1.1 is simple, here only shows the derivative of part 5

Consider a single sample, $\nabla y * log(p) = \nabla log \frac{exp(o_i)}{exp(o_1)+exp(o_2)+...+exp(o_{10})}$, where oi is the correct class

$$= \frac{exp(o_1)+exp(o_2)+...+exp(o_{10})}{exp(o_i)} \nabla \frac{exp(o_i)}{exp(o_1)+exp(o_2)+...+exp(o_{10})}$$

If j = i, $= \frac{exp(o_j)}{exp(o_1)+exp(o_2)+...+exp(o_{10})} - 1$

If j != i, $= \frac{exp(o_j)}{exp(o_1)+exp(o_2)+...+exp(o_{10})}$

Hence, the code looks like:

```
def grad_ce(target, predict):
  return predict - target
```

1.1.1 - 1.1.4

```
# Implementation of a neural network using only Numpy - trained using
gradient descent with momentum
def relu(x):
  return np.clip(x, 0, None)

# input (N, 10)
# output (N, 10)
def softmax(x):
  predict = x - x.max(axis=1, keepdims=True)
  return np.exp(predict) / np.exp(predict).sum(axis=1, keepdims=True)

def compute_layer(x, w, b):
  return x @ w + b

def average_ce(target, prediction):

  mult = target * np.log(prediction)
  sum = np.sum(mult, axis=1)
  return - np.mean(sum)
```

1.2 here shows how four derivatives are calculated:

$$\frac{dL}{dw_o} = \frac{dL}{do}\frac{do}{dw_o} = h^T\frac{dL}{do}$$

$$\frac{dL}{db_o} = \frac{dL}{do}\frac{do}{db_o} = 1\frac{dL}{do}$$

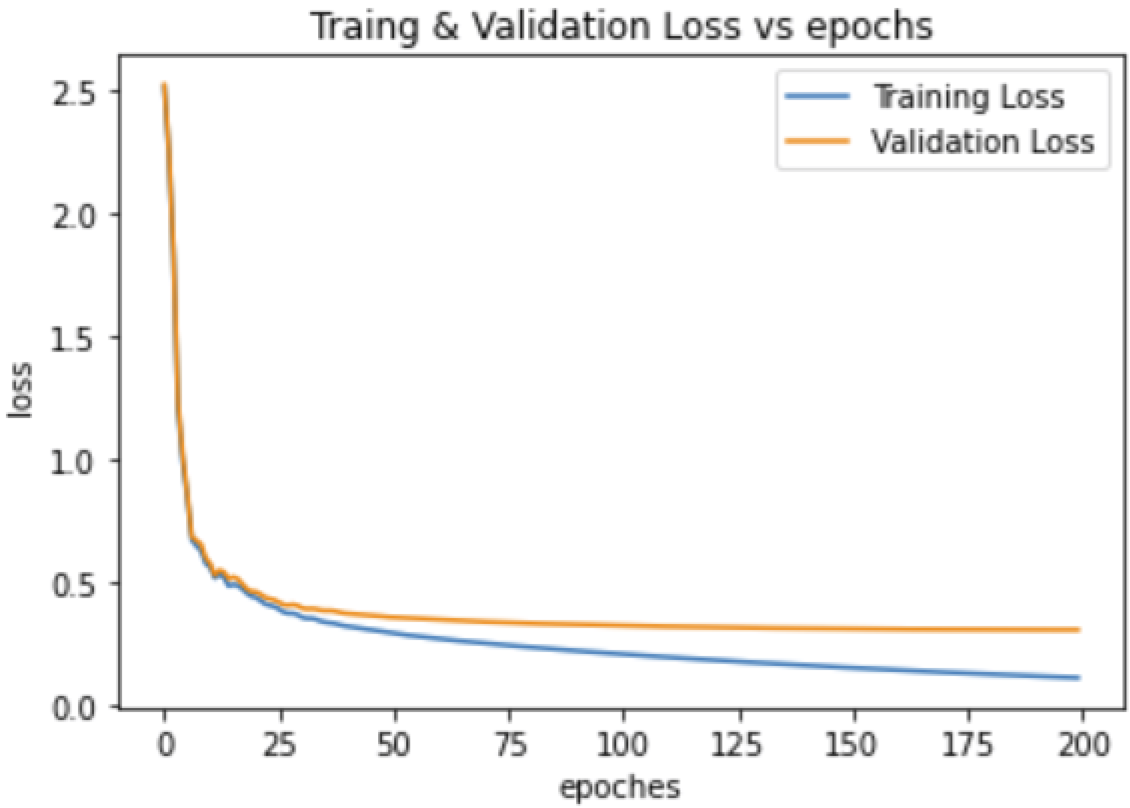
$$\frac{dL}{dw_h} = \frac{dL}{do}\frac{do}{dh}\frac{dh}{dw_h} = x^T(dRelu \otimes \frac{dL}{do}w_o^T)$$

$$\frac{dL}{db_h} = \frac{dL}{do}\frac{do}{dh}\frac{dh}{db_h} = 1(dRelu \otimes \frac{dL}{do}{w_o}^T)$$

