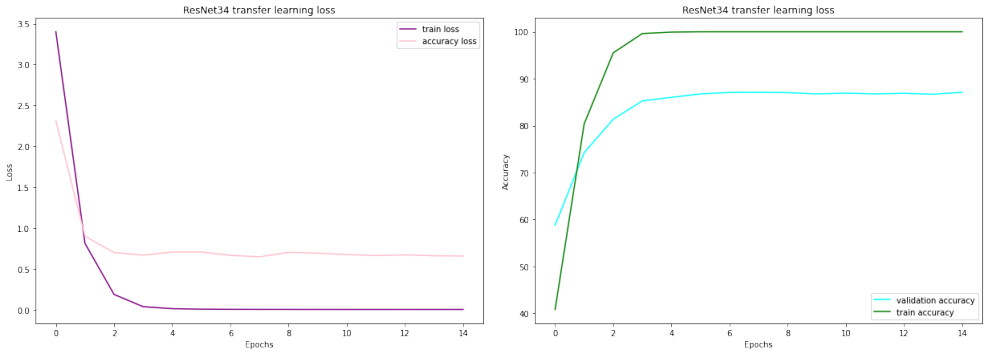

Figure 10: Loss and accuracy for Resnet34

Vgg16 :
The second network proposed is Vgg16. It makes an improvement over AlexNet by replacing large kernel-sized filters with multiple (3 X 3) kernel-sized filters one after another. Batch size was decreased to 32 and as a consequence also learning rate was decreased to e^{-3}, because of problems on Cuda's memory. Without decreasing learning rate, the loss exploded. For that reason, results were lower both on the validation set and on the test set, as shown in table 17. Loss and accuracy behaviour are portrayed in picture 11.

Learning rate	Weight decay	Step-policy	Gamma	Test-Accuracy
e^{-3}	$5 * 10^{-6}$	10	0.2	78.29

Table 17: Best parameters of Alexnet tested on Vgg16

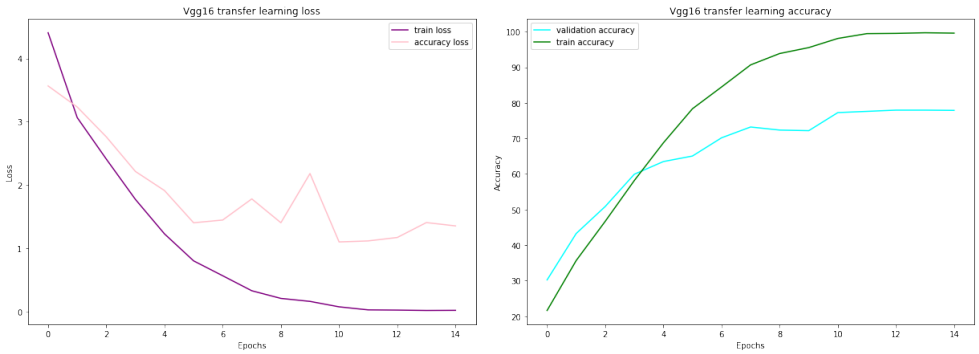

Figure 11: Loss and accuracy for Vgg16.

7 Conclusions

From the different trials, it could be observed that loading a convolutional neural network with a pretrained model,lead to a huge increment of accuracy and to a decrement of loss, also when some layers of Alexnet are freezed. Data augmentation may be a good tool to avoid overfitting, but in this case study, does not improve too much loss and accuracy, sometimes it could also lead to a worsening. Talking about nets, using bigger nets than Alexnet could lead to achieve better performance, as in Resnet34's case.