EP3260:Fundamentals of Machine Learning over Networks Computer Assignment 6

Group 3

Notation

Upper-case letters with a double underline denotes matrices, e.g., $\underline{\underline{\boldsymbol{A}}}$. Lower-case letters with a single underline denotes vectors, e.g., $\underline{\boldsymbol{a}}$. The i-th element of the vector $\underline{\boldsymbol{a}}$ is denoted by either $a\left[i\right]$ or a_i, and element in the i-th row and j-th column of the matrix $\underline{\underline{\boldsymbol{A}}}$ is denoted by $\underline{\underline{\boldsymbol{A}}}\left[i,j\right]$. An i-th column vector of a matrix $\underline{\underline{\boldsymbol{A}}}$ is denoted as either $\underline{\boldsymbol{A}}_i$ or $\underline{\underline{\boldsymbol{A}}}[:,i]$. $\mathrm{Diag}\left(\underline{\boldsymbol{a}}\right)$ creates a diagonal matrix by stacking the vector $\underline{\boldsymbol{a}}$ elements along the diagonal elements, whereas $\mathrm{diag}\left(\underline{\underline{\boldsymbol{A}}}\right)$ renders a vector by extracting the diagonal elements of the matrix $\underline{\underline{\boldsymbol{A}}}$. The element-wise multiplication and division are denoted by $\odot$ and $\oslash$, respectively.

Computer Assignment 6

Consider MNIST dataset. Consider a deep neural network (DNN) with J layers and $\{N_j\}$ neurons on layer j.

a) Train DNN using SGD and your choices of hyper-parameters, L, and $\{N_j\}_{j\in[J]}$. Report the convergence rate on the training as well as the generalization performance. Feel free to change SGD to any other solver of your choice (give explanation for the choice).

b) Repeat part a) with mini-batch GD of your choice of the mini-batch size, retrain DNN, and show the performance measures. Compare the training performance (speed, accuracy) using various adaptive learning rates (constant, diminishing, AdaGrad, RMSProp).

c) Consider design of part a). Fix $\sum_j N_j$. Investigate shallower networks (smaller J) each having potentially more neurons versus deeper network each having fewer neurons per layer, and discuss pros and cons of these two DNN architectures.

d) Split the dataset to 6 random disjoint subsets, each for one worker, and repeat part a) on master-worker computational graph.

e) Consider a two-star topology with communication graph $(1,2)-3-4-(5,6)$. Repeat part a) using your choice of distributed optimization solver. You can add communication efficiency to the iterations, if you like!

f) To promote sparse solutions, you may use $\ell 1$ regularization or a so-called dropout technique. Explain how you incorporate each of these approaches in the training? Compare their training performance and the size of the final trained models.

g) Improving the smoothness of an optimization landscape may substantially improve the convergence properties of first-order iterative algorithms. Batch-normalization is a relatively simple technique to smoothen the landscape [Santurkar-2018]. Using the materials of the course, propose an alternative approach to improve the smoothness. Provide numerical justification for the proposed approach.

1 Preliminaries

Figure 1 illustrates the (fully-connected) feed forward total J layers DNN, i.e., with $(J-1)$ hidden layers. Observe that 0-th layer interface implies an input layer with N_0 neurons corresponding to the length of the input feature vector $\boldsymbol{x}_i$, and the J-th layer implies an output layer with N_J neurons corresponding to the number of outputs that are observed. Moreover, note that the input layer is not counted in the total number of layers in this DNN nomenclature.

For the given dataset $\mathcal{D} \coloneqq \{\boldsymbol{x}_i, \boldsymbol{t}_i\}_{i=1}^N$, the output of a DNN, at the given i-th input sample $\boldsymbol{x}_i \in \mathbb{R}^{N_0}$ can be described by

$$\boldsymbol{y}_i \coloneqq f^{(J)}\left(\boldsymbol{W}^{(J)} f^{(J-1)}\left(\boldsymbol{W}^{(J-1)} \cdots f^{(1)}\left(\boldsymbol{W}^{(1)}\boldsymbol{x}_i + \boldsymbol{b}_1\right) \cdots + \boldsymbol{b}^{(J-1)}\right) + \boldsymbol{b}^{(J)}\right) , \tag{1}$$

where $\boldsymbol{W}^{(j)}$ is a weight matrix connecting neurons between j and $j-1$ layers such that the dimension of the matrix is $N_{L_j} \times N_{L_{j-1}}$, and N_{L_j} corresponds to the number of neurons at j-th hidden layer interface. The bias is denoted by $\boldsymbol{b}^{(j)}$, and the activation function at j-th layer is $f^{(j)}$.

We assume an $L2$ squared norm cost function of a (fully-connected) DNN, i.e.,

$$
\begin{aligned}
& L\left(\left\{\boldsymbol{W}^{(j)}, \boldsymbol{b}^{(j)}\right\}_{j=1}^{J}\right) \\
& \quad := \frac{1}{2N}\sum_{i=1}^{N}\left\| f^{(J)}\left(\boldsymbol{W}^{(J)} f^{(J-1)}\left(\boldsymbol{W}^{(J-1)}\cdots f^{(1)}\left(\boldsymbol{W}^{(1)}\boldsymbol{x}_i + \boldsymbol{b}_1\right)\cdots + \boldsymbol{b}^{(J-1)}\right) + \boldsymbol{b}^{(J)}\right) - \boldsymbol{t}_i\right\|_2^2 \\
& \quad + \lambda R\left(\left\{\boldsymbol{W}^{(j)}, \boldsymbol{b}^{(j)}\right\}_{j=1}^{J}\right)
\end{aligned} \tag{2}
$$

where the actual/observed output corresponding to an i-th sample is $\boldsymbol{t}_i \in \mathbb{R}^{N_{L_J}}$. The regularization factor is λ and the regularization function is $R\left(\left\{\boldsymbol{W}^{(j)}, \boldsymbol{b}^{(j)}\right\}_{j=1}^{J}\right)$. For this project, we have employed ℓ_2 norm as a regularization function of the weights and biases.

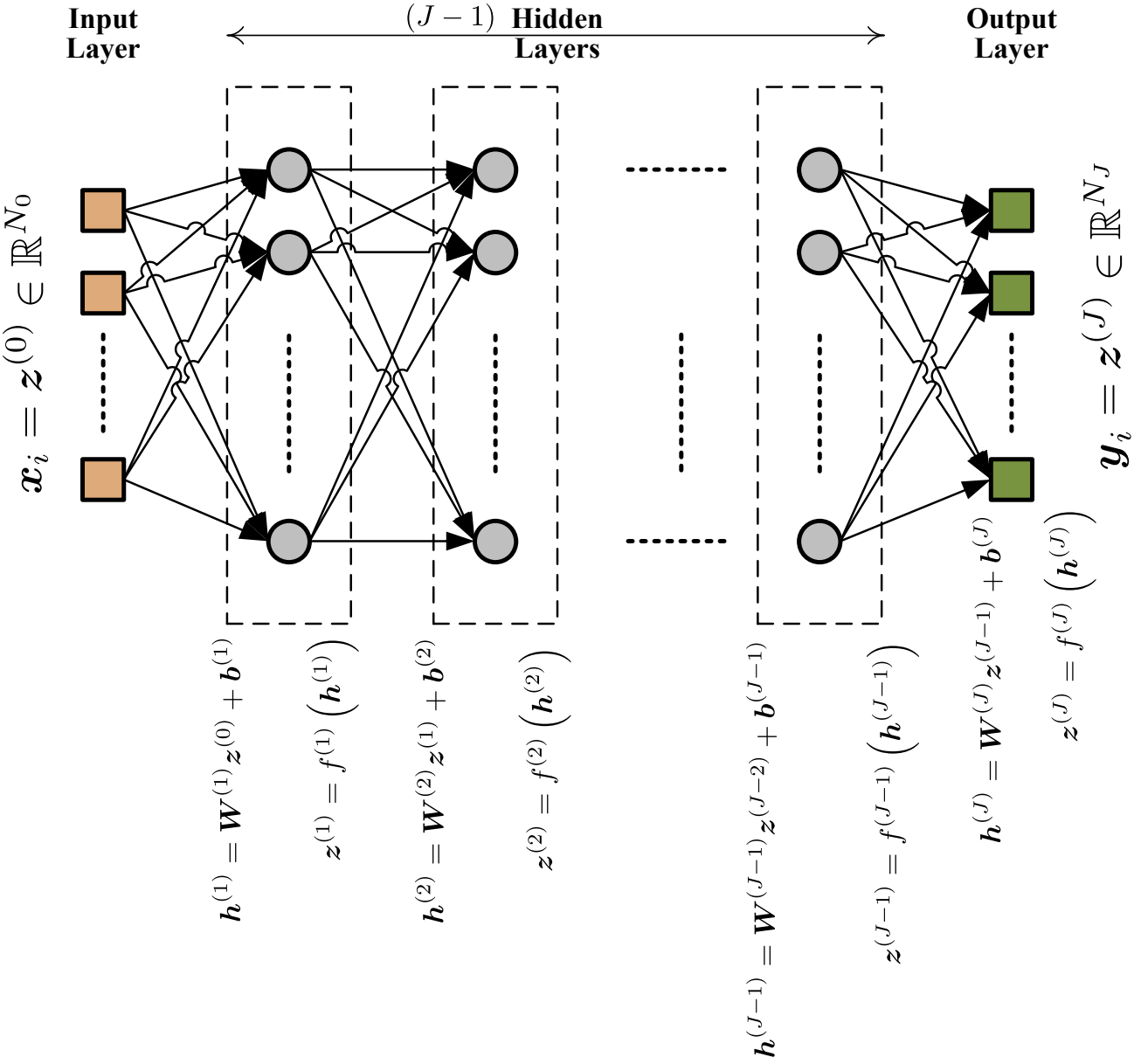


Figure 1: Block diagram of a deep fully-connected and feed-forward neural network with J layers, i.e,, $J-1$ hidden layers.

1.1 Gradients for the J layers DNN

In order to employ first-order (stochastic) optimization methods, e.g., stochastic gradient descent (SGD), we need the derivatives with respect to the weight matrices $\boldsymbol{W}^{(j)}$ and the biases $\boldsymbol{b}^{(j)}$.

The gradients at the considered j-th layer of aforementioned (fully-connected) feedforward DNN can be summarized as following with ℓ_2 regularization. The details of the gradient derivations for the $J = 3$ layers are given in Appendix A. Based on those derivations, we generalize it for arbitrary J layers (or equivalently $(J-1)$ hidden layers).

$$\frac{\partial L_i}{\partial \boldsymbol{W}^{(j)}} = \underbrace{\boldsymbol{G}^{(j)}\left(\boldsymbol{h}^{(j)}\right)}_{(j)\text{th layer}} \underbrace{\left[\boldsymbol{W}^{(j+1)}\right]^{\mathrm{T}} \boldsymbol{G}^{(j+1)}\left(\boldsymbol{h}^{(j+1)}\right)}_{(j+1)\text{th layer}} \cdots \underbrace{\left[\boldsymbol{W}^{(J)}\right]^{\mathrm{T}} \boldsymbol{G}^{(J)}\left(\boldsymbol{h}^{(J)}\right)}_{(J)\text{th layer}} \boldsymbol{e}_i \left[\boldsymbol{z}^{(j-1)}\right]^{\mathrm{T}} + \lambda \boldsymbol{W}^{(j)} \tag{3}$$

$$\equiv \left[\underbrace{g^{(j)}\left(\boldsymbol{h}^{(j)}\right)}_{(j)\text{th layer}} \odot \underbrace{\left[\boldsymbol{W}^{(j+1)}\right]^{\mathrm{T}} g^{(j+1)}\left(\boldsymbol{h}^{(j+1)}\right)}_{(j+1)\text{th layer}} \odot \cdots \odot \underbrace{\left[\boldsymbol{W}^{(J)}\right]^{\mathrm{T}} g^{(J)}\left(\boldsymbol{h}^{(J)}\right)}_{(J)\text{th layer}} \odot \boldsymbol{e}_i \right] \left[\boldsymbol{z}^{(j-1)}\right]^{\mathrm{T}} + \lambda \boldsymbol{W}^{(j)} \tag{4}$$

and

$$\frac{\partial L_i}{\partial \boldsymbol{b}^{(j)}} = \underbrace{\boldsymbol{G}^{(j)}\left(\boldsymbol{h}^{(j)}\right)}_{(j)\text{th layer}} \underbrace{\left[\boldsymbol{W}^{(j+1)}\right]^{\mathrm{T}} \boldsymbol{G}^{(j+1)}\left(\boldsymbol{h}^{(j+1)}\right)}_{(j+1)\text{th layer}} \cdots \underbrace{\left[\boldsymbol{W}^{(J)}\right]^{\mathrm{T}} \boldsymbol{G}^{(J)}\left(\boldsymbol{h}^{(J)}\right)}_{(J)\text{th layer}} \boldsymbol{e}_i + \lambda \boldsymbol{b}^{(j)} \tag{5}$$

$$\equiv \left[\underbrace{g^{(j)}\left(\boldsymbol{h}^{(j)}\right)}_{(j)\text{th layer}} \odot \underbrace{\left[\boldsymbol{W}^{(j+1)}\right]^{\mathrm{T}} g^{(j+1)}\left(\boldsymbol{h}^{(j+1)}\right)}_{(j+1)\text{th layer}} \odot \cdots \odot \underbrace{\left[\boldsymbol{W}^{(J)}\right]^{\mathrm{T}} g^{(J)}\left(\boldsymbol{h}^{(J)}\right)}_{(J)\text{th layer}} \odot \boldsymbol{e}_i \right] + \lambda \boldsymbol{b}^{(j)}\,, \tag{6}$$