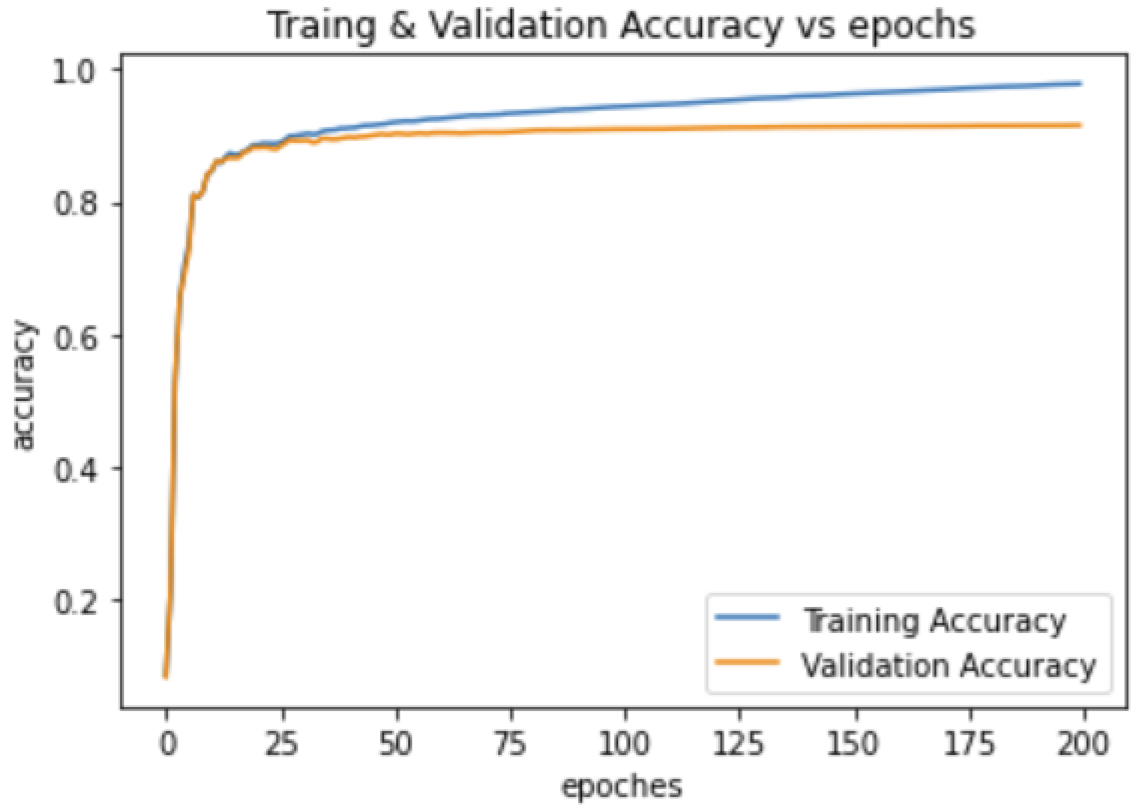
Hence, the code looks like:

```
def back_propagation(x, y, w_o, b_o, w_h, b_h, h, predict):
  # (K, N) @ (N, 10) = (K, 10)
  do = grad_ce(y, predict)
  dw_o = np.transpose(h) @ do / y.shape[0]

  # (1, N) @ (N, 10) = (1, 10)
  one = np.ones((1, y.shape[0]))
  db_o = one @ do / y.shape[0]

  # (N, K)
  dRelu = h
  dRelu[dRelu > 0] = 1

  # (784, N) @ ((N, 10) @ (10, K) * (N, K)) = (784, K)
  dw_h = np.transpose(x) @ (do @ np.transpose(w_o) * dRelu) / y.shape[0]

  # (1, N) @ ((N, 10) @ (10, K) * (N, K)) = (1, K)
  one = np.ones((1, x.shape[0]))
  db_h = one @ (do @ np.transpose(w_o) * dRelu) / y.shape[0]

  return dw_o, db_o, dw_h, db_h
```

1.3 After training, the loss and accuracy images look like below.



Traing & Validation Loss vs epochs
Training Loss
Validation Loss
loss
2.5
2.0
1.5
1.0
0.5
0.0
0
25
50
75
100
125
150
175
200
epoches


Traing & Validation Accuracy vs epochs
1.0
0.8
0.6
0.4
0.2
accuracy
0
25
50
75
100
125
150
175
200
epoches
Training Accuracy
Validation Accuracy