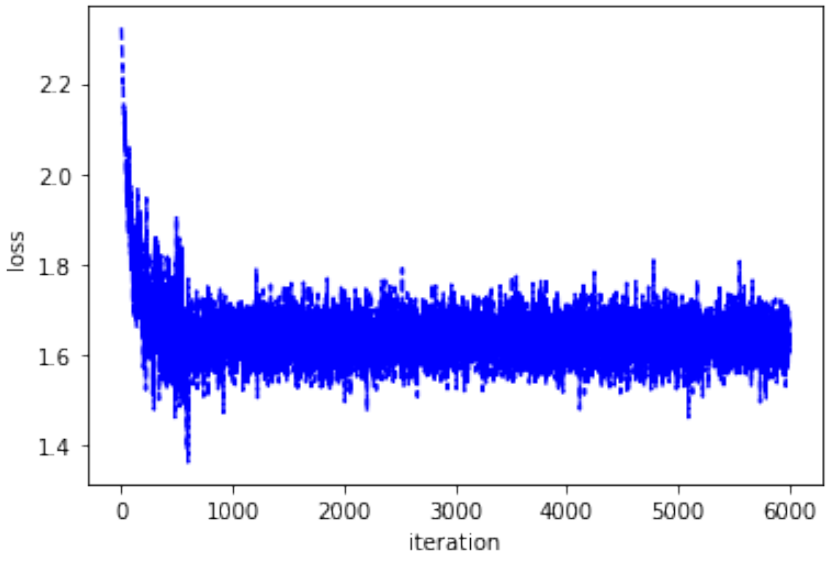
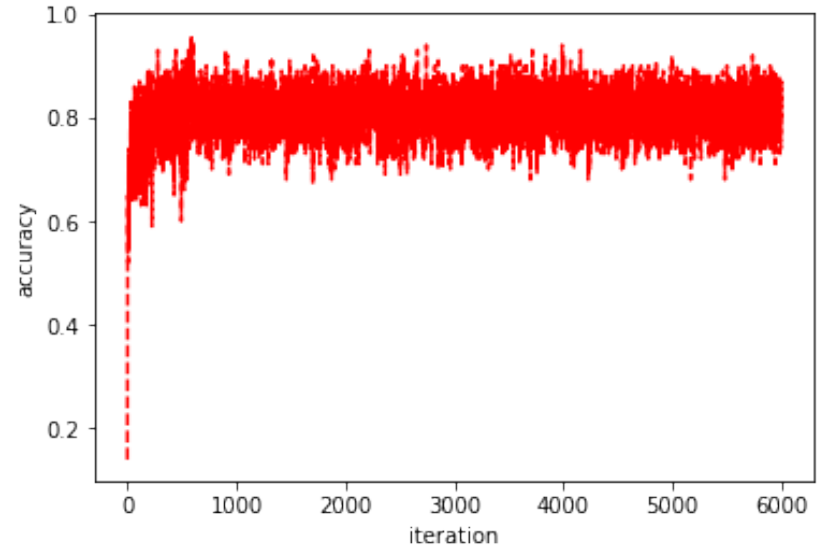
HW2 — Classification with Softmax and MLP

2. Softmax for MNIST Classification

A Softmax layer was trained on the MNIST dataset with the following hyper-parameters:
- batch_size = 100
- max_epoch = 10
- learning_rate = 0.01
- lamda = 0.5

Multiple learning rates and regularization values were experimented with and found that very high regularization (>1) negatively affected test accuracy. Learning rates less than 0.01 took longer to converge, as well as some very high learning rates (~1)

The final training and test performance is recorded below

	Training Data	Test Data
Loss	1.6794	n/a
		
Accuracy	0.7500	0.8208
		

3. MLP for MNIST Classification

3.1 MLP with Euclidean Loss and Sigmoid Activation

	Training	Validation	Test
Average Loss	0.3007	0.2902	n/a
Average Accuracy	0.7228	0.7680	0.7447

3.2 MLP with Euclidean Loss and ReLU Activation

	Training	Validation	Test
Average Loss	0.2342	0.2141	n/a
Average Accuracy	0.8093	0.8550	0.8291

