Deep Learning: Assignment 1

Dhruba Pujary
University of Amsterdam
11576200
`dhruba.pujary@student.uva.nl`

1 MLP backprop and NumPy implementation

Linear: $\hat{x}^{(l)} = W^{(l)}x^{(l-1)} + b^{(l)}, l = 1, \ldots, N$
ReLU: $ReLU(\hat{x}^{(l)}) = max(0, \hat{x}^{(l)})$
Softmax: $softmax(\hat{x}^{(N)}) = \frac{exp(\hat{x}^N)}{\sum_{i=1}^{d_N} exp(\hat{x}^N)_i}$
Cross Entropy: $L(x^{(N)}, t) = -\sum_i t_i \log x_i^{(N)}$

1.1 Analytically derivation of Gradients

Question 1.1 a

1. $\frac{\partial L}{\partial \mathbf{x}^{(N)}}$

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

$$= -\frac{t_j}{x_j^{(N)}}, \forall j \in 1, \ldots d_N \tag{2}$$

$$\frac{\partial L}{\partial \mathbf{x}^{(N)}} = \begin{bmatrix} -\frac{t_1}{x_1^N} \\ . \\ . \\ . \\ -\frac{t_{d_N}}{x_{d_N}^N} \end{bmatrix} \tag{3}$$

2. $\frac{\partial \mathbf{x}^N}{\partial \hat{\mathbf{x}}^{(N)}}$
Lets define δ_{ij} such that for i=j, $\delta_{ij} = 1$, and 0 otherwise. This definition will be used throughout the assignment.

$$\frac{\partial x_i^N}{\partial \hat{x}_j^{(N)}} = \frac{\partial\big(\frac{exp(\hat{x}_i^{(N)})}{\sum exp(\hat{x}_i^{(N)})}\big)}{\partial \hat{x}_j^{(N)}} \tag{4}$$

$$= \frac{\delta_{ij} exp(\hat{x}_i^{(N)}) \sum_i^{(N)} exp(\hat{x}_i^{(N)}) - exp(\hat{x}_i^{(N)}) \sum_{i=1}^{d_N} \delta_{ij} exp(\hat{x}_i^{(N)})}{\big(\sum_{i=1}^{d_N} exp(\hat{x}_i^{(N)})\big)^2} \tag{5}$$

Case: i=j (6)

$$= \frac{exp(\hat{x}_i^{(N)}) \sum_i^{(N)} exp(\hat{x}_i^{(N)}) - exp(\hat{x}_i^{(N)}) exp(\hat{x}_i^{(N)})}{\big(\sum_{i=1}^{d_N} exp(\hat{x}_i^{(N)})\big)^2} \tag{7}$$

Case: i≠j (8)

$$= -\frac{exp(\hat{x}_i^{(N)}) exp(\hat{x}_j^{(N)})}{\big(\sum_{i=1}^{d_N} exp(\hat{x}_i^{(N)})\big)^2} \tag{9}$$

$$\tag{10}$$

Lets define $\frac{exp(\hat{x}_i^{(N)})}{\sum_i^{(N)} exp(\hat{x}_i^{(N)})} = x_i^{(N)}$, and similarly $\frac{exp(\hat{x}_j^{(N)})}{\sum_i^{(N)} exp(\hat{x}_i^{(N)})} = x_j^{(N)}$ as column vectors, then in matrix notation

$$\frac{\partial \mathbf{x}^N}{\partial \hat{\mathbf{x}}^{(N)}} = \begin{bmatrix} x_1^{(N)}(1 - x_1^{(N)}) & -x_1^{(N)} x_2^{(N)} & \dots & -x_1^{(N)} x_{d_N}^{(N)} \\ -x_2^{(N)} x_1^{(N)} & x_2^{(N)}(1 - x_2^{(N)}) & \dots & -x_1^{(N)} x_{d_N}^{(N)} \\ \vdots & \vdots & \vdots & \vdots \\ -x_{d_N}^{(N)} x_1^{(N)} & -x_1^{(N)} x_2^{(N)} & \dots & -x_{d_N}^{(N)}(1 - x_{d_N}^{(N)}) \end{bmatrix} \tag{11}$$

This can be computer in matrix form as follows: $\mathtt{x} \odot (\mathtt{I} - \mathtt{x}^T)$ (with broadcast for correct matrix operation)

3. $\frac{\partial \mathbf{x}^{(l<N)}}{\partial \hat{\mathbf{x}}^{(l<N)}}$

$$\frac{\partial x_i^{(l<N)}}{\partial \hat{x}_j^{(l<N)}} = \frac{\partial(max(0, \hat{x}_i^{(l)}))}{\partial \hat{x}_j^l} \tag{12}$$

$$= \delta_{ij} \frac{\partial(max(0, \hat{x}_i^{(l)}))}{\partial \hat{x}_j^l} \tag{13}$$

