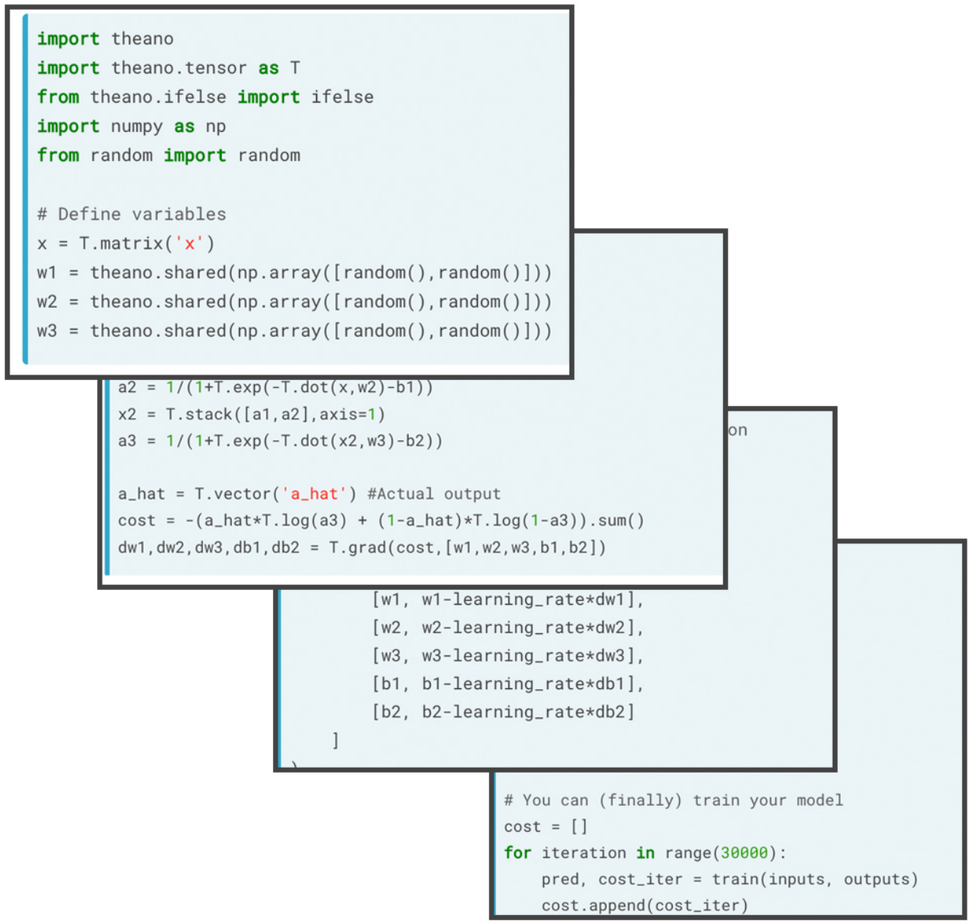
What is Keras?

INTRODUCTION TO DEEP LEARNING WITH KERAS

Miguel Esteban
Data Scientist & Founder

Theano vs Keras



```
import theano
import theano.tensor as T
from theano.ifelse import ifelse
import numpy as np
from random import random

# Define variables
x = T.matrix('x')
w1 = theano.shared(np.array([random(),random()]))
w2 = theano.shared(np.array([random(),random()]))
w3 = theano.shared(np.array([random(),random()]))
```

```
a2 = 1/(1+T.exp(-T.dot(x,w2)-b1))
x2 = T.stack([a1,a2],axis=1)
a3 = 1/(1+T.exp(-T.dot(x2,w3)-b2))

a_hat = T.vector('a_hat') #Actual output
cost = -(a_hat*T.log(a3) + (1-a_hat)*T.log(1-a3)).sum()
dw1,dw2,dw3,db1,db2 = T.grad(cost,[w1,w2,w3,b1,b2])
```

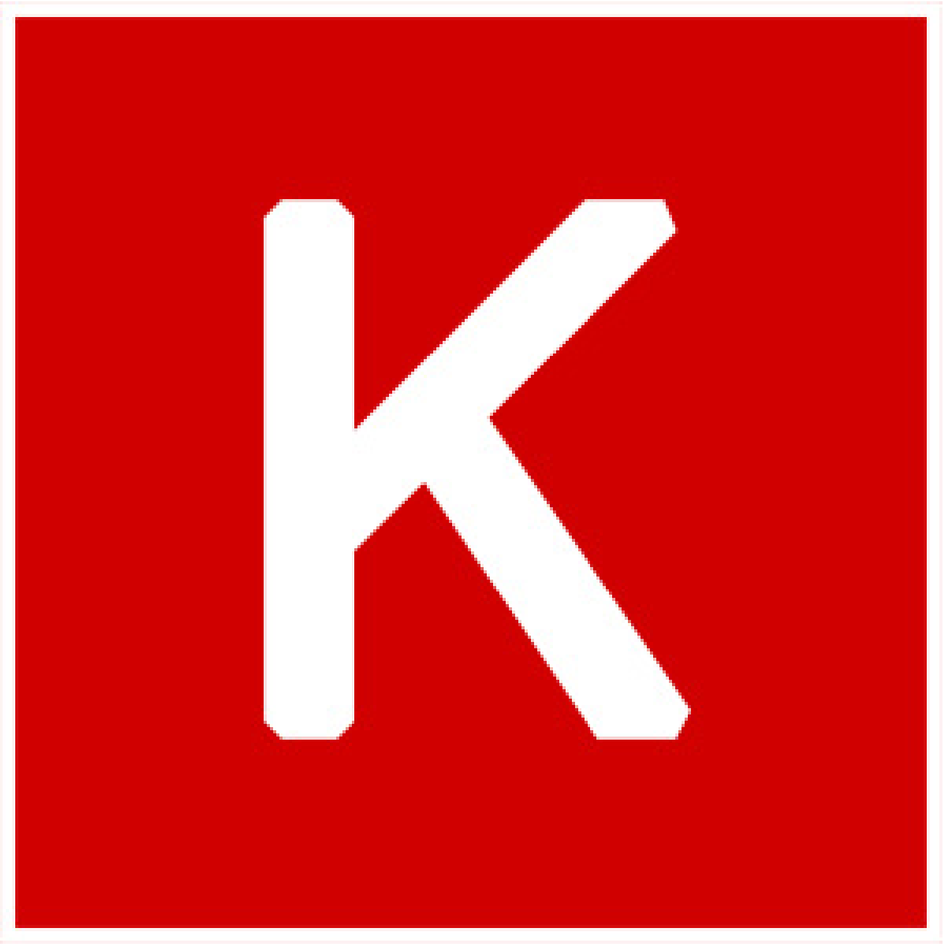
```
on
        [w1, w1-learning_rate*dw1],
        [w2, w2-learning_rate*dw2],
        [w3, w3-learning_rate*dw3],
        [b1, b1-learning_rate*db1],
        [b2, b2-learning_rate*db2]
    ]
```

```
# You can (finally) train your model
cost = []
for iteration in range(30000):
    pred, cost_iter = train(inputs, outputs)
    cost.append(cost_iter)
```

```
from keras.layers import Dense
from keras.models import Sequential

# Define model and add layers
model = Sequential()
model.add(Dense(2,input_shape=(2,),activation='sigmoid'))
model.add(Dense(1,activation='sigmoid'))

model.compile(optimizer='adam',loss='categorical_crossentropy')

# Train model
model.fit(inputs,outputs)
```

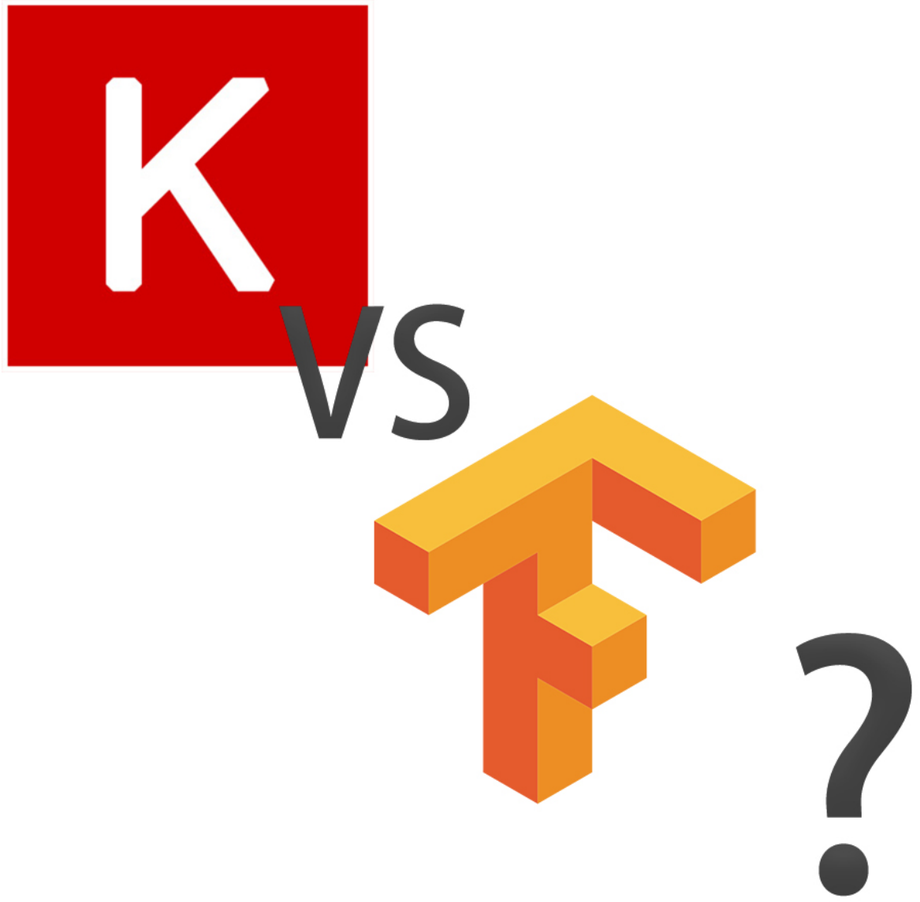
Keras

- Deep Learning Framework
- Enables fast experimentation
- Runs on top of other frameworks
- Written by François Chollet



