Assignment 1: MLPs, CNNs and Backpropagation

Victor Zuanazzi
12325724
University of Amsterdam
victorzuanazzi@gmail.com

1 MLP backprop and NumPy implementation

1.1 Analytical derivation of gradients

1.1.1

$$\frac{\partial L}{\partial x_i^{(N)}} = \frac{\partial\big(-\sum_i t_i \log x_i^{(N)}\big)}{\partial x_i^{(N)}} \tag{1}$$

$$= -\frac{t_i}{x_i^{(N)}} = \begin{cases} -\frac{1}{x_{argmax}^{(N)}} \\ 0 \end{cases} \quad \text{if } i \neq argmax \tag{2}$$

$$\frac{\partial L}{\partial x^{(N)}} = \left(\frac{-t}{x^{(N)}}\right) \in \mathbb{R}^{1\times d_N} \tag{3}$$

Where eq 3 is the vector version of 2 and the division denotes element wise division.

1.1.2

This derivative is more digestable if we compute it first for $i \neq j$, then for $i = j$. It follows that for $i \neq j$:

$$\frac{\partial x_i^{(N)}}{\partial \tilde{x}_j^{(N)}} = -\frac{\exp(\tilde{x}_i)\exp(\tilde{x}_j)}{(\sum_k \exp(\tilde{x}_k))^2} \tag{4}$$

$$= -\frac{\exp(\tilde{x}_i)}{(\sum_k \exp(\tilde{x}_k))}\frac{\exp(\tilde{x}_j)}{(\sum_k \exp(\tilde{x}_k))} = -x_i^{(N)}x_j^{(N)} \tag{5}$$

And for $i = j$:

$$\frac{\partial x_i^{(N)}}{\partial \tilde{x}_j^{(N)}} = \frac{\exp(\tilde{x}_i)\sum_k \exp(\tilde{x}_k) - \exp(\tilde{x}_j)^2}{(\sum_k \exp(\tilde{x}_k))^2} \tag{6}$$

$$= \frac{\exp(\tilde{x}_i)}{\sum_k \exp(\tilde{x}_k)}\frac{\sum_k \exp(\tilde{x}_k) - \exp(\tilde{x}_j)}{\sum_k \exp(\tilde{x}_k)} \tag{7}$$

$$= \frac{\exp(\tilde{x}_i)}{\sum_k \exp(\tilde{x}_k)}\big(1 - \frac{\exp(\tilde{x}_j)}{\sum_k \exp(\tilde{x}_k)}\big) \tag{8}$$

$$= x_i^{(N)}(1 - x_j^{(N)}) \tag{9}$$

By using the $\mathbb{1}$, the indicator function,we can write the 5 and 9 as follows

$$\frac{\partial x^{(N)}}{\partial \tilde{x}^{(N)}} = \Big(x_i^{(N)}(\mathbb{1}\{i=j\} - x_j^{(N)})\Big)_{i,j=1}^{d_N,d_N} \in \mathbb{R}^{d_N\times d_N} \tag{10}$$

Submitted to the Spring 2019 Amsterdam Deep Learning course

1.1.3

For $l < N$, all hidden layers.
The derivative of the ReLU function is not defined at $x = 0$, the following calculations consider it to be 1 at this point.

$$\frac{\partial x^{(l)}}{\partial \tilde{x}^{(l)}} = \frac{\partial}{\partial \tilde{x}^{(l)}} ReLU(\tilde{x}^{(l)}) \tag{11}$$

$$= \frac{\partial}{\partial \tilde{x}^{(l)}} \max(0, \tilde{x}^{(l)}) = \begin{cases} 1 & \text{if } \tilde{x}_i^{(l)} \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

$$= diag(\mathbb{1}\{\tilde{x}^{(l)} \geq 0\}) \in \mathbb{R}^{d_l \times d_l} \tag{13}$$

Where $diag$ indicates the terms sit on the main diagonal of the $d_l \times d_l$ matrix.

1.1.4

$$\frac{\partial \tilde{x}^{(l)}}{\partial x^{(l-1)}} = \frac{\partial}{\partial x^{(l-1)}} W^{(l)} x^{(l-1)} + b^{(l)} = W^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}} \tag{14}$$

$$\frac{\partial \tilde{x_i}^{(l)}}{\partial x_j^{(l-1)}} = \frac{\partial}{\partial x_j^{(l-1)}} \sum_j W_{ij}^{(l)} x_j^{(l-1)} + b_i^{(l)} = W_{ij}^{(l)} \tag{15}$$

Where eq. 14 is the vector version of eq. 15.

