Machine Learning and Artificial Intelligence
Homework 3

Debora Caldarola s263626

Prof. Barbara Caputo, A.Y. 2018/2019

1 Introduction

Deep Learning is a recent trend in machine learning that attempts to replicate the architecture of a brain in a computer. It is believed many levels of processing exists inside the brain, in which each level is learning features at increasing levels of abstraction.
Deep Neural Networks (**DNN**) constitute a framework for different machine learning algorithms to allow them to process complex data inputs. In DNN, each level extracts features from the output of the previous one.
Different kinds of DNN have been developed during the years and we are going to take into consideration three of them: **Traditional Neural Networks**, made of hidden and fully connected layers; **Convolutional Neural Networks** (**CNN**); **Residual Neural Networks** (**ResNet**).

1.1 Homework Sub-tasks

1. Train a traditional neural network made of two hidden layers and a Fully Connected (FC) layer. Express considerations about the final accuracy and the training losses.
2. Train a Convolutional Neural Network (CNN) is the following cases: traditional network with 32/32/32/64, or 128/128/128/256, or 256/256/256/512, or 512/512/512/1024 convolutional filters; modify the network with 128/128/128/256 filters introducing the Batch Normalization, a fully connected layer and Dropout 0.5 on the FC layer. Comment the behaviour of the final accuracy. Augment the data by performing random horizontal flip and random crop.
3. Load ResNet18 and use the best augmentation schema found in the previous step. Compare the obtained accuracy with the other cases study.

2 Data Preparation

The dataset used in this experience is the so called *CIFAR 100*, already included in `PyTorch` (Figure 1). It consists of 60000 32x32 colour images divided into 100 classes with 600 samples each. There are a training set and a test set for each class, that respectively contain 500 and 100 images. The 100 classes are grouped into 20 superclasses.

2.1 Programming Environment Setup

The exercise will be solved using Python 3.7 programming language. GOOGLE COLAB framework is used to compute the requested data. Moreover, the following libraries and packages will be referenced:

- `numpy`, the fundamental package for scientific computing with Python;

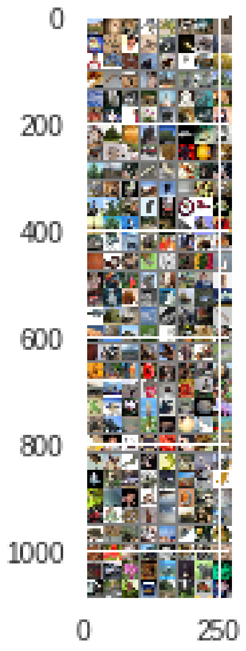


Figure 1: Samples of CIFAR 100 dataset

- `matplotlib.pyplot`: `matplotlib` is a Python 2D plotting library; `pyplot` provides a MATLAB-like interface;
- `PyTorch`, an open-source machine learning library and deep learning framework for Python, based on Torch. The used modules and classes are `nn` and `optim`, specialized on creation and training of neural networks.
- `torchvision`, a package made of popular datatsets and common image transformations for computer vision.

3 Traditional Neural Networks

When traditional Neural Networks are used to process images, each pixel is connected to a neuron. That results in a huge matrix containing all the values for each pixel. In a Fully Connected (FC) layer, each neuron is fully connected to everything that comes from the input.
In the presented example, the net is made of two hidden layers and it is ended by a FC one, which is used to produce a fixed-size output vector.
Training parameters: 4096 neurons; 256 batch size; 20 epochs; 32x32 resolution; Adam solver with 0.0001 learning rate to optimize the parameters.
At each epoch, the percentage of accuracy of the current network is scored on the test set and training loss is calculated. The chart in Figure 2 shows their trends. As expected, loss decreases with respect to the number of epochs: that means the model is improving its ability to learn from the training samples. As a consequence, accuracy improves; moreover, we can assume the model didn't overfit the data yet because accuracy keeps increasing. Low accuracy values result from the high number of classes in the dataset (100), which implies that choosing at random how to classify a sample has a percentage of success equal to 1%.
Because of the high values of loss and the low values of accuracy, we can assert the network is not performing very well. Poor performance might originate from the large number of parameters the network has to learn for each of the 4096 neurons, which is equal to 1024 (images resolution is 32x32) and overfitting problems might occur more often.