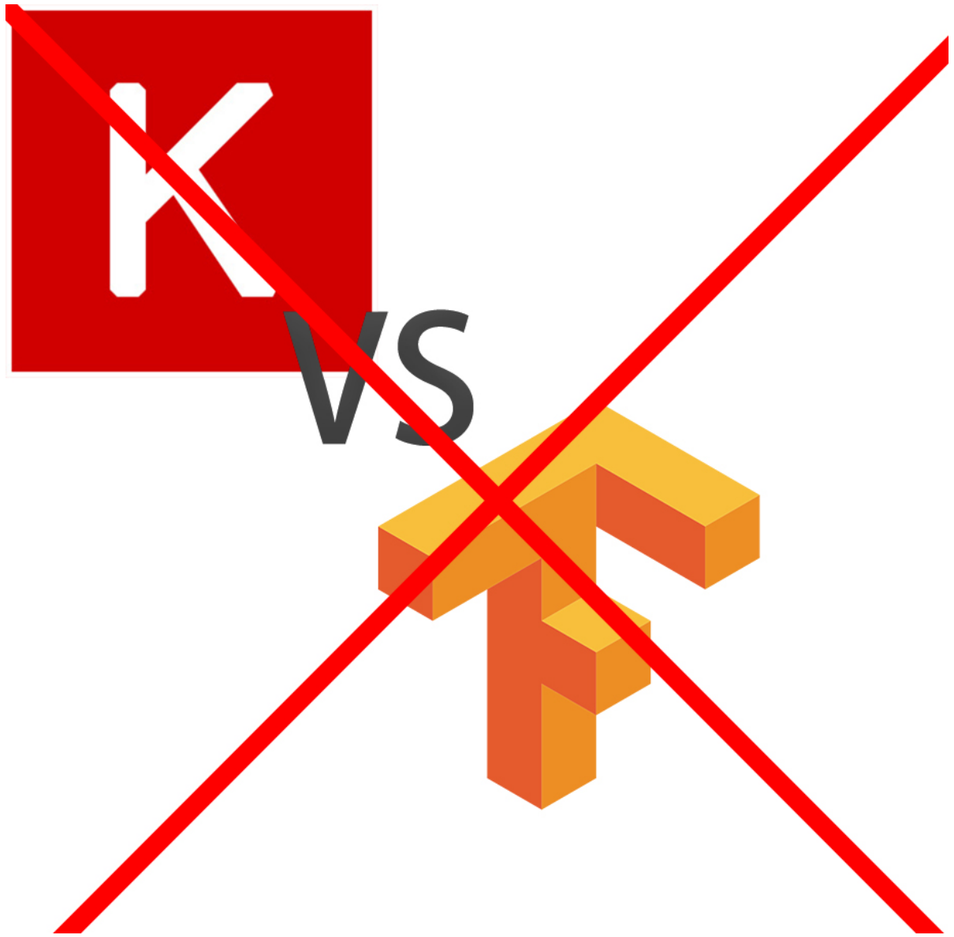
Why use Keras?

- Fast industry-ready models
- For beginners and experts
- Less code
- Build any architecture
- Deploy models in multiple platforms

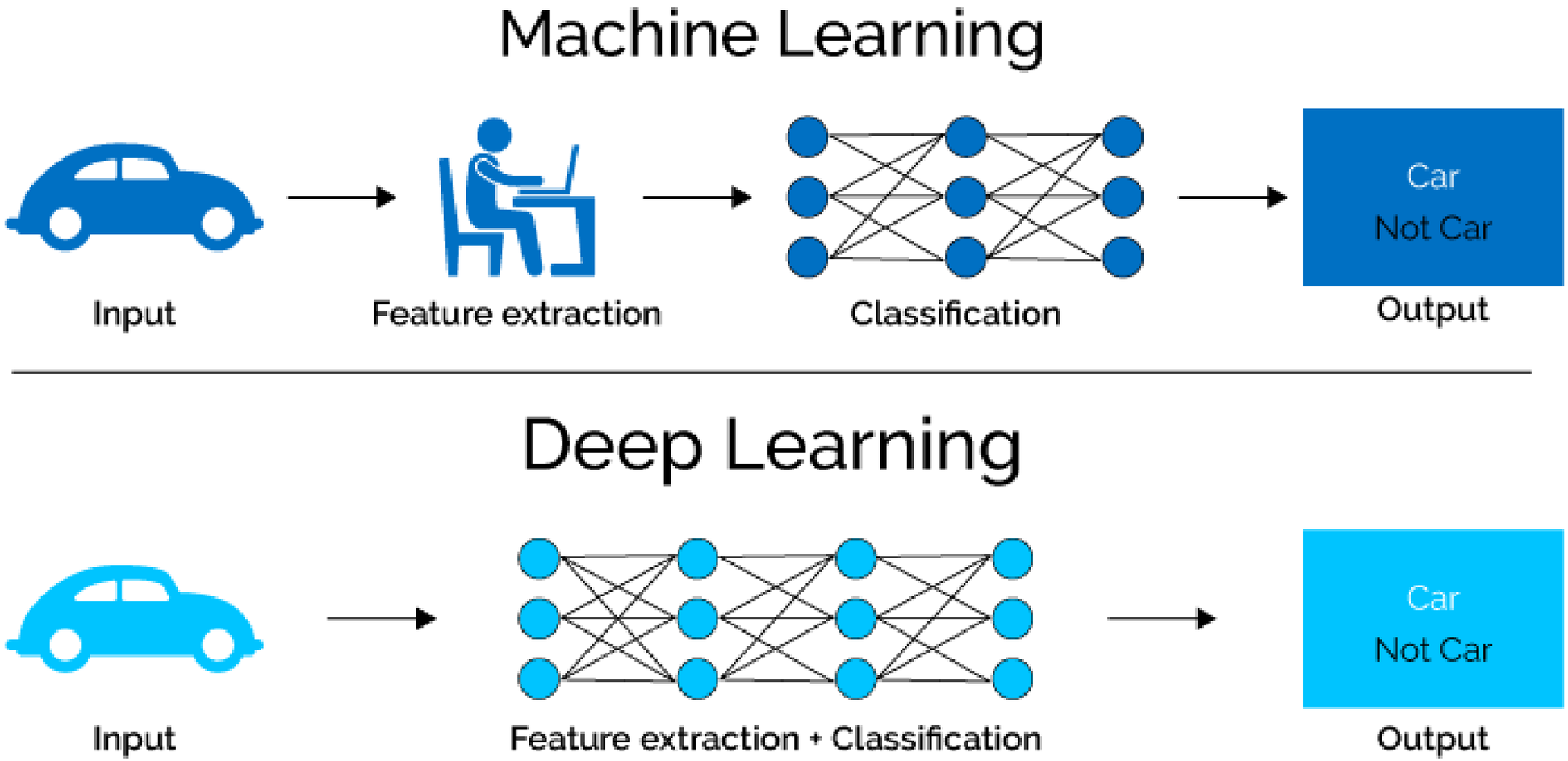


Keras + TensorFlow

- TensorFlow's high level framework of choice
- Keras is complementary to TensorFlow
- You can use TensorFlow for low level features

