Artificial Intelligence II

Lesson 8 - Deep Learning

First Code
Academy

Today’s Plan

Teach Back	00 - 5 min
Convolutional Neural Networks	10 - 15 min
Facial Recognition	15 - 20 min
Quiz	20-25 min
Break	25 - 28 min
Project - Recognize Handwritten Numbers	28 - 55 min
Lesson Recap	55 - 60 min

Key Terms

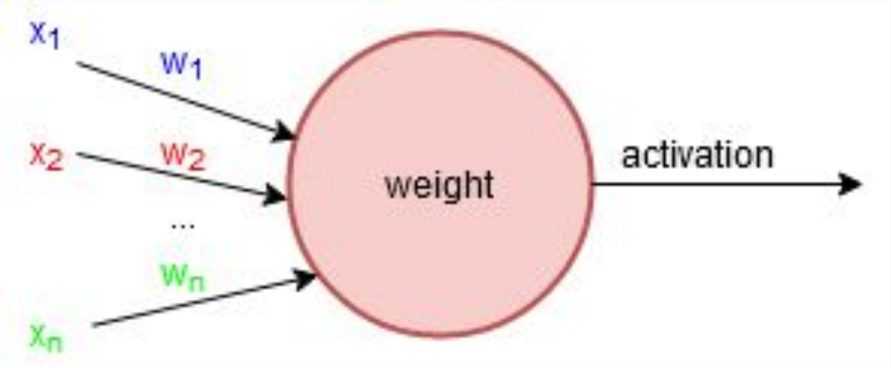
Convolutional Neural Networks

Facial Recognition

What did we learn last time?

Neuron

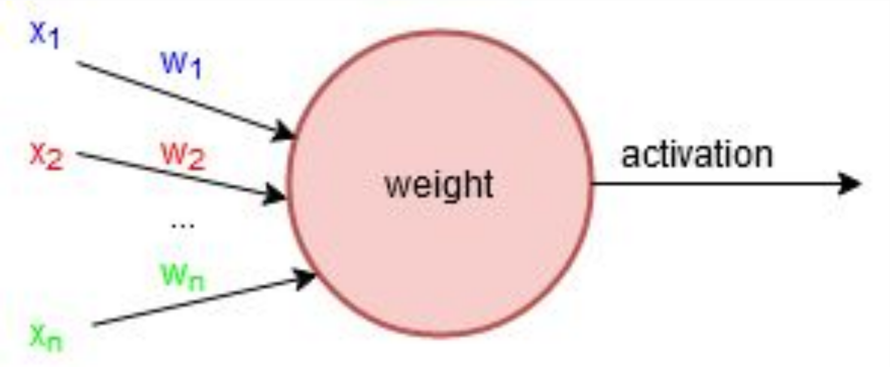
In AI, a “neuron” takes a sum of incoming values and multiplies each with a unique **weight**



