DD2424 - Assignment 1 Bonus

Timo Nicolai

April 9, 2019

1 Improving Accuracy

I went through the list of suggested improvements and skipped the ones that would have been too computationally intensive for the hardware available to me (foremost hyperparameter search which I suspect could have had a significant impact on final classifier performance but would have taken several hours to run for a reasonable number of parameter values and learning rate decay which did not seem to have a large impact when applied during the first hundred or so epochs and would likely have required longer training and a search over the optimal learning rate schedule). I also didn't implement Xavier initialization which is not all that different from random initialization with fixed standard deviation for a single layer network.

I progressed in an incremental fashion, i.e. if a technique yielded performance improvements I used it in combination with all further techniques.

Overall, adding more training data caused the largest jump in validation set performance. Most other techniques only produced meager improvements (or none at all), maybe with the exception of ensemble learning which boosted validation set performance by about 1%.

1.1 More Training Data

One would expect large performance gains when greatly increasing the amount of training data and indeed, using all training samples (except 1000 for the validation set) and training with the same training parameters as before ($\eta = 0.01$, batch size 100, 40 epochs), the network achieves a total classification accuracy of around 41.4% on the validation set. Figure 1 shows the corresponding learning curves and Figure 2 shows the validation set confusion matrix (I only evaluated performance on the test set again once I had tested all the difference improvements so as not to include or discard them based on test set performance).

1.2 Early Stopping

I implemented early stopping as follows: I stopped the training once the validation cost did not improve over its previous best value for a certain number of consecutive epochs (here I choose 10). The final network parameters were then

chosen as those that achieved the best validation set **accuracy** at the end of some epoch (accuracy is not exactly "smooth" across epochs and thus it makes more sense to use the cost to determine when to stop training).

Figure 3 shows learning curves for this case, the training terminated after 91 epochs. The fact that the validation error flatlines relatively early might suggest that fine-tuning of the learning rate and the regularization parameter is in order but as mentioned before I did not perform any sort of gridsearch in the interest of time.

The best performing network parameters yielded only a slight improvement of around 0.5% (see Figure 4) which I suspect might not correlate with increased test set performance. Because the best accuracy occurred at around 20 epochs I did not stop training prematurely in all further experiments but still always chose the intermediate network parameters.

1.3 Data Augmentation

To keep things simple I applied small random variations to brightness, contrast and saturation to each of the images in the training set. Figure 5 shows some of the original images from the training set and Figure 6 shows the same images after applying random augmentations (the results are somewhat subtle in some cases).

I did however not do this on a per batch basis (that would have been prohibitively expensive). Instead I randomly transformed each image once and added the result of this transformation to the training set, effectively doubling the size of the training set. Unfortunately this did not improve final validation accuracy at all, as show in Figure 7. I did not use data augmentation while training the final network.

1.4 Random Shuffling

Random shuffling of the training data mainly resulted in a large jitter of the learning curves but no perceptible improvement in their overall trends as can be seen in Figure 8. I still decided to use shuffling because I suspected it might improve performance of an ensemble classifier (see next section).

1.5 Ensemble Learning

For the final classifier I trained ten networks with different random parameter initializations on different training sets obtained via bagging (sampling with replacement from the original training set), I randomly shuffled the training data at the beginning of each epoch and chose the best intermediate parameters after 40 epochs of training for each weak learner. Figure 9 shows the performance of this ensemble on the validation set which is 1% higher than before. The final performance this ensemble achieves on the test set is shown in Figure 10. At around 40% it is sadly a lot lower than the validation set performance albeit still higher than the result before optimization.

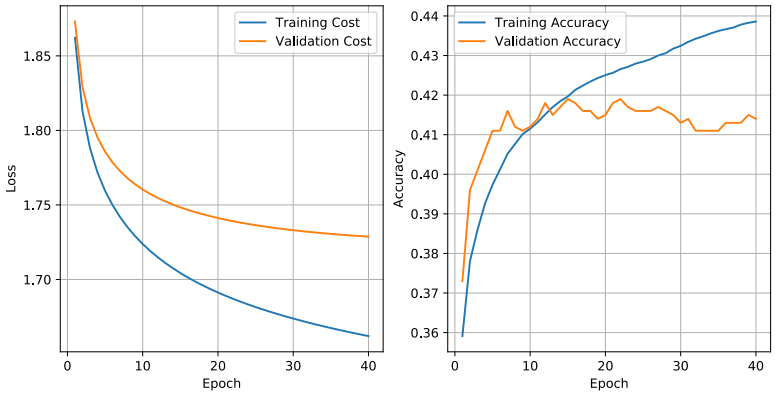

Figure 1: Learning curves for full training set.