1.1.5

Let denote $\tilde{x} \in \mathbb{R}^m$ and $W \in \mathbb{R}^{m \times d}$.

$$\frac{\partial \tilde{x}_k^{(l)}}{\partial W_{ij}} = \frac{\partial}{\partial W_{ij}} \big(\sum_u W_{ku} x_u^{(l-1)} + b_k^{(l)} \big) = \begin{cases} x_j & \text{if } i = k \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

$$\frac{\partial \tilde{x}_k^{(l)}}{\partial W^{(l)}} = \begin{bmatrix} \frac{\partial \tilde{x}_k}{\partial W_{11}} & \cdots & \frac{\partial \tilde{x}_k}{\partial W_{1d}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \tilde{x}_k}{\partial W_{m1}} & \cdots & \frac{\partial \tilde{x}_k}{\partial W_{mn}} \end{bmatrix} \in R^{m \times d} \tag{17}$$

Thus, for the k-th row of the matrix:

$$\frac{\partial \tilde{x}_k^{(l)}}{\partial W^{(l)}} = \begin{bmatrix} & 0 & \\ x_1^{(l-1)} & \cdots & x_d^{(l-1)} \\ & 0 & \end{bmatrix} \in R^{m \times d} \tag{18}$$

Generally:

$$\frac{\partial \tilde{x}^{(l)}}{\partial W^{(l)}} = \begin{bmatrix} \begin{bmatrix} x_1^{(l-1)} & \cdots & x_d^{(l-1)} \\ & \vdots & \\ & 0 & \end{bmatrix} \\ \vdots \\ \begin{bmatrix} & 0 & \\ x_1^{(l-1)} & \cdots & x_d^{(l-1)} \\ & 0 & \end{bmatrix} \\ \vdots \\ \begin{bmatrix} & 0 & \\ & \vdots & \\ x_1^{(l-1)} & \cdots & x_d^{(l-1)} \end{bmatrix} \end{bmatrix} \in \mathbb{R}^{m \times (m \times d)} \tag{19}$$

1.1.6

$$\frac{\partial \tilde{x}^{(l)}}{\partial b^{(l)}} = \frac{\partial}{\partial b^{(l)}} W^{(l)} x^{(l-1)} + b^{(l)} = \mathbf{1} \in \mathbb{R}^{d_l \times d_l} \tag{20}$$

Question 1.1b

From 3 and 3:

1.1.7

$$\frac{\partial L}{\partial \tilde{x}^{(N)}} = \frac{\partial L}{\partial x^{(N)}} \frac{\partial x^{(N)}}{\partial \tilde{x}^{(N)}} \tag{21}$$

$$= \left(\frac{-t_i}{x^{(N)}} \right)_{i=1}^{d_N} \left(x_i^{(N)} (\mathbb{1}\{i=j\} - x_j^{(N)}) \right)_{i,j=1}^{d_N, d_N} \tag{22}$$

$$= [-t_1/x_1 \quad \dots \quad -t_{n_N}/x_{n_N}] \begin{bmatrix} x_1^{(N)}(1-x_1^{(N)}) & \dots & -x_1^{(N)} x_{d_N}^{(N)} \\ -x_2^{(N)} x_1^{(N)} & \dots & -x_2^{(N)} x_{d_N}^{(N)} \\ \vdots & \ddots & \vdots \\ -x_{d_N}^{(N)} x_1^{(N)} & \dots & x_{d_N}^{(N)}(1-x_{d_N}^{(N)}) \end{bmatrix} \in \mathbb{R}^{1 \times d_N} \tag{23}$$

Finaly, we can write 23 as:

$$\frac{\partial L}{\partial \tilde{x}^{(N)}} = \left(-x_i^{(N)} \sum_{j=1}^{d_N} \frac{t_j}{x_j} \left(\mathbb{1}\{i=j\} - x_j^{(N)} \right) \right)_{i=1}^{d_N} \in \mathbb{R}^{1 \times d_N} \tag{24}$$

1.1.8

From eq 13 we can derive the following.

$$\frac{\partial L}{\partial \tilde{x}^{(l)}} = \frac{\partial L}{\partial x^{(l)}} \frac{\partial x^{(l)}}{\partial \tilde{x}^{(l)}} = \frac{\partial L}{\partial x^{(l)}} diag(\mathbb{1}\{\tilde{x}^l > 0\}) \mathbb{R}^{d_u \times d_u} \tag{25}$$

1.1.9

From eq 15 we can derive the following.

$$\frac{\partial L}{\partial x^{(l)}} = \frac{\partial L}{\partial \tilde{x}^{(l+1)}} \frac{\partial \tilde{x}^{(l+1)}}{\partial x^{(l)}} = \frac{\partial L}{\partial \tilde{x}^{(l+1)}} W^{(l+1)} \tag{26}$$

1.1.10

From eqs. 24 and 19 we can derive the dimensionality of $\frac{\partial L}{\partial W^{(l)}}$.

