Homework 3 – Machine Learning (2018/19)

Angelo Laudani s253177

Deep Learning

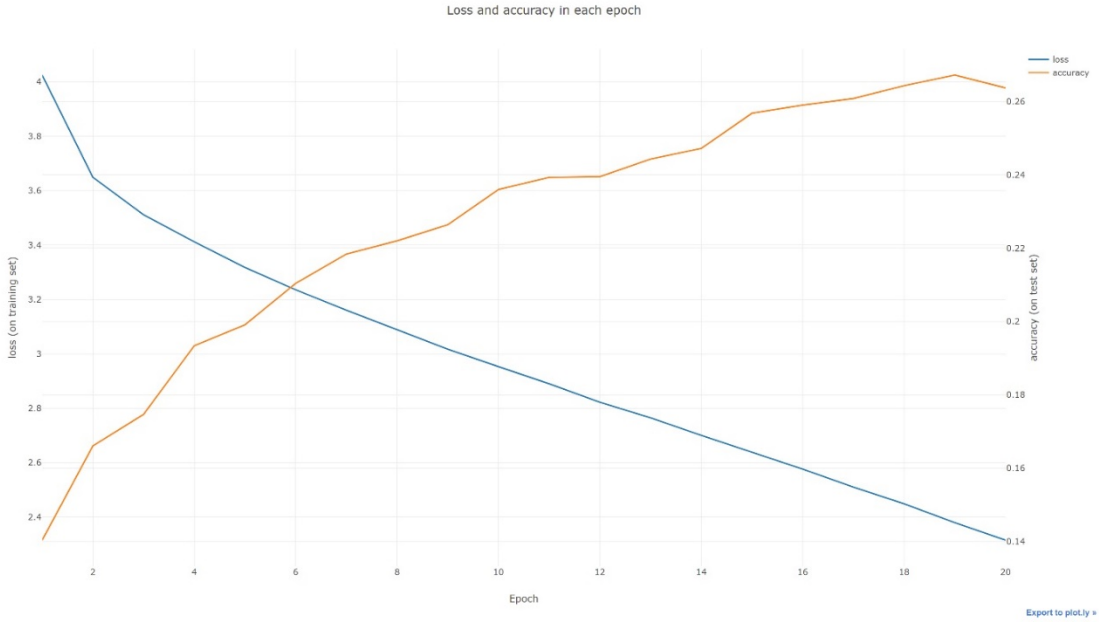
Introduction

For this homework several Neural Networks, whit different parameters, where tested on a big image dataset: CIFAR100. This dataset consists of 100 different classes, for a total number of 60.000 images. At first the performance of a simple Neural Network is evaluated, then a Convolutional Neural Network (with multiple attempts using a different number of convolutional filters, batch normalization and dropout, data augmentation) and finally a pretrained ResNet model, finetuned on our dataset. For achieving the best results, the data is first standardized and normalized before every training phase.

1 – Traditional Neural Network

A traditional neural network, with two hidden layer and a last FC one, is trained on the dataset with the following parameters:

256 Batch Size, 20 epochs and Adam solver with learning rate at 0.0001.

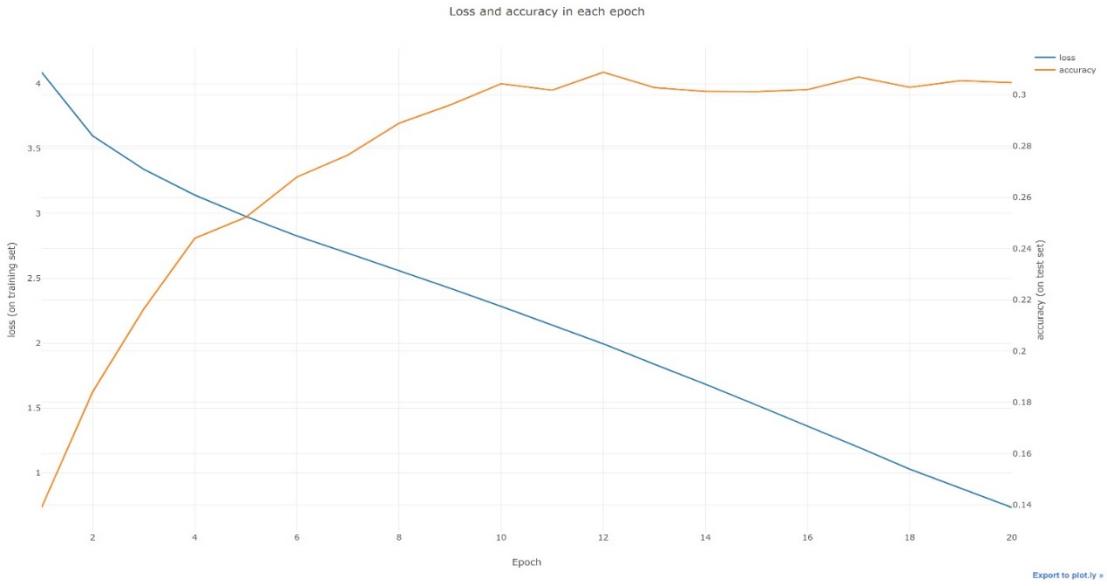


The two fully connected hidden layers (with a simple *sigmoid* activation function) and the last one perform quite well on the test set if we consider the basic architecture of the network and the complex one of the dataset, achieving an accuracy of **26%** after 20 epochs. The loss function though is still high, with a final value of **2.3** on the training set. The traditional Neural Network is definitely not the best class for image recognition, it is not able to scale well on a full image with its fully connected structure.

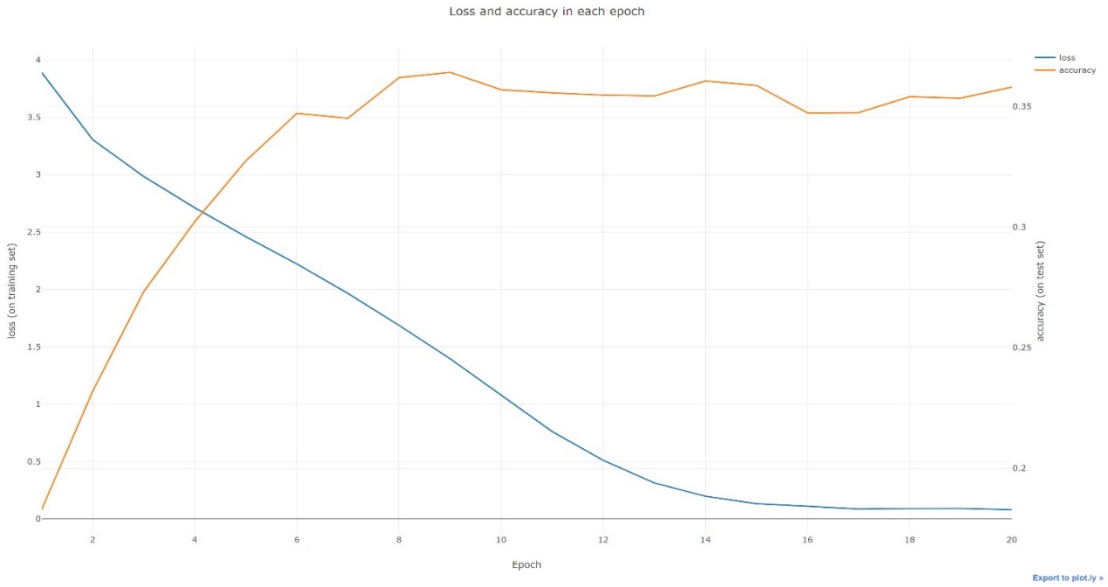
2 – Convolutional Neural Network

The Convolutional Neural Network, made of four convolutional layers (the third one followed by a pooling operation) and two linear ones, performs quite better than the traditional one.

