FMAN45 Machine Learning - Assignment 3

Erik Waldemarson

1 Common Layers and Backpropagation

Here I derive some equations that will be needed later.

1.1 Exercise 1

For partial-derivative of loss function L with respect to x_i:

$$\frac{\partial L}{\partial x_i} = \sum_{l=1}^{n} \frac{\partial L}{\partial y_l}\frac{\partial y_l}{\partial x_i} = \sum_{l=1}^{n} \frac{\partial L}{\partial y_l}\frac{\partial}{\partial x_i}(\sum_{k=1}^{m} W_{lk}x_k + b_l) = \sum_{l=1}^{n} W_{li}\frac{\partial L}{\partial y_l}, \quad \forall i \quad =>$$

$$\frac{\partial L}{\partial \mathbf{x}} = \mathbf{W}^T\frac{\partial L}{\partial \mathbf{y}}. \tag{1}$$

With respect to W_{ij}:

$$\frac{\partial L}{\partial W_{ij}} = \sum_{l=1}^{n} \frac{\partial L}{\partial y_l}\frac{\partial y_l}{\partial W_{ij}} = \sum_{l=1}^{n} \frac{\partial L}{\partial y_l}\frac{\partial}{\partial W_{ij}}(\sum_{k=1}^{m} W_{lk}x_k + b_l) = \frac{\partial L}{\partial y_i}x_j, \quad \forall i,j \quad =>$$

$$\frac{\partial L}{\partial \mathbf{W}} = \frac{\partial L}{\partial \mathbf{y}}\mathbf{x}^T, \tag{2}$$

where $(\frac{\partial L}{\partial \mathbf{W}})_{ij} = \frac{\partial L}{\partial W_{ij}}$.

With respect to b_i:

$$\frac{\partial L}{\partial b_i} = \sum_{l=1}^{n} \frac{\partial L}{\partial y_l}\frac{\partial y_l}{\partial b_i} = \sum_{l=1}^{n} \frac{\partial L}{\partial y_l}\frac{\partial}{\partial b_i}(\sum_{k=1}^{m} W_{lk}x_k + b_l) = \frac{\partial L}{\partial y_i}, \quad \forall i \quad =>$$

$$\frac{\partial L}{\partial \mathbf{b}} = \frac{\partial L}{\partial \mathbf{y}}. \tag{3}$$

1.2 Exercise 2

For a batch size N we have:

$$\mathbf{Y} = \mathbf{W}\mathbf{X} + \mathbf{b}\mathbb{1}_N^T, \tag{4}$$

where $\mathbb{1}_N$ is an N-length vector of all ones.

For $\mathbf{X}$:

$$\frac{\partial L}{\partial \mathbf{x}^{(i)}} = \mathbf{W}^T \frac{\partial L}{\partial \mathbf{y}^{(i)}}, \quad \forall i$$

$$\frac{\partial L}{\partial \mathbf{X}} = \mathbf{W}^T \frac{\partial L}{\partial \mathbf{Y}}. \tag{5}$$

For $\mathbf{W}$:

$$\frac{\partial L}{\partial W_{ij}} = \sum_{l=1}^{N}\sum_{k=1}^{n} \frac{\partial L}{\partial y_k^{(l)}} \frac{\partial y_k^{(l)}}{\partial W_{ij}} = \sum_{l=1}^{N} \frac{\partial L}{\partial y_i^{(l)}} x_j^{(l)}, \quad \forall i, j \quad =>$$

$$\frac{\partial L}{\partial \mathbf{W}} = \frac{\partial L}{\partial \mathbf{Y}} \mathbf{X}^T. \tag{6}$$

For $\mathbf{b}$:

$$\frac{\partial L}{\partial b_i} = \sum_{l=1}^{N}\sum_{k=1}^{n} \frac{\partial L}{\partial y_k^{(l)}} \frac{\partial y_k^{(l)}}{\partial b_i} = \sum_{l=1}^{N} \frac{\partial L}{\partial y_i^{(l)}}, \quad \forall i \quad =>$$

$$\frac{\partial L}{\partial \mathbf{b}} = \frac{\partial L}{\partial \mathbf{Y}} \mathbb{1}_N. \tag{7}$$