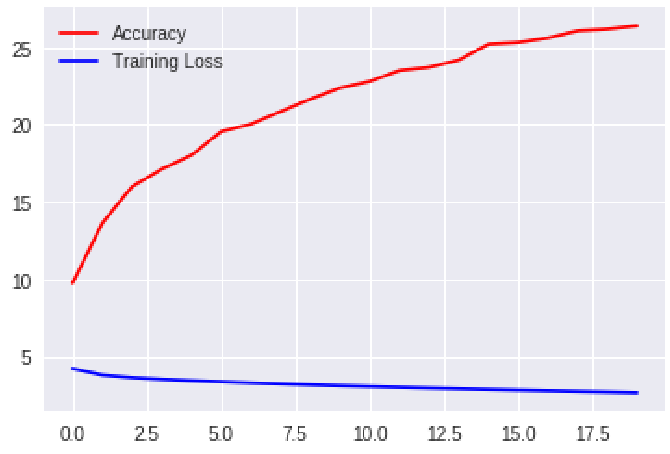


Figure 2: Traditional NN - Trend of training losses and accuracies

4 Convolutional Neural Network (CNN)

A Convolutional Neural Network is a sequence of Convolutional Layers, interspersed with activaction functions, such as the ReLU (Rectified Linear Unit) function.
CNN aims to learn a reduced number of parameters compared to the ones involved in a traditional NN, which are stored in the so called *filter*. The filter is applied on a small window of the image and the convolution among the parameters belonging to the same subwindow is computed. The result is a smaller matrix, called *feature map*, which maps the image on a new feature space.
The first CNN is trained with a number of convolutional filters equal to 32/32/32/64.
The resulting accuracy and training loss trends are plotted in Figure 3. The net behaves better than the traditional one exposed in Section 3: lower values of loss and higher accuracies are reached within the same number of iterations over the whole dataset. The highest percentage of accuracy is 30% with a loss of 2.222. We may assume the expected improvement happened because of the architecture of the net itself: in fact, thanks to the presence of the filters, less parameters have to be learnt and the risk of overfitting is reduced.

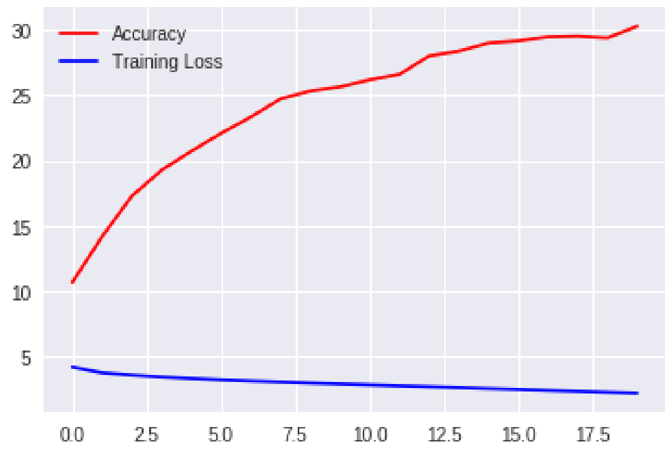
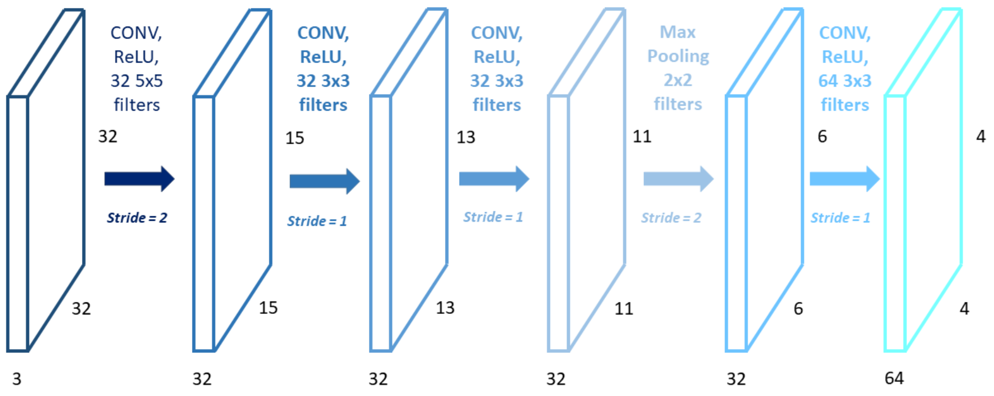


