ECE324 Assignment 2

Frederick Boyd

September 19, 2019

1 Solve by Inspection

i. The answer we discussed in class is not unique. An example of another set of weights and biases that works is:

$$\begin{bmatrix} w_0 & w_1 & w_2 \\ w_3 & w_4 & w_5 \\ w_6 & w_7 & w_8 \end{bmatrix} = \begin{bmatrix} 2 & -2 & 2 \\ -2 & 2 & -2 \\ 2 & -2 & 2 \end{bmatrix}$$

$$b = -8$$

ii. $2^9 = 512$ unique inputs.

iii. Assuming we are always looking for an 'x' that spans the entire grid, then the solution scales to a 4x4 and an NxN problem. This would work because the weights would just need follow a similar pattern. The weights where the 'x' is expected to be would have a value of 1 and all other weights would be -1. The bias would scale similarly.

iv. No, because this would involve finding single-neuron parameters that have to activate the neuron in multiple scenarios. This complicates the problem significantly, making it difficult to find these parameters by hand.

2 Outputs to Hand In

2.1 Investigating Effect of Parameters on Accuracy

The effect of the following parameters were investigated to see what effect they have on the training and validation accuracy. Values of parameters other than the one being investigated were kept constant. Those default values are:

- Number of Epochs: 50
- Learning Rate: 0.2
- Activation Function: Linear
- Seed: 1

Effects of changing:

1. **Number of Epochs:** See Table 1.
2. **Learning Rate:** See Table 2.
3. **Activation Function:** See Table 3.

   Using the default values in 2.1, it appears that ReLU is the best activation function. Compared to the linear and sigmoid activation functions, the ReLU function offers a good balance between the linear function's simplicity and the sigmoid's complexity.

Table 1: Effect of the number of epochs on training and validation accuracies.

# of Epochs	Final Training Accuracy	Final Validation Accuracy
5	0.795	0.85
10	0.95	0.8
25	0.975	0.9
50	0.975	0.9
75	0.98	0.9
100	0.98	0.9
500	0.98	0.9
1000	0.98	0.9

Table 2: Effect of the learning rate on training and validation accuracies.

Learning Rate	Final Training Accuracy	Final Validation Accuracy
0.01	0.4	0.25
0.05	0.965	0.85
0.1	0.975	0.9
0.25	0.985	0.9
0.5	0.665	0.35
0.75	0.665	0.35
1	0.665	0.35

4. **Seed:** See Table 4.

   The training and validation accuracies differ slightly because the seed determines what random values the weights and bias initialize to. This, in turn, impacts the accuracies because the final weights and bias are directly determined from the random initial values.

   NOTE: When investigating the effect of various seeds on accuracy, the learning rate was kept constant at **0.25**, rather than the default value specified in section 2.1.

2.2 Best Parameters

The following parameters generated the highest training and validation accuracies. These values were determined empirically.

- Learning Rate: 0.8
- Activation Function: Sigmoid
- Number of Epochs: 500
- Seed: 1
- Training Accuracy: 0.99
- Validation Accuracy: 1.0

Another set of parameters were empirically obtained that generate high training and validation accuracies. However, these are rather extreme values. More specifically, the learning rate is typically a value between 0 and 1, which is not the case for the following set of values. This second set of parameters is listed because while it achieves similar accuracies compared to the parameters above in fewer epochs, it requires the use of extreme values which may not be considered valid within the field of maching learning.

- Learning Rate: 1000

Table 3: Effect of the activation function on training and validation accuracies.

Activation Function	Final Training Accuracy	Final Validation Accuracy
Linear	0.975	0.9
ReLU	0.665	0.65
Sigmoid	0.98	0.85

Table 4: Effect of the activation function on training and validation accuracies. The learning rate for these values was kept constant at 0.25, rather than the default value specified in section 2.1.