1.3 Exercise 3

The derivative of the loss when using ReLu as activation function:

$$\frac{\partial L}{\partial x_i} = \frac{\partial L}{\partial y_i}\frac{\partial y_i}{\partial x_i} = \frac{\partial L}{\partial y_i}\frac{\partial}{\partial x_i}\max(x_i, 0), \quad \forall i,$$

derivative of $\max(x, 0)$ is 0 if $x < 0$ and 1 if $x > 0$. The derivative is undefined at the origin but if we define this as 0 I get:

$$\frac{\partial L}{\partial x_i} = \frac{\partial L}{\partial y_i}\theta(x_i), \tag{8}$$

where θ is the Heaviside step function defined as

$$\theta(x_i) = \begin{cases} 1, & x_i > 0 \\ 0, & x_i \leq 0. \end{cases} \tag{9}$$

1.4 Exercise 4

The derivative of the loss function when using log-likelihood with Softmax:

$$\frac{\partial L}{\partial x_i} = \frac{\partial}{\partial x_i}(-x_c + \log(\sum_{k=1}^{m} e^{x_j})) = \frac{e^{x_i}}{\sum_{k=1}^{m} e^{x_j}} - \frac{\partial}{\partial x_i} x_c = y_i - \frac{\partial}{\partial x_i} x_c, \quad \forall i \quad =>$$

Now we have to take into account that the loss function is actually the average of several batches, so we have to take into account a factor $1/N$ since $\frac{\partial}{\partial x_i}(x_1 + ... + x_i + ... + x_N)/N = 1/N$. Which gives:

$$\frac{\partial L}{\partial x_i} = \begin{cases} y_i/N, & i \neq c \\ (y_c - 1)/N, & i = c. \end{cases} \tag{10}$$

2 Classifying Handwritten Digits - MNIST

Here the task is to classify 10 (digits 0-9) classes of MNIST dataset.

2.1 Exercise 6

The model had in total (5 x 5 x 16 + 16) + (5 x 5 x 16 x 16 + 16) + (10 x 784 + 10) = 14 682 parameters.

I got a 97.84% accuracy on the test set using just the baseline. I calculate precision and recall with the formulas:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{11}$$

and

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{12}$$

Where TP = True Positive, FP = False positive, FN = False Negative and TN = True Negative. Now these terms usually only make sense with binary classification, i.e. classes = 0, 1. However we can treat this as a binary problem if we treat the class of interest as "0" and every other class as "1".

I get the following:

Recall = [0.9946, 0.9770, 0.9643, 0.9866, 0.9774, 0.9968, 0.9721, 0.9442, 0.9916, 0.9809].

Precision = [0.9815, 0.9864, 0.9901, 0.9766, 0.9686, 0.9802, 0.9815, 0.9908, 0.9326, 0.9949].

We have both good precision and recall.

Overall this classfication task is pretty easy for modern neural networks as we get very high precision, recall and accuracy. In Table 2 you can see that most of the misclassified images would be difficult to classify even for a human so you can't blame it too much.

The Kernels used in the first convolutional layer have been plotted in Figure 1.