Figure 3: CNN - Trend of training losses and accuracies

The effect of the applied convolutional filters and the consequent feature maps are represented in Figure 4; the dimension of the activation map is calculated according to the formula $dim_{output} = \frac{N-F}{stride}+1$, where N is the dimension of the input and F is the kernel size. In agreement with those results, the number of parameters can be compared to the case study of the traditional NN, where the total amount for each



CONV, ReLU, 32 5x5 filters
Stride = 2
CONV, ReLU, 32 3x3 filters
Stride = 1
CONV, ReLU, 32 3x3 filters
Stride = 1
Max Pooling 2x2 filters
Stride = 2
CONV, ReLU, 64 3x3 filters
Stride = 1
32
32
3
15
15
32
13
13
32
11
11
32
6
6
32
4
4
64

Figure 4: Convolutional filters 32/32/32/64

layer is equal to $n_{neurons} \cdot (32x32) = 2^{12} \cdot 2^{10} = 2^{22}$. The number of parameters in each convolutional layer is instead equal to: $n_{parameters} = (n_{channels} \cdot dim_{kernel} + 1) \cdot n_{filters}$.

- 1st layer: $n_{parameters} = (3 \cdot 5 \cdot 5 + 1) \cdot 32 = 76 \cdot 32 = 2432$.
- 2nd layer: $n_{parameters} = (32 \cdot 3 \cdot 3 + 1) \cdot 32 = 288 \cdot 32 = 9216$.
- 3rd layer: $n_{parameters} = (32 \cdot 3 \cdot 3 + 1) \cdot 32 = 288 \cdot 32 = 9216$.
- 4th layer: $n_{parameters} = (32 \cdot 3 \cdot 3 + 1) \cdot 64 = 288 \cdot 64 = 18432$.

Those values demonstrate how the convolutional NN is more efficient than the traditional one thanks to less parameters to be learnt at each epoch.

4.1 Convolutional Filters

Different numbers of convolutional filters are now taken into account. In particular, an incremental increase is analyzed: 128/128/128/256, 256/256/256/512 and 512/512/512/1024 filters.
The percentage of accuracy and the training loss are stored for each batch and Figure 5, Figure 6 and Figure 7 show the results in the three exposed different cases.

The general consideration that can be expressed is that the more convolutional filters are inserted in the network, the more the accuracy grows and the loss decreases. At the end of the training, the CNN with 128/128/128/256 filters stabilizies its accuracy around the percentage of 32-33%; when 256/256/256/512 filters are used, the fluctuations go around the value of 34%; in the last case, the final accuracy is about 35%. The fact that the growth is not so meaningful combined to the stabilization of the accuracy might mean we overestimated the number of filters and the model starts overfitting the data. At the same time, the increase in the number of filters brings to an exponential growth of the computational time: in fact, the first net took about six minutes to produce the results; the training of the second network lasted 14 minutes; the third CNN used almost 46 minutes.