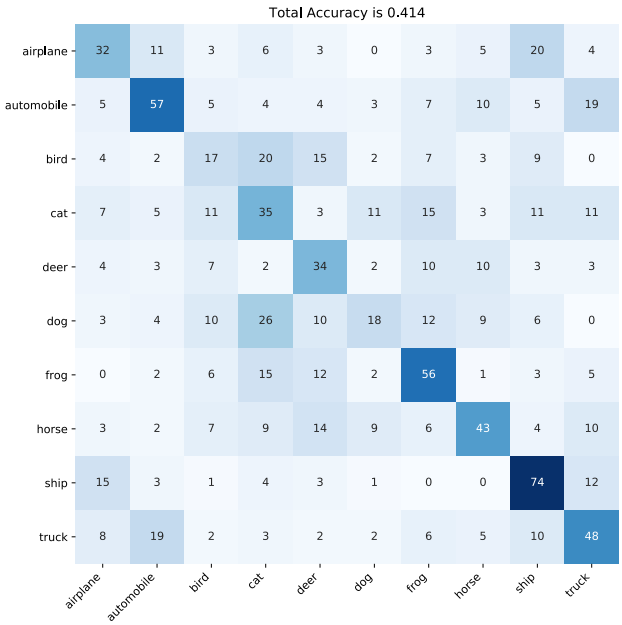

Figure 2: Confusion matrix for full training set.

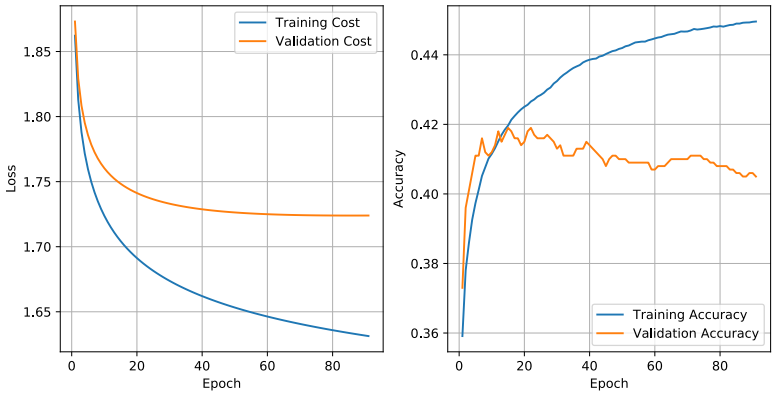

Figure 3: Learning curves for full training set with early stopping.

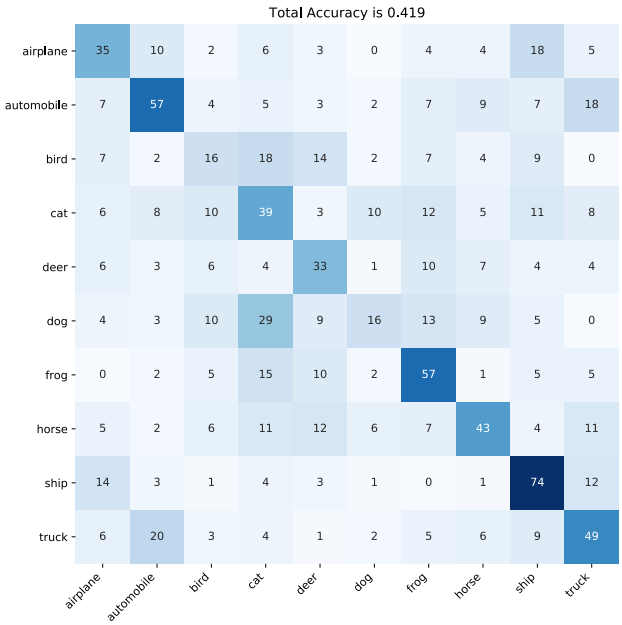

Figure 4: Confusion matrix for full training set with early stopping.