where

- $\boldsymbol{e}_i = \boldsymbol{y}_i - \boldsymbol{t}_i \equiv \boldsymbol{z}^{(J)} - \boldsymbol{t}_i$ corresponds to an error vector, i.e., the difference between the output rendered by the considered DNN with the current weights and biases, and the actual output ,
- $\boldsymbol{z}^{(j)} = f^{(j)}\left(\boldsymbol{h}^{(j)}\right)$ denotes the output *after* the $f^{(j)}(\cdot)$ activation function at the jth layer,
- $\boldsymbol{h}^{(j)} = \boldsymbol{W}^{(j)}\boldsymbol{z}^{(j-1)} + \boldsymbol{b}^{(j)}$ is an (intermediate) output *before* the jth activation function and *after* the affine transformation of $\boldsymbol{z}^{(j-1)}$ by the weight matrix $\boldsymbol{W}^{(j)}$ and the bias $\boldsymbol{b}^{(j)}$,
- $g^{(j)}\left(\boldsymbol{h}^{(j)}\right) \coloneqq \frac{\partial}{\partial \boldsymbol{h}^{(j)}}\left[f^{(j)}\left(\boldsymbol{h}^{(j)}\right)\right] = f^{(j)\prime}\left(\boldsymbol{h}^{(j)}\right)$ corresponds to the first-order derivative of the activation function with respect to the input $\boldsymbol{h}^{(j)}$, e.g., derivative of the squared exponential or radial basis function with the σ^2 hyperparameter is $f^{(j)}(\boldsymbol{u}) = \exp\left(-\frac{\boldsymbol{u}\odot\boldsymbol{u}}{2\sigma^2}\right) \Rightarrow f^{(j)\prime}(\boldsymbol{u}) = \frac{-\boldsymbol{u}}{\sigma^2} \odot \exp\left(-\frac{\boldsymbol{u}\odot\boldsymbol{u}}{2\sigma^2}\right)$, and
- $\boldsymbol{G}^{(j)}\left(\boldsymbol{h}^{(j)}\right) \coloneqq \mathrm{Diag}\left(g^{(j)}\left(\boldsymbol{h}^{(j)}\right)\right)$ is a diagonal matrix which puts the respective vector elements of $\boldsymbol{h}^{(j)}$ along the main diagonal of $\boldsymbol{G}^{(j)}$.

1.2 MNIST Data, Performance Evaluation and Metrics

For the given MNIST dataset, the input image dimension is 28×28, which is vectorized such that the input feature vector reads $\boldsymbol{x}_i \in \mathbb{R}^{784}$. The dimension of the output is $\boldsymbol{y}_i, \boldsymbol{t}_i \in \mathbb{R}^{10}$.

The training dataset $\mathcal{D}^{\text{train}} \coloneqq \left\{\boldsymbol{x}_k^{\text{train}} \in \mathbb{R}^{784}, \widetilde{y}_k^{\text{train}} \in \mathbb{R}\right\}_{k=1}^{60000}$ comprises 60000/60K samples. We map the original training output $\widetilde{y}_k^{\text{train}} \in \mathbb{R} \mapsto \boldsymbol{y}_k^{\text{train}} \in \mathbb{R}^{10}$ corresponding to respective one of the 10 digits, without loss of generality, for the implementation purpose.

At every epoch, we utilize all the training data samples. Within an epoch cycle, we randomly select the non-overlapping sub dataset of samples, denoted as $\mathcal{B}$, according to the given mini-batch size. Thus, one iteration corresponds to the update of the DNN parameters, i.e., weights and biases, by utilization of the randomly chosen mini-batch from the training dataset. After the elapse of one epoch cycle, the iteration continues with randomly selected mini-batch dataset from the training dataset. The training performance is evaluated as an average of squared error loss, i.e., $(1/|\mathcal{B}|) \sum_{i \in [|\mathcal{B}|]} \|\boldsymbol{y} - \boldsymbol{t}\|_2^2$.

The test dataset $\mathcal{D}^{\text{test}} \coloneqq \{\boldsymbol{x}_k^{\text{test}} \in \mathbb{R}^{784}, y_k^{\text{test}} \in \mathbb{R}\}_{k=1}^{10000}$ consists of 10000/10K samples. We test the performance of the trained DNN by utilizing all the test dataset $\mathcal{D}^{\text{test}}$ every iterations or/and every elapse of the epoch cycle, which is referred to as the classification accuracy. The classification accuracy, denoted as ρ, corresponds to the ratio between the number of mismatches in predicted and true output and the total number of test dataset, i.e., $\rho \coloneqq \frac{1}{10000} \sum_{k=1}^{10000} \mathbb{I}\left(\widehat{y}_k, y_k^{\text{test}}\right)$ where the final predicted index is $\widehat{y}_k = \arg\max_{\ell \in \{1,\ldots,10\}} \{\boldsymbol{y}_k[1], \ldots, \boldsymbol{y}_k[\ell], \ldots, \boldsymbol{y}_k[10]\}$. The indicator function $\mathbb{I}\left(\widehat{y}_k, y_k^{\text{test}}\right)$ renders 0 if $\widehat{y}_k = y_k^{\text{test}}$ otherwise it renders 1.

2 Exercise (6a)

We have considered 5 layers DNN, i.e., 4 hidden layers, with 500 neurons each at every hidden layer. We have employed so-called scaled exponential linear units (SELU) [9] activation functions for the hidden layers while for the output layer we have employed softmax activation function, unless specified otherwise. Furthermore, we have not included the biases, i.e., $\{\boldsymbol{b}^{(j)}\}$, particularly in our 5 layers DNN, since they seem to not improve the performance and also slightly degrade the training and testing performance in particular for 5 layers DNN.

We would also like to highlight that we have tried other activation functions for 5 layers DNN, in particular sigmoid and squared exponential functions. However, the gradients of such activation functions in deep network was very small and thus the learning performance was quite poor.

We have employed SGD optimization algorithm to train the suitable weights of the 5 layers DNN, unless otherwise mentioned.

Figure 2 shows the training performance evaluated as an average of squared error loss versus number of iterations utilizing SGD with mini-batch size 1, the regularization parameter $\lambda = 0.001$, and fixed step size $\alpha = 0.0275$. Figure 3 shows the classification accuracy against iterations. We observe in Figure 3 that nearly after 105K iterations, the classification accuracy is around 96.24%. We observe that the classification accuracy has been very noisy in the first half of the iterations due to noisy gradients.

3 Exercise (6b)

Figure 4 shows the training performance of SGD with mini-batch size 100, the regularization parameter $\lambda = 0.001$, and fixed step size $\alpha = 0.0275$. Figure 5 shows the classification accuracy nearly 96.76% in about 1200 iterations. With the increase of the mini-batch size from 1 to 100, we can see relatively faster convergence rate of the training. Moreover, with the increase of the mini-batch from 1 to 100, we observe that the classification performance is less noisy compared to the case of mini-batch size 1 in Figure 3.

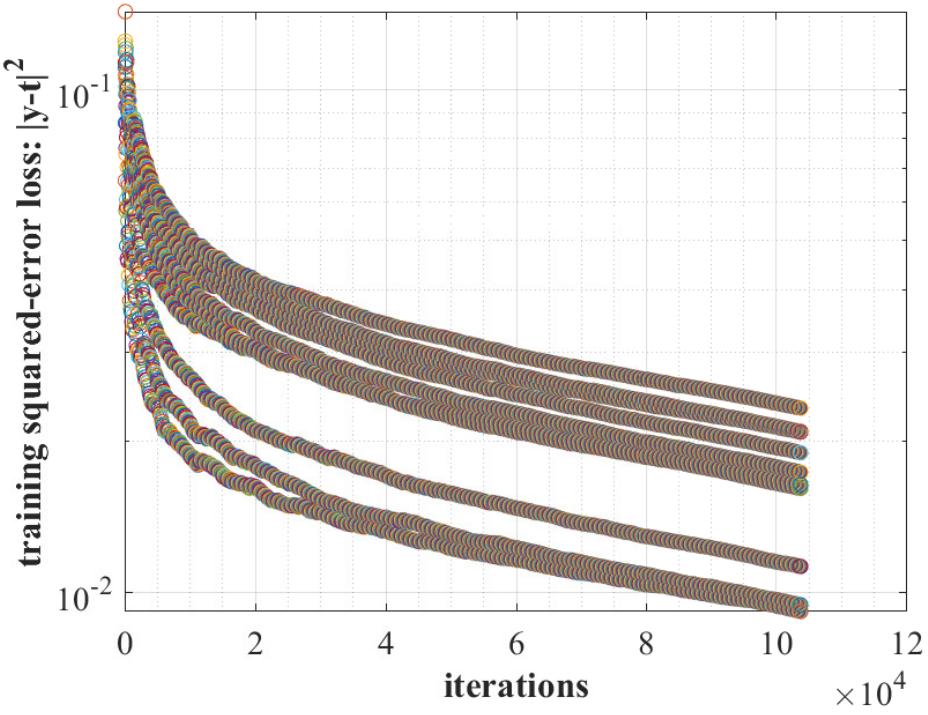


Figure 2: Squared-error loss versus iterations during the training of 5 layers DNN with (mini-batch size 1) SGD optimization. The 10 curves correspond to respective 10 outputs of MNIST data.

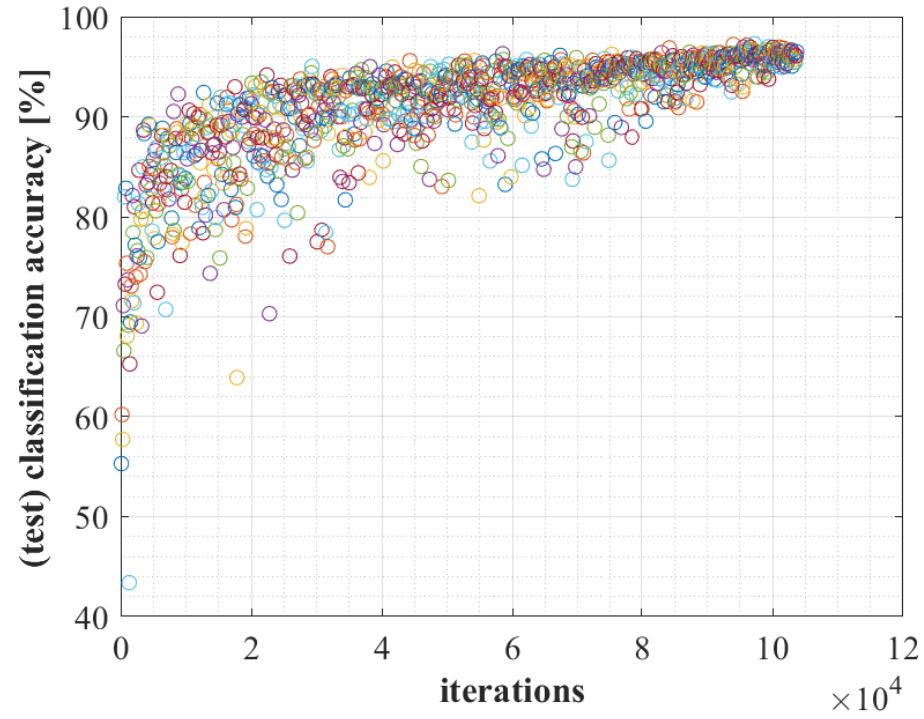


Figure 3: Classification accuracy versus iterations. At every iteration, the whole test dataset is evaluated to assess the classification accuracy with respect to the true output.

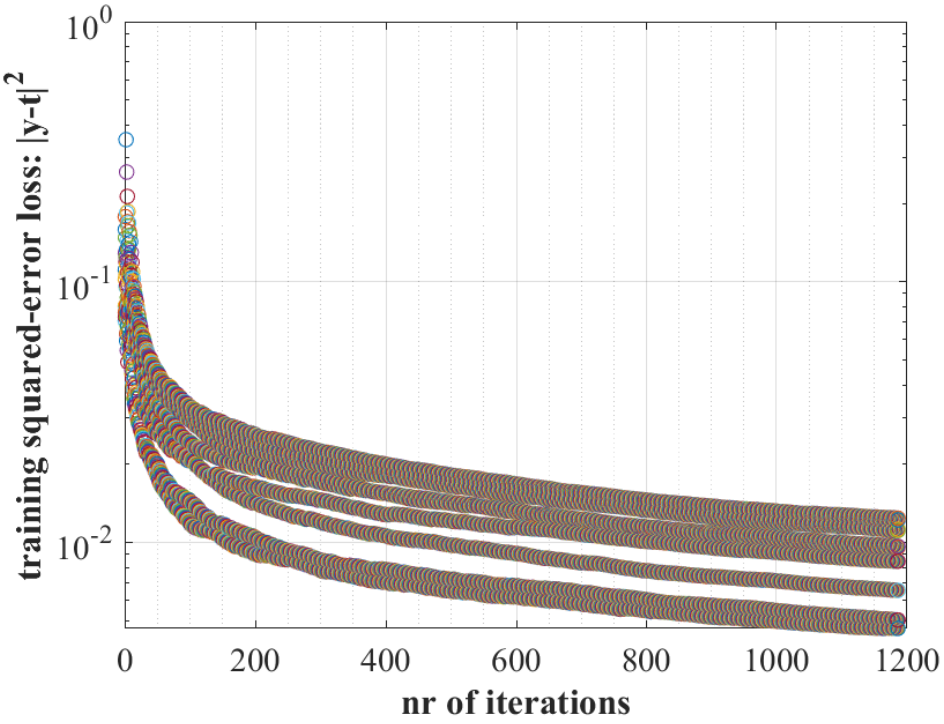


Figure 4: Squared-error loss versus iterations during the training of 5 layers DNN with (mini-batch size 100) SGD optimization with fixed step-size $\alpha = 0.0275$. The 10 curves correspond to respective 10 outputs of the MNIST dataset.