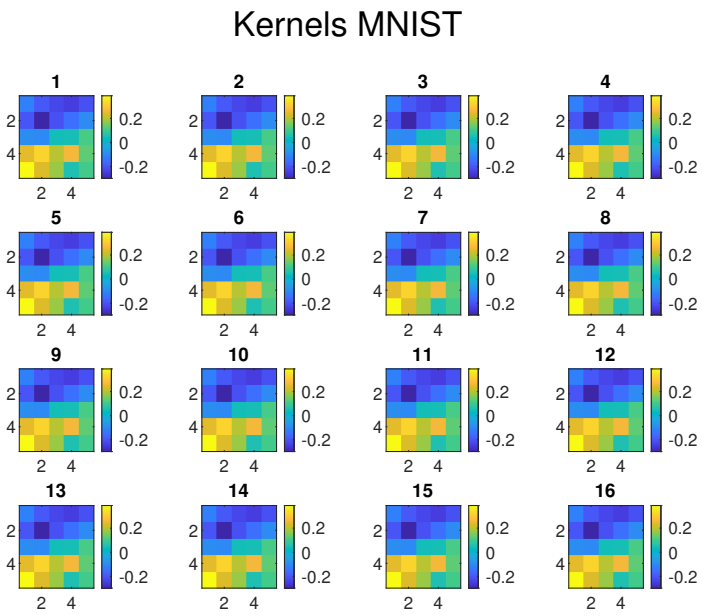

Figure 1: Kernels for first convolutional layer of final model for MNIST image classification task.

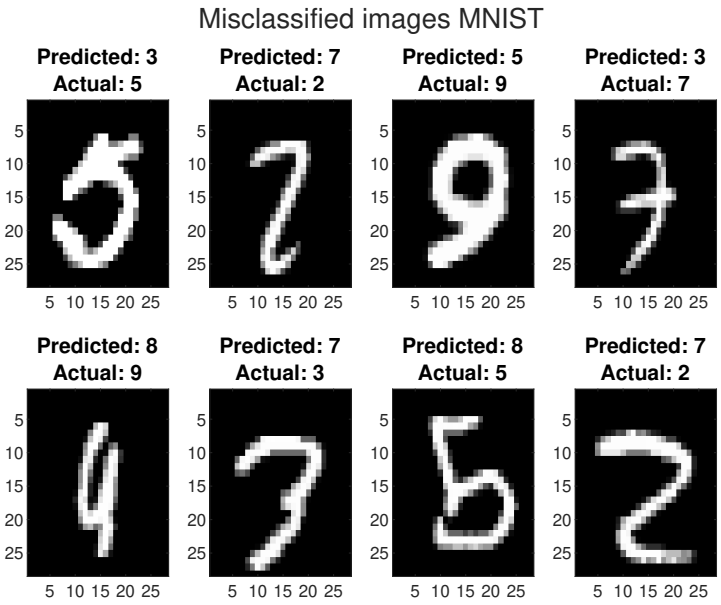

Figure 2: Some examples of misclassified images from the test-data by the final model of the MNIST image classification task.

3 Classifying Images - CIFAR

Here the task is to classify 10 classes of CIFAR dataset.

3.1 Exercise 7

The model had in total (5 x 5 x 3 x 16 + 16) + (5 x 5 x 16 x 16 + 16) + (10 x 4096 + 10) = 48 602 parameters.

I started by running first the baseline and then I ran an additional 3000 iterations changing some of the parameters every 1000th iteration (8000 in total). The different settings have been gathered in Table 2.

As you can see in the table, there was a third model that I actually choose to skip (and by that I mean I reverted back to model #2). I did it because I felt that the training was going to slowly, so I increased the learning rate to make it faster (with the risk of being less accurate) and also increased the momentum to try to push it out of a local minimum (risk of passing over global minimum with higher momentum).

	Predicted									
Actual	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**0**
1	1114	6	2	0	0	0	2	11	0	0
2	0	1018	4	0	0	0	5	5	0	0
3	0	1	1000	0	3	0	4	2	0	0
4	0	2	0	959	0	1	3	9	8	0
5	0	2	19	0	864	2	1	2	2	0
6	3	1	1	3	1	939	0	3	7	0
7	1	10	3	0	1	0	1009	4	0	0
8	0	1	2	1	0	0	1	965	4	0
9	2	0	6	9	14	0	12	19	941	0
0	0	1	0	0	1	0	1	2	0	975

Table 1: Confusion Matrix MNIST.

I increased the batch size because I looked up the CIFAR-10 problem online and found recommendations for higher batch sizes (64-256). Since I had a large dataset this shouldn't be a big problem.

If I'm being honest I can't really motivate a lot of these choices and overall this solution was done very ad hoc. I just mostly guessed new parameters and it happened to give a good accuracy after just a few tries.

In the end I ended up with a test accuracy of 55.22%.

I calculate precision and recall and get:

Recall = [0.5913, 0.6635, 0.4078, 0.3806, 0.4609, 0.4957, 0.6235, 0.5614, 0.6918, 0.6072].

