GDG Ogbomoso

Building A Machine Learning Model With Tensorflow To Classify Images

Hi!
My name is
OLOKUN ADEMOLA.

I’m a Data Science and Artificial Intelligence Enthusiast.

you can find me on twitter:
@introvert_olo

A little about me

- **Data Science and AI Beginner**
- **I hate snakes, but I play with Python and Anaconda**
- **Winner of Access Bank, AFF Data Hack 2018**
- **A product of AI Saturdays and Data Science Nigeria**
- **Game of Thrones freak!**

LET’S DIVE IN!

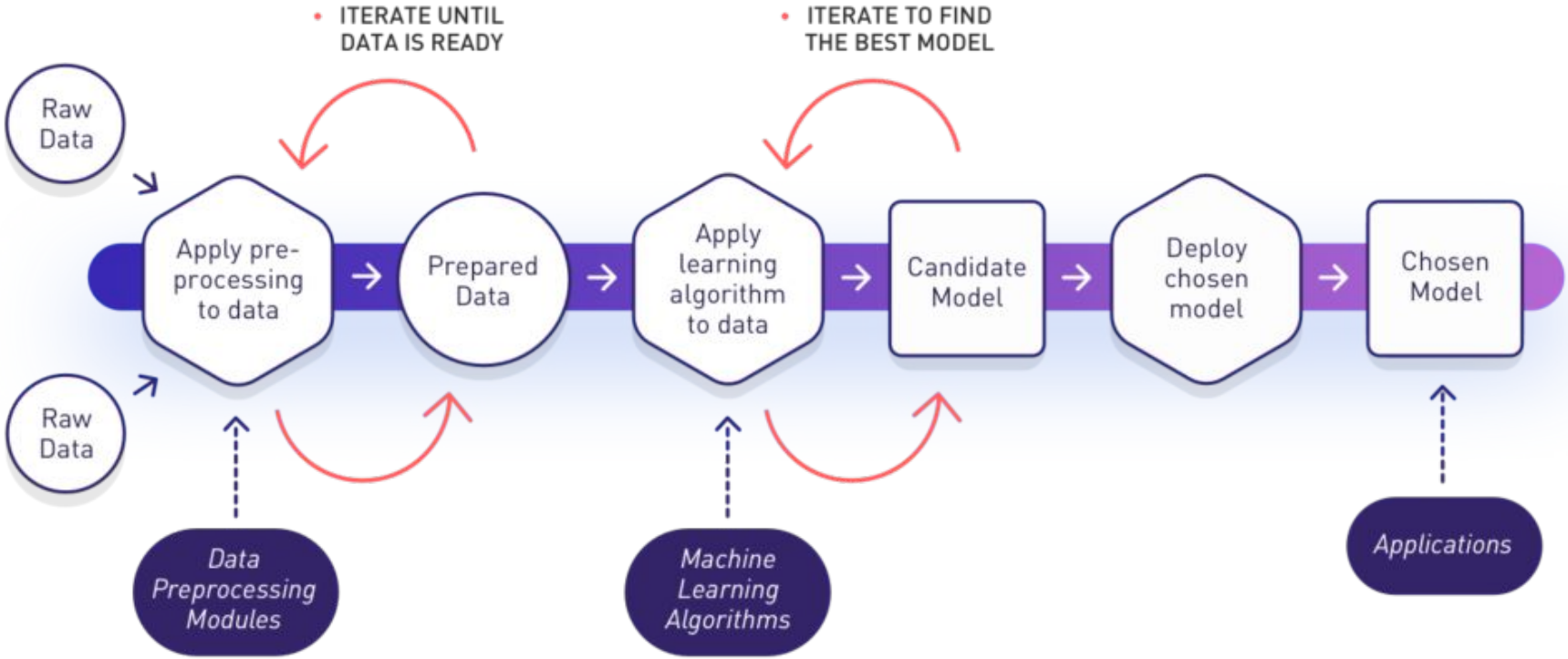

OOOOOH....
I"M READY, ARE YOU READY?

WHAT IS MACHINE LEARNING?

Machine Learning is the practice of using algorithms to parse data, learn from the data and use insights from the data to make a prediction about something without explicit programming.

The fundamental goal of Machine Learning is to successfully interpret data that has never been seen before.

The Process


ITERATE UNTIL DATA IS READY
ITERATE TO FIND THE BEST MODEL
Raw Data
Raw Data
Apply pre-processing to data
Prepared Data
Apply learning algorithm to data
Candidate Model
Deploy chosen model
Chosen Model
Data Preprocessing Modules
Machine Learning Algorithms
Applications

Opportunities of machine learning

Machine Learning can be split into:

