POLITECNICO DI TORINO

Corso di Laurea Magistrale
in Data Science and Engineering

Machine Learning and Deep Learning Report
Exam session: Winter 2021

Image recognition on Caltech-101 using different CNN models and techniques

Emanuele Fasce

1. INTRODUCTION

In this report several Convolutional Neural Networks (CNN) are implemented and used for classification on the Caltech-101 dataset using Pytorch.
The starting point of this work is the AlexNet network.
After a first step in which AlexNet is trained from scratch, some useful techniques are exploited, like pretraining the network on the ImageNet dataset, and the data augmentation technique.
In the last stage of this report, other CNN models are used.
Hyperparameters are modified manually at each run. This choice is preferred due to hardware constraints and due to the need of readjusting the hyper-parameters by analyzing the results of preceding trainings.
While all the results are reported in tables, the training graphs are available at the following link:

2. DATASET AND PREPROCESSING

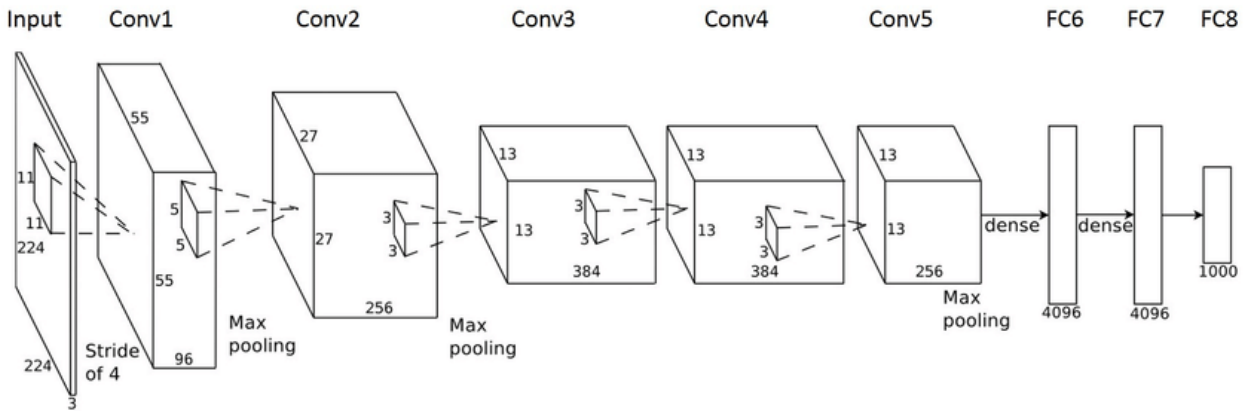
The dataset Caltech-101 is composed of 5784 labeled and 2893 unlabeled pictures of 101 different categories of objects selected from the ImageNet dataset. Original images (300x200 pixels) are resized, cropped in the center, and transformed into RGB tensors with dimensions 3x224x224. Two different normalizations are used with the pretrained and non-pretrained network since the mean and the standard variation of the pictures in the Imagenet dataset are different from the mean and standard variation of pictures in Caltech-101 dataset. Half of the labeled set is split into the training set and half into the validation set, in such a way that half samples of each class belongs to the training set and the other half to the validation set. This is obtained by the parameter *stratify* in the *train_test_split* function or also by exploiting the fact that in the file that contains the labeled dataset classes are already well divided, so it is enough to put the picture described in a row in the training set and the picture described in the following row in the validation set. The unlabeled set is used as test set.

3. ALEXNET

A CNN is a neural network in which there is at least a convolutional layer.
There are several reasons why this type of network is much more effective than traditional neural networks when dealing with image recognition. Some of these reasons are discussed below.

Firstly, it has sparse connections instead of fully connected connections, and this leads to reduced parameters. This is very useful when dealing with high dimensional data like pictures.
Secondly, the convolution operation provides translational equivariance: the position of the object in the image should not be fixed in order for it to be detected by the CNN.
Thirdly, CNNs use a very important concept of subsampling or pooling in which the most prominent pixels are propagated to the next layer dropping the rest. This provides a fixed size output matrix and a reduction in parameters without losing important information.

AlexNet is an extremely popular CNN used in computer vision. It was proposed by Hinton et Al. in one of the most influential papers published in the field of computer vision. It became the first Deep CNN based large database image recognition winner in 2012.
As shown below, it contains eight layers; the first five are convolutional layers, some of them followed by max-pooling layers, and the last three are fully connected layers.