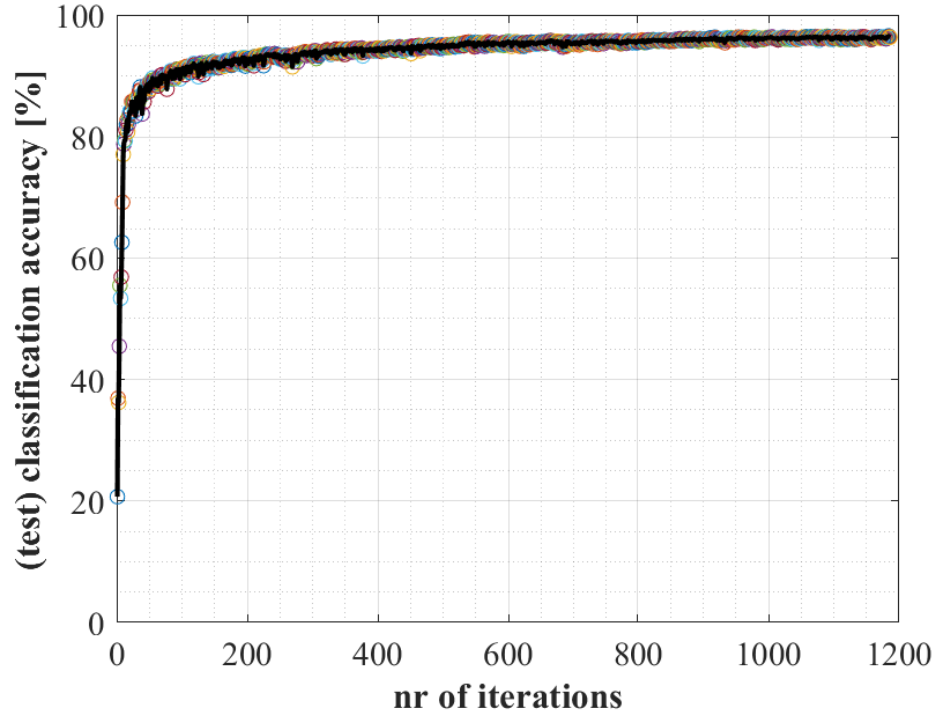


Figure 5: Classification accuracy versus iterations of (mini-batch 100) SGD with fixed step-size $\alpha = 0.0275$. At every iteration, the whole test dataset is evaluated to assess the classification accuracy with respect to the true output.

We have also evaluated the performance of three adaptive learning rate algorithms, namely diminishing, AdaGrad and RMSProp. The AdaGrad and RMSProp schemes are outlined in Algorithm 1 and Algorithm 2, respectively.For the diminishing step-size, we have employed $\alpha = \frac{\alpha}{1+\beta\alpha\ell}$

Figure 6 shows the training convergence rates of three considered adaptive learning rate methods for 5 layers DNN with the considered hyperparameters given in Table 1. Figure 7 shows the classification performance on the test dataset for these adaptive methods. Furthermore, Figure 8 compares the classification performance against iterations of these considered adaptive schemes with the fixed step-size. Evidently for the chosen hyperparameters, we observe that the performance rendered by RMSProp is quite noisy relative to AdaGrad. For some iterations, AdaGrad method in general seems to offer slightly better performance than the other considered adaptive schemes with some increased complexity. AdaGrad can achieve nearly 96.76% classification accuracy on the test dataset in about 1200 training iterations or equivalently 2 epochs for the considered mini-batch size of 100 samples. RMSProp is noisy for the considered hyperparameters but in some occasions the accuracy is similar to diminishing step-size method. Nevertheless, we have observed that early-stopping of the fixed step-size SGD renders relatively good classification performance. There might be room to improve the classification accuracy further, but changing the number of neurons per hidden layer appropriately and tuning the hyperparameters. Moreover, instead of feedforward DNN, one could also employ convolutional DNN to possibly further enhance the accuracy performance.

Algorithm 1: AdaGrad for an ℓ-th iteration

Initialize Weights/biases for a given layer denoted as $\boldsymbol{X}_{(0)}$, for brevity

Inputs : Weight/bias update at previous $(\ell-1)$-th iteration $\boldsymbol{X}_{(\ell-1)}$,
All the gradients w.r.t. weight/bias for a given layer until ℓ-th iteration $\left\{\frac{\partial L_k}{\partial \boldsymbol{X}}\right\}_{k=1}^{\ell}$,
Step-size $\alpha \in \mathbb{R}$, $\xi \in \mathbb{R}$

$\boldsymbol{g}_{(k)} \coloneqq \frac{\partial L_k}{\partial \boldsymbol{X}} \quad \forall k = 1, \ldots, \ell$

$\boldsymbol{v}_{(\ell)} = \frac{1}{\ell}\sum_{k=1}^{\ell} \left(\boldsymbol{g}_{(k)} \odot \boldsymbol{g}_{(k)}\right)$

$\boldsymbol{V}_{(\ell)}^{-\frac{1}{2}} = \operatorname{diag}\left(\mathbf{1} \oslash \left(\sqrt{\boldsymbol{v}_{(\ell)}} + \xi\mathbf{1}\right)\right)$

$\boldsymbol{X}_{(\ell)} = \boldsymbol{X}_{(\ell-1)} - \alpha \boldsymbol{V}_{(\ell)}^{-\frac{1}{2}} \boldsymbol{g}_{(\ell)}$

Output: Weight/bias update $\boldsymbol{X}_{(\ell)}$

Algorithm 2: RMSProp for an ℓ-th iteration

Initialize Weights/biases for a given layer denoted as $\boldsymbol{X}_{(0)}$, for brevity

Inputs : Previous $(\ell-1)$-th iteration updates, namely $\boldsymbol{X}_{(\ell-1)}$ and $\boldsymbol{v}_{(\ell-1)}$,
Gradient w.r.t. weight/bias for a given layer at ℓ-th iteration $\frac{\partial L_\ell}{\partial \boldsymbol{X}}$,
Step-size $\alpha \in \mathbb{R}$, $\xi \in \mathbb{R}$, and $\beta \in [0, 1)$

$\boldsymbol{g}_{(\ell)} \coloneqq \frac{\partial L_\ell}{\partial \boldsymbol{X}}$

$\boldsymbol{v}_{(\ell)} = \beta \boldsymbol{v}_{(\ell-1)} + (1-\beta)\left(\boldsymbol{g}_{(k)} \odot \boldsymbol{g}_{(k)}\right)$

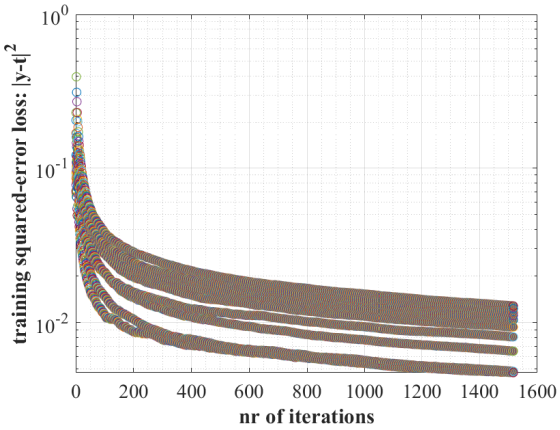
$\boldsymbol{V}_{(\ell)}^{-\frac{1}{2}} = \operatorname{diag}\left(\mathbf{1} \oslash \left(\sqrt{\boldsymbol{v}_{(\ell)}} + \xi\mathbf{1}\right)\right)$

$\boldsymbol{X}_{(\ell)} = \boldsymbol{X}_{(\ell-1)} - \alpha \boldsymbol{V}_{(\ell)}^{-\frac{1}{2}} \boldsymbol{g}_{(\ell)}$

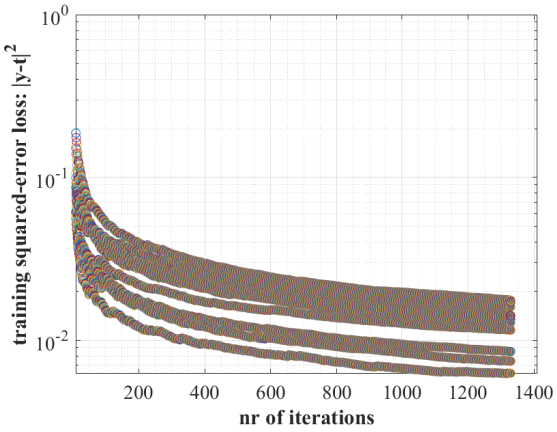
Output: Weight/bias update $\boldsymbol{X}_{(\ell)}$

4 Exercise (6c)

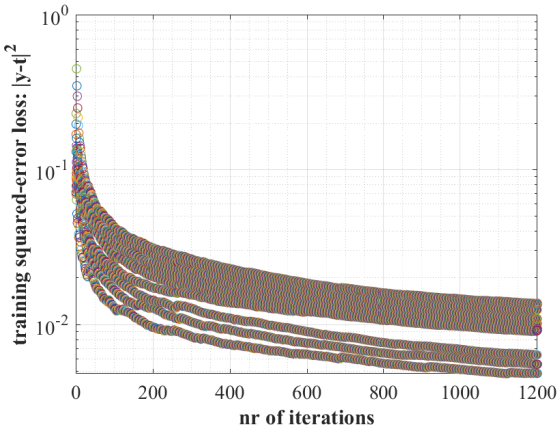
We have also investigated the performance of a relatively shallow neural network, 2 layers DNN, i.e., a single hidden layer. For the sake of comparison between 2 layers and 5 layers DNN, we have considered



(a) Diminishing

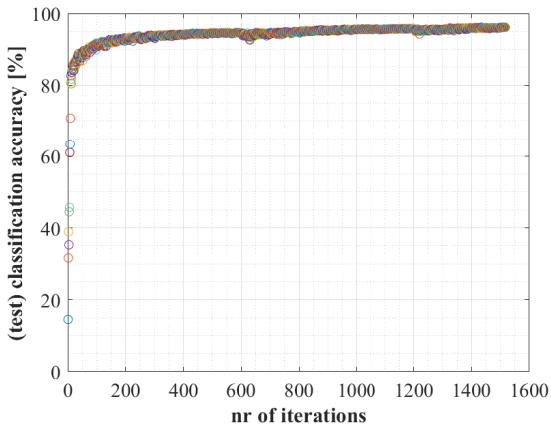


(b) RMSProp

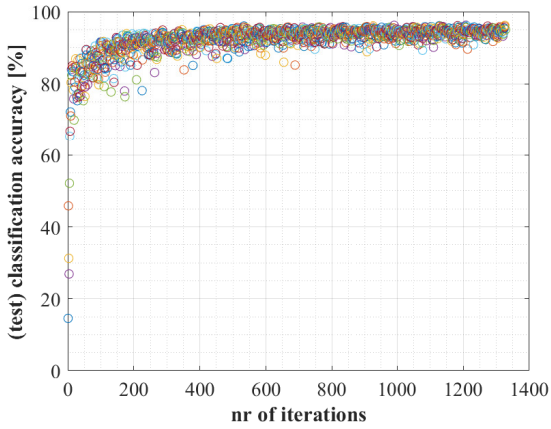


(c) AdaGrad

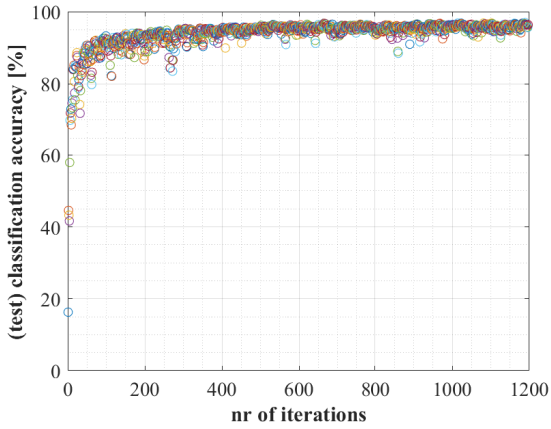
Figure 6: Squared error loss vs. iterations



(a) Diminishing



(b) RMSProp



(c) AdaGrad

Figure 7: Classification accuracy vs. iterations

Table 1: Some Hyperparameters

Method	**Lambda λ**	**(Initial) step-size α**	**Other hyperparameters**	**Mini-batch-size**
SGD	0.001	fixed 0.0275	–	100
SGD	0.001	diminishing but initial 0.0275	$\beta = 0.001$	100
RMSProp	0.001	initial 0.0009	$\beta = 0.9, \xi = 10^{-12}$	100
AdaGrad	0.001	initial 0.0009	$\xi = 10^{-12}$	100

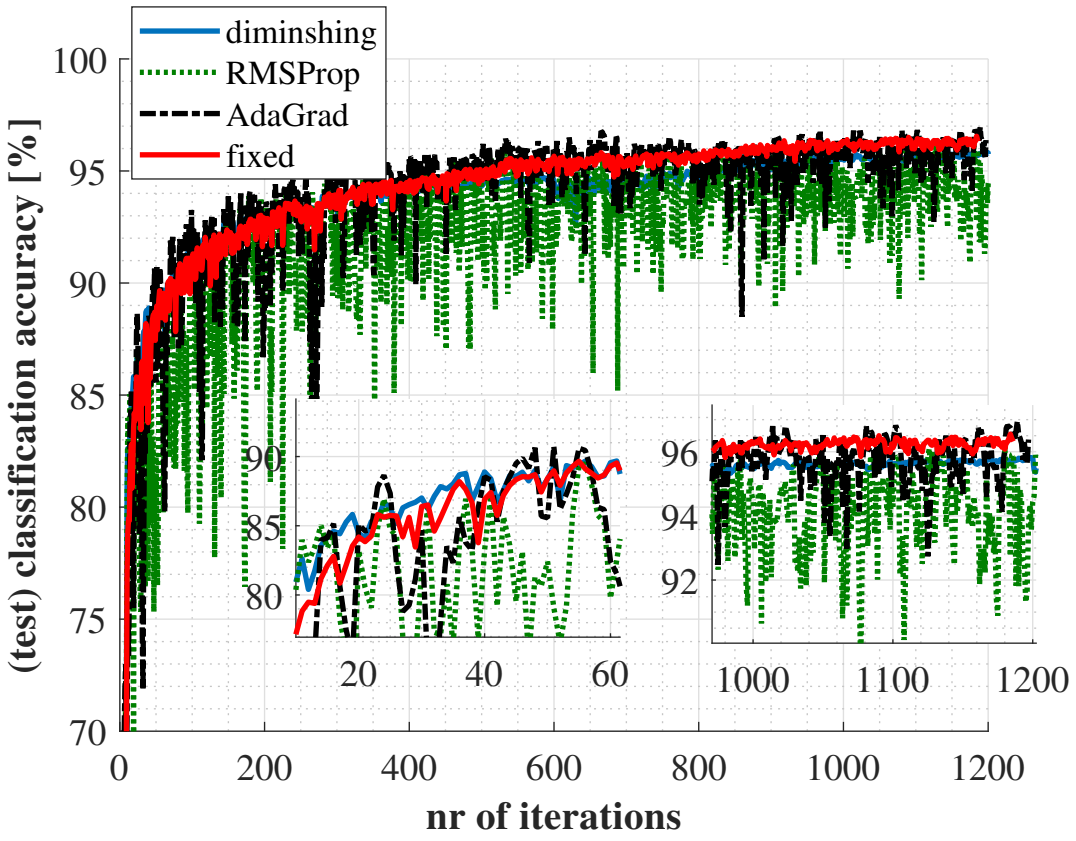


Figure 8: Classification performance comparison between fixed step-size and adaptive rates, namely diminshing, RMSProp, and AdaGrad.

2000 neurons in a single hidden layer (or 2 layers DNN) since in 5 layers DNN we have 500 neurons in each hidden layer, i.e., total 2000 neurons in all 4 hidden layers.

In case of 2 layers DNN, we have considered two different set-up with different activation functions in the hidden layer and output layer, referred as 2 layers DNN type-a and type-b.

1. Type-a (2 layers DNN): we have same activation function set-up as in 5 layers DNN, i.e., SELU for the hidden layer and softmax activation functions for the output layer.

2. Type-b (2 layers DNN): squared exponential (aka radial basis function) activation function for the hidden layer and no activation function (or identity function) for the output layer. For this exercise, the length scale $\sigma^2 = 0.05$ is selected for the squared exponential function. Moreover, we have trained bias for the type-b set-up as well.

We have employed SGD with fixed step-size for training both 2 and 5 layers DNN.

Figure 9 compares the classification performance of 5 layers DNN and both types of 2 layers DNN. We can construe from the figure that type-b (2 layers DNN) has relatively better performance than type-a (2 layers DNN) after 400 iterations. Moreover, the performance of type-b nearly approaches the performance of 5 layers DNN, but still not as good as the deep architecture. Thus, at least for this exercise,

we can construe that the deep architecture seems to be better than the shallow one if the appropriate activation functions are chosen.

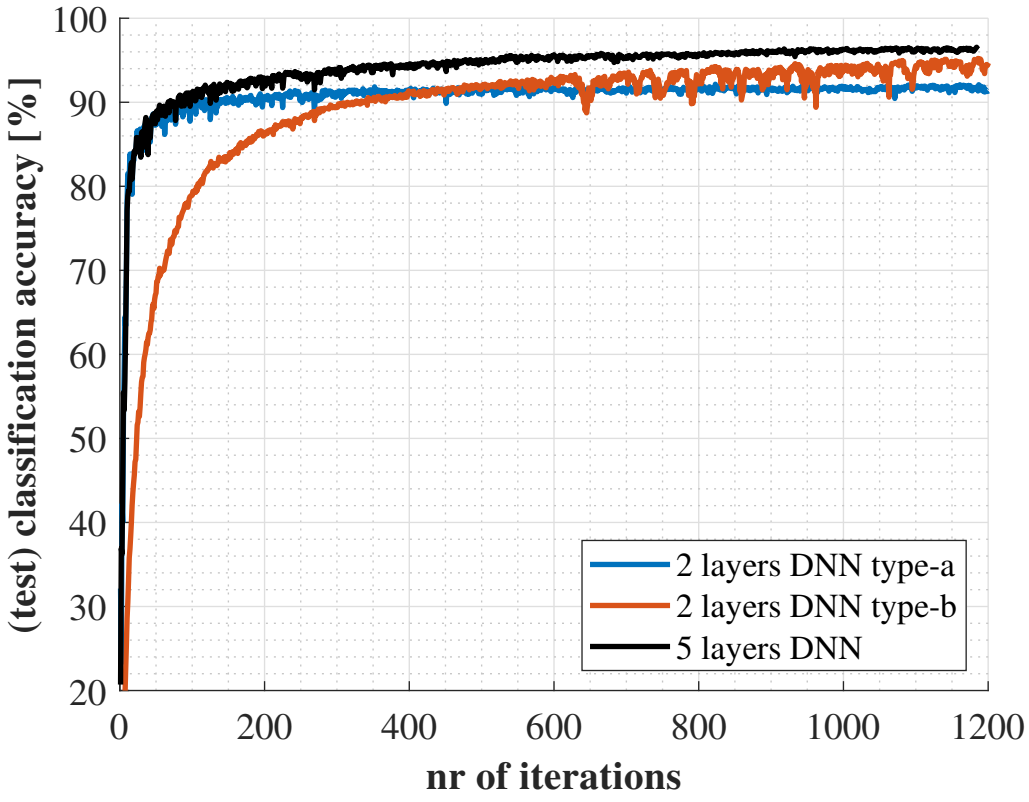


Figure 9: Comparison of classification accuracy among 2 layers DNN types, i.e., type-a and type-b, and 5 layers DNN.

5 Exercise (6d)

Decentralized DNN with 6 workers

For master-slave mode, the original MNIST dataset is divided into 6 parts. The topology of this mode is shown in Figure 10.

Algorithm 3: Decentralized DNN

Initialize W_1, b_1;
for *i=1,2,...* **do**
  Master node broadcast w_i, b_i;
  All workers train their own data set and get the optimized w_i^{id} and b_i^{id};
  Master node collects W_i^{id}, b_i^{id} and computes $W_{i+1} = \frac{1}{N}\sum_{id\in[N]} W_i^{id}$, $b_{i+1} = \frac{1}{N}\sum_{id\in[N]} b_i^{id}$
end

Experimental result

We have considered 3 layers DNN, i.e., 2 hidden layers with 1000 neurons each at every hidden layer. We have employed SELU activation functions for the hidden layers while for the output layer we have employed softmax activation function. Furthermore, we have not included the biases.

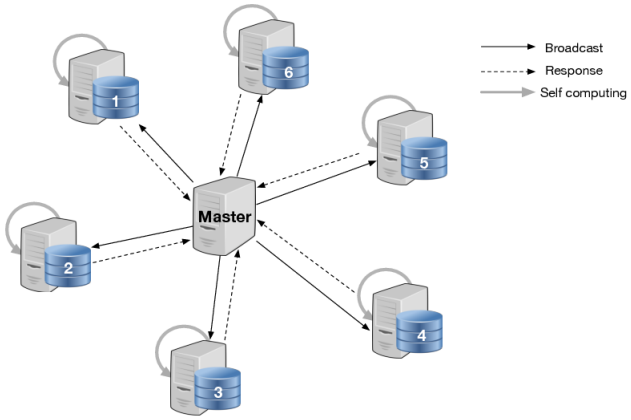


Figure 10: Master-slave structure.

We have employed SGD optimization algorithm to train the suitable weights of the 3 layers DNN, unless otherwise mentioned.

Similar as 6(a), we have Figure 11 to show the classification accuracy for each worker against iterations. Since we simulate the distribution using serial execution in Matlab, we got six curves one after another. Figure 11 only shows two epochs (where each worker finish training their dataset) and we see the trend that the accuracy is increased for all of them after the first round communication.

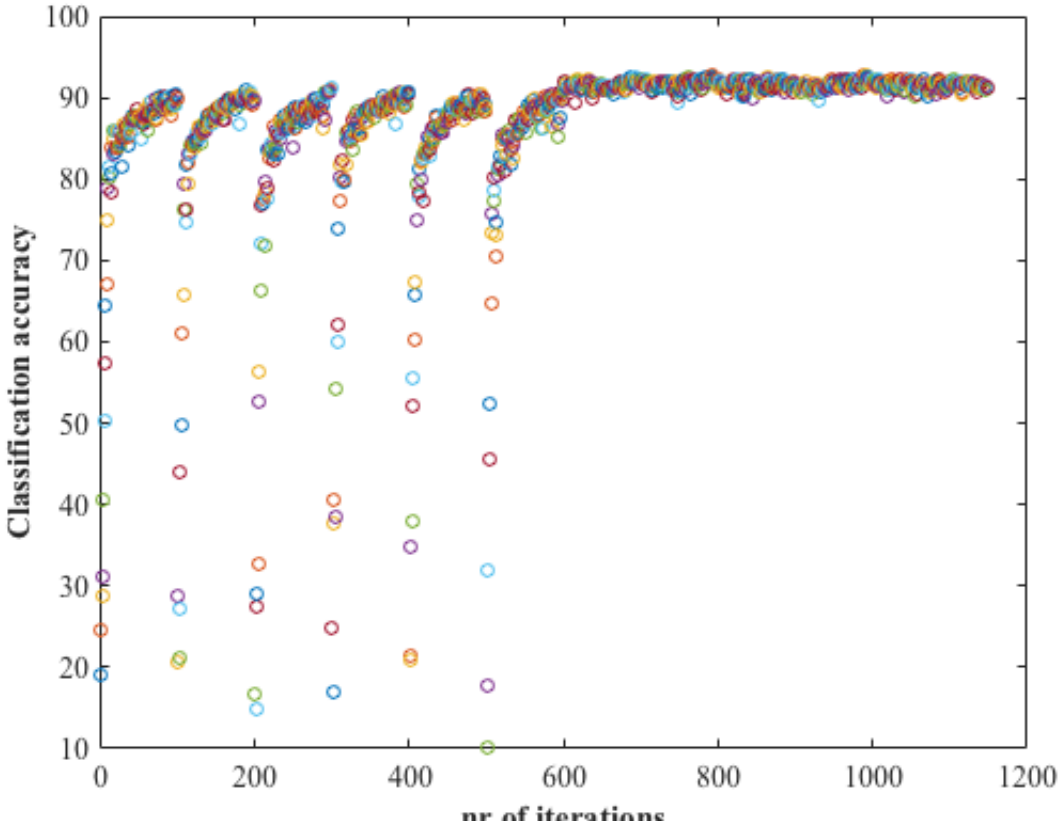


Figure 11: Intermediate classification accuracy output versus iterations. Every worker works in serial order and output one after another in the same figure.

To accelerate the training process, we omitted the intermediate output such as Figure 11. We trained the network for nearly 105k iterations and the figure 12 shows classification accuracy against epochs. We

observe that the classification accuracy is around 95.40% for a 3-layer DNN in master-slave structure. The result is closed to a centralized result.

