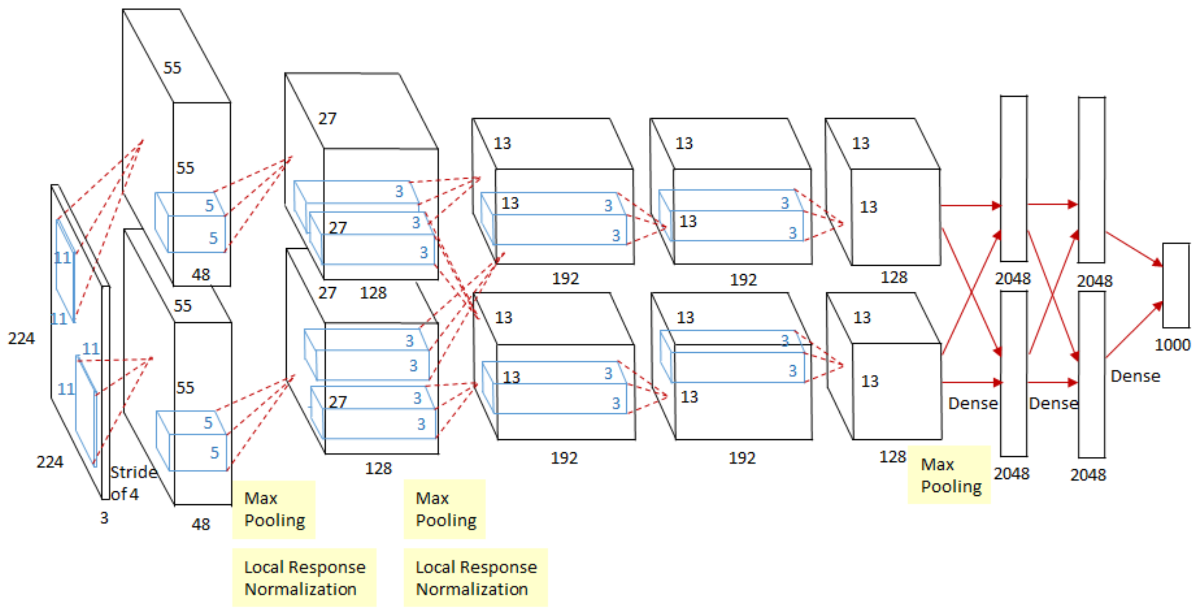
POLITECNICO DI TORINO

Homework 2

Student name: *Erich Malan*
s267475

Course: *Machine Learning and Deep Learning* – Professor: *Barbara Caputo*
Due date: *August, 2020*



AlexNet - Image Classification

Introduction

The aim of this assignment is the image classification of the Caltech101 Dataset through AlexNet, a convolutional neural network designed by Alex Krizhevsky, first by training from scratch, then by using the pretrained on ImageNet version and finally fine-tuning hyperparameters. Along this report are presented and discussed the related results and comparisons.

Data

0.1. Data exploration. The dataset proposed is The Caltech101 dataset, which is composed of 8677 images subdivided into 101 categories. In fact there is a background class that has to be removed. Each folder's name corresponds to a target and contains the images of such category. The images' shape is the same of size [224,224,3], in fact are squared pictures of 224x224 RGB encoded, and so they have 3 channels. The train.txt and test.txt files denote the split that must be performed to subdivide train's dataset from test's one. As it can be seen in *figure 1* the distribution is not uniform but this aspect will not be covered by specific handmade treatments.