Case: i=j (14)

$$= \frac{\partial(max(0, \hat{x}_i^{(l)}))}{\partial \hat{x}_j^l} \tag{15}$$

$$= \begin{cases} 0, \text{ if } \hat{x}_i^{(l)} \leq 0 \\ 1, \text{ otherwise} \end{cases} \tag{16}$$

Case: i≠j (17)

$$= 0 \tag{18}$$

$$\frac{\partial \mathbf{x}^{(l<N)}}{\partial \hat{\mathbf{x}}^{(l<N)}} = \begin{bmatrix} \begin{cases} 0, \text{ if } \hat{x}_1^{(l)} \leq 0 \\ 1, \text{ otherwise} \end{cases} & 0 & \dots & 0 \\ 0 & \begin{cases} 0, \text{ if } \hat{x}_2^{(l)} \leq 0 \\ 1, \text{ otherwise} \end{cases} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \begin{cases} 0, \text{ if } \hat{x}_{d_{(l)}}^{(l)} \leq 0 \\ 1, \text{ otherwise} \end{cases} \end{bmatrix} \tag{19}$$

4. $\frac{\partial \hat{\mathbf{x}}^{(l)}}{\partial \mathbf{x}^{(l-1)}}$

$$\frac{\partial \hat{x}_i^{(l)}}{\partial x_j^{(l-1)}} = \frac{\partial(\sum_{k=1}^{d_{l-1}} W_{ik}^{(l)} x_k^{(l-1)} + b_i)}{\partial x_j^{(l-1)}} \tag{20}$$
$$= \sum_{k=1}^{d_{(l-1)}} W_{ik}^{(l)} \delta_{kj} \tag{21}$$
$$= W_{ij}^{(l)} \tag{22}$$

$$\frac{\partial \hat{\mathbf{x}}^{(l)}}{\partial \mathbf{x}^{(l-1)}} = \begin{bmatrix} W_{11} & W_{12} & \dots & W_{1d_{l-1}} \\ W_{21} & W_{22} & \dots & W_{2d_{l-1}} \\ \vdots & \vdots & \vdots & \vdots \\ W_{d_l 1} & W_{12} & \dots & W_{1d_{l-1}} \end{bmatrix} \tag{23}$$

5. $\frac{\partial \hat{x}^{(l)}}{\partial W^{(l)}}$

$$\frac{\partial \hat{x}_i^{(l)}}{\partial W_{jk}^{(l)}} = \frac{\partial(\sum_{s=1}^{d_{l-1}} W_{is}^{(l)} x_s^{(l-1)} + b_i^{(l)})}{\partial W_{jk}^{(l)}} \tag{24}$$
$$= \sum_{s=1}^{d_{l-1}} \delta_{ij}\delta_{ks} x_s^{(l-1)} \tag{25}$$
$$= \delta_{ij} x_k^{(l-1)} \tag{26}$$
$$\text{Case: i=j} \tag{27}$$
$$= x_k^{(l-1)} \tag{28}$$
$$\text{Case: i}\neq\text{j} \tag{29}$$
$$= 0 \tag{30}$$

This will eventually be a 3-Dimensional matrix which can be squeezed along dim(l) to get 2D matrix (removes the zero entries).

6. $\frac{\partial \hat{\mathbf{x}}^{(l)}}{\partial \mathbf{b}^{(l)}}$

$$\frac{\partial \hat{x}_i^{(l)}}{\partial b_j^{(l)}} = \frac{\partial(\sum_{s=1^{d_{l-1}}} W^{(l) x_s^{(l-1)}} + b_i^{(l)})}{\partial b_j^{(l)}} \tag{31}$$
$$= \delta_{ij} \tag{32}$$
$$\text{Case: i=j} \tag{33}$$
$$= 1 \tag{34}$$
$$\text{Case: i} \neq \text{j} \tag{35}$$
$$= 0 \tag{36}$$

$$\frac{\partial \hat{\mathbf{x}}^{(l)}}{\partial \mathbf{b}^{(l)}} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} \tag{37}$$

Question 1.1 b

1. $\frac{\partial L}{\partial \hat{\mathbf{x}}^{(N)}} = \frac{\partial L}{\partial \mathbf{x}^{(N)}} \frac{\partial \mathbf{x}^N}{\partial \hat{\mathbf{x}}^{(N)}}$

   This is direct matrix multiplication of $(\frac{\partial L}{\partial \mathbf{x}^{(N)}})^T$ which is (1,d_N) dimension and $\frac{\partial \mathbf{x}^N}{\partial \hat{\mathbf{x}}^{(N)}}$ is of (d_N, d_N)