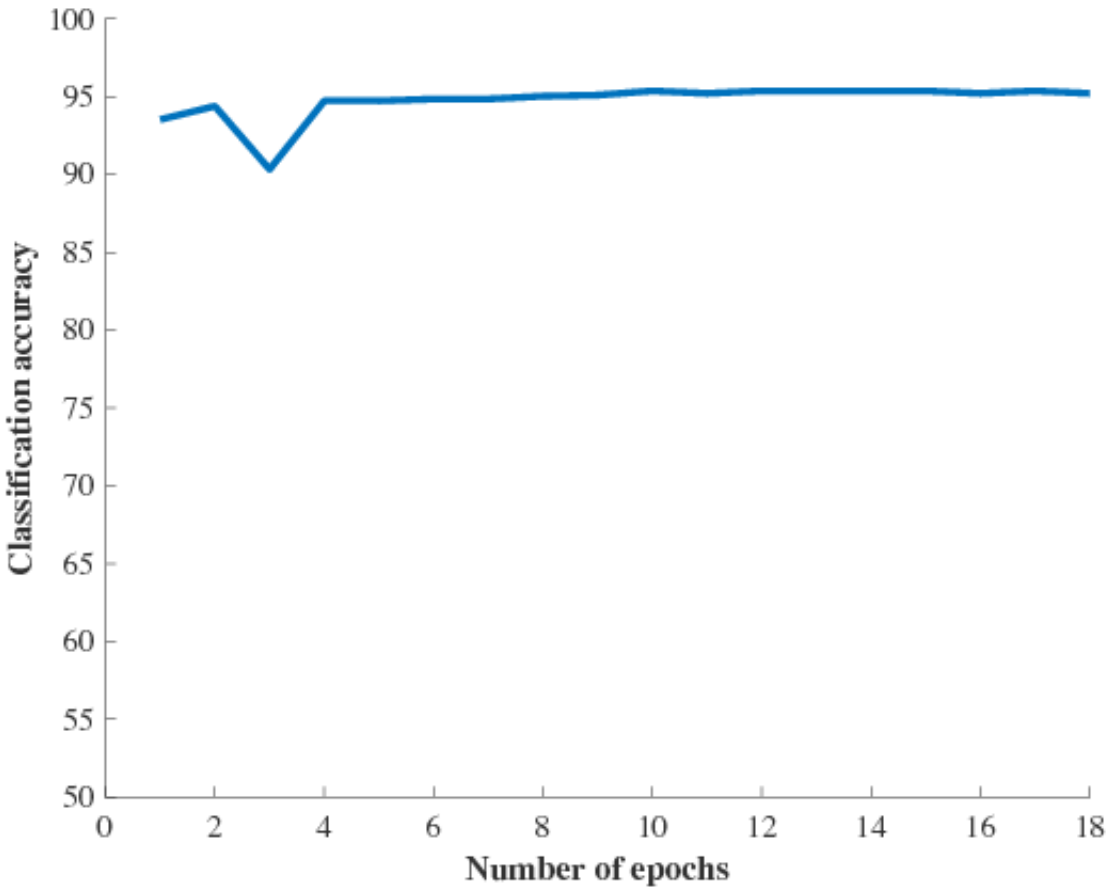


Figure 12: Average classification accuracy versus epochs. At every epoch, the whole test dataset is evaluated to assess the classification accuracy with respect to the true output.

6 Exercise (6e)

Decentralized DNN with a two-star topology.

For two-star topology, it has a similar process as the master-slave mode. The topology is shown in Figure 13. Node 3 and 4 contains data set and in the meantime collects results from node 1, 2 and node 5, 6 respectively to reach consensus.

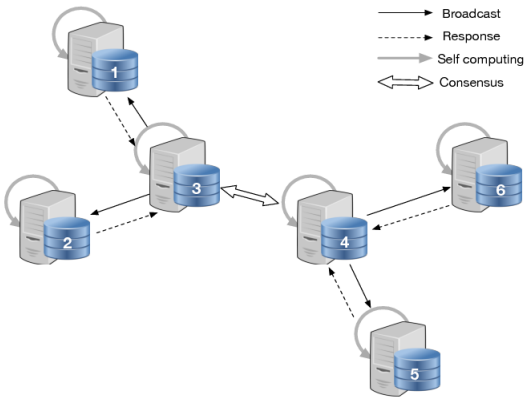


Figure 13: Two-star topology.

Algorithm 4: Two-star topology DNN

Initialize W_1, b_1;
for *i=1,2,...* **do**
  Communication nodes broadcast w_i, b_i;
  All workers train their own data set and get the optimized w_i^{id} and b_i^{id}
  Solve in parallel

$$\bar{W_i^3} = \frac{1}{N1}\sum_{id\in[N1]} W_i^{id}, \bar{b_i^3} = \frac{1}{N1}\sum_{id\in[N1]} b_i^{id}, \text{N1=[1,2,3];}$$

$$\bar{W_i^4} = \frac{1}{N2}\sum_{id\in[N2]} W_i^{id}, \bar{b_i^4} = \frac{1}{N2}\sum_{id\in[N2]} b_i^{id}, \text{N2=[4,5,6];}$$

  Communication nodes agree on $W_{i+1} = \frac{1}{2}\sum(\bar{W_i^3} + \bar{W_i^4})$, $b_{i+1} = \frac{1}{2}\sum(\bar{b_i^3} + \bar{b_i^4})$
end

There is no difference when we simulate each node's behavior in Matlab in serial order between master-slave topology and two-star topology, and the outcome is the average weight matrix and average bias. Thus, the simulation result is similar to the (6d). Compared with master-slave topology, one can avoid single node failure in two-star topology but needs to consider about the communication between star nodes for reaching consensus.

7 Exercise (6f)

In this exercise, we have employed dropout technique for the training of 5 layers DNN. For the training of the weights, we have utilized SGD with fixed step-size $\alpha = 0.0275$.

For the given droupout probability, say p, we randomly select the indices of the output vector of the given hidden layer $\boldsymbol{z}^{(j)}$ according to p probability, which are replaced with 0. The non-zero elements of the output vector are scaled by $\frac{1}{1-p}$ to compensate for the loss of dropped element indices.

Figure 14 compares the classification accuracy on the test data of the 5 layers DNN with and without dropout. Since we have employed $L2$ regularization for the training of the DNN, the additional dropout technique on top of the $L2$ regularization seems to improve the classification performance marginally.

8 Exercise (6g)

An alternative approach to improve the smoothness is to use $L1$ batch normalization.

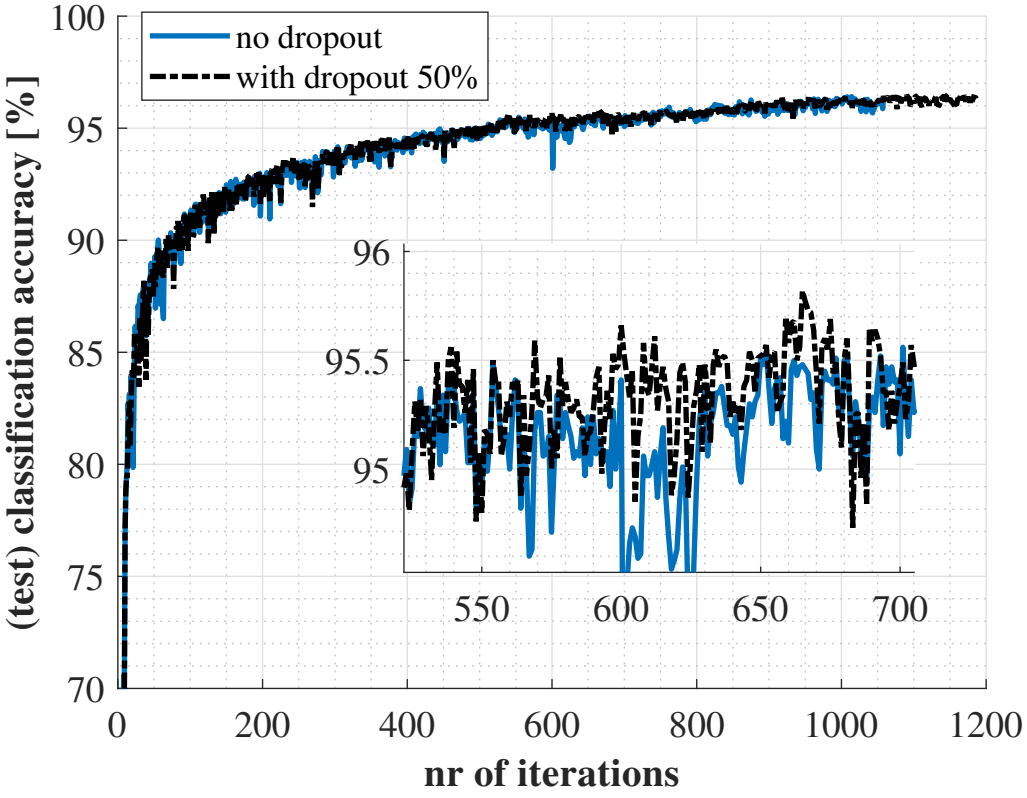


Figure 14: Comparison of classification accuracy between dropout and no dropout in 5 layers DNN.

Algorithm 5: L1BN Algorithm with statistics $\{\mu, \sigma\}$ [11]

Require: a mini-batch of pre-activation samples for training $\mathcal{B} = \{x_1, \cdots x_m\}$, learning rate eta, moving-average momentum α;

Ensure: updated parameters $\{\mu, \sigma, \gamma, \beta\}$ normalized pre-activation $\mathcal{B}_{tr}^N = \{x_1^N, \cdots x_m^N\}$;

Forward:

$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$

$\sigma_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} |x_i - \mu_{\mathcal{B}}|$

$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}} + \epsilon}$

$\hat{x}_i^N \leftarrow \gamma \hat{x}_i + \beta \equiv \text{L1BN}_{tr}(x_i)$

Backward:

$\gamma \leftarrow \gamma - \eta \frac{\partial f}{\partial \gamma}$

$\beta \leftarrow \beta - \eta \frac{\partial f}{\partial \beta}$

Update statistics:

$\mu \leftarrow \alpha\mu + (1-\alpha)\mu_{\mathcal{B}}$

$\sigma \leftarrow \alpha\sigma + (1-\alpha)\sigma\mathcal{B}$

From the simulation tests for deep linear networks in [10] as shown in Figure 15, L1 normalization can imporove the smoothness and achive good accuracy.

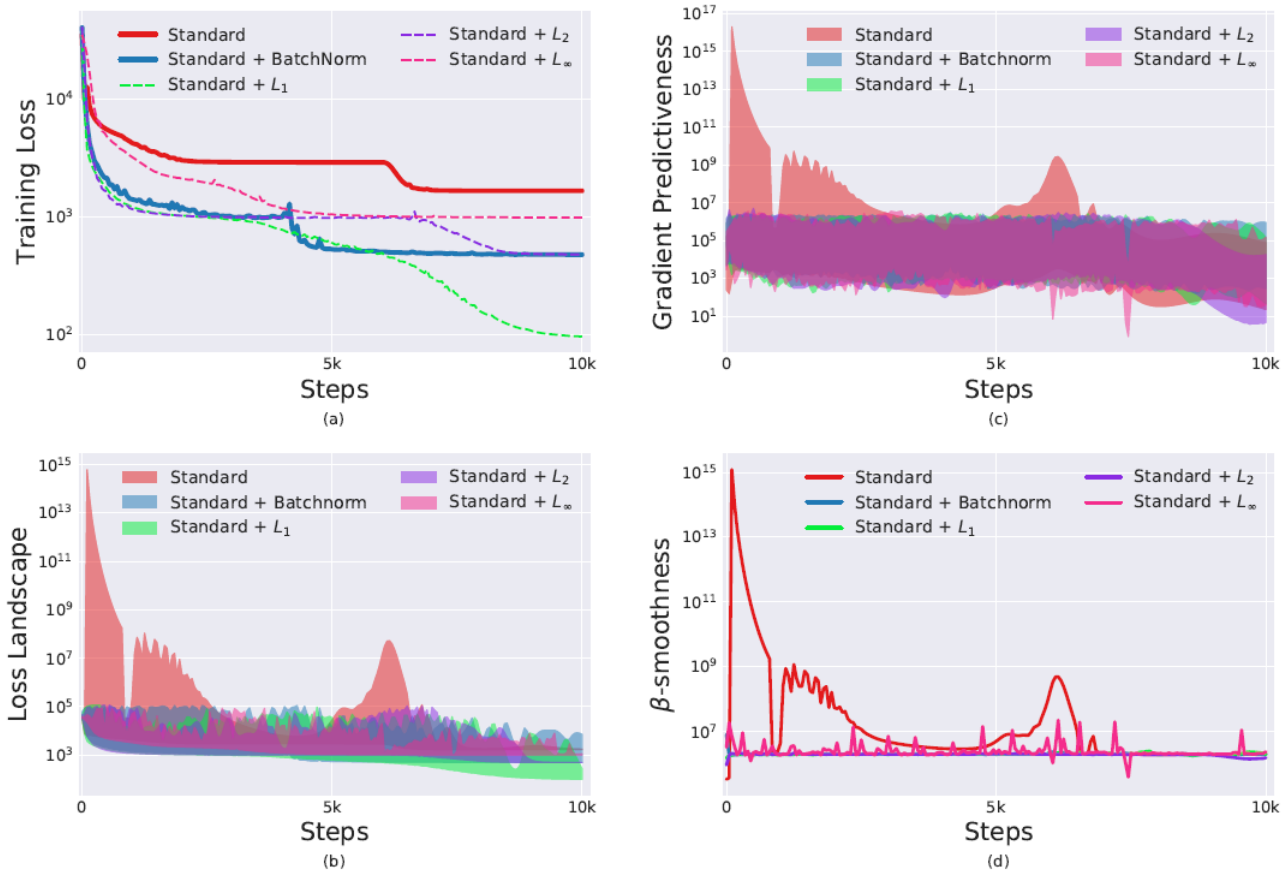


Figure 15: Evaluation of deep linear networks trained with different l_p normalization strategies. We observe that networks with any normalization strategy have improved performance and smoothness of the loss landscape over standard training [10]

9 README: How to run the MATLAB script

The main script to run all the above test scenarios is `ca6__run_me.m`. Select the appropriate test case number defined by the input `select_test_case_nr` in the main script to run the following test scenarios (with the default hyperparameters).