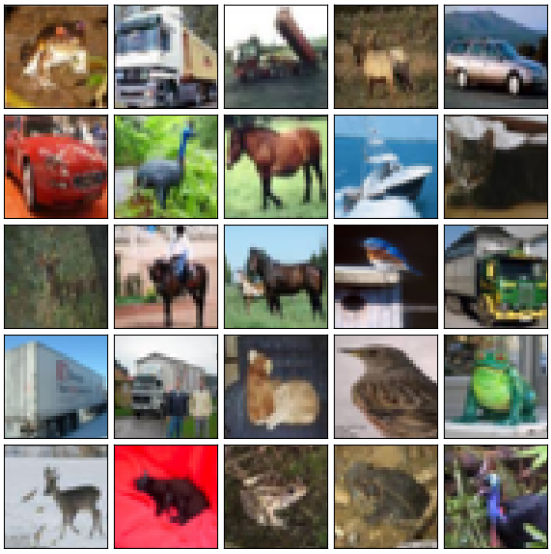

Figure 5: Some unaugmented training samples.

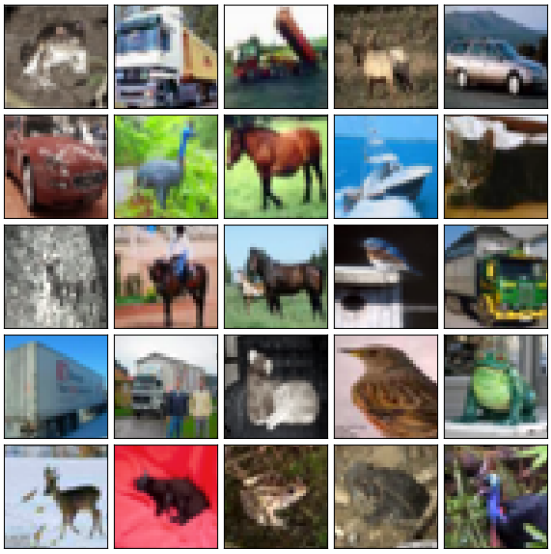

Figure 6: Training samples from Figure 7 after applying random augmentations.

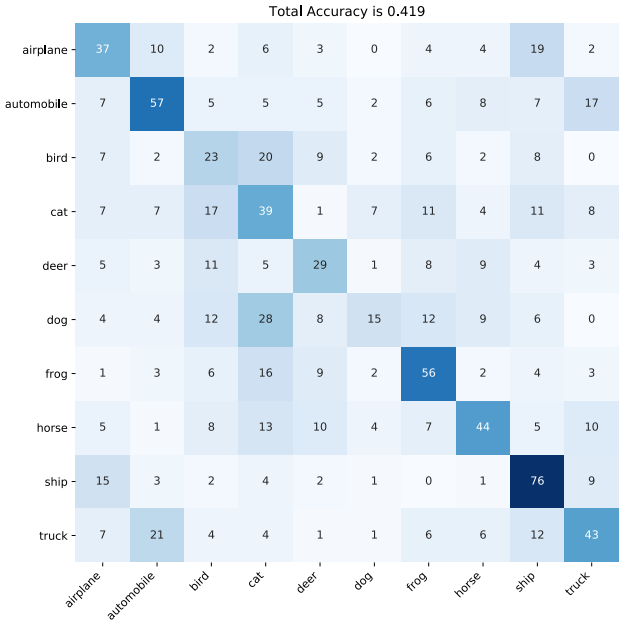

Figure 7: Confusion matrix for a network trained with augmented training data.

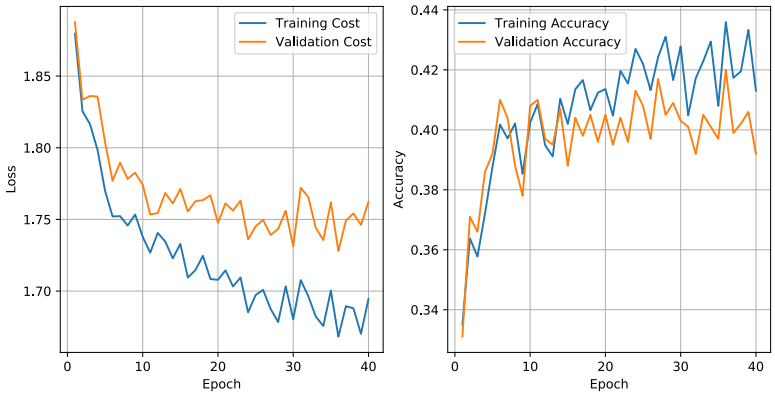

Figure 8: Learning curves for full training with random shuffling between epochs.

