Task 1

In Figure 1 we see the convergence graph for three different learning rates. This is a typical example of one value being too low, one adequate and one too high. The convergence for the lowest learning rate (blue) is too slow since we see that a higher learning rate (orange) converges faster. The highest learning rate (green) causes instability in the network resulting in a complete breakdown of learning.
An interesting feature in this graph is the fact that the accuracy for the medium learning rate (orange) briefly peaks around epoch 500. This hints at the fact that the learning rate at that point in time might have become too great to maintain high accuracy. A good next step in implementation would be to gradually decrease the learning rate with increasing accuracy or number of epochs.

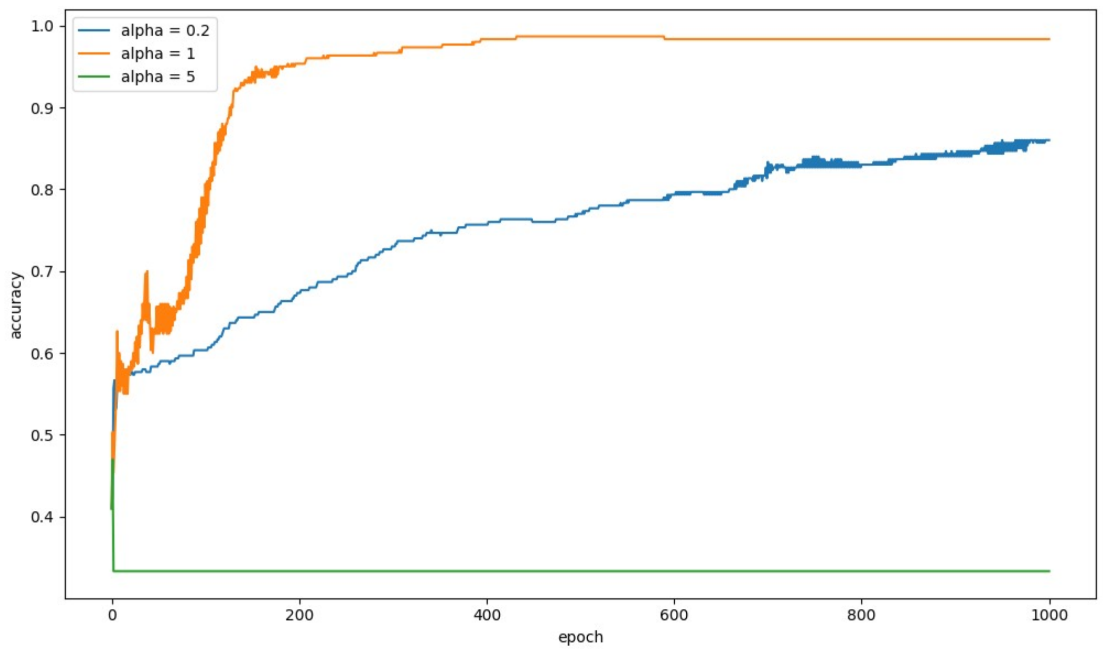

Figure 1. Convergence for different learning rates.

$$p_k(s) = \frac{e^{s_k}}{\sum_{c=1}^{K} e^{s_c}} \qquad l = -\sum_{k=1}^{K} t_k \log(p_k)$$

$l(p_k(s)) = ?$ = FORWARD MESSAGE

$$= -\sum_{k=1}^{K} t_k \left[\log e^{s_k} - \log \sum_{c=1}^{K} e^{s_c}\right] = \boxed{-\sum_{k=1}^{K} t_k \left[s_k - \log \sum_{c=1}^{K} e^{s_c}\right]}$$

BACKWARD MESSAGE

$$l = -\sum_{k=1}^{K} t_k \log p_k \qquad \frac{\partial l}{\partial p} = -\frac{t}{p}$$

$$p = \frac{\exp(s)}{\sum \exp(s)} \qquad \frac{\partial p}{\partial s} = \operatorname{diag} p - p p^T$$

$$\frac{\partial l}{\partial s} = \frac{\partial l}{\partial p} \cdot \frac{\partial p}{\partial s} = -\frac{t}{p} @ \left[\operatorname{diag} p - p p^T \right] =$$

$$= \left[-\frac{t_1}{p_1} p_1 + \frac{t_1}{p_1} p_1 p_1 + \frac{t_2}{p_2} p_1 p_2 + \dots , -\frac{t_2}{p_2} p_2 + \frac{t_2}{p_2} p_2 p_1 + \dots , \dots \right] =$$

$$= \left[-t_1 + \underbrace{\left[\sum_{k=1}^{K} t_k\right]}_{=1} p_1 , -t_2 + \underbrace{\left[\sum_{k=1}^{K} t_k\right]}_{=1} p_2 , \dots \right] = \boxed{-t + p}$$

