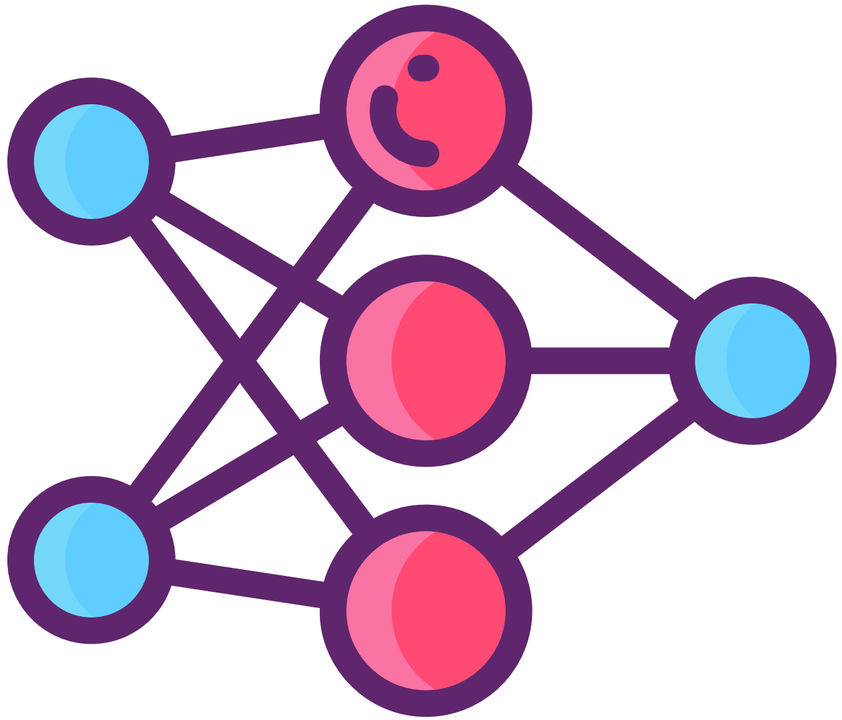
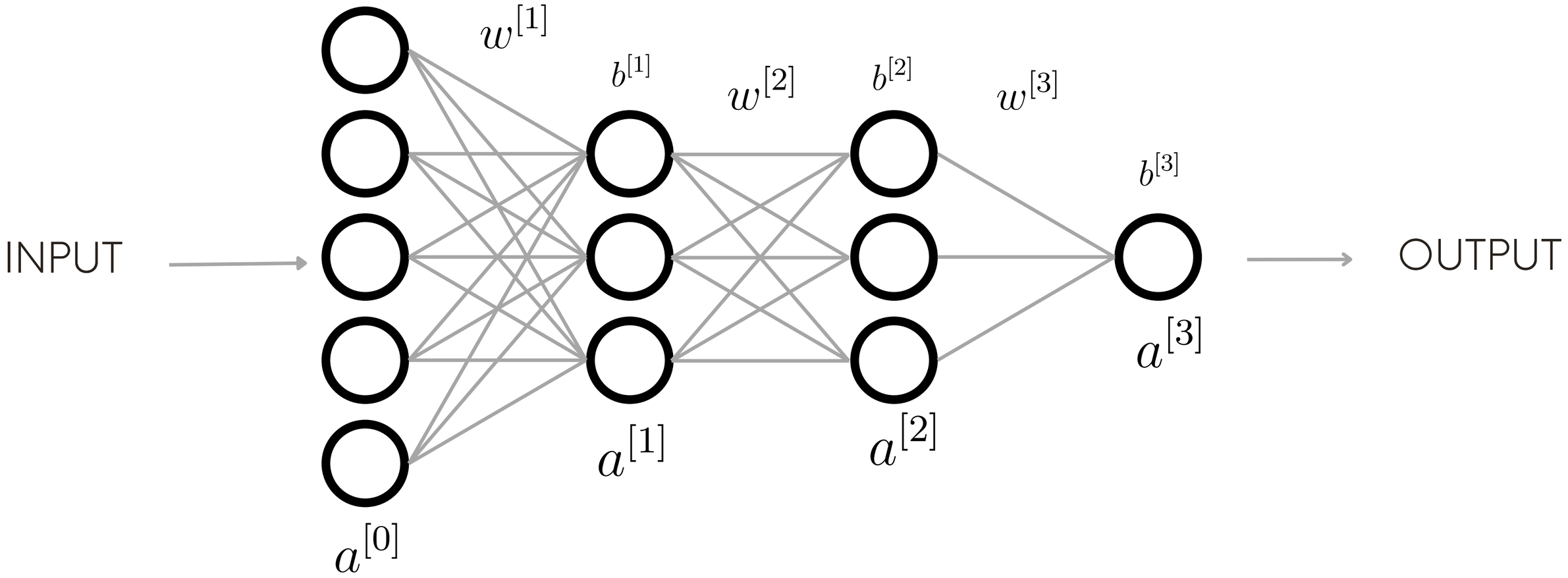
DEEP NEURAL NETWORKS