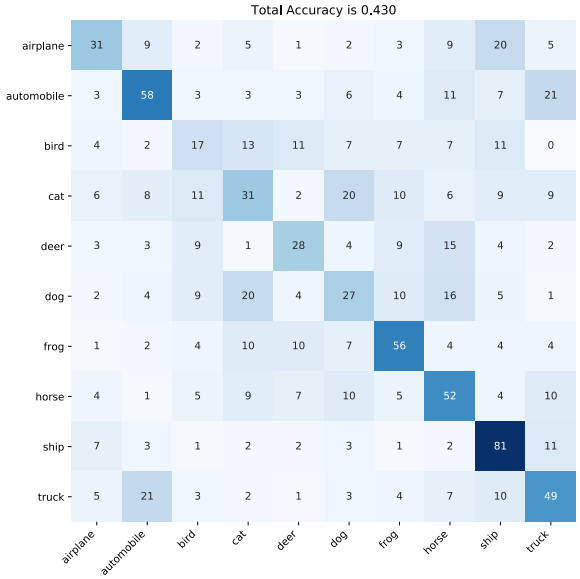

Figure 9: Confusion matrix for an ensemble of 10 networks trained via bagging (validation set).

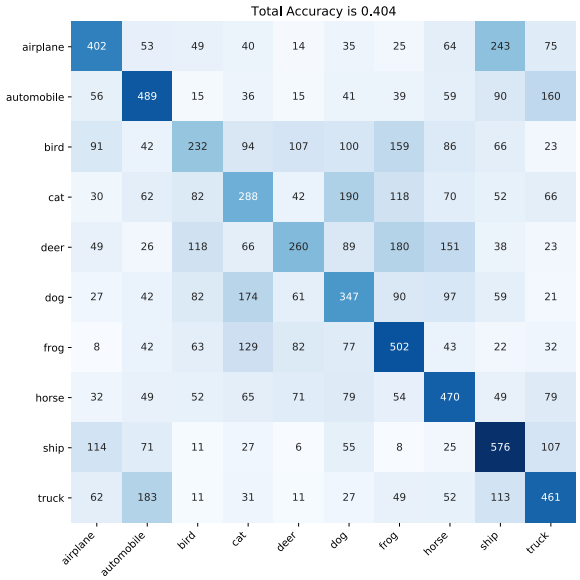

Figure 10: Confusion matrix for an ensemble of 10 networks trained via bagging (test set).

data dimensions	batch size	λ	$\epsilon_{rel,max}$
20	1	0	3.66e-9
20	1	0.5	1.57e-6
20	20	0	4.56-7
20	20	0.5	4.35e-4

Figure 11: Maximum relative SVM gradient error between analytical and numerical gradient. Maximum taken over all elements of a gradient matrix resulting from the same random weight matrix initialization. Numerical gradient obtained via simple forward difference with $h = 1e - 6$.

2 SVM Loss

For this part I dropped the bias vector and instead appended an extra row of ones to all input data matrices (and a corresponding extra column to the weight matrix). This simplifies the gradient computations. My implementation follows the following analytical formula for the weight matrix gradient:

$$\frac{\partial l_{SVM,i}}{\partial w_j} = \begin{cases} -\left(\sum_{j \neq y_i} \mathbb{1}(w_j^T x_i - w_{y_i}^T x_i + 1 > 0)\right) \cdot x_i & \text{if } j = y_i \\ \mathbb{1}(w_j^T x_i - w_{y_i}^T x_i + 1 > 0) \cdot x_i & \text{if } j \neq y_i \end{cases}$$

I was able to implement this in an efficient vectorized form on a per-batch basis

I verified the correctness of my implementation as before by comparing analytical and numerical gradients for different batch size and regularization parameters. Figure 11 displays the maximum relative error for several of these cases. These values are somewhat low across the board and so I am certain that my implementation functions correctly.

I trained four different networks with the SVM loss using similar setups as during the initial training runs with the cross entropy loss. I had to lower the learning rate to achieve good results, $\eta = 0.01$ already caused the training to diverge as can be seen in Figure 12.

Figure 13 displays final classification accuracy on the test set and confusion matrices for these four networks. Using the SVM loss did not yield an improvement over the accuracy achieved with the cross entropy loss for the respective best networks. At the same time, increased regularization proved to be slightly less detrimental to classification accuracy when the SVM loss was used.

Figure 14 shows a more direct comparison of final test set accuracies for all networks trained with sensible parameters.

