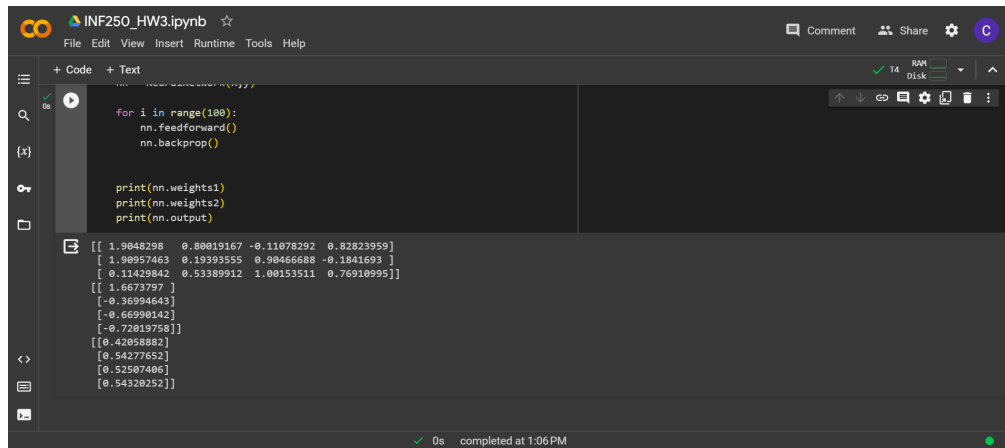


HW3: ML, Using Google's Colab, Verification "by hand," Teachable Machine

Q1

Backdrop 100 times:

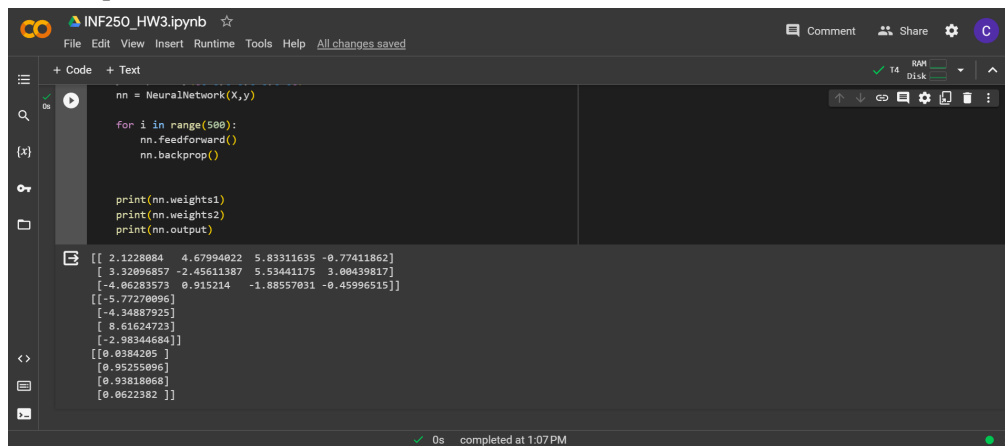


A Google Colab notebook titled 'INF250_HW3.ipynb' showing the execution of a neural network training script. The code includes a loop for 100 iterations, printing weights and output. The output shows a 3x3 weight matrix, a 3x1 weight vector, and a 3x1 output vector.

```
for i in range(100):  
    nn.feedforward()  
    nn.backprop()  
  
print(nn.weights1)  
print(nn.weights2)  
print(nn.output)
```

```
[[ 1.9848298  0.80819167 -0.11078292  0.82823959]  
 [ 1.98957463  0.19393555  0.90466688 -0.1841693 ]  
 [ 0.11429842  0.53389912  1.08153511  0.76910995]]  
[[ 1.6673797 ]  
 [-0.36994643]  
 [-0.66998142]  
 [-0.72019758]]  
[[0.42858882]  
 [0.54277652]  
 [0.52507406]  
 [0.54320252]]
```

Backdrop 500 times:

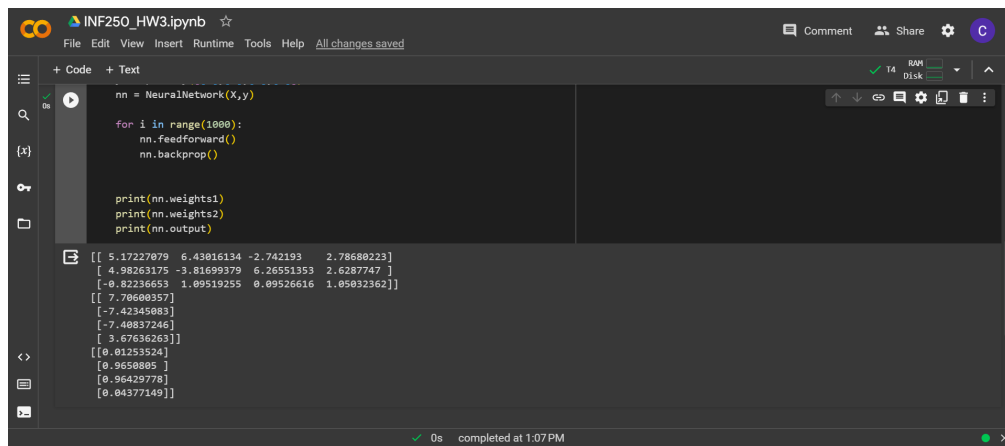


A Google Colab notebook titled 'INF250_HW3.ipynb' showing the execution of a neural network training script. The code includes a loop for 500 iterations, printing weights and output. The output shows a 3x3 weight matrix, a 3x1 weight vector, and a 3x1 output vector.

```
nn = NeuralNetwork(X,y)  
  
for i in range(500):  
    nn.feedforward()  
    nn.backprop()  
  
print(nn.weights1)  
print(nn.weights2)  
print(nn.output)
```

```
[[ 2.1228084  4.67994022  5.83311635 -0.77411862]  
 [ 3.32096857 -2.45611387  5.53441175  3.00439817]  
 [-4.06283573  0.915214 -1.88557031 -0.45996515]]  
[[-5.77270096]  
 [-4.34837925]  
 [ 8.61624723]  
 [-2.98344684]]  
[[0.0384205 ]  
 [0.95255096]  
 [0.93818068]  
 [0.0622382 ]]
```

Backdrop 1000 times:



A Google Colab notebook titled 'INF250_HW3.ipynb' showing the execution of a neural network training script. The code includes a loop for 1000 iterations, printing weights and output. The output shows a 3x3 weight matrix, a 3x1 weight vector, and a 3x1 output vector.

```
nn = NeuralNetwork(X,y)  
  
for i in range(1000):  
    nn.feedforward()  
    nn.backprop()  
  
print(nn.weights1)  
print(nn.weights2)  
print(nn.output)
```

```
[[ 5.17227079  6.43016134 -2.742193  2.78680223]  
 [ 4.98263175 -3.81699379  6.26551353  2.6287747 ]  
 [-0.82236653  1.09519255  0.09526616  1.05032362]]  
[[ 7.78600357]  
 [-7.42345083]  
 [-7.40837246]  
 [ 3.67636263]]  
[[0.01253524]  
 [0.96508005]  
 [0.96429778]  
 [0.04377149]]
```

Backdrop 5000 times:

```

nn = NeuralNetwork(X,y)

for i in range(5000):
    nn.feedforward()
    nn.backprop()

print(nn.weights1)
print(nn.weights2)
print(nn.output)

```

```

[[ 3.79596152 -2.04690372  3.55892456  6.65194896]
 [ 2.78266873  2.55900056  2.52783882  7.16577672]
 [-5.0331172   2.14180634 -4.73202113 -3.19474024]]
[[-7.07100496]
 [-5.32154784]
 [-6.46903052]
 [11.19454632]]
[[0.011847 ]
 [0.98838852]
 [0.99294082]
 [0.00934226]]

```

Backdrop 20000 times:

```

nn = NeuralNetwork(X,y)

for i in range(20000):
    nn.feedforward()
    nn.backprop()

print(nn.weights1)
print(nn.weights2)
print(nn.output)

```

```

[[ 7.59200053  4.34449507  5.28335212 -4.53237508]
 [-3.33893469  4.47633784  5.40971412  7.72371795]
 [ 0.02229893  0.14274579 -0.60847909  1.13742893]]
[[-10.55979748]
 [ 7.16429051]
 [ 8.89834933]
 [-10.55977457]]
[[0.00172387]
 [0.99325499]
 [0.99332958]
 [0.00835961]]

```

Pattern: It seems that with the increase of the input range for each backdrop to perform, the output values get closer and closer to those desired (ie. 0, 1, 1, 0). Empirically, this may be due to the fact that the NN learns with each iteration of the backdrop.