Unlike a regular Neural Network, the layers of a Convolutional Network have neurons arranged in 3 dimensions (width, height, depth) making the explicit assumption that the inputs are images, the forward function will be more efficient to implement this way. While the convolutional layer will compute the output of neurons that are connected to local regions in the input, the pool layer will perform a downsampling operation along the spatial dimensions. The used activation function is the *ReLu*.

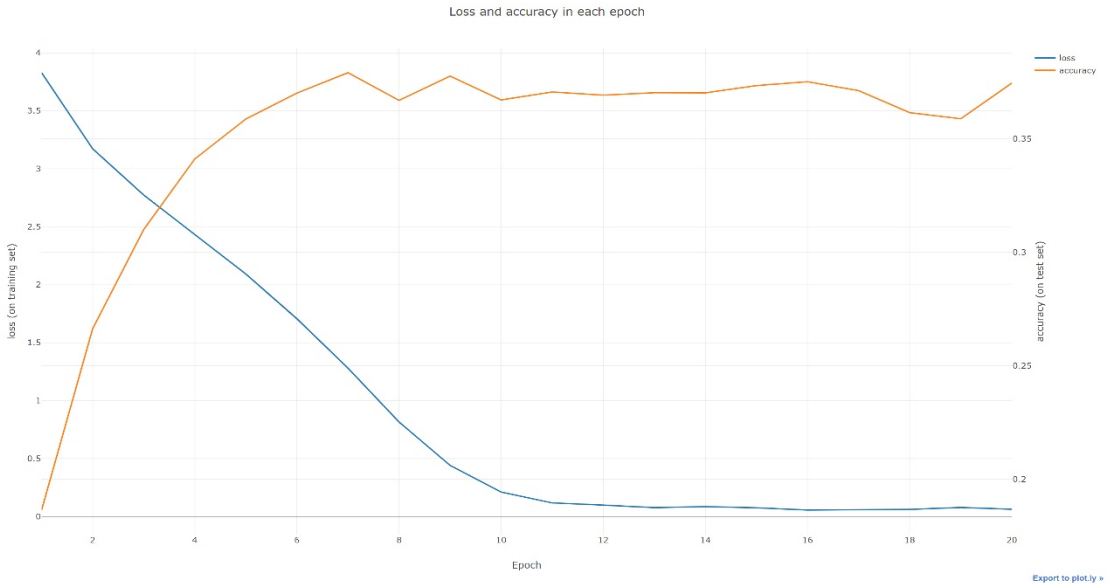


The first Convolutional Network, with filter size of the convolutional layers 32/32/32/64 scores a **30%** on the test set, with a loss of **0.7**. By increasing the filter size of the layers to 128/128/128/256 the results are even better.