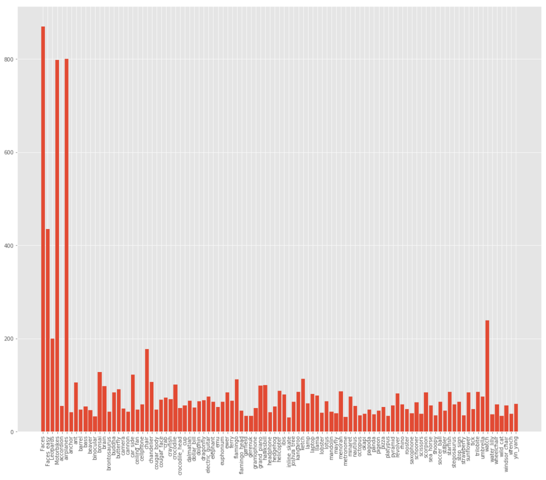


Figure 1: *Distribution of target class*

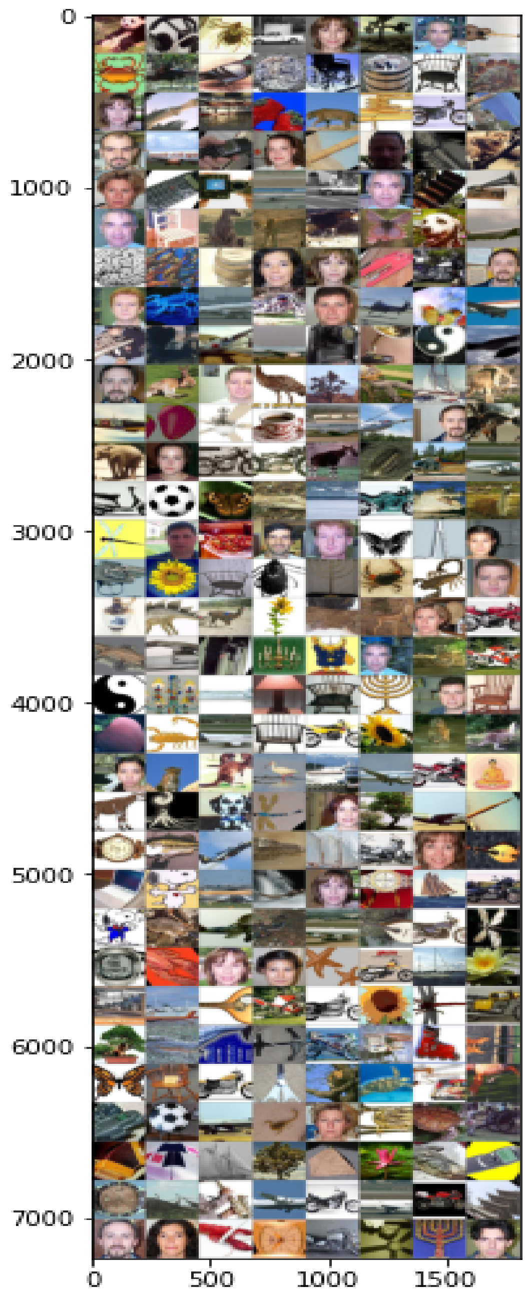


Figure 2: *some random images picked from train set*

Data preparation.