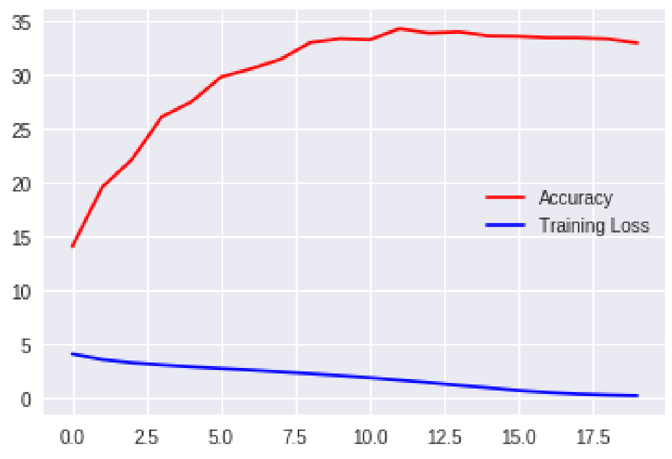


Figure 5: CNN accuracy and loss, 128/128/128/256 filters

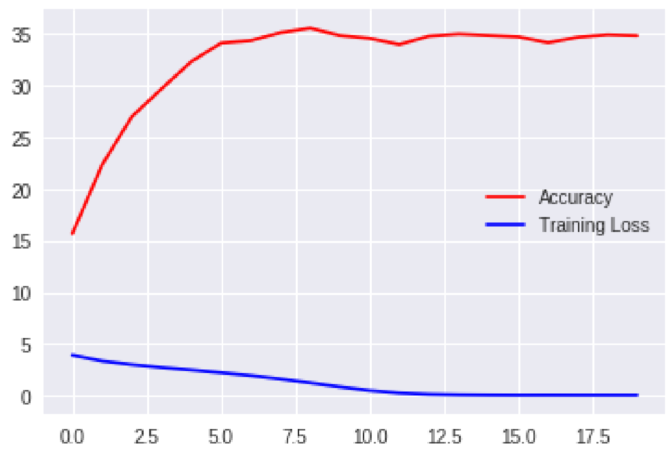


Figure 6: CNN accuracy and loss, 256/256/256/512 filters

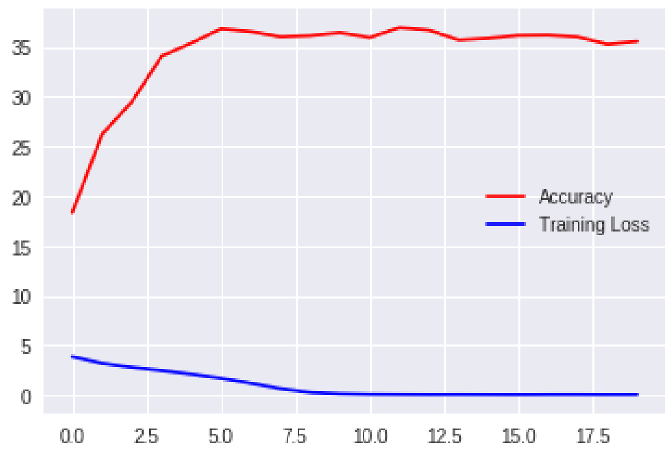


Figure 7: CNN accuracy and loss, 512/512/512/1024 filters

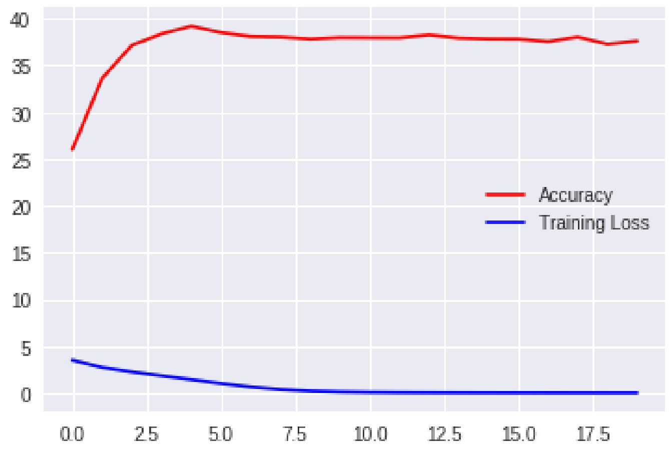


Figure 8: CNN with Batch Normalization layers - Accuracy and loss trends

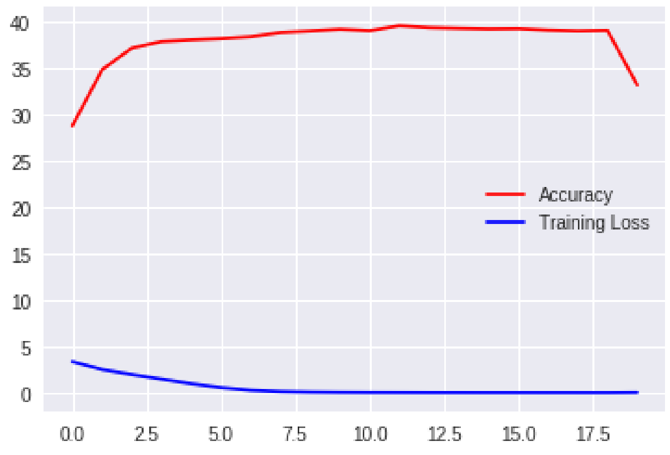


Figure 9: CNN with Batch Normalization layers, wider FC1 - Accuracy and loss trends

4.2 Batch Normalization Layer, Fully Connected Layer and Dropout 0.5

Starting from the network with 128/128/128/256 filters, the following modifications are made:

(a) A *Batch Normalization Layer* is added after each convolutional layer. Its scope is to normalize the data in order to avoid reducing of the gradient in the last layers. In fact, such thing happens because of the non-linearity of the output as the resulting effect of the ReLU function.
Given X data, the normalized output is described by:

$$\hat{x}(k) = \frac{x(k) - E(x(k))}{\sqrt{Var(x(k))}},$$

where the mean $E(x(k))$ and the variance $Var(x(k))$ are calculated at training time on each batch. The producted result in terms of accuracy and loss is shown in Figure 8.

