# REPORT FOR THE FIRST HOMEWORK OF THE ADVANCED MACHINE LEARNING COURSE

## *Question 2*

a) Verify that the loss function defined in Eq. (5) has gradient w.r.t $z_i^{(3)}$ as below:

$$\frac{\delta J}{\delta\left(z_i^{(3)}\right)}\left(\theta, \{x_i, y_i\}^N{}_{i=1}\right) = \frac{1}{N}\left(\psi\left(z_i^{(3)}\right) - \Delta_i\right)$$

**Answer:**

Based on the definition of the SoftMax function

$$\psi(u_i) = \frac{e^{u_i}}{\sum_j e^{u_j}}$$

we have that $\psi(u_i)$ is defined based on each output, with a index $j$ that represents the class of the output, and each class that is normalized by the summation of all the activations.

We also define the cross-entropy function as:

$$J(\theta, x_i, y_i) = -\log P\left(Y = y_i, X = x_i\right)$$

$$= -\log f_\theta(x_i)_{y_i} = -\log \psi\left(z_i^{(3)}\right)_{y_i}$$

$$J(\theta, x_i, y_i) = -\log\left[\frac{\exp\left(z_i^{(3)}\right)_{y_i}}{\sum_{j=1}^K \exp\left(z_i^{(3)}\right)_j}\right]$$

Then we must average over the whole training set, so we obtain:

$$J\left(\theta, \{x_i, y_i\}_{i=1}^N\right) = \frac{1}{N}\sum_{i=1}^N -\log\left[\frac{\exp\left(z_i^{(3)}\right)_{y_i}}{\sum_{j=1}^K \exp\left(z_i^{(3)}\right)_j}\right]$$

In order to explain it in a more simplified way, let's see the derivation of the SoftMax function since it is the essence of the loss function.

We consider two cases, one where the output index is equal to the input index, the other when they are not equal:

$$(\Delta_i)_j = \begin{cases} 1, & if\ \ j = y_i \\ 0, & otherwise \end{cases}$$

Case 1:

$$\frac{\delta\psi(u_i)}{\delta u_j} = \frac{e^{u_i}\sum_j e^{u_j} - e^{u_i}e^{u_i}}{\left(\sum_j e^{u_j}\right)^2} = \psi(u_i)(1-\psi(u_i))$$

Case 2:

$$\frac{\psi(u_i)}{\delta u_j} = \frac{0 - e^{u_i}e^{u_j}}{\left(\sum_j e^{u_j}\right)^2} = -\psi(u_i)\psi(u_j)$$

Now let's go through the derivation of the cross-entropy function:

$$\frac{\delta J\left(\theta, \{x_i, y_i\}_{i=1}^N\right)}{\delta\, z_i^{(3)}} = -\frac{1}{N}\sum_i y_i \cdot \frac{1}{z_i^{(3)}}$$

Since there are two cases, we would combine them here together, and obtain the result we expected:

$$\frac{\delta J\left(\theta, \{x_i, y_i\}_{i=1}^N\right)}{\delta\, z_i^{(3)}} = -\frac{1}{N}\sum_{i\neq j}\left[y_i \cdot \frac{1}{z_i^{(3)}}\frac{\delta\hat{y}_i}{\delta\left(z_i^{(3)}\right)_j} - y_j \cdot \frac{1}{\hat{y}_j}\frac{\delta y_j}{\delta\left(z_i^{(3)}\right)_j}\right]$$

$$= -\frac{1}{N}\sum_{i\neq j}\left[y_i \cdot \frac{1}{z_i^{(3)}}\left(-z_i^{(3)}\hat{y}_j\right) - y_i \cdot \frac{1}{\hat{y}_j}\hat{y}_j(1-\hat{y}_j)\right] = \frac{1}{N}\sum_{i\neq j}\left[\left(y_i\hat{y}_j\right) + \left(y_j\hat{y}_j\right) - y_j\right]$$

$$= \frac{1}{N}\sum_i\left[y_i\hat{y}_j - y_i\right] = \frac{1}{N}\left[\hat{y}_j - y_i\right] = \frac{1}{N}\left(\psi\left(z_i^{(3)}\right) - \Delta_i\right)$$

