Deep Learning Practical 1

Eui Yeon Jang
12661635

1 MLP backprop and NumPy implementation

Question 1.1 a) Linear Module

Gradient with respect to $\boldsymbol{W}$:

$$
\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{W}} \implies \left[\frac{\partial L}{\partial \boldsymbol{W}}\right]_{ij} = \frac{\partial L}{\partial W_{ij}} &= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \frac{\partial Y_{mn}}{\partial W_{ij}} \\
&= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \frac{\partial}{\partial W_{ij}} \left(\left(\sum_k X_{mk} W_{nk} \right) + B_{mn} \right) \\
&= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \left(\left(\sum_k \frac{\partial}{\partial W_{ij}} X_{mk} W_{nk} \right) + \frac{\partial}{\partial W_{ij}} B_{mn} \right) \\
&= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \sum_k X_{mk} \frac{\partial}{\partial W_{ij}} W_{nk} \\
&= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \sum_k X_{mk} \delta_{in} \delta_{jk} \\
&= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} X_{mj} \delta_{in} \\
&= \sum_{m} \frac{\partial L}{\partial Y_{mi}} X_{mj}
\end{aligned}
$$

In matrix form:

$$
\frac{\partial L}{\partial \boldsymbol{W}} = \left(\frac{\partial L}{\partial \boldsymbol{Y}} \right)^T \boldsymbol{X}
$$

Gradient with respect to $\boldsymbol{b}$:

$$\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{b}} \implies \left[\frac{\partial L}{\partial \boldsymbol{b}}\right]_j = \frac{\partial L}{\partial b_j} &= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \frac{\partial Y_{mn}}{\partial b_j} \\
&= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \frac{\partial}{\partial b_j} \left(\left(\sum_k X_{mk} W_{nk} \right) + B_{mn} \right) \quad \text{where } B_{mn} = b_n \\
&= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \left(\left(\frac{\partial}{\partial b_j} \sum_k X_{mk} W_{nk} \right) + \frac{\partial b_n}{\partial b_j} \right) \\
&= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \frac{\partial b_n}{\partial b_j} \\
&= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \delta_{jn} \\
&= \sum_{m} \frac{\partial L}{\partial Y_{mj}} \cdot 1
\end{aligned}$$

In matrix form:

$$\frac{\partial L}{\partial \boldsymbol{b}} = \boldsymbol{1}^T \frac{\partial L}{\partial \boldsymbol{Y}},$$

where $\boldsymbol{1} \in \mathbb{R}^S$.

Gradient with respect to $\boldsymbol{X}$:

$$\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{X}} \implies \left[\frac{\partial L}{\partial \boldsymbol{X}}\right]_{ij} = \frac{\partial L}{\partial X_{ij}} &= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \frac{\partial Y_{mn}}{\partial X_{ij}} \\
&= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \frac{\partial}{\partial X_{ij}} \left(\left(\sum_k X_{mk} W_{nk} \right) + B_{mn} \right) \\
&= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \left(\left(\sum_k \frac{\partial}{\partial X_{ij}} X_{mk} W_{nk} \right) + \frac{\partial}{\partial X_{ij}} b_n \right) \\
&= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \sum_k W_{nk} \frac{\partial}{\partial X_{ij}} X_{mk} \\
&= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \sum_k W_{nk} \delta_{im} \delta_{jk} \\
&= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} W_{nj} \delta_{im} \\
&= \sum_{n} \frac{\partial L}{\partial Y_{in}} W_{nj}
\end{aligned}$$

In matrix form:

$$\frac{\partial L}{\partial \boldsymbol{X}} = \frac{\partial L}{\partial \boldsymbol{Y}} \boldsymbol{W}$$

Question 1.1 b) Activation Module

$$
\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{X}} \implies \left[\frac{\partial L}{\partial \boldsymbol{X}}\right]_{ij} = \frac{\partial L}{\partial X_{ij}} &= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \frac{\partial Y_{mn}}{\partial X_{ij}} \\
&= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \frac{\partial}{\partial X_{ij}} h(X_{mn}) \\
&= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} h'(X_{mn}) \frac{\partial}{\partial X_{ij}} X_{mn} \\
&= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} h'(X_{mn}) \delta_{im}\delta_{jn} \\
&= \frac{\partial L}{\partial Y_{ij}} h'(X_{ij}) \\
&= \left[\frac{\partial L}{\partial \boldsymbol{Y}} \circ h'(\boldsymbol{X})\right]_{ij}
\end{aligned}
$$