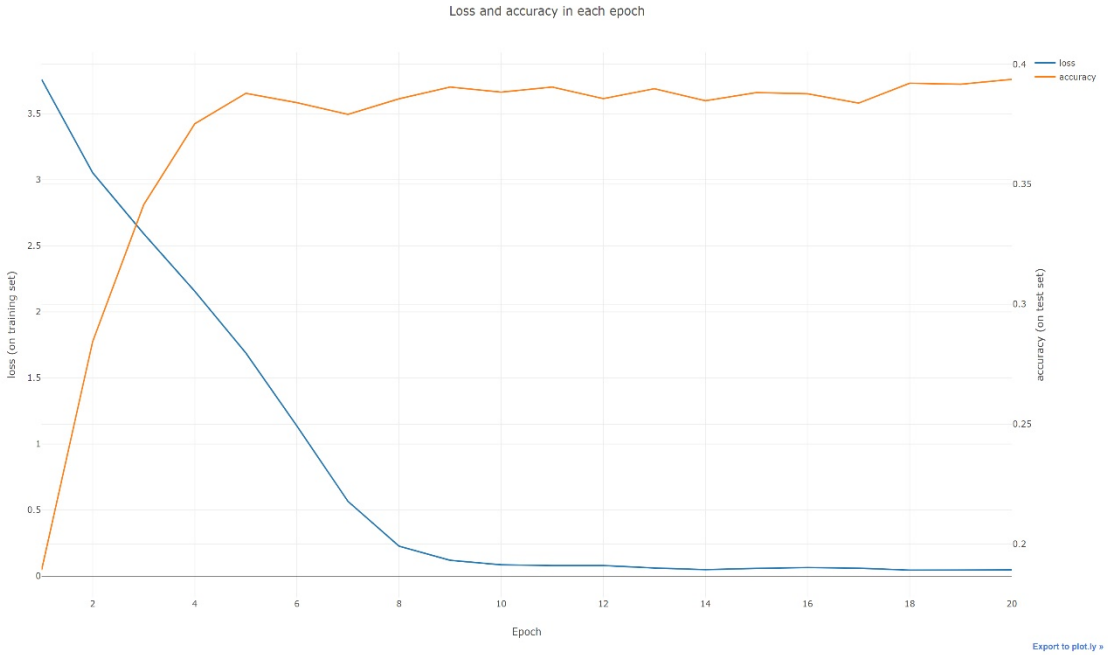


The accuracy is of **35%** and the loss of **0.08**, whit a very efficient slop towards zero. The result is somehow expected, since the filter size represents the special extent (in terms of width and height) of the connectivity between the neuron and the input volume. With a filter size of 128, each neuron of the convolutional layer will have weights to a region of [128x128x3] in the input volume. The last FC layer will compute the class scores, resulting in volume of size [1x1xNum_Classes].

However, increasing the filter size in the training phase of our model, despite massively increasing the computational time, will not score a significantly better result, as seen in the case of 256/256/256/512 filter size (accuracy **37%** and loss **0.06**)

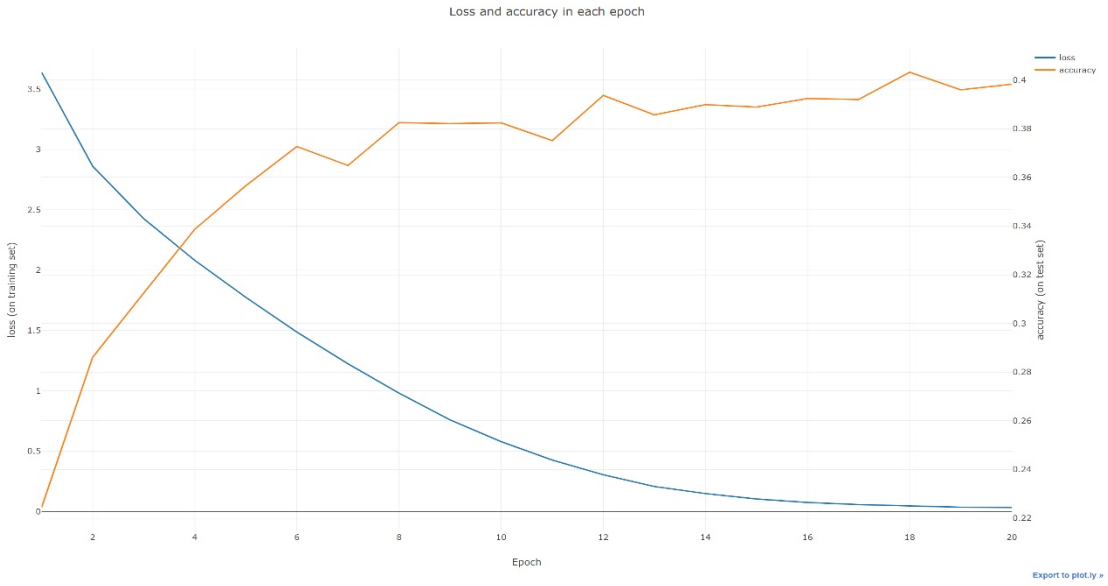


And in the case of 512/512/512/1024 filter size (accuracy **39%** and loss **0.04**). The trad off with the needed computational time is not worth it.



