Lab 1 - Extra points

Victor Stenmark (vstenm)

April 2, 2024

Contents

1 Improve performance of the network

1.1 Increased amount of training data

I followed the instructions and used 49000 images for training data and only 1000 images for validation data. Using the parameters in table 1 resulted in a test accuracy of 41.29%.

llambda	= 0
n_epochs	= 40
eta	= 0.001
batch_size	= 100

Table 1: Parameters used for the run with the increased amount of training data.

1.2 Grid search

I ran grid search for three values for each of the three parameters. The specific values can be found in table 2. The increased amount of training data was also used here.

lambda:	[0.1, 0.01, 0.001]
eta:	[0.001, 0.005, 0.01]
batch_size:	[5, 50, 100]

Table 2: Values tested in grid search.

The results of the grid search can be found in table 3. All runs were rather close but the one combination that yielded the greatest validation accuracy on the last epoch was $\lambda = 0.01$, $\eta = 0.001$, and batch_size = 100.

Validation acc	**Testing acc**	**lambda λ**	**eta η**	**Batch Size**	**Num Epochs**
0.428	0.413	0.01	0.001	100	40
0.425	0.41	0.1	0.005	100	40
0.425	0.4082	0.1	0.001	50	40
0.423	0.4115	0.1	0.005	50	40
0.423	0.4099	0.01	0.001	50	40
0.423	0.4092	0.1	0.001	100	40
0.422	0.4108	0.1	0.01	100	40
0.419	0.4161	0.01	0.005	50	40
0.419	0.4134	0.01	0.005	100	40
0.419	0.4115	0.001	0.001	100	40
0.417	0.4091	0.001	0.001	50	40
0.413	0.3921	0.01	0.01	5	40
0.412	0.4131	0.01	0.01	50	40
0.411	0.4139	0.01	0.01	100	40
0.411	0.41	0.001	0.005	50	40
0.411	0.4068	0.1	0.01	50	40
0.41	0.403	0.001	0.001	5	40
0.41	0.4021	0.1	0.001	5	40
0.406	0.4063	0.001	0.01	50	40
0.406	0.386	0.001	0.005	5	40
0.404	0.4118	0.001	0.005	100	40
0.402	0.4087	0.001	0.01	100	40
0.402	0.3975	0.01	0.001	5	40
0.402	0.384	0.1	0.005	5	40
0.397	0.4017	0.01	0.005	5	40
0.39	0.3944	0.1	0.01	5	40
0.384	0.3834	0.001	0.01	5	40

Table 3: Results from grid search sorted by validation acccuracy on the last epoch.

1.3 Step decay

Playing around with step decay, I found that decaying the learning rate after only a few epochs would cause the loss to plateau too early. This led to worse model performance. Having it set to a higher number like 20 or 30 resulted in a gradual loss until epoch 20 or 30 and then a complete plateau. Overall, I did not get any peformance gains for using step decay on its own. The best results came however for $n = 30$ where the accuracy on the test dataset were 39.04% using the parameters in table 1 but only using one batch for training data.

1.4 Network with the best test accuracy

Overall, the network with the best test accuracy had an accuracy of 41.33%. This network used the increased amount of training data, the parameters from the first row of table 3, and a step decay after 30 epochs. It's clear looking at this that it was primarily the added training data that lead to the performance bump. Grid search and step decay did not contribute much if anything at all.

2 Multiple binary cross-entropy loss

2.1 Derivative of $\partial l_{\textbf{multiple bce}}/\partial s$

$$l_{\text{multiple bce}} = -\frac{1}{K}\sum_{1}^{K}[(1-y_k)\log(1-p_k) + y_k\log(p_k)]$$

$$p = \sigma(s) = \frac{1}{1+\exp(s)} = \frac{\exp(s)}{\exp(s)+1}$$

Since $p = \sigma(s)$, we have a composite function and need to apply the chain rule. For simplicity's sake, let's start with calculating the derivative of $\partial l_{\text{multiple bce}}/\partial s_k$ where s_k is the k:th element in s. Then from that we can construct $\partial l_{\text{multiple bce}}/\partial s$.