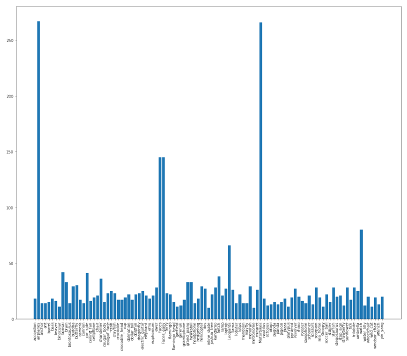


Figure 3: *Distribution of target class in training set*

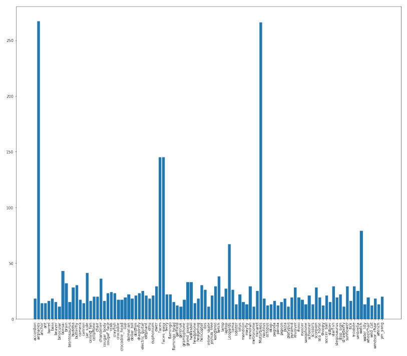


Figure 4: *Distribution of target class in validation set*

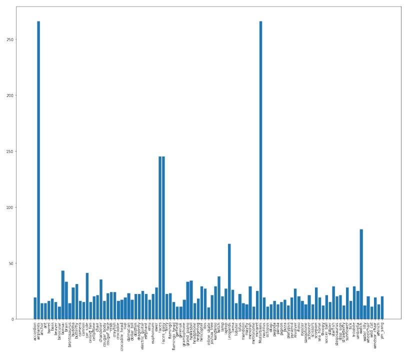


Figure 5: *Distribution of target class in test set*

Accordingly to the assignment requirements data must be loaded using Caltech class. Some specific methods of the class had to be implemented to get it work. By passing the folder containing data it maps each PIL loaded image along to its label, and in order to get each item by simply passing an index to the method __getitem__ it returns the tensor of the image and the corresponding label. An other method has been implemented to perform a split between Train and Validation subsets with a ratio of 0.5 by maintaining the same distribution of targets (for example if train contains 1000 faces, val contains ≈1000 faces too normally a percentage would have more sense but in this case the split is 50%, the ≈ is to denote that some targets are odds so it was impossible state the perfect half). An other action performed by the Caltech class is the image transformation while getting an element, each image is subject to the following pipeline:
1) Resize(256)
2) CenterCrop(224)
3) ToTensor()
4) Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))

CenterCrop and ToTensor are particularly important to match AlexNet inputs.

Training from Scratch

Training & Validating. AlexNet is composed by 8 layers, the first five are Convolutional layers while the last three are fully connected that contains most of the neurons. In particular the fully connected part is grouped and then named as the classificator, by an operation of flattening it receives a 1d-array of size 4096 and as default outputs an array of size 1000. The output array is the one that must be modified in order to fit the 101 classes.

Hyperparameter	Value	Description
NUM_CLASSES	101	Number of targets, defines the number of nodes of the last FC layer
BATCH_SIZE	256	This is restricted to the limitation of the GPU, by the way larger batch sizes allow higher learning rates but deeper models are subject to hw limitations.
LR	1e-3	Initial learning rate, too high values could lead the model to diverge while too low could not improve enough the model.
OPTIMIZER	SGD	Stocastic Gradient Descent optimizer, is the algorithm that modifies the learning rate and the weights aiming to decrease the Loss.
MOMENTUM	0.9	The momentum modify the behaviour of the optimizer by accelerating the convergence towards relevant directions and reducing fluctuations towards irrelevant ones. Values near 1 mean lower variation compared to lower values.
WEIGHT_DECAY	5e-5	The weight decay try to regularize the complexity of a neural network by limiting the weights to become too much large and overfit the data. Small values of weigth decay allow more freedom to the model's weights to become larger.
NUM_EPOCHS	30	Number of training epoch, total number of iteration of the dataset within the model must get the lowest loss as possible
STEP_SIZE	20	The step size defines the number of epochs that must be performed before updating the Learning Rate (normally by decreasing it progressively)
GAMMA	0.1	Decay rate of Learning Rate after the epochs defined by the STEP SIZE

AlexNet Default Parameters. Neural networks need a lot of training on large datasets before performing well, in this case the net is trained from scratch and without optimization lead to a very low accuracy = 0.09 (both validation and test sets). In *figure 6* and *figure 7* it can be seen that the loss are decreasing very slowly but they are not diverging. The accuracy initially increases but get soon stuck on a sort of local minimum. Further test presented try to find out the best model training by changing hyperparameters accordingly to their importance, by performing a search based on 'keep the best, search the next' approach.