b) To compute the effect of the weight matrix $W^{(2)}$ on the loss in Eq. (5) as incurred by the network, we compute the partial derivative of the loss function with respect to $W^{(2)}$. This is done by applying the chain rule. Verify that the partial derivative of the loss w.r.t. W (2) is:

$$\frac{\delta J}{\delta W^{(2)}}\left(\theta, \{x_i, y_i\}_{i=1}^N\right) = \sum_{i=1}^N \frac{\delta J}{\delta z_i^{(3)}} \cdot \frac{\delta z_i^{(3)}}{\delta W^{(2)}} = \sum_{i=1}^N \frac{1}{N}\left(\psi\left(z_i^{(3)}\right) - \Delta_i\right) a_i^{(2)^T}$$

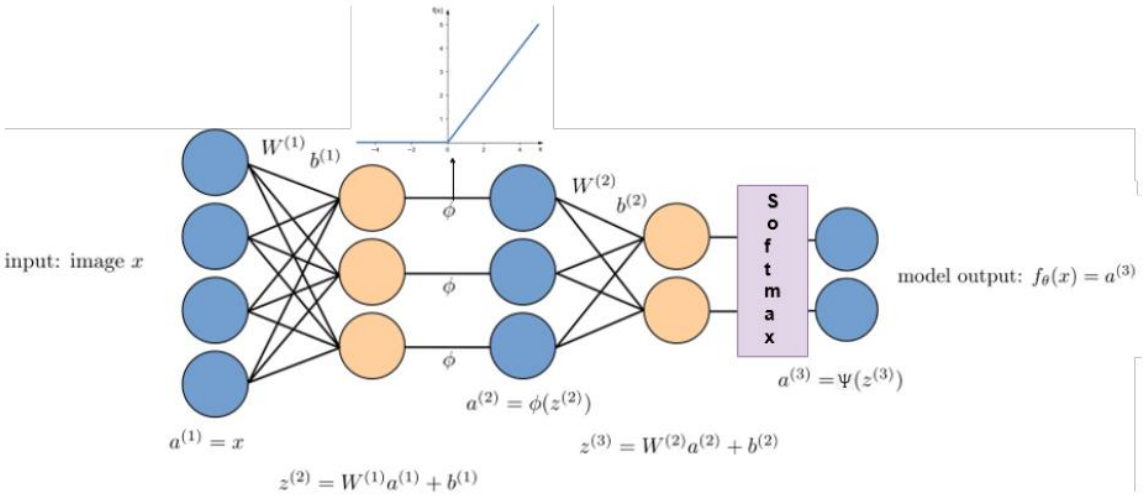
Similarly verify that the regularized loss in Eq. (6) has the derivative:

$$\frac{\delta\tilde{J}}{\delta W^{(2)}} = \sum_{i=1}^N \frac{1}{N}\left(\psi\left(z_i^{(3)}\right) - \Delta_i\right) a_i^{(2)^T} + 2\,\lambda W^{(2)}$$

You may refer to the formulae of matrix backpropagation seen in the class given $Y = XW$, then $\frac{\delta L}{\delta X} = \frac{\delta L}{\delta Y}W^T$ and $\frac{\delta L}{\delta W} = X^T\frac{\delta L}{\delta Y}$. But you should always watch out for the different formulation in Eq. (3), as also pointed out above.

**Answer:**

First, let's take a closer look at the structure of the neural network:




In order to minimize the difference between our neural network's output (predicted class) and the target, we need to know how the model performance changes with respect to each parameter in our model. In other words, we need to define the relationship between our cost function (J) and each weight (W). We can then update these weights in an iterative process using gradient descent. Since $W^{(2)}$ is the coefficient of $z^{(3)}$, and $z^{(3)}$ is an input to $a^{(3)}$, by the same logic $a^{(3)}$ is a component of $J\left(\theta, \{x_i, y_i\}_{i=1}^{N}\right)$. Then, given that:

$$a^{(3)} = \psi\left(z^{(3)}\right)$$

we can then use a repeated application of the chain rule, going backward through the neural network, in order to obtain the gradient of the loss with respect to each neuron of the network:

$$\psi\left(z^{(3)}\right) = \psi\left(W^{(2)}a^{(2)} + b^{(2)}\right) = \psi\left(W^{(2)}\phi\left(z^{(2)}\right) + b^{(2)}\right) = \psi\left(W^{(2)}\phi\left(W^{(1)}a^{(1)} + b^{(1)}\right) + b^{(2)}\right)$$