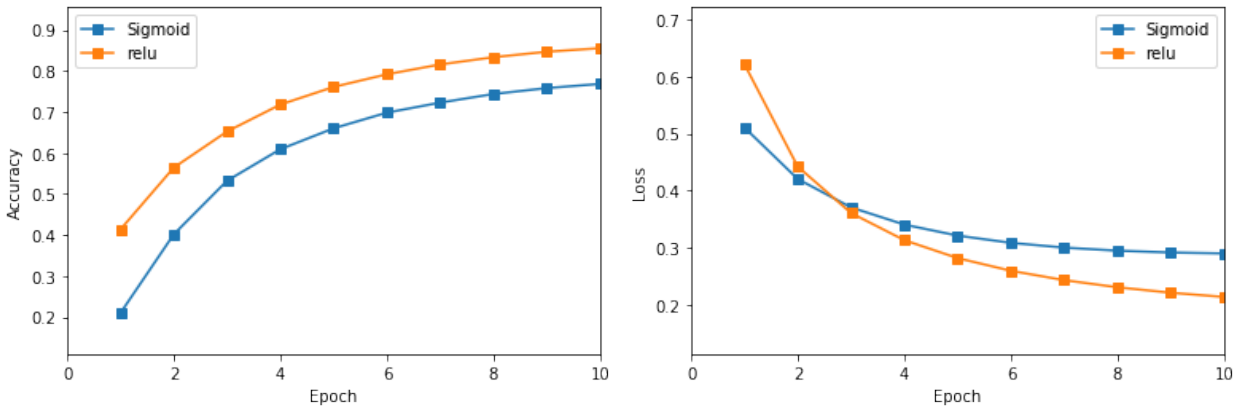
3.3 MLP with Softmax Cross-Entropy and Sigmoid Activation

	Training	Validation	Test
Average Loss	1.9105	1.8862	n/a
Average Accuracy	0.6807	0.7232	0.7016

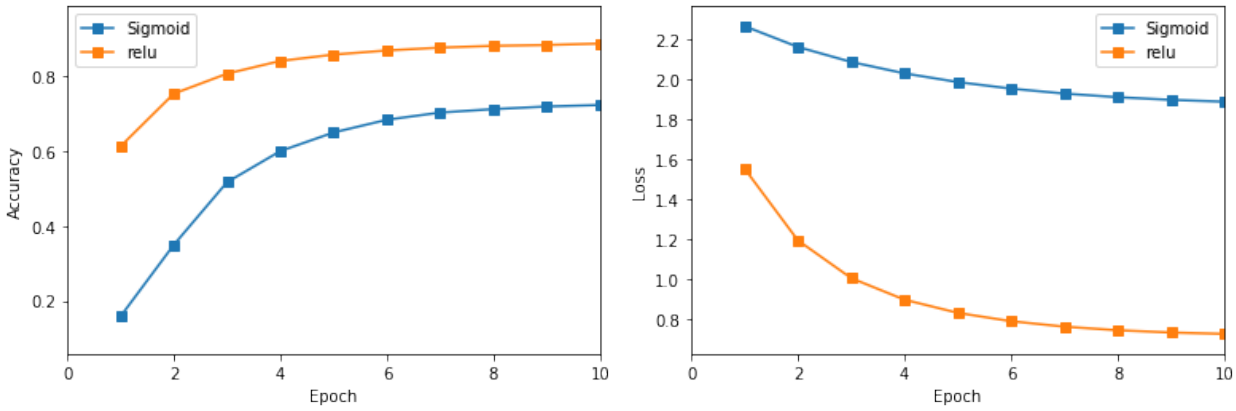
3.4 MLP with Softmax Cross-Entropy and ReLU Activation

	Training	Validation	Test
Average Loss	0.8055	0.7251	n/a
Average Accuracy	0.8456	0.8874	0.8595

Sigmoid vs ReLU



Based on experiments 3.1 and 3.2 it seems that ReLU is the better activation function. ReLU results in both a lower loss as well as higher overall accuracy. However, it seems Sigmoid converges in slightly fewer iterations.



