In [1]:
```
import tensorflow as tf
```

In [2]:
```
import numpy as np
import pandas as pd
import os

# Define logical operations
orLogic = {"x1": [0, 0, 1, 1], "x2": [0, 1, 0, 1], "y": [0, 1, 1, 1]}
dor = pd.DataFrame(data=orLogic)

andLogic = {"x1": [0, 0, 1, 1], "x2": [0, 1, 0, 1], "y": [0, 0, 0, 1]}
dand = pd.DataFrame(data=andLogic)

xorLogic = {"x1": [0, 0, 1, 1], "x2": [0, 1, 0, 1], "y": [0, 1, 1, 0]}
dxor = pd.DataFrame(data=xorLogic)

nandLogic = {"x1": [0, 0, 1, 1], "x2": [0, 1, 0, 1], "y": [1, 1, 1, 0]
dnand = pd.DataFrame(data=nandLogic)

# Define neural network model
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(16, input_dim=2, activation='relu'),
    tf.keras.layers.Dense(8, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(optimizer='adam',
              loss='mean_squared_error',
              metrics=['binary_accuracy'])
```

```
/Users/akheruddinahmed/anaconda3/lib/python3.11/site-packages/keras/
src/layers/core/dense.py:86: UserWarning: Do not pass an `input_shap
e`/`input_dim` argument to a layer. When using Sequential models, pr
efer using an `Input(shape)` object as the first layer in the model
instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwar
gs)
```

In [3]:
```
x_or= dor[["x1","x2"]]
y_or = dor["y"]
```

In [4]:
```
model.fit(x_or,y_or,epochs=250)
```

```
Epoch 1/250
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 581ms/step - binary_accuracy: 1.0000 - l
oss: 0.2219
Epoch 2/250
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - binary_accuracy: 0.7500 - lo
ss: 0.2212
Epoch 3/250
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - binary_accuracy: 0.7500 - lo
ss: 0.2204
Epoch 4/250
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - binary_accuracy: 0.7500 - lo
ss: 0.2197
Epoch 5/250
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - binary_accuracy: 0.7500 - lo
ss: 0.2189
Epoch 6/250
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - binary_accuracy: 0.7500 - lo
ss: 0.2182
Epoch 7/250
```

In [5]:
```
x_test = dor[["x1","x2"]]
model.predict(x_test).round()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
```

Out[5]:
```
array([[0.],
       [1.],
       [1.],
       [1.]], dtype=float32)
```

In [6]:
```
x_and= dand[["x1","x2"]]
y_and = dand["y"]

print(x_and)
print(y_and)
```

```
   x1  x2
0   0   0
1   0   1
2   1   0
3   1   1
0    0
1    0
2    0
3    1
Name: y, dtype: int64
```

In [7]:
```
model.fit(x_and,y_and,epochs=250)
```

```
Epoch 1/250
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - binary_accuracy: 0.5000 - lo
ss: 0.4599
Epoch 2/250
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - binary_accuracy: 0.5000 - lo
ss: 0.4596
Epoch 3/250
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - binary_accuracy: 0.5000 - lo
ss: 0.4586
Epoch 4/250
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - binary_accuracy: 0.5000 - lo
ss: 0.4569
Epoch 5/250
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - binary_accuracy: 0.5000 - lo
ss: 0.4547
Epoch 6/250
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - binary_accuracy: 0.5000 - lo
ss: 0.4519
Epoch 7/250
```

In [8]:
```
x_nand= dnand[["x1","x2"]]
y_nand = dnand["y"]
```

In [9]:
```
model.fit(x_nand,y_nand,epochs=250)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - binary_accuracy: 0.0000e+00
- loss: 0.3409
Epoch 31/250
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - binary_accuracy: 0.0000e+00
- loss: 0.3389
Epoch 32/250
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - binary_accuracy: 0.0000e+00
- loss: 0.3369
Epoch 33/250
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - binary_accuracy: 0.0000e+00
- loss: 0.3350
Epoch 34/250
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - binary_accuracy: 0.0000e+00
- loss: 0.3332
Epoch 35/250
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - binary_accuracy: 0.0000e+00
- loss: 0.3314
Epoch 36/250
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - binary_accuracy: 0.0000e+00
- loss: 0.3296
```

In [10]:
```
x_xor= dxor[["x1","x2"]]
y_xor= dxor["y"]
```

In [11]:
```
model.fit(x_xor,y_xor,epochs=250)
```

```
Epoch 1/250
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - binary_accuracy: 0.5000 - lo
ss: 0.2485
Epoch 2/250
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - binary_accuracy: 0.5000 - lo
ss: 0.2484
Epoch 3/250
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - binary_accuracy: 0.5000 - lo
ss: 0.2484
Epoch 4/250
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - binary_accuracy: 0.5000 - lo
ss: 0.2483
Epoch 5/250
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - binary_accuracy: 0.5000 - lo
ss: 0.2483
Epoch 6/250
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - binary_accuracy: 0.5000 - lo
ss: 0.2482
Epoch 7/250
```

In [12]:
```
x_test = dnand[["x1","x2"]]

model.predict(x_test).round()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step
```

Out[12]:
```
array([[0.],
       [1.],
       [0.],
       [0.]], dtype=float32)
```

In [13]:
```
d1_test = {"x1":[0],"x2":[0] }
d1_test = pd.DataFrame(data=d1_test)
x_test_1 = d1_test[["x1","x2"]]

x_test_1


model.predict(x_test_1).round()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
```

Out[13]:
```
array([[0.]], dtype=float32)
```

In [ ]:
```

```