Before starting, we should also mention that the derivation of the ReLU function ($\phi$) will be:

$$\phi'(x) = \begin{cases} 1 & if\ x > 0 \\ 0 & otherwise \end{cases}$$

First, we want to calculate the derivative of the loss function with respect to $W^{(2)}$ and $b^{(2)}$

$$\frac{\delta J}{\delta W^{(2)}} = \frac{1}{N}\frac{\delta J}{\delta a^{(3)}}\frac{\delta a^{(3)}}{\delta z^{(3)}}\frac{\delta z^{(3)}}{\delta W^{(2)}}$$

$$\frac{\delta J}{\delta b^{(2)}} = \frac{1}{N}\frac{\delta J}{\delta a^{(3)}}\frac{\delta a^{(3)}}{\delta z^{(3)}}\frac{\delta z^{(3)}}{\delta b^{(2)}}$$

Looking at the first two terms in the above equations:

$$\frac{\delta J}{\delta a^{(3)}} \frac{\delta a^{(3)}}{\delta z^{(3)}}$$

we see that the component $\delta a^{(3)}$ can be cancelled out, so we end up with $\frac{\delta J}{\delta z^{(3)}}$, which we already solved in the previous part, and it is equal to:

$$\frac{1}{N}\left(\psi\left(z_i^{(3)}\right) - \Delta_i\right)$$

Since $z^{(3)} = W^{(2)}a^{(2)} + b^{(2)}$ and we want to compute $\frac{\delta z^{(3)}}{\delta W^{(2)}}$, the derivative over $b^{(2)}$ is equal to 0, therefore:

$$\frac{\delta z^{(3)}}{\delta W^{(2)}} = a_i^{(2)^T}$$

$$\frac{\delta J}{\delta W^{(2)}} = \frac{1}{N}\left(\psi\left(z_i^{(3)}\right) - \Delta_i\right)\, a_i^{(2)^T}$$

As of solving the derivative with the regularization term, we need to compute also the derivative of:

$$\frac{\delta\left(\lambda\left(\left|\left|W^{(1)}\right|\right|_2^2 + \left|\left|W^{(2)}\right|\right|_2^2\right)\right)}{\delta W^{(2)}}$$

Since we are only computing the derivative with respect to $W^{(2)}$, the derivative of the first term inside the regularization is going to be equal to zero. (i.e., $\frac{\delta\left(\lambda\left(\left|\left|W^{(1)}\right|\right|_2^2\right)\right)}{\delta W^{(2)}} = 0$).

Therefore:

$$\frac{\delta\left(\lambda\left(\left|\left|W^{(2)}\right|\right|_2^2\right)\right)}{\delta W^{(2)}} = \frac{\delta\left(\lambda\left(\sum_{p=1}^{3}\sum_{q=1}^{10}\left(W_{pq}^{(1)}\right)^2\right)\right)}{\delta W^{(2)}} = 2\lambda W^{(2)}$$

$$\frac{\delta \tilde{J}}{\delta W^{(2)}} = \sum_{i=1}^{N}\frac{1}{N}\left(\psi\left(z_i^{(3)}\right) - \Delta_i\right)a_i^{(2)^T} + 2\,\lambda W^{(2)}$$

c) We can repeatedly apply chain rule as discussed above to obtain the derivatives of the loss with respect to all the parameters for the model $\theta = \left(W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}\right)$. Derive the expressions for the derivatives of the regularized loss in Eq. (6) w.r.t. $W^{(1)}, b^{(1)}, b^{(2)}$ now.