The **activation** is how “sure” our neuron is

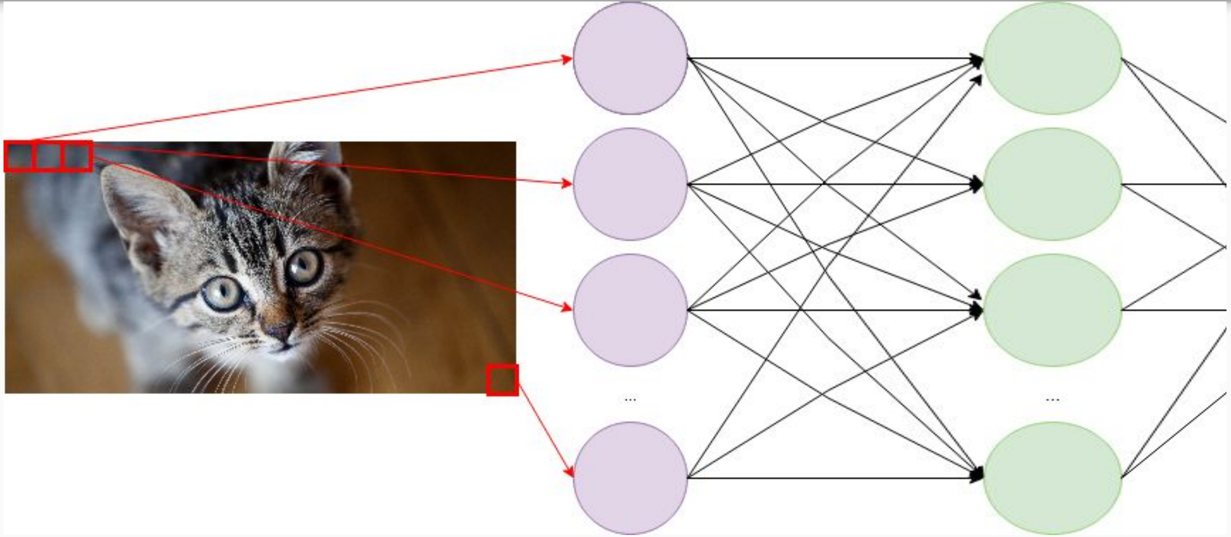
Neuron

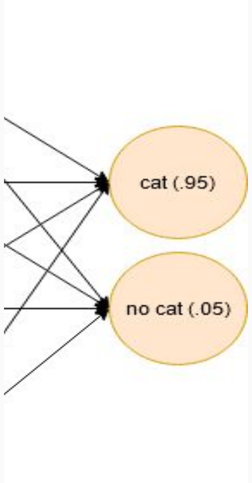
We first add all the inputs and their **weights**



$$sum = w_1x_1 + w_2x_2 + w_3x_3$$

Neural Networks



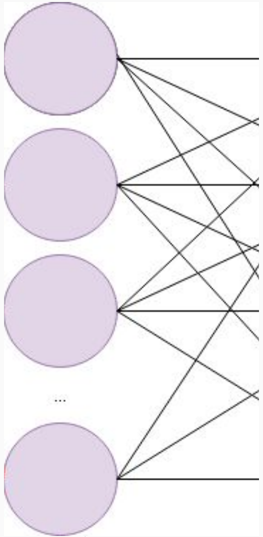


Input Layer takes in information from outside world

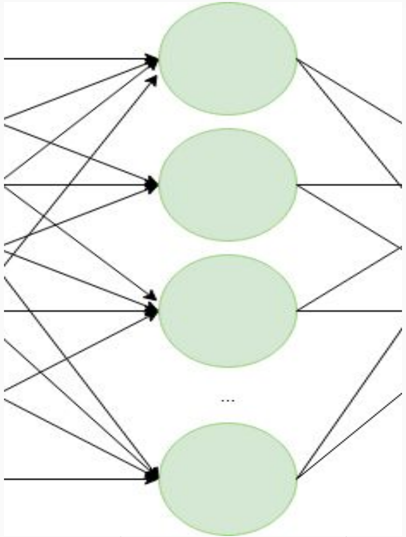
Hidden Layer(s) do extra processing if we need it.

Output Layer classifies the input and tells us how confident it is in the result.

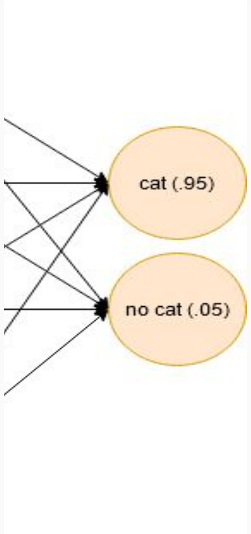
Backpropagation



Continue this way until the input layer.



Pass the error to the previous layer and calculate this layer's



First calculate the **error** in the output layer.

Deep Learning

Deep Learning

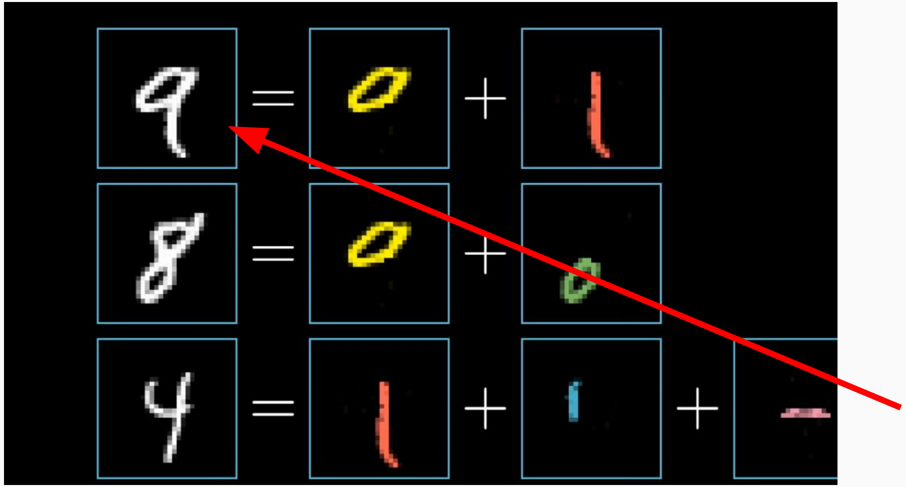
Deep learning just involves using a neural network with at least one **hidden layer**.