The experiments 3.3 and 3.4 support the idea that ReLU outperforms the Sigmoid activation function.

Euclidean Loss vs Softmax Cross-Entropy

By comparing experiments 3.1 and 3.3, as well as experiments 3.2 and 3.4, we are able to see that the Softmax Cross-Entropy outperforms the Euclidean Loss depending on the type of activation function used. In experiments 3.1 and 3.3 a Sigmoid activation was used and the Euclidean Loss outperformed the Softmax Cross-Entropy, however in experiments 3.2 and 3.4 a ReLU activation was used and the Softmax Cross-Entropy outperformed the Euclidean Loss. The reason for this variance in performance may be due to the tendency for gradients to vanish in sigmoid and softmax layers. This could result in a network featuring both sigmoid and softmax layers to underperform.

4. 2-Layer MLP

I have decided to construct a 2 layer MLP featuring the following architecture:
FC Layer —> ReLU —> FC Layer —> ReLU —> FC Layer —> Softmax Cross-Entropy

The hyper-parameters used are listed below:
- batch_size = 100
- max_epoch = 10
- init_std = 0.01
- learning_rate_SGD = 0.001
- weight_decay = 0.1

The hyper-parameters affect performance in the same way as in part 2. Please refer to part 2 for more details.

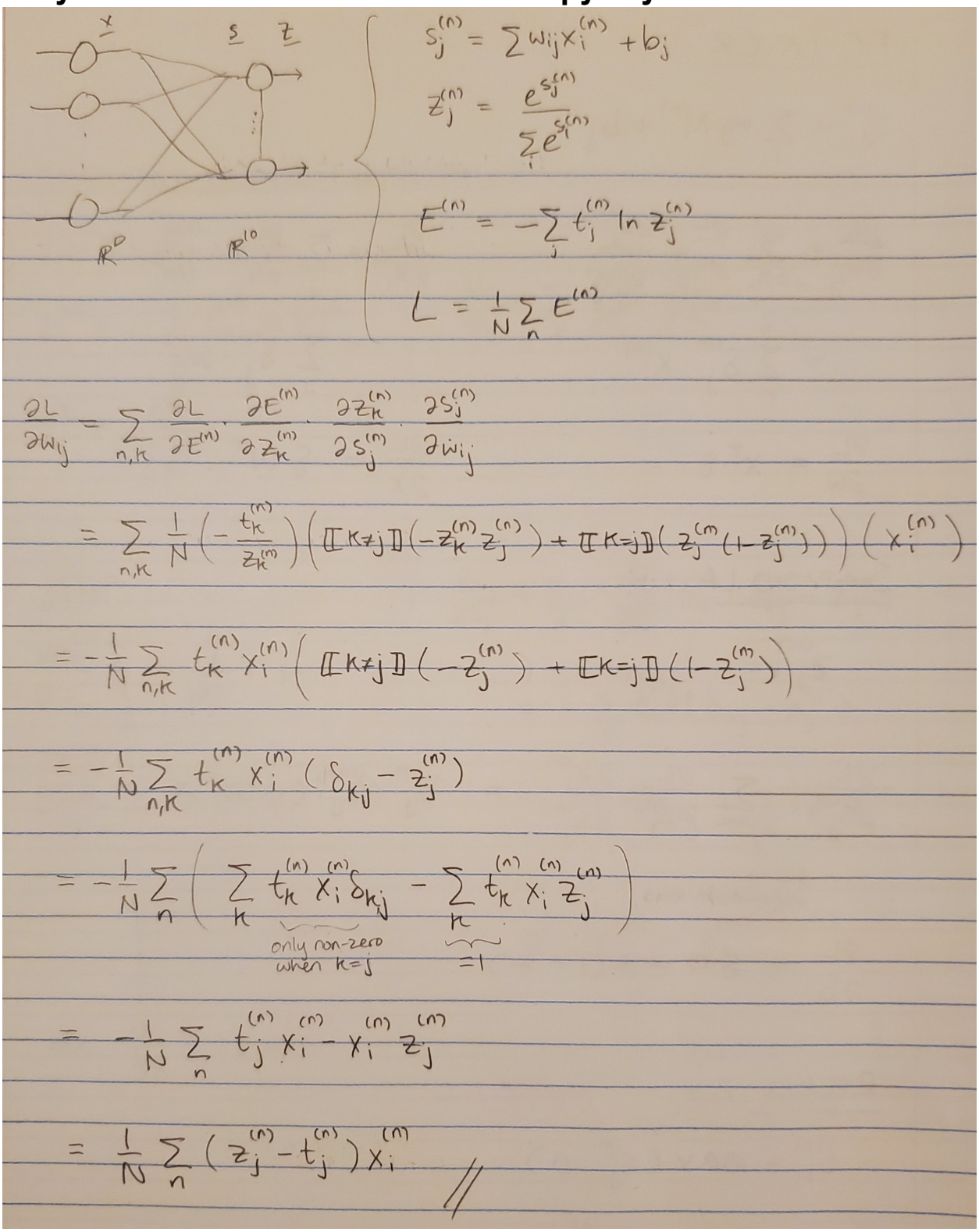
	Training	Validation	Test
Average Loss	0.7569	0.6797	n/a
Average Accuracy	0.8591	0.8988	**0.8728**