2. $\frac{\partial L}{\partial \mathbf{x}^{(l<N)}} = \frac{\partial L}{\partial \mathbf{x}^{(l+1)}} \frac{\partial \mathbf{x}^{(l+1)}}{\partial \hat{\mathbf{x}}^{(l<N)}}$

$$\frac{\partial L}{\partial x_j^{(l<N)}} = \sum_{i=1}^{d_{l+1}} \frac{\partial L}{\partial \hat{x}_i^{(l+1)}} \frac{\partial \hat{x}_i^{(l+1)}}{\partial x_j^{(l<N)}} \tag{38}$$

$$= \sum_{i=1}^{d_{l+1}} \frac{\partial L}{\partial \hat{x}_i^{(l+1)}} W_{ij}^{(l+1)} \tag{39}$$

$$\frac{\partial L}{\partial \mathbf{x}^{(l<N)}} = (\mathbf{W}^{(l+1)})^T \frac{\partial L}{\partial \hat{\mathbf{x}}^{(l+1)}} \tag{40}$$

3. $\frac{\partial L}{\partial \hat{\mathbf{x}}^{(l<N)}} = \frac{\partial L}{\partial \mathbf{x}^{(l<N)}} \frac{\partial \mathbf{x}^{(l<N)}}{\partial \hat{\mathbf{x}}^{(l<N)}}$

$$\frac{\partial L}{\partial \hat{x}_j^{(l<N)}} = \frac{\partial L}{\partial x_j^{(l<N)}} \frac{\partial x_j^{(l<N)}}{\partial \hat{x}_j{}^{(l<N)}} \tag{41}$$

$$= \frac{\partial (max(0, \hat{x}_j^{(l)}))}{\partial \hat{x}_j^l} \sum_{i=1}^{d_{l+1}} \frac{\partial L}{\partial \hat{x}_i^{(l+1)}} W_{ij}^{(l+1)} \tag{42}$$

$$\tag{43}$$

$$\frac{\partial L}{\partial \hat{\mathbf{x}}^{(l<N)}} = (\mathbf{W}^{(l+1)})^T \frac{\partial L}{\partial \hat{\mathbf{x}}^{(l+1)}} \frac{\partial \mathbf{x}^{(l<N)}}{\partial \hat{\mathbf{x}}^{(l<N)}} \tag{44}$$

4. $\frac{\partial L}{\partial \mathbf{W}^{(l)}} = \frac{\partial L}{\partial \hat{\mathbf{x}}^{(l)}} \frac{\partial \hat{\mathbf{x}}^{(l)}}{\partial \mathbf{W}^{(l)}}$

$$\frac{\partial L}{\partial W_{jk}^{(l)}} = \sum_{i=1}^{d_l} \frac{\partial L}{\partial \hat{x}_i^{(l)}} \frac{\partial \hat{x}_i^{(l)}}{\partial W_{jk}^{(l)}} \tag{45}$$

$$= \frac{\partial L}{\partial \hat{x}_i^{(l)}} x_k^{(l-1)} \tag{46}$$

$$\tag{47}$$

$$\frac{\partial L}{\partial \mathbf{W}^{(l)}} = \frac{\partial L}{\partial \hat{\mathbf{x}}_i^{(l)}} (\mathbf{x}^{(l-1)})^T \tag{48}$$

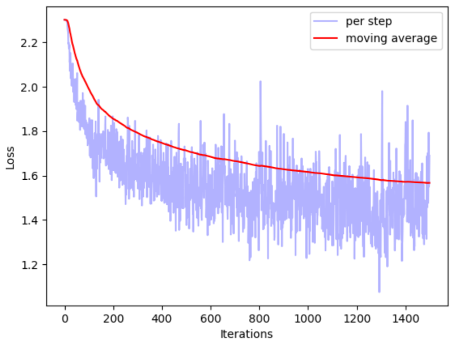
5. $\frac{\partial L}{\partial \mathbf{b}^{(l)}} = \frac{\partial L}{\partial \hat{\mathbf{x}}^{(l)}} \frac{\partial \hat{\mathbf{x}}^{(l)}}{\partial \mathbf{W}^{(l)}}$

$$\frac{\partial L}{\partial b_j^{(l)}} = \sum_{i=1}^{d_l} \frac{\partial L}{\partial \hat{x}_i^{(l)}} \frac{\partial \hat{x}_i^{(l)}}{\partial b_j^{(l)}} \tag{49}$$

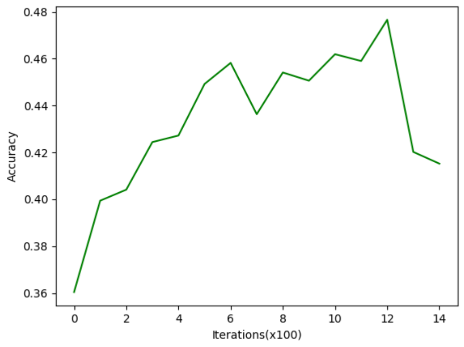
$$= \frac{\partial L}{\partial \hat{x}_j^{(l)}} \tag{50}$$

$$\frac{\partial L}{\partial \mathbf{b}^{(l)}} = \frac{\partial L}{\partial \hat{\mathbf{x}}^{(l)}} \tag{51}$$