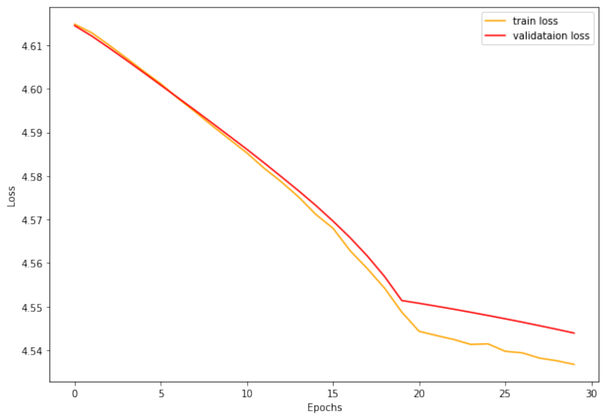


Figure 6: *Alexnet (default) Loss lineplot*

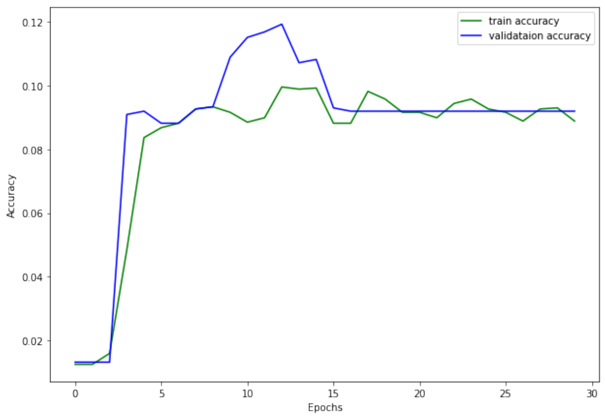


Figure 7: *Alexnet (default) Accuracy lineplot*

AlexNet - LEARNING RATE. So the first main problem concerns the learning rate that is probably too low, I tried to increase it by 10. Tables below represents the results obtained

Learning Rate	Validation loss	Validation Accuracy
1e-3	4.55	0.09
1e-2	3.24	0.31
1e-1	2.85	0.40

The learning rate was definetely too low, the best result occurs for $1x10^{-1}$ which is an impressive result (compared to the others 2) but as shows a deeper look at the loss lineplot of *figure 8* it comes to be disclose the limit of the divergent behaviour of the parameter (it starts to behave as a sort of randomized sawtooth function that continue decrease but shows to be less smooth than 0.01 with an higher final gap between train and val loss). Other attempt with such a learning rate performed in order to optimize the STEP SIZE parameter suffered of this overstated 'jump' by showing hops and stops. Stops mean that the loss is increasing and it's due to the fact I implemented a control that saves the best model but if after 5 epochs the model is not improving it gets back trying to search another path, while hops literally means that the loss plot was diverging. Below are reported train/validation accuracies' and losses' lineplots (*figure 8* and *figure 9*).

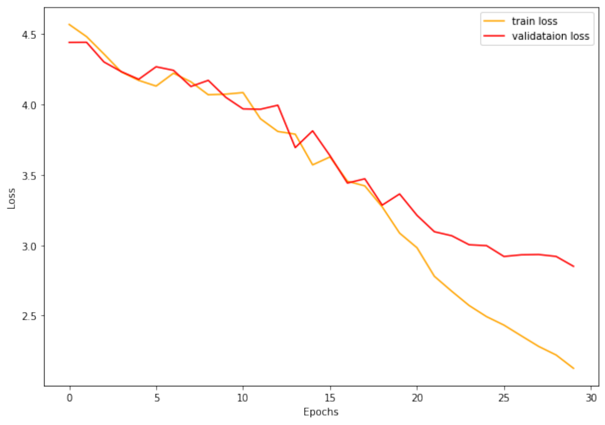


Figure 8: *LR 1e-1 Loss lineplot*

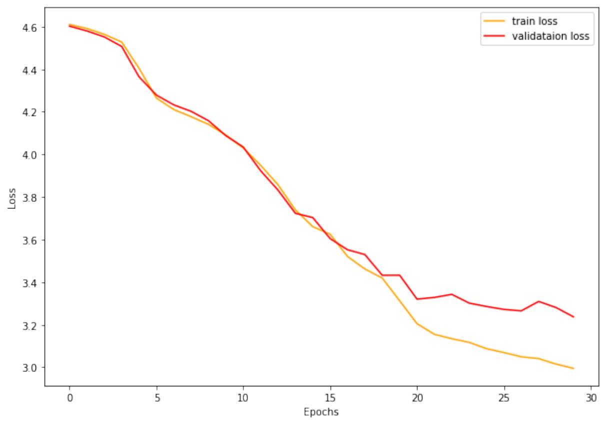


Figure 9: *LR 1e-2 Loss lineplot*

AlexNet - STEP SIZE. The second hyperparamenter to be tested is the step size, that is directly connected to the learning rate because it defines when it is decreased by the multiplication of GAMMA. I applied a 5 step down policy and after 3 attempt it is easy demonstrated that the model decrease too much early the learning rate. The loss increase and the accuracy decrease are almost linear, so I decided to leave the default parameter.

