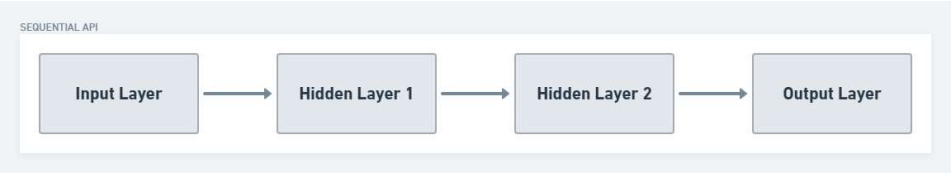
SEQUENTIAL API




```
from keras.models import Sequential
from keras.layers import Dense
model = Sequential()
model.add(Dense(2, input_dim=2, activation='relu'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 2)                 6

=================================================================
Total params: 6
Trainable params: 6
Non-trainable params: 0
_________________________________________________________________
```

```
from keras.models import Sequential
from keras.layers import Dense
model = Sequential()
model.add(Dense(2, input_dim=2, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_1 (Dense)             (None, 2)                 6

 dense_2 (Dense)             (None, 1)                 3

=================================================================
Total params: 9
Trainable params: 9
Non-trainable params: 0
_________________________________________________________________
```

OR GATE TRAINING USING KERAS

```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense

x_train = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y_train = np.array([[0], [1], [1], [1]])

model = Sequential()
model.add(Dense(2, input_dim=2, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='sgd', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=1)


x_test = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y_pred = model.predict(x_test)

y_pred = np.round(y_pred)
print("Prediction: ", y_pred)
```

```
1/1 [==============================] - 1s 872ms/step - loss: 0.5221 - accuracy: 1.0000
1/1 [==============================] - 0s 112ms/step
Prediction:  [[1.]
 [1.]
 [1.]
 [1.]]
```

FOR 100 EPOCHS

```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense

x_train = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y_train = np.array([[0], [1], [1], [1]])

model = Sequential()
model.add(Dense(2, input_dim=2, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='sgd', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=100)


x_test = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y_pred = model.predict(x_test)

y_pred = np.round(y_pred)
print("Prediction: ", y_pred)
```

```
Epoch 1/100
1/1 [==============================] - 1s 521ms/step - loss: 0.6374 - accuracy: 0.7500
Epoch 2/100
1/1 [==============================] - 0s 9ms/step - loss: 0.6364 - accuracy: 0.7500
Epoch 3/100
1/1 [==============================] - 0s 10ms/step - loss: 0.6354 - accuracy: 0.7500
Epoch 4/100
1/1 [==============================] - 0s 7ms/step - loss: 0.6344 - accuracy: 0.7500
Epoch 5/100
1/1 [==============================] - 0s 9ms/step - loss: 0.6334 - accuracy: 0.7500
Epoch 6/100
```

```
1/1 [==============================] - 0s 6ms/step - loss: 0.6324 - accuracy: 0.7500
Epoch 7/100
1/1 [==============================] - 0s 6ms/step - loss: 0.6315 - accuracy: 0.7500
Epoch 8/100
1/1 [==============================] - 0s 6ms/step - loss: 0.6305 - accuracy: 0.7500
Epoch 9/100
1/1 [==============================] - 0s 6ms/step - loss: 0.6295 - accuracy: 0.7500
Epoch 10/100
1/1 [==============================] - 0s 6ms/step - loss: 0.6286 - accuracy: 0.7500
Epoch 11/100
1/1 [==============================] - 0s 6ms/step - loss: 0.6276 - accuracy: 0.7500
Epoch 12/100
1/1 [==============================] - 0s 6ms/step - loss: 0.6267 - accuracy: 0.7500
Epoch 13/100
1/1 [==============================] - 0s 6ms/step - loss: 0.6258 - accuracy: 0.7500
Epoch 14/100
1/1 [==============================] - 0s 6ms/step - loss: 0.6248 - accuracy: 0.7500
Epoch 15/100
1/1 [==============================] - 0s 6ms/step - loss: 0.6239 - accuracy: 0.7500
Epoch 16/100
1/1 [==============================] - 0s 6ms/step - loss: 0.6230 - accuracy: 0.7500
Epoch 17/100
1/1 [==============================] - 0s 6ms/step - loss: 0.6221 - accuracy: 0.7500
Epoch 18/100
1/1 [==============================] - 0s 6ms/step - loss: 0.6212 - accuracy: 0.7500
Epoch 19/100
1/1 [==============================] - 0s 5ms/step - loss: 0.6203 - accuracy: 0.7500
Epoch 20/100
1/1 [==============================] - 0s 6ms/step - loss: 0.6194 - accuracy: 0.7500
Epoch 21/100
1/1 [==============================] - 0s 6ms/step - loss: 0.6185 - accuracy: 0.7500
Epoch 22/100
1/1 [==============================] - 0s 6ms/step - loss: 0.6176 - accuracy: 0.7500
Epoch 23/100
1/1 [==============================] - 0s 6ms/step - loss: 0.6167 - accuracy: 0.7500
Epoch 24/100
1/1 [==============================] - 0s 7ms/step - loss: 0.6158 - accuracy: 0.7500
Epoch 25/100
1/1 [==============================] - 0s 6ms/step - loss: 0.6150 - accuracy: 0.7500
Epoch 26/100
1/1 [==============================] - 0s 6ms/step - loss: 0.6141 - accuracy: 0.7500
Epoch 27/100
1/1 [==============================] - 0s 7ms/step - loss: 0.6132 - accuracy: 0.7500
Epoch 28/100
1/1 [==============================] - 0s 6ms/step - loss: 0.6124 - accuracy: 0.7500
Epoch 29/100
1/1 [==============================] - 0s 6ms/step - loss: 0.6115 - accuracy: 0.7500
```