Seed	Final Training Accuracy	Final Validation Accuracy
1	0.985	0.9
5	0.99	0.9
10	0.985	0.9
69	0.99	0.85
420	0.99	0.85
69 420	1.0	0.85
1 000 000	0.99	0.85

- Activation Function: Sigmoid
- Number of Epochs: 5
- Seed: 1
- Training Accuracy: 0.99
- Validation Accuracy: 1.0

2.3 Written Questions

1. **Low Learning Rate**

   See Figures 1, 2, and 3 for plots of loss, accuracies, and a visualization of final weights, respectively. A learning rate of 0.01 was chosen. All other parameters were kept at their default values. The training and validation accuracies at the end of 50 epochs is significantly below 1.0, as seen in Figure 2. Although the accuracy is gradually increasing during the last 20-30 epochs, it does so at a very slow rate. A similar issue can be seen with the plot of training and validation losses in Figure 1. Although the average loss is decreasing, it does so at a rate that is not fast enough to reach approximately zero, even after 50 epochs. Since the learning rate is used to calculate the rate at which parameters should be adjusted, these issues are a result of the gradient descent calculations being scaled down too much. This, in turn, results in the single neuron not being able to minimize losses and achieve a reasonable accuracy within 50 epochs.

2. **High Learning Rate**

   See Figures 4, 5, and 6 for plots of loss, accuracies, and a visualization of final weights, respectively. A learning rate of 0.5 was chosen. All other parameters were kept at their default values. This learning rate resulted in exponentially increasing losses and oscillating accuracies. Although the visualization of weights using the `dispKernel` function looks good, a further investigation of the weights reveals that they are on the order of magnitude of 10^{22}. The poor accuracies and ever-increasing losses are the result of the weights being adjusted too drastically, resulting in oscillations as the algorithm attempts to minimize losses. These oscillations can be seen in the training and validation accuracies in Figure 5.

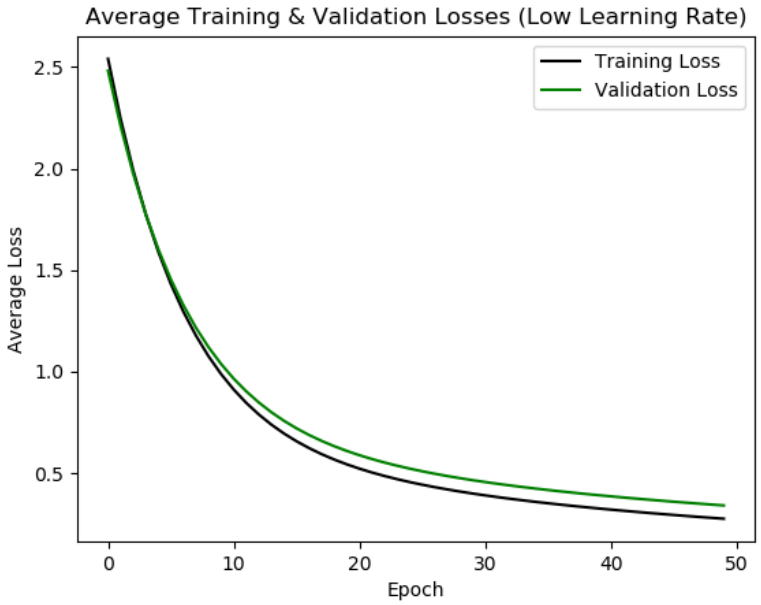

Figure 1: Average loss against the current epoch using a low learning rate.

3. **Good Learning Rate**

   See Figures 7, 8, and 9 for plots of loss, accuracies, and a visualization of final weights, respectively. A learning rate of 0.15 was chosen. All other parameters were kept at their default values.

4. **Plots for Three Activation Functions**

   Plots comparing the linear, ReLU, and sigmoid activation functions. Parameters were kept at their defaults except for the number of epochs, which was increased to 100.

   (a) **Linear Activation Function**

   See Figures 10, 11, and 12 for plots of loss, accuracies, and a visualization of final weights, respectively.

   (b) **ReLU Activation Function**

   See Figures 13, 14, and 15 for plots of loss, accuracies, and a visualization of final weights, respectively.

   (c) **Sigmoid Activation Function**

   See Figures 16, 17, and 18 for plots of loss, accuracies, and a visualization of final weights, respectively.