Learning Rate	Step Size	Validation loss	Validation Accuracy
1e-2	20	3.24	0.31
1e-2	15	3.49	0.26
1e-2	10	4.18	0.15
1e-2	5	4.24	0.09

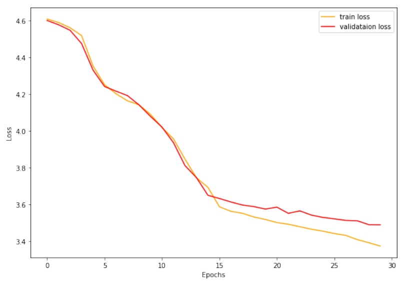


Figure 10: *STEP SIZE 15, Loss lineplot*

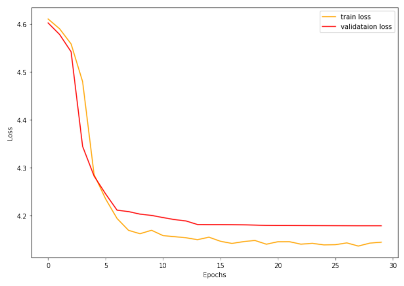


Figure 11: *STEP SIZE 10, Accuracy lineplot*

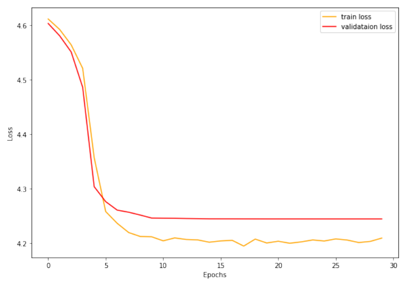


Figure 12: *STEP SIZE 5, Accuracy lineplot*

AlexNet - WEIGHT DECAY. Next two more values of weight decay had been tested, one lower one higher to give more or less freedom state to weights and see their behaviour in terms of loss and accuracy.

Learning Rate	Step Size	Weight Decay	Validation loss	Validation Accuracy
1e-2	20	5e-4	3.34	0.28
1e-2	20	5e-5	3.24	0.31
1e-2	20	5e-6	3.35	0.29

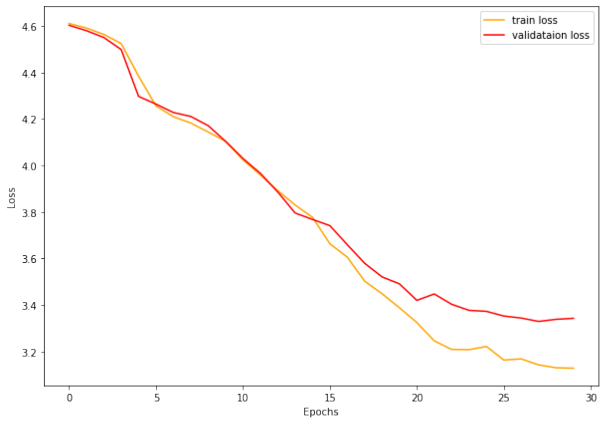


Figure 13: *Weight decay 5e-4 Loss lineplot*

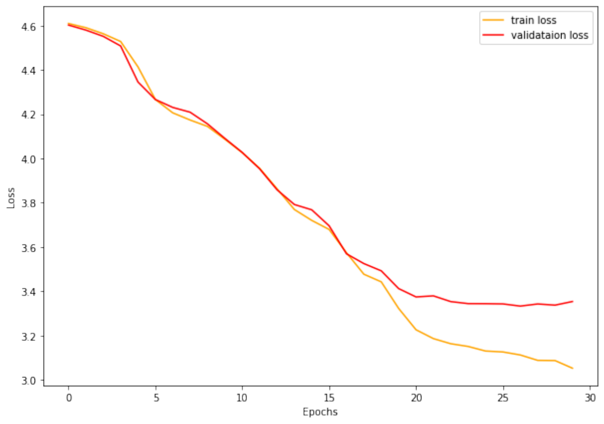


Figure 14: *Weight decay 5e-6 Loss lineplot*

We don't see in this case big differences in terms of loss and accuracy neither I've been able to state a particular model's overfitting due to the higher freedom of weights, but maybe because I had been cautious by testing inside a close range.