$$\frac{\partial L}{\partial W^{(l)}} = \frac{\partial L}{\partial \tilde{x}^{(l)}} \frac{\partial \tilde{x}^{(l)}}{\partial W^{(l)}} \in \mathbb{R}^{1 \times d_l \times d_{l-1}} \tag{27}$$

In order to avoid a tensor product, we can use the particular composition of $\frac{\partial \tilde{x}^{(l)}}{\partial W^{(l)}}$ to find the partial derivative in question. Let's denote

$$\frac{\partial L}{\partial \tilde{x}^{(l)}} = [l_1, \dots, l_{d_l}] \tag{28}$$

Then we have

$$\frac{\partial L}{\partial \tilde{x}^{(l)}} \frac{\partial \tilde{x}^{(l)}}{\partial W^{(l)}} = [l_1, \ldots, l_{d_l}] \begin{bmatrix} \begin{bmatrix} x_1^{(l-1)} & \ldots & x_{d_{l-1}}^{(l-1)} \\ & \vdots & \\ & 0 & \end{bmatrix} \\ \vdots \\ \begin{bmatrix} & 0 & \\ & \vdots & \\ x_1^{(l-1)} & \ldots & x_{d_{l-1}}^{(l-1)} \end{bmatrix} \end{bmatrix} \tag{29}$$

$$= \begin{bmatrix} l_1 x_1^{(l-1)} & l_1 x_2^{(l-1)} & \ldots & l_1 x_{d_{l-1}}^{(l-1)} \\ \vdots & \ddots & & \vdots \\ l_{d_l} x_1^{(l-1)} & \ldots & & l_{d_l} x_{d_{l-1}}^{(l-1)} \end{bmatrix} \tag{30}$$

$$= [l_1, \ldots, l_{d_l}]^T \begin{bmatrix} x_1^{(l-1)} \ldots x_{d_{l-1}}^{(l-1)} \end{bmatrix} \tag{31}$$

$$= \left[\frac{\partial L}{\partial \tilde{x}^{(l)}} \right]^T x^{(l-1)} \in \mathbb{R}^{1 \times d_l \times d_{l-1}} \tag{32}$$

1.1.11

From eq 20 we can derive the following.

$$\frac{\partial L}{\partial b^{(l)}} = \frac{\partial L}{\partial \tilde{x}^{(l)}} \frac{\partial \tilde{x}^{(l)}}{\partial b^{(l)}} \tag{33}$$

$$= \frac{\partial L}{\partial \tilde{x}^{(l)}} \mathbf{1} = \frac{\partial L}{\partial \tilde{x}^{(l)}} \in \mathbb{R}^{1 \times d_l} \tag{34}$$

Question 1.1c

All derivatives with respect to x, and by consequence $\tilde{x}$ will have an extra dimension to store the derivatives with respect to each data point of the batch. The same will happen to the derivatives with respect to b, the most intuitive explanation to it is that the bias is a feature with constant value across data points, so we have to store the derivatives with respect to b for each evaluated data point of the batch. The dimensions will change to:

$$\frac{\partial L}{\partial \tilde{x}^{(l)}} \in \mathbb{R}^{1 \times n_l \times B} \tag{35}$$

$$\frac{\partial L}{\partial b^{(l)}} \in \mathbb{R}^{1 \times n_l \times B} \tag{36}$$

$$\frac{\partial L}{\partial \tilde{x}^{(N)}} \in \mathbb{R}^{1 \times d_N \times B} \tag{37}$$

$$\frac{\partial \tilde{x}^{(l)}}{\partial b^{(l)}} \in \mathbb{R}^{d_l \times d_l \times B} \tag{38}$$

$$\frac{\partial \tilde{x}^{(l)}}{\partial x^{(l-1)}} \in \mathbb{R}^{d_l \times d_{l-1} \times B} \tag{39}$$

$$\frac{\partial x^{(l)}}{\partial \tilde{x}^{(l)}} \in \mathbb{R}^{d_l \times d_l \times B} \tag{40}$$

$$\frac{\partial x^{(N)}}{\partial \tilde{x}^{(N)}} \in \mathbb{R}^{d_N \times d_N \times B} \tag{41}$$

$$\frac{\partial L}{\partial x^{(N)}} \in \mathbb{R}^{1 \times d_N \times B} \tag{42}$$

1.2 NumPy Implementation

The accuracy and loss curves of the implementation using the default parameters are shown in figure 1 and 2.