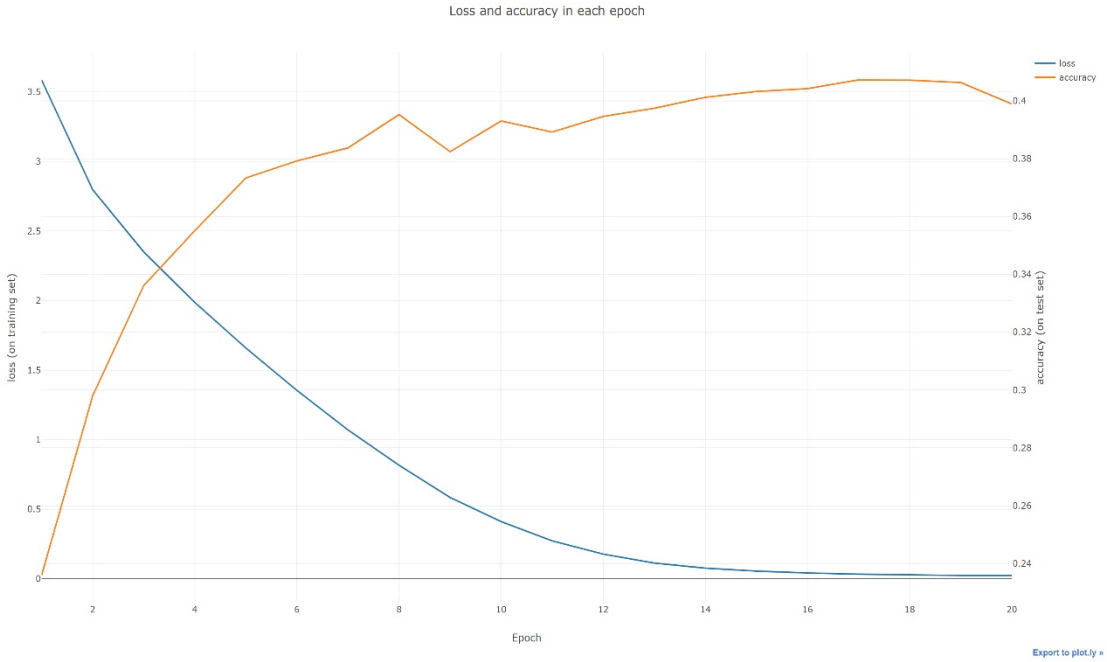
3 – Convolutional Neural Network with modification

The previous two results may suggest that better result are to be searched in other parameters or layers, and not only in the filter size. The first attempt is to introduce a Batch Normalization after each convolutional layer. Batch Normalization introduces more stability in a Neural Network by adjusting and scaling the activations, so that the network produces for any parameter an activation with the desired distribution.



With the simple introduction of Batch Normalization for each convolutional layer, the Convolutional Neural Network with filter 128/128/128/256 improves its accuracy from 35% to a value of almost **40%** and a loss of **0.03** (compared to the previous 0.08).