1. The test case number `select_test_case_nr = 1` corresponds to running the (centralized) 5 layers DNN with the given `mini_batch_size` in the main script. This test case scenario runs all the three optimization algorithms, namely, SGD, AdaGrad, and RMSProp. By default, the `mini_batch_size` is `100`. This test case can regenerate the results depicted in Exercise (6b) and also for Exercise (6a) by changing the `mini_batch_size`.

2. The test case number `select_test_case_nr = 2` corresponds to running the (centralized) 2 layers DNN with SELU activation function for the hidden layer and softmax for the output layer. This test case can regenerate the results depicted in Exercise (6c).

3. The test case number `select_test_case_nr = 3` corresponds to running the (centralized) 2 layers DNN with RBF activation function for the hidden layer and identity/no nonlinear function for the output layer. This test case can regenerate the results depicted in Exercise (6c).

4. The test case number `select_test_case_nr = 4` corresponds to running the (decentralized) 3 layers DNN with SELU activation function for the hidden layer and softmax for the output layer. This test case can regenerate the results depicted in Exercise (6d) and Exercise (6e).

5. The test case number `select_test_case_nr = 5` corresponds to running the (centralized) 5 layers DNN with dropout of 50% probability. This test case can regenerate the results depicted in Exercise (6f).

One could simply run the main script with the default set of parameters and select the appropriate test case numbers as described above.

Appendix A Gradients for the 3 layers DNN

For brevity, we assume 2 hidden layers, i.e., $J = 3$ layers, for the gradients computation with ℓ_2 regularization such that the DNN output vector $\boldsymbol{y}_i$ of an i-th input vector $\boldsymbol{x}_i$ reads

$$\boldsymbol{y}_i \coloneqq f^{(3)}\left(\boldsymbol{W}^{(3)} f^{(2)}\left(\boldsymbol{W}^{(2)} f^{(1)}\left(\boldsymbol{W}^{(1)}\boldsymbol{x}_i + \boldsymbol{b}_1\right) + \boldsymbol{b}^{(2)}\right) + \boldsymbol{b}^{(3)}\right) . \tag{7}$$

The regularized norm-2 cost function for the 2 hidden layers in (7) can be described as

$$\begin{aligned} & L\left(\left\{\boldsymbol{W}^{(j)}, \boldsymbol{b}^{(j)}\right\}_{j=1}^{3}\right) \\ & \quad \coloneqq \frac{1}{2N}\sum_{i=1}^{N}\left\{\left\|\underbrace{f^{(3)}\left(\boldsymbol{W}^{(3)} f^{(2)}\left(\boldsymbol{W}^{(2)} f^{(1)}\left(\boldsymbol{W}^{(1)}\boldsymbol{x}_i + \boldsymbol{b}_1\right) + \boldsymbol{b}^{(2)}\right) + \boldsymbol{b}^{(3)}\right)}_{\coloneqq \boldsymbol{y}_i} - \boldsymbol{t}_i\right\|_2^2\right. \\ & \quad \left. + \lambda\sum_{j=1}^{3}\left(\left\|\boldsymbol{W}^{(j)}\right\|_2^2 + \left\|\boldsymbol{b}^{(j)}\right\|_2^2\right)\right\} \end{aligned} \tag{8}$$

Let us define some new intermediate vectors with their corresponding differentials,

$$\boldsymbol{y}_i = \boldsymbol{z}^{(3)} = f^{(3)}\left(\boldsymbol{h}^{(3)}\right) \tag{9}$$

$$\Rightarrow d\boldsymbol{z}^{(3)} = \underbrace{f^{(3)\prime}\left(\boldsymbol{h}^{(3)}\right)}_{:=g^{(3)}(\cdot)} \odot d\boldsymbol{h}^{(3)} := g^{(3)}\left(\boldsymbol{h}^{(3)}\right) \odot \, d\boldsymbol{h}^{(3)} \equiv \underbrace{\text{Diag}\left(g^{(3)}\left(\boldsymbol{h}^{(3)}\right)\right)}_{:=\boldsymbol{G}^{(3)}} d\boldsymbol{h}^{(3)} = \boldsymbol{G}^{(3)} d\boldsymbol{h}^{(3)} \tag{10}$$

$$\boldsymbol{h}^{(3)} = \boldsymbol{W}^{(3)}\boldsymbol{z}^{(2)} + \boldsymbol{b}^{(3)} \tag{11}$$

$$\Rightarrow d\boldsymbol{h}^{(3)} = d\boldsymbol{W}^{(3)}\, \boldsymbol{z}^{(2)} + \boldsymbol{W}^{(3)} d\boldsymbol{z}^{(2)} + d\boldsymbol{b}^{(3)} \tag{12}$$

$$\boldsymbol{z}^{(2)} = f^{(2)}\left(\boldsymbol{h}^{(2)}\right) \tag{13}$$

$$\Rightarrow d\boldsymbol{z}^{(2)} = \underbrace{f^{(2)\prime}\left(\boldsymbol{h}^{(2)}\right)}_{:=g^{(2)}(\cdot)} \odot d\boldsymbol{h}^{(2)} := g^{(2)}\left(\boldsymbol{h}^{(2)}\right) \odot \, d\boldsymbol{h}^{(2)} \equiv \underbrace{\text{Diag}\left(g^{(2)}\left(\boldsymbol{h}^{(2)}\right)\right)}_{:=\boldsymbol{G}^{(2)}} d\boldsymbol{h}^{(2)} = \boldsymbol{G}^{(2)} d\boldsymbol{h}^{(2)} \tag{14}$$

$$\boldsymbol{h}^{(2)} = \boldsymbol{W}^{(2)}\boldsymbol{z}^{(1)} + \boldsymbol{b}^{(2)} \tag{15}$$

$$\Rightarrow d\boldsymbol{h}^{(2)} = d\boldsymbol{W}^{(2)}\, \boldsymbol{z}^{(1)} + \boldsymbol{W}^{(2)} d\boldsymbol{z}^{(1)} + d\boldsymbol{b}^{(2)} \tag{16}$$

$$\boldsymbol{z}^{(1)} = f^{(1)}\left(\boldsymbol{h}^{(1)}\right) \tag{17}$$

$$\Rightarrow d\boldsymbol{z}^{(1)} = \underbrace{f^{(1)\prime}\left(\boldsymbol{h}^{(1)}\right)}_{:=g^{(1)}(\cdot)} \odot d\boldsymbol{h}^{(1)} := g^{(1)}\left(\boldsymbol{h}^{(1)}\right) \odot \, d\boldsymbol{h}^{(1)} \equiv \underbrace{\text{Diag}\left(g^{(1)}\left(\boldsymbol{h}^{(1)}\right)\right)}_{:=\boldsymbol{G}^{(1)}} d\boldsymbol{h}^{(1)} = \boldsymbol{G}^{(1)} d\boldsymbol{h}^{(1)} \tag{18}$$

$$\boldsymbol{h}^{(1)} = \boldsymbol{W}^{(1)}\boldsymbol{x}_i + \boldsymbol{b}^{(1)} \equiv \boldsymbol{W}^{(1)} \underbrace{\boldsymbol{z}^{(0)}}_{:=\boldsymbol{x}_i} + \boldsymbol{b}^{(1)} \tag{19}$$

$$\Rightarrow d\boldsymbol{h}^{(1)} = d\boldsymbol{W}^{(1)}\, \boldsymbol{x}_i + d\boldsymbol{b}^{(1)} \, . \tag{20}$$

We now describe the loss function (8) in a succinct form, i.e.,

$$L = \frac{1}{N}\sum_{i=1}^{N}\left\{\frac{1}{2}\left\{\|\boldsymbol{y}_i - \boldsymbol{t}_i\|_2^2 + \lambda \sum_{j=1}^{3}\left(\left\|\boldsymbol{W}^{(j)}\right\|_2^2 + \left\|\boldsymbol{b}^{(j)}\right\|_2^2\right)\right\}\right\} \tag{21}$$

$$= \frac{1}{N}\sum_{i=1}^{N} L_i \, , \tag{22}$$

where

$$L_i \coloneqq \frac{1}{2}\left\|\boldsymbol{y}_i - \boldsymbol{t}_i\right\|_2^2 + \frac{\lambda}{2}\sum_{j=1}^{3}\left(\left\|\boldsymbol{W}^{(j)}\right\|_2^2 + \left\|\boldsymbol{b}^{(j)}\right\|_2^2\right) \tag{23}$$

$$\equiv \frac{1}{2}\left\|\underbrace{\boldsymbol{z}^{(3)} - \boldsymbol{t}_i}_{=\boldsymbol{e}_i}\right\|_2^2 + \frac{\lambda}{2}\sum_{j=1}^{3}\left(\left\|\boldsymbol{W}^{(j)}\right\|_2^2 + \left\|\boldsymbol{b}^{(j)}\right\|_2^2\right) \tag{24}$$

$$\equiv \frac{1}{2}\left\|\boldsymbol{e}_i\right\|_2^2 + \frac{\lambda}{2}\sum_{j=1}^{3}\left(\left\|\boldsymbol{W}^{(j)}\right\|_2^2 + \left\|\boldsymbol{b}^{(j)}\right\|_2^2\right)\,, \tag{25}$$

where we defined the error

$$\boldsymbol{e}_i \coloneqq \boldsymbol{z}^{(3)} - \boldsymbol{t}_i = \boldsymbol{y}_i - \boldsymbol{t}_i\,. \tag{26}$$

In order to derive the gradients, we introduce some facts and notations for brevity as given below.

- Trace and Frobenius product are related as

$$\langle \boldsymbol{A}, \boldsymbol{BC}\rangle \coloneqq \mathrm{Tr}\left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{BC}\right) \coloneqq \boldsymbol{A} : \boldsymbol{BC}\,. \tag{27}$$

- The cyclic properties of Trace/Frobenius product can be summarized as following:

$$\boldsymbol{A} : \boldsymbol{BC} = \boldsymbol{BC} : \boldsymbol{A} \tag{28}$$

$$= \boldsymbol{AC}^{\mathrm{T}} : \boldsymbol{B} \tag{29}$$

$$= \text{etc.} \tag{30}$$

Thus, the loss function per i-th sample in (23) can be rewritten as

$$L_i = \frac{1}{2}\underbrace{\left(\underbrace{\boldsymbol{z}^{(3)}}_{=\boldsymbol{y}_i} - \boldsymbol{t}_i\right)}_{=\boldsymbol{e}_i} : \left(\underbrace{\boldsymbol{z}^{(3)}}_{=\boldsymbol{y}_i} - \boldsymbol{t}_i\right) + \frac{\lambda}{2}\sum_{j=1}^{3}\left(\boldsymbol{W}^{(j)} : \boldsymbol{W}^{(j)} + \boldsymbol{b}^{(j)} : \boldsymbol{b}^{(j)}\right). \tag{31}$$

To this end, we now calculate the differentials first and obtain the desired gradients subsequently.

$$dL_i = \boldsymbol{e}_i : d\boldsymbol{z}^{(3)} + \lambda\sum_{j=1}^{3}\left(\boldsymbol{W}^{(j)} : d\boldsymbol{W}^{(j)} + \boldsymbol{b}^{(j)} : d\boldsymbol{b}^{(j)}\right) \tag{32}$$

Utilizing the differentials of $d\boldsymbol{z}^{(3)}$ and $d\boldsymbol{h}^{(3)}$ in (10) and (12), respectively, such that the above differential dL_i reads

$$dL_i = \boldsymbol{e}_i : g^{(3)}\left(\boldsymbol{h}^{(3)}\right) \odot d\boldsymbol{h}^{(3)} + \lambda\sum_{j=1}^{3}\left(\boldsymbol{W}^{(j)} : d\boldsymbol{W}^{(j)} + \boldsymbol{b}^{(j)} : d\boldsymbol{b}^{(j)}\right) \tag{33}$$

$$= \boldsymbol{e}_i : \boldsymbol{G}^{(3)}\, d\boldsymbol{h}^{(3)} + \lambda\sum_{j=1}^{3}\left(\boldsymbol{W}^{(j)} : d\boldsymbol{W}^{(j)} + \boldsymbol{b}^{(j)} : d\boldsymbol{b}^{(j)}\right) \tag{34}$$

$$= \boldsymbol{e}_i : \boldsymbol{G}^{(3)}\left[d\boldsymbol{W}^{(3)}\,\boldsymbol{z}^{(2)} + \boldsymbol{W}^{(3)} d\boldsymbol{z}^{(2)} + d\boldsymbol{b}^{(3)}\right] + \lambda\sum_{j=1}^{3}\left(\boldsymbol{W}^{(j)} : d\boldsymbol{W}^{(j)} + \boldsymbol{b}^{(j)} : d\boldsymbol{b}^{(j)}\right) \tag{35}$$

$$\equiv \boldsymbol{e}_i : g^{(3)}\left(\boldsymbol{h}^{(3)}\right) \odot \left[d\boldsymbol{W}^{(3)}\,\boldsymbol{z}^{(2)} + \boldsymbol{W}^{(3)} d\boldsymbol{z}^{(2)} + d\boldsymbol{b}^{(3)}\right] + \lambda\sum_{j=1}^{3}\left(\boldsymbol{W}^{(j)} : d\boldsymbol{W}^{(j)} + \boldsymbol{b}^{(j)} : d\boldsymbol{b}^{(j)}\right) \tag{36}$$