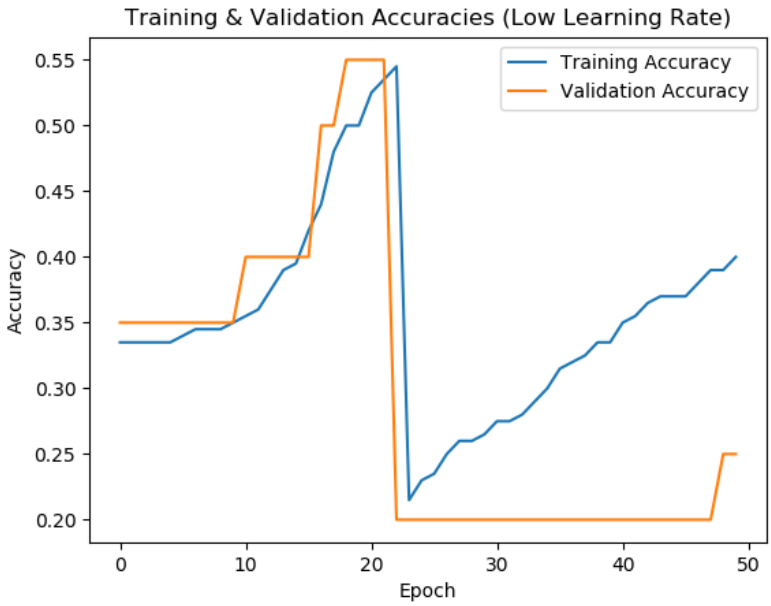

Figure 2: Training and validation accuracies against the current epoch using a low learning rate.

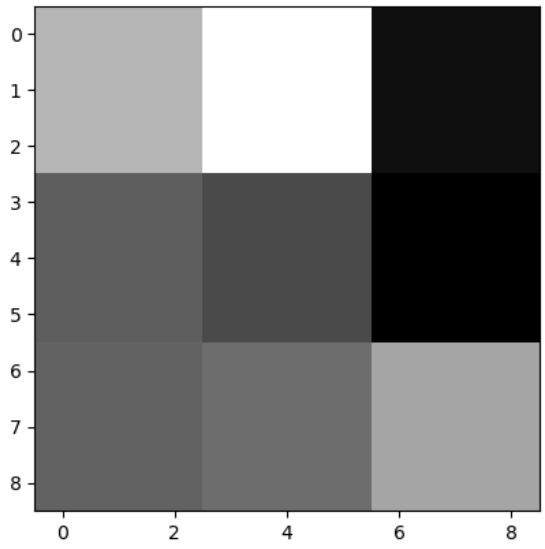

Figure 3: Visualization of final weights with the `dispKernel` function using a low learning rate.

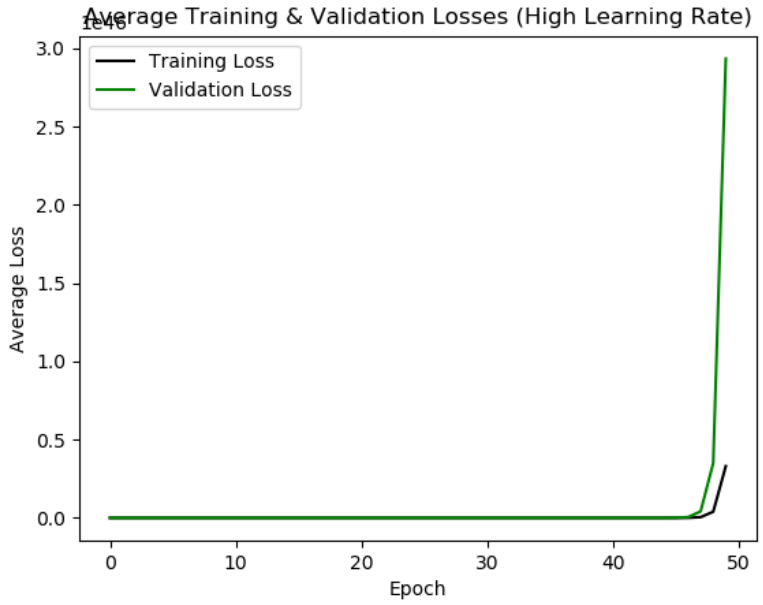

Figure 4: Average loss against the current epoch using a high learning rate.