In matrix form:

$$\frac{\partial L}{\partial \boldsymbol{X}} = \frac{\partial L}{\partial \boldsymbol{Y}} \circ h'(\boldsymbol{X}),$$

where $\circ$ is the Hadamard product (element-wise product).

Question 1.1 c) Softmax and Loss Modules

i)

$$
\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{X}} \implies \left[\frac{\partial L}{\partial \boldsymbol{X}}\right]_{ij} = \frac{\partial L}{\partial X_{ij}} &= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \frac{\partial Y_{mn}}{\partial X_{ij}} \\
&= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \frac{\partial}{\partial X_{ij}} [\text{softmax}(\boldsymbol{X})]_{mn}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial}{\partial X_{ij}} [\text{softmax}(\boldsymbol{X})]_{mn} &= \frac{\partial}{\partial X_{ij}} \frac{\exp(X_{mn})}{\sum_k \exp(X_{ik})} \\
&= \frac{\sum_k \exp(X_{ik}) \frac{\partial}{\partial X_{ij}} \exp(X_{mn}) - \exp(X_{mn}) \frac{\partial}{\partial X_{ij}} \sum_k \exp(X_{ik})}{\left(\sum_k \exp(X_{ik})\right)^2}
\end{aligned}
$$

If $i = m$ and $j = n$:

$$
\begin{aligned}
\frac{\partial}{\partial X_{ij}} [\text{softmax}(\boldsymbol{X})]_{mn} &= \frac{\sum_k \exp(X_{ik}) \frac{\partial}{\partial X_{ij}} \exp(X_{mn}) - \exp(X_{mn}) \frac{\partial}{\partial X_{ij}} \sum_k \exp(X_{ik})}{\left(\sum_k \exp(X_{ik})\right)^2} \\
&= \frac{\sum_k \exp(X_{ik}) \exp(X_{ij}) - \exp(X_{ij}) \exp(X_{ij})}{\left(\sum_k \exp(X_{ik})\right)^2} \\
&= \frac{\exp(X_{ij}) \left(\sum_k \exp(X_{ik}) - \exp(X_{ij})\right)}{\left(\sum_k \exp(X_{ik})\right)\left(\sum_k \exp(X_{ik})\right)} \\
&= \frac{\exp(X_{ij})}{\sum_k \exp(X_{ik})} \frac{\sum_k \exp(X_{ik}) - \exp(X_{ij})}{\sum_k \exp(X_{ik})} \\
&= \frac{\exp(X_{ij})}{\sum_k \exp(X_{ik})} \left(1 - \frac{\exp(X_{ij})}{\sum_k \exp(X_{ik})}\right) \\
&= [\text{softmax}(\boldsymbol{X})]_{ij} \left(1 - [\text{softmax}(\boldsymbol{X})]_{ij}\right)
\end{aligned}
$$

If $i \neq m$ and/or $j \neq n$:

$$\begin{aligned}
\frac{\partial}{\partial X_{ij}}[\text{softmax}(\boldsymbol{X})]_{mn} &= \frac{\sum_k \exp(X_{ik}) \frac{\partial}{\partial X_{ij}} \exp(X_{mn}) - \exp(X_{mn}) \frac{\partial}{\partial X_{ij}} \sum_k \exp(X_{ik})}{\left(\sum_k \exp(X_{ik})\right)^2} \\
&= \frac{\sum_k \exp(X_{ik}) \exp(X_{mn}) \delta_{im}\delta_{jn} - \exp(X_{mn}) \exp(X_{ij})}{\left(\sum_k \exp(X_{ik})\right)^2} \\
&= -\frac{\exp(X_{mn}) \exp(X_{ij})}{\left(\sum_k \exp(X_{ik})\right)\left(\sum_k \exp(X_{ik})\right)} \\
&= -\frac{\exp(X_{mn})}{\sum_k \exp(X_{ik})} \frac{\exp(X_{ij})}{\sum_k \exp(X_{ik})} \\
&= -[\text{softmax}(\boldsymbol{X})]_{mn}[\text{softmax}(\boldsymbol{X})]_{ij}
\end{aligned}$$

ii)

