Ryan Ott
14862565
ryan.ott@student.uva.nl

1 Linear Module and Activation Module

1.a Linear Module

$$\left[\frac{\partial L}{\partial \mathbf{W}}\right]_{ij} = \frac{\partial L}{\partial W_{ij}} = \sum_{s,n} \frac{\partial L}{\partial Y_{sn}} \frac{\partial Y_{sn}}{\partial W_{ij}} \tag{1}$$

$$\frac{\partial Y_{sn}}{\partial W_{ij}} = \frac{\partial}{\partial W_{ij}} \left(\sum_m [\mathbf{X}]_{sm} [\mathbf{W}^\top]_{mn} + [\mathbf{B}]_{sn} \right) = \sum_m X_{sm} \frac{\partial W_{nm}}{\partial W_{ij}} + \frac{\partial B_{sn}}{\partial W_{ij}} \tag{2}$$

$$= \sum_m X_{sm} \delta_{ni} \delta_{mj} + 0 = X_{sj} \delta_{ni} \tag{3}$$

$$\sum_{s,n} \frac{\partial L}{\partial Y_{sn}} \frac{\partial Y_{sn}}{\partial W_{ij}} = \sum_{s,n} \frac{\partial L}{\partial Y_{sn}} X_{sj} \delta_{ni} = \sum_s \frac{\partial L}{\partial Y_{si}} X_{sj} \tag{4}$$

$$\therefore \frac{\partial L}{\partial \mathbf{W}} = \left(\frac{\partial L}{\partial \mathbf{Y}} \right)^\top \mathbf{X} \quad \in \mathbb{R}^{N \times M} \tag{5}$$

1.b

$$\left[\frac{\partial L}{\partial \mathbf{b}}\right]_{j} = \frac{\partial L}{\partial b_j} = \sum_{s,n} \frac{\partial L}{\partial Y_{sn}} \frac{\partial Y_{sn}}{\partial b_j} \tag{6}$$

$$\frac{\partial Y_{sn}}{\partial b_j} = \frac{\partial}{\partial b_j} \left(\sum_m [\mathbf{X}]_{sm} [\mathbf{W}^\top]_{mn} + [\mathbf{B}]_{sn} \right) = \sum_m \frac{\partial X_{sm} W_{nm}}{\partial b_j} + \frac{\partial B_{sn}}{\partial b_j} \tag{7}$$

$$= 0 + \delta_{nj} = \delta_{nj} \tag{8}$$

$$\sum_{s,n} \frac{\partial L}{\partial Y_{sn}} \frac{\partial Y_{sn}}{\partial b_j} = \sum_{s,n} \frac{\partial L}{\partial Y_{sn}} \delta_{nj} = \sum_s \frac{\partial L}{\partial Y_{sj}} \tag{9}$$

$$\therefore \frac{\partial L}{\partial \mathbf{b}} = \sum_s \frac{\partial L}{\partial \mathbf{Y}_s} \quad \in \mathbb{R}^{1 \times N} \tag{10}$$

For clarification, in equation (10) we sum over the rows $s \in S$ of $\mathbf{Y}$, so the j-th element of $\mathbf{b}$ is the sum of all elements in position j of the rows of $\mathbf{Y}$.

1.c

$$\left[\frac{\partial L}{\partial \mathbf{X}}\right]_{ij} = \frac{\partial L}{\partial X_{ij}} = \sum_{s,n} \frac{\partial L}{\partial Y_{sn}} \frac{\partial Y_{sn}}{\partial X_{ij}} \tag{11}$$

$$\frac{\partial Y_{sn}}{\partial X_{ij}} = \frac{\partial}{\partial X_{ij}} \left(\sum_m [\mathbf{X}]_{sm} [\mathbf{W}^\top]_{mn} + [\mathbf{B}]_{sn} \right) = \sum_m \frac{\partial X_{sm}}{\partial X_{ij}} W_{nm} + \frac{\partial B_{sn}}{\partial X_{ij}} \tag{12}$$

$$= \sum_m \delta_{si} \delta_{mj} W_{nm} + 0 = \delta_{si} W_{nj} \tag{13}$$

$$\sum_{s,n} \frac{\partial L}{\partial Y_{sn}} \frac{\partial Y_{sn}}{\partial X_{ij}} = \sum_{s,n} \frac{\partial L}{\partial Y_{sn}} \delta_{si} W_{nj} = \sum_n \frac{\partial L}{\partial Y_{in}} W_{nj} \tag{14}$$

$$\therefore \frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{Y}} \mathbf{W} \quad \in \mathbb{R}^{S \times M} \tag{15}$$