AND MATHEMATICS BEHIND THEM FROM SCRATCH



ARCHITECTURE



IMPORTING DEPENDENCIES

```
import numpy as np
import pandas as pd
import random
```

FORMING A DATASET

```
# Create the features as a NumPy array with 20 rows and 5 columns
features = np.random.randint(6, size=(20, 5))

# Create the labels as a NumPy array with 20 elements
labels = np.random.randint(2, size=20)

# Create a dictionary with the feature and label arrays
data = {f"feature {i+1}": features[:,i] for i in range(5)}
data['labels'] = labels

# Create a Pandas DataFrame with the feature and label arrays
df = pd.DataFrame(data)
```

	feature 1	feature 2	feature 3	feature 4	feature 5	labels
0	0	3	5	0	0	1
1	2	0	5	2	0	0
2	4	3	3	4	1	0
3	1	0	1	5	3	0
4	4	4	3	3	0	0
5	4	5	4	1	5	0
6	5	2	1	2	2	1
7	5	2	4	1	5	1
8	3	4	3	4	0	1
9	1	4	2	2	2	0
10	1	3	5	5	4	1
11	1	4	5	0	3	1
12	4	3	1	4	2	1
13	2	4	2	1	1	1
14	5	3	5	4	5	0
15	0	2	2	1	0	1
16	4	3	2	3	3	0
17	5	4	5	3	4	0
18	0	4	2	1	5	1
19	5	1	1	0	5	0

INITIALISING WEIGHTS, BIASES AND ACTIVATIONS

```
w1 = np.random.randn(3,5) * 0.01
w2 = np.random.randn(3,3) * 0.01
w3 = np.random.randn(1,3) * 0.01

b1 = np.zeros((3,1))
b2 = np.zeros((3,1))
b3 = np.zeros((1,1))
```

```
a0 = np.zeros((5,1))
a1 = np.zeros((3,1))
a2 = np.zeros((3,1))
a3 = np.zeros((1,1))
z1 = np.zeros((3,1))
z2 = np.zeros((3,1))
z3 = np.zeros((1,1))
```

FOWARD PASS TO SET VALUE OF ACTIVATIONS

```
# features variable contains all the training examples as a numpy array
# label variale contains the features
x = np.transpose(features)
y = np.transpose(labels)

#Forward Pass
a0 = x
z1 = np.dot(w1,a0) + b1
a1 = np.maximum(0,z1)
z2 = np.dot(w2,a1) + b2
a2 = np.maximum(0,z2)
z3 = np.dot(w3,a2) + b3
print(z3.shape)
a3 = 1/(1 + np.exp(-z3))
```

```
print('w1', w1)
print('w2', w2)
print('w3', w3)
```

```
w1 [[ 1.58302508e-02  1.73055621e-02 -8.33510453e-05  4.88404344e-03
  -2.79817476e-03]
 [ 3.17871212e-03 -4.45282930e-03  5.89123166e-03  1.10903285e-03
   4.89872582e-03]
 [-6.60564901e-03  8.68040481e-04 -2.18391081e-02 -1.97517176e-02
   6.45944868e-03]]
w2 [[ 0.01943714  0.00459172 -0.00890106]
 [ 0.00131624 -0.03110848 -0.01463569]
 [-0.00235501  0.01066755  0.01312488]]
w3 [[-0.00019923  0.00371024 -0.00162017]]
```

APPLYING BACKPROPAGATION AND CALCULATING GRADIENTS

```
#BackProp
dZ3 = a3 - y

dW3 = np.dot(dZ3 , np.transpose(a2))/20
db3 = np.sum(dZ3, axis=1, keepdims=True)
temp1 = 1 * (z2>0)
dZ2 = np.dot(np.transpose(w3),dZ3) * temp1
dW2 = np.dot(dZ2 , np.transpose(a1))/20
db2 = np.sum(dZ2, axis=1, keepdims=True)
temp2 = 1 * (z1>0)
dZ1 = np.dot(np.transpose(w2),dZ2) * temp2
dW1 = np.dot(dZ1 , np.transpose(a0))/20
db1 = np.sum(dZ1, axis=1, keepdims=True)
```

THE BATCH SIZE HAS BEEN SET AS 20.
IT CAN BE VARIED FOR MINI BATCH G.D.