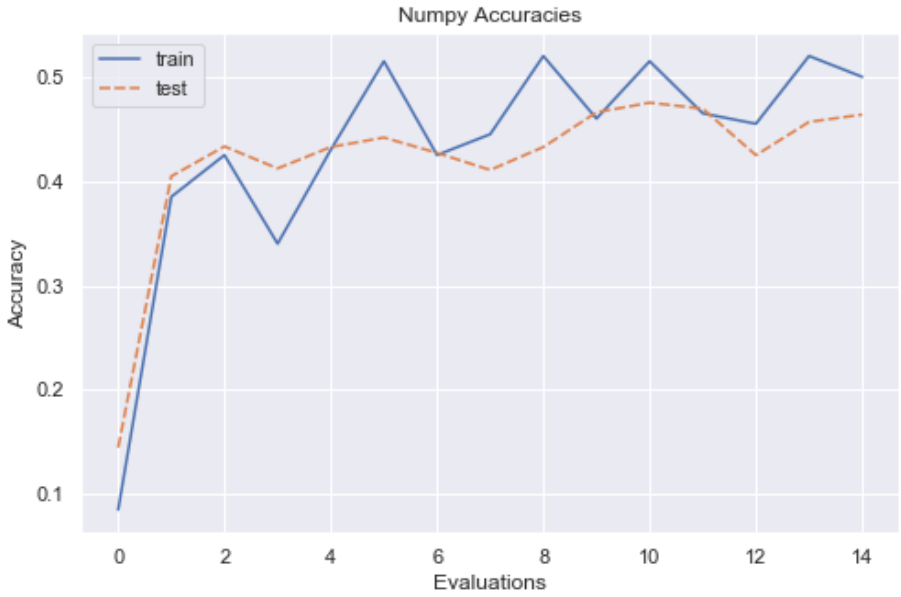

Figure 1: Numpy MLP Accuracies

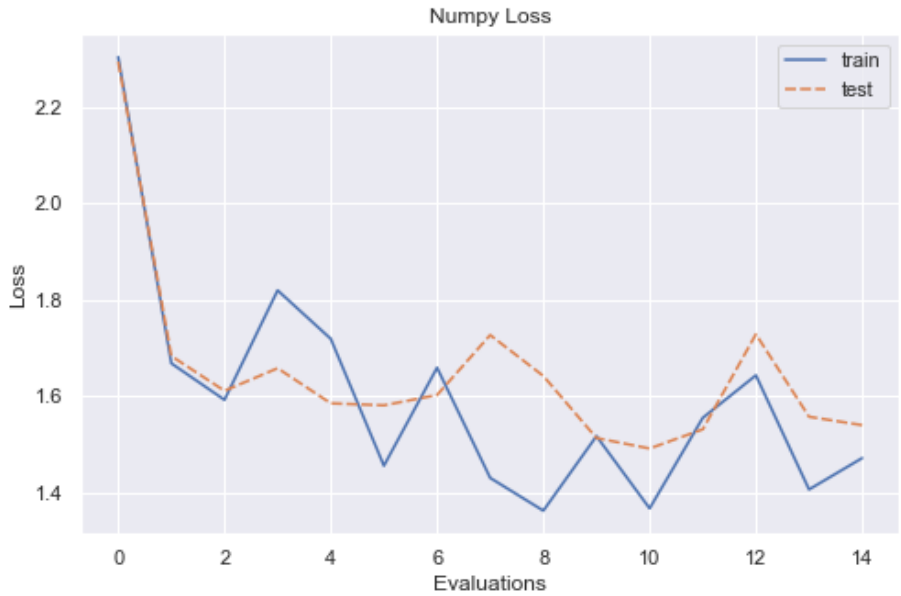

Figure 2: Numpy MLP Loss

2 PyTorch MLP

3 Custom Module: Batch Normalization

By using the default parameters the model could not achieve 50% accuracy in the test set.

