Kuba Kowalski

# Neural Networks

Description of the project

## 1. Introduction

We are going to focus about the specific problem. Considered that having data set called CIFAR10 which contains set of labeled images, we will try to build neural net which recognizes image and gives us a correct label. Our data is 4-dimensional (*batch, channel, width, height*) and the result is 2-dimensional (*batch, index*). Theano library helps me to implement my network.

## 2. Specifics

I have implemented a few types of layers:

a) Convolutions + ReLU
I have connected this two layers and treated it like one. I am using this layer for reducing *height* and *width* in our data and increase *channel* coordinate. Thanks to ReLU we have kind of non-linear function.
Arguments (*outputChannel, size*). First is how many channels we want to have on output and the second is the size of „window" which will be looking on our image and each time it will give us one point on output.

b) Pooling
Now, this layer reduces *height* and *width* coordinates twice or more by counting a mean on adjacent points.
Arguments (*size*). It is the size of „window" of nearest neighbours.

c) Filter + ReLU
Here again I joined two layers. It flattens our data to two dimensional.
Arguments (*outputChannel*). It is the size of flattened data (all coordinate except first, because *batch* is „untouchable").

d) SoftMax
It helps to choose the best option by normalize result.

When we know that, I can show how my neural network which looks like on table 1 (let say our batch size equals 3).

## 3. Changes and files

In my project directory there are few files. Each file is a step forward to reach my final solution. The files are:

a) `Project.ipynb`
It is rewrite code from lecutre's one. Of course I have to change MINST dataset to CIFAR10.

b) `Project-Copy0.ipynb`
Here are some minor changes. First of all I changed structure of code. Here I builded my network in more objective style, using classes. Network itself changed a lot. Other changes are: decrease size of minibatch, increase weigh decay and also my learning rate smaller, but slower going down.

Tablica 1. Buliding neural net.

| layer | current shape |
|---|---|
| Start | (3, 3, 32, 32) |
| Convolutions(40, 3) | (3, 40, 30, 30) |
| Pooling(2) | (3, 40, 15, 15) |
| Convolutions(60, 4) | (3, 60, 12, 12) |
| Pooling(2) | (3, 60, 6, 6) |
| Convolutions(90, 3) | (3, 90, 4, 4) |
| Filter(1500) | (3, 1500) |
| Filter(600) | (3, 600) |
| Filter(10) | (3, 10) |
| SoftMax() | (3, 10) |

c) `Project-Copy1.ipynb`
Network from previous file was pretty good, but I have to revert other changes. Learning rate now is the same as it was at the beginning and size of minibatch now equals 100.

d) `Project-Copy2.ipynb`
Here I increased a lot learning rate

e) `Project-Copy3.ipynb`
All the output channels in Convolutions layer was doubled.

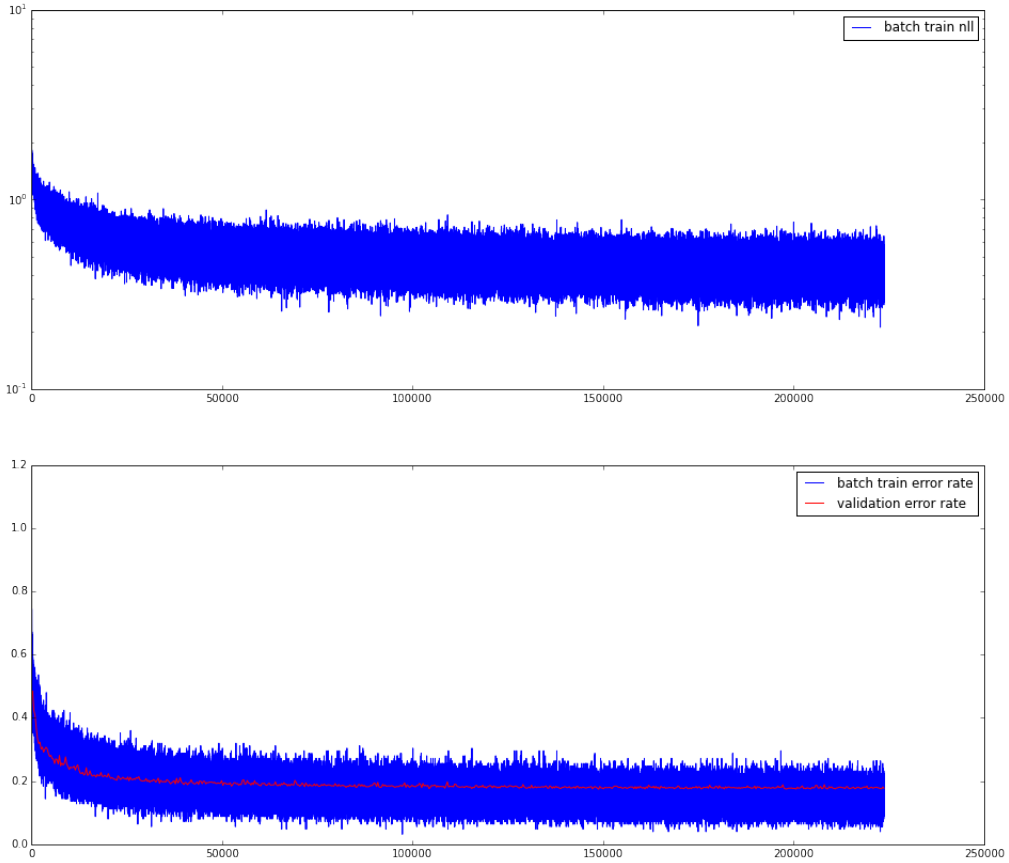
f) `Project-Copy4.ipynb`
All the output channels in Filter layer was also increased.

g) `Project-Copy5.ipynb`
I decided to go all the way and increase every thing. It means minibatch size, size of output channels in Convolutions layer, but learning rate is a little bit smaller.

g) `ProjectFinal.ipynb`
All previous changes improve my solution by max 0.5%. That is not bad, but here is the really good one. I was able to implement this thanks to Michał Łowicki. The idea is simple, every time we pass image through my network, I shift it by some random number of pixels with rolling. It helps to generalize our model. I have to also speed up learning rate, otherwise the learning took forever.

## 4. Results

In conclusion, my network works pretty good, it is quite simple, but works very slow. If there is something that I can work further on my project, this is it. However, I am patient and satisfied. Especially, I am glad to implement this last change, it helps me a lot. On image 1 you can see how error rate evaluate during the learning. My final success rate: **81.66%**.


batch train nll
$10^{1}$
$10^{0}$
$10^{-1}$
0
50000
100000
150000
200000
250000
batch train error rate
validation error rate
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000

Rysunek 1.