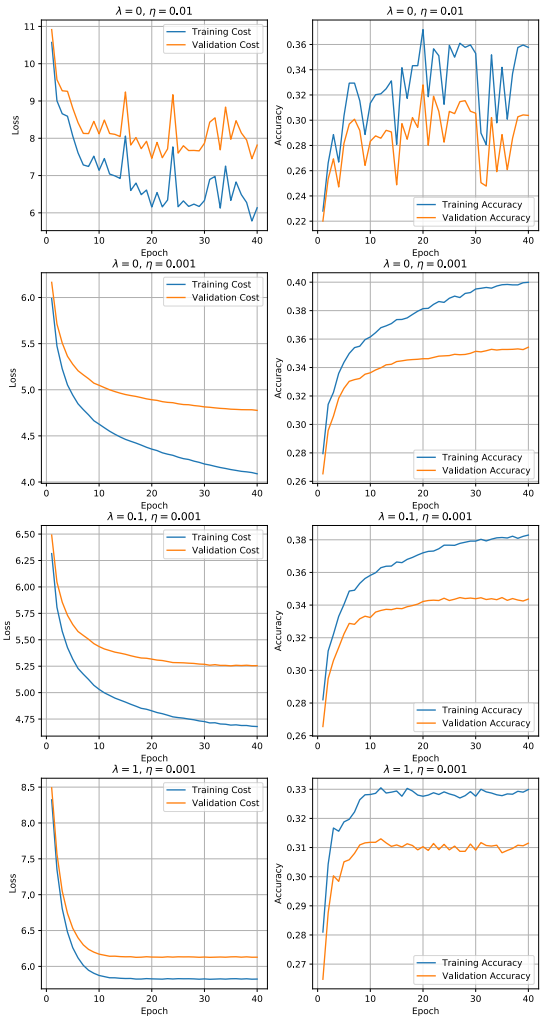

Figure 12: Learning curves for networks training with SVM loss and different learning rates and regularization parameters.

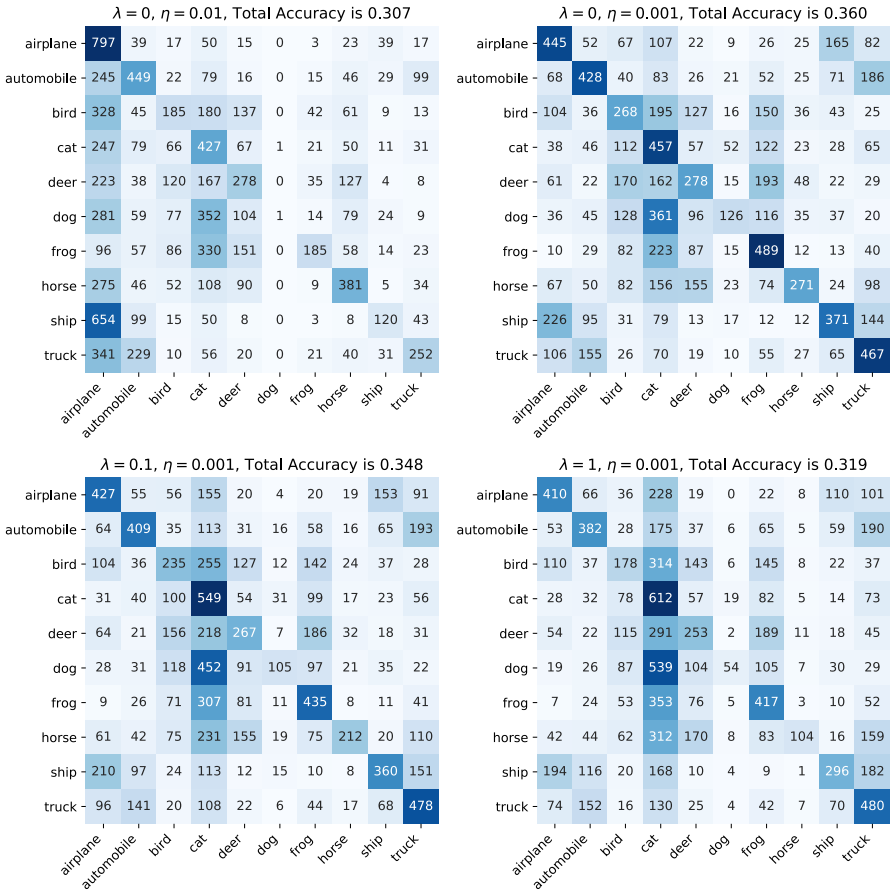

Figure 13: Test set confusion matrices corresponding to the networks from Figure 12.

Cross Entropy Loss

η	λ	Accuracy
0.01	0	0.367
0.01	0.1	0.334
0.01	1	0.219

SVM loss

η	λ	Accuracy
0.001	0	0.360
0.001	0.1	0.353
0.001	1	0.332

Figure 14: Comparison of achieved test set accuracy for networks trained with cross entropy and SVM loss.