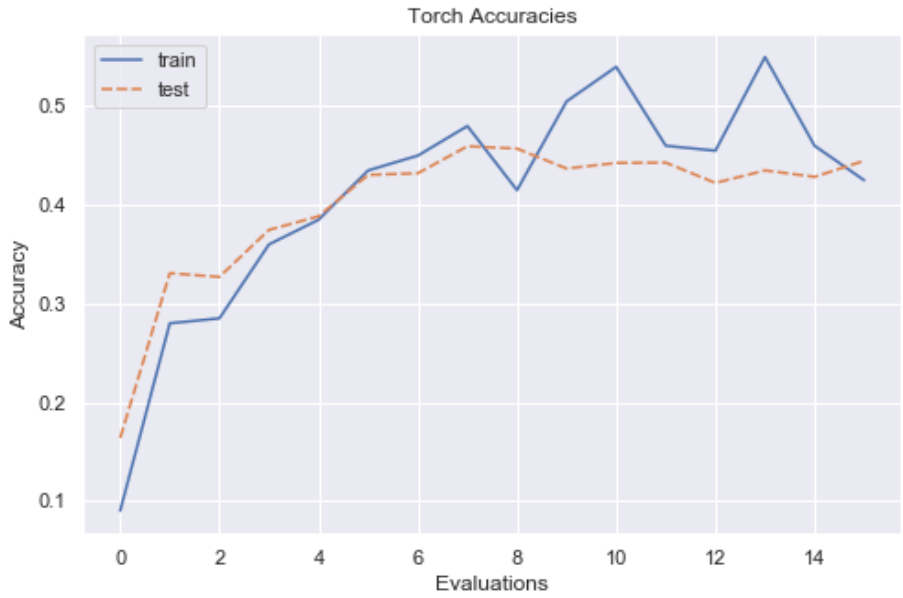

Figure 3: Torch MLP Accuracies

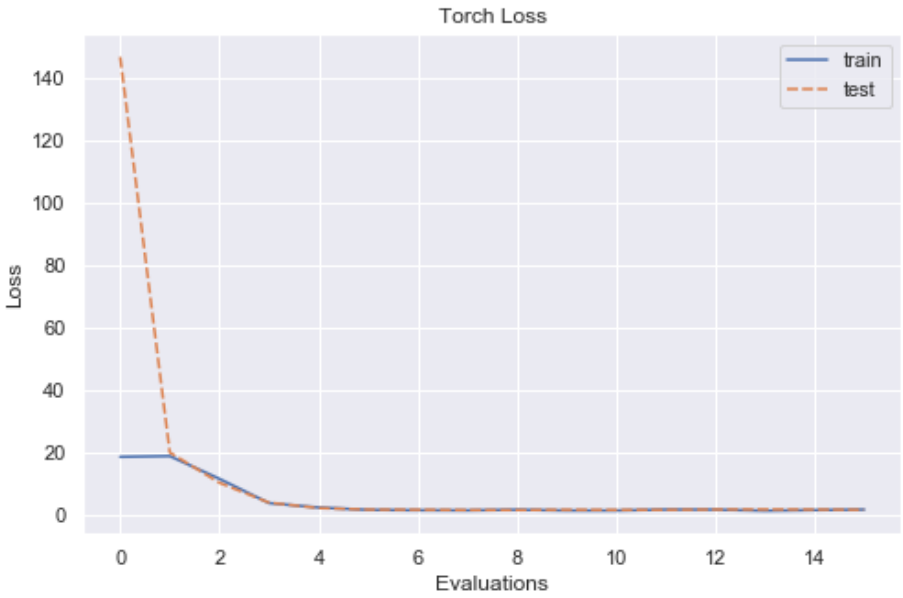

Figure 4: Torch MLP Loss

I played with different hyper-parameters to get a feeling of how each one of them impacts the performance of the model. However, hand tuning hyper-parameters can be very time consuming and

	hidden layers	learning rate	batch size	max steps	Accuracy Train	Accuracy Test
Default	100	2e-3	200	1500	42.5%	44.4%
GA1	1000, 500, 250, 100	1.68e-4	1609	4681	100%	53.7%
GA1 + dropout	1000, 500, 250, 100	1.68e-4	1609	4681	63.5%	49.7%
GA1 + dropout + batchnorm	1000, 500, 250, 100	1.68e-4	1609	4681	82.6%	51.6%
GA2	1000, 100	1.50e-4	1824	3030	100%	50,6%
GA3	1000, 1000	5.72e-5	809	4365	100%	53.5%

Table 1: Comparison between archtectures

ineffective, I decided to optimize the process. Instead of doing an exhaustive grid search or naive random search, I implemented a simple evolutionary algorithm to carry the optimization. The task was to optimize for test accuracy. The details of the implementation are beyond the scope of this report, the code is attached to this report.

The optimizer managed to find multiple architectures that completely overfit the test set. Some of them are shown below. The 52% test accuracy was achieved by GA1 and GA3 in table 1. It is interesting to see how the two, very different architectures achieve similar results.

To come to a model that does not overfit, I added a dropout layer after every linear layer. The configuration of GA1 was used. As in the table 1 *GA + dropout* does not overfit the training set. The performance in the test set is, however, negatively impacted. In an attempt to improve the model further, a batch normalization layer was added before every linear layer. However, accuracy of the test data did not improve further.

The figures 3 and 4 show the progress of the loss and accuracy for the default model. The figures 5 and 6 show the progress of loss and accuracy for GA1.