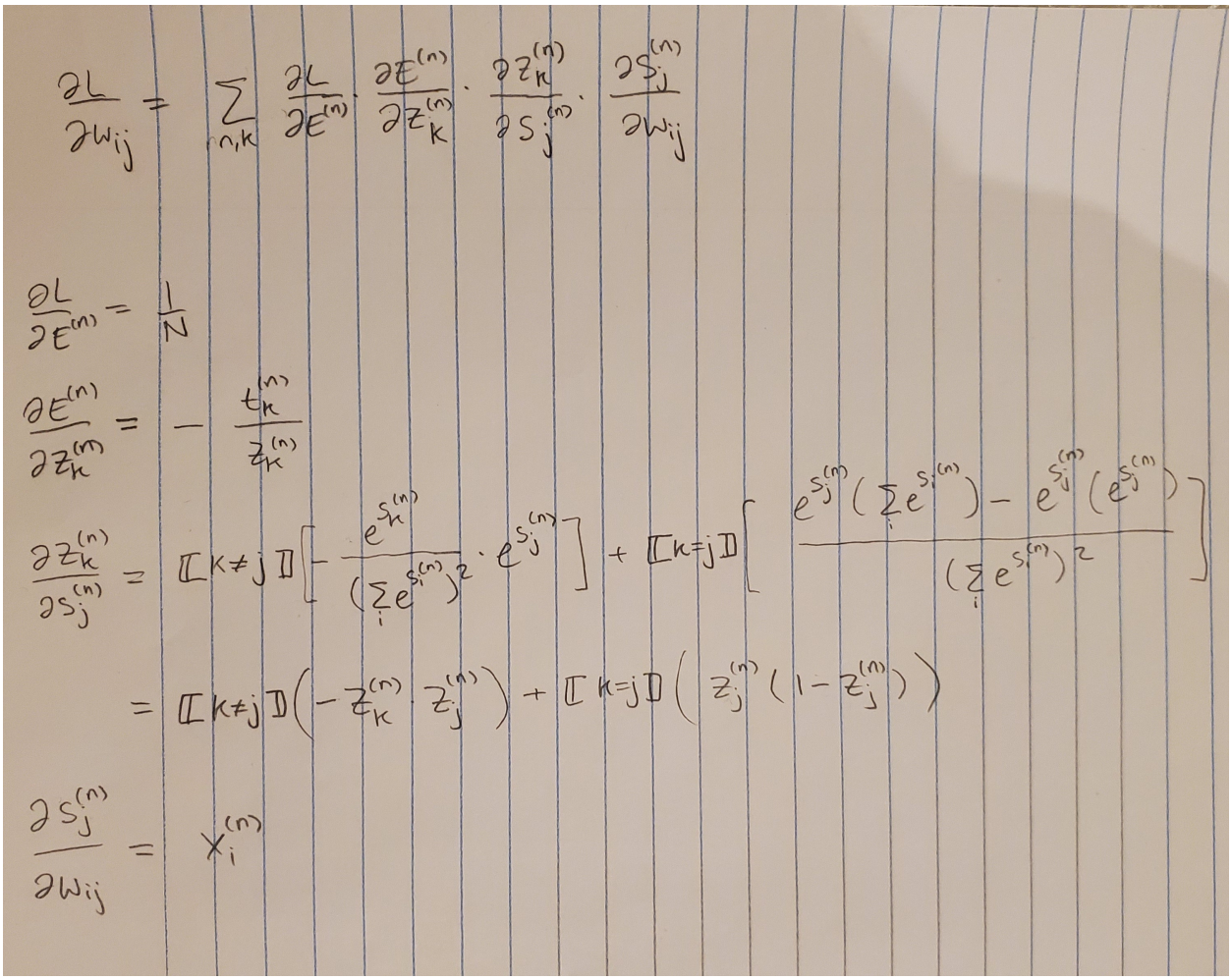
1 vs 2 Layer Networks

My constructed network is a simple extension of experiment 3.4 by adding a hidden layer. They use the same activations and loss function, with the only difference being the number of hidden layers. Comparing the performance of my network with experiment 3.4 suggests the deeper a network is, the better the accuracy and overall performance. This is because deeper networks allows them to learn features at various levels of abstraction.