Precision = [0.6220, 0.7000, 0.3890, 0.3890, 0.4190, 0.4060, 0.6890, 0.6260, 0.7070, 0.5750].

As one can see the precision is much worse.

Overall it's clear from the accuracy, precision and recall that this problem is a lot more difficult to solve than the MNIST problem.

In Figure 4 you can see that the misclassified images could for the most part have been predicted by a human.

The Kernels used in the first convolutional layer have been plotted in Figure 1.

Table 2: Models and parameters used for training models for task CIFAR-10 classification.

Model	Iterations	Learning rate	Momentum	Batch size	Weight decay
Baseline	5000	10^{-3}	0.95	16	10^{-3}
#1	1000	10^{-4}	0.95	64	10^{-4}
#2	1000	10^{-4}	0.95	64	10^{-4}
#3 (skipped)	1000	10^{-4}	0.95	64	10^{-4}
#4	1000	10^{-3}	0.99	64	10^{-4}

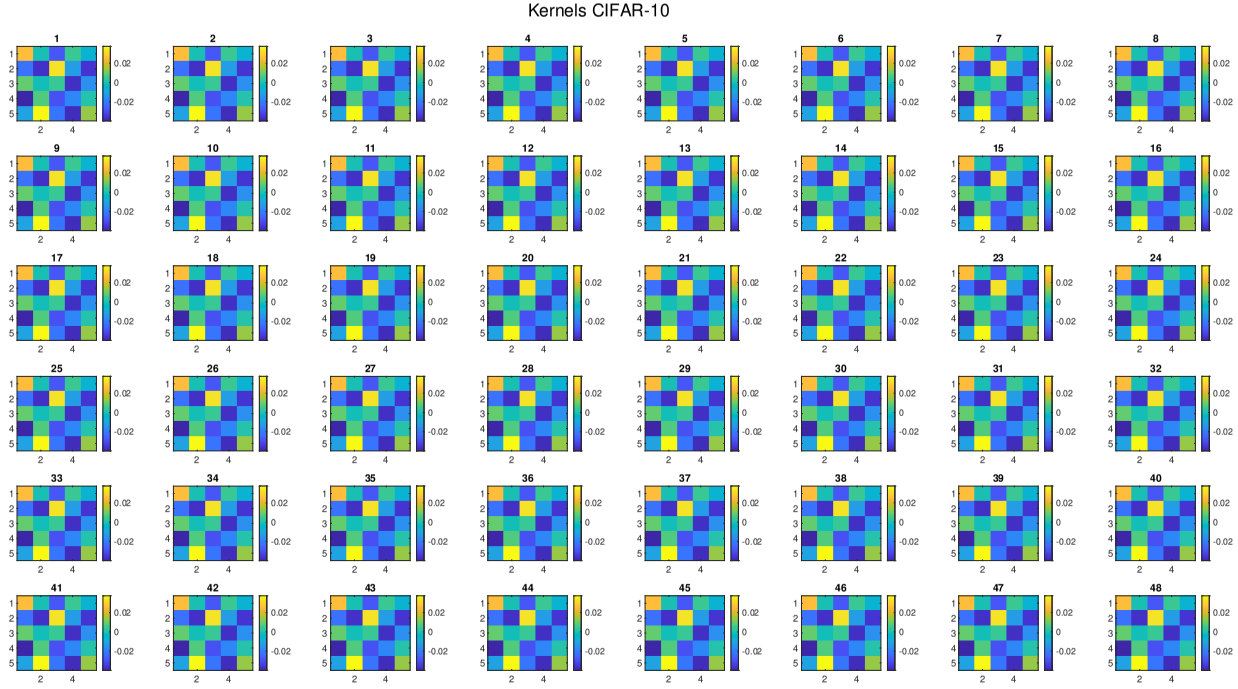

Figure 3: Kernels for first convolutional layer of final model for CIFAR-10 image classification task.