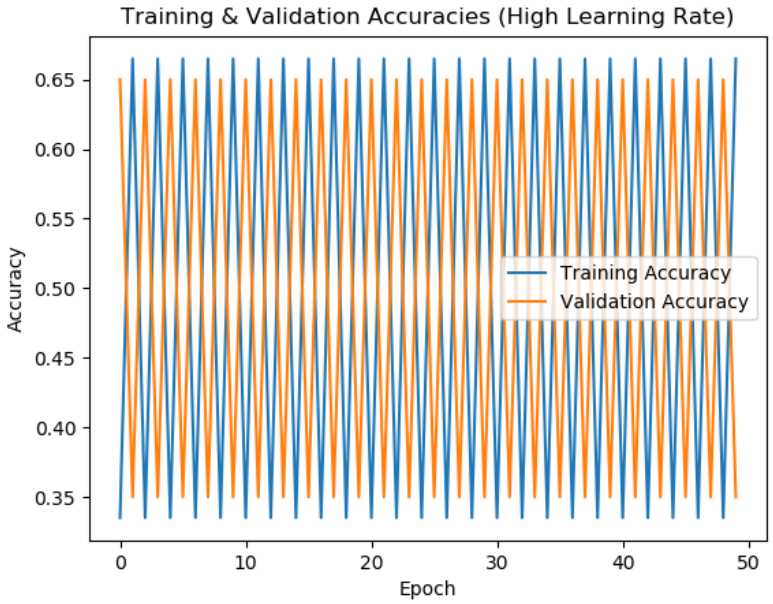

Figure 5: Training and validation accuracies against the current epoch using a high learning rate. The oscillations can be attributed to the algorithm adjusting weight values too drastically as it attempts to minimize losses.

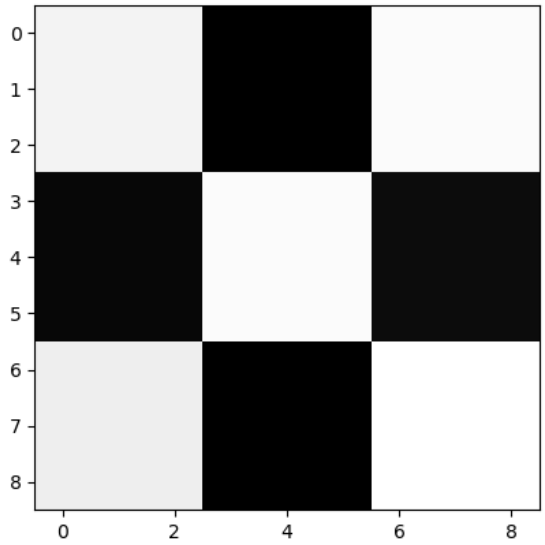

Figure 6: Visualization of the final weights with the `dispKernel` function using a high learning rate. The final weights are on the order of magnitude of 10^{22}.

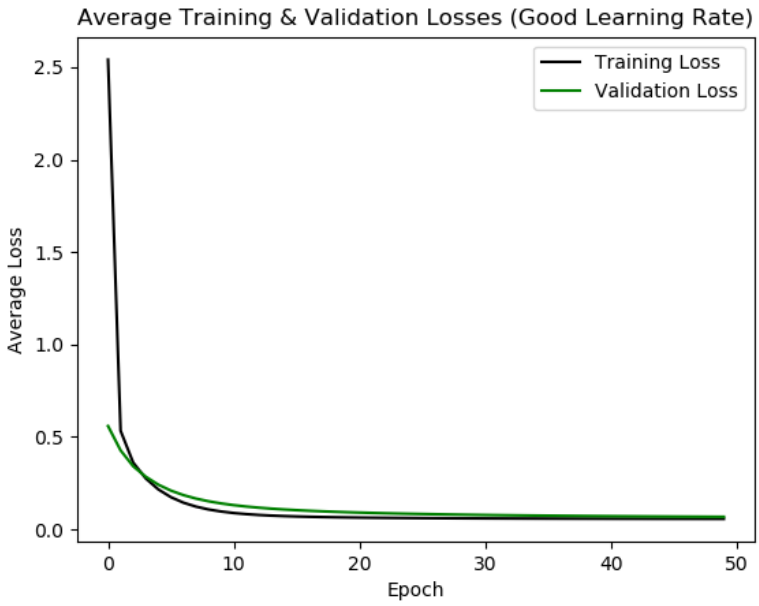

Figure 7: Average loss against the current epoch using a good learning rate.