AlexNet - GAMMA. The last hyperparamenter to be tested is GAMMA which is the multiplier factor of the LEARNING RATE after K-STEPs, and in this case is constant over the time. We can notice by considering graphs that an higher GAMMA leads to a 0.30 loss decrease compared to initial value of 0.1 even if it present a spike near the 25th epoch.

Learning Rate	Step Size	Weight Decay	Gamma	Validation loss	Validation Accuracy
1e-2	20	5e-5	0.1	3.24	0.31
1e-2	20	5e-5	0.2	3.31	0.30
1e-2	20	5e-5	0.3	3.16	0.30
1e-2	20	5e-5	0.5	2.95	0.35

AlexNet - Testing. In the end the best model has revealed to be the first mod applied to the learning rate even on the test phase.

Learning Rate	Step Size	Weight Decay	Gamma	Validation loss	Validation Accuracy	Test accuracy
1e-1	20	5e-5	0.1	2.85	0.40	0.41
1e-2	20	5e-5	0.5	2.95	0.35	0.37

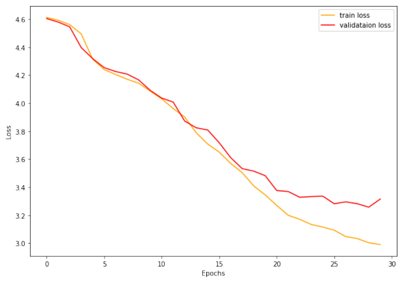


Figure 15: *Gamma 0.2 Loss lineplot*

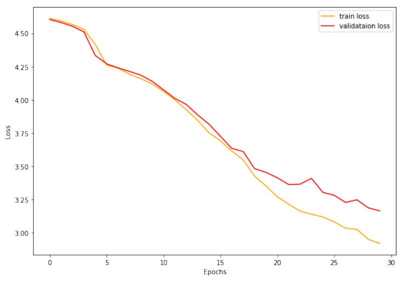


Figure 16: *Gamma 0.3 Loss lineplot*

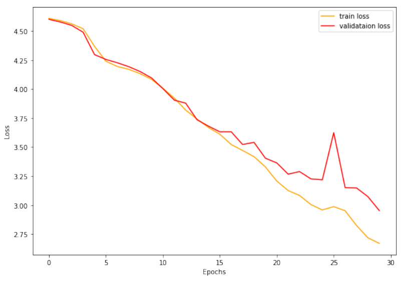


Figure 17: *Gamma 0.5 Loss lineplot*

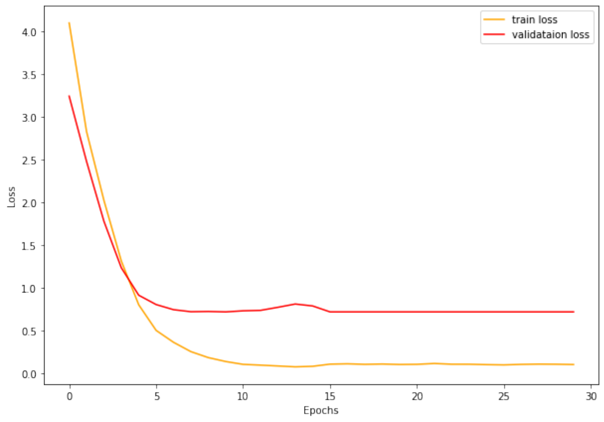


Figure 18: *Alexnet pretrained (default) Loss lineplot*

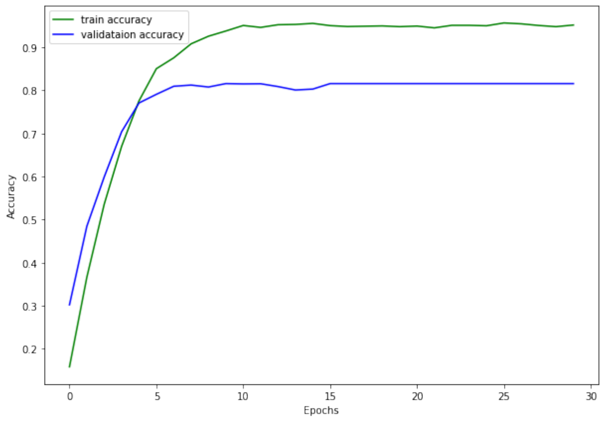


Figure 19: *Alexnet pretrained (default) Accuracy lineplot*

Transfer Learning

Default. Normally, as enounced before, neural networks need large datasets to reach sufficient performances. Pytorch provides for each standard neural network, the same model but already trained on ImageNet dataset, a dataset composed of 14,197,122 images indexed over 21841 synsets. The main idea is that the model get a great kick-start on the configuration of neurons, the features extracted from the beginning probably are the ones approssimatively right to be classified, they just need to be correctly mapped to the new targets neurons stack, training the FC-layers more generally, or also only the last fully connected one that identifies the specific image classification problem. Normalization applied : mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225] The train,val,test sets must be reloaded because the normalization applied must be the same as the one applied towards images of imagenet, otherwise the pretrain is unuseful. In fact the accuracy doubles the alexnet trained from scratch, and the loss drastically decrease below 1, this happens in just few epochs. (*figure 18* and *figure 19*)