Note: Natural logarithm is assumed.

$$\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{X}} \implies \left[\frac{\partial L}{\partial \boldsymbol{X}}\right]_{ij} = \frac{\partial L}{\partial X_{ij}} = \frac{\partial}{\partial X_{ij}} \left(-\frac{1}{S}\sum_{mk} T_{mk} \log(X_{mk})\right) \\
&= -\frac{1}{S}\sum_{mk} \frac{\partial}{\partial X_{ij}} T_{mk} \log(X_{mk}) \\
&= -\frac{1}{S}\sum_{mk} T_{mk} \frac{\partial}{\partial X_{ij}} \log(X_{mk}) \\
&= -\frac{1}{S}\sum_{mk} \frac{T_{mk}}{X_{mk}} \frac{\partial}{\partial X_{ij}} X_{mk} \\
&= -\frac{1}{S}\sum_{mk} \frac{T_{mk}}{X_{mk}} \delta_{im}\delta_{jk} \\
&= -\frac{1}{S}\frac{T_{ij}}{X_{ij}}
\end{aligned}$$

In matrix form:

$$\frac{\partial L}{\partial \boldsymbol{X}} = -\frac{1}{S}\boldsymbol{T} \oslash \boldsymbol{X},$$

where $\oslash$ is the Hadamard division (element-wise division).

Question 1.1 d) Bonus

1)

$$\begin{aligned}
Y_{21} &= \sum_{l=-1}^{0} \sum_{m=-1}^{0} K_{(l+2)(m+2)} X_{(2-l)(1-m)} \\
&= K_{11}X_{32} + K_{12}X_{31} + K_{21}X_{22} + K_{22}X_{21}
\end{aligned}$$

2)

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial K_{11}} &= \sum_{i,j} \frac{\partial \mathcal{L}}{\partial Y_{ij}} \frac{\partial Y_{ij}}{\partial K_{11}} \\
&= \sum_{i,j} \frac{\partial \mathcal{L}}{\partial Y_{ij}} \frac{\partial}{\partial K_{11}} \sum_{l=-1}^{0} \sum_{m=-1}^{0} K_{(l+2)(m+2)} X_{(i-l)(j-m)} \\
&= \sum_{i,j} \frac{\partial \mathcal{L}}{\partial Y_{ij}} \sum_{l=-1}^{0} \sum_{m=-1}^{0} X_{(i-l)(j-m)} \frac{\partial}{\partial K_{11}} K_{(l+2)(m+2)} \\
&= \sum_{i,j} \frac{\partial \mathcal{L}}{\partial Y_{ij}} \sum_{l=-1}^{0} \sum_{m=-1}^{0} X_{(i-l)(j-m)} \delta_{(l+2)1} \delta_{(m+2)1} \\
&\text{where } \delta = 1 \text{ if } l = m = -1 \\
&= \sum_{i,j} \frac{\partial \mathcal{L}}{\partial Y_{ij}} X_{(i+1)(j+1)} \quad \text{given } 1 \le i+1 \le 3, 1 \le j+1 \le 3 \\
&= \sum_{i=1}^{2} \sum_{j=1}^{2} \frac{\partial \mathcal{L}}{\partial Y_{ij}} X_{(i+1)(j+1)} \\
&= \frac{\partial \mathcal{L}}{\partial Y_{11}} X_{22} + \frac{\partial \mathcal{L}}{\partial Y_{12}} X_{23} + \frac{\partial \mathcal{L}}{\partial Y_{21}} X_{32} + \frac{\partial \mathcal{L}}{\partial Y_{22}} X_{33}
\end{aligned}
$$

3)