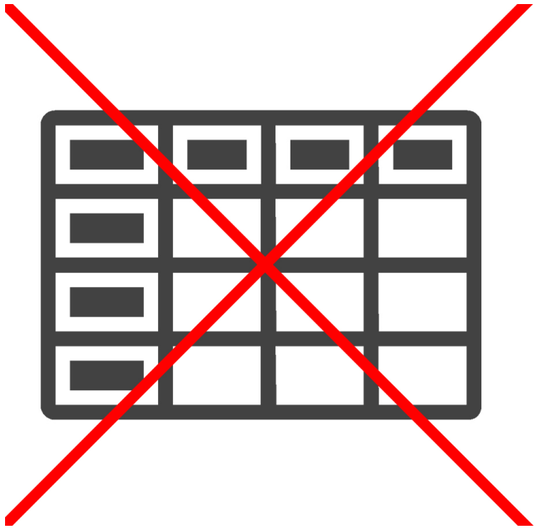


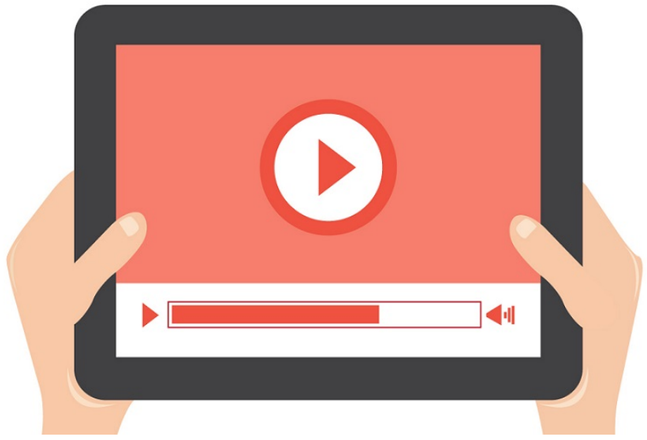
Feature Engineering



[1] Towards Data Science

Unstructured data



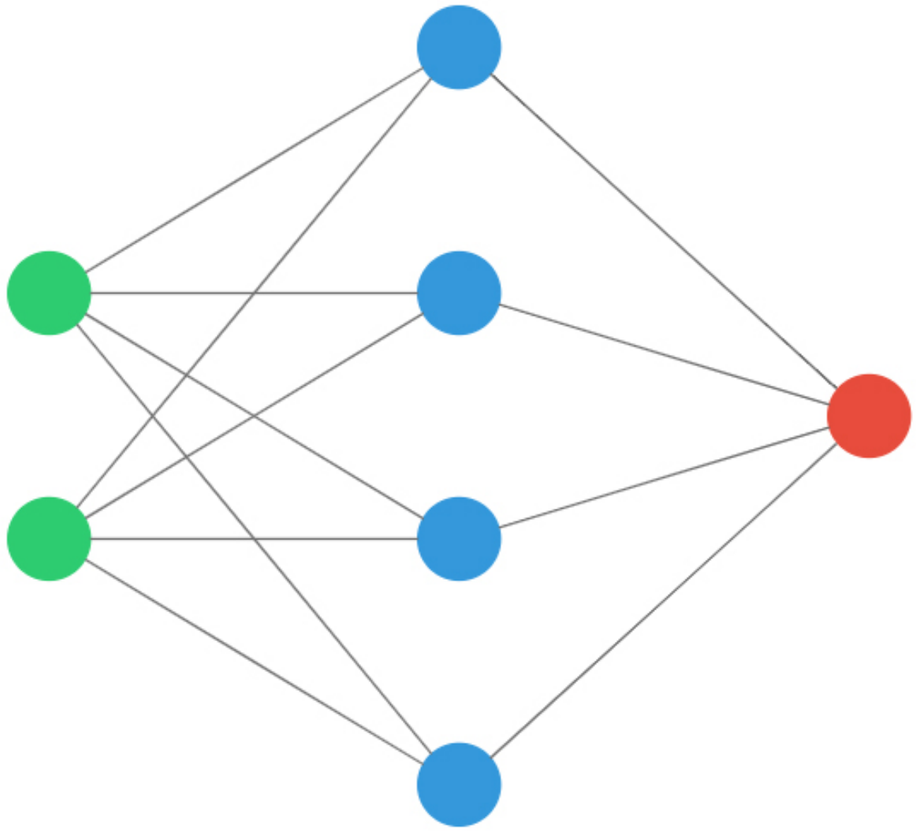


So, when to use neural networks?

- Dealing with unstructured data
- Don't need easily interpretable results
- You can benefit from a known architecture

Example: Classify images of cats and dogs

- **Images -> Unstructured data**
- You don't care about why the network knows it's a cat or a dog
- You can benefit from convolutional neural networks



Let's practice!