$$\frac{\partial l_{\text{multiple bce}}}{\partial s_k} = \frac{\partial l_{\text{multiple bce}}}{\partial p_k} * \frac{\partial p_k}{\partial s_k}$$

$$\frac{\partial p_k}{\partial s_k} = \frac{(e^{s_k}+1)*e^{s_k} - e^{s_k}*e^{s_k}}{(e^{s_k}+1)^2} = \frac{e^{s_k}}{(e^{s_k}+1)^2}$$

$$\frac{\partial l_{\text{multiple bce}}}{\partial p_k} = -\frac{1}{K}((-1)*(1-y_k)*\frac{1}{1-p_k} + \frac{y_k}{p_k}) = -\frac{1}{K}((y_k-1)*\frac{1}{1-p_k} + \frac{y_k}{p_k})$$

$$\frac{\partial l_{\text{multiple bce}}}{\partial s_k} = -\frac{1}{K}((y_k-1)*\frac{1}{1-p_k} + \frac{y_k}{p_k}) * \frac{e^{s_k}}{(e^{s_k}+1)^2}$$

Now when we have the expression for $\frac{\partial l_{\text{multiple bce}}}{\partial s_k}$, we get the correct expresison with

$$\frac{\partial l_{\text{multiple bce}}}{\partial s} = (-\frac{1}{K}((y_0-1)*\frac{1}{1-p_0} + \frac{y_0}{p_0})*\frac{e^{s_k}}{(e^{s_k}+1)^2}, ..., -\frac{1}{K}((y_K-1)*\frac{1}{1-p_K} + \frac{y_K}{p_K})*\frac{e^{s_K}}{(e^{s_K}+1)^2})$$

2.2 Final test accuracy

Using the parameters in table 4 and the same training and test data as in the first part, the network got an accuracy of 38.37 %.

llambda = 0.0
n_epochs = 40
num_batches = 100
eta = 0.02

Table 4: Parameters for multiple binary cross entropy run

2.3 Histogram plot

The plots in figures 1 and 2 look identical at first glance. But if we look at the raw data in table 5 and in table 6, we see that there are some very slight differences. These differences are not however large enough to show any real qualitative difference between the new training method and the old training method with softmax + cross entropy.

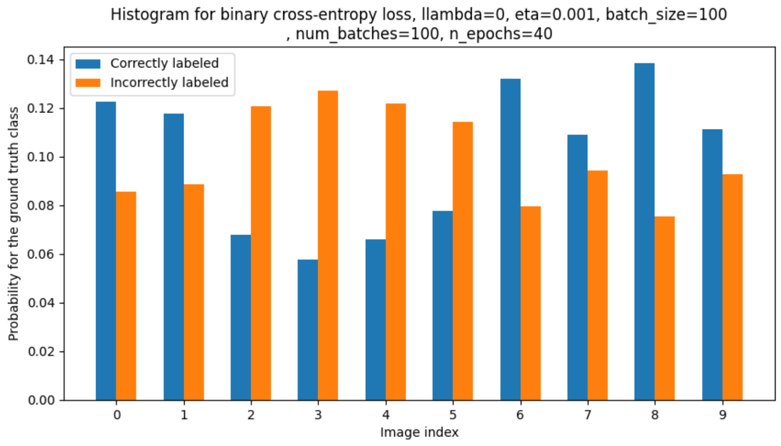

Figure 1: Histogram showing the probability for the ground truth class for both correctly labeled and incorrectly labeled images using cross-entropy loss.

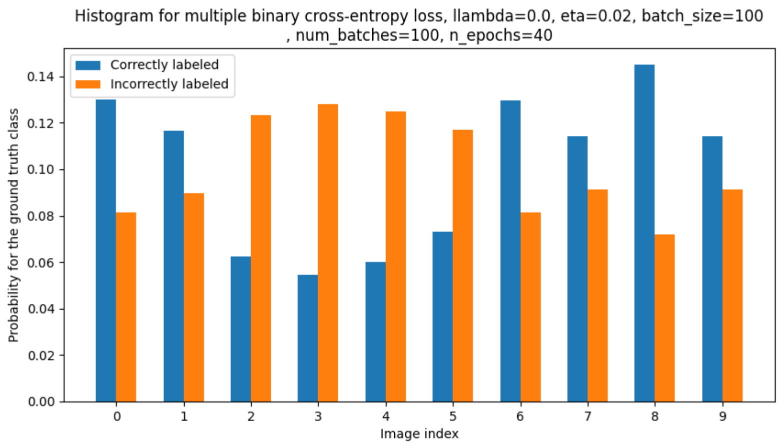

Figure 2: Histogram showing the probability for the ground truth class for both correctly labeled and incorrectly labeled images using multiple binary cross-entropy loss.

image index	0	1	2	3	4	5	6	7	8	9
multiple	0.1300	0.1168	0.0623	0.0547	0.0602	0.0730	0.1298	0.1142	0.1449	0.1142
regular	0.1226	0.1178	0.0677	0.0577	0.0659	0.0777	0.1321	0.1091	0.1382	0.1111

Table 5: Table showing correctly labeled data from both table 1 and 2. Regular here is the cross entropy loss used in the first part of the lab. Multiple is the multiple binary cross-entropy loss.

image index	0	1	2	3	4	5	6	7	8	9
multiple	0.0813	0.0896	0.1235	0.1282	0.1248	0.1168	0.0815	0.0912	0.0720	0.0912
regular	0.0854	0.0886	0.1208	0.1272	0.1219	0.1144	0.0794	0.0942	0.0754	0.0928

Table 6: Table showing incorrectly labeled data from both table 1 and 2. Regular here is the cross entropy loss used in the first part of the lab. Multiple is the multiple binary cross-entropy loss.