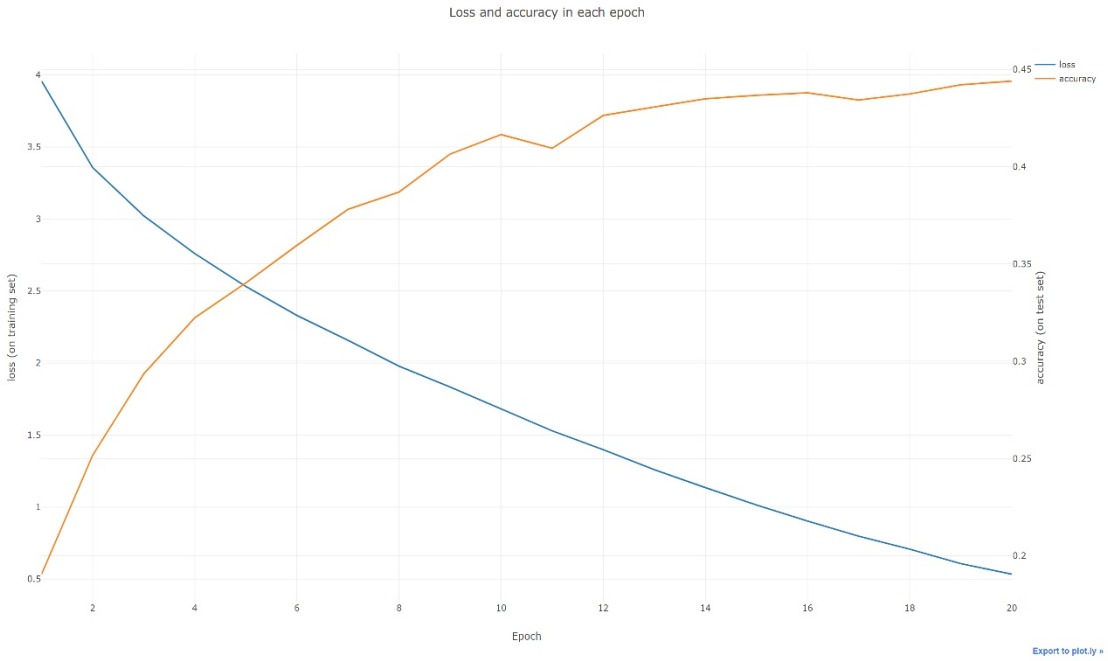
Changing the number of neurons of the first FC1 of this modified network (from 4096 to 8192), on the other hand, won't produced any useful effect.



Accuracy and loss remain the same, differently from adding a Dropout of 0.5.

A dropout layer will block the information from certain neurons at every training step, choosing those neurons with the specified probability, and prevents the dangerous effect of co-adaptation.

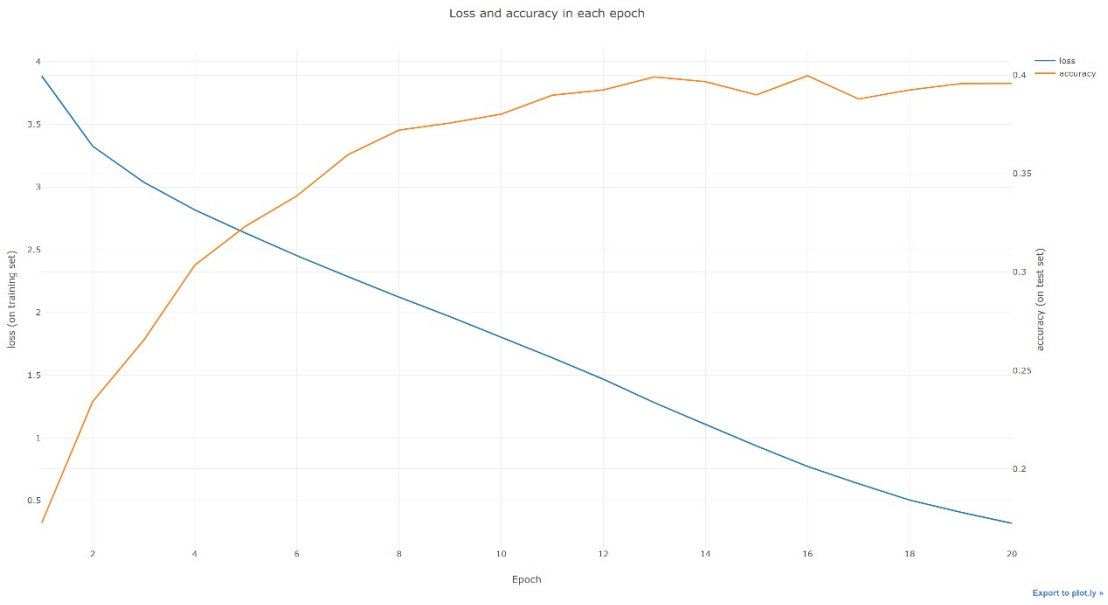
The Convolutional Neural Network, with Batch Normalization and Dropout, will achieve an accuracy of **44%** and a loss on the training set of **0.5**. This last number, while higher compared to previous ones, may have a positive meaning: the effect of the dropout layer may have prevented the overfitting of the model on the training set.



