DeepLearning Assignment 1

Harm Manders
10677186
University of Amsterdam

1 MLP backprop and NumPy Implementation

1.1 Evaluating the Gradients

a) Linear Module

Find closed form expressions for $\frac{\partial L}{\partial \mathbf{W}}, \frac{\partial L}{\partial b}, \frac{\partial L}{\partial \mathbf{X}}$ in terms of the gradients of the loss with respect to the output features.

Derivative with respect to W

$$\begin{aligned}
\frac{\partial L}{\partial W_{ij}} &= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \frac{\partial Y_{mn}}{\partial W_{ij}} \\
\frac{\partial Y mn}{\partial W_{ij}} &= \frac{\partial}{\partial W_{ij}}(X \cdot W + b) = X \\
\frac{\partial L}{\partial W_{ij}} &= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} X \\
&= X^T \frac{\partial L}{\partial Y}
\end{aligned}$$

Derivative with respect to X

$$\begin{aligned}
\frac{\partial L}{\partial X_{ij}} &= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \frac{\partial Y_{mn}}{\partial X_{ij}} \\
\frac{\partial Y_{mn}}{\partial X_{ij}} &= \frac{\partial}{\partial X_{ij}}(X \cdot W + b) = W \\
\frac{\partial L}{\partial X_{ij}} &= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} W \\
&= \frac{\partial L}{\partial Y} W^T
\end{aligned}$$

Derivative with respect to b

$$
\begin{aligned}
\frac{\partial L}{\partial b_{ij}} &= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \frac{\partial Y_{mn}}{\partial b_i} \\
\frac{\partial Y_{mn}}{\partial X_{ij}} &= \frac{\partial}{\partial X_{ij}} (X \cdot W + b) = \mathbf{1} \\
\frac{\partial L}{\partial b_{ij}} &= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \mathbf{1} \\
&= \mathbf{1}_{1,m} \frac{\partial L}{\partial Y}
\end{aligned}
$$

b) Activation Module

$$
\begin{aligned}
\frac{\partial L}{\partial X_{ij}} &= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \frac{\partial Y_{mn}}{\partial b_i} \\
Y &= h(X) \\
\frac{\partial Y}{\partial X_{ij}} &= h'(X) \\
\frac{\partial L}{\partial X_{ij}} &= \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \circ h'(X) \\
&= \frac{\partial L}{\partial Y} \circ h'(X)
\end{aligned}
$$

c) Softmax and Loss Modules

1.2 NumPy Implementation

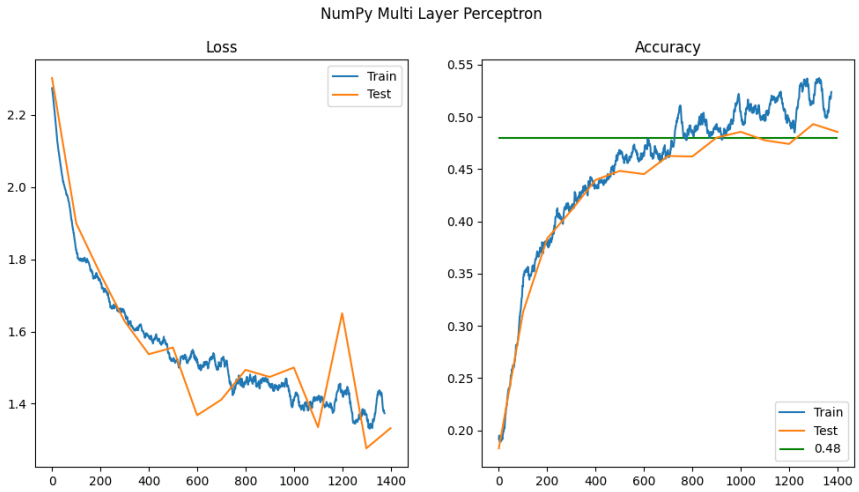

Figure 1: Test/train scores of the MLP Numpy model

2 PyTorch MLP

2.1 PyTorch Implementation

When training the model with default parameters the model reaches approximately 0.49 test accuracy.