$$\tag{52}$$



(a) Training Loss



(b) Test Accuracy

Figure 1: Loss and Accuray Curves for MLP NumPy implementation

Question 1.1 c

$L_{total} = \frac{1}{B} \sum_s^B L_{individual}\left(x^{(0),s}, t_s\right)$

To calculate $\frac{\partial \mathcal{L}}{\partial \mathbf{x}^N}$ we need to do it over batches :

$$\begin{aligned}
\frac{\partial L}{\partial (x_i^k)^N} &= \frac{1}{B} \partial \frac{\left(\sum_s^B L_{individual}\left(x^{(0),s}, t_s\right)\right)}{\partial L^k} \frac{\partial L^k}{\partial (x_i^k)^N} \\
&= -\frac{1}{B} \sum_s \delta_{k,s} \frac{\partial L^k}{\partial (x_i^k)^N} \\
&= -\frac{1}{B} \frac{t_i^k}{x_i^k}
\end{aligned}$$

The gradients calculation in last layer will be divided by batch size. Other calculation will remain the same.

1.2 NumPy implementation

Loss and Accuracy plot of numpy MLP for default parameters are in Figure 1.

2 PyTorch MLP

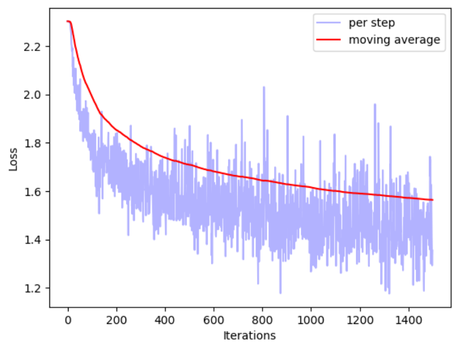
Question 2

Pytorch MLP loss and accuracy curves are in Figure 2. Small difference in accuracy is observed when running the code on GPU vs CPU.

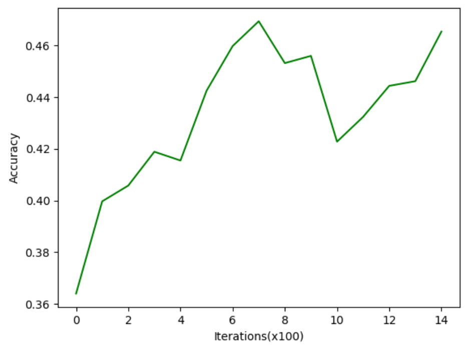
Experiments: The base model has 1 hidden layer with 100 neurons and trained using SGD optimizer with default settings.

Initialization and number of Iteration: Using default settings in pytoch. i.e default initialization of weights and bias gives best accuracy of 0.439 (Fig.3(b)) in 1500 iterations. Increasing the number of iteration to 3000 gives a better accuracy of 0.459. Also, the initial loss values in Fig. 3(a) shows that initialization of weights and bias is an area of improvement. The weights are initialized using a normal distribution with 0 mean and 1e-4 standard deviation and trained for 3000 iteration. Best accuracy achieved is 0.488 (Fig 4).

Learning Rate: From Fig. 4(a), the loss curve seems to converge with too much fluctuation on the accuracy. This indicates that the learning rate is big and reducing the learning rate would improve