VECTORIZED BACKPROP MATH

$$dZ^{[L]} = A^{[L]} - Y$$

$$dW^{[L]} = \frac{1}{m} dZ^{[L]} A^{[L]^T}$$

$$db^{[L]} = \frac{1}{m} np.\text{sum}(\text{d}Z^{[L]}, axis = 1, keepdims = True)$$

$$dZ^{[L-1]} = W^{[L]^T} dZ^{[L]} g'^{[L]}(Z^{[L-1]})$$

$$\vdots$$

$$dZ^{[1]} = W^{[L]^T} dZ^{[2]} g'^{[1]}(Z^{[1]})$$

$$dW^{[1]} = \frac{1}{m} dZ^{[1]} A^{[1]^T}$$

$$db^{[1]} = \frac{1}{m} np.\text{sum}(\text{d}Z^{[1]}, axis = 1, keepdims = True)$$

UPDATING VALUES OF THE GRADIENTS

USING A CONSTANT LEARNING RATE. WE CAN APPLY CONCEPTS OF MOMENTUM AND ACCELERATION HERE

```
learning_rate = 0.1
w3 = w3 - (learning_rate*dW3)
w2 = w2 - (learning_rate*dW2)
w1 = w1 - (learning_rate*dW1)
b3 = b3 - (learning_rate*db3)
b2 = b2 - (learning_rate*db2)
b1 = b1 - (learning_rate*db1)
```

FINDING NEW OUTPUTS

```
x = np.transpose(features)
y = np.transpose(labels)

#Forward Pass
a0 = x
z1 = np.dot(w1,a0) + b1
a1 = np.maximum(0,z1)
z2 = np.dot(w2,a1) + b2
a2 = np.maximum(0,z2)
z3 = np.dot(w3,a2) + b3
print(z3.shape)
a3 = 1/(1 + np.exp(-z3))
```

```
dW1
```

```
array([[ 4.56985799e-07,  3.87253205e-07,  9.39629686e-08,
         2.74761758e-07,  1.82224167e-07],
       [-8.52973824e-06,  9.14875452e-08, -8.87027033e-07,
        -4.93586132e-06, -3.59387158e-06],
       [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00]])
```

FLOW CHART

IMPORTING DEPENDENCIES

```
import numpy as np
import pandas as pd
import random
```

FORMING/IMPORTING DATASET

```
# Create the features as a NumPy array with 20 rows and 5 columns
features = np.random.randint(6, size=(20, 5))

# Create the labels as a NumPy array with 20 elements
labels = np.random.randint(2, size=20)

# Create a dictionary with the feature and label arrays
data = {f"feature {i+1}": features[:,i] for i in range(5)}
data['labels'] = labels
```

INITIALISING WEIGHTS & BIASES

```
w1 = np.random.randn(3,5) * 0.01
w2 = np.random.randn(3,3) * 0.01
w3 = np.random.randn(1,3) * 0.01

b1 = np.zeros((3,1))
b2 = np.zeros((3,1))
b3 = np.zeros((1,1))
```

```
a0 = np.zeros((5,1))
a1 = np.zeros((3,1))
a2 = np.zeros((3,1))
a3 = np.zeros((1,1))
z1 = np.zeros((3,1))
z2 = np.zeros((3,1))
z3 = np.zeros((1,1))
```

FORWARD PASS

```
#Forward Pass
a0 = x
z1 = np.dot(w1,a0) + b1
a1 = np.maximum(0,z1)
z2 = np.dot(w2,a1) + b2
a2 = np.maximum(0,z2)
z3 = np.dot(w3,a2) + b3
print(z3.shape)
a3 = 1/(1 + np.exp(-z3))
```

BACKPROPAGATION & UPDATING GRADIENTS

```
#BackProp
dZ3 = a3 - y

dW3 = np.dot(dZ3 , np.transpose(a2))/20
db3 = np.sum(dZ3, axis=1, keepdims=True)
temp1 = 1 * (z2>0)
dZ2 = np.dot(np.transpose(w3),dZ3) * temp1
dW2 = np.dot(dZ2 , np.transpose(a1))/20
db2 = np.sum(dZ2, axis=1, keepdims=True)
temp2 = 1 * (z1>0)
dZ1 = np.dot(np.transpose(w2),dZ2) * temp2
dW1 = np.dot(dZ1 , np.transpose(a0))/20
db1 = np.sum(dZ1, axis=1, keepdims=True)
```

```
learning_rate = 0.1
w3 = w3 - (learning_rate*dW3)
w2 = w2 - (learning_rate*dW2)
w1 = w1 - (learning_rate*dW1)
b3 = b3 - (learning_rate*db3)
b2 = b2 - (learning_rate*db2)
b1 = b1 - (learning_rate*db1)
```