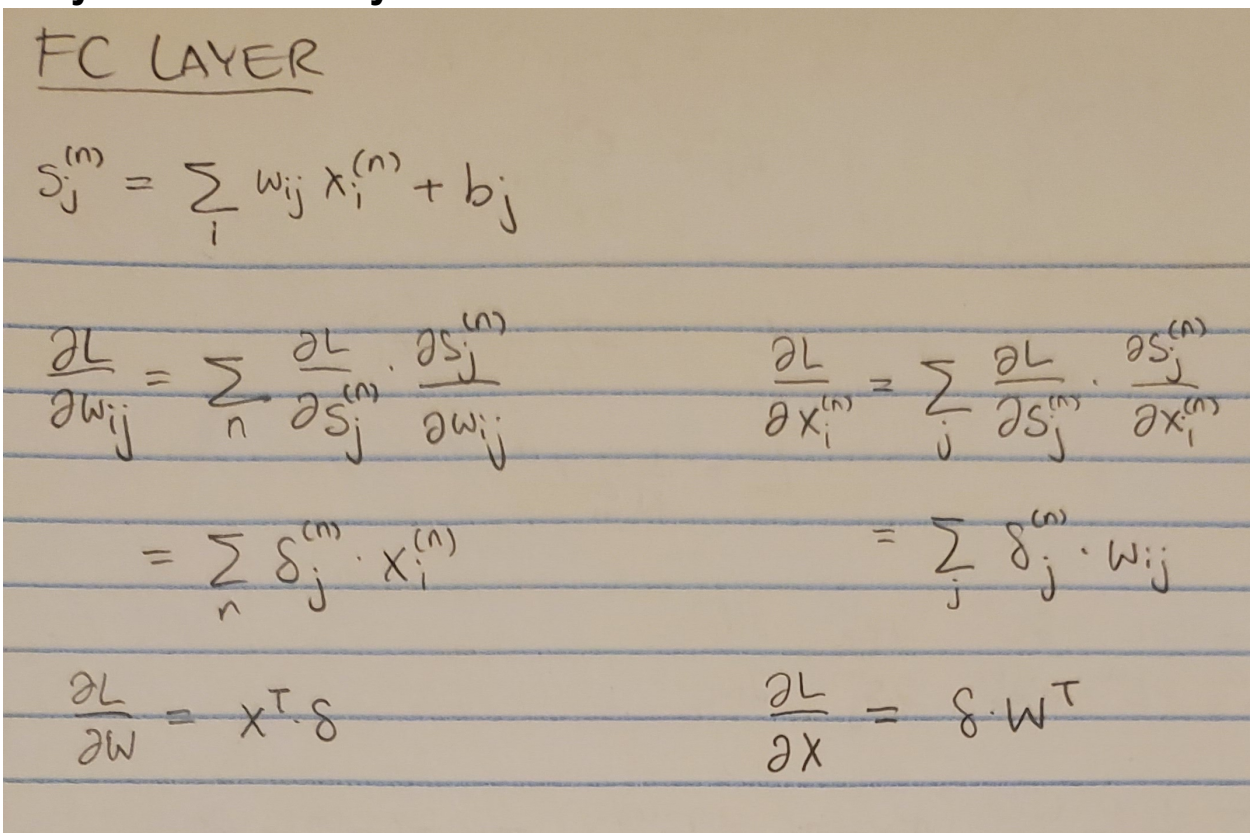
5. Appendix: Mathematical Derivations

Fully Connected Softmax Cross Entropy Layer

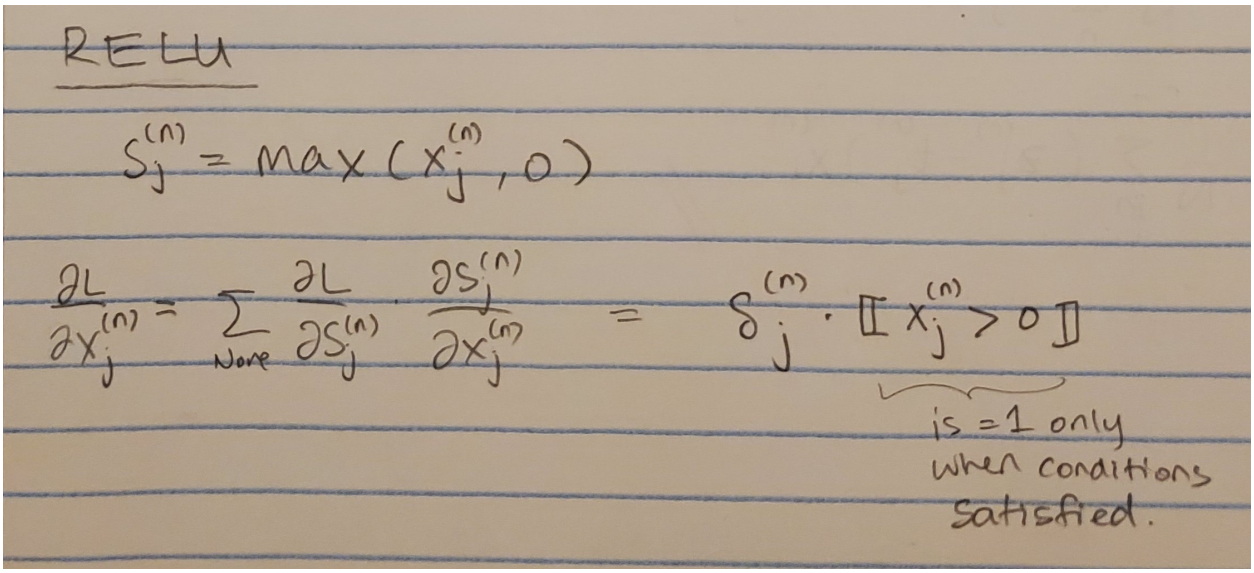



$\frac{\partial L}{\partial w_{ij}} = \sum_{n,k} \frac{\partial L}{\partial E^{(n)}} \cdot \frac{\partial E^{(n)}}{\partial z_k^{(n)}} \cdot \frac{\partial z_k^{(n)}}{\partial s_j^{(n)}} \cdot \frac{\partial s_j^{(n)}}{\partial w_{ij}}$
$\frac{\partial L}{\partial E^{(n)}} = \frac{1}{N}$
$\frac{\partial E^{(n)}}{\partial z_k^{(n)}} = -\frac{t_k^{(n)}}{z_k^{(n)}}$
$\frac{\partial z_k^{(n)}}{\partial s_j^{(n)}} = [\![k \neq j]\!] \left[-\frac{e^{s_k^{(n)}}}{(\sum_i e^{s_i^{(n)}})^2} \cdot e^{s_j^{(n)}} \right] + [\![k = j]\!] \left[\frac{e^{s_j^{(n)}} (\sum_i e^{s_i^{(n)}}) - e^{s_j^{(n)}} (e^{s_j^{(n)}})}{(\sum_i e^{s_i^{(n)}})^2} \right]$
$= [\![k \neq j]\!] \left(-z_k^{(n)} \cdot z_j^{(n)} \right) + [\![k = j]\!] \left(z_j^{(n)} (1 - z_j^{(n)}) \right)$
$\frac{\partial s_j^{(n)}}{\partial w_{ij}} = x_i^{(n)}$