1.d Activation Module

$$\left[\frac{\partial L}{\partial \mathbf{X}}\right]_{ij} = \frac{\partial L}{\partial X_{ij}} = \sum_{s,m} \frac{\partial L}{\partial Y_{sm}} \frac{\partial Y_{sm}}{\partial X_{ij}} \tag{16}$$

$$= \sum_{s,m} \frac{\partial L}{\partial Y_{sm}} \frac{\partial h(X_{sm})}{\partial X_{ij}} \tag{17}$$

$$= \sum_{s,m} \frac{\partial L}{\partial Y_{sm}} \delta_{si}\delta_{mj} = \frac{\partial L}{\partial Y_{ij}} \tag{18}$$

$$\therefore \frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{Y}} \circ h'(\mathbf{X}) \quad \in \mathbb{R}^{S\times M} \tag{19}$$

From equation (17) to (18) we get two Kronecker deltas because the derivative of the element-wise activation function is zero for all elements except the one we are taking the derivative of. This makes intuitive sense, given we are differentiating an element-wise function. The final derivative of the loss w.r.t. the input $\mathbf{X}$ is the Hadamard product of the derivative of the loss w.r.t. the output $\mathbf{Y}$ and the element-wise derivative of the activation function h w.r.t. the input $\mathbf{X}$. We can assume the shapes of $\mathbf{X}$ and $\mathbf{Y}$ are compatible, since $\mathbf{Y} = h(\mathbf{X})$

2 Softmax, Loss Modules and Residuals

2.a Softmax & Loss Modules

2.b Residuals

Which constraints does the residual connection place on N_1 and N_1, the numbers of neurons in the two linear layers of the LAL module?
The first linear layer L_1 must have have F rows to allow for the matrix multiplication with the input $\mathbf{X} \in \mathbb{R}^{S\times F}$. It will then have an arbitrary output dimension (that is compatible with the activation function). The second linear layer L_2 must however have the exact same shape as X to allow for the element-wise addition of the residual connection. So $N_2 = S \times F$.

2.c

How does adding the residual connection change $\frac{\partial L}{\partial \mathbf{X}}$?
The derivative of the loss now has an additional term, which is the derivative of the residual connection. This is the identity function, so the derivative is 1. The derivative of the loss w.r.t. the input $\mathbf{X}$ is now the sum of the derivative of the loss w.r.t. the output $\mathbf{Y}$ plus the derivative of the residual connection w.r.t. the input $\mathbf{X}$.

$$\frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{Y}}\mathbf{W} + \frac{\partial \mathbf{X}}{\partial \mathbf{X}} \tag{20}$$

$$= \frac{\partial L}{\partial \mathbf{Y}}\mathbf{W} + \mathbf{I} \tag{21}$$

2.d

Briefly explain how your answer to (c) improves the stability of training a deep neural network made up of many such residual blocks, also known as ResNet.

- **Vanishing gradients:** By adding residual connections, the gradient of the loss w.r.t. the input $\mathbf{X}$ can take an alternative path through the network and is far less likely to vanish. Applying the chain rule many times makes us multiply small numbers by each other, making the gradient smaller as it propagates. By adding the input element-wise, we keep an alternative, non-vanishing path in the backward.
- **Reduced Error Propagation:** The network is able to learn more complex functions and isn't limited to passing on the errornous output of the previous layer. It can learn more low-level features from the input or highly process them to learn high-level features.

- **Smoother Loss Landscape:** Because the gradients dont vanish as easily, there are less sharp edges in the loss surface. Also, by allowing both high and low-level features to be passed on, the problem can be modeled better, thus smoothing the loss surface.

3 NumPy Implementation

Test accuracy was 49.88% using the default settings. We see that the training loss is monotonically decreasing, which is what we hope for. The network is learning something with each iteration. When looking at validation accuracy over the epochs, we see that it is increasing, but not monotonically. This is because the network is generally getting better at modelling the data, but sometimes it might be over or under fitting, thus performing worse on unseen data.

4 PyTorch Implementation

Test accuracy was 47.89% using the default settings. Very similar to the NumPy implementation, with a monotonically decreasing training loss and a validation accuracy curve that increases but not monotonically. Noteworthy is that the network seems to make a bigger jump in terms of training loss improvment after the first epoch and that it seems to over/underfit more, as the validation accuracy fluctuates and dips more considerably. This is further made evident by the slightly lower test accuracy.

5 Optimization

5.a Hessian Matrix at Local Minimum having Positive Eigenvalues

Show that the Hessian matrix of a function $f(\mathbf{x})$ at a local minimum has only positive eigenvalues.
From the assignment description we know that if H at point pp is positive definite, then we have are at a strictly local minimum. To prove that the eigenvalues λ of H at a local minimum are positive, we must show that the inequality from Lecture 3 slide 31 holds only for positive eigenvalues:

$$\mathbf{x}^\top H\mathbf{x} > 0 \quad \forall \mathbf{x} \in \mathbb{R}^n \setminus \{0\} \tag{22}$$