To obtain the gradient of L_i with respect to $\boldsymbol{W}^{(3)}$, we now set $d\boldsymbol{z}^{(2)} = 0$, $d\boldsymbol{W}^{(j)} = 0 \; \forall j \neq 3$, and $d\boldsymbol{b}^{(j)} \; \forall j$ in (35) or (36), yielding

$$dL_i = \boldsymbol{e}_i : \boldsymbol{G}^{(3)} \left[d\boldsymbol{W}^{(3)} \; \boldsymbol{z}^{(2)} \right] + \lambda \boldsymbol{W}^{(3)} : d\boldsymbol{W}^{(3)} \tag{37}$$

$$= \left(\boldsymbol{G}^{(3)} \right)^{\mathrm{T}} \; \boldsymbol{e}_i \; \left(\boldsymbol{z}^{(2)} \right)^{\mathrm{T}} : d\boldsymbol{W}^{(3)} + \lambda \boldsymbol{W}^{(3)} : d\boldsymbol{W}^{(3)} \tag{38}$$

$$\equiv \left[g^{(3)} \left(\boldsymbol{h}^{(3)} \right) \; \odot \; \boldsymbol{e}_i \right] \; \left(\boldsymbol{z}^{(2)} \right)^{\mathrm{T}} : d\boldsymbol{W}^{(3)} + \lambda \boldsymbol{W}^{(3)} : d\boldsymbol{W}^{(3)} \tag{39}$$

$$\Rightarrow \frac{\partial L_i}{\partial \boldsymbol{W}^{(3)}} = \left(\boldsymbol{G}^{(3)} \right)^{\mathrm{T}} \; \boldsymbol{e}_i \; \left(\boldsymbol{z}^{(2)} \right)^{\mathrm{T}} + \lambda \boldsymbol{W}^{(3)} \tag{40}$$

$$= \boldsymbol{G}^{(3)} \; \boldsymbol{e}_i \; \left(\boldsymbol{z}^{(2)} \right)^{\mathrm{T}} + \lambda \boldsymbol{W}^{(3)} \tag{41}$$

$$\equiv \left[g^{(3)} \left(\boldsymbol{h}^{(3)} \right) \; \odot \; \boldsymbol{e}_i \right] \; \left(\boldsymbol{z}^{(2)} \right)^{\mathrm{T}} + \lambda \boldsymbol{W}^{(3)} \tag{42}$$

Similarly, to obtain the gradient of L_i with respect to $d\boldsymbol{b}^{(3)}$, we set $d\boldsymbol{z}^{(2)} = 0$, $d\boldsymbol{W}^{(j)} = 0 \; \forall j$, and $d\boldsymbol{b}^{(j)} \; \forall j \neq 3$ in (35) or (36), yielding

$$dL_i = \boldsymbol{e}_i : \boldsymbol{G}^{(3)} d\boldsymbol{b}^{(3)} + \lambda \boldsymbol{b}^{(3)} : d\boldsymbol{b}^{(3)} \tag{43}$$

$$\equiv \boldsymbol{e}_i : g^{(3)} \left(\boldsymbol{h}^{(3)} \right) \; \odot d\boldsymbol{b}^{(3)} + \lambda \boldsymbol{b}^{(3)} : d\boldsymbol{b}^{(3)} = \left[g^{(3)} \left(\boldsymbol{h}^{(3)} \right) \; \odot \; \boldsymbol{e}_i \right] : d\boldsymbol{b}^{(3)} + \lambda \boldsymbol{b}^{(3)} : d\boldsymbol{b}^{(3)} \tag{44}$$

$$\Rightarrow \frac{\partial L_i}{\partial \boldsymbol{b}^{(3)}} = \left(\boldsymbol{G}^{(3)} \right)^{\mathrm{T}} \; \boldsymbol{e}_i \; + \lambda \boldsymbol{b}^{(3)} \tag{45}$$

$$= \boldsymbol{G}^{(3)} \; \boldsymbol{e}_i \; + \lambda \boldsymbol{b}^{(3)} \tag{46}$$

$$\equiv \left[g^{(3)} \left(\boldsymbol{h}^{(3)} \right) \; \odot \; \boldsymbol{e}_i \right] + \lambda \boldsymbol{b}^{(3)} \tag{47}$$

To this end, we utilize the differentials of $d\boldsymbol{z}^{(2)}$ in (10) and $d\boldsymbol{h}^{(2)}$ in (12) such that the above differential dL_i in (35) while setting $d\boldsymbol{W}^{(j)} = 0 \; \forall j \neq 2$ and $d\boldsymbol{b}^{(3)} = 0 \; \forall j \neq 2$ can be shown as

$$dL_i = \boldsymbol{e}_i : \boldsymbol{G}^{(3)} \; \left[\boldsymbol{W}^{(3)} d\boldsymbol{z}^{(2)} \right] + \lambda \left(\boldsymbol{W}^{(2)} : d\boldsymbol{W}^{(2)} + \boldsymbol{b}^{(2)} : d\boldsymbol{b}^{(2)} \right) \tag{48}$$

$$= \boldsymbol{e}_i : \boldsymbol{G}^{(3)} \boldsymbol{W}^{(3)} \boldsymbol{G}^{(2)} d\boldsymbol{h}^{(2)} + \lambda \left(\boldsymbol{W}^{(2)} : d\boldsymbol{W}^{(2)} + \boldsymbol{b}^{(2)} : d\boldsymbol{b}^{(2)} \right) \tag{49}$$

$$= \left(\boldsymbol{G}^{(3)} \boldsymbol{W}^{(3)} \boldsymbol{G}^{(2)} \right)^{\mathrm{T}} \boldsymbol{e}_i : d\boldsymbol{h}^{(2)} + \lambda \left(\boldsymbol{W}^{(2)} : d\boldsymbol{W}^{(2)} + \boldsymbol{b}^{(2)} : d\boldsymbol{b}^{(2)} \right) \tag{50}$$

$$= \left(\boldsymbol{G}^{(3)} \boldsymbol{W}^{(3)} \boldsymbol{G}^{(2)} \right)^{\mathrm{T}} \boldsymbol{e}_i : \left[d\boldsymbol{W}^{(2)} \; \boldsymbol{z}^{(1)} + \boldsymbol{W}^{(2)} d\boldsymbol{z}^{(1)} + d\boldsymbol{b}^{(2)} \right]$$
$$+ \lambda \left(\boldsymbol{W}^{(2)} : d\boldsymbol{W}^{(2)} + \boldsymbol{b}^{(2)} : d\boldsymbol{b}^{(2)} \right) \tag{51}$$

To obtain the gradient of L_i with respect to $\boldsymbol{W}^{(2)}$, we now set $d\boldsymbol{z}^{(1)} = 0$ and $d\boldsymbol{b}^{(2)} = 0$ in (51), yielding

$$dL_i = \left(\boldsymbol{G}^{(3)} \boldsymbol{W}^{(3)} \boldsymbol{G}^{(2)} \right)^{\mathrm{T}} \; \boldsymbol{e}_i \left(\boldsymbol{z}^{(1)} \right)^{\mathrm{T}} : \; d\boldsymbol{W}^{(2)} \; + \lambda \left(\boldsymbol{W}^{(2)} : d\boldsymbol{W}^{(2)} \right) \tag{52}$$

$$\Rightarrow \frac{\partial L_i}{\partial \boldsymbol{W}^{(2)}} = \left(\boldsymbol{G}^{(3)} \boldsymbol{W}^{(3)} \boldsymbol{G}^{(2)} \right)^{\mathrm{T}} \; \boldsymbol{e}_i \left(\boldsymbol{z}^{(1)} \right)^{\mathrm{T}} + \lambda \boldsymbol{W}^{(2)} = \boldsymbol{G}^{(2)} \left(\boldsymbol{W}^{(3)} \right)^{\mathrm{T}} \boldsymbol{G}^{(3)} \; \boldsymbol{e}_i \; \left(\boldsymbol{z}^{(1)} \right)^{\mathrm{T}} + \lambda \boldsymbol{W}^{(2)} \; . \tag{53}$$

To compute the gradient $\frac{\partial L_i}{\partial \boldsymbol{b}^{(2)}}$, we set $d\boldsymbol{z}^{(1)} = 0$ and $d\boldsymbol{W}^{(2)} = 0$ in the above differential (51) such that

$$dL_i = \left(\boldsymbol{G}^{(3)}\boldsymbol{W}^{(3)}\boldsymbol{G}^{(2)}\right)^{\mathrm{T}} \boldsymbol{e}_i : \; d\boldsymbol{b}^{(2)} \; + \lambda \left(\boldsymbol{b}^{(2)} : d\boldsymbol{b}^{(2)}\right) \tag{54}$$

$$\Rightarrow \frac{\partial L_i}{\partial \boldsymbol{b}^{(2)}} = \left(\boldsymbol{G}^{(3)}\boldsymbol{W}^{(3)}\boldsymbol{G}^{(2)}\right)^{\mathrm{T}} \boldsymbol{e}_i + \lambda \boldsymbol{b}^{(2)} = \boldsymbol{G}^{(2)} \left(\boldsymbol{W}^{(3)}\right)^{\mathrm{T}} \boldsymbol{G}^{(3)} \, \boldsymbol{e}_i + \lambda \boldsymbol{b}^{(2)} \, . \tag{55}$$

To compute the gradient $\frac{\partial L_i}{\partial \boldsymbol{W}^{(2)}}$ in the element-wise notation, the above differential can be rewritten as

$$dL_i = \boldsymbol{e}_i : g^{(3)}\left(\boldsymbol{h}^{(3)}\right) \; \odot \; \left[\boldsymbol{W}^{(3)} d\boldsymbol{z}^{(2)}\right] + \lambda \left(\boldsymbol{W}^{(2)} : d\boldsymbol{W}^{(2)} + \boldsymbol{b}^{(2)} : d\boldsymbol{b}^{(2)}\right) \tag{56}$$

$$= \left(g^{(3)}\left(\boldsymbol{h}^{(3)}\right) \; \odot \; \boldsymbol{e}_i\right) : \left[\boldsymbol{W}^{(3)} d\boldsymbol{z}^{(2)}\right] + \lambda \left(\boldsymbol{W}^{(2)} : d\boldsymbol{W}^{(2)} + \boldsymbol{b}^{(2)} : d\boldsymbol{b}^{(2)}\right) \tag{57}$$

$$= \left(\boldsymbol{W}^{(3)}\right)^{\mathrm{T}} \left(g^{(3)}\left(\boldsymbol{h}^{(3)}\right) \; \odot \; \boldsymbol{e}_i\right) : \left[d\boldsymbol{z}^{(2)}\right] + \lambda \left(\boldsymbol{W}^{(2)} : d\boldsymbol{W}^{(2)} + \boldsymbol{b}^{(2)} : d\boldsymbol{b}^{(2)}\right) \tag{58}$$

$$= \left(\boldsymbol{W}^{(3)}\right)^{\mathrm{T}} \left(g^{(3)}\left(\boldsymbol{h}^{(3)}\right) \; \odot \; \boldsymbol{e}_i\right) : \left[g^{(2)}\left(\boldsymbol{h}^{(2)}\right) \; \odot \; d\boldsymbol{h}^{(2)}\right] + \lambda \left(\boldsymbol{W}^{(2)} : d\boldsymbol{W}^{(2)} + \boldsymbol{b}^{(2)} : d\boldsymbol{b}^{(2)}\right) \tag{59}$$

$$= g^{(2)}\left(\boldsymbol{h}^{(2)}\right) \; \odot \; \left[\left(\boldsymbol{W}^{(3)}\right)^{\mathrm{T}} \left(g^{(3)}\left(\boldsymbol{h}^{(3)}\right) \; \odot \; \boldsymbol{e}_i\right)\right] : \left[d\boldsymbol{h}^{(2)}\right] + \lambda \left(\boldsymbol{W}^{(2)} : d\boldsymbol{W}^{(2)} + \boldsymbol{b}^{(2)} : d\boldsymbol{b}^{(2)}\right) \tag{60}$$

$$= g^{(2)}\left(\boldsymbol{h}^{(2)}\right) \; \odot \; \left[\left(\boldsymbol{W}^{(3)}\right)^{\mathrm{T}} \left(g^{(3)}\left(\boldsymbol{h}^{(3)}\right) \; \odot \; \boldsymbol{e}_i\right)\right] : \left[d\boldsymbol{h}^{(2)}\right]$$
$$+ \lambda \left(\boldsymbol{W}^{(2)} : d\boldsymbol{W}^{(2)} + \boldsymbol{b}^{(2)} : d\boldsymbol{b}^{(2)}\right) \tag{61}$$

$$= g^{(2)}\left(\boldsymbol{h}^{(2)}\right) \; \odot \; \left[\left(\boldsymbol{W}^{(3)}\right)^{\mathrm{T}} \left(g^{(3)}\left(\boldsymbol{h}^{(3)}\right) \; \odot \; \boldsymbol{e}_i\right)\right] : \left[d\boldsymbol{W}^{(2)} \; \boldsymbol{z}^{(1)} + \boldsymbol{W}^{(2)} d\boldsymbol{z}^{(1)} + d\boldsymbol{b}^{(2)}\right]$$
$$+ \lambda \left(\boldsymbol{W}^{(2)} : d\boldsymbol{W}^{(2)} + \boldsymbol{b}^{(2)} : d\boldsymbol{b}^{(2)}\right) \tag{62}$$