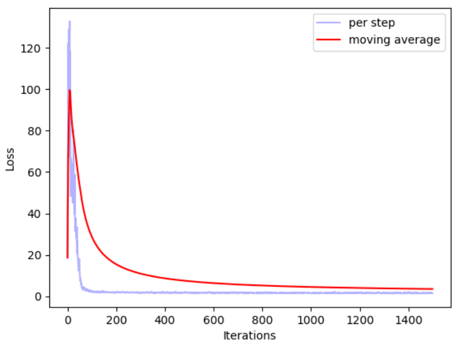


(a) Training Loss

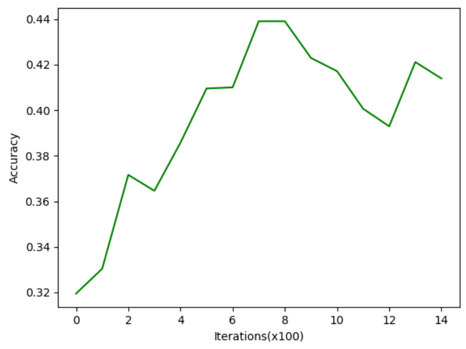


(b) Test Accuracy

Figure 2: Loss and Accuracy Curves for MLP Pytorch implementation

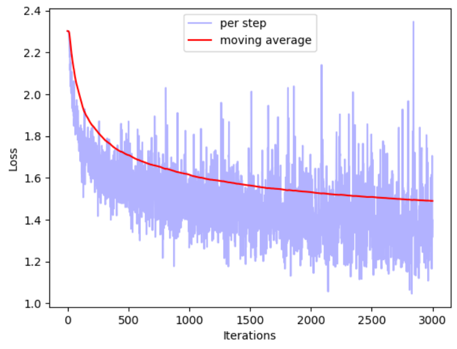


(a) Training Loss

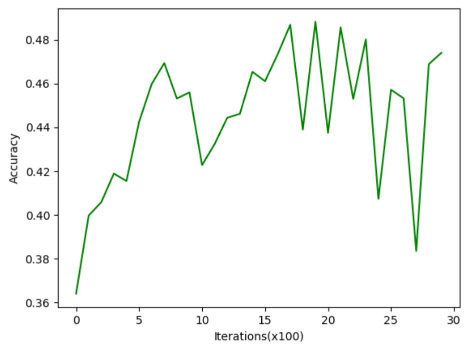


(b) Test Accuracy

Figure 3: Loss and Accuracy Curves with default parameters

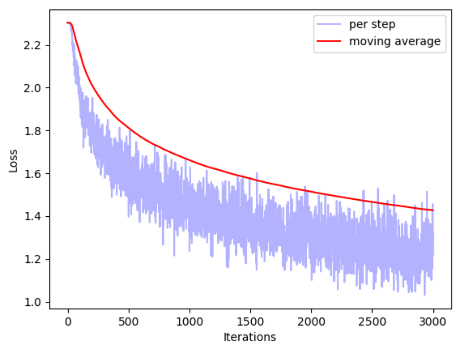


(a) Training Loss

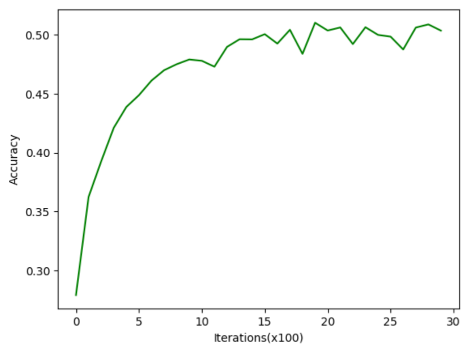


(b) Test Accuracy

Figure 4: Loss and Accuracy Curves with initialization and increasing number of Iteration

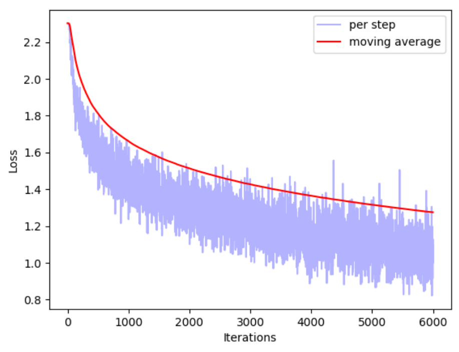


(a) Training Loss

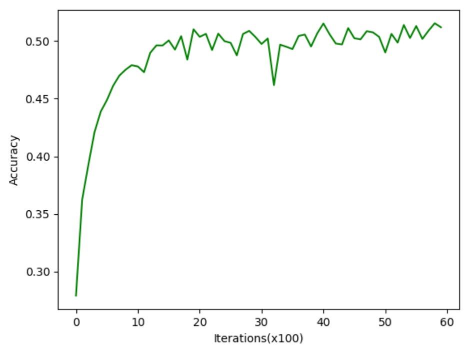


(b) Test Accuracy

Figure 5: Loss and Accuracy Curves with Learning Rate



(a) Training Loss



(b) Test Accuracy

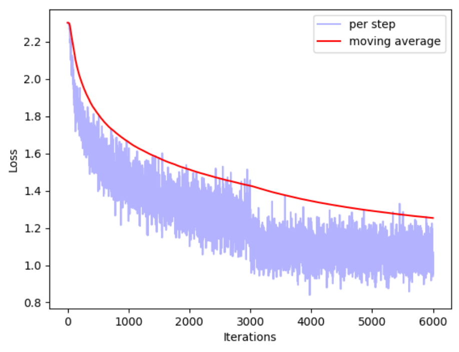
Figure 6: Loss and Accuracy Curves with increasing number of iteration

the model. A series of learning rate tests are carried and the one (lr = 8e-4) which gave the best accuracy of 0.51 is chosen (Fig. 5). Further increasing the number of iteration to 6000 gives very small improvement (Fig. 6).

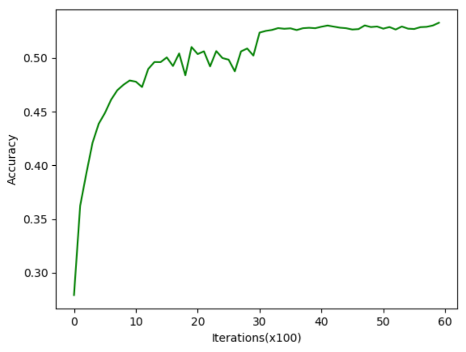
Learning Rate Decay: From Fig. 6(a) and 6(b), the model seems to converge again and the next modification would be decaying the learning rate after fixed number of iteration. The learning rate is dropped to one-tenth of current learning rate after half of training steps, i.e at 3000 step the learning rate is dropped from 8e-4 to 8e-5. The accuracy further improve to 0.532 by using learning rate decay schedule (Fig. 7).