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial X_{12}} &= \sum_{i,j} \frac{\partial \mathcal{L}}{\partial Y_{ij}} \frac{\partial Y_{ij}}{\partial X_{12}} \\
&= \sum_{i,j} \frac{\partial \mathcal{L}}{\partial Y_{ij}} \frac{\partial}{\partial X_{12}} \sum_{l=-1}^{0} \sum_{m=-1}^{0} K_{(l+2)(m+2)} X_{(i-l)(j-m)} \\
&= \sum_{i,j} \frac{\partial \mathcal{L}}{\partial Y_{ij}} \sum_{l=-1}^{0} \sum_{m=-1}^{0} K_{(l+2)(m+2)} \frac{\partial}{\partial X_{12}} X_{(i-l)(j-m)} \\
&= \sum_{i,j} \frac{\partial \mathcal{L}}{\partial Y_{ij}} \sum_{l=-1}^{0} \sum_{m=-1}^{0} K_{(l+2)(m+2)} \delta_{(i-l)1} \delta_{(j-m)2} \\
&\text{where } \delta = 1 \text{ if } l = i-1, \quad m = j-2 \\
&= \sum_{i,j} \frac{\partial \mathcal{L}}{\partial Y_{ij}} K_{(i+1)j} \quad \text{given } 1 \le i+1 \le 2 \\
&= \sum_{j} \frac{\partial \mathcal{L}}{\partial Y_{1j}} K_{2j} = \frac{\partial \mathcal{L}}{\partial Y_{11}} K_{21} + \frac{\partial \mathcal{L}}{\partial Y_{12}} K_{22}
\end{aligned}
$$

Question 1.2

The loss and accuracy curves of both the training and test data set, with default parameter values, are shown in Figure 1. The best training accuracy goes up to 0.55, and the best test accuracy to 0.49.

2 PyTorch MLP

Question 2.1

The initial implementation of PyTorch MLP with default parameter values achieved maximum accuracy of 0.49 on the test set as with the NumPy MLP. To improve the performance, the following modifications were made.

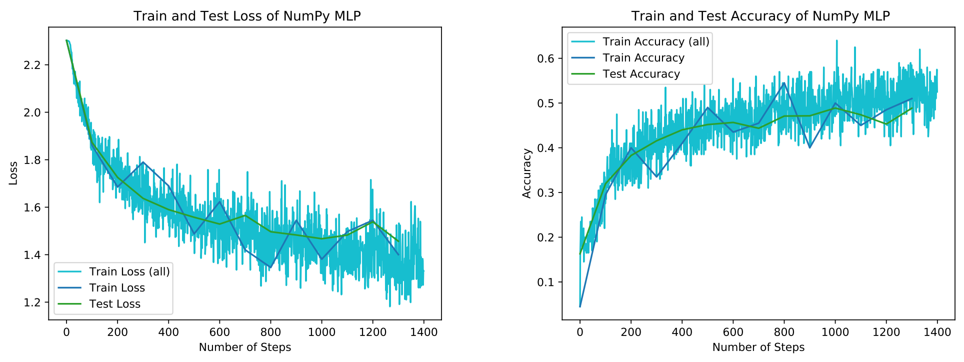

Figure 1: Cross entropy loss and accuracy values of NumPy MLP classifier with default parameter values.

The number of training steps were modified first – increased to 2400 steps and then 3000. 2400 steps achieved maximum of 0.51 accuracy on the test set, and increasing to 3000 steps made no difference to the maximum. Then the complexity of the network was increased – two hidden layers with 500 and 100 neurons, respectively. The training steps were kept at 3000 to allow to train the now more complex network, however, the accuracy did not increase beyond 0.20. The network was then optimised using SGD with momentum 0.9, which brought the maximum accuracy to 0.54 (this model is hereafter referred to as the SGD version). Using Adam did not do better than 0.50.

A even more complex network with 3 hidden layers with 1000, 500, 100 neurons were implemented with Adam optimiser (with its default parameter values and our pre-specified learning rate), however, it did not improve upon the maximum of 0.54 (this model is referred to as the Adam version).