INTRODUCTION TO DEEP LEARNING WITH KERAS

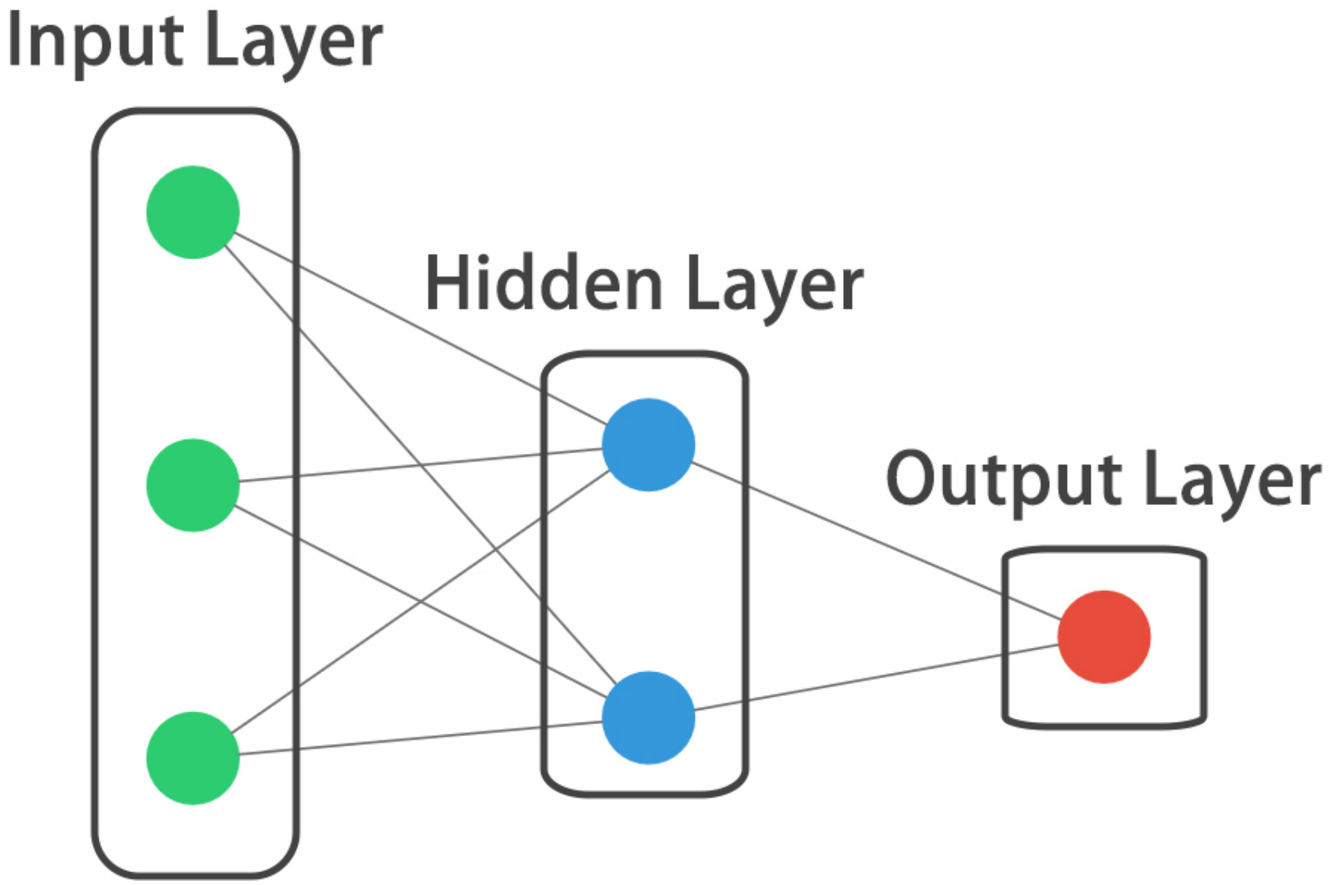
Your first neural network

INTRODUCTION TO DEEP LEARNING WITH KERAS

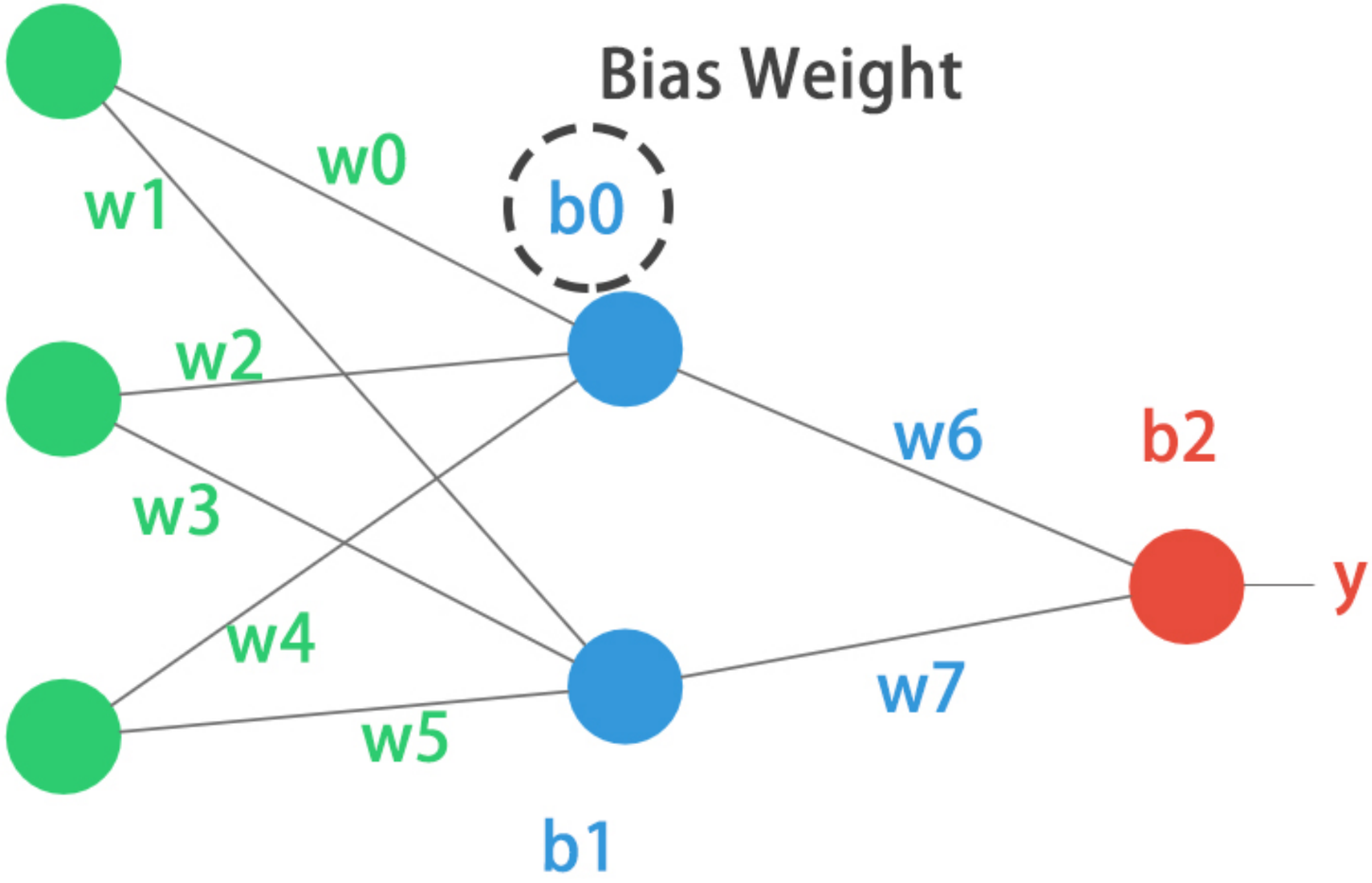
Miguel Esteban
Data Scientist & Founder

datacamp

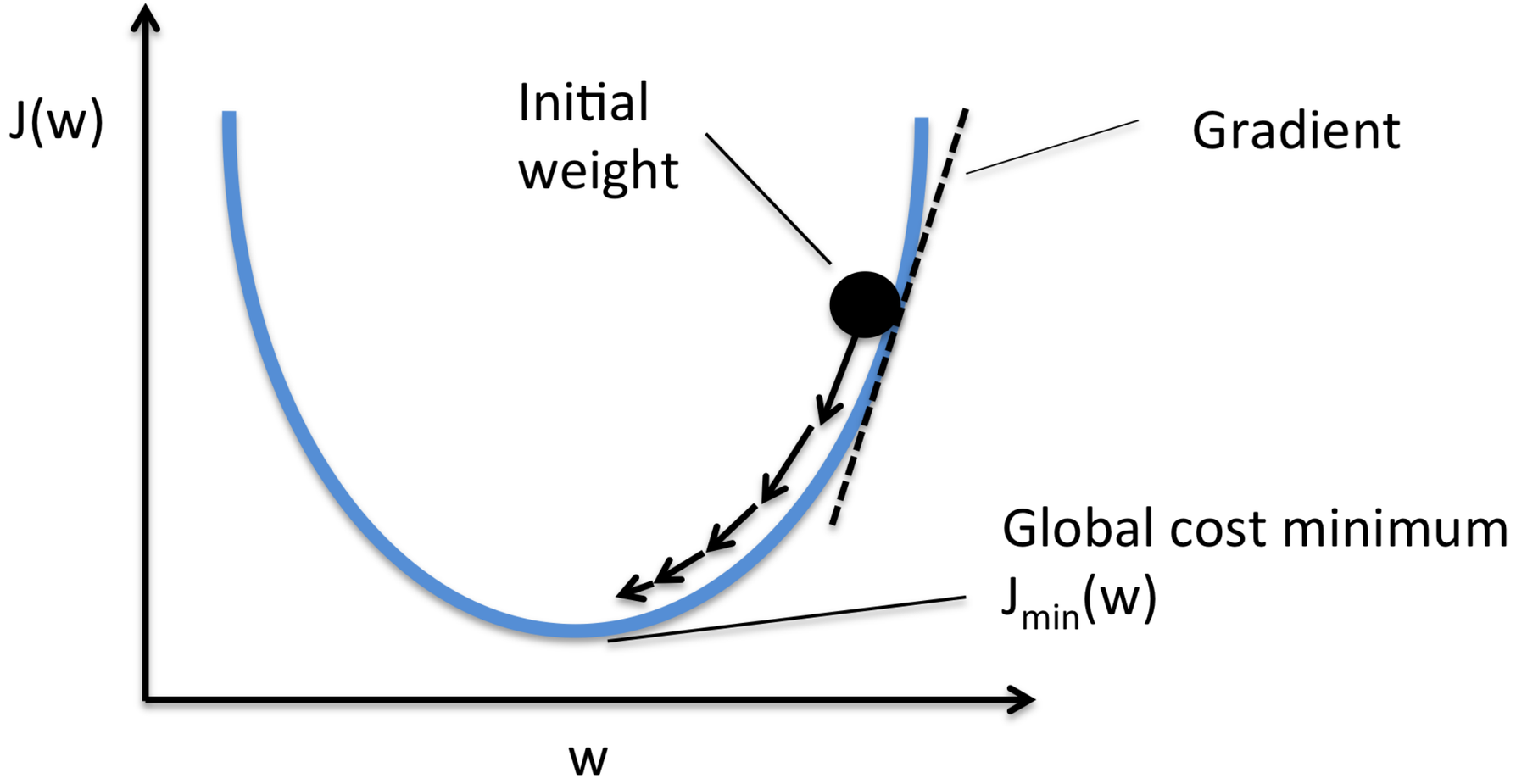
A neural network?