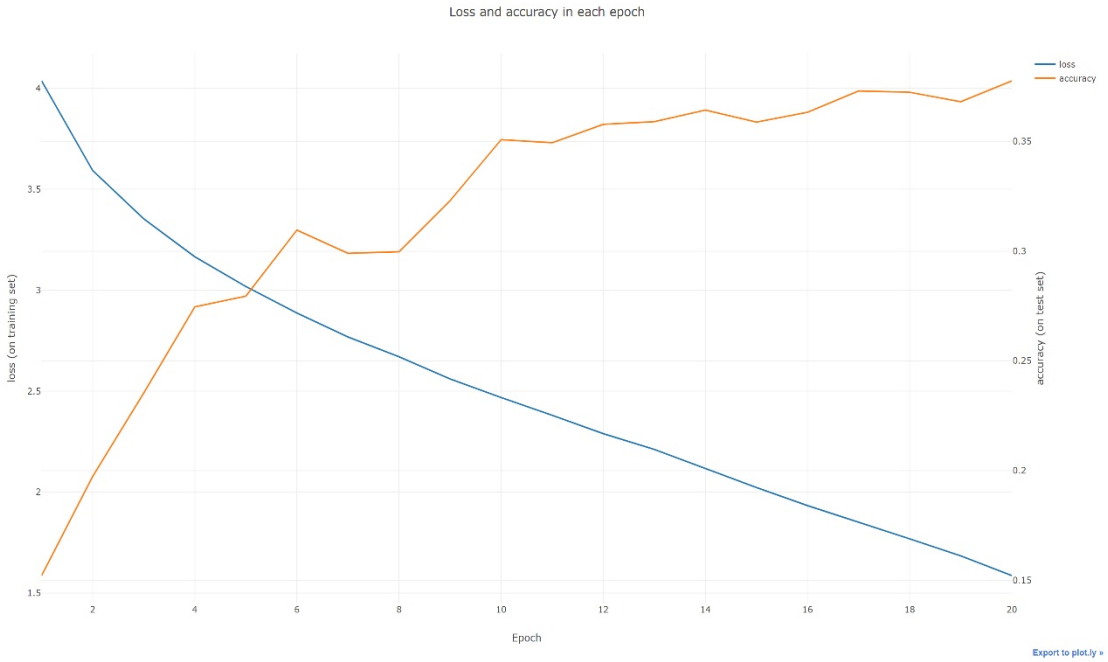
4 – Data Augmentation

Another attempt that can be made in order to increase the score of our model is to introduce data augmentation. Data augmentation is a technique that allows us to "create" more data starting from the same dataset. With some basic transformation we can create different copies of each instance within a dataset, copies which will help our network to differentiate signal from noise.

Using a random horizontal flip as transformation on the nonmodified Convolutional Network we obtain:



Whit a final accuracy on the test set of **40%** and loss **0.3**. The use of "random crop" of the images as transformation will produce slightly worse results:



Scoring a **38%** of accuracy and loss **0.37**, whit almost no performance cost.

5 – ResNet18

Finally, finetuning on our model an already pretrained ResNet18. This allows us to achieve incredible results even with a small amount of data, since we are note training the entire network, and the part trained is not trained from scratch. The pre-trained model will already have learned features that are relevant to our own classification problem.

ResNet18 scores an impressive accuracy of **80%** on the test set, with a loss on the training set of **0.045**.