Paolo Alberto
Student ID: s278098

Machine Learning and Deep Learning
Politecnico di Torino

Homework 2 report

The purpose of homework 2 is to train a Convolutional Neural Network based on the Alexnet architecture, on the Caltech-101 dataset. The training was done mainly on a Nvidia GTX 1050 GPU, resorting to the help of Colab for the most computationally expensive parts of the assignment.

1 Data preparation

Taking as a reference the ImageFolder dataset, an implementation of the Caltech class is built, starting from the provided template. In this dataset the images are stored directly in memory as PIL images, and the "BACKGROUND_Google" category gets filtered out.
The provided splits are fed to this class in order to create a train and a test dataset. After this procedure the train_test_split function is applied to the indices of the training set, in order to split them into training and validation. The stratify option is used to make sure that each class is equally represented into both training and validation sets.

2 Training from scratch

The first run was performed on the untrained Alexnet model: a convolutional neural network composed of 5 convolutional layers and 3 fully connected ones. Initially the hyperparameters provided in the template were used, and training was performed, while keeping only the best performing model on the validation set.
After this first training phase, the test accuracy obtained on the validation set was around 12%. Experimenting with different hyperparameters can lead to improvements, but even with the best combination the accuracy never exceeded 60%. The results of the best models obtained with this optimization process are shown in the table below.

Hyperparameters					Validation accuracy	Test accuracy
LR	Epochs	Step size	Gamma	Batch size		
0.001	30	20	0.1	256	12.2%	
0.01	30	20	0.1	256	29.4%	
0.01	30	20	0.05	256	30.9%	
0.01	40	30	0.1	256	31.2%	
0.01	40	30	0.1	128	55.2%	
0.01	40	30	0.1	64	58.3%	57.8%
0.01	40	30	0.05	64	57.2%	

Table 1: Hyperparameter optimization with Alexnet model trained from scratch

The optimization process was mainly done on learning rate, epochs and batch size. Also some small tweaks were applied to the parameters related to the step-down policy (step size and gamma).
The learning rate was increased in order to make the model converge faster, and changing this metric alone proved to double the accuracy on the validation set.
After the learning rate optimization, the number of epochs was raised in order to increase the rounds of optimization applied during the training phase. The batch size was decreased, so that the model could start to learn features before seeing all of the available data. This is especially true because of the small dataset used.

While a great increase in accuracy was obtained with hyperparameter optimization, there is still a lot of room for improvement with this classifier: we will see how the same model, pre-trained on a much larger dataset, can provide a great increase on performance.

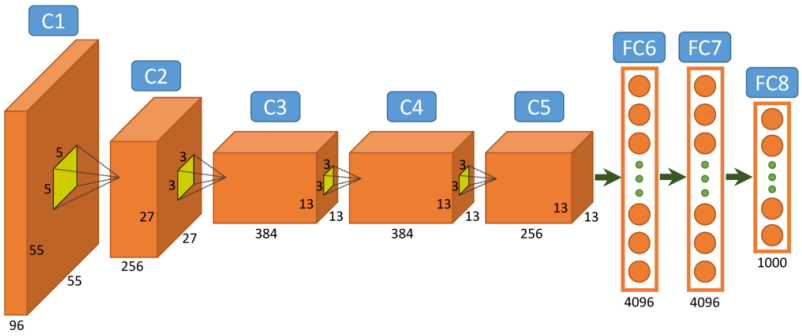

Figure 1: AlexNet schema

We can have interesting insights by plotting the graphs of the loss for the initial model and for the one with the highest accuracy, obtained after the hyperparameter optimization: we can observe that, while the optimization of the parameters was successful in increasing the accuracy, it created overfitting on the validation set. Behind this results, there is the fact that the network, after some epochs, has learned too well the features of the training set.

This overfitting effect underlines the importance of having a very large dataset in a visual recognition task of this kind. Making the most of the available pre-trained models is crucial in deep learning. Because of that, in the next section we will repeat the experiments with the pre-trained AlexNet model.