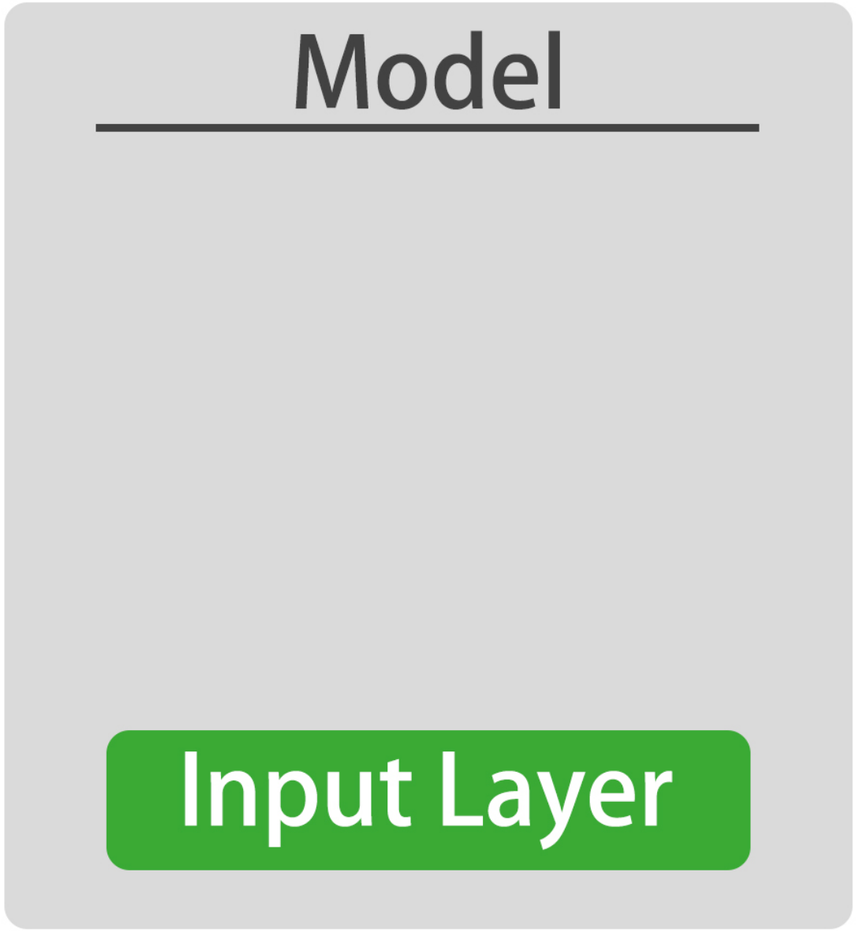
Parameters



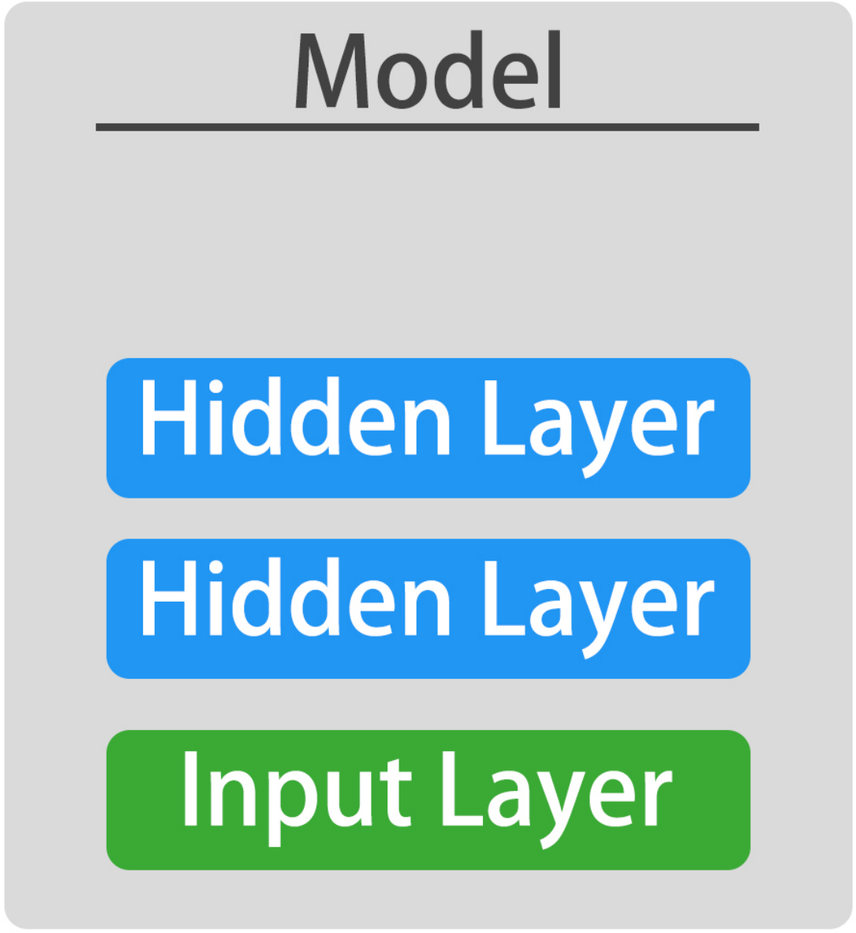
Gradient descent



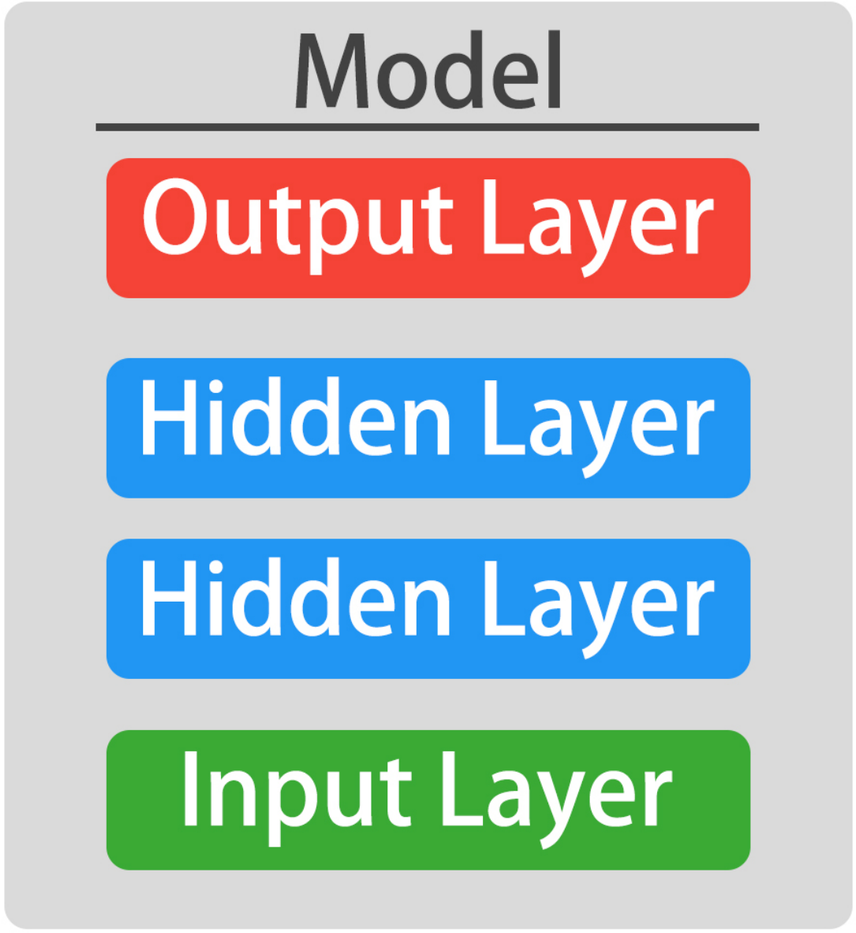
The sequential API



The sequential API

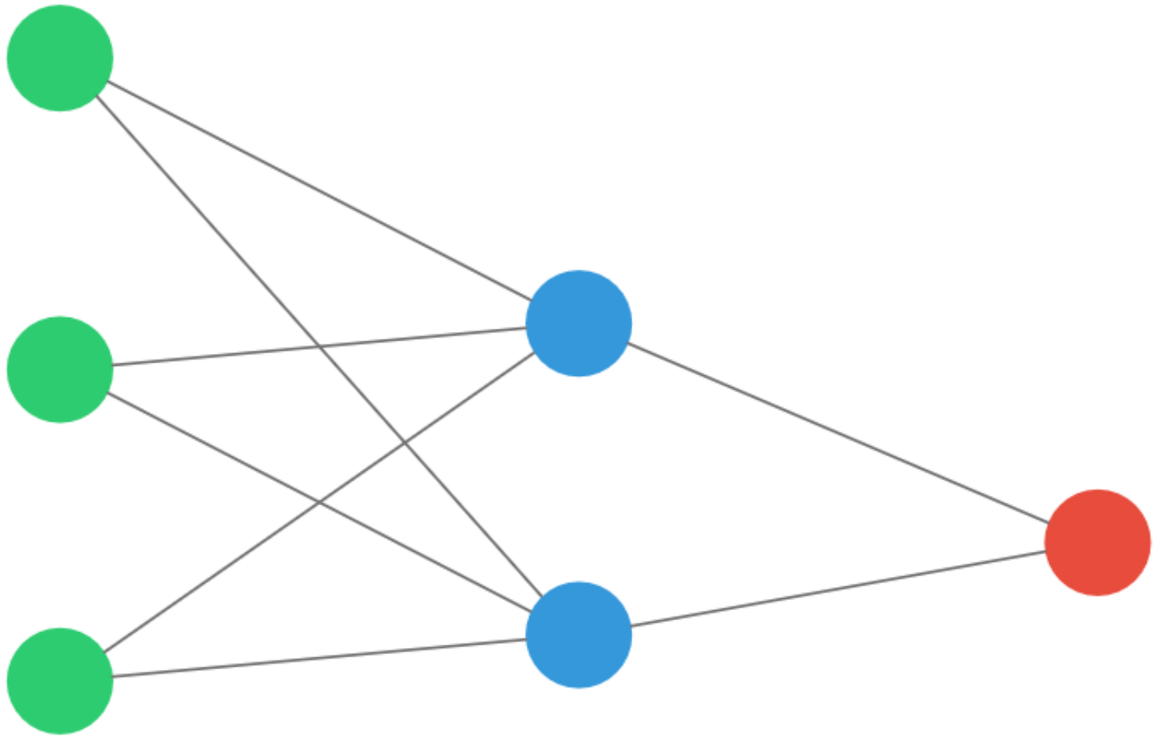


The sequential API



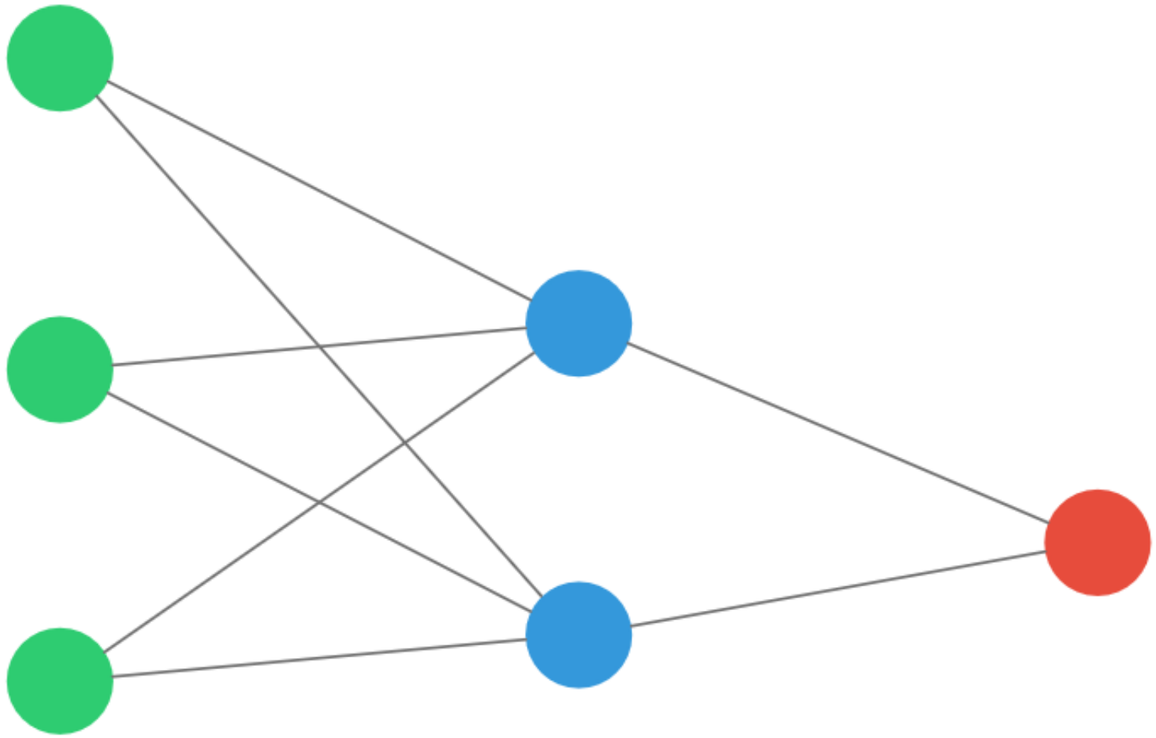
Defining a neural network

```
from keras.models import Sequential
from keras.layers import Dense
# Create a new sequential model
model = Sequential()
# Add and input and dense layer
model.add(Dense(2, input_shape=(3,)))
# Add a final 1 neuron layer
model.add(Dense(1))
```