4. EXPERIMENTS WITH ALEXNET

It can be noticed that importing the Alexnet network is very easy on Pytorch but this model has to be modified. Since the Caltech-101 dataset contains only 101 classes while Imagenet contains 1000 classes, the last fully connected layer of Alexnet is modified with *out_features=101.*

The Alexnet network is trained with different combinations of hyperparameters and evaluated to see which combination gives the best accuracy on the validation set. The winner combination is used to fit the model on the test set. The main hyperparameters used in this experiment are explained below.

The **number of epochs** indicates the number of passes of the entire training dataset the machine learning algorithm has completed.
The **batch size** defines the number of samples to work through before updating the internal model parameters.
The **learning rate** determines the step size at each iteration while moving toward the minimum of the loss function.
Weight decay is an hyperparameter used also in SGD to find a local optimum with small-magnitude parameters, while **momentum** is used to diminish the fluctuations in weight changes over consecutive iterations. The learning rate is multiplied by the **gamma** parameter after the number of epochs indicated by the **step size** parameter.

Optimizers in machine learning are used to tune the parameters of a neural network in order to minimize the cost function. Different optimizers are used in this experiment.
It can be noticed that each optimizer works better with different learning rates. For example, using SGD with a LR=1e-2 leads to suboptimal results, so the learning rate is kept lower in the other trials, and the best accuracy on the validation set is found with LR=1e-3.
The Adam and AdamW optimizers in this case do not work well with LR=1e-3, so training is performed with LR=1e-4. The Adadelta optimizer instead allows LR=1e-1.

The model that performs better on the validation set and has a low validation loss is then tested on test set (the model that performs better on the validation set in terms of accuracy is the number 6, but its validation loss is high). The accuracy reached on the test set with the model 3 is 57.76%.
Results are summed up in the table below.

Number	Optimizer	LR	Epochs	Momentum	Weight decay	Stepsize	Gamma	TypeNet	Pretrained	Acc.Test	Acc.Val	Acc.Train	Loss Val	Loss Train
1	SGD	1,00E-04	30	0.9	5,00E-05	20	0.1	AlexNet	No	-	56.34	99.82	4.06	0.004
2	SGD	1,00E-02	30	0.9	5,00E-05	30	0.1	AlexNet	No	-	14.97	15.31	4.00	3.95
3	SGD	1,00E-03	70	0.9	5,00E-05	40	0.1	AlexNet	No	57.76	57.98	99.79	3.48	0.005
4	Adam	1,00E-04	30	-	default (0)	20	0.1	AlexNet	No	-	58.09	99.61	3.81	0.008
5	Adadelta	1,00E-02	30	-	default(0)	20	0.1	AlexNet	No	-	44.39	64.21	2.60	1.39
6	Adadelta	1,00E-01	50	-	default(0)	30	0.1	AlexNet	No	-	58.47	99.75	5.85	0.006
7	AdamW	1,00E-04	25	-	default(0.01)	25	0.1	AlexNet	No	-	53.63	97.30	3.47	0.15

5. TRANSFER LEARNING

Since the Caltech-101 dataset is not very big, the transfer learning technique is exploited.
It is a research problem that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem.

In this case, the network that has been trained on the Imagenet dataset, which is very rich and contains more than 14 million images, could be useful on the Caltech-101 dataset, which contains only 5784 images and its classes are present in the Imagenet dataset. Loading the pretrained model is very easy in Pytorch with *pretrained=True*.

In the first stage of this experiment, the pretrained AlexNet network gets trained also on the Caltech-101 training set and is evaluated on the Caltech-101 validation set.
On a second stage, only the fully connected layers or the convolutional layers are trained with the best hyperparameters found before.

Training only the fully connected layers or both the fully connected and the convolutional layers lead to very similar results, while training only the convolutional layers leads to worse results.
A reasonable hypothesis could be that since the ImageNet dataset is already very rich, comprehends all the classes present in the Caltech dataset and the pictures in Imagenet dataset are not that different from the pictures in Caltech-101 dataset, training only the fully connected layers is enough to obtain satisfying results since the pretrained convolutional layers are already useful.
Avoiding to train fully connected layers leads to suboptimal results, but this is not unexpected, since the fully connected layer of Alexnet was modified as discussed before (due to the different number of classes) and should be trained again.

