096260 - Deep Learning

Homework 2

Dmitry Gaidar – 314052903

Meghan Mergui - 342397478

Henrique Lispector – 990512212

1. Model architecture description and training procedure.

For our model, we tried three different architectures: n2, n3, and n4. Architectures n1 and n5 can be seen commented in the code, but were not used in the end.

a) Model n2.

This architecture contains 15 layers plus a dropout layer in between layers 13 and 14, if the dropout probability variable is different than zero.

The first layer is a horizontal flip data augmentation layer. Through the BatchFlip function, images are horizontally flipped and no parameters are added to the network.

Then, a convolution layer is added with a 3-input image channel, 32 output channels and 5x5 convolution filter, resulting in 2,432 parameters.

The third layer is a SpatialMaxPooling layer that looks at 2x2 windows and finds the maximum value in those windows to highlight specific features in an image and down-sample the input representation. The operation on this layer does not add any parameters to the network.

The fourth layer is comprised of the ReLU activation function, which also does not add any parameters to the network.

Then, for the network to converge in a more rapid manner, a Spatial Batch Normalization layer is added with a dimensional input of 32 channels. This layer adds 64 parameters to the network.

Next, we added a spatial convolution layer with a 32-channel input image, 64 output channels and 3x3 convolution filter, which produces around 18,496 parameters.

We repeat some of the layers described above and in the end we reshape a 3D tensor of 24x4x4 into a 1D tensor of 24x4x4, add a fully connected layer, add a linear layer with 10 (# of classes we want to classify images) as the output and the final layer is a LogSoftMax that will give us the score probability of the image being in one of each of the ten categories predefined.
The total number of parameters in this neural network is 47,618.

b) Model n3:

This architecture consists of 35 layers, with a function "block" (which executes 3 layers at successively) being called 9 different times.
The block function runs a Spatial Convolution layer (with 8 parameters at most), a Spatial Batch Normalization layer and a ReLU layer respectively. The ReLU layer adds no parameters to the network and the Spatial Batch Normalization adds very little parameters to the network.

The first layer horizontally flips the images with the BatchFlip function, which does not add any parameters to the network.

Then, the block function is called three times in a row. The first block function takes in 3 input planes in the image given to the feedforward function, does a convolution with filter 5x5, step 1 for both the width and height dimension, and padding of 2 per width and height. The number of output planes is 64. The number of parameters added in this layer is 4,864. The second block function takes in 64 input planes and outputs 32 input planes doing a convolution with 1x1 filter. 2,080 parameters are produced in this layer. The third block function takes in 32 input planes and outputs 32 input planes with convolution made by a filter of 1x1 also, and it produces 1,056 parameters.

Then a 2D max-pooling operation was added through the SpatialMaxPooling function with filter 2x2, adding no parameters to the network.

A dropout layer is added if the dropout variable is different then 0.

Then another 6 block functions are executed in a similar way, but, instead of doing max pooling, a 2D average-pooling operation is done after the first 3 block functions and after the last 3 block functions.
In the end we will have 10 as the number of output planes to do the classification and the last layer will be a LogSoftMax to take care of the probability scores for each category.
The total number of parameters used in this network was 49,838.

c) Model n4

This architecture is designed in a similar way to “n3” in a sense that it also has 35 layers, but instead of using a block function, the three layers of the block function from “n3” are hardcoded and are in a different order. We start by flipping the images in the first layer with the BatchFlip function, as we did in “n2” and “n3”. Then we have 10 sets of 3 layers in the following order: SpatialConvolution, ReLU and SpatialBatchNormalization. In the final set, we add a SpatialMaxPooling layer in between the SpatialConvolution and ReLU. The final 3 layers are a View (reshaping 3D tensor to 1D tensor of dimension 1296), Linear (converting the number of outputs to 10, which is the number of classes we want) and LogSoftMax.
The total number of parameters used in this network was 47,234.

d) Data Augmentation.

We tried 3 types of data augmentation model: hflip, randomcrop and changing random pixel from an image. Yet, in our tests, and in our final model we are only using hflip, as it takes a lot of time already.

With hflip, we get our best accuracy of 79% in our final model, when doing it with 100 epochs and we get the final graph below.

The third data augmentation model that we tried was changing random pixels. We first select a number of pixel (0-10). Then we randomize pixel location, and we change the value of the pixel. Yet, this method worsen the model accuracy, so we decided not to use it at the end.

e) Regularization

DropOut layers were used as a regularization method. The effect was pretty little (less than 1% of improvement with great number of epochs, reduced accuracy with small number of epochs).

f) Optimization details.

We tried three different optimization models: Adam, Adagrad and SGD.