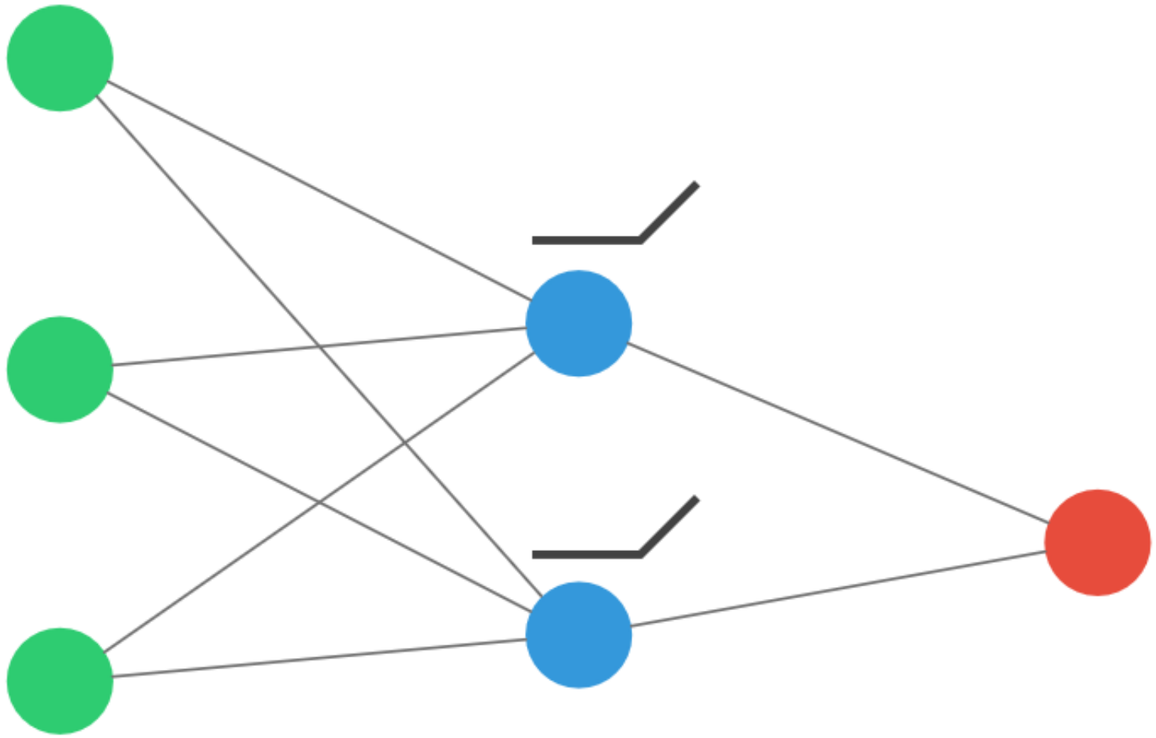
Adding activations

```
from keras.models import Sequential
from keras.layers import Dense
# Create a new sequential model
model = Sequential()
# Add and input and dense layer
model.add(Dense(2, input_shape=(3,)))
# Add a final 1 neuron layer
model.add(Dense(1))
```



Adding activations

```
from keras.models import Sequential
from keras.layers import Dense
# Create a new sequential model
model = Sequential()
# Add and input and dense layer
model.add(Dense(2, input_shape=(3,),
                activation="relu"))
# Add a final 1 neuron layer
model.add(Dense(1))
```

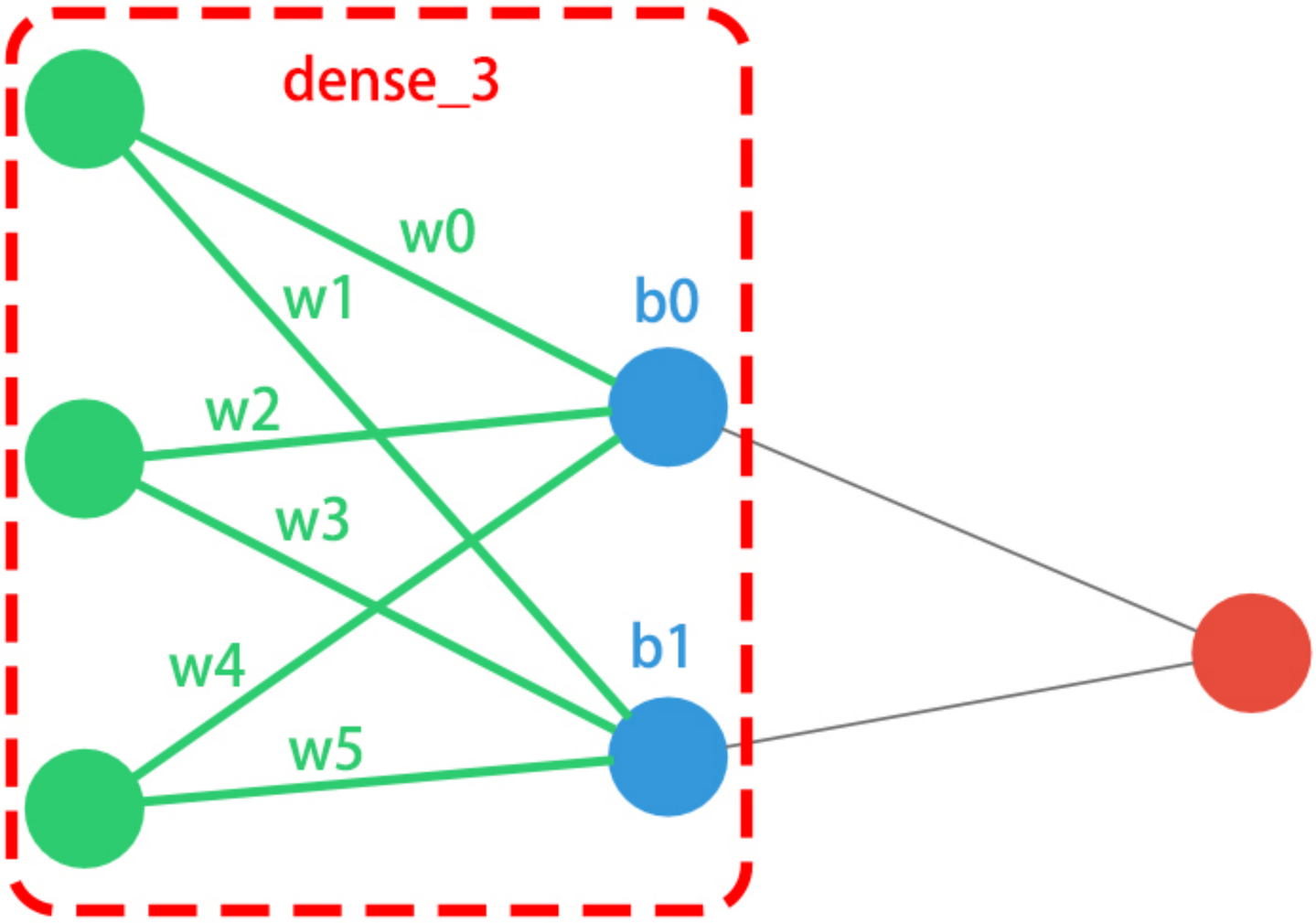


Summarize your model!

