# Technion - Israel Institute of Technology

Deep Learning (097200) - Hw 2 : Convolutional Neural Networks

Alon Naor, 304818735 Daniel Engelsman, 300546173

## Explaination of the Error

In the former submission we mistakenly kept optimizing the model and updated its weights during validation phase. As a result, we constantly got a fake improvement on both training and validation datasets. We once again submitting the project in attempt to show a realistic implementation.

## Summary of Attempts

In this excercise we were asked to build a CNN classifier for the CIFAR 10 dataset. Starting from the given template in the tutorials slides, the work process was conducted mainly via trial and error. Going onwards, looking for the "sensitivity points" of the model, the suspicous parameters left constant from a certain attempt (*@ pale yellow*).

| # | Layers | lr | Optimizer | Loss func. | Batch | Dropout | Stride | Kernel | Padding | Epochs | Accuracy |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 0.5 | Adam | Cross Entropy | 50 | 0.5 | *1* | 5 | 2 | 5 | 71.26 |
| 2 | 2 | 0.5 | Adam | Cross Entropy | 50 | 0.5 | 1 | 5 | 2 | 5 | 71.51 |
| 3 | 2 | 0.5 | SGD | Cross Entropy | 50 | 0.1 | 1 | 5 | 2 | 5 | 69.25 |
| 4 | 2 | 0.1 | Adam | Cross Entropy | 50 | 0.1 | 1 | 5 | 2 | 20 | 73.75 |
| 5 | 2 | 0.1 | SGD | Cross Entropy | 50 | 0.1 | 1 | 5 | 2 | 5 | 70.45 |
| 6 | 3 | 0.01 | Adam | Cross Entropy | 50 | 0.1 | 1 | 5 | 2 | 5 | 71.95 |
| 7 | 3 | 0.01 | SGD | NLLLoss | 50 | 0.1 | 1 | 5 | 2 | 10 | 71.99 |
| 8 | 3 | 0.001 | Adam | NLLLoss | 50 | 0.1 | 1 | 5 | 2 | 10 | 72.65 |
| 9 | 3 | 0.001 | Adam | NLLLoss | 50 | 0.1 | 1 | *3* | *1* | 15 | 73.33 |
| 10 | 3 | 0.001 | SGD | Cross Entropy | 50 | 0.1 | 1 | 3 | 1 | 15 | 72.11 |
| 11 | 3 | 0.001 | SGD | Cross Entropy | *100* | *0.33* | 1 | 3 | 1 | 25 | 78.11 |
| 12 | 4 | 0.001 | Adam | Cross Entropy | 100 | 0.33 | 1 | 3 | 1 | 25 | 79.55 |
| 13 | 4 | 0.001 | Adam | Cross Entropy | 100 | 0.33 | 1 | 3 | 1 | 15 | 80.22 |
| 14 | 4 | 0.001 | SGD | *NLLLoss* | 100 | 0.33 | 1 | 3 | 1 | 15 | 79.17 |
| 15 | *6* | 0.001 | *Adam* | NLLLoss | 100 | 0.33 | 1 | 3 | 1 | 25 | 81.22 |
| 16 | 6 | 0.001 | Adam | NLLLoss | 100 | 0.33 | 1 | 3 | 1 | 25 | 82.45 |
| 17 | 6 | 0.001 | Adam | NLLLoss | 100 | 0.33 | 1 | 3 | 1 | 25 | 82.37 |
| 18 | 6 | 0.001 | Adam | NLLLoss | 100 | 0.33 | 1 | 3 | 1 | 25 | 82.91 |
| 19 | 6 | 0.001 | Adam | NLLLoss | 100 | 0.33 | 1 | 3 | 1 | 50 | 82.33 |
| **20** | **6** | **0.001** | **Adam** | **NLLLoss** | **100** | **0.33** | **1** | **3** | **1** | **375** | **83.47** |

Figure 1: Table of attempts

Starting with a very basic model, comprised of only 2 convolution (=**conv**.) layers, we slowly started elaborating the model towards more layers. Eventually the "Holy grail" was found when longer epochs were taken. Since time was either a constraint, and every epoch was taken aprx. $\backsim 4$ [mins], we decided to choose a certain constellation with which we did the master train.

# The Model

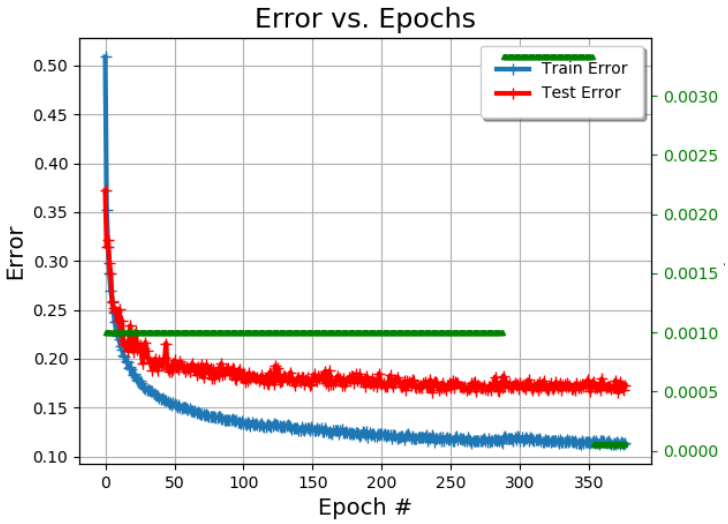
Total parameters : 46,110 Layers : 6 Stride, Padding : 1
Learning rate : 0.001 Optimizer : Adam Loss: NLLLoss

Conv2d(3-32) $\Rightarrow$ PReLU $\Rightarrow$ BatchNorm2d
Conv2d(19-19) $\Rightarrow$ PReLU $\Rightarrow$ BatchNorm2d
Conv2d(19-19) $\Rightarrow$ PReLU $\Rightarrow$ BatchNorm2d
Conv2d(19-38) $\Rightarrow$ PReLU $\Rightarrow$ BatchNorm2d $\Rightarrow$ MaxPool2d(2)
Conv2d(38-38) $\Rightarrow$ PReLU $\Rightarrow$ BatchNorm2d $\Rightarrow$ MaxPool2d(2)
Conv2d(38-38) $\Rightarrow$ PReLU $\Rightarrow$ BatchNorm2d $\Rightarrow$ MaxPool2d(2)
AvgPool(k=1,s=1) $\Rightarrow$FC(4*8*19, 10) $\Rightarrow$ DropOut(p=0.33) $\Rightarrow$ LogSoftMax(1)

Each layer is computed on a 16 base (for convenience), and some has been multiplied [$\times 1, \times 2$...] to gain more depth and manage total parameters $< 50{,}000$. We chose Padding and Kernel sizes as [1, 3] such that each layers' output sizes, won't be affected.

# Performances

After not less than 375 epochs, that were combined together we manage to tune the model with a sufficient final accuracy of : **83.47** [%]. Attached here are relevant graphs :




The thick green line along stands for the *learning rate* along epochs. Loss function is highly affected by the learning since it's defining the vicinity to a local minimum. After *280* epochs we changed it from $0.001 \Rightarrow 0.0033$ in order to accelerate convergence, but without further change. Afterwards we decreased to *learning rate* $0.0033 \Rightarrow 0.0001$, and neither does here it affected.