These networks sometimes use many hidden layers, so they become deep!

Deep learning helps us find more complicated **patterns in data**

Deep Learning

Why do we need many hidden layers?



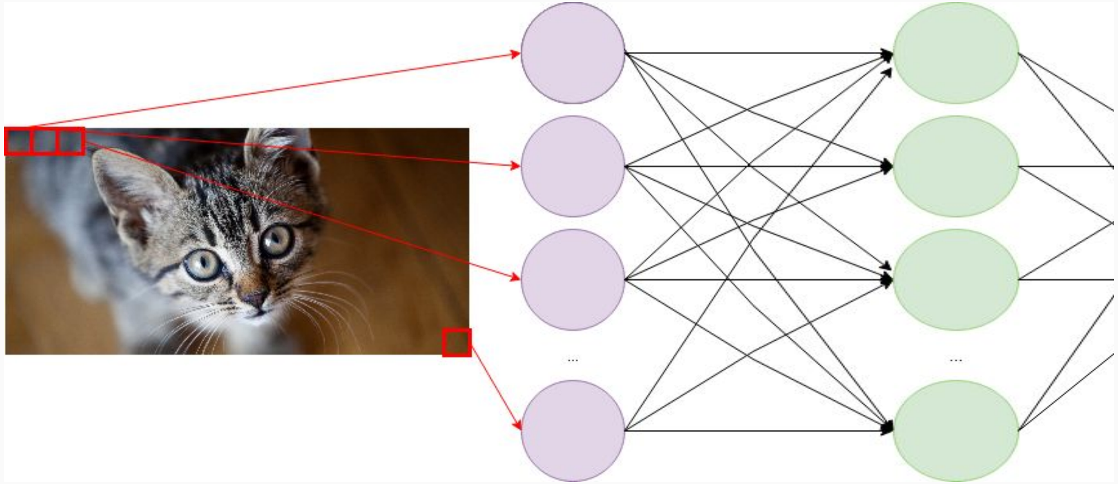
Detect a simple feature in each layer

Ex. “9” has a loop on top and a straight line going down

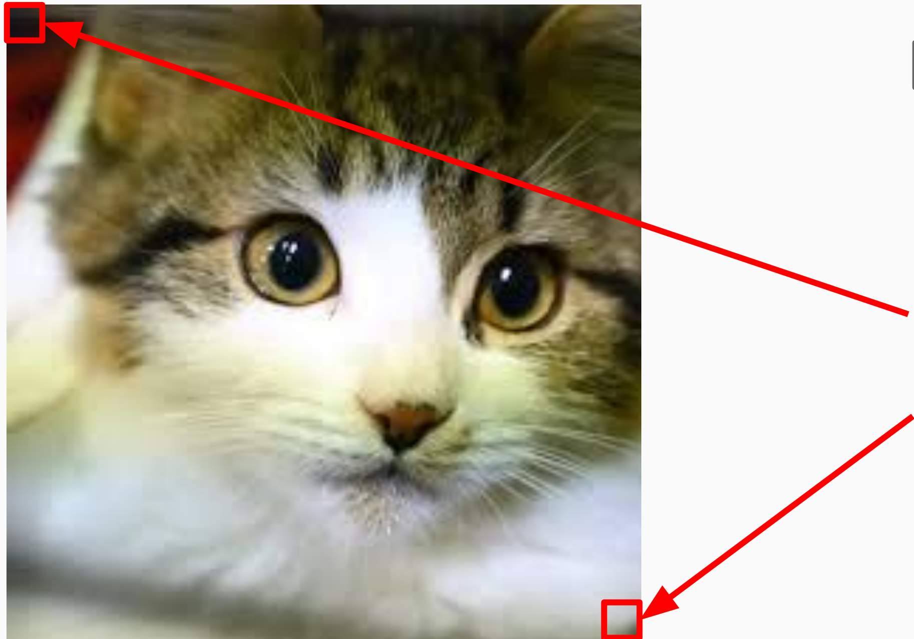
Convolutional Neural Networks

Convolutional Neural Networks

We learned that every neuron in the input layer was connected to ever neuron in the next layer.