We now set $d\boldsymbol{z}^{(1)} = 0$ and $d\boldsymbol{b}^{(2)} = 0$ such that

$$dL_i = g^{(2)}\left(\boldsymbol{h}^{(2)}\right) \; \odot \; \left[\left(\boldsymbol{W}^{(3)}\right)^{\mathrm{T}} \left(g^{(3)}\left(\boldsymbol{h}^{(3)}\right) \; \odot \; \boldsymbol{e}_i\right)\right] : \left[d\boldsymbol{W}^{(2)} \; \boldsymbol{z}^{(1)}\right] + \lambda \left(\boldsymbol{W}^{(2)} : d\boldsymbol{W}^{(2)}\right) \tag{63}$$

$$= g^{(2)}\left(\boldsymbol{h}^{(2)}\right) \; \odot \; \left[\left(\boldsymbol{W}^{(3)}\right)^{\mathrm{T}} \left(g^{(3)}\left(\boldsymbol{h}^{(3)}\right) \; \odot \; \boldsymbol{e}_i\right)\right] \left(\boldsymbol{z}^{(1)}\right)^{\mathrm{T}} : \left[d\boldsymbol{W}^{(2)}\right] + \lambda \left(\boldsymbol{W}^{(2)} : d\boldsymbol{W}^{(2)}\right) \tag{64}$$

$$\Rightarrow \frac{\partial L_i}{\partial \boldsymbol{W}^{(2)}} = g^{(2)}\left(\boldsymbol{h}^{(2)}\right) \; \odot \; \left[\left(\boldsymbol{W}^{(3)}\right)^{\mathrm{T}} \left(g^{(3)}\left(\boldsymbol{h}^{(3)}\right) \; \odot \; \boldsymbol{e}_i\right)\right] \left(\boldsymbol{z}^{(1)}\right)^{\mathrm{T}} + \lambda \boldsymbol{W}^{(2)} \, . \tag{65}$$

Similarly, the gradient $\frac{\partial L_i}{\partial \boldsymbol{b}^{(2)}}$ in the element-wise notation can be expressed as

$$\frac{\partial L_i}{\partial \boldsymbol{b}^{(2)}} = g^{(2)}\left(\boldsymbol{h}^{(2)}\right) \; \odot \; \left[\left(\boldsymbol{W}^{(3)}\right)^{\mathrm{T}} \left(g^{(3)}\left(\boldsymbol{h}^{(3)}\right) \; \odot \; \boldsymbol{e}_i\right)\right] + \lambda \boldsymbol{b}^{(2)} \, . \tag{66}$$

To compute the gradient $\frac{\partial L_i}{\partial \boldsymbol{W}^{(1)}}$ and $\frac{\partial L_i}{\partial \boldsymbol{b}^{(1)}}$, we utilize the differential (51) and set $d\boldsymbol{W}^{(2)} = 0$ and

$d\boldsymbol{b}^{(2)} = 0$, such that

$$dL_i = \left(\boldsymbol{G}^{(3)}\boldsymbol{W}^{(3)}\boldsymbol{G}^{(2)}\right)^{\mathrm{T}} \boldsymbol{e}_i : \left[\boldsymbol{W}^{(2)} d\boldsymbol{z}^{(1)}\right] + \lambda \left(\boldsymbol{W}^{(1)} : d\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)} : d\boldsymbol{b}^{(1)}\right) \tag{67}$$

$$= \left(\boldsymbol{G}^{(3)}\boldsymbol{W}^{(3)}\boldsymbol{G}^{(2)}\boldsymbol{W}^{(2)}\right)^{\mathrm{T}} \boldsymbol{e}_i : \left[d\boldsymbol{z}^{(1)}\right] + \lambda \left(\boldsymbol{W}^{(1)} : d\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)} : d\boldsymbol{b}^{(1)}\right) \tag{68}$$

$$= \left(\boldsymbol{G}^{(3)}\boldsymbol{W}^{(3)}\boldsymbol{G}^{(2)}\boldsymbol{W}^{(2)}\right)^{\mathrm{T}} \boldsymbol{e}_i : \left[\boldsymbol{G}^{(1)} d\boldsymbol{h}^{(1)}\right] + \lambda \left(\boldsymbol{W}^{(1)} : d\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)} : d\boldsymbol{b}^{(1)}\right) \tag{69}$$

$$= \left(\boldsymbol{G}^{(3)}\boldsymbol{W}^{(3)}\boldsymbol{G}^{(2)}\boldsymbol{W}^{(2)}\boldsymbol{G}^{(1)}\right)^{\mathrm{T}} \boldsymbol{e}_i : \left[d\boldsymbol{h}^{(1)}\right] + \lambda \left(\boldsymbol{W}^{(1)} : d\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)} : d\boldsymbol{b}^{(1)}\right) \tag{70}$$

$$= \left(\boldsymbol{G}^{(3)}\boldsymbol{W}^{(3)}\boldsymbol{G}^{(2)}\boldsymbol{W}^{(2)}\boldsymbol{G}^{(1)}\right)^{\mathrm{T}} \boldsymbol{e}_i : \left[d\boldsymbol{W}^{(1)}\, \boldsymbol{x}_i + d\boldsymbol{b}^{(1)}\right] + \lambda \left(\boldsymbol{W}^{(1)} : d\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)} : d\boldsymbol{b}^{(1)}\right). \tag{71}$$

The gradients are

$$\frac{\partial L_i}{\partial \boldsymbol{W}^{(1)}} = \left(\boldsymbol{G}^{(3)}\boldsymbol{W}^{(3)}\boldsymbol{G}^{(2)}\boldsymbol{W}^{(2)}\boldsymbol{G}^{(1)}\right)^{\mathrm{T}} \boldsymbol{e}_i\, \boldsymbol{x}_i^{\mathrm{T}} + \lambda \boldsymbol{W}^{(1)} \tag{72}$$

and

$$\frac{\partial L_i}{\partial \boldsymbol{b}^{(1)}} = \left(\boldsymbol{G}^{(3)}\boldsymbol{W}^{(3)}\boldsymbol{G}^{(2)}\boldsymbol{W}^{(2)}\boldsymbol{G}^{(1)}\right)^{\mathrm{T}} \boldsymbol{e}_i + \lambda \boldsymbol{b}^{(1)}. \tag{73}$$

In order to obtain the gradients $\frac{\partial L_i}{\partial \boldsymbol{W}^{(1)}}$ and $\frac{\partial L_i}{\partial \boldsymbol{b}^{(1)}}$ in element-wise notation, then we continue with the differential (62) and set $d\boldsymbol{W}^{(2)} = 0$ and $d\boldsymbol{b}^{(2)} = 0$ yielding

$$dL_i = g^{(2)}\left(\boldsymbol{h}^{(2)}\right) \odot \left[\left(\boldsymbol{W}^{(3)}\right)^{\mathrm{T}} \left(g^{(3)}\left(\boldsymbol{h}^{(3)}\right) \odot \boldsymbol{e}_i\right)\right] : \left[\boldsymbol{W}^{(2)} d\boldsymbol{z}^{(1)}\right] + \lambda \left(\boldsymbol{W}^{(1)} : d\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)} : d\boldsymbol{b}^{(1)}\right) \tag{74}$$

$$= \left(\boldsymbol{W}^{(2)}\right)^{\mathrm{T}} g^{(2)}\left(\boldsymbol{h}^{(2)}\right) \odot \left[\left(\boldsymbol{W}^{(3)}\right)^{\mathrm{T}} \left(g^{(3)}\left(\boldsymbol{h}^{(3)}\right) \odot \boldsymbol{e}_i\right)\right] : \left[d\boldsymbol{z}^{(1)}\right]$$
$$+ \lambda \left(\boldsymbol{W}^{(1)} : d\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)} : d\boldsymbol{b}^{(1)}\right) \tag{75}$$

$$= \left(\boldsymbol{W}^{(2)}\right)^{\mathrm{T}} g^{(2)}\left(\boldsymbol{h}^{(2)}\right) \odot \left[\left(\boldsymbol{W}^{(3)}\right)^{\mathrm{T}} \left(g^{(3)}\left(\boldsymbol{h}^{(3)}\right) \odot \boldsymbol{e}_i\right)\right] : \left[g^{(1)}\left(\boldsymbol{h}^{(1)}\right) \odot d\boldsymbol{h}^{(1)}\right]$$
$$+ \lambda \left(\boldsymbol{W}^{(1)} : d\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)} : d\boldsymbol{b}^{(1)}\right) \tag{76}$$

$$= g^{(1)}\left(\boldsymbol{h}^{(1)}\right) \odot \left(\boldsymbol{W}^{(2)}\right)^{\mathrm{T}} g^{(2)}\left(\boldsymbol{h}^{(2)}\right) \odot \left[\left(\boldsymbol{W}^{(3)}\right)^{\mathrm{T}} \left(g^{(3)}\left(\boldsymbol{h}^{(3)}\right) \odot \boldsymbol{e}_i\right)\right] : \left[d\boldsymbol{h}^{(1)}\right]$$
$$+ \lambda \left(\boldsymbol{W}^{(1)} : d\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)} : d\boldsymbol{b}^{(1)}\right) \tag{77}$$

$$= g^{(1)}\left(\boldsymbol{h}^{(1)}\right) \odot \left(\boldsymbol{W}^{(2)}\right)^{\mathrm{T}} g^{(2)}\left(\boldsymbol{h}^{(2)}\right) \odot \left[\left(\boldsymbol{W}^{(3)}\right)^{\mathrm{T}} \left(g^{(3)}\left(\boldsymbol{h}^{(3)}\right) \odot \boldsymbol{e}_i\right)\right] : \left[d\boldsymbol{W}^{(1)}\, \boldsymbol{x}_i + d\boldsymbol{b}^{(1)}\right]$$
$$+ \lambda \left(\boldsymbol{W}^{(1)} : d\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)} : d\boldsymbol{b}^{(1)}\right) \tag{78}$$

The gradients $\frac{\partial L_i}{\partial \boldsymbol{W}^{(1)}}$ and $\frac{\partial L_i}{\partial \boldsymbol{b}^{(1)}}$ are

$$\frac{\partial L_i}{\partial \boldsymbol{W}^{(1)}} = g^{(1)}\left(\boldsymbol{h}^{(1)}\right) \odot \left(\boldsymbol{W}^{(2)}\right)^{\mathrm{T}} g^{(2)}\left(\boldsymbol{h}^{(2)}\right) \odot \left[\left(\boldsymbol{W}^{(3)}\right)^{\mathrm{T}} \left(g^{(3)}\left(\boldsymbol{h}^{(3)}\right) \odot \boldsymbol{e}_i\right)\right] \boldsymbol{x}_i^{\mathrm{T}} + \lambda \boldsymbol{W}^{(1)} \tag{79}$$

and

$$\frac{\partial L_i}{\partial \boldsymbol{b}^{(1)}} = g^{(1)}\left(\boldsymbol{h}^{(1)}\right) \odot \left(\boldsymbol{W}^{(2)}\right)^{\mathrm{T}} \; g^{(2)}\left(\boldsymbol{h}^{(2)}\right) \; \odot \; \left[\left(\boldsymbol{W}^{(3)}\right)^{\mathrm{T}} \left(g^{(3)}\left(\boldsymbol{h}^{(3)}\right) \; \odot \; \boldsymbol{e}_i\right)\right] + \lambda \boldsymbol{b}^{(1)} \,, \tag{80}$$

respectively.

The total gradient with respect to $\boldsymbol{W}^{(\ell)}$ or $\boldsymbol{b}^{(\ell)}$ over all the samples is the mean of all the gradients, i.e.,

$$L = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial L_i}{\partial \boldsymbol{W}^{(\ell)}} \tag{81}$$

or

$$L = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial L_i}{\partial \boldsymbol{b}^{(\ell)}} \,. \tag{82}$$

References

[1] Bubeck, Sbastien, ”Convex optimization: Algorithms and complexity,” Foundations and Trends in Machine Learning, vol. 8, no.3–4, pp. 231–357, 2015.

[2] L. Bottou, F. Curtis, J. Norcedal, ”Optimization Methods for Large-Scale Machine Learning,” SIAM Rev., 60, no. 2, pp. 223-311, 2018.

[3] T. Hastie, R. Tibshirani, and J. Friedman, ”The Elements of Statistical Learning: Data Mining, Inference and Prediction,” Second edition, Springer, 2009.

[4] C. D. Meyer, ”Matrix Analysis and Applied Linear Algebra,” SIAM, 2000.

[5] J. R. Magnus and H. Neudecker, ”Matrix differential calculus with applications in statistics and econometrics,” 2nd Edition, Wiley, UK, 1999.

[6] K. P. Murphy, ”Machine learning - a probabilistic perspective,” MIT Press, 2012.

[7] M. Schmidt,N. Le Roux, F. Bach, ”Minimizing finite sums with the stochastic average gradient,” Mathematical Programming, 2017.

[8] R. Johnson and T. Zhang, ”Accelerating stochastic gradient descent using predictive variance reduction,” NIPS, 2013.

[9] G. Klambauer, T. Unterthiner, and A. Mayr, ”Self-Normalizing Neural Networks”, in arXiv:1706.02515v5, 7 Sep. 2017.

[10] S. Santurkar, D. Tsipras, A. Ilyas, A. Madry, “How Does Batch Normalization Help Optimization?(No, It Is Not About Internal Covariate Shift)”. arXiv preprint arXiv:1805.11604, 2018.

[11] S. Wu, G. Li, L. Deng, L. Liu, D. Wu, Y. Xie, L. Shi, “L1-norm batch normalization for efficient training of deep neural networks,” IEEE transactions on neural networks and learning systems, 2018.