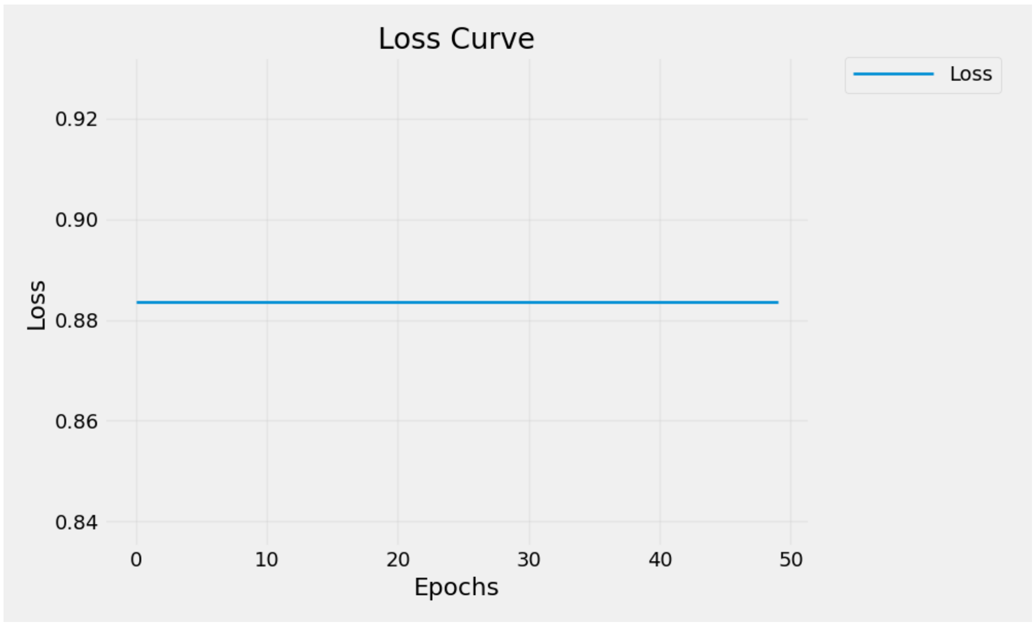
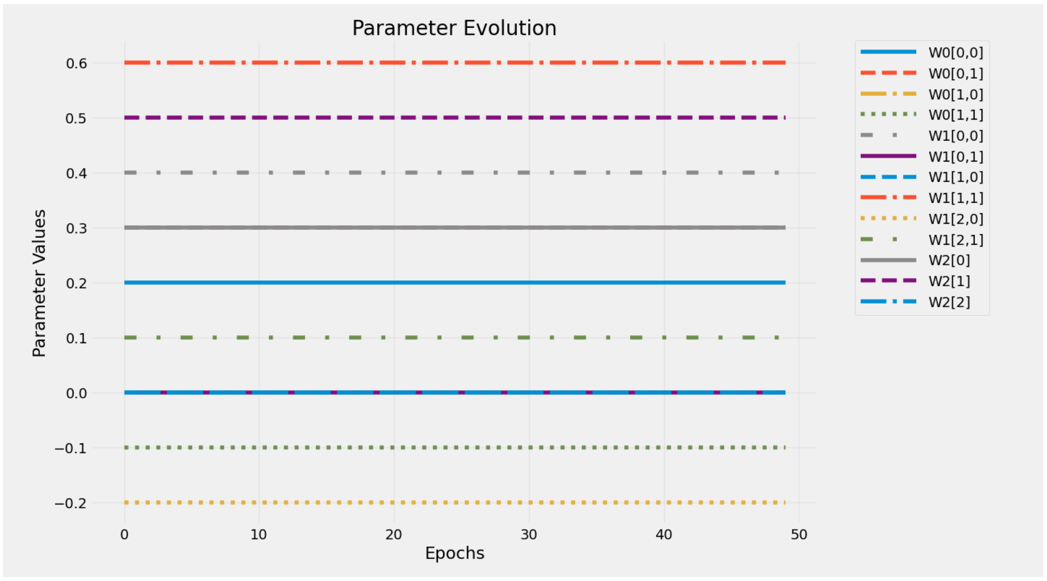
i) Initial Run: Run the provided network as-is. Document the loss curve and explain why the network fails to converge.