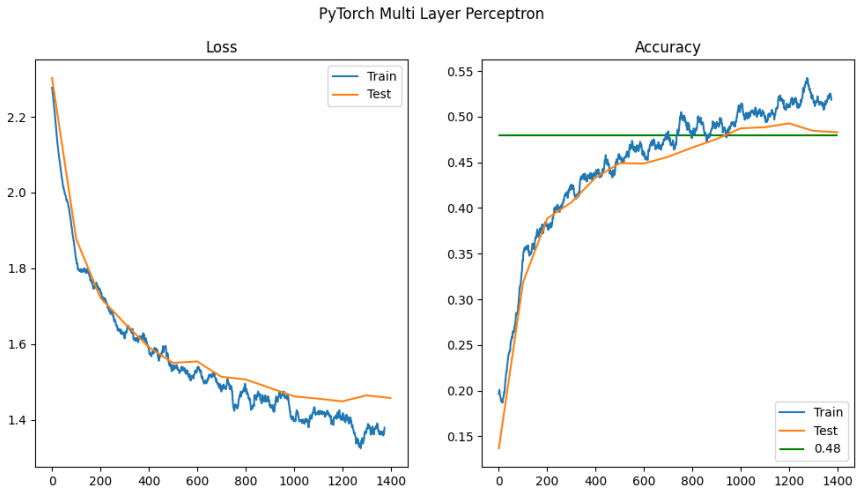

Figure 2: Test/train scores of the MLP PyTorch model

Different hyper-parameters are explored to see what kind of accuracy's can be reached.

Wide and Shallow approach

```
python train_mlp_pytorch.py --dnn_hidden_units 800,600,400
--optim adam --learn_rate 0.0001 --max_steps 5000
```

Here I make a very wide model with lots of nodes for each hidden layer but not a lot of hidden layers. The model tends to over-fit and the test accuracy stays around 0.53.

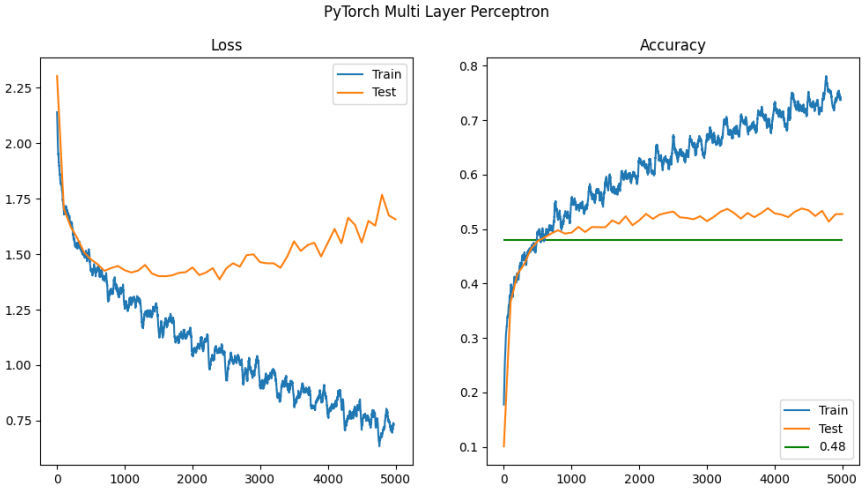

Figure 3: Test/train scores of a wide MLP PyTorch model.
Shallow and wide model with 3 hidden layers 800,600,400 nodes respectively.

Deep and Narrow approach

```
python train_mlp_pytorch.py --dnn_hidden_units 100,100,50,50,50,50
--optim adam --learn_rate 0.0001 --max_steps 5000
```

Here a model is made with a lot of layers which don't have a lot of nodes.
The model seems to over-fit less than the wide-shallow model but the test accuracy also stays around 0.53

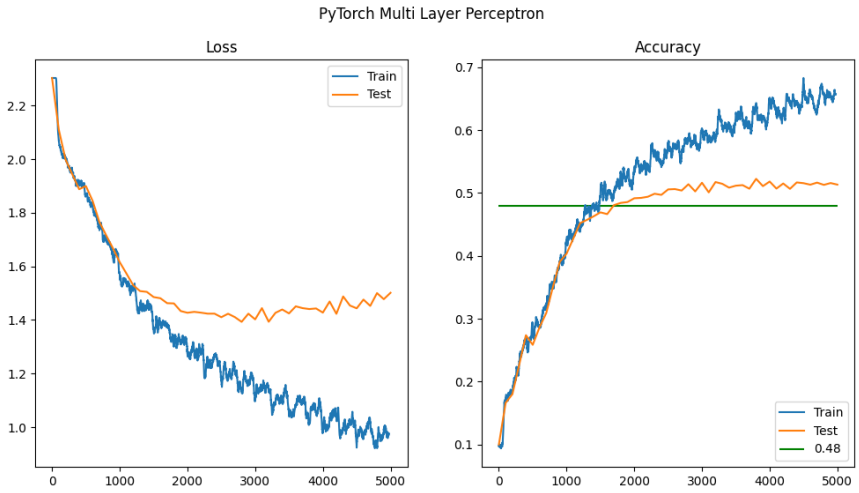

Figure 4: Test/train scores of a deeper MLP PyTorch model.
Deep and narrow model with 6 hidden layers 100,100,50,50,50,50 nodes respectively.

2.2 Benefits and/or Drawbacks of Tanh

Advantages

- ELU is computationally very cheap
- TanH has a smooth gradient (no sharp turn at x=0)
- Tanh will produce a number between -1 and 1 making it also which reduces the effect of exploding layers.

3 Custom Module: Layer Normalization

3.1 Comparing Layer with Batch Normalization

With Batch Normalization you normalise your features batch wise. This is useful because features with different value ranges will become centered around 0 with a variance of 1. The problem with Batch Normalization is that the batch size has a lot of impact on the noisiness of the normalization. If you have a small batch size there can be quite some difference between batches but if batches are bigger the variance will reduce.

With Layer Normalization you don't find this batch size dependence.

4 PyTorch CNN

4.1 Transfer Learning

```
python train_convnet_pytorch.py --pretrained
```

This pre-trained model performs pretty well on our data set. No layers were frozen for this test and only the final classification layer was replaced.
This model seems to be less prone to over-fitting.

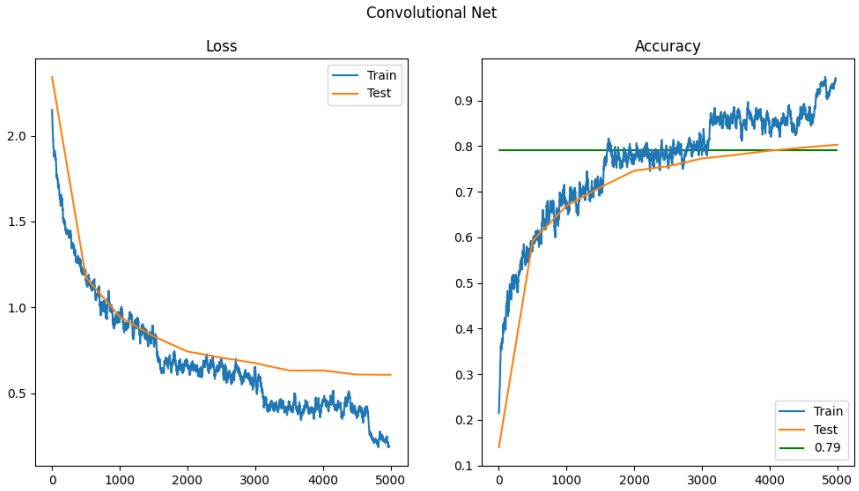

Figure 5: Test/train scores of the PyTorch CNN model.

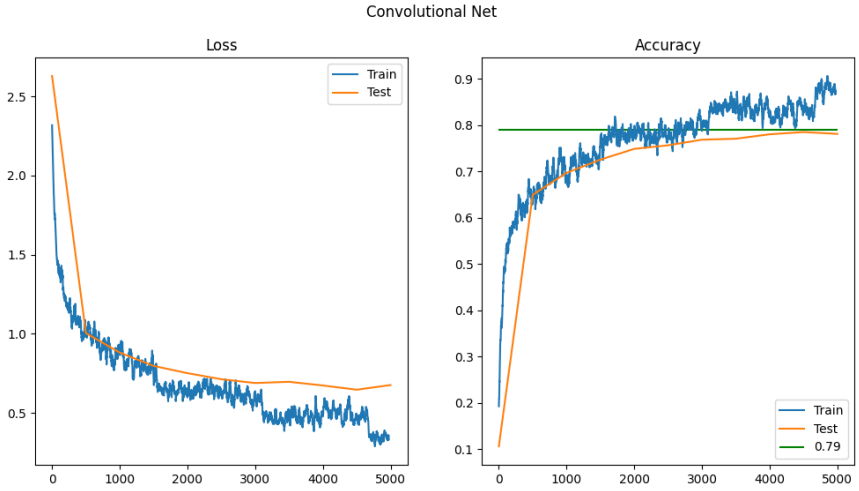

Figure 6: Test/train scores of ResNet18.