Final Deep Learning Project Report

1. We've created a sequence of 72 experiments, where we checked a few hyperparameters: filters_per_layer ([32], [64], [128], [256]), layers_per_block (1, 2, 3, 4) and hidden_dims ([100], [150], [200]). After the creation of matching graphs, we've looked at test accuracy graphs for most "stable" graphs where the accuracies were the highest. At the end, we chose the hyperparameters: filters_per_layer = [256], layers_per_block = 3 and hidden_dims = [150]. The accuracy on the test set of this graph reached 72.986%.
2. <u>Answers to the questions after the experiments:</u>

   A. Momentum helps to avoid local minima and helps to converge faster.

   B. Adam has two improvements comparing to the simple SGD:

   1. It uses adaptive learning rate, which means that it uses different learning rate for each parameter.
   2. It uses momentum.

   C. The best optimization algorithm is Adam with learning rate of 0.00058.

   <u>About the experiments themselves:</u>

   We've checked 5 different learning rates for Adam and for SGD with momentum (we set the momentum to be 0.9).

   For each of the optimization algorithms we commented the unnecessary line in the file experiments.py to get the right graphs.

   Then we plotted the graphs and compared them, there we saw that the best SGD with momentum learning rate was 0.0035 and reached 71.053% accuracy on the test set, and the best Adam learning rate was 0.00058 and reached 74.186% accuracy on the test set.

   Therefore, the better optimization algorithm from our results is Adam.

3. <u>Answers to the questions after the experiments:</u>
   Batch normalization is used to normalize the input to the activation function. It handles internal covariate shift, which is the change in the distribution of the input to a layer of a deep neural network. It is also used to speed up the training process. The batch normalization did improve the network performance. The best results were with the same parameters as in the previous section, but with batch normalization.
   <u>About the experiments themselves:</u>
   We edited the model.py file, there we added the batch normalization function. Then we checked some learning rates that worked the best on experiment 2.
   All accuracies got better across the board, and we got the best graph when learning rate was 0.00058, with the accuracy 78.306%.
4. <u>Answers to the questions after the experiments:</u>
   We should use regularization to prevent overfitting - by using regularization, we force the model to generalize better, and not to memorize the training data.
   The regularization affects the train accuracy and loss by reducing the train accuracy and increasing the train loss. It affects the val and test accuracy and loss by increasing the val and test accuracy and decreasing the val and test loss. The best regularization method was dropout.
   <u>About the experiments themselves:</u>
   We commented out the dropout when we measured the L2 regularization, and when we measured the results with dropout – we haven't passed the parameter weight_decay to Adam optimizer.
   After plotting the graphs, we could see that the best accuracies were achieved when we used dropout regularization – 78.533% with the learning rate 0.0005.

Summary Up to Here

After examining all results, we can conclude that the best architecture we got is:

A. Layers per block = 3.
B. Filters per layer = [256].
C. Hidden dims = [150].
D. Dropout = 0.2.
E. Adam optimizer (without L2 regularization, seems like dropout did a great regularization).
F. Batch normalization was applied.

The best accuracy on the loss set we got is 78.533%, with the architecture mentioned above.

In assignment 2 our best accuracy was about 65%, therefore we managed to better it in the newer best architecture by 14%.

The things that made our results much better is using dropout and Adam optimization – dropout made it harder for the model to memorize previous images in addition to forcing the model to not be dependent only on particular parts, whereas Adam generalized the model.

Answers to the questions after the experiments:

The pretrained ResNet50 accuracy was 97.604 and the loss is 0.125 on the test set. It is well-fit the data. It has better accuracy than the non-pretrained ResNet.

About the experiments themselves:

We run two experiments, where the first one is with pretrained=True and the second one is with pretrained=False. We can see that the first experiment has better accuracy and loss than the second one. Both are well-fit the data - the accuracies persist at roughly the same level of the train and test of each graph, even though the pretrained model scores much higher.

Project Summary

Firstly, we tested the same model from assignment 2 with the different hyper parameters: filters_per_layer, layers_per_block, hidden_dims and two probabilities for dropout. There we concluded the best parameters: [256], 3, [150], p=0.2 respectively.

After that, we tested different learning rates - SGD with momentum and Adam optimizers tested against every learning rate. We saw that the best accuracy we got was Adam with the learning rate 0.00058.

Then we added batch normalization to our model and performed experiments with it. The accuracies got better thanks to it, across all different learning rates that were tested.

Then, we added batch normalization to our model, which further improved the results.

Finally, our last experiment on our model involved testing two different regularizations – L2 and dropout. We saw that dropout did a better job.