FOR 2000 EPOCHS

```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense

x_train = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y_train = np.array([[0], [1], [1], [1]])

model = Sequential()
model.add(Dense(2, input_dim=2, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='sgd', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=2000)


x_test = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y_pred = model.predict(x_test)

y_pred = np.round(y_pred)
print("Prediction: ", y_pred)
```

```
Epoch 1/2000
1/1 [==============================] - 1s 506ms/step - loss: 1.0636 - accuracy: 0.2500
Epoch 2/2000
1/1 [==============================] - 0s 16ms/step - loss: 1.0442 - accuracy: 0.0000e+00
Epoch 3/2000
1/1 [==============================] - 0s 10ms/step - loss: 1.0260 - accuracy: 0.0000e+00
Epoch 4/2000
1/1 [==============================] - 0s 10ms/step - loss: 1.0083 - accuracy: 0.2500
Epoch 5/2000
1/1 [==============================] - 0s 10ms/step - loss: 0.9911 - accuracy: 0.2500
Epoch 6/2000
1/1 [==============================] - 0s 14ms/step - loss: 0.9745 - accuracy: 0.2500
Epoch 7/2000
1/1 [==============================] - 0s 5ms/step - loss: 0.9583 - accuracy: 0.2500
Epoch 8/2000
1/1 [==============================] - 0s 5ms/step - loss: 0.9427 - accuracy: 0.2500
Epoch 9/2000
1/1 [==============================] - 0s 6ms/step - loss: 0.9276 - accuracy: 0.2500
Epoch 10/2000
1/1 [==============================] - 0s 5ms/step - loss: 0.9129 - accuracy: 0.2500
Epoch 11/2000
1/1 [==============================] - 0s 6ms/step - loss: 0.8986 - accuracy: 0.2500
Epoch 12/2000
1/1 [==============================] - 0s 6ms/step - loss: 0.8849 - accuracy: 0.2500
Epoch 13/2000
1/1 [==============================] - 0s 6ms/step - loss: 0.8715 - accuracy: 0.2500
Epoch 14/2000
1/1 [==============================] - 0s 6ms/step - loss: 0.8585 - accuracy: 0.2500
Epoch 15/2000
1/1 [==============================] - 0s 6ms/step - loss: 0.8460 - accuracy: 0.2500
Epoch 16/2000
1/1 [==============================] - 0s 6ms/step - loss: 0.8338 - accuracy: 0.2500
Epoch 17/2000
1/1 [==============================] - 0s 7ms/step - loss: 0.8220 - accuracy: 0.2500
Epoch 18/2000
1/1 [==============================] - 0s 8ms/step - loss: 0.8106 - accuracy: 0.2500
Epoch 19/2000
1/1 [==============================] - 0s 35ms/step - loss: 0.7995 - accuracy: 0.2500
Epoch 20/2000
1/1 [==============================] - 0s 18ms/step - loss: 0.7888 - accuracy: 0.2500
Epoch 21/2000
1/1 [==============================] - 0s 19ms/step - loss: 0.7784 - accuracy: 0.2500
Epoch 22/2000
1/1 [==============================] - 0s 10ms/step - loss: 0.7683 - accuracy: 0.2500
Epoch 23/2000
1/1 [==============================] - 0s 8ms/step - loss: 0.7585 - accuracy: 0.2500
Epoch 24/2000
1/1 [==============================] - 0s 9ms/step - loss: 0.7490 - accuracy: 0.2500
Epoch 25/2000
1/1 [==============================] - 0s 8ms/step - loss: 0.7398 - accuracy: 0.2500
Epoch 26/2000
1/1 [==============================] - 0s 10ms/step - loss: 0.7309 - accuracy: 0.2500
Epoch 27/2000
1/1 [==============================] - 0s 9ms/step - loss: 0.7223 - accuracy: 0.2500
Epoch 28/2000
```

```
1/1 [==============================] - 0s 8ms/step - loss: 0.7139 - accuracy: 0.2500
```