Q2

Below, there are 4 screenshots that include my calculations based on the NN weights, 001, 011, 101, 111 (in that exact order). The highlighted parts are my Y output:

W020=	-10.5598	X1=	0	-5.33876	W020 part	Total = W020 + W120 + W220 + W320	
W010=	7.592005	X2=	0			-6.36149	
W110=	-3.33893	X3=	1				
W210=	0.022299	0.022299	1 + e^=	1.977948		1+e^(total)=	0.001723817
W120=	7.164291	X1=	0	3.83738	W120 part		
W011=	4.344495	X2=	0				
W111=	4.476338	X3=	1				
W211=	0.142746	0.142746	1 + e^=	1.866974			
W220=	8.898349	X1=	0	3.135834	W220 part		
W012=	5.283352	X2=	0				
W112=	5.409714	X3=	1				
W212=	-0.60848	-0.60848	1 + e^=	2.837634			
W320=	-10.5598	X1=	0	-7.99594	W320 part		
W013=	-4.53238	X2=	0				
W113=	7.723718	X3=	1				
W213=	1.137429	1.137429	1 + e^=	1.320642			

W020=	-10.5598	X1=	0	-0.36964	W020 part	Total = W020 + W120 + W220 + W320	
W010=	7.592005	X2=	1			4.99221	
W110=	-3.33893	X3=	1				
W210=	0.022299		$-3.31664 \cdot 1 + e^{\wedge}$	28.56745		$1 + e^{\wedge}(\text{total}) =$	0.993255164
W120=	7.164291	X1=	0	7.094327	W120 part		
W011=	4.344495	X2=	1				
W111=	4.476338	X3=	1				
W211=	0.142746		$4.619084 \cdot 1 + e^{\wedge}$	1.009862			
W220=	8.898349	X1=	0	8.825805	W220 part		
W012=	5.283352	X2=	1				
W112=	5.409714	X3=	1				
W212=	-0.60848		$4.801235 \cdot 1 + e^{\wedge}$	1.00822			
W320=	-10.5598	X1=	0	-10.5583	W320 part		
W013=	-4.53238	X2=	1				
W113=	7.723718	X3=	1				
W213=	1.137429		$8.861147 \cdot 1 + e^{\wedge}$	1.000142			

W020=	-10.5598	X1=	1	-10.5546	W020 part	Total = W020 + W120 + W220 + W320	
W010=	7.592005	X2=	0			5.003406	
W110=	-3.33893	X3=	1				
W210=	0.022299		$7.614304 \cdot 1 + e^{\wedge}$	1.000493		$1 + e^{\wedge}(\text{total}) =$	0.993329756
W120=	7.164291	X1=	1	7.084577	W120 part		
W011=	4.344495	X2=	0				
W111=	4.476338	X3=	1				
W211=	0.142746		$4.487241 \cdot 1 + e^{\wedge}$	1.011252			
W220=	8.898349	X1=	1	8.816124	W220 part		
W012=	5.283352	X2=	0				
W112=	5.409714	X3=	1				
W212=	-0.60848		$4.674873 \cdot 1 + e^{\wedge}$	1.009327			
W320=	-10.5598	X1=	1	-0.3427	W320 part		
W013=	-4.53238	X2=	0				
W113=	7.723718	X3=	1				
W213=	1.137429		$-3.39495 \cdot 1 + e^{\wedge}$	30.81305			

W020=	-10.5598	X1=	1	-10.415	W020 part	Total = W020 + W120 + W220 + W320	
W010=	7.592005	X2=	1			-4.77597	
W110=	-3.33893	X3=	1				
W210=	0.022299		$4.27537 \cdot 1 + e^{\wedge}$	1.013907		$1 + e^{\wedge}(\text{total}) =$	0.008359395
W120=	7.164291	X1=	1	7.163374	W120 part		
W011=	4.344495	X2=	1				
W111=	4.476338	X3=	1				
W211=	0.142746		$8.963579 \cdot 1 + e^{\wedge}$	1.000128			
W220=	8.898349	X1=	1	8.897978	W220 part		
W012=	5.283352	X2=	1				
W112=	5.409714	X3=	1				
W212=	-0.60848		$10.08459 \cdot 1 + e^{\wedge}$	1.000042			
W320=	-10.5598	X1=	1	-10.4224	W320 part		
W013=	-4.53238	X2=	1				
W113=	7.723718	X3=	1				
W213=	1.137429		$4.328772 \cdot 1 + e^{\wedge}$	1.013184			

^It's noticeable that put together, the Y output values produced are close to the ones desired [0, 1, 1, 0], and they closely match the output given from the run with 20000 iterations (see last backdrop screenshot).

Q3

<https://teachablemachine.withgoogle.com/models/52zJlgLIA/> ← link to my teachable model

<https://drive.google.com/drive/folders/1FBqDXfgp6-77GsCnESeB3S4QAv8g3N5H?usp=sharing> ← link to all the videos demonstrating the training process and final results **I will also include these videos as a .mov file in the zipped folder, as an alternative**