Results are summed up in the table below. For every group of parameters to optimize (all, only convolutional layers or only fully connected), the best model is tested on the test set.

Number	Optimizer	LR	Epochs	Momentum	Weight deca	Stepsize	Gamma	TypeNet	Trained layers	Pretrained	Acc.Test	Acc.Val	Acc.Train	Loss Val	Loss Train
1	SGD	1,00E-04	30	0.9	5,00E-05	20	0.1	AlexNet	All	Yes	-	83.19	99.89	0.83	0.006
2	Adam	1,00E-04	30	-	default(0)	20	0.1	AlexNet	All	Yes	-	83.26	99.93	0.92	0.0004
3	AdamW	1,00E-04	40	-	default(0)	20	0.1	AlexNet	All	Yes	84.41	84.47	99.93	0.82	8.18e-5
4	SGD	1,00E-04	30	0.9	5,00E-05	20	0.1	AlexNet	Conv	Yes	-	48.65	53.66	2.42	2.05
5	Adam	1,00E-04	30	-	default(0)	20	0.1	AlexNet	Conv	Yes	66.67	65.11	98.78	1.90	0.07
6	AdamW	1,00E-04	40	-	default(0)	20	0.1	AlexNet	Conv	Yes	-	64.45	98.72	1.96	0.08
7	SGD	1,00E-04	30	0.9	5,00E-05	20	0.1	AlexNet	FC	Yes	-	83.02	99.34	0.64	0.04
8	Adam	1,00E-04	30	-	default(0)	20	0.1	AlexNet	FC	Yes	-	83.67	99.93	0.88	0.0004
9	AdamW	1,00E-04	40	-	default(0)	20	0.1	AlexNet	FC	Yes	84.96	84.37	99.93	0.79	0.0008

6. DATA AUGMENTATION

Data augmentation in data analysis are techniques used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data. It acts as a regularizer and helps reduce overfitting when training a machine learning model.

This technique is implemented by applying on the pictures belonging to the Caltech-101 training set some transformations. The used Pytorch transforms, applied on each training picture with a given probability, have parameters that randomly change in a given interval.

When training Alexnet from scratch using data augmentation, validation losses slightly decrease more but the accuracies are very similar to the ones obtained with the standard training.
When using the pretrained Alexnet and training all the layers, the accuracies often worsen with respect to the standard training, and the validation losses do not decrease. However, freezing convolutional layers and training only the fully connected layers lead to accuracies which are very similar to those obtained without data augmentation.

One could argue that when using these transforms on the training set, images become too different from the ones in the test set and therefore this procedure leads to worse results. This effect is obviously minimized when training only the fully connected layers.

Results are summed up in the table below. The non pretrained and pretrained model that perform better on the validation set are then tested on the test set, obtaining accuracies of 56.61% and 84.82%.
The best models without data augmentation obtained 57.76% and 84.96%.
Therefore, one can confidently say that significant improvements using the data augmentation technique are not observed.