2.4 Underfitting vs overfitting

Looking at the graphs in figures 3 and 4, it's easy to see that the training and validation curves are a lot more correlated for the new training method. This would suggest that the model continually learns generalized features not exclusive to the training data. Therefore, looking at only these two plots, we can assume that the new model using multiple binary cross-entropy loss overfits less than the old model.

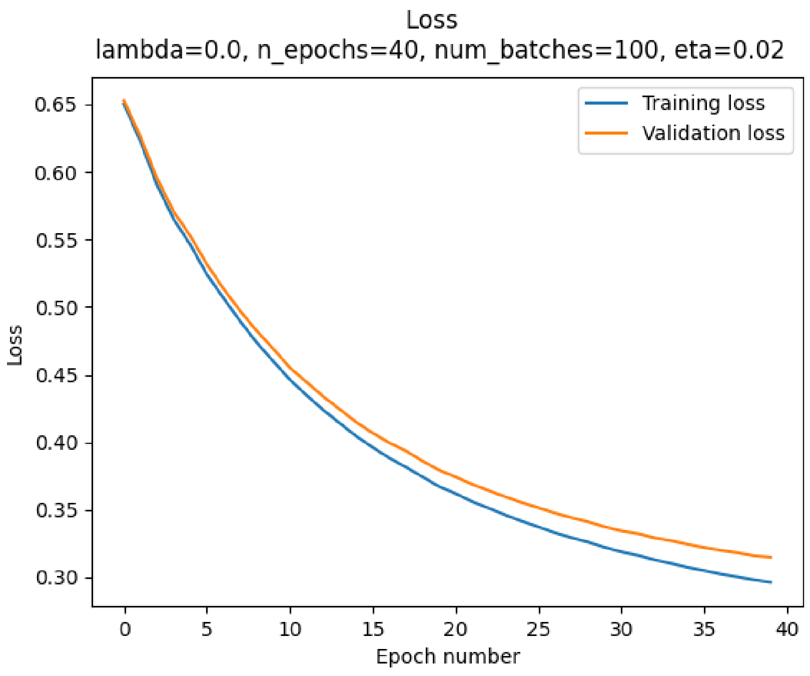

Figure 3: Graph showing training and validation loss using multiple binary cross-entropy loss and the parameters at the top of the graph.

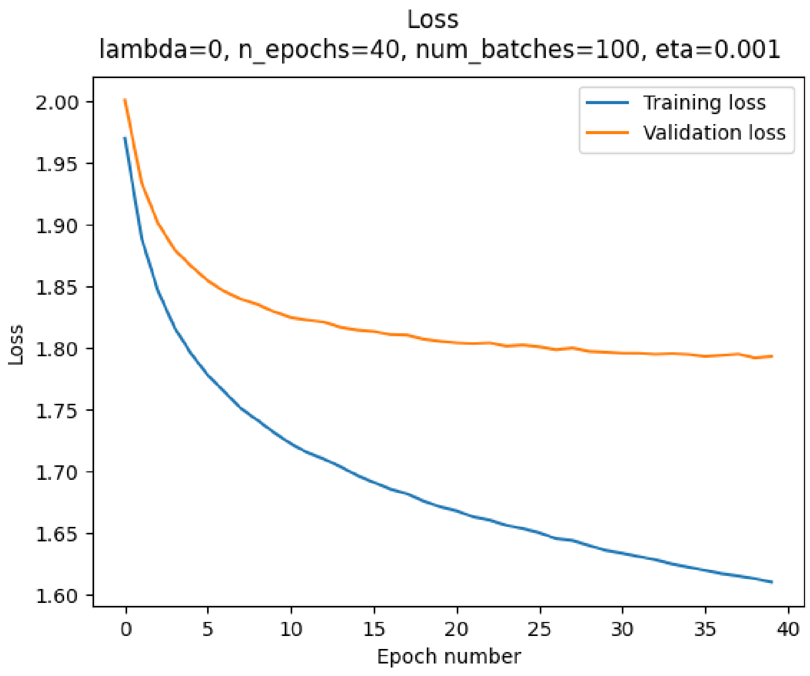

Figure 4: Graph showing training and validation loss using the old training method with softmax and cross-entropy. The parameters of the network can be seen at the top of the graph.