Although the SGD and Adam versions achieve the same maximum accuracy, Adam version reaches the 0.50s mark and 0.53 mark sooner (around step 800 and step 2000, respectively) compared to the SGD version (around step 1600 and 2400). However, the SGD version is preferred due to the fact that it's a simpler model.

Figure 2 shows the loss and accuracy curves of PyTorch MLP with tuned parameter values: 3000 training steps, two hidden layers with 500 and 100 neurons, respectively, momentum of 0.9 for the SGD optimiser and the rest as default. This model achieved best train accuracy of 0.63 and test accuracy of 0.54. In the figure, the trend of both train and test loss decreasing and accuracy increasing can be observed. However, the gaps between the train and test loss and accuracy towards the later steps may be an indication of overfitting.

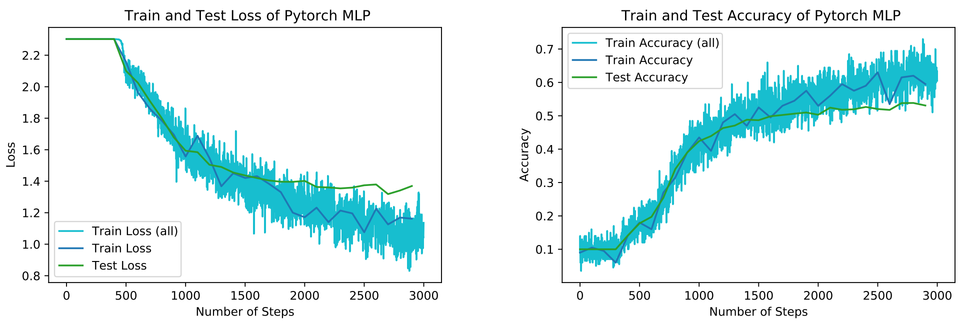

Figure 2: Cross entropy loss and accuracy values of Pytorch MLP classifier with tuned parameter values.

Question 2.2

The advantage of Tanh over ELU would that Tanh keeps the activations in the range of [-1, 1], whereas ELU can blow up the activations for $x > 0$. The drawbacks would be that Tanh produces non-sparse model and suffers from the vanishing gradient problem, compared to ELU which has stronger gradients.

When training the above classifier with similar parameter values using Tanh, the maximum accuracy achieved was 0.44, which was only acquired over a much longer training time.

3 Custom Module: Layer Normalisation

Question 3.2 a)

Gradient with regard to $\boldsymbol{\gamma}$:

$$\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{\gamma}} \implies \left[\frac{\partial L}{\partial \boldsymbol{\gamma}}\right]_i = \frac{\partial L}{\partial \gamma_i} &= \sum_{s,j} \frac{\partial L}{\partial Y_{sj}} \frac{\partial Y_{sj}}{\partial \gamma_i} \\
&= \sum_{s,j} \frac{\partial L}{\partial Y_{sj}} \frac{\partial}{\partial \gamma_i} \left(\gamma_j \hat{X}_{sj} + \beta_j\right) \\
&= \sum_{s,j} \frac{\partial L}{\partial Y_{sj}} \frac{\partial}{\partial \gamma_i} \gamma_j \hat{X}_{sj} + \frac{\partial}{\partial \gamma_i} \beta_j \\
&= \sum_{s,j} \frac{\partial L}{\partial Y_{sj}} \hat{X}_{sj} \frac{\partial}{\partial \gamma_i} \gamma_j \\
&= \sum_{s,j} \frac{\partial L}{\partial Y_{sj}} \hat{X}_{sj} \delta_{ij} \\
&= \sum_{s} \frac{\partial L}{\partial Y_{si}} \hat{X}_{si}
\end{aligned}$$

In matrix form:

$$\frac{\partial L}{\partial \boldsymbol{\gamma}} = \left(\frac{\partial L}{\partial \boldsymbol{Y}} \circ \hat{\boldsymbol{X}}\right)^T \mathbf{1}$$