The algorithm fails to converge initially because we are only marking the zero value weights as trainable. Since there is no bias term being used, and ReLU(0, max(0)) = 0, these values will remain zero throughout the network. The gradient will also be zero due to this, so no learning can occur. Learning rate also appears to be a little low.



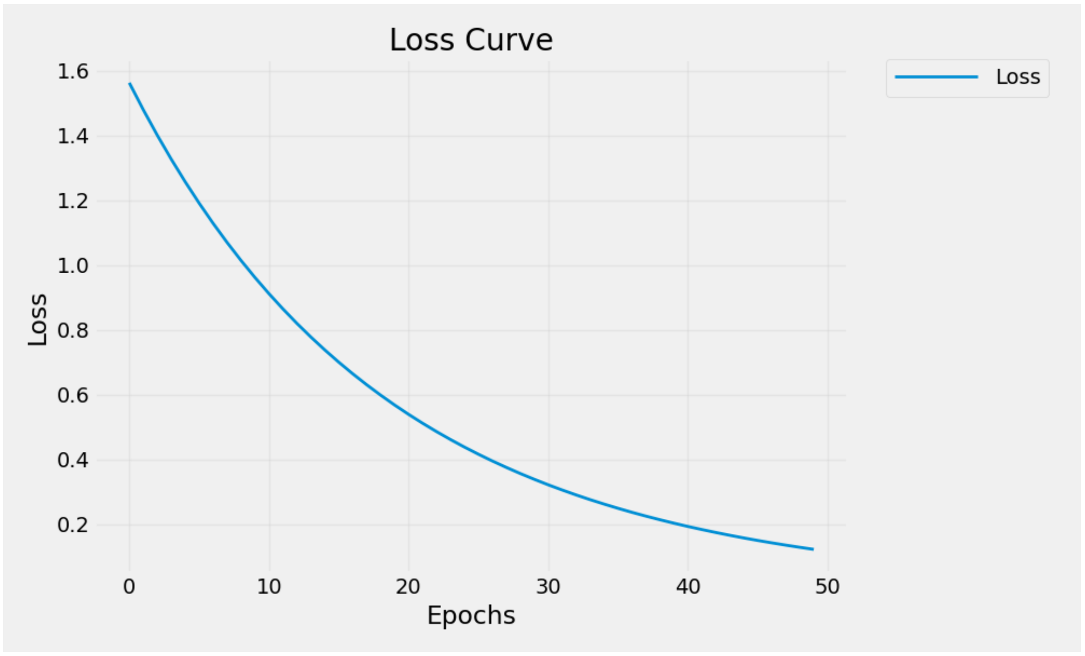


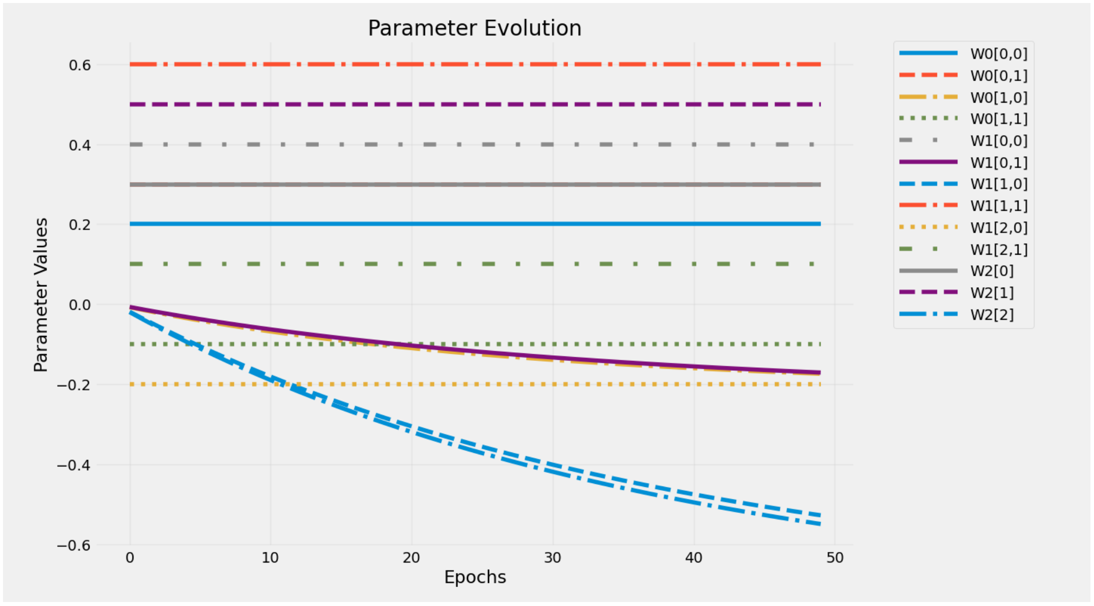
ii) Modify and Observe: Make one modification at a time. Document each noteworthy change, including plots showing the effects on loss and parameter evolution.