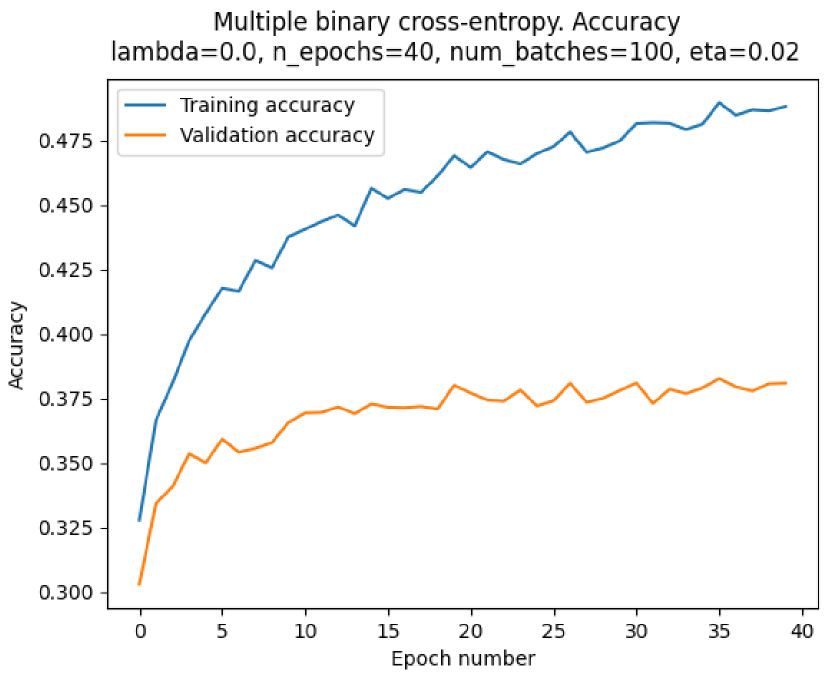

Figure 5: Graph showing the training accuracy and the validation accuracy of the model using multiple binary cross-entropy loss.

However, looking also at the accuracy of each of the models after every epoch, it's not so clear that the new model (using multiple binary cross-entropy loss) overfits less than the old one. In fact, the training accuracy and validation accuracy seem to diverge even more in the new model. This can be seen in figures 5 and 6.

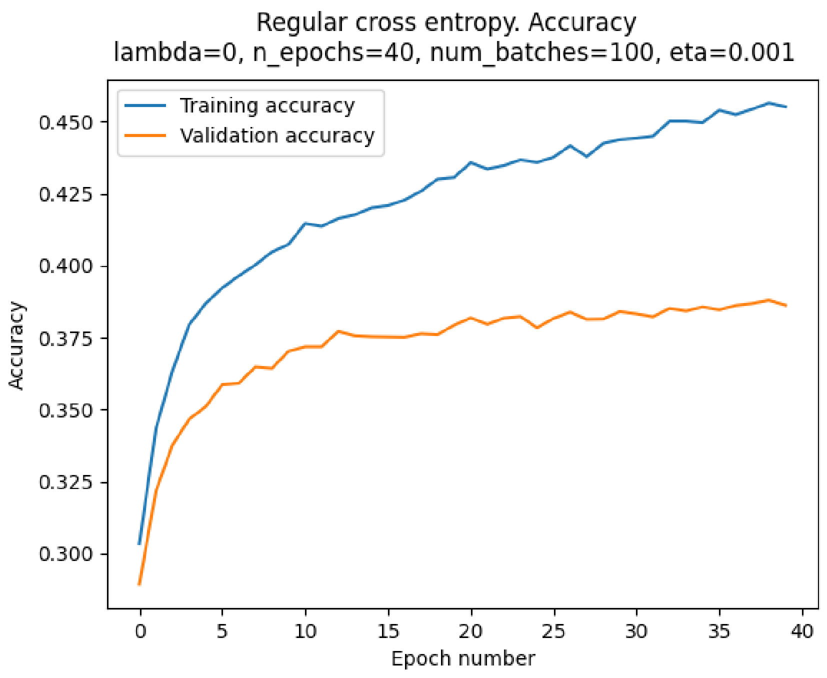

Figure 6: Graph showing the training accuracy and the validation accuracy of the model using the regular cross-entropy loss.