- If the accuracy is not increasing then we need to modify the learning rate.
- This means that sgd is not reaching the learning rate.
- When you need to use many epochs use "Verbose" to save time

```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense

x_train = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y_train = np.array([[0], [1], [1], [1]])

model = Sequential()
model.add(Dense(2, input_dim=2, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='sgd', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=2000, verbose = 0)


x_test = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y_pred = model.predict(x_test)

y_pred = np.round(y_pred)
print("Prediction: ", y_pred)
```

```
1/1 [==============================] - 0s 41ms/step
Prediction:  [[0.]
 [1.]
 [1.]
 [1.]]
```

AND GATE TRAINING USING KERAS

```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense

x_train = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y_train = np.array([[0], [0], [0], [1]])

model = Sequential()
model.add(Dense(2, input_dim=2, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='sgd', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=10)


x_test = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y_pred = model.predict(x_test)

y_pred = np.round(y_pred)
print("Prediction: ", y_pred)

loss, accuracy = model.evaluate(x_train, y_train )
print ("loss: ", loss)
print ("accuracy: ", accuracy)

pred = model.predict(x_train)
pred = (model.predict(x_train)>0.5).astype(int)
pred
```

```
Epoch 1/10
1/1 [==============================] - 1s 653ms/step - loss: 0.9005 - accuracy: 0.7500
Epoch 2/10
1/1 [==============================] - 0s 14ms/step - loss: 0.8990 - accuracy: 0.7500
Epoch 3/10
1/1 [==============================] - 0s 14ms/step - loss: 0.8975 - accuracy: 0.7500
Epoch 4/10
1/1 [==============================] - 0s 16ms/step - loss: 0.8960 - accuracy: 0.7500
Epoch 5/10
1/1 [==============================] - 0s 14ms/step - loss: 0.8946 - accuracy: 0.7500
Epoch 6/10
1/1 [==============================] - 0s 15ms/step - loss: 0.8931 - accuracy: 0.7500
Epoch 7/10
1/1 [==============================] - 0s 14ms/step - loss: 0.8916 - accuracy: 0.7500
Epoch 8/10
1/1 [==============================] - 0s 17ms/step - loss: 0.8902 - accuracy: 0.7500
Epoch 9/10
1/1 [==============================] - 0s 16ms/step - loss: 0.8888 - accuracy: 0.7500
Epoch 10/10
1/1 [==============================] - 0s 18ms/step - loss: 0.8874 - accuracy: 0.7500
1/1 [==============================] - 0s 66ms/step
Prediction:  [[0.]
 [0.]
 [0.]
 [0.]]
1/1 [==============================] - 0s 221ms/step - loss: 0.8860 - accuracy: 0.7500
loss:  0.8859630823135376
accuracy:  0.75
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 29ms/step
array([[0],
       [0],
       [0],
       [0]])
```

FOR 100 EPOCHS

```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense

x_train = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y_train = np.array([[0], [0], [0], [1]])

model = Sequential()
model.add(Dense(2, input_dim=2, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='sgd', loss='binary_crossentropy', metrics=['accuracy'])
```

```
model.fit(x_train, y_train, epochs=2000, verbose=0)


x_test = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y_pred = model.predict(x_test)

y_pred = np.round(y_pred)
print("Prediction: ", y_pred)

loss, accuracy = model.evaluate(x_train, y_train )
print ("loss: ", loss)
print ("accuracy: ", accuracy)

pred = model.predict(x_train)
pred = (model.predict(x_train)>0.5).astype(int)
pred
```

```
1/1 [==============================] - 0s 72ms/step
Prediction:  [[0.]
 [0.]
 [0.]
 [0.]]
1/1 [==============================] - 0s 174ms/step - loss: 0.3964 - accuracy: 0.7500
loss:  0.39641082286834717
accuracy:  0.75
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 16ms/step
array([[0],
       [0],
       [0],
       [0]])
```

XOR GATE TRAINING FROM KERAS

```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense

x_train = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y_train = np.array([[0], [1], [1], [0]])

model = Sequential()
model.add(Dense(2, input_dim=2, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='sgd', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=10000, verbose=0)


x_test = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y_pred = model.predict(x_test)

y_pred = np.round(y_pred)
print("Prediction: ", y_pred)

loss, accuracy = model.evaluate(x_train, y_train )
print ("loss: ", loss)
print ("accuracy: ", accuracy)

pred = model.predict(x_train)
pred
```

```
1/1 [==============================] - 0s 42ms/step
Prediction:  [[0.]
 [1.]
 [1.]
 [0.]]
1/1 [==============================] - 0s 146ms/step - loss: 0.0294 - accuracy: 1.0000
loss:  0.029370509088039398
accuracy:  1.0
1/1 [==============================] - 0s 17ms/step
array([[0.04451367],
       [0.9865529 ],
       [0.9872084 ],
       [0.04451367]], dtype=float32)
```

This shows that by running many epochs we can even train XOR gate.