Assignment 2 Report

Deep Learning – CIFAR10 Classifier

Roee Shraga & Avihay Levi

---

https://github.com/shraga89/DeepLearningCourse.git

Model Architecture:

The CNN we designed first augment the input data by flipping it horizontally in a random manner. the next layers are 3 blocks that made of: spatial convolution -> max pooling -> ReLU, with dropout and batch normalization layers in between. the convolution layers reshape the data and convert its dimension from 32 to 64 and back to 32. The last layers are 2 fully connected linear layers that reduce the dimensions from 288 (as a 1D vector) to 89 and then to 10, as the number of the classes in the data.

```
(1): nn.BatchFlip
(2): cudnn.SpatialConvolution(3 -> 32, 5x5)
(3): cudnn.SpatialMaxPooling(2x2, 2,2)
(4): cudnn.ReLU
(5): nn.SpatialBatchNormalization (4D) (32)
(6): cudnn.SpatialConvolution(32 -> 64, 3x3)
(7): cudnn.SpatialMaxPooling(2x2, 2,2)
(8): cudnn.ReLU
(9): nn.Dropout(0.300000)
(10): nn.SpatialBatchNormalization (4D) (64)
(11): cudnn.SpatialConvolution(64 -> 32, 1x1)
(12): cudnn.SpatialMaxPooling(2x2, 2,2)
(13): cudnn.ReLU
(14): nn.Dropout(0.200000)
(15): nn.View(288)
(16): nn.Linear(288 -> 89)
(17): cudnn.ReLU
(18): nn.Linear(89 -> 10)
(19): nn.LogSoftMax
```

Figure 1 - Model Architecture

We had another architecture in which we changed the last fully connected layers to a convolution layer (that keeps the dimensions the same). This change reduced the number of parameters to half of the original number. We added more depth in the form of more convolution layers, but the results we got were less accurate.

Training Procedure:

We started with normalizing our data with respect to the mean value and standard deviation of the training data. We shuffled the data to prevent bias and unbalanced results. Moreover, we read some papers that argue that shuffling methods helps to perform better.

The loss criterion we use is Class Negative Log Likelihood (ClassNLL), which is useful for classification tasks and performed better on our tests. We used **Adam** (Adaptive Moment estimation) as our gradient

descent optimization algorithm. According to Sebastian Ruder's article that gives an overview on gradient descent algorithms, Adam tends to be the best overall pick. Nevertheless, we tried SGD and AdaDelta but Adam gave us the best results.

We tuned the parameters of Adam and eventually chose: $\alpha\ (learning\ rate) = 0.003$, $\beta_1\ (first\ moment\ coefficient) = 0.9$, $\beta_2\ (seconed\ moment\ coefficient) = 0.999, \epsilon = 0.001$. We tuned the parameters based on the article that presented Adam (Kingma & Lei Ba, 2015) and based on the results we got.

The data augmentation method that worked the best for us was horizontal flip.

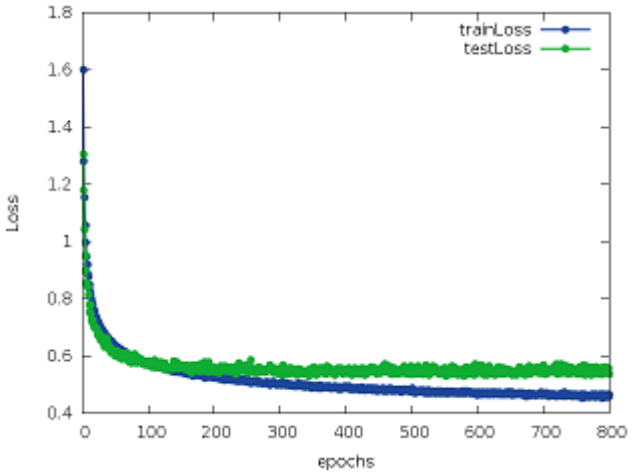
We chose a batch size of 128 and ran 800 epochs.

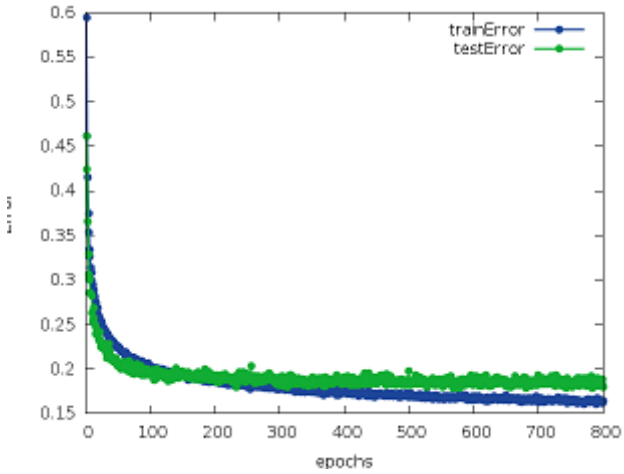
Moreover we attempted to run 5000 epochs and it slightly improved our results, so we submitted the "longer version". For readability, we used the graphs of 800 epochs.

