I tested implementing the a), b), d) and e) improvements. I first tested these specifically to know how much they improved the network specifically. I used the parameters I got from the best model in the previous exercise, which had the best accuracy:

$\lambda = 0, \mathrm{N}_{\mathrm{Epochs}} = 40, \mathrm{N}_{\mathrm{Batch\ size}} = 100, \eta = 0.01.$

Training Accuracy	Validation Accuracy	Testing Accuracy
0.4379	0.4100	0.4094

Table 2.a) Used all training data and cut of the last 1000 pictures for validation.

Training Accuracy	Validation Accuracy	Testing Accuracy
0.4041	0.3666	0.3764

Table 3.b) Used the same amount of data from exercise 1 and ran for more epochs = 100.

Training Accuracy	Validation Accuracy	Testing Accuracy
0.4032	0.3690	0.3782

Table 4.d) Decayed learning rate with factor 0.9 for each 40 epochs. Used the same amount of data from exercise 1

Training Accuracy	Validation Accuracy	Testing Accuracy
0.4147	0.3629	0.3697

Table 5.e) Xavier initialization of the weight. Used the same amount of data from exercise 1.

By using Xavier initialization, we make sure that the weights are not too small but not too big to diminish the effect of vanishing activations and stops exploding activations. The formula states $var(\mathrm{W}) = 1/\mathrm{input\_nodes}$, thus I multiplied with the factor $1/\sqrt{3027}$ to the variance of the initialization of the weight matrix.

	Training Accuracy	Validation Accuracy	Testing Accuracy
Best model from Excersice 1	0.4167	0.3634	0.3667
With all improvements a), b),d), e)	0.4160	0.4090	0.4018

Table 6. Final accuracy comparison between the first model and with improvements

Adding each improvement, specifically increased the test accuracy compared to the best model from exercise 1. And with a combination of all the improvement showed a test accuracy of approximately 40%. Notice that the training, validation and test accuracy are not so much different, which indicates that the improved model was not overfitted.