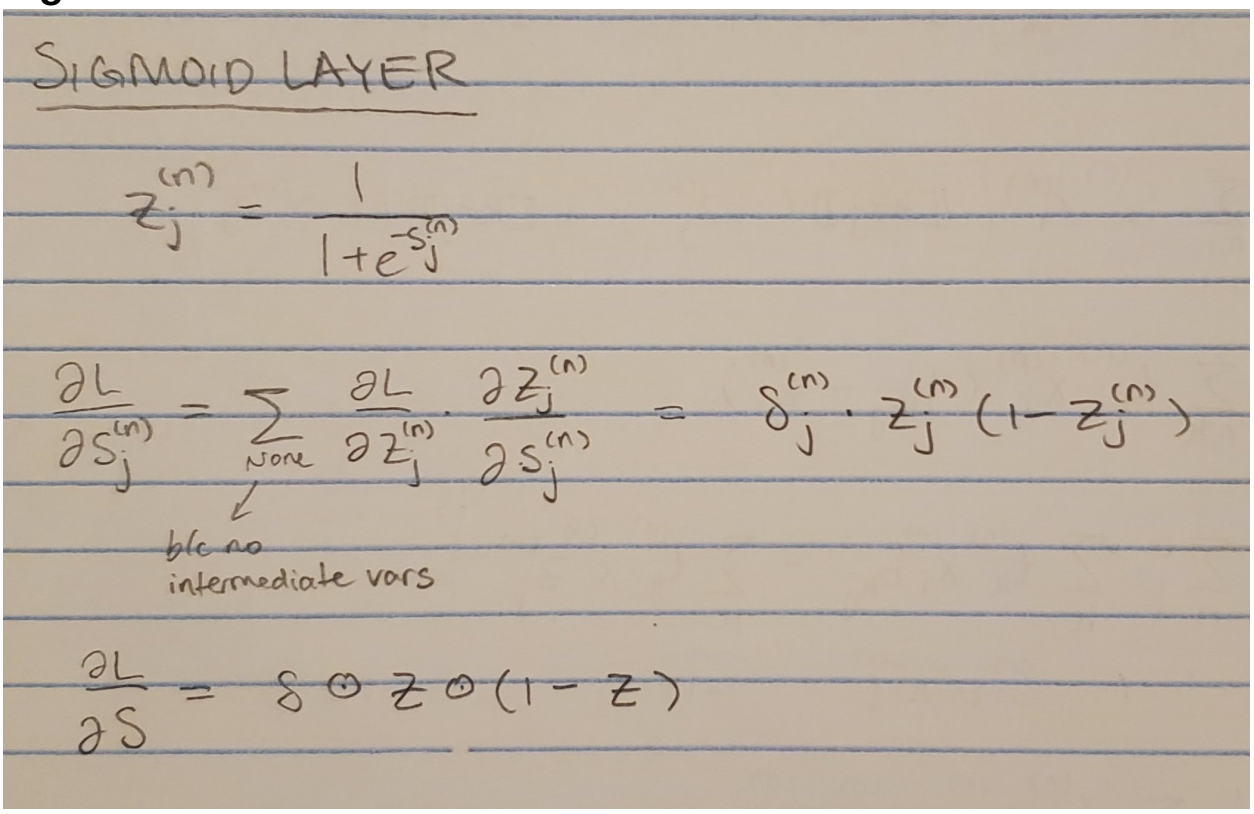
Fully Connected Layer



ReLU Activation



Sigmoid Activation



Softmax Cross-Entropy