Suppose that H has an eigenvalue λ, we can then write the eigenvalue decomposition of H as:

$$H\mathbf{x} = \lambda\mathbf{x} \tag{23}$$

If $\lambda = 0$, then there is some eigenvector $\mathbf{x}$ such that $H\mathbf{x} = 0$. But then the inequality (21) would no longer hold because $\mathbf{x}^\top H\mathbf{x} = 0$. So H would not be positive definite if it had an eigenvalue of 0 and by extension we would not be at a local minimum.
If $\lambda < 0$, then there is some eigenvector $\mathbf{x}$ such that $H\mathbf{x} = \lambda\mathbf{x}$. If we examine this further, we get:

$$\mathbf{x}^\top H\mathbf{x} = \mathbf{x}^\top \lambda\mathbf{x} \tag{24}$$

$$\implies \mathbf{x}^\top H\mathbf{x} = \lambda\mathbf{x}^\top\mathbf{x} \tag{25}$$

$$\implies \mathbf{x}^\top H\mathbf{x} = \lambda|\mathbf{x}|^2 \tag{26}$$

We see that the inequality (21) would no longer hold because $\mathbf{x}^\top H\mathbf{x} = \lambda|\mathbf{x}|^2 < 0$ due to $\lambda < 0$. So H would not be positive definite if it had a negative eigenvalue and by extension we would not be at a local minimum. But then $\mathbf{x}^\top H\mathbf{x} = \lambda\mathbf{x}^\top\mathbf{x}$, which is negative since $\mathbf{x}^\top\mathbf{x} > 0$ and $\lambda < 0$. Thus H is not positive definite.
And so for H at a local minimum, all eigenvalues must be positive.

5.b Exponentially Higher Number of Saddle Points than Local Minima

For a location to be a local minimum, all eigenvalues must be positive as proven in the previous question. For a location to be a saddle point however, there must be at least one negative and one postive eigenvalue. If we take the sign of an eigenvalue to be a binary variable, we can see that there are 2^n possible combinations of eigenvalue signs for n dimensions. Out of all combinations, only one is a local minimum, while almost all remaining combinations are saddle points (except for the one where all eigenvalues are negative, which would be a local maximum). As dimensionality increases, it becomes exponentially more likley that not all eigenvalues will be positive. So the number of saddle points is exponentially higher than the number of local minima.

5.c Gradient Descent around a Saddle Point

Firstly, because as the assignment description states, at a saddle point p the gradient is zero $\nabla_{\mathbf{x}} f(\mathbf{x}_p) = 0$, when we look at the gradient descent update formula we see that the update will be zero as well:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \cdot \nabla_{\mathbf{w}} f(\mathbf{w}_t) \tag{27}$$

$$\implies \mathbf{w}_{t+1} = \mathbf{w}_t - \eta \cdot 0 \tag{28}$$

$$\implies \mathbf{w}_{t+1} = \mathbf{w}_t \tag{29}$$

So in the close vicinity of a saddle point, the gradient descent update will be minimally small, harming the effectiveness of the learning.
Furthermore, when looking at Lecture 3 slide 34, we can see using the 2nd order Taylor expansion that after an update the new loss can be approximated as:

$$\mathcal{L}(w' - \varepsilon g) \approx \mathcal{L}(w') - \varepsilon g^T g + \frac{1}{2}\varepsilon^2 g^T H g \tag{30}$$

Where $g^T g$ (the first order term) represents the magnitude of the gradient, and $g^T H g$ (second order term) represents the curvature in the direction of the gradient. If $\frac{1}{2}\varepsilon^2 g^T H g > \varepsilon g^T g$, the update step can increase the loss because the effect of the curvature (the second-order term) is greater than the effect of the gradient (the first-order term).

6 Precision, Recall, Accuracy and F1-beta Score

6.a When to use which metric

Precision: Used when the cost of false positives is high. For example, in email spam detection (labeling a legitimate email as spam is worse than missing a spam email) and in legal systems (labeling an innocent person as guilty is highly undesirable).
Recall: Used when missing true positives is costly. For example, in fraud detection systems (missing a fraudulent transaction can be very costly, rather one would do an unnecessary double check) and in medical diagnosis (missing a disease can be fatal).
Accuracy: Used when the costs of false positives and false negatives are relatively balanced and the class distribution is even. For example, in a balanced binary classification problem like sentiment analysis (positive vs negative reviews) and in gender classification based on images.

6.b Confusion Matrix

Created using NumPy, we get the confusion matrices on the CIFAR10 dataset for the two models as seen in the `.png` files.

6.c F1-beta Score

Using NumPy, a metrics table for both the NumPy and the PyTorch model was created (also shown in the `.png`) files.
For the NumPy model we see that the *cars, trucks, ships* and *airplanes* classes have the highest precision, with the classes for which the model showed high precision also being the ones with highest recall. Especially ships seem to be easily classifyable as it has the highest F1-1 score (which is the harmonic mean of precision and recall), it has one of the highest precision scores and the highest recall score, so scores even more highly in the F1-10 metric, which weighs recall higher than precision.
For the PyTorch model we see that *trucks* now actually beats *cars* in terms of precision, but *ships* having the highest recall and F1-1 score. Noticable for this model is the very low recall for *cats*, which is the lowest of all classes. This is also reflected in the F1-10 score, which is the lowest by far for both models. This might explain the lower overall accuracy of the PyTorch model compared to the NumPy one.