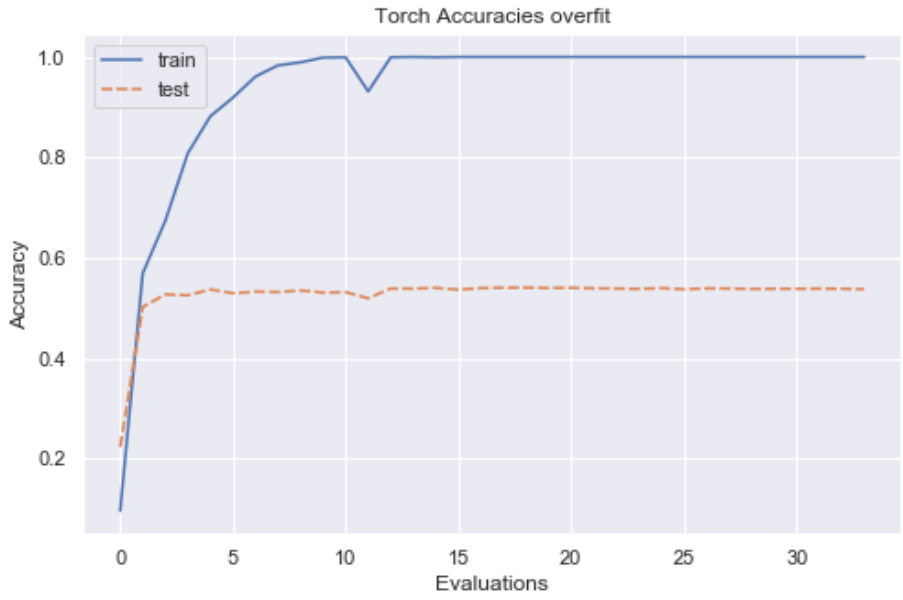

Figure 5: Torch GA1 MLP Accuracies

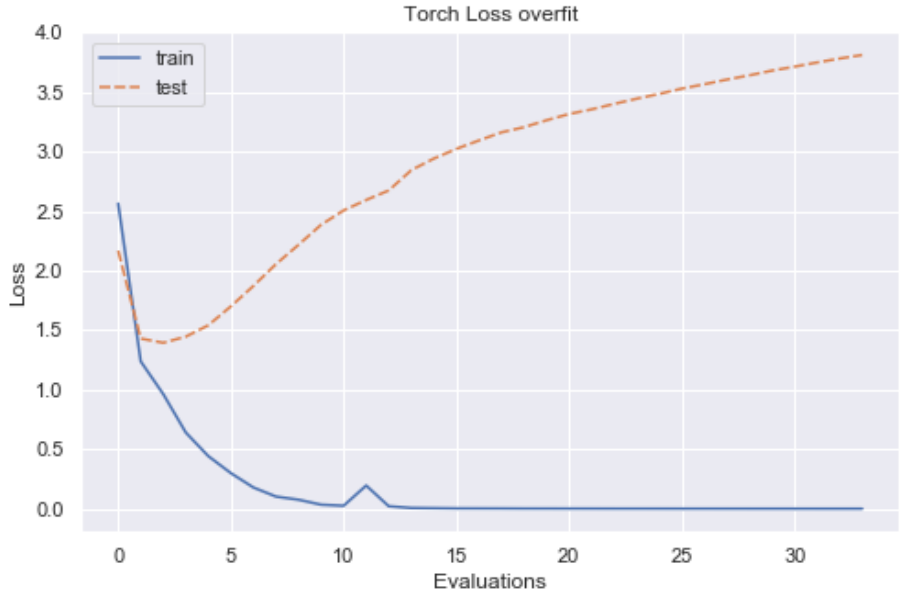

Figure 6: Torch MLP Loss

3.1 Automatic differentiation

The implementation follows attached to the report.

3.2 Manual implementation of backwards pass

a)

We have that

$$\frac{\partial y_i^s}{\partial \gamma_j} = \mathbb{1}\{i=j\}\hat{x}_j^s \tag{43}$$

$$\left(\frac{\partial L}{\partial \gamma}\right)_j = \sum_s \sum_i \frac{\partial L}{\partial y_i^s}\frac{\partial y_i^s}{\partial_j} \tag{44}$$

$$= \sum_s \sum_i \frac{\partial L}{\partial y_i^s}\mathbb{1}\{i=j\}\hat{x}_j^s \tag{45}$$

$$= \sum_s \frac{\partial L}{\partial y_i^s}\hat{x}_i^s \tag{46}$$

Similar applies with respect to β.

$$\frac{\partial y_i^s}{\partial \beta_j} = \mathbb{1}\{i=j\} \tag{47}$$

$$\left(\frac{\partial L}{\partial \beta}\right)_j = \sum_s \sum_i \frac{\partial L}{\partial y_i^s}\frac{\partial y_i^s}{\partial \beta} \tag{48}$$

$$= \sum_s \sum_i \frac{\partial L}{\partial y_i^s}\mathbb{1}\{i=j\} \tag{49}$$

$$= \sum_s \frac{\partial L}{\partial y_i^s} \tag{50}$$

The last derivative is much uglier. I will present it in pieces.