Optimization method 1: Adam

Best accuracy: 79%
Corresponding num of epochs: 100

Optimization method 2: Adagrad

Best accuracy: 69%
Corresponding num of epochs: 100

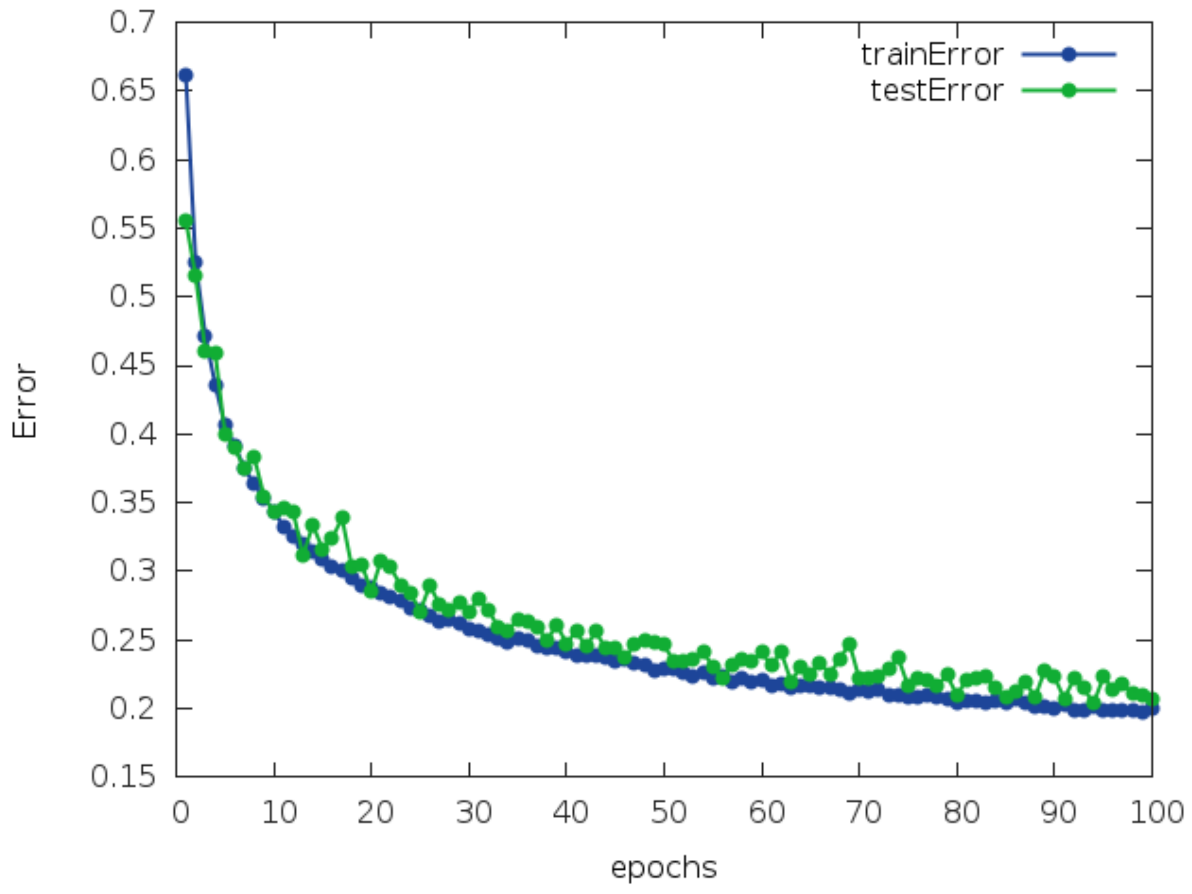
Optimization method 3: SGD

Best accuracy: 49%

Corresponding num of epochs: 100

2. Convergence graphs for final model (error and loss, function of time).

Our best model was with Adam optimization method, 100 epochs, Batch size 512, dropout 0.5, Network n3.



3. Summary of attempts and conclusions.

In order to compare different networks and methods, we have created a Perl script (CompareResultsFINALMODEL.pl) which run our models (t5.FINALMODEL.lua) with all possible combinations of predefined parameters. The best result is selected then, and the model is saved in model.t7 .

We tried to run different models, combining different Optimization Methods (Sgd, Adagrad, Adam), Num of Epochs (100, 60), dropout (0 or 0.5) and the different network N2, N3, N4, and we got the following results:

The best result (79%) we achieve with network n3, dropout(0.5) layer, batch number 512, adam optimization and 100 epochs.
We found that number of batches doesn't affect the accuracy.
We found that data augmentation doesn't significantly affect the accuracy (and significantly increases run time).
We found that adding DropOut layer slightly improves the accuracy (<1%).
After 60 epochs the accuracy almost has no improvements (when "adam" optimization is used).

4. Github link to access all the files of this homework.

https://github.com/hlispector/DeepLearning_HW2