Convolutional Neural Networks



However….

These two regions of the image don’t really influence each other!

Convolutional Neural Networks

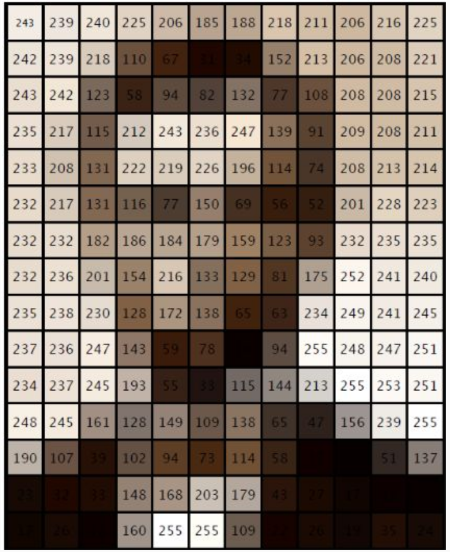
To make our model smaller, we have an input neuron take in an entire **region** of the input image.

This works since pixels that are nearby usually influence each other a lot.

Convolutional Neural Networks

It makes finding patterns easier, since the images are simpler.

What do you think is in this image?



243	239	240	225	206	185	188	218	211	206	216	225
242	239	218	110	67	31	34	152	213	206	208	221
243	242	123	58	94	82	132	77	108	208	208	215
235	217	115	212	243	236	247	139	91	209	208	211
233	208	131	222	219	226	196	114	74	208	213	214
232	217	131	116	77	150	69	56	52	201	228	223
232	232	182	186	184	179	159	123	93	232	235	235
232	236	201	154	216	133	129	81	175	252	241	240
235	238	230	128	172	138	65	63	234	249	241	245
237	236	247	143	59	78	[illegible]	94	255	248	247	251
234	237	245	193	55	33	115	144	213	255	253	251
248	245	161	128	149	109	138	65	47	156	239	255
190	107	39	102	94	73	114	58	[illegible]	[illegible]	51	137
23	32	33	148	168	203	179	43	27	17	[illegible]	[illegible]
17	26	[illegible]	160	255	255	109	22	26	19	35	24

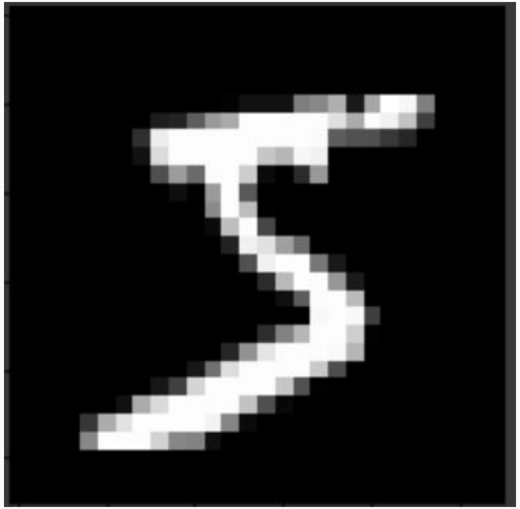
Common patterns still remain even if we simplify the image a bit!

Quiz: bit.ly/FCA_Quiz_AI

Project : Number Classification

Number Classification

How can we make our computer learn what number this is?



Use deep learning!

Google Colab

Google colab allows us to run Python notebooks using AI.