$$\left(\frac{\partial L}{\partial x}\right)_j^r = \sum_s \sum_i \frac{\partial L}{\partial y_i^s} \frac{\partial y_i^s}{\partial x_j^r} \tag{51}$$

$$\frac{\partial y_i^s}{\partial x_j^r} = \frac{\partial}{\partial x_j^r}\left(\gamma_i \hat{x}_i^s + \beta_i\right) = \gamma_i \frac{\partial \hat{x}_i^s}{\partial x_j^r} \tag{52}$$

$$\frac{\partial \hat{x}_i^s}{\partial x_j^r} = \frac{\partial}{\partial x_j^r}\left(\frac{x_i^s - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}\right) \tag{53}$$

$$= \frac{1}{\sigma_i^2 + \epsilon}\left[\left(\mathbb{1}\{i=j\}\mathbb{1}\{r=s\} - \frac{\partial \mu_i}{\partial x_j^r}\right)\sqrt{\sigma_i^2 + \epsilon} - \frac{x_i^s - \mu_i}{2\sqrt{\sigma_i^2 + \epsilon}}\frac{\partial \sigma_i^2}{\partial x_j^r}\right] \tag{54}$$

$$= \frac{\mathbb{1}\{i=j\}\mathbb{1}\{r=s\} - \frac{\partial \mu_i}{\partial x_j^r}}{\sqrt{\sigma_i^2 + \epsilon}} - \frac{x_i^s - \mu_i}{2(\sigma_i^2 + \epsilon)^{\frac{3}{2}}}\frac{\partial \sigma_i^2}{\partial x_j^r} \tag{55}$$

$$\frac{\partial \mu_i}{\partial x_j^r} = \frac{\partial}{\partial x_j^r}\left(\frac{1}{B}\sum_{s=1}^{B} x_i^s\right) \tag{56}$$

$$= \mathbb{1}\{i=j\}\frac{1}{B}\sum_{s=1}^{B}\mathbb{1}\{s=r\} = \frac{\mathbb{1}\{i=j\}}{B} \tag{57}$$

$$\frac{\partial \sigma_i^2}{\partial x_j^r} = \frac{\partial}{\partial x_j^r}\left(\frac{1}{B}\sum_{s=1}^{B}(x_i^s - \mu_i)^2\right) \tag{58}$$

$$= \mathbb{1}\{i=j\}\frac{1}{B}\sum_{s=1}^{B}\left(\frac{\partial}{\partial x_j^r}(x_i^s - \mu_i)^2\right) \tag{59}$$

$$= \frac{2}{B}\sum_s = 1^B(x_i^s - \mu_i)\left(\mathbb{1}\{i=j\}\mathbb{1}\{s=r\} - \frac{\mathbb{1}\{i=j\}}{B}\right) \tag{60}$$

$$= \frac{2\mathbb{1}\{i=j\}}{B}\left((x_i^r - \mu_i) - \frac{1}{B}\sum_{s=1}^{B}(x_i^s - \mu_i)\right) \tag{61}$$

$$= \frac{2\mathbb{1}\{i=j\}}{B}(x_i^r - \mu_i) \tag{62}$$

We can then use 62, 57, 55 to 52 calculate 51.

$$\left(\frac{\partial L}{\partial x}\right)_j^r = \sum_s \sum_i \frac{\partial L}{\partial y_i^s}\frac{\partial y_i^s}{\partial x_j^r} \tag{63}$$

$$= \sum_s \sum_i \frac{\partial L}{\partial y_i^s}\gamma_i \left[\frac{\mathbb{1}\{i=j\}\mathbb{1}\{s=r\} - \frac{\mathbb{1}\{i=j\}}{B}}{\sqrt{\sigma_i^2+\epsilon}} - \frac{x_i^s-\mu_i}{2(\sigma_i^2+\epsilon)^{\frac{3}{2}}}\frac{2\mathbb{1}\{i=j\}}{B}(x_i^r-\mu_i)\right] \tag{64}$$

$$= \sum_s \sum_i \frac{\partial L}{\partial y_i^s}\gamma_i \mathbb{1}\{i=j\}\frac{B-1}{B}\left[\frac{\mathbb{1}\{r=s\}}{\sqrt{\sigma_i^2+\epsilon}} - \frac{x_i^s-\mu_i}{B(\sigma_i^2+\epsilon)^{\frac{3}{2}}}(x_i^r-\mu_i)\right] \tag{65}$$

$$= \sum_s \sum_i \frac{\partial L}{\partial y_i^s}\gamma_i \mathbb{1}\{i=j\}\frac{B-1}{B}\left[\frac{\mathbb{1}\{r=s\}}{\sqrt{\sigma_i^2+\epsilon}} - \frac{x_i^s-\mu_i}{B(\sigma_i^2+\epsilon)^{\frac{3}{2}}}\right] \tag{66}$$

$$= \sum_s \frac{\partial L}{\partial y_j^s}\gamma_j \frac{B-1}{B}\left[\frac{\mathbb{1}\{r=s\}}{\sqrt{\sigma_j^2+\epsilon}} - \frac{x_j^s-\mu_j}{B(\sigma_j^2+\epsilon)^{\frac{3}{2}}}\right] \tag{67}$$