Number	Optimizer	LR	Epochs	Momentum	Weight decay	Stepsize	Gamma	TypeNet	Trained layers	Transforms	Pretrained	Acc.Test	Acc.Val	Acc.Train	Loss Val	Loss Train
1	SGD	1,00E-03	50	0.9	5,00E-05	25	0.1	AlexNet	All	ColorJitter	No	-	57.12	98.44	3.31	0.06
2	SGD	1,00E-03	50	0.9	5,00E-05	25	0.1	AlexNet	All	RandomAffine	No	56.61	57.50	95.81	3.39	0.14
3	SGD	1,00E-03	50	0.9	5,00E-05	25	0.1	AlexNet	All	RandomHorizontalSplit	No	-	55.56	91.07	3.06	0.29
4	SGD	1,00E-04	40	0.9	5,00E-05	25	0.1	AlexNet	All	RandAffine + RandHorSplit	Yes	-	80.77	95.53	1.01	0.13
5	SGD	1,00E-04	40	0.9	5,00E-05	25	0.1	AlexNet	All	RandHorSplit	Yes	-	78.76	98.58	1.18	0.04
6	SGD	1,00E-04	40	0.9	5,00E-05	25	0.1	AlexNet	All	ColorJitter	Yes	-	80.42	97.33	1.08	0.09
7	SGD	1,00E-03	80	0.9	5,00E-05	25	0.1	AlexNet	All	Randomcrop	Yes	-	77.42	98.16	1.08	0.05
8	SGD	1,00E-03	40	0.9	5,00E-05	25	0.1	AlexNet	FC	RandHorSplit + ColorJitter	Yes	84.82	83.22	99.10	0.93	0.03
9	SGD	1,00E-03	40	0.9	5,00E-05	25	0.1	AlexNet	FC	Split + ColorJit + RandAff + Crop	Yes	-	74.10	90.38	1.24	0.31
10	SGD	1,00E-03	40	0.9	5,00E-05	25	0.1	AlexNet	FC	Random Perspective	Yes	-	82.46	98.61	0.96	0.04
11	SGD	1,00E-03	60	0.9	5,00E-05	25	0.1	AlexNet	FC	RandomPersp + Gaussianblur	Yes	-	82.12	98.78	1.07	0.04
12	SGD	1,00E-04	60	0.9	5,00E-05	25	0.1	AlexNet	FC	RandHorSplit	Yes	-	80.77	98.75	0.84	0.06

7. BEYOND ALEXNET

Several other architectures are used to see how they behave on this dataset.
Sometimes also the *in_features* parameter of the last fully connected layer has to be changed due to different architectures (as well as the *out_features* parameter for the different number of classes).
Among these networks, the one that if pretrained gives the best accuracy on the validation dataset is the ResNext50, which reaches an accuracy of 95.54% on the test set. if not pretrained, the best accuracy on the test set is given by the Densenet121.
An interesting detail to notice is that not necessarily deeper networks obtain higher accuracies: Resnet18 works better on the validation set than its deeper variants both when pretrained and when trained from scratch.

Number	Optimizer	LR	Epochs	Momentum	Weight decay	Stepsize	Gamma	TypeNet	Trained layers	Pretrained	Acc.Test	Acc.Val	Acc.Train	Loss Val	Loss Train
1	SGD	1,00E-03	30	0.9	5,00E-05	10	0.1	Vgg16	All	Yes	90.32	90.38	99.93	0.42	0.0008
2	SGD	1,00E-03	30	0.9	5,00E-05	10	0.1	Resnet18	All	Yes	93.25	93.18	99.86	0.26	0.03
3	SGD	1,00E-03	30	0.9	5,00E-05	10	0.1	Densenet121	All	Yes	90.49	91.32	99.89	0.36	0.009
4	SGD	1,00E-03	30	0.9	5,00E-05	10	0.1	Googlenet	All	Yes	93.67	93.04	99.06	0.26	0.14
5	SGD	1,00E-03	30	0.9	5,00E-05	10	0.1	Resnet50	All	Yes	91.11	90.83	99.86	0.39	0.01
6	SGD	1,00E-03	30	0.9	5,00E-05	10	0.1	Resnext50	All	Yes	95.54	95.05	99.93	0.19	0.01
7	SGD	1,00E-02	60	0.9	5,00E-05	20	0.1	Resnext50	All	No	63.46	60.85	99.75	2.57	0.01
8	SGD	1,00E-02	60	0.9	5,00E-05	20	0.1	Vgg16	All	No	49.46	50.00	99.79	4.88	0.007
9	SGD	1,00E-02	30	0.9	5,00E-05	15	0.01	Vgg16	All	No	48.15	47.82	95.05	3.20	0.17
10	SGD	1,00E-02	60	0.9	5,00E-05	25	0.1	Resnet50	All	No	57.86	56.15	99.68	3.37	0.005
11	SGD	1,00E-02	60	0.9	5,00E-05	25	0.1	Densenet121	All	No	70.16	69.81	99.93	1.44	0.01
12	SGD	1,00E-02	60	0.9	5,00E-05	25	0.1	Resnet152	All	No	56.51	54.87	99.75	8.32	0.01
13	SGD	1,00E-02	40	0.9	5,00E-05	20	0.1	Resnet18	All	No	63.84	62.10	99.93	1.80	0.003
14	SGD	1,00E-02	40	0.9	5,00E-05	20	0.1	Resnet18	All	Yes	93.70	93.36	99.93	0.24	0.01