Train Loss	Train Acc	Val Loss	Val Acc	Test Accuracy
0.0997	0.95	0.7162	0.82	0.81

Hyperparameters finetuning. The same searching path applied before is now applied to the pretrained AlexNet.

Learning Rate	Step Size	Weight Decay	Gamma	Validation loss	Validation Accuracy
1e-1	20	5e-5	0.1	0.72	0.82
1e-2	20	5e-5	0.1	0.75	0.82
1e-3	20	5e-5	0.1	0.75	0.81
1e-2	20	5e-5	0.1	0.75	0.82
1e-2	15	5e-5	0.1	0.77	0.82
1e-2	5	5e-5	0.1	0.74	0.84
1e-2	20	5e-4	0.1	0.73	0.81
1e-2	20	5e-5	0.1	0.75	0.82
1e-2	20	5e-6	0.1	0.75	0.81
1e-2	20	5e-5	0.1	0.75	0.82
1e-2	20	5e-5	0.3	0.75	0.82
1e-2	20	5e-5	0.5	0.83	0.78

Also in this case the default parameters played their good role but this time the hyper-parameters' finetuning did not lead to any interesting improvement, each attempt got the loss decrease saturated in few epochs.

Transfer learning - Test.

Learning Rate	Step Size	Weight Decay	Gamma	Validation loss	Validation Accuracy	Test Accuracy
1e-2	20	5e-5	0.1	0.71	0.82	0.81
1e-2	5	5e-5	0.1	0.74	0.84	0.82

Freeze layers. The freeze layers' technique is sometimes used with pretrained networks, and consists of not to optimize weights of some layers while training the network. Here the proposal is to train only FC layers first, then only the convolutional ones.

Layers freezed	Weights optimized	Validation loss	Validation Accuracy
None	57,417,637	0.72	0.82
Convolutional	54,947,941	0.60	0.84
Fully connected	2,469,696	2.03	0.56

As it could be imagined, by training only the convolutional layers the model see an accuracy drop and an higher loss, in fact the classification part is performed over FC-layers. In particular the label assignment occurs at the last layer that in this case has been changed to fit the number of desired classes and this clearly states that at least the last layer must be trained to obtain a good classification accuracy. Freezing only convolutional layers instead increased the performances of the model, probably because the features extractor trained in ImageNet is more general and suffers less of data overfitting.

Freeze layers - Test.