$$= \frac{\gamma_j}{B\sqrt{\sigma_j^2+\epsilon}}\sum_s \frac{\partial L}{\partial y_j^s}\left(B\mathbb{1}\{r=s\} - 1 - \frac{x_j^s-\mu_j}{\sigma_j^2+\epsilon}\right) \tag{68}$$

$$= \frac{\gamma_j}{B\sqrt{\sigma_j^2+\epsilon}}\left(B\frac{\partial L}{\partial y_j^s} - \frac{\partial L}{\partial \beta_j} - \frac{\partial L}{\partial \gamma_j}\frac{x_j^s-\mu_j}{\sigma_j^2+\epsilon}\right) \tag{69}$$

$$\tag{70}$$

4 PyTorch CNN

The convnet implementation follows attached. The results can be seen in figures 7 and 8. As expected, the conv-net outperforms its MLP counterpart.

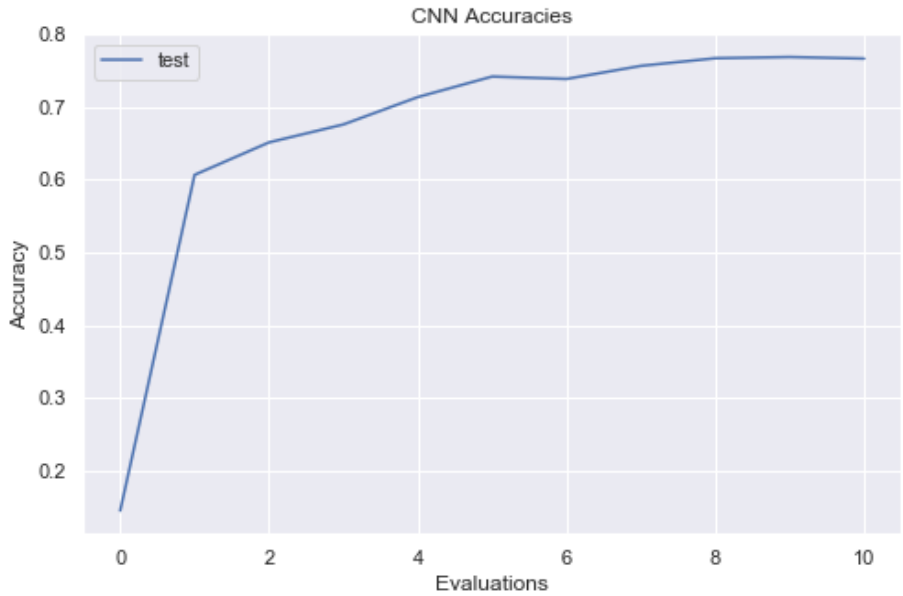

Figure 7: Torch CNN Accuracies

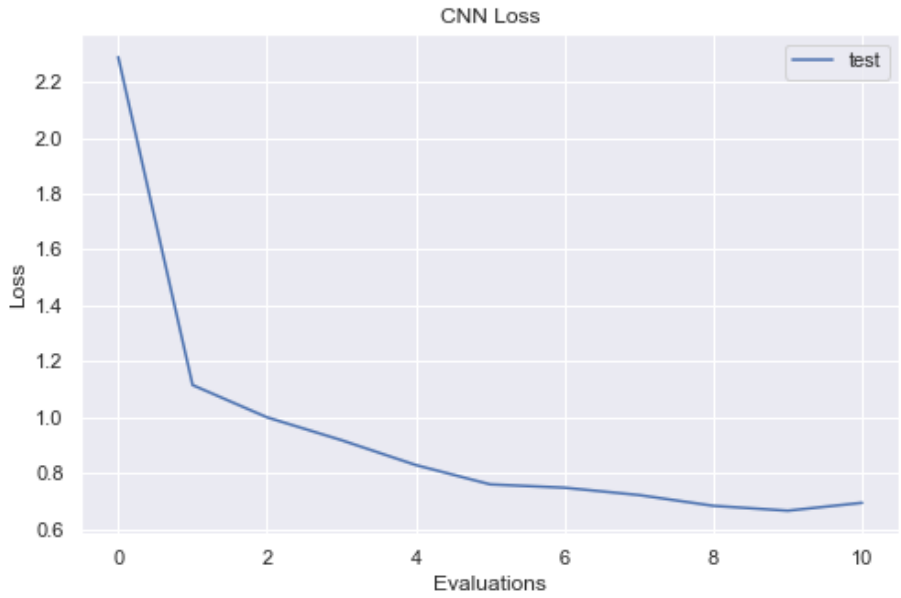

Figure 8: Torch CNN Loss

References

The implementation of this code can be found in the repository: https://github.com/VictorZuanazzi/Deep_Learning