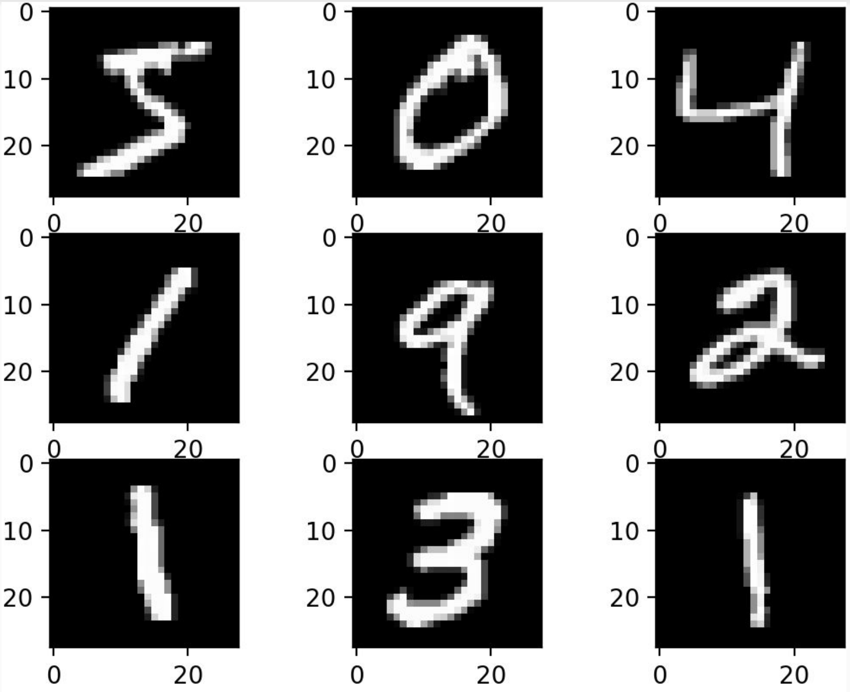
No need to have a powerful computer for simple projects!

https://colab.research.google.com

MNIST Dataset

The MNIST dataset contains thousands of handwritten numbers.

Many people train their AI models on these images.