ONE HOT

SOFTMAX INVARIANT TO SHIFTS IN INPUTS?

$$p_k(s) \overset{?}{=} p_k(s+r)$$

$$\hookrightarrow r = \begin{bmatrix} r \\ r \\ \vdots \\ r \end{bmatrix} \text{ CONSTANT}$$

$$p_k(s) = \frac{\exp(s_k)}{\sum \exp(s)} \overset{?}{=} \overset{\overset{p_k(s+r)}{\parallel}}{\frac{\exp(s_k + r)}{\sum \exp(s+r)}} =$$

$$= \frac{\exp(s_k)\cancel{\exp(r)}}{\left[\sum \exp(s)\right]\cancel{\exp(r)}} = \frac{\exp(s_k)}{\sum \exp(s)} = p_k(s)$$

□

Task 3

The weights all begin at relative amplitude 1 since that is defined. We can see in Figure 1 that the first five layers of the network all have relatively small amplitudes close to zero. This makes sense because neural networks in general are non-linear function estimators and the activation function in this network is the ReLU, which has the non-linearity threshold at zero.

By having weights around zero, changing them slightly gives the network the greatest amount of leverage when learning—it can flip the sign or keep the current layer's output close to the zero-threshold of the next layer.

The biggest change comes at the last layer. We are teaching the network to one-hot encode its output, which means it has to sort out the information it was passing around till then and give a definite answer. It pushes some outputs down to zero and it pushes others to one.

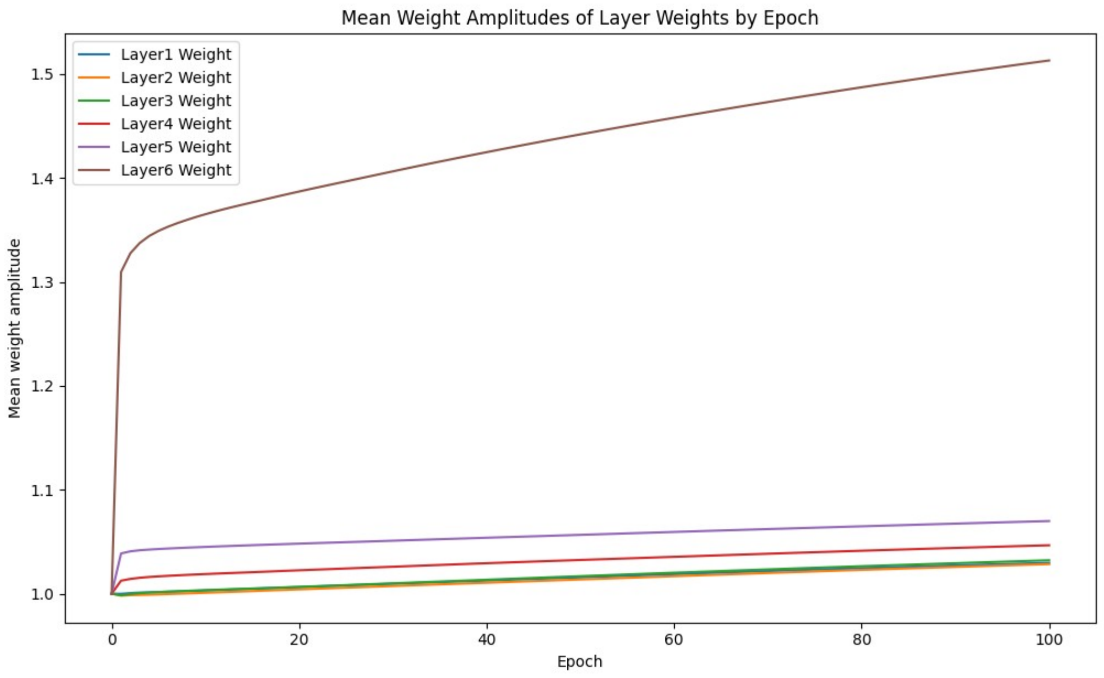

Figure 1. Mean weight amplitudes of layer weights by epoch.