<https://github.com/manu-2213/QMUL_MLSociety/tree/main/Lectures/Lecture5>

weights=np.array(weights)

print(weights)

x\_train = df2\_num.iloc[:,1:-1]

x\_train = np.array(x\_train)

y\_train = df2\_num['SalePrice']

y\_train = np.array(y\_train)

df2=pd.read\_csv('./train.csv')

df2\_num = df2.select\_dtypes(include = ['float64', 'int64'])

x\_test = df\_num.iloc[:,1:]

x\_test = np.array(x\_test)

def zscore\_normalize\_features(X):

X\_norm = (X - X\_mu) / X\_sigma

return (X\_norm, X\_mu, X\_sigma)

X\_norm\_test, X\_mu, X\_sigma = zscore\_normalize\_features(x\_test)

X\_norm\_test, X\_mu\_test, X\_sigma\_test = zscore\_normalize\_features(x\_test)

inputs = [[1.0, 2.0, 3.0, 2.5],

[2.0, 5.0, -1.0, 2.0],

[-1.5, 2.7, 3.3, -0.8]]

weights = [[0.2, 0.8, -0.5, 1.0],

[0.5, -0.91, 0.26, -0.5],

[-0.26, -0.27, 0.17, 0.87]]

biases = [2.0, 3.0, 0.5]

layer\_outputs1 = np.dot(inputs, np.array(weights).T) + biases

print(layer\_outputs1)

weights2 = np.random.rand(len(weights), len(weights[0]))

print(weights2)

layer\_outputs2= np.dot(inputs, np.array(weights2).T) + biases

#We flatten the weights and inputs to a 1 dimensional array, so we can aggregate to one output.

weights2\_flat = weights2.flatten() # Flatten weights to a single 1D array

inputs\_flat = np.array(inputs).flatten() # Flatten inputs to a single 1D array

combined\_bias = np.sum(biases)

single\_output = np.dot(inputs\_flat, weights2\_flat) + combined\_bias

print("Single Output:", single\_output)

#Question 2

def relu(x):

return np.maximum(0,x)

relu(layer\_outputs1)

relu(layer\_outputs2)