CIFAR10 classifier

Objection:

implement a classifier for the CIFAR10 dataset under a limitation of maximum 50,000 trainable parameters and achieve at least 80% accuracy on the test-set

---

The model I used for this task is Convolutional Neural Networks (CNN)

Model description:

The network is consisting of 4 main convolutional layers, where each layer includes several more convolutional layers that are not increasing the input dimension, with a fully connected layer at the end.
after each convolutional layer a normalization and activation function (ReLU) was used. when the dimension got bigger, an average pool or max pool was performed.

The network structure:

- Layer0:

$$3 \to 3$$

- Layer1:

$$3 \to 10,\ \ 10 \to 10, 10 \to 10$$

- Layer2:

$$10 \to 30, \qquad 30 \to 30 + average\ pool,\ \ 30 \to 30 + \max pool$$

- Layer3:

$$30 \to 60 + \max pool$$

The training process:

- Data augmentation
  First, in order not to deviate from 50,000 parameters and still be able to achieve good results, I increased the training set artificially. For that means, for each sample (a picture) in the training set, I added a flipped/ rotated version of it, as well as automatic augmentation.
  Overall, the train set has increased to 150,000 samples.

- Hyperparameters
  In order to achieve best possible results for the given network structure, I run the model few times, each time with different values of hyperparameters. After this process, the values of the hyperparameters I used are:

$$batch\ size\ =\ 250\ ;\ \ learning\ rate\ =\ 0.03\ ;\ epochs\ =\ 20\ ;\ dropout\ probability\ =\ 0.2$$

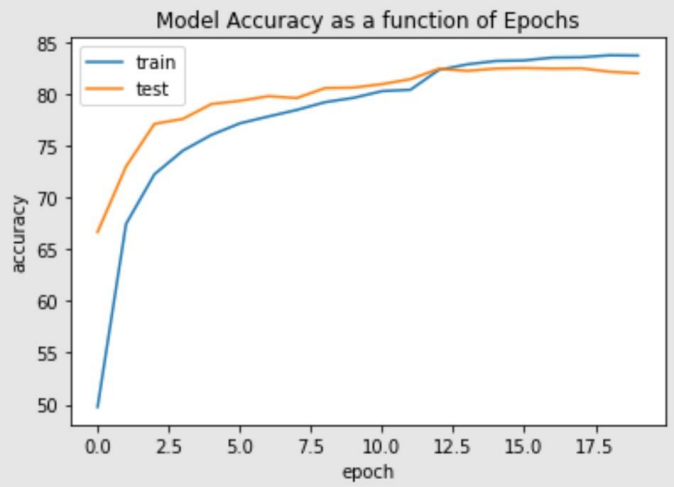
- Optimization
  The loss function I used was the negative log-likelihood loss (NLLL). After the model predicted each batch from the training set, the loss was calculated on (on this batch) and afterward the derivatives of the hyperparameters in order to decrease the loss.

Final results

Eventually, the model accuracy achieved on the test set was 82%

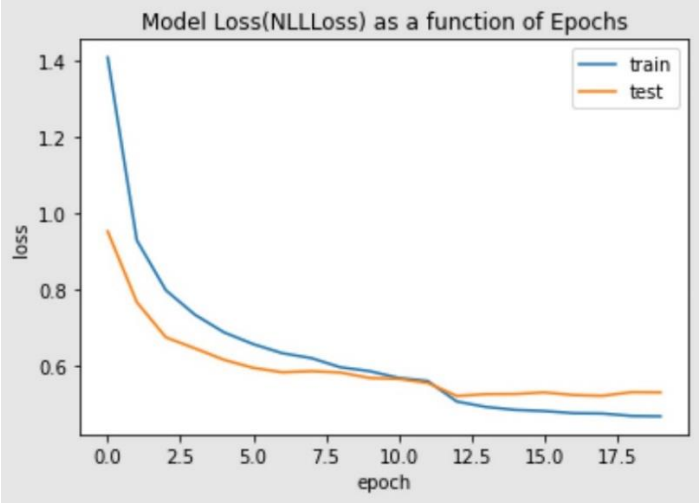
Visualization of the training process:

- Graph 1 – model accuracy at the end of each epoch, for both training set and test set



We can see that although the model didn't use the test set for fixing the hyperparameters, it still achieves better results on the test set. This probably stems from the fact that the training set has been augmented and thus is harder to predict.
We can also see that after the 12'th epoch, the model starts to overfit

- Graph 2 – the loss of the model at the end of each epoch, for both the training set and test set.



As expected, we can see the same results from the previous graph from a different angle.

Attached files:

- $train.py$ – train the model and save the final weights to pkl file
- $eval.py$ – load the final weights and test the model on the test-set
- $weights.pkl$ – final weights