Graphs and Results:

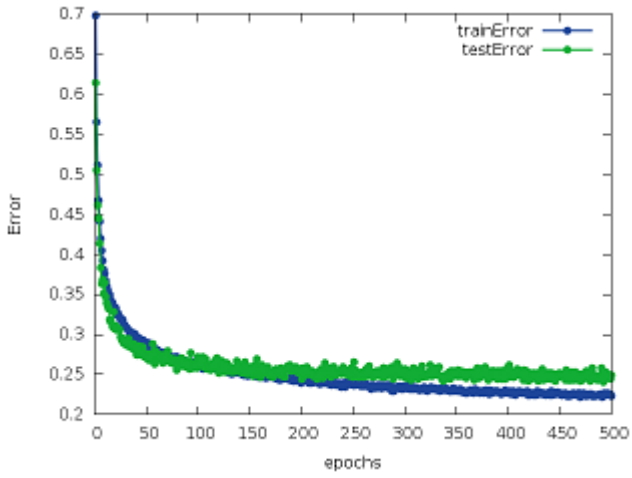
Data Augmentation Methods:

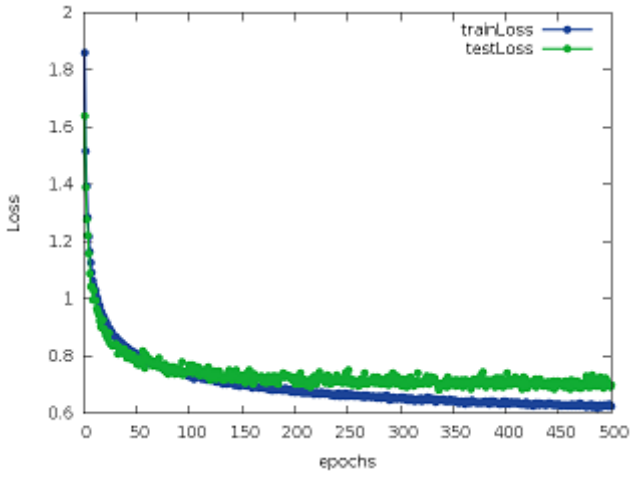
	hFlip	**vFlip**	**vFlip and hFlip comb.**
best accuracy	**82.1**	75.12	75.69
number of epochs to get the best accuracy	~2000	~200	~375