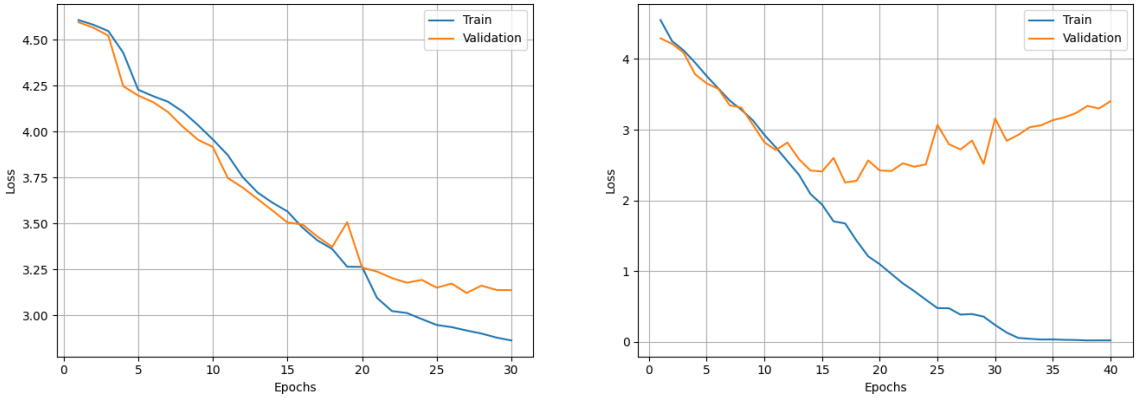

Figure 2: Loss graphs before and after hyperparameter optimization

3 Transfer learning

The model used is the same as before, but now we load the weights trained on the ImageNet dataset: a collection of over 14 millions images. After changing the normalize function in order to use Imagenet's mean and standard deviation, the model is trained again with the default parameters (LR = 0.001, 30 epochs and step size = 20). A significant improvement is observed, with an accuracy of more than 80% on the test set.

Various experiments are then performed on the hyperparameters. The learning rate is increased in order to make the model capable of learning the features of the dataset in a faster way. This time, however, increasing the LR by a factor of 10 didn't prove to be as beneficial as before. This could be explained by the fact that the pre-trained model has already learned a great number of features beforehand. Decreasing the batch size proves to be beneficial for the validation accuracy, as stated in the previous section.

In transfer learning, we can also experiment by freezing some layers during training, by setting the requires_grad parameter of the layer weight to false. Using the best performing hyperparameters found earlier, we notice that training only the fully connected layers doesn't seem to affect the final result too much, with accuracy changing from 0.849 to 0.84. On the other hand, when training only on the convolutional layers, the model becomes significantly worse, with an accuracy of around 0.6.

Hyperparameters					Validation accuracy	Test accuracy
LR	Epochs	Step size	Gamma	Batch size		
0.001	30	20	0.1	256	81.6%	
0.01	30	20	0.1	256	82.8%	
0.01	30	20	0.1	64	83.2%	
0.005	30	20	0.1	64	83.6%	
0.005	40	30	0.1	64	84.9%	84.6%
0.005	40	30	0.05	64	83.8%	
Training only the classifier						
0.005	40	30	0.1	64	84%	83.4%
Training only the features						
0.005	40	30	0.1	64	60.1%	60.5%

Table 2: Hyperparameter optimization with pre-trained Alexnet model

4 Data augmentation

In order to artificially augment the dataset size, in literature it is considered good practice to apply transformations, with the goal of creating new images, while preserving the class. In addition to the preprocessing transformations used until now, 3 different kinds of transformations are applied to the images:

- Color Jittering

  Changing the brightness, saturation and contrast of an image can be useful when working with images taken in different settings and with different cameras.

- Random horizontal flip with a probability of 0.5

  Can be useful to increase the size of the dataset, since the images in our dataset can be flipped without losing their class.

- Random rotation in the (-15°, +15°) range

  Useful to simulate photos taken from slightly different angles and point of view.

Transformations	**Accuracy**
Resize(256) Centercrop(224)	84.9%
Resize(256) Centercrop(224) Jittering	84.2%
Resize(256) Centercrop(224) RandomHorizontalFlip(0.5)	83.4%
Resize(256) Centercrop(224) RandomRotation(-15, +15)	82.3%
Resize(256) Centercrop(224) Jittering RandomHorizontalFlip(0.5) RandomRotation (-15, +15)	81.8%

Table 3: Data augmentation results

The random rotations were the ones that created the greater drop in accuracy. This was particularly true when increasing the degrees of rotation to more than 25°. With this kind of transformation alone the accuracy plummets to about 60%.

5 Beyond Alexnet

The AlexNet CNN can be considered a pretty naive model, compared to deeper models like ResNets or VGGs. In this section we explore the use of this larger and more accurate models. All of the networks used for this task were pre-trained on the standard ImageNet Dataset.

VGG16 is a popular convolutional neural network model that improves on the AlexNet structure by increasing the number of convolutional layers and replacing large kernel-sized filters with smaller ones (3x3, which is the smallest size to capture the notion of left/right, up/down and center). An implementation of the pre trained VGG16 model, without any changes to the hyperparameters, shows the power of this network, with an accuracy of above 87%.

Of the networks that were implemented, the best performing one is the pre-trained ResNet34 network. This model is composed of 34 layers and about 21 million parameters. The final accuracy on the test set is 93%.
All of these models were trained with a learning rate of 0.005, 40 epochs and a batch size of 64.
In general, it is safe to say that the more complex models perform a better classification.

Model	**Val Accuracy**	**Test Accuracy**
AlexNet	84.9%	84.6%
VGG16	87.3%	87.5%
ResNet34	93.2%	93%

Table 4: Data augmentation results