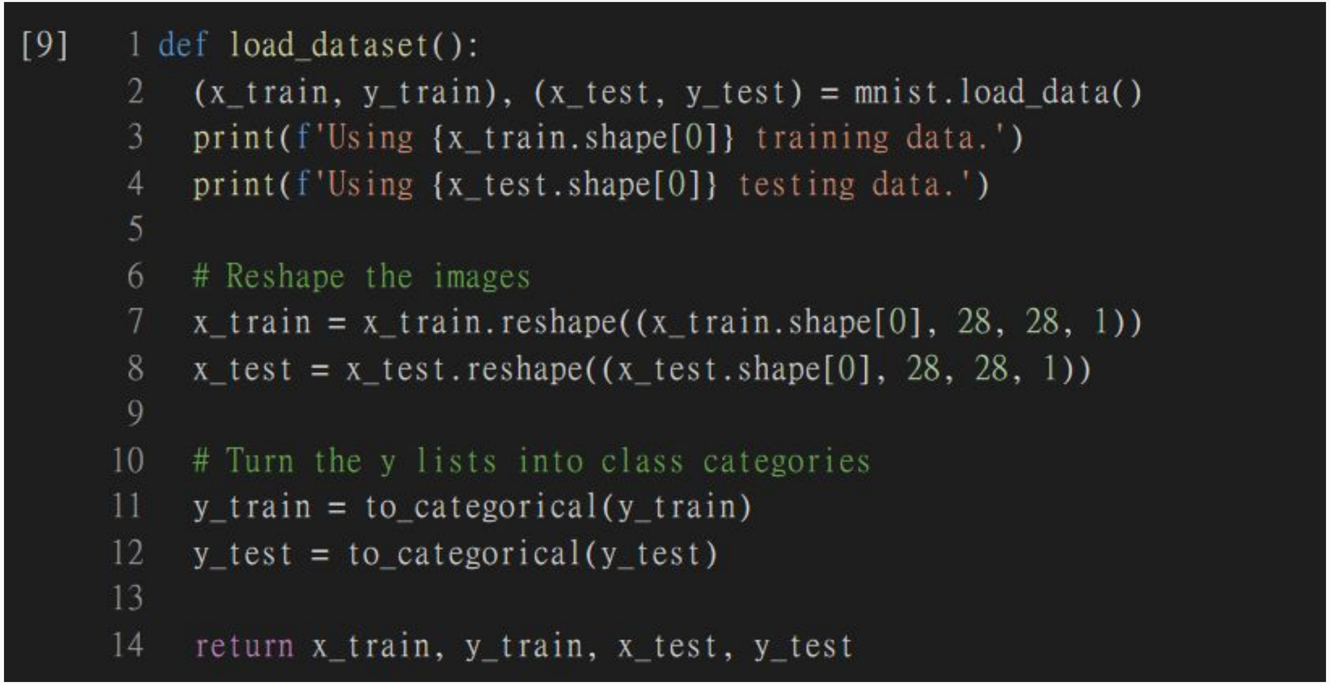
Get the starter file

http://bit.ly/FCA_AI2_Starter

Project Overview

1. Load the data
2. Do some preparation on the images
3. Create our model and train it
4. Make a graph with our error

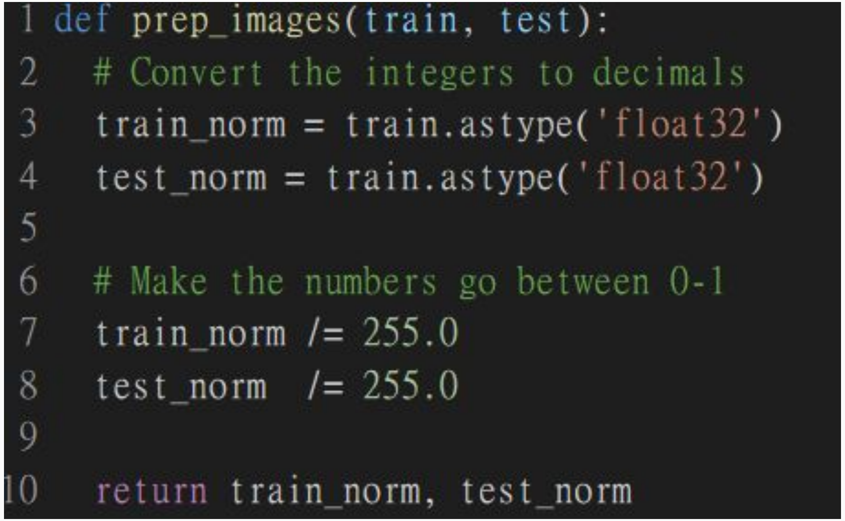
Load the Dataset



```python
def load_dataset():
  (x_train, y_train), (x_test, y_test) = mnist.load_data()
  print(f'Using {x_train.shape[0]} training data.')
  print(f'Using {x_test.shape[0]} testing data.')

  # Reshape the images
  x_train = x_train.reshape((x_train.shape[0], 28, 28, 1))
  x_test = x_test.reshape((x_test.shape[0], 28, 28, 1))

  # Turn the y lists into class categories
  y_train = to_categorical(y_train)
  y_test = to_categorical(y_test)

  return x_train, y_train, x_test, y_test
```