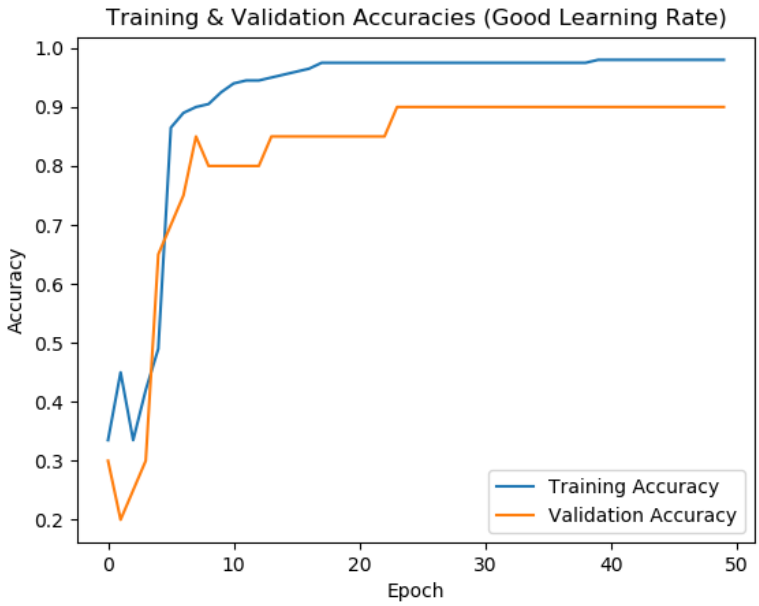

Figure 8: Training and validation accuracies against the current epoch using a good learning rate.

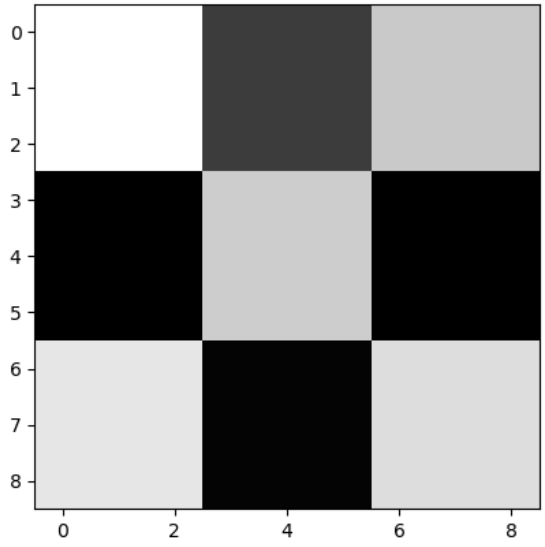

Figure 9: Visualization of the final weights with the `dispKernel` function using a good larning rate.