Layers freezed	Validation loss	Validation Accuracy	Test Accuracy
Convolutional	0.60	0.84	0.84

Data augmentation. When it occurs that the dataset is small to train a net it could be applied some transformation to the training set to fake a wider and broader dataset. The images are still processed with the same transformation process as before with some additional transformations such as different rotations, flips or color that are randomly applied over the epochs. Three different transformation pipelines are provided but I decided to keep for all of them flips and rotations because in my opinion the model should be more steady towards them. Then for each one I applied some odd transformation such as color jitter and color jitter after grayscale mapping.

Transformations	Validation loss	Validation Accuracy
RandomHorizontalFlip RandomVerticalFlip RandomRotation RandomPerspective	1.21	0.69
RandomHorizontalFlip RandomVerticalFlip RandomRotation RandomPerspective ColorJitter	1.27	0.69
RandomHorizontalFlip RandomVerticalFlip RandomRotation RandomPerspective Grayscale ColorJitter	0.76	0.81

ColorJitter is supposed to change the hue or the saturation of the image, while grayscale maps the image into a single channel repeated (RGB has three channels, that are mapped into grayscale 0.3R + 0.59G + 0.11B and then are repeated to recreate the 3 channels requested).

Data augmentation - Test.

Trasformation	Validation loss	Validation Accuracy	Test Accuracy
3	0.76	0.81	0.82

The performance of the model resulted to be worst in terms of time caused by more transformations but also in terms of loss and accuracy (except for the third attempt) probably because the test set is very close to the training one.

Type	Test Accuracy
AlexNet	0.09
AlexNet (finetuned)	0.41
AlexNet - Transfer learning	0.81
AlexNet - Transfer learning (finetuned)	0.82
AlexNet - Freeze layers	0.84
AlexNet - Data augmentation	0.82

Beyond AlexNet

Resnet50. The resnet50 is one of the most performing nets, but is much more complex than alexnet. Based on residual blocks it has 50 layers, but only the last one is FC. It goes much deeper but to get it work it must be decreased a lot the batch size (from 256 to 16) with Tesla K80 on Colab, otherwise Cuda would get 'out of memory' error.

VGG16. VGG16 is one of most famous net and as the name suggest has 16 layers which last 3 are FC type. Differently to the resnet, its structure follows a linear pipeline as does AlexNet, but is much deeper and it needed the batch size to be decreased to 8 to get work.

Net	Train Loss	Train Accuracy	Validation loss	Validation Accuracy	Test Accuracy
Resnet50	0.001	1.00	0.34	0.92	0.91
VGG16	0.050	0.98	0.66	0.84	0.84

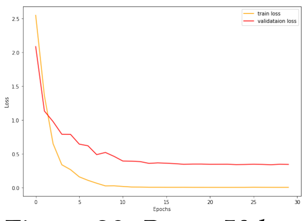


Figure 20: *Resnet50 loss*

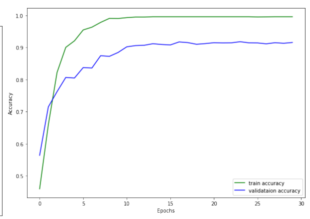


Figure 21: *Resnet50 accuracy*

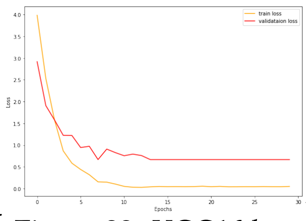


Figure 22: *VGG16 loss*

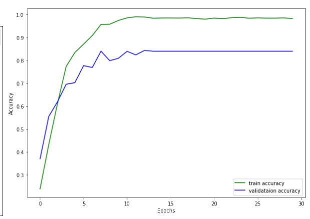


Figure 23: *VGG16 accuracy*

Performances Alexnet - Resnet50 - VGG16. In order to do a basic comparison I trained these networks with almost the default parameters without freeze or data augmentation, to better comprehend the results. In the end Resnet50 with the defaults parameters without freeze demonstrated to be the most performing model in terms of accuracy, while AlexNet in terms of time. VGG16 probably had to be runned with a different set of hyperparameters because has taken lots of time (much more than Resnet probably because of the half batch size) and the results are not such impressive. In the end the Resnet50 confirmed its reputation of general well performing model.