Prepare the Images

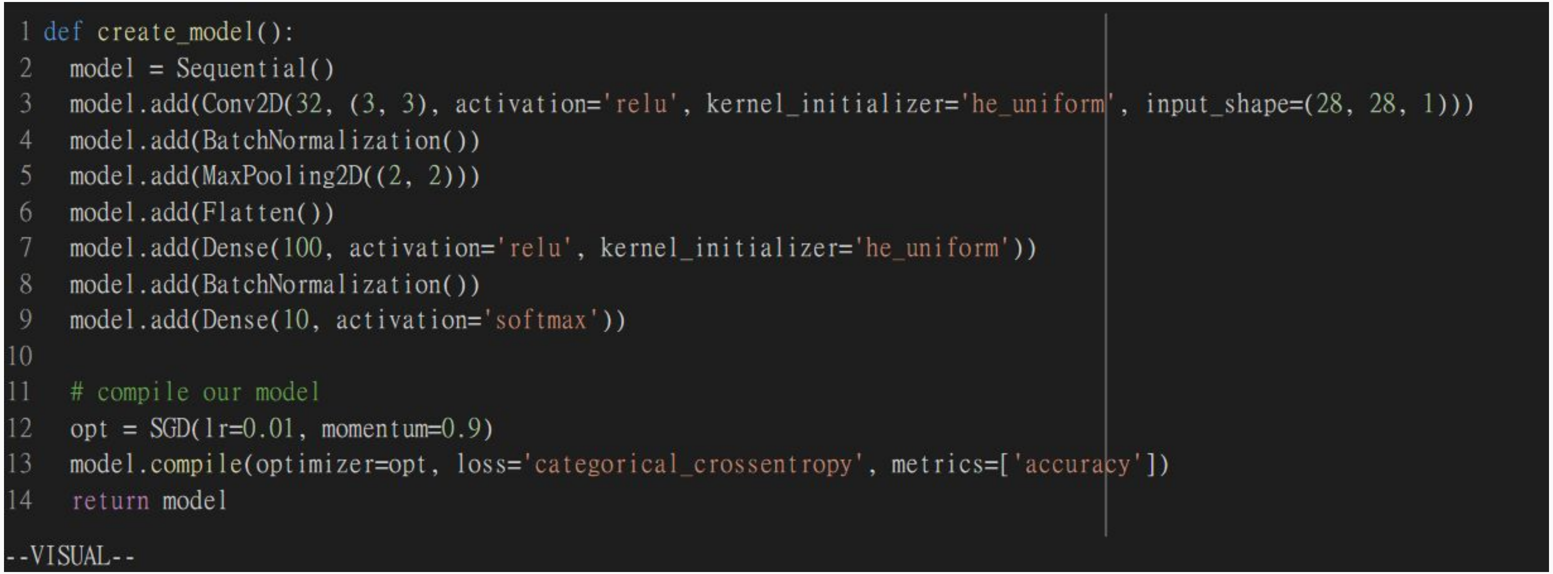
Before training our model, we have to change our image format slightly.



```python
def prep_images(train, test):
  # Convert the integers to decimals
  train_norm = train.astype('float32')
  test_norm = train.astype('float32')

  # Make the numbers go between 0-1
  train_norm /= 255.0
  test_norm  /= 255.0

  return train_norm, test_norm
```

Create our model

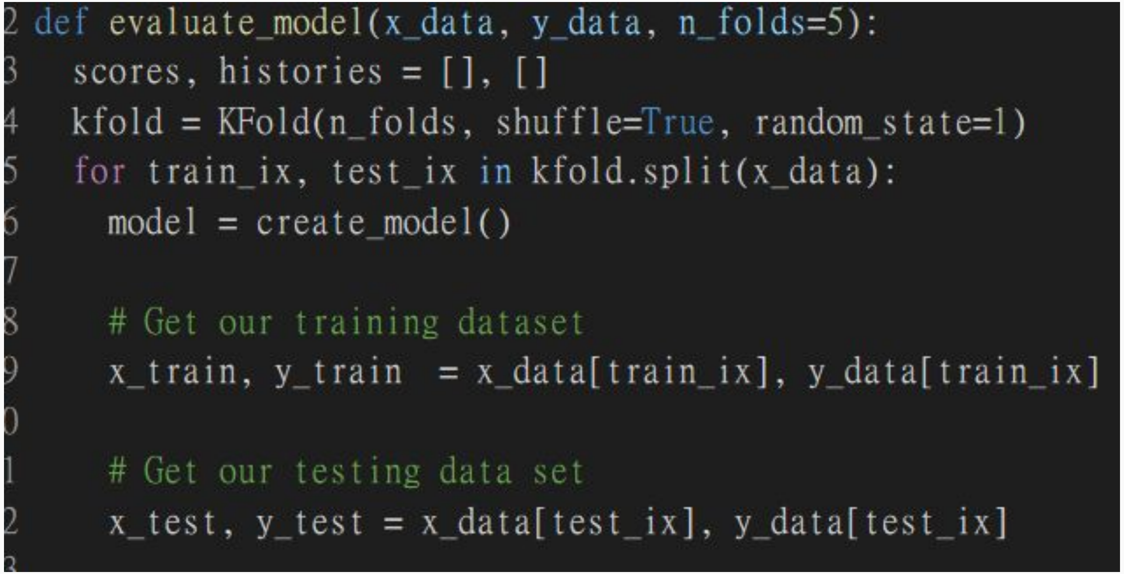
Add all the layers to our model!



```
def create_model():
  model = Sequential()
  model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', input_shape=(28, 28, 1)))
  model.add(BatchNormalization())
  model.add(MaxPooling2D((2, 2)))
  model.add(Flatten())
  model.add(Dense(100, activation='relu', kernel_initializer='he_uniform'))
  model.add(BatchNormalization())
  model.add(Dense(10, activation='softmax'))

  # compile our model
  opt = SGD(lr=0.01, momentum=0.9)
  model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
  return model

--VISUAL--
```