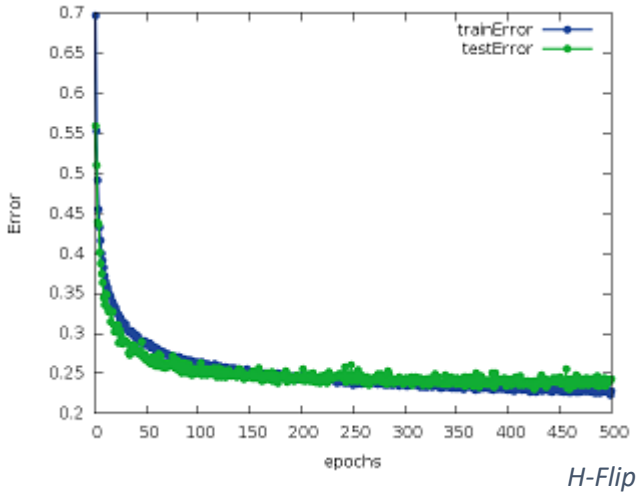


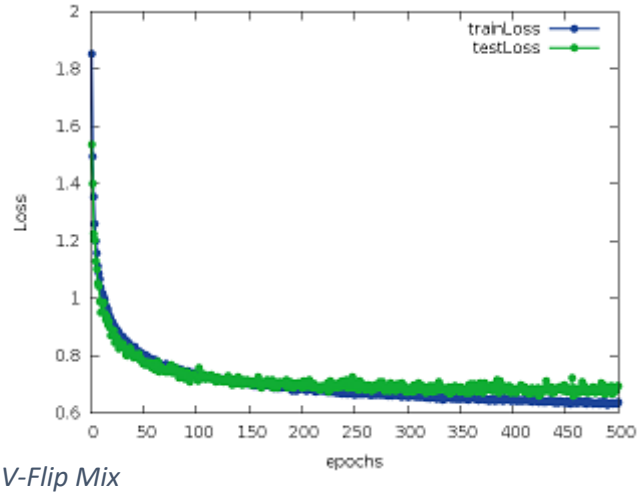
H-Flip





V-Flip

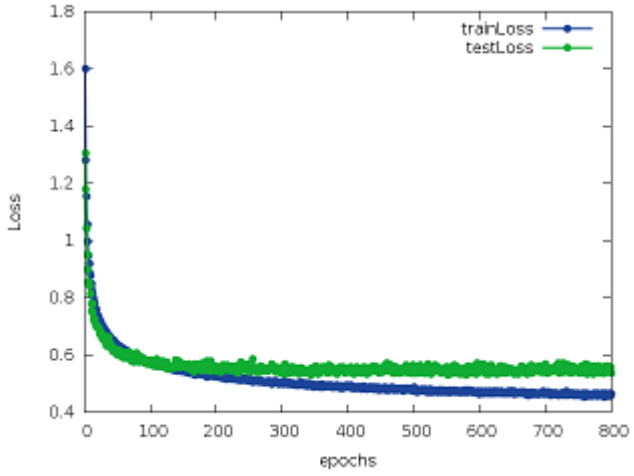


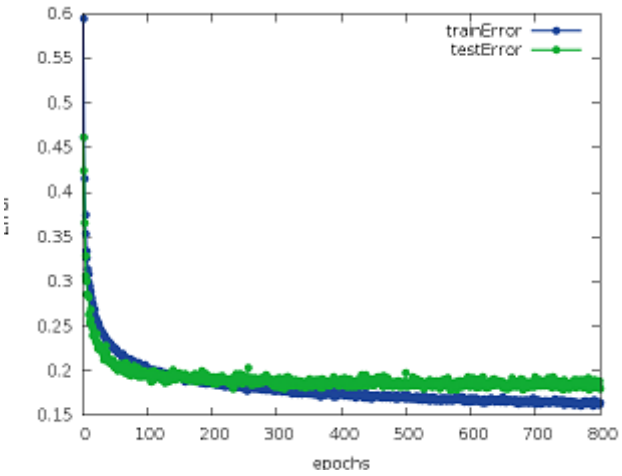


H-Flip & V-Flip Mix

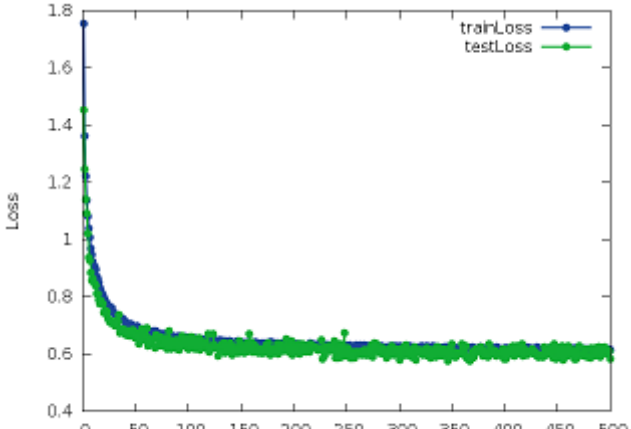
Optimization Methods:

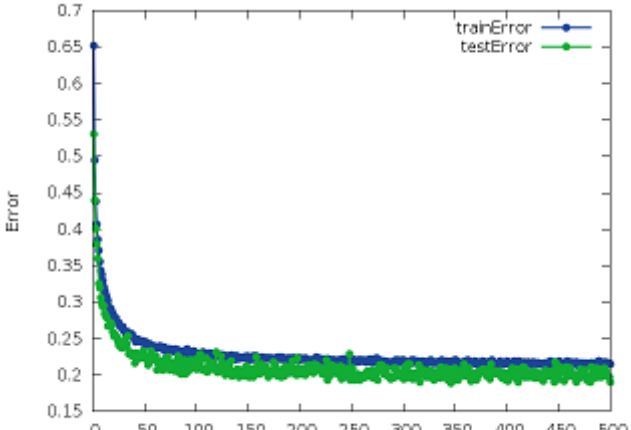
	ADAM	**SGD**
best accuracy	**82.1**	80.97
number of epochs to get the best accuracy	~500	~275





ADAM





Summary & Conclusion:

We build our model through trial and error and got to our proposed architecture. As we mentioned before, we also build a different deeper model that used a convolution layer instead of fully connected classification layers, but was less accurate.

The data augmentation method that improved our results was horizontal flip. Vertical flip had no positive effect on the results. We can suggest that flipping the pictures upside-down dose not add any useful information to the test procedure because none of the objects in the test set pictures are presented upside-down. For example: the algorithm will probably never get an input of a truck/dog upside-down, were a truck heading east or west is naturally frequent.

We can say that the parameters limit demands made this task challenging.

To further improve the performance, we chose negative log likelihood as out loss criterion (as mentioned above), Adam as our gradient descent optimization algorithm and tuned their parameters.