**Answer:**

We are interested in solving the following equation:

$$\frac{\delta J}{\delta W^{(1)}} = \frac{\delta J}{\delta z^{(3)}} \frac{\delta z^{(3)}}{\delta a^{(2)}} \frac{\delta a^{(2)}}{\delta z^{(2)}} \frac{\delta z^{(2)}}{\delta W^{(1)}} = ?$$

To do so, we proceed as follows:

$$\frac{\delta J}{\delta z^{(3)}} = \frac{1}{N}\left(\psi\left(z_i^{(3)}\right) - \Delta_i\right)$$

$$\frac{\delta z^{(3)}}{\delta a^{(2)}} = W^{(2)}$$

$$\frac{\delta a^{(2)}}{\delta z^{(2)}} = \phi'\left(z^{(2)}\right)$$

$$\frac{\delta z^{(2)}}{\delta W^{(1)}} = a^{(1)^T}$$

Therefore:

$$\frac{\delta J}{\delta W^{(1)}} = \frac{1}{N}\left(\psi\left(z_i^{(3)}\right) - \Delta_i\right) W^{(2)} \phi'\left(z^{(2)}\right) a^{(1)^T}$$

As for the next derivative:

$$\frac{\delta J}{\delta b^{(1)}} = \frac{\delta J}{\delta z^{(3)}} \frac{\delta z^{(3)}}{\delta a^{(2)}} \frac{\delta a^{(2)}}{\delta z^{(2)}} \frac{\delta z^{(2)}}{\delta b^{(1)}} = ?$$

We have already computed the first three terms of the chain rule. The only remaining fraction of the equation is $\frac{\delta z^{(2)}}{\delta b^{(1)}}$ which is equal to 1. Therefore:

$$\frac{\delta J}{\delta b^{(1)}} = \frac{1}{N}\left(\psi\left(z_i^{(3)}\right) - \Delta_i\right) W^{(2)} \phi'\left(z^{(2)}\right)$$

Then, we are only left with computing:

$$\frac{\delta J}{\delta b^{(2)}} = \frac{\delta J}{\delta z^{(3)}} \frac{\delta z^{(3)}}{\delta b^{(2)}}$$

$$\frac{\delta z^{(3)}}{\delta b^{(2)}} = 1$$

$$\frac{\delta J}{\delta b^{(2)}} = \frac{1}{N}\left(\psi\left(z_i^{(3)}\right) - \Delta_i\right)$$

Since we have already computed the derivative of $ReLU(z^{(2)})$, which is equal to:

$$\phi'(z^{(2)}) = \frac{\delta a^{(2)}}{\delta z^{(2)}} = \begin{cases} 1 & if\ z^{(2)} > 0 \\ 0 & otherwise \end{cases} = sgn(a^{(2)})$$

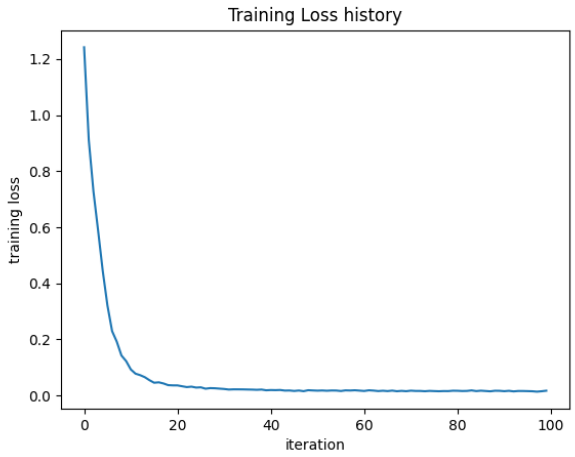
We can now see that the Jacobian $\frac{\delta z^{(2)}}{\delta a^{(1)}}$ is a diagonal matrix where the entry at $(i, i)$ is the derivative of the ReLU function applied to $a^{(1)}$.