Gradient with regard to $\boldsymbol{\beta}$:

$$\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{\beta}} \implies \left[\frac{\partial L}{\partial \boldsymbol{\beta}}\right]_i = \frac{\partial L}{\partial \beta_i} &= \sum_{s,j} \frac{\partial L}{\partial Y_{sj}} \frac{\partial Y_{sj}}{\partial \beta_i} \\
&= \sum_{s,j} \frac{\partial L}{\partial Y_{sj}} \frac{\partial}{\partial \beta_i} \left(\gamma_j \hat{X}_{sj} + \beta_j\right) \\
&= \sum_{s,j} \frac{\partial L}{\partial Y_{sj}} \frac{\partial}{\partial \beta_i} \gamma_j \hat{X}_{sj} + \frac{\partial}{\partial \beta_i} \beta_j \\
&= \sum_{s,j} \frac{\partial L}{\partial Y_{sj}} \delta_{ij} \\
&= \sum_{s} \frac{\partial L}{\partial Y_{si}} \cdot 1
\end{aligned}$$

In matrix form:

$$\frac{\partial L}{\partial \boldsymbol{\beta}} = \left(\frac{\partial L}{\partial \boldsymbol{Y}}\right)^T \mathbf{1}$$

Gradient with regard to $\boldsymbol{X}$:

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{X}} \implies \left[\frac{\partial L}{\partial \boldsymbol{X}}\right]_{ri} = \frac{\partial L}{\partial X_{ri}} &= \sum_{s,j} \frac{\partial L}{\partial Y_{sj}} \frac{\partial Y_{sj}}{\partial X_{ri}} \\ &= \sum_{s,j} \frac{\partial L}{\partial Y_{sj}} \frac{\partial}{\partial X_{ri}} \left(\gamma_j \hat{X}_{sj} + \beta_j\right) \\ &= \sum_{s,j} \frac{\partial L}{\partial Y_{sj}} \gamma_j \frac{\partial \hat{X}_{sj}}{\partial X_{ri}}\end{aligned}$$

Using equation (1):

$$\begin{aligned}\frac{\partial L}{\partial X_{ri}} &= \sum_{s,j} \frac{\partial L}{\partial Y_{sj}} \gamma_j \frac{\partial \hat{X}_{sj}}{\partial X_{ri}} \\ &= \sum_{s,j} \frac{\partial L}{\partial Y_{sj}} \gamma_j \frac{\delta_{rs}}{\sqrt{\sigma_s^2 + \varepsilon}} \left[\delta_{ij} - \frac{1}{M}\left(1 + \hat{X}_{sj}\hat{X}_{si}\right)\right] \\ &= \frac{1}{\sqrt{\sigma_r^2 + \varepsilon}} \left[\sum_j \frac{\partial L}{\partial Y_{rj}} \gamma_j \delta_{ij} - \sum_j \frac{\partial L}{\partial Y_{rj}} \gamma_j \frac{1}{M}\left(1 + \hat{X}_{rj}\hat{X}_{ri}\right)\right] \\ &= \frac{1}{\sqrt{\sigma_r^2 + \varepsilon}} \left[\frac{\partial L}{\partial Y_{ri}} \gamma_i - \frac{1}{M}\sum_j \frac{\partial L}{\partial Y_{rj}} \gamma_j \left(1 + \hat{X}_{rj}\hat{X}_{ri}\right)\right] \\ &= \frac{1}{\sqrt{\sigma_r^2 + \varepsilon}} \left[\frac{\partial L}{\partial Y_{ri}} \gamma_i - \frac{1}{M}\sum_j \frac{\partial L}{\partial Y_{rj}} \gamma_j - \frac{\hat{X}_{ri}}{M}\sum_j \frac{\partial L}{\partial Y_{rj}} \gamma_j \hat{X}_{rj}\right]\end{aligned}$$