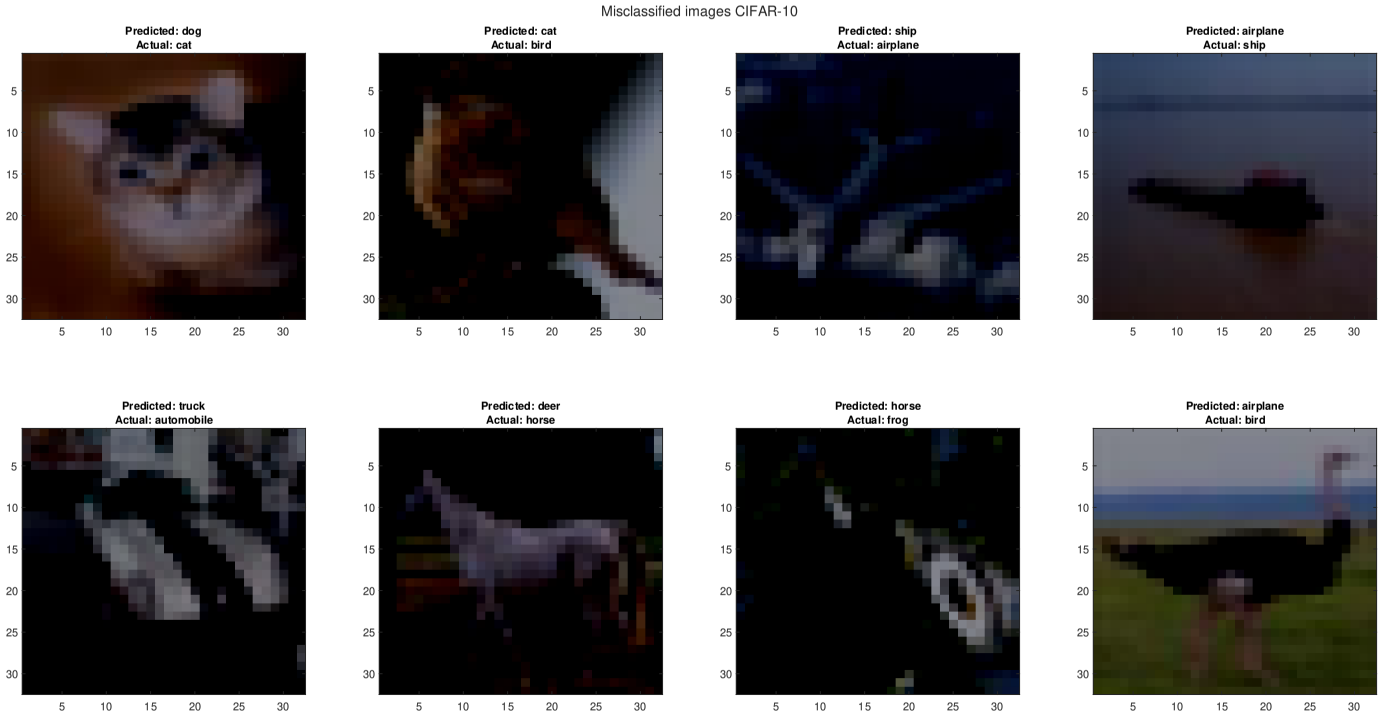

Figure 4: Some examples of misclassified images from the test-data by the final model of the CIFAR-10 image classification task.

	Predicted									
Actual	**Airplane**	**Automobile**	**Bird**	**Cat**	**Deer**	**Dog**	**Frog**	**Horse**	**Ship**	**Truck**
Airplane	622	29	73	20	21	11	13	21	129	61
Automobile	41	700	11	21	9	6	11	16	57	128
Bird	94	11	389	94	140	76	87	65	25	19
Cat	38	21	93	389	67	173	97	73	17	32
Deer	45	15	135	73	419	40	115	132	13	13
Dog	20	10	100	228	67	406	38	98	15	18
Frog	14	9	60	64	89	22	689	31	7	15
Horse	25	5	60	81	75	61	24	626	4	39
Ship	101	69	14	28	8	13	7	6	707	47
Truck	52	186	19	24	14	11	24	47	48	575

Table 3: Confusion Matrix CIFAR-10