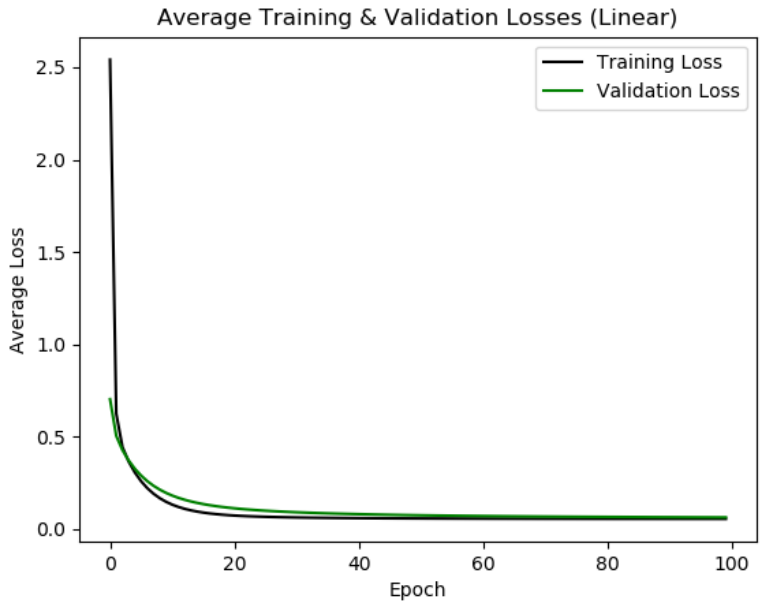

Figure 10: Average loss against the current epoch using the linear activation function.

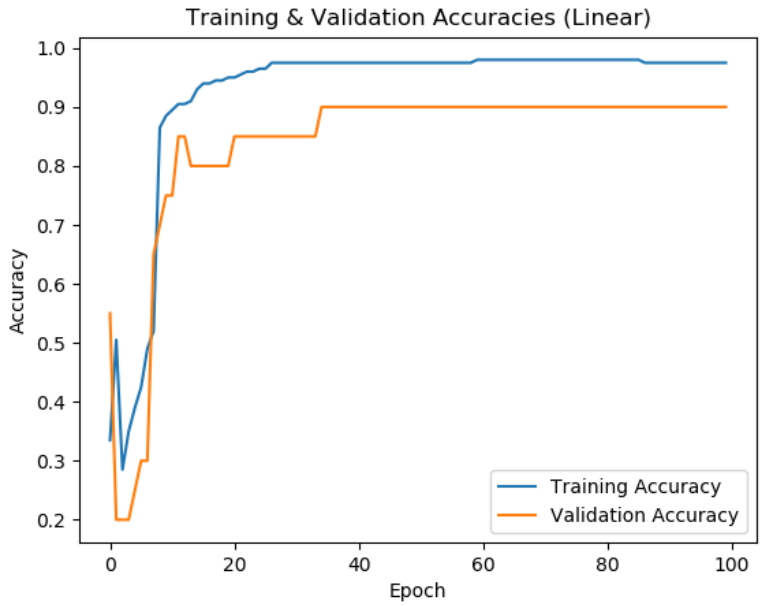

Figure 11: Training and validation accuracies against the current epoch using the linear activation function.

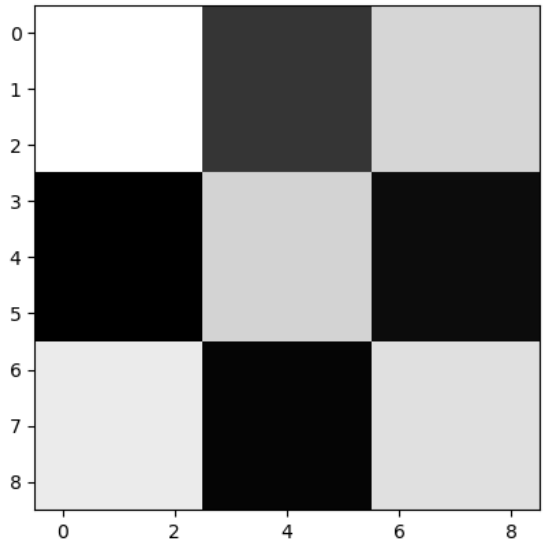

Figure 12: Visualization of final weights with the `dispKernel` function using the linear activation function.