Equation (1) is obtained by using equations (2), (3), (4), (5), (6):

$$\begin{aligned}\frac{\partial \hat{X}_{sj}}{\partial X_{ri}} &= \frac{\partial \hat{X}_{sj}}{\partial X_{ri}} + \frac{\partial \hat{X}_{sj}}{\partial \sigma_s^2}\frac{\partial \sigma_s^2}{\partial X_{ri}} + \frac{\partial \hat{X}_{sj}}{\partial \mu_s}\frac{\partial \mu_s}{\partial X_{ri}} \\ &= \frac{\delta_{rs}\delta_{ij}}{\sqrt{\sigma_s^2 + \varepsilon}} - \frac{1}{2}(X_{sj} - \mu_s)(\sigma_s^2 + \epsilon)^{-1.5}\frac{2\delta_{rs}}{M}(X_{si} - \mu_s) - \frac{1}{\sqrt{\sigma_s^2 + \varepsilon}}\frac{\delta_{rs}}{M} \\ &= \frac{1}{\sqrt{\sigma_s^2 + \varepsilon}}\left[\delta_{rs}\delta_{ij} - \frac{1}{2}(X_{sj} - \mu_s)\frac{1}{\sigma_s^2 + \epsilon}\frac{2\delta_{rs}}{M}(X_{si} - \mu_s) - \frac{\delta_{rs}}{M}\right] \\ &= \frac{1}{\sqrt{\sigma_s^2 + \varepsilon}}\left[\delta_{rs}\delta_{ij} - \frac{(X_{sj} - \mu_s)(X_{si} - \mu_s)}{\sqrt{\sigma_s^2 + \epsilon}\sqrt{\sigma_s^2 + \epsilon}}\frac{\delta_{rs}}{M} - \frac{\delta_{rs}}{M}\right] \\ &= \frac{\delta_{rs}}{\sqrt{\sigma_s^2 + \varepsilon}}\left[\delta_{ij} - \hat{X}_{sj}\hat{X}_{si}\frac{1}{M} - \frac{1}{M}\right] \\ &= \frac{\delta_{rs}}{\sqrt{\sigma_s^2 + \varepsilon}}\left[\delta_{ij} - \frac{1}{M}\left(1 + \hat{X}_{sj}\hat{X}_{si}\right)\right] \qquad (1)\end{aligned}$$

$$\begin{aligned}\frac{\partial \hat{X}_{sj}}{\partial X_{ri}} &= \frac{1}{\sqrt{\sigma_s^2 + \varepsilon}}\frac{\partial X_{sj}}{\partial X_{ri}} \\ &= \frac{\delta_{rs}\delta_{ij}}{\sqrt{\sigma_s^2 + \varepsilon}} \qquad (2)\end{aligned}$$

$$\frac{\partial \hat{X}_{sj}}{\partial \mu_s} = -\frac{1}{\sqrt{\sigma_s^2 + \varepsilon}} \qquad (3)$$

$$\frac{\partial \mu_s}{\partial X_{ri}} = \frac{\delta_{rs}}{M} \tag{4}$$

$$\begin{aligned}
\frac{\partial \hat{X}_{sj}}{\partial \sigma_s^2} &= \frac{\partial}{\partial \sigma_s^2}(X_{sj} - \mu_s)(\sigma_s^2 + \epsilon)^{-0.5} \\
&= (X_{sj} - \mu_s)\frac{\partial}{\partial \sigma_s^2}(\sigma_s^2 + \epsilon)^{-0.5} \\
&= -\frac{1}{2}(X_{sj} - \mu_s)(\sigma_s^2 + \epsilon)^{-1.5}
\end{aligned} \tag{5}$$