(b) Batch Normalization layers are kept and the first fully connected layers is expanded from 4096 to 8192 neurons (Figure 9).

(c) The network is regularized with the *Dropout 0.5* method to avoid co-adaptation without reducing its capacity (Figure 10).

The introduction of data normalization and regularization on the network definitely improved the performance of our CNN: in all the different cases, the medium accuracy is around 39%. The following main

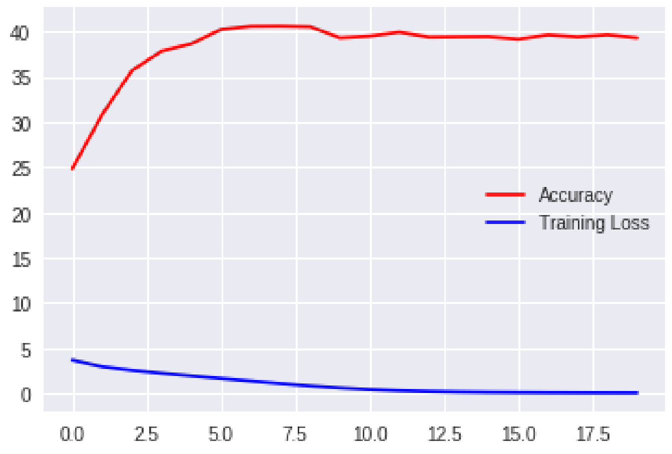


Figure 10: CNN with Batch Normalization layers and Dropout 0.5 - Accuracy and loss trends

aspects can be pointed out:

- In situation (a), the effect of the Batch Normalization layers can be especially appreciated in the first iterations on the dataset, where there is a fast growth of the accuracy scored on the test set.
- In situation (b), in the last iteration the value of the accuracy suddenly drops from 39% to 33%. That might be happening because of overfitting.
- The main aspect in the three study cases is that while the training loss keeps decreasing, the accuracy tends to stabilize around a constant value. A possible explanation of that behaviour is that the network got stuck in a local minimum because it kept receiving the same inputs with consequent no changes in the gradient it steps along. Then the result is that the predictions remain the same.
- The best result is obtained when Batch Normalization layers are used combined to the Dropout 0.5 regularization method.

4.3 Random Horizontal Flipping and Random Crop

Random horizonal flipping and random crop are two of the main methods used to perform ***Data Augmentation***.
Their introduction should make the classification problem harder for the network during the training session because more various inputs are provided; at the same time, the testing phase will be easier.
The effect of those two methods was tested separately and the results are depicted in Figure 11 and Figure 12.
The beneficial effect of those modifications can be rapidly seen: the accuracy trend doesn't reach a constant value but keeps improving. Then means the network didn't stuck on a local minimum thanks to the augmented inputs.
An ulterior improvement can be made combining data augmentation with the introduction of Batch Normalization layers and Dropout 0.5, as shown in Figure 13 (here random horizontal flipping is used): the best accuracy is now equal to 45%.

5 Residual Neural Network (ResNet)

Residual Neural Networks introduce a different architecture from what we have seen since now. Inside there are two main paths the input can follow: the first one is a traditional convolutional network; the second one skips all the connections and connects the input to the output produced by the current layer

