Deep Learning Final Project Report

Group 7: Krystof Kadlec, Andrei-Cristian Baraian

The aim of this project is to train a neural network which performs a classification of 32x32 grayscale images into 9 classes. There are classes called airplane, automobile, bird, cat, deer, dog, frog, horse and ship.

Almost all the methods that we used throughout the project were usually in some way used or shown in the assignments that we completed during the course for which preparation we would like to thank - it taught us quite a lot.

**Data loading:**

Both training (36K) and validation (9K) data were downloaded using the supplied tools. We defined a class Data where the datasets (path to them to be precise) are read from a .csv file and are then returned in a form of a sample. Each sample consists of an image and a label. There is a possibility of transforming data in this class too.

**Data preparation:**

We first tried training our data with just basic transformations - normalizing to mean 0,5 and standard deviation of 0,5 and converting to tensors. This way we had only 36K training data. Later we decided to go for augmentation in order to get more robust results using RandomWindowDrop(42) and RandomHorizontalFlip, this way we got 72K training samples in total.

Both train and valid loader were prepared in batches of size 64 (the train loader with shuffling and valid loader without it) using DataLoader with 64 workers.

Later we realised that resizing our original images (32x32) to size 224x224 is quite important for the ResNet networks so we included this transformation as well.

**Network architectures:**

Overall we tried quite a few architectures. We tried a network with just 1 hidden layer to get the notebook working, the network defined in Assignment 4, VGG\_16 and couple of versions of ResNet (namely 18 and 34) with modified first layer (as our input images have just one channel compared to 3 in the original version). In the end we decided to stick with ResNet34 with block sizes 3, 4, 6 and 3.

**Training:**

We first tried training the networks on our machines. That worked with simpler networks but we soon realised that we have to train our networks using more powerful machines. For that we moved into Google Colab where there is free GPU available and performed the training there using cuda().

We were experimenting with batch sizes, learning rate, network architectures, training on data with or without augmentation, weight decay and number of epochs. Later we went for decreasing learning rate during training using a scheduler in order to fine tune in later stages of the training which turned out to be quite useful.

**Some intermediate results of our training (without scheduler):**

For ResNet18, 5 epochs, learning rate 0,01, every epoch (72K training data) took about 70s, we got validation accuracy of 65,3%. After augmenting the data, every epoch took about 90s, but we got final validation accuracy of 72,4%. When we increased the number of epochs to 20, we introduced also weight decay of 0,0001 to avoid overfitting. It happened anyway as we got final validation accuracy of about 73% whereas training accuracy close to 99%.

For ResNet34, 5 epochs and augmented data every epoch took about 130s, we got validation accuracy of 69,9%. After increasing number of epochs to 20 and introducing weight decay of 0,0003 we got validation accuracy of 73%, again overfitting occurred as training accuracy was about 94,4%.

**Final results of our training:**

We used ResNet34, batch size 64, augmented data, weight decay 0,005, momentum 0,9, learning rate was updated using a scheduler in the following way: for epochs 1-5 0,01, for epochs 6-8 0,002, for epochs 9-11 0,0004, for epochs 12-14 0,00008 and for epochs 15-30 0,000016. The training took overall about 6h on Colab GPU but there was little to none improvement in validation accuracy during last 15 epochs so training could be made shorter.

Final validation accuracy was and final testing accuracy (on the server) was 90,52%.

Here is a shareable link to our .ipynb file: <https://drive.google.com/open?id=1WzZGGVuag64icqxaQGCG7M6ikSh2JU-w>.

**Tensorboard visualizations:**