### *Question 3*

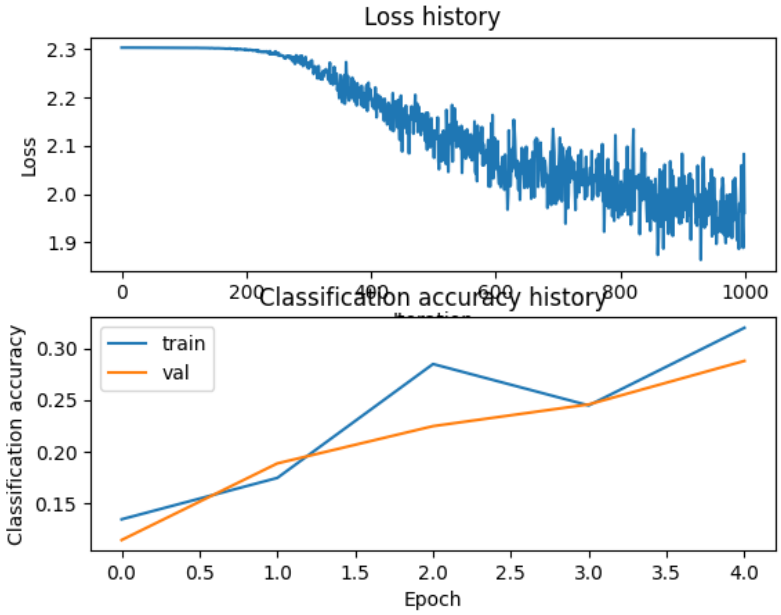
b) Your task is to debug the model training and come up with better hyper-parameters to improve the performance on the validation set. Visualize the training and validation performance curves to help with this analysis. Once you have tuned your hyper-parameters and get validation accuracy greater than 48% run your best model on the test set once and report the performance.

**Answer:**

Our final model's loss is lower than **0.02** as indicated in the graph below:




Also, by looking at the loss and at the model's accuracy over each epoch, we can see that the validation set accuracy is equal to 0.29 (**29%**)




At first, the plan was to implement several different values as our parameters:

- num_iters (number of iterations)
- batch_size (size of the batch)
- learning_rate (learning rate)
- learning_rate_decay (learning rate decay)
- reg. (regulation parameter)

We did not, however, achieve the expected results, therefore we had the idea of keeping the default given values and only change the hidden_size (number of hidden layers). The hidden_size values we chose were:

[50 : 75 : 125 : 250]

Among them, the best set of units in our hidden layers resulted to be **250**.

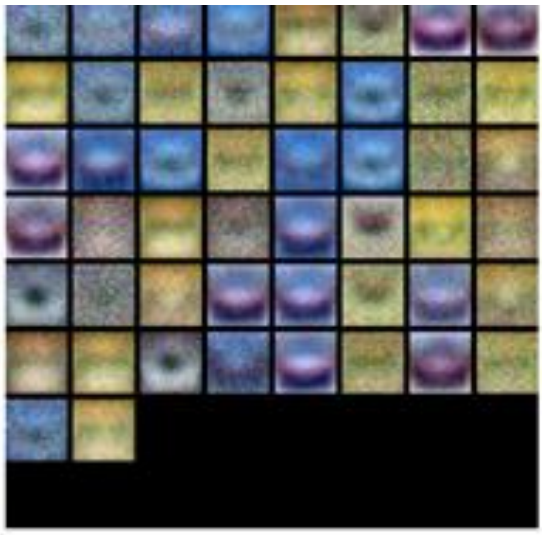
After that, we passed different values to our parameters, and achieved **29%** of val_acc (validation accuracy). Now that we achieved the optimal number of units to be used, we can optimize further by using the grid search for all the other 5 parameters:

- num_iters (number of iterations) = **600**
- batch_size (size of the batch) = **1024**
- learning_rate (learning rate) = **0.002**
- learning_rate_decay (learning rate decay) = **0.88**
- reg. (regulation parameter) = **0.2**