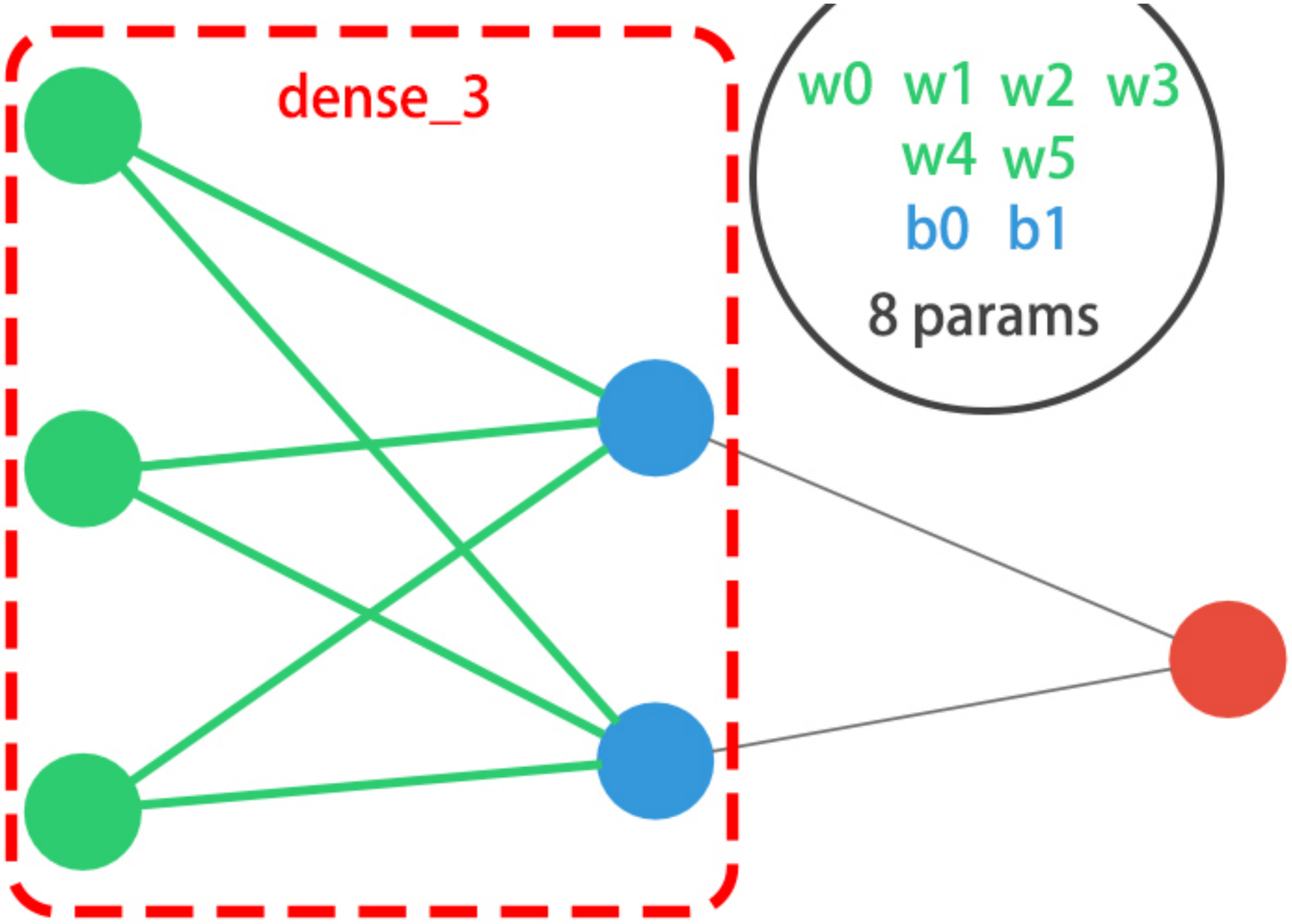
```
model.summary()
```

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 2)                 8
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 3
=================================================================
Total params: 11
Trainable params: 11
Non-trainable params: 0
_________________________________________________________________
```

Visualize parameters



Visualize parameters



Summarize your model!

```
model.summary()
```

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 2)                 --> 8 <--
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 3
=================================================================
Total params: 11
Trainable params: 11
Non-trainable params: 0
_________________________________________________________________
```

Let's code!

INTRODUCTION TO DEEP LEARNING WITH KERAS

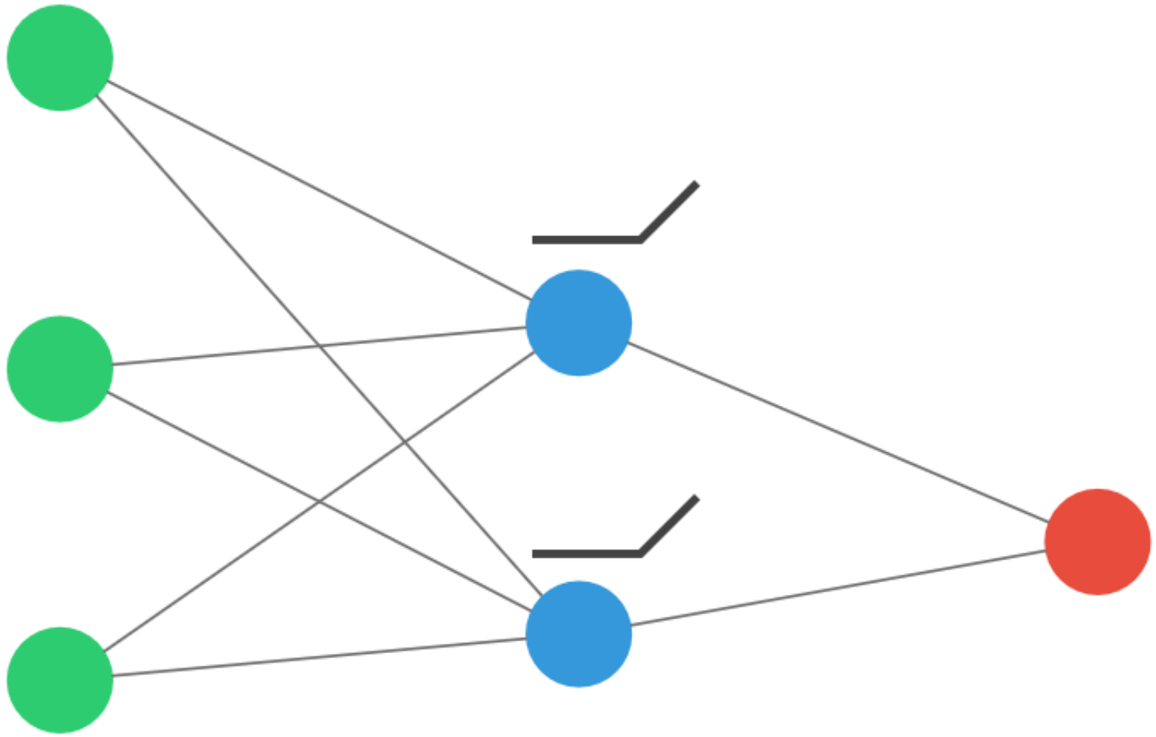
Surviving a meteor strike

INTRODUCTION TO DEEP LEARNING WITH KERAS

Miguel Esteban
Data Scientist & Founder

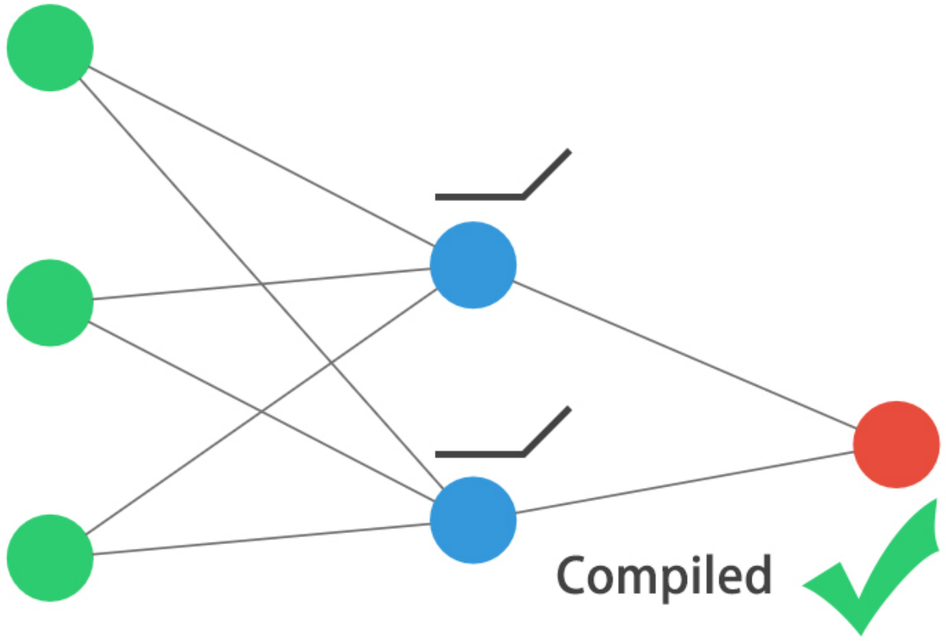
Recap

```
from keras.models import Sequential
from keras.layers import Dense

# Create a new sequential model
model = Sequential()
# Add and input and dense layer
model.add(Dense(2, input_shape=(3,),
                activation="relu"))

# Add a final 1 neuron layer
model.add(Dense(1))
<
```



Compiling

```
# Compiling your previously built model
model.compile(optimizer="adam", loss="mse")
```



Training

```
# Train your model
model.fit(X_train, y_train, epochs=5)
```

```
Epoch 1/5
1000/1000 [==============================] - 0s 242us/step - loss: 0.4090
Epoch 2/5
1000/1000 [==============================] - 0s 34us/step - loss: 0.3602
Epoch 3/5
1000/1000 [==============================] - 0s 37us/step - loss: 0.3223
Epoch 4/5
1000/1000 [==============================] - 0s 34us/step - loss: 0.2958
Epoch 5/5
1000/1000 [==============================] - 0s 33us/step - loss: 0.2795
```

Predicting

```
# Predict on new data
preds = model.predict(X_test)

# Look at the predictions
print(preds)
```