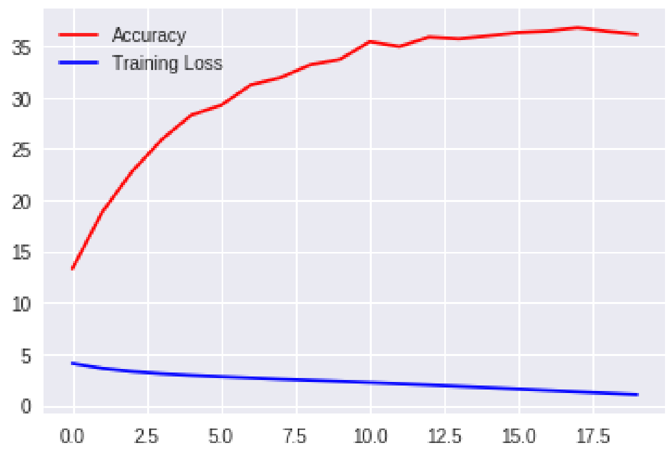


Figure 11: Random horizontal flipping on CNN network

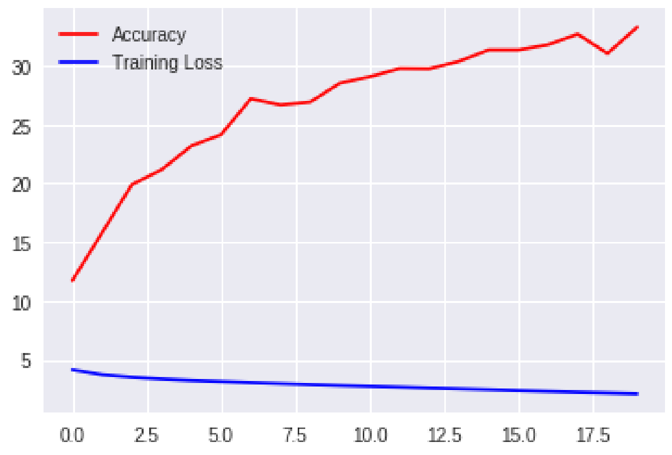


Figure 12: Random crop on CNN network

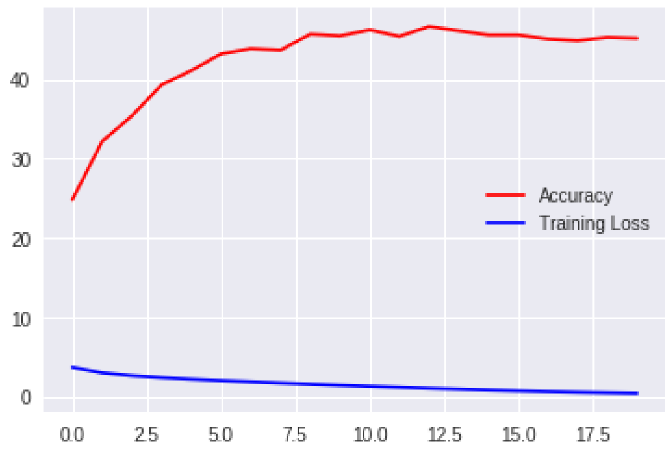


Figure 13: Random horizontal flipping on CNN with Batch Normalization and Dropout 0.5

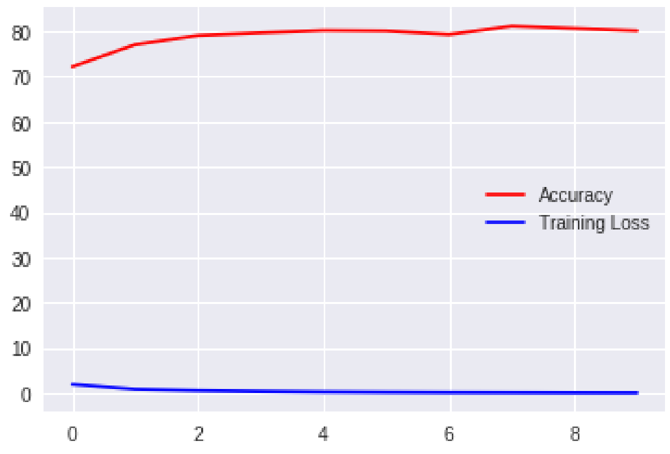


Figure 14: ResNet - Accuracy and training loss trends

so that they can be summed. In this way, the resulting total output is simply an update of the initial input. Thanks to the skip connections and Batch Normalization layers, loss goes to zero very quickly (Figure 14).
In order to train the network, input images were subjected to random horizontal flipping and the parameters were changed to 128 batch size, 10 epochs and 224x224 resolution.
Figure 14 shows there are multiple benefits coming from the use of the ResNet:

- As expected, training loss goes to zero very quickly and reaches the lowest value seen so far, equal to 0.038 in the last iteration. It is interesting to point out that value is smaller with respect to traditional NN too.
- Accuracy values almost double the ones obtained with other networks.
- Thanks to data augmentation, no local minimums paralize the accuracy.

6 Conclusions

So far we have been discussing models inspired by low-level processing in the brain. Three different models have been studied: traditional neural networks, convolutional NN and residual networks. They have produced useful results but also showed different weaknesses, such as computational time, low accuracy values, overfitting and adjustment to local optimums.
Several solutions have been presented, such as regularization, data augmentation and Batch normalization.