As final project of NN I design and implement neural network which aims in recognizing pictures from cifar10 database. Cifar-10 consists of 60000 images. Each picture has 32x32 pixels with RGB colors, and belongs to one of 10 categories (cars, lorries, ships, planes, dogs, cats, deers, birds, frogs and horses). The net is expected to reach 75% success rate. As input my network take a four dimensional tensor (RGB color, height coordinate, width coordinate, sample).

I do my best to make my model as easy to configure as it is possible. To build proper network we just need to choose number and types of layers, amount of neurons in layer, and regularize few parameters. It comes to, create a list of layers and pass proper parameter to network builder.

Net can be build of following layers

- Convolutional layer – very useful for picture recognizing. It enables network to find patterns between adjacent inputs. Contiguous pixels are mashed up with each other, with some NxN filter, so the output pixel contains parts of whole bunch of pixels
- Maxpool – divides the input image into a set of non-overlapping rectangles and for each such sub-region, outputs the maximum value.
- AffineLayer – workhorse of my network. It consists of multiple neurons computing output signal as linear combination of inputs
- Batch normalization - the distribution of each layer's inputs changes during training. This type of layer is intended to normalize inputs of layer
- TanH, ReLu, Sigmoid – activation functions which are placed at the top of AffineLayer
- Softmax layer – is an outpu tlayer placed at the top of last affine layer. It computes probabilities of belonging image to each class.

To improve learning ability of network it is possible to regularize training by following parameters

- initial learning rate (alpha)– it describes how quickly network parameters are updated. Actual learning rate for each BGD iteration equals alpha * 0.99993^i
- Momentum ratio 0.9 (lambda) – parameter which is responsible for which part of previous gradient will be added to current change
- weight decay ratio (gamma) – describes penalty of weight increase

Input set is divide into three subsets. Train subset contains 40000 images divide into 400 minibatches, 100 images each. It is used to train my network. I'm training my model with regard to batch gradient descent principal. After each mini-batch network updates its parameters. After running training on all mini-batches, validation process starts. Net uses 10000 images validation subset to check accuracy, gained after training epoch. When success rate is sufficient, net use 10000 pictures test subset to check overall accuracy.

Final configuration of my net contains two convolutional layers with maxpooling placed over each. Further comes an affine layer of 1000 neurons, with ReLu layer as activation function. Last layer is another affine layer with softmax. Moreover I use dropouts with 0.3 dropout rate to avoid variation of my net. I resign from using batch normalization because, it wouldn't work properly when used more then one time. I wasn't able to find the bug.

My net gained 77.5 accuracy after training.