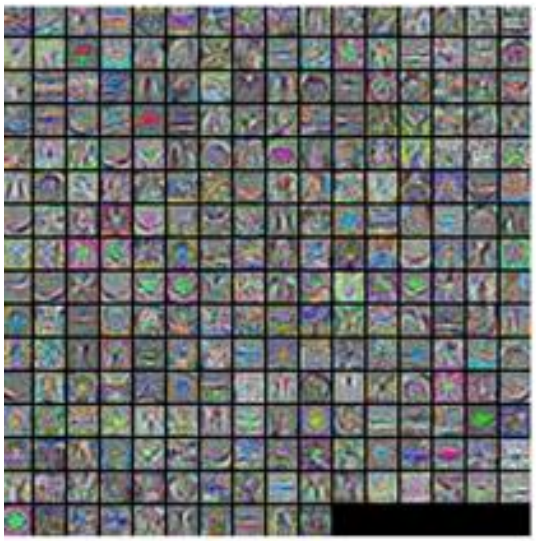
These are the best values for the parameters. Now we can implement them in our model, that we run for 2000 iterations. These lead us to the results below:

- Validation accuracy = **0.569**
- Test accuracy = **0.565**

The figure below is a representation of the Weights of the network:



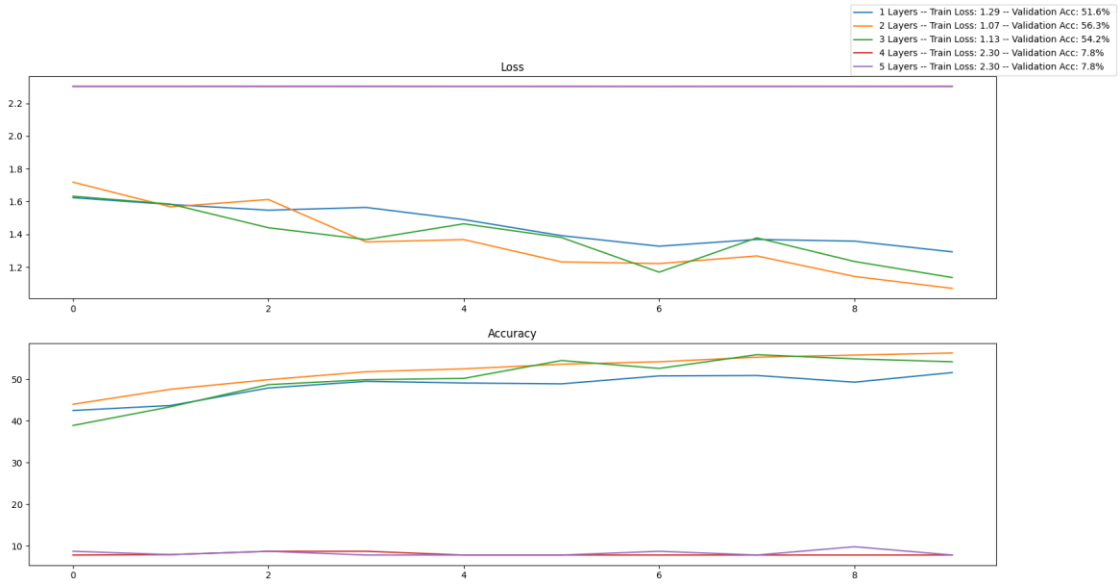
W also attach a representation of the Weights of the Best Network:



### *Question 4*

Now that you can train the two-layer network to achieve reasonable performance try the network depth to see if you can improve the performance. Experiment with networks of at least 2, 3, 4 and 5 layers of your chosen configuration. Report the training and validation accuracies for these models and discuss our observation. Run the evaluation on the test set with your best model and report the test accuracy.

**Answer:**




Looking at the results above, we see that by starting with one layer during the training phase we can obtain **1.29** as loss and **51.6%** as validation accuracy; this can be improved by adding an additional layer: in fact, as mentioned, the model with 2 layers resulted in a training loss of **1.07** and a validation accuracy of **56.3%**, which ended up being our best performance. The 3rd model with 3 layers also performed very well, having a training loss of **1.13** with a **54.2%** validation accuracy.

With the 4th and the 5th models we got some strange results: at the beginning we expected the model to get overfitted, but instead, after looking at the metrics (training loss = **2.30** and validation accuracy = **7.8%**) we have noticed that both models were underfitted. We even tried many times to change the number of units, but couldn't improve neither of them.

These are our final results, after running the evaluation set on our best model (the 2nd model):

```
Using device: cuda
Files already downloaded and verified
Accuracy of the network on the 1000 test images: 54.5 %
```