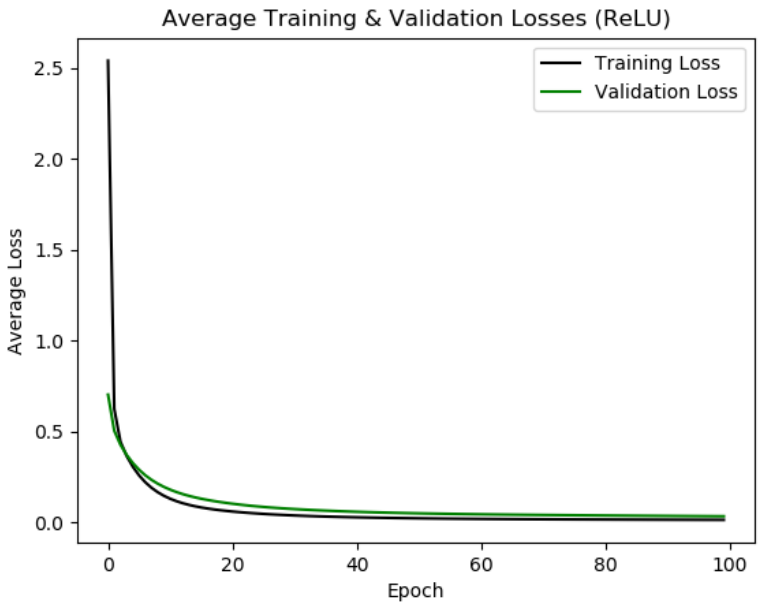

Figure 13: Average loss against the current epoch using the ReLU activation function.

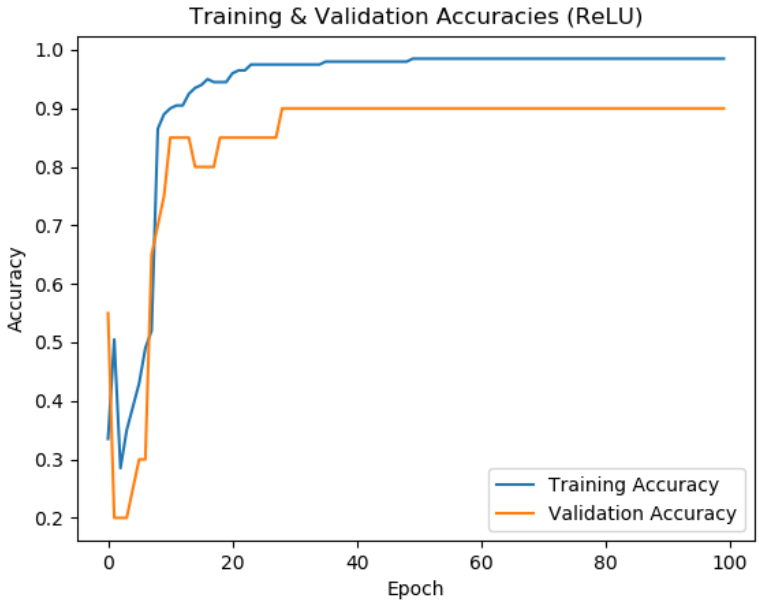

Figure 14: Training and validation accuracies against the current epoch using the ReLU activation function.

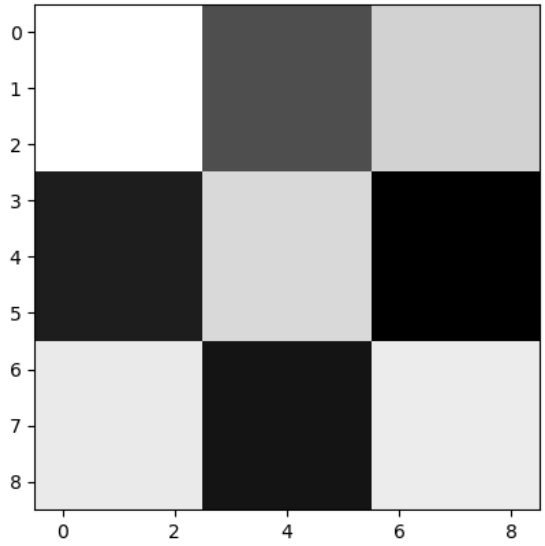

Figure 15: Visualization of the final weights with the `dispKernel` function using the ReLU activation function.

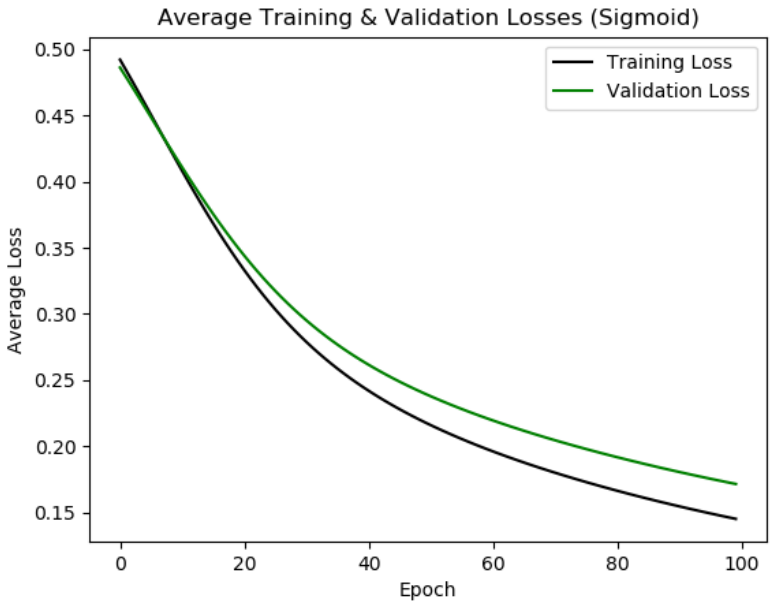

Figure 16: Average loss against the current epoch using the sigmoid activation function.

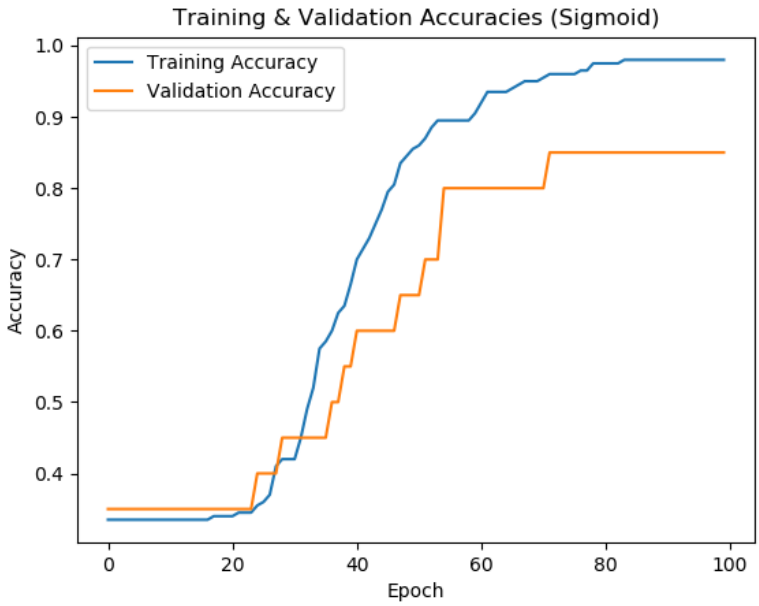

Figure 17: Training and validation accuracies against the current epoch using the sigmoid activation function.

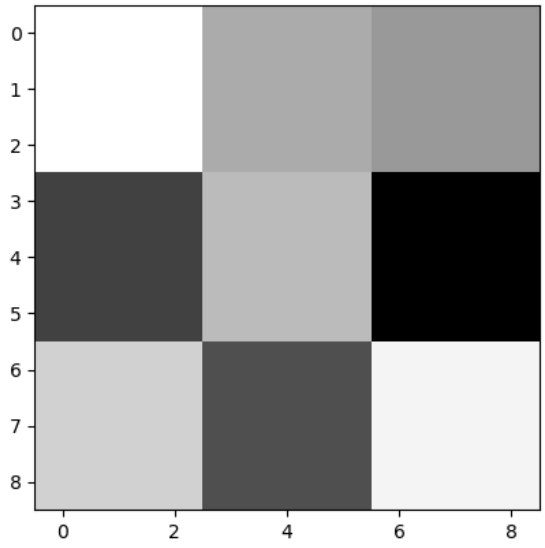

Figure 18: Visualization of the final weights with the `dispKernel` function using the sigmoid activation function.