$$\begin{aligned}
\frac{\partial \sigma_s^2}{\partial X_{ri}} &= \frac{1}{M}\sum_{k=1}^{M} \frac{\partial}{\partial X_{ri}}(X_{sk} - \mu_s)^2 \\
&= \frac{1}{M}\sum_{k=1}^{M} 2(X_{sk} - \mu_s)\frac{\partial (X_{sk} - \mu_s)}{\partial X_{ri}} \\
&= \frac{1}{M}\sum_{k=1}^{M} 2(X_{sk} - \mu_s)\left(\frac{\partial X_{sk}}{\partial X_{ri}} - \frac{\partial \mu_s}{\partial X_{ri}}\right) \\
&= \frac{2}{M}\sum_{k=1}^{M}(X_{sk} - \mu_s)(\delta_{rs}\delta_{ki} - \frac{\delta_{rs}}{M}) \\
&= \frac{2\delta_{rs}}{M}\sum_{k=1}^{M}(X_{sk} - \mu_s)(\delta_{ki} - \frac{1}{M}) \\
&= \frac{2\delta_{rs}}{M}\sum_{k=1}^{M}(X_{sk} - \mu_s)\delta_{ki} - \frac{2\delta_{rs}}{M}\sum_{k=1}^{M}\frac{(X_{sk} - \mu_s)}{M} \\
&= \frac{2\delta_{rs}}{M}(X_{si} - \mu_s) - \frac{2\delta_{rs}}{M}\sum_{k=1}^{M}\frac{X_{sk}}{M} + \frac{2\delta_{rs}}{M}\sum_{k=1}^{M}\frac{\mu_s}{M} \\
&= \frac{2\delta_{rs}}{M}(X_{si} - \mu_s) - \frac{2\delta_{rs}}{M}\mu_s + \frac{2\delta_{rs}}{M}\mu_s \\
&= \frac{2\delta_{rs}}{M}(X_{si} - \mu_s)
\end{aligned} \tag{6}$$

Question 3.2 d)

Batch normalisation (BN) addresses issues of high-order interactions in a network, where change in parameters in one layer affects the following layers in a chain reaction-like fashion, which requires a low learning rate to manage exploding/vanishing gradients. BN ensures that the values of the activations are independent of other layers, thus allowing higher learning rates and consequently faster training with strong gradients. Moreover, the noise of taking the statistics over the mini-batch offers model regularisation and mitigates overfitting.

However, this means that there is a lower bound limit on the batch size. The larger the batch size, the more stable the BN. The mini-batches need to be large enough to circumvent too noisy statistics. This may be problematic in settings where the batch sizes have to be small, for example, online learning tasks. Furthermore, BN is difficult to use with recurrent neural networks because it would require to store the statistics of each time step due to the varying sequence lengths.

Whereas BN takes statistics over the samples, layer normalisation (LN) takes statistics over the feature dimensions. LN, according to the original paper, has been designed to overcome the shortcoming of BN. Namely, it can be applied to recurrent neural networks in a straightforward manner, and works well especially for long sequences and smaller batch sizes. It also has been shown, empirically, to reduce training time for RNNs and improve generalisation performance.

LN is independent of the samples in the batch and is robust to the batch sizes, therefore, the change in batch size does not have an effect – LN can work with batch size 1. However, BN still outperforms LN in convolutional neural networks.

4 PyTorch CNN

Question 4 a)

The loss and accuracy curves of the ConvNet with default parameter values are shown in Figure 3. The test accuracy jumps drastically from the initial 0.12 at step 0 to 0.60 at step 500; it reaches 0.78 test accuracy around step 3000 and slowly increases to 0.80 by step 4500. The train and test curves for both loss and accuracy stay together, i.e. there is no divergence.

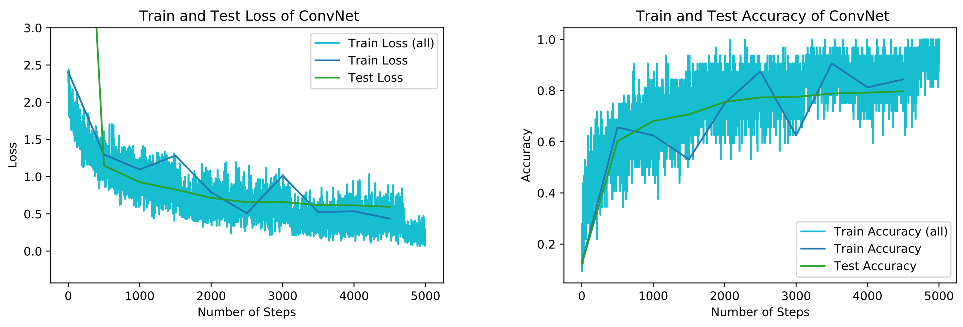

Figure 3: Cross entropy loss and accuracy values of Pytorch ConvNet classifier with default parameter values. The y-axis representing the loss has been capped for more detailed view of the loss curve – at step 0 the test loss is at around 10.