Adding more neurons: The best accuracy obtained by using 1 hidden layer with 100 neurons is 0.532. Increasing the number of nodes to 400 improves the accuracy to 0.542, and can further be improved to 0.551 by having 1000 neurons. However, total number of parameters for the model with 1000 neurons would be around 3.08 million for input image size of 32x32x3 and 10 classes. The number of parameters for 400 neurons models is about 1.26 millions which is very less than 1000 neurons for the slight improvement in accuracy.

Adding more layers: If a models gives better accuracy just adding more layers and keeping the number of trainable parameters comparable then it is viable option. Thus the next modification is to introduce a new layer with weight initialization and initial learning rate for good accuracy. Using a models with two hidden layers with 300 and 400 neurons respectively gives a total number of 1.04 millions trainable parameters. However, the accuracy of the model trained using SGD is lower than expected (Fig. 8). To check the quality of the model, it was trained with ADAM optimizer and it gave

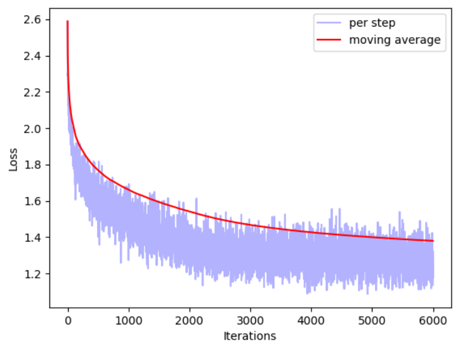


(a) Training Loss

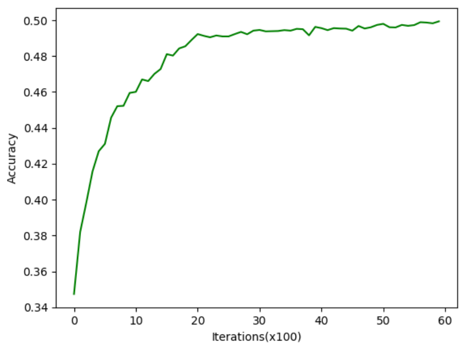


(b) Test Accuracy

Figure 7: Loss and Accuracy Curves with Learning rate schedule



(a) Training Loss



(b) Test Accuracy

Figure 8: Loss and Accuracy Curves with two hidden layers with SGD

0.543 accuracy (Fig. 9). Changing SGD to momentum with 0.9 gave better result with accuracy of 0.557. Using a different model with 400 and 600 neurons respectively for the two hidden layers gave much better accuracy of 0.562 with 1.47 millions trainable parameters. This model is considered to be considered the best model (Fig. 10).

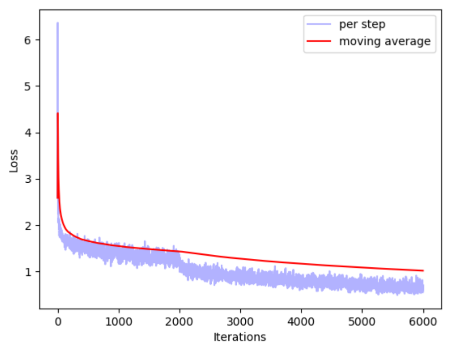
Other experiments: Batch Normalization was also used during experiments which remove the dependency on initialization and gives a steady progress during training. Also, using different batch size gave better estimates of gradients and the improvemt in accuracy as well as loss function is very steady. However these parameters are not changed for the best trained model.[1]

From Fig. 10(a), the best model has almost converged with loss getting close to zero which is a good sign. But by looking at the accuracy curve (Fig. 10(b)), after the scheduled learning rate drop, the model accuracy seems to come down gradually. This could be because of over fitting the model to training data. Training for more number of iteration would make the model worse.

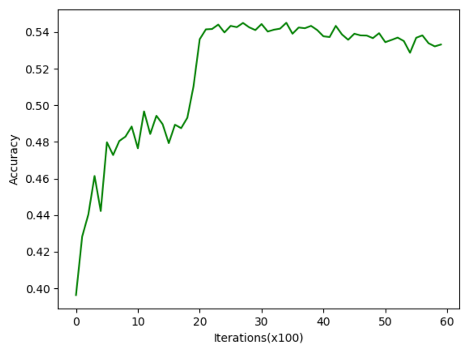
3 Custom Module: Batch Normalization

mean: $\mu_i = \frac{1}{B}\sum_{s=1}^{B} x_i^s$
variance: $\sigma_i^2 = \frac{1}{B}\sum_{s=1}^{B}(x_i^s - \mu_i)^2$

[1]because of time limitations.