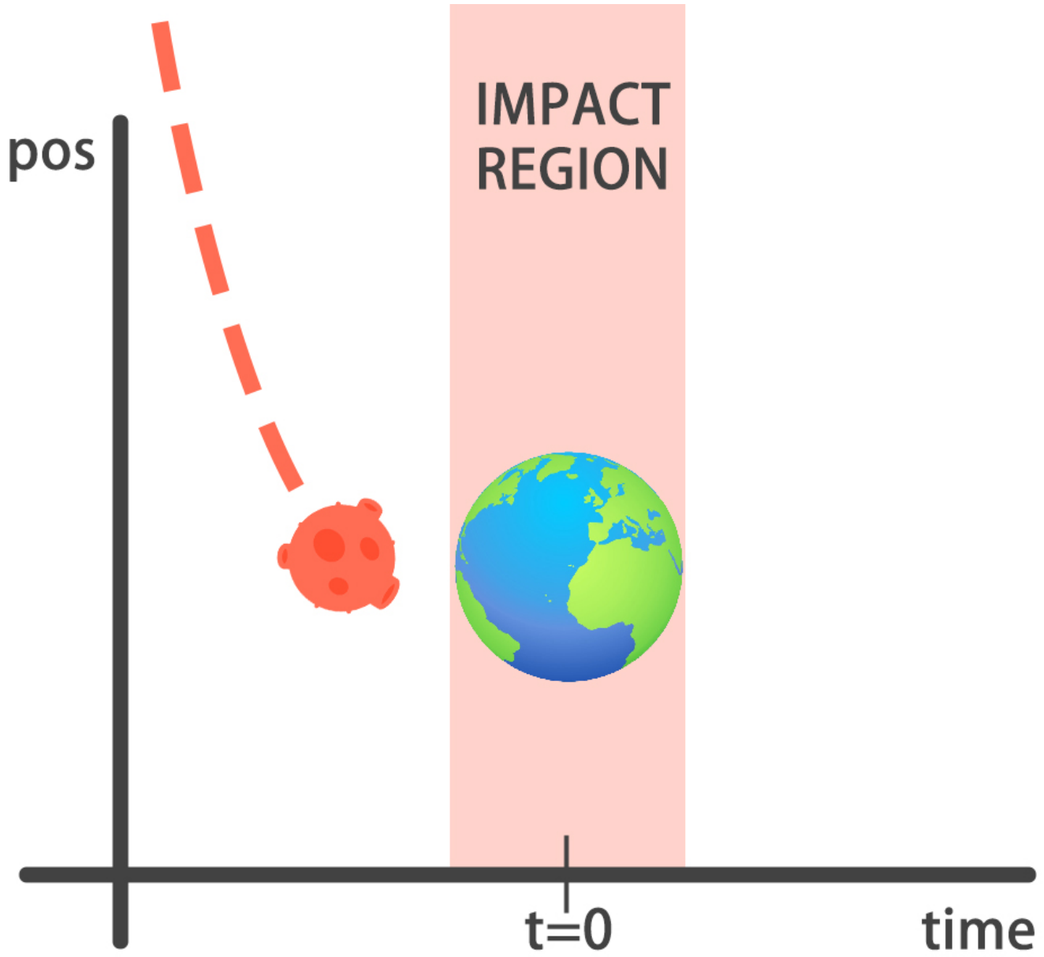
```
array([[0.6131608 ],
       [0.5175948 ],
       [0.60209155],
       ...,
       [0.55633   ],
       [0.5305591 ],
       [0.50682044]])
```

Evaluating

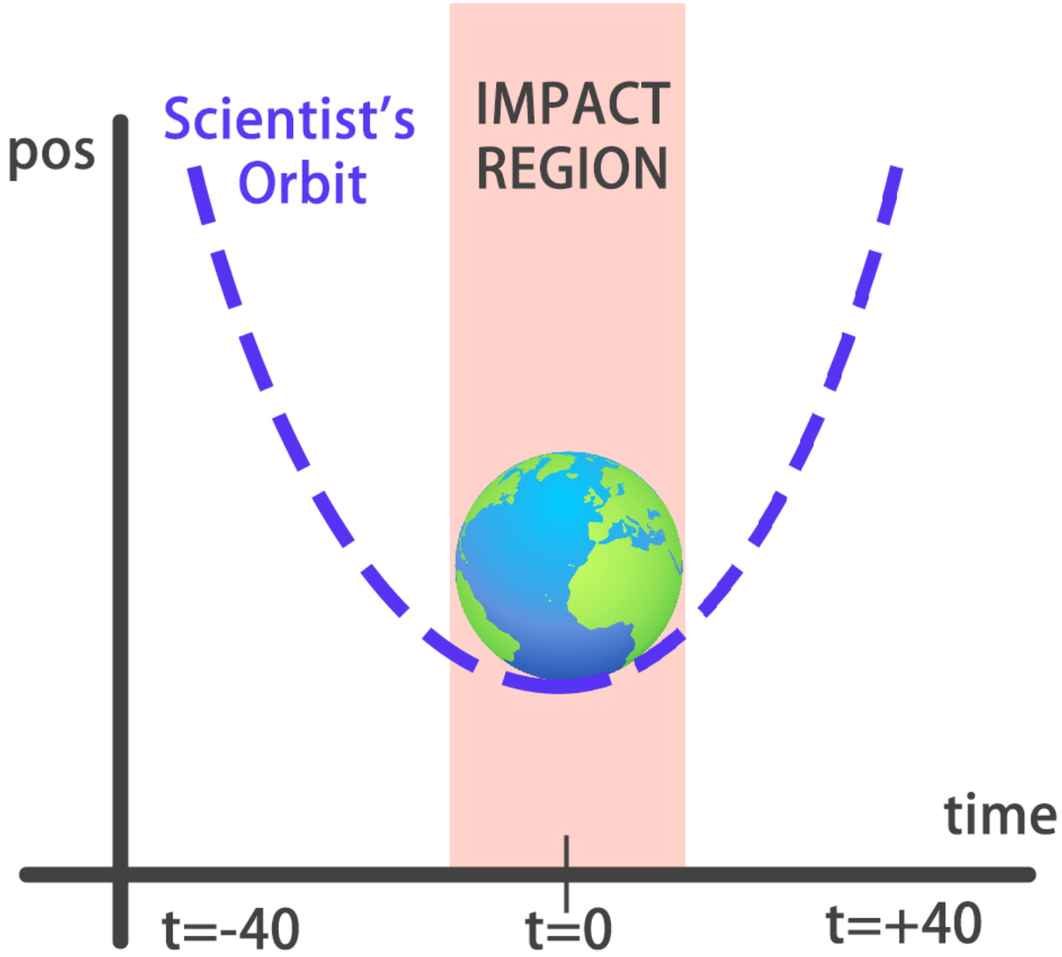
```
# Evaluate your results
model.evaluate(X_test, y_test)
```

```
1000/1000 [==============================] - 0s 53us/step
0.25
```

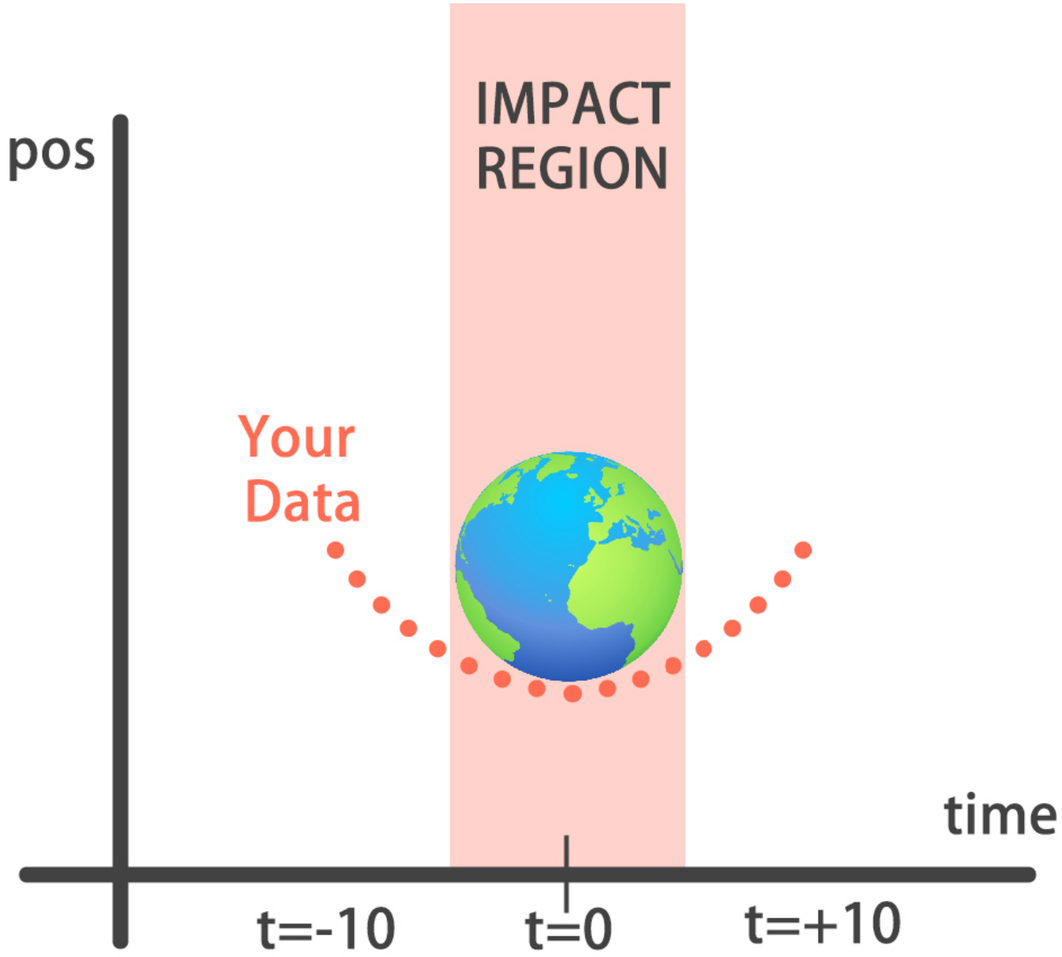
The problem at hand



Scientific prediction



Your task



Let's save the earth!

INTRODUCTION TO DEEP LEARNING WITH KERAS