1. Adding bias

We can see that simply adding a 1 for the bias, so we do not have dead neurons and gradients everywhere, allows the network to improve its loss significantly. Only the selected weights from the masks are able to be learned from, however.

Epoch 50: Loss = 0.12237604707479477, y_pred = 1.3498228788375854

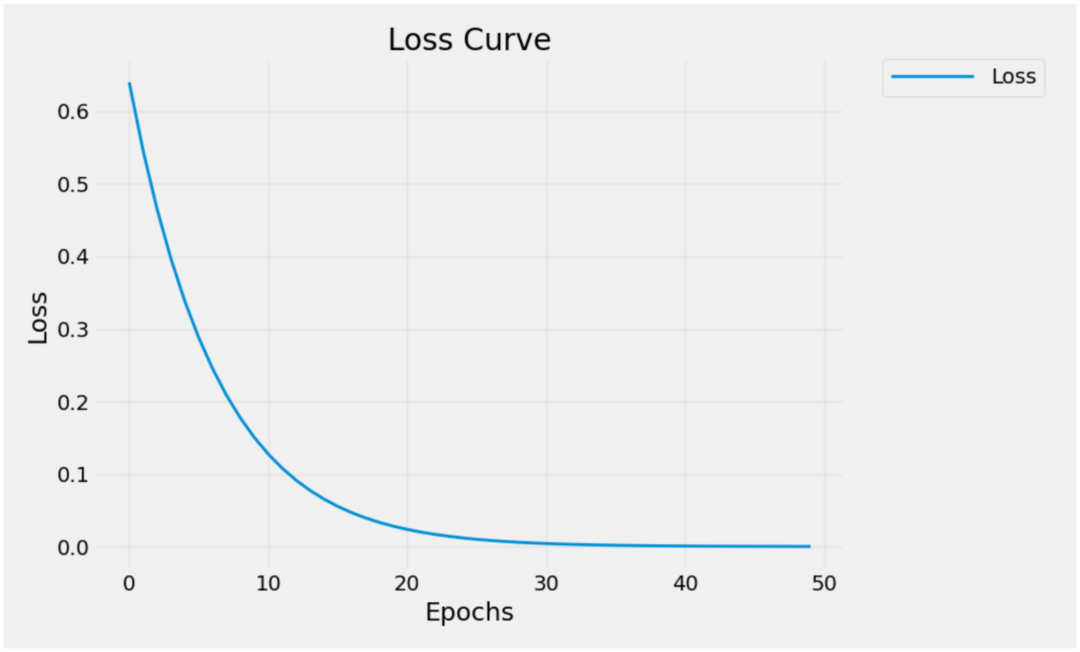


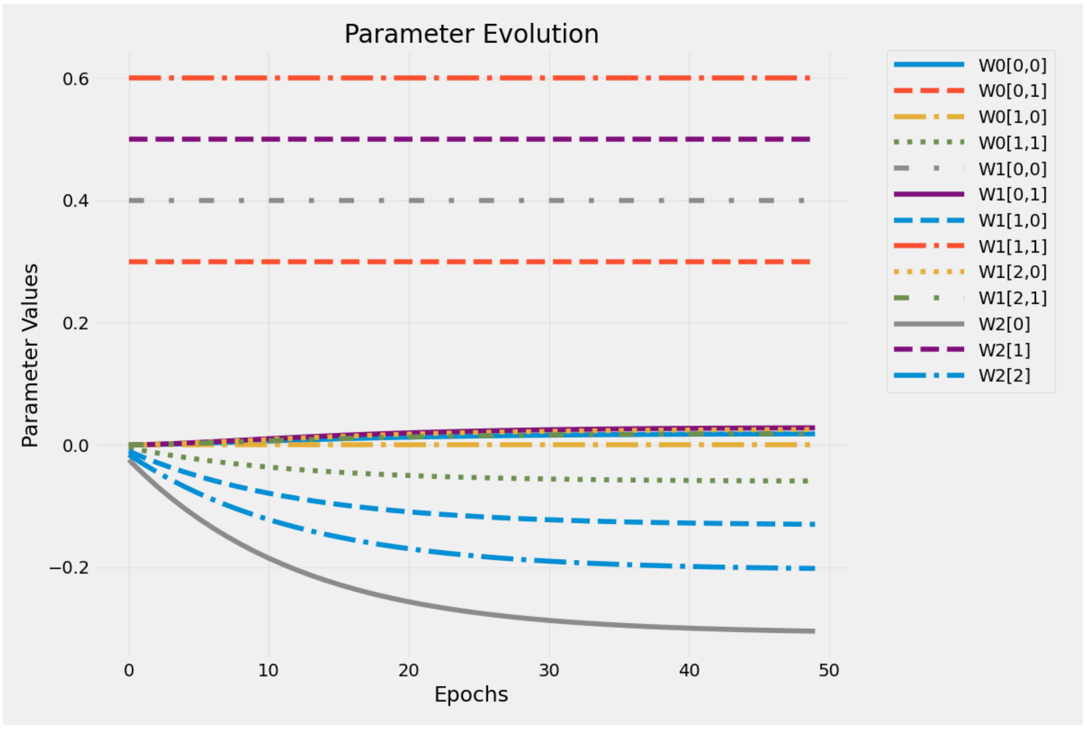


2. Marking random weights as trainable

Since we're only using the zero weights as trainable, it would be a good idea to randomly mark some weights as trainable because we are ignoring the majority of them, and having the same starting value for each trainable weight may not be the best idea.

Epoch 50: Loss = 0.00014321059279609472, y_pred = 1.0119670629501343

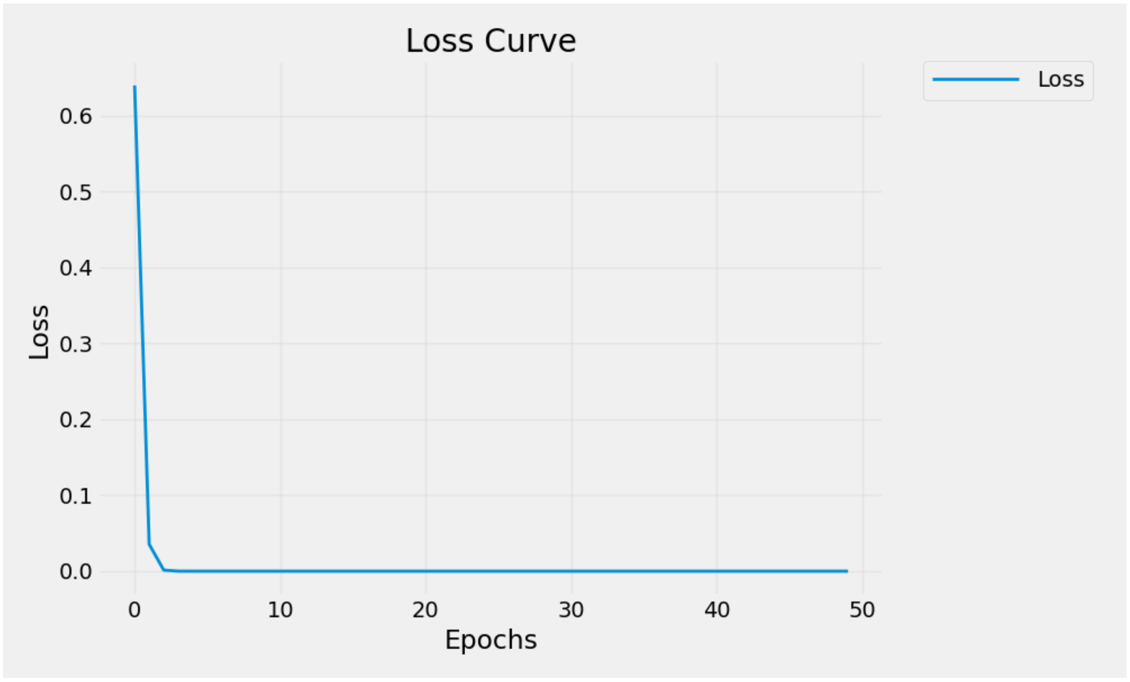


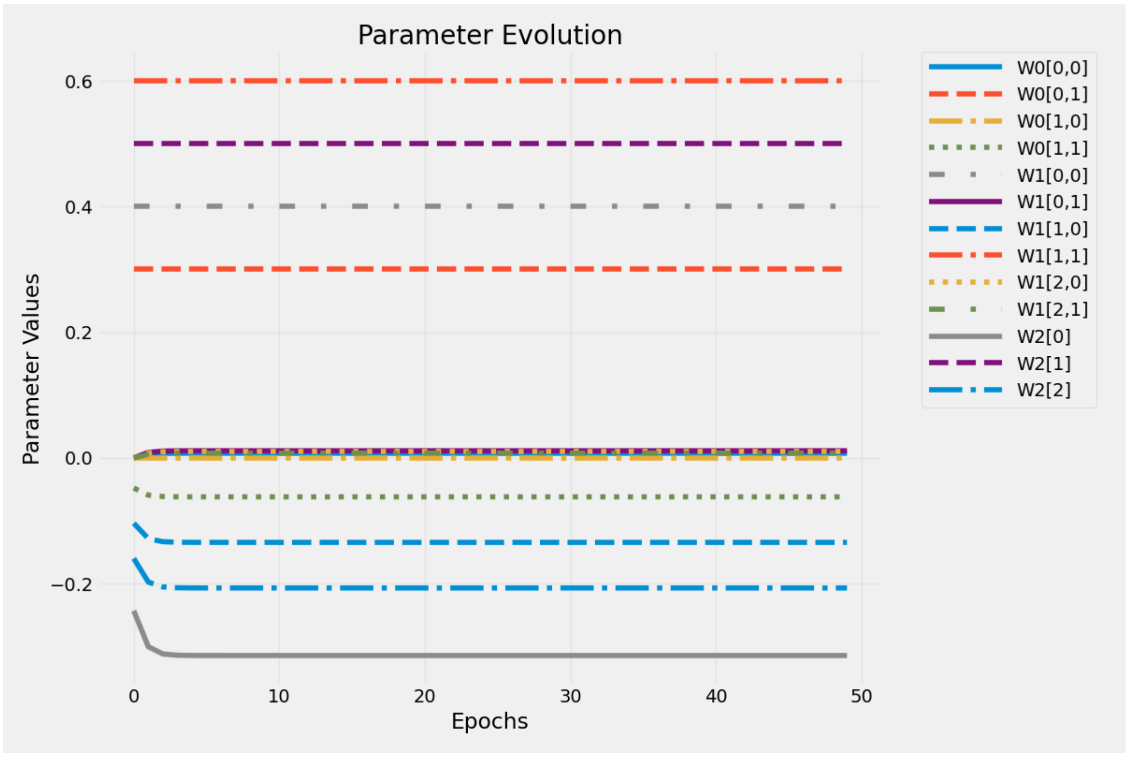


3. Learning rate

Tweaking the learning rate can sometimes help. In this case, it helped the network to converge to the optimal solution very quickly. It was updated from 0.01 to 0.1

Epoch 50: Loss = 0.0, y_pred = 1.0





iii) Analyze: Compare the different modifications. Which ones improved convergence? Discuss how the initial zero values for weights led to degenerate learning, and explain how your modifications addressed these issues.

Adding bias, incrementing learning rate, and randomly marking random weights as trainable improved convergence. I tried using different activation functions, but they did not seem to help. Initial zero values led to degenerate learning because we only used zero value weights as trainable values. This, coupled with the lack of a bias term and ReLU activation, led to degenerate learning. My modifications addressed these by marking more of the weights as trainable (especially the nonzero ones) and adding a bias. These two changes help us produce nonzero output from each layer.