Explanation of Code Design Choices

Neural Network stored in custom class:

The neural network was created, using a custom class. This was done, to allow for easy and intuitive access to specific weights, biases, and neurons. Additionally, there are several repetitive operations (such as calculating gradient, updating weights/biases, calculating output) that must be performed, which can be executed easily as methods for the custom class.

The alternative was creating many different variables to store the all the necessary information, which would require a lot more memorisation, and generally be a lot clunkier.

Partial derivatives stored in a way that mimics neural networks shape:

The partial derivatives of cost with respect to the weights/biases, aren't just stored in one big column vector. The partial derivatives for the weights, are stored in a list of 3 matrices, just like the actual weights, and partial derivatives for the biases, are stored in a list of 3 vectors, just like the biases. (Biases are stored as columns, whereas bias derivatives are stored as rows, to properly represent the mathematical definition, so transposing is required, when zipping the two together)

So they're stored in a way that perfectly mimics the shape of the neural network. This allows for easy access to any particular partial derivative we want. If we threw them all in one big column vector, it would be impossible to access a particular partial derivative out of the tens of thousands that exist.

Why was stochastic gradient descent used:

To help decrease how long it takes to train the neural network, stochastic gradient descent was implemented. So basically, in normal gradient descent, for every epoch/lap of training, the gradient of cost "∇C" is calculated (ALL training instances are apart of cost here), and then the weights/biases are updated according to ∇C.

In stochastic gradient descent, for every epoch/lap of training, all the training instances are grouped into small, randomly sampled batches. Then, the gradient of cost for each one of these batches is calculated "∇C_k", then the weights/biases are updated according to ∇C_k. Of course, ∇C_k is much faster to calculate than ∇C, but there are a lot of ∇C_k's compared to ∇C, to the point where the overall number of training instances involved is the same.

HOWEVER, with ∇C (normal gradient descent), the weights/biases are only updated once every epoch, when ∇C has been calculated, which takes a while. Whereas with ∇C_k (stochastic), the weights/biases are calculated multiple times every epoch (equal to the number of batches), at the end of every ∇C_k, which is much faster. So the general idea with stochastic gradient descent, is the weights/biases are updated much more rapidly compared to normal gradient descent, so convergence to the local minimum takes a lot less time.