Training our Model

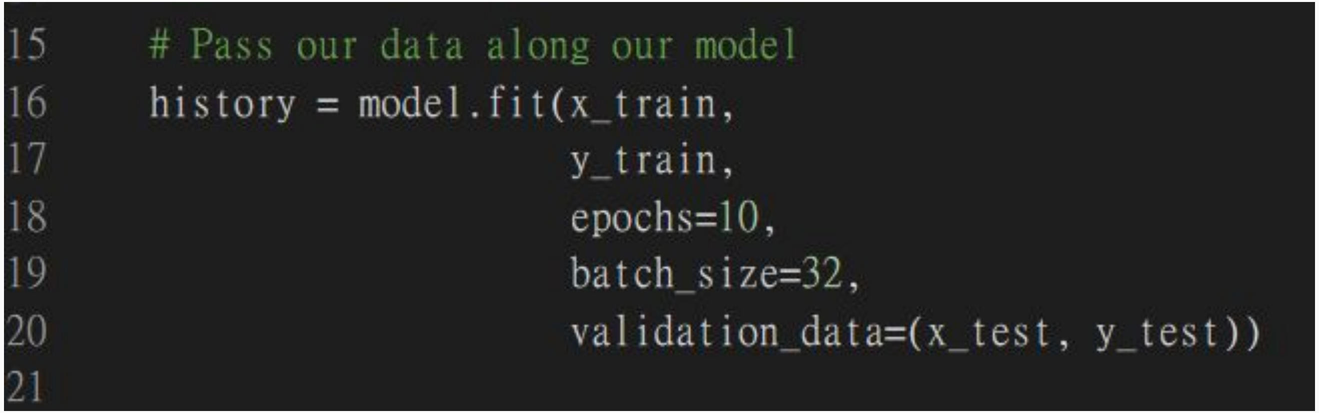
We train our model using **k-folds** method, which splits training data into various segments and uses each segment as a small “testing” set.



```
def evaluate_model(x_data, y_data, n_folds=5):
  scores, histories = [], []
  kfold = KFold(n_folds, shuffle=True, random_state=1)
  for train_ix, test_ix in kfold.split(x_data):
    model = create_model()

    # Get our training dataset
    x_train, y_train  = x_data[train_ix], y_data[train_ix]

    # Get our testing data set
    x_test, y_test = x_data[test_ix], y_data[test_ix]
```

Fitting the Data

Next, we pass our data along to our model.

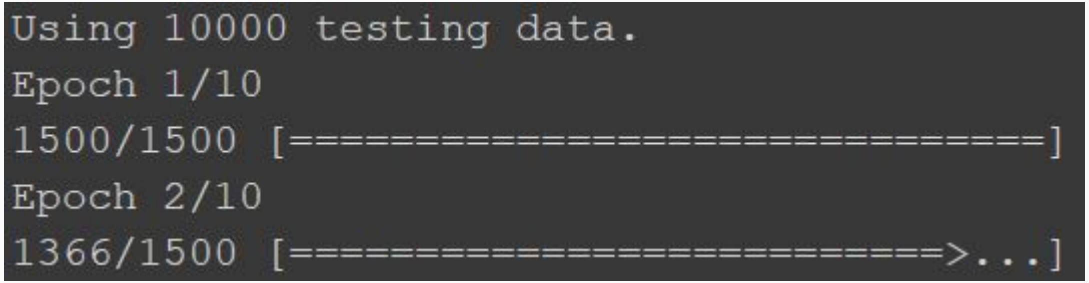


```
    # Pass our data along our model
    history = model.fit(x_train,
                        y_train,
                        epochs=10,
                        batch_size=32,
                        validation_data=(x_test, y_test))

```

Run It!

You’ll have to wait a while for it to train….

An **epoch** is an iteration over the entire set, so we repeat the training many many times.



```
Using 10000 testing data.
Epoch 1/10
1500/1500 [==============================]
Epoch 2/10
1366/1500 [==========================>...]
```

Recap of Key Terms

Key Terms

Deep Learning

Convolutional Neural Networks

That’s it for today!

Artificial Intelligence II

Lesson 8 - Deep Learning

First Code
Academy