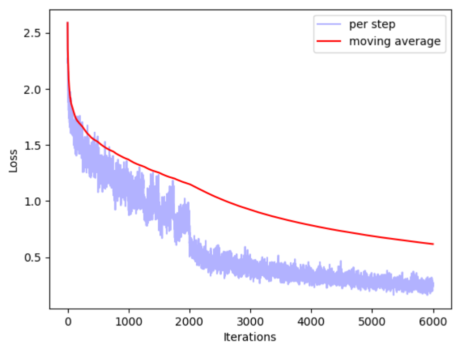


(a) Training Loss

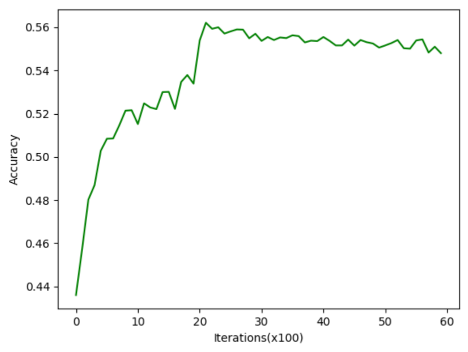


(b) Test Accuracy

Figure 9: Loss and Accuracy Curves with two hidden layers with ADAM



(a) Training Loss



(b) Test Accuracy

Figure 10: Loss and Accuracy Curves with two hidden layers (400,600) and SGD with momentum

normalize: $\hat{x}_i^s = \frac{x_i^s - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}$
$y_i^s = \gamma_i \hat{x}_i^s + \beta_i$

3.1 Manual implementation of backward pass

Question 3.2 a

1. $\frac{\partial L}{\partial \gamma} = \frac{\partial L}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \gamma}$

$$\frac{\partial L}{\partial \gamma_j} = \sum_s \sum_i \frac{\partial L}{\partial y_i^s} \frac{\partial y}{\partial \gamma_j} \tag{53}$$
$$= \sum_s \sum_i \frac{\partial L}{\partial y_i^s} \delta_{ij} \hat{x}_i^s \tag{54}$$
$$= \sum_s \frac{\partial L}{\partial y_j^s} \hat{x}_j^s \tag{55}$$
$$\tag{56}$$

2. $\frac{\partial L}{\partial \beta} = \frac{\partial L}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \beta}$

$$\frac{\partial L}{\partial \beta_j} = \sum_s \sum_i \frac{\partial L}{\partial y_i^s} \frac{\partial y}{\partial \beta_j} \tag{57}$$

$$= \sum_s \sum_i \frac{\partial L}{\partial y_i^s} \delta_{ij} \tag{58}$$

$$= \sum_s \frac{\partial L}{\partial y_j^s} \tag{59}$$

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

$$\frac{\partial L}{\partial x_j^r} = \sum_s \sum_i \frac{\partial L}{\partial y_i^s} \frac{\partial y_i^s}{\partial x_j^r} \tag{60}$$

$$= \sum_s \sum_i \frac{\partial L}{\partial y_i^s} \gamma_i \frac{\partial \hat{x}_i^s}{\partial x_j^r} \tag{61}$$

$$= \sum_s \sum_i \frac{\partial L}{\partial y_i^s} \gamma_i \frac{\partial (\frac{x_i^s - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}})}{\partial x_j^r} \tag{62}$$

$$= \sum_s \sum_i \frac{\partial L}{\partial y_i^s} \gamma_i (\frac{\frac{\partial x_i^s - \mu_i}{\partial x_j^r} (\sigma_i^2 + \epsilon)^{\frac{1}{2}} - (x_i^s - \mu_i) \frac{\partial (\sigma_i^2 + \epsilon)^{\frac{1}{2}}}{\partial x_j^r}}{\sigma_i^2 + \epsilon}) \tag{63}$$

$$= \sum_s \sum_i \frac{\partial L}{\partial y_i^s} \gamma_i \big((\delta_{ij} \delta_{sr} - \frac{\partial \mu_i}{\partial x_j^r})(\sigma_i^2 + \epsilon)^{\frac{-1}{2}} - (x_i^s - \mu_i) \frac{1}{2} (\sigma_i^2 + \epsilon)^{\frac{-3}{2}} \frac{\partial \sigma_i^2}{\partial x_j^r} \big) \tag{64}$$

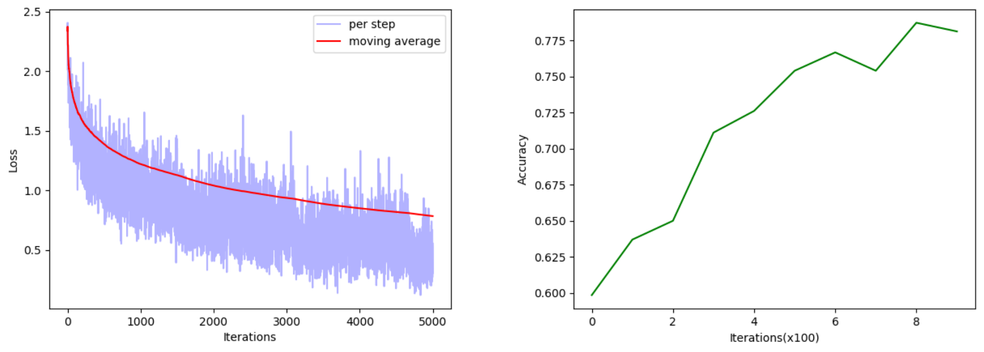
$$\text{Evaluating } \frac{\partial \sigma_i^2}{\partial x_j^r} \tag{65}$$

$$= \frac{\partial \frac{1}{B} \sum_{k=1}^B (x_i^k - \mu_i)^2}{\partial x_j^r} \tag{66}$$

$$= \frac{2}{B} \sum_{k=1}^B (x_i^k - \mu_i)(\delta_{ij} \delta_{kr} - \frac{1}{B} \delta_{ij}) \tag{67}$$

$$= \frac{2}{B} \sum_{k=1}^B (x_i^k - \mu_i) \delta_{ij} \delta_{kr} - \frac{2}{B} \sum_{k=1}^B (x_i^k - \mu_i) \frac{1}{B} \delta_{ij}) \tag{68}$$

$$\tag{69}$$



(a) Training Loss (b) Test Accuracy

Figure 11: Loss and Accuracy Curves of Conv Net

$$= \frac{2}{B}(x_i^r - \mu_i)\delta_{ij} - \frac{2}{B^2}\delta_{ij}\sum_{k=1}^{B} x_i^k + \frac{2}{B^2}\delta_{ij}\sum_{k=1}^{B}\mu_i \tag{70}$$

$$= \frac{2}{B}(x_i^r - \mu_i)\delta_{ij} - \frac{2}{B}\delta_{ij}\mu_i + \frac{2}{B}\delta_{ij}\mu_i \tag{71}$$

$$= \frac{2}{B}(x_i^r - \mu_i)\delta_{ij} \tag{72}$$

Putting back into the equation: (73)

$$= \sum_s\sum_i \frac{\partial L}{\partial y_i^s}\gamma_i\Big((\delta_{ij}\delta_{sr} - \frac{\partial \mu_i}{\partial x_j^r})(\sigma_i^2+\epsilon)^{\frac{-1}{2}} - (x_i^s-\mu_i)\frac{1}{2}(\sigma_i^2+\epsilon)^{\frac{-3}{2}}\frac{2}{B}(x_i^r-\mu_i)\delta_{ij}\Big) \tag{74}$$

$$= \sum_s\sum_i \frac{\partial L}{\partial y_i^s}\gamma_i(\delta_{ij}\delta_{sr} - \frac{1}{B}\delta_{ij})(\sigma_i^2+\epsilon)^{\frac{-1}{2}} - \tag{75}$$

$$\sum_s\sum_i \frac{\partial L}{\partial y_i^s}\gamma_i(x_i^s-\mu_i)\frac{1}{2}(\sigma_i^2+\epsilon)^{\frac{-3}{2}}\frac{2}{B}(x_i^r-\mu_i)\delta_{ij} \tag{76}$$

$$= \sum_s\sum_i \frac{\partial L}{\partial y_i^s}\gamma_i\delta_{ij}\delta_{sr}(\sigma_i^2+\epsilon)^{\frac{-1}{2}} - \sum_s\sum_i \frac{\partial L}{\partial y_i^s}\gamma_i\frac{1}{B}\delta_{ij}(\sigma_i^2+\epsilon)^{\frac{-1}{2}} - \tag{77}$$

$$\sum_s \frac{\partial L}{\partial y_j^s}\gamma_i(x_j^s-\mu_j)\frac{1}{2}(\sigma_j^2+\epsilon)^{\frac{-3}{2}}\frac{2}{B}(x_j^r-\mu_j) \tag{78}$$

$$= \frac{\partial L}{\partial y_j^r}\gamma_j(\sigma_i^2+\epsilon)^{\frac{-1}{2}} - \sum_s \frac{\partial L}{\partial y_j^s}\gamma_j\frac{1}{B})(\sigma_j^2+\epsilon)^{\frac{-1}{2}} - \tag{79}$$

$$(\sigma_j^2+\epsilon)^{\frac{-3}{2}}\frac{\gamma_i}{B}(x_j^r-\mu_j)\sum_s \frac{\partial L}{\partial y_j^s}(x_j^s-\mu_j) \tag{80}$$

$$= \frac{\gamma_j}{B}(\sigma_i^2+\epsilon)^{\frac{-1}{2}}\Big(B\frac{\partial L}{\partial y_j^r} - \sum_s \frac{\partial L}{\partial y_j^s} - (x_j^r-\mu_j)(\sigma_j^2+\epsilon)^{-1}\sum_s \frac{\partial L}{\partial y_j^s}(x_j^s-\mu_j)\Big) \tag{81}$$

(82)

4 PyTorch CNN

:

Using Convolutional Neural Networks, gives the best accuracy of 0.78. With increasing number of iteration, the models accuracy is increasing as well as the loss functions is decreasing. The initial model accuracy at 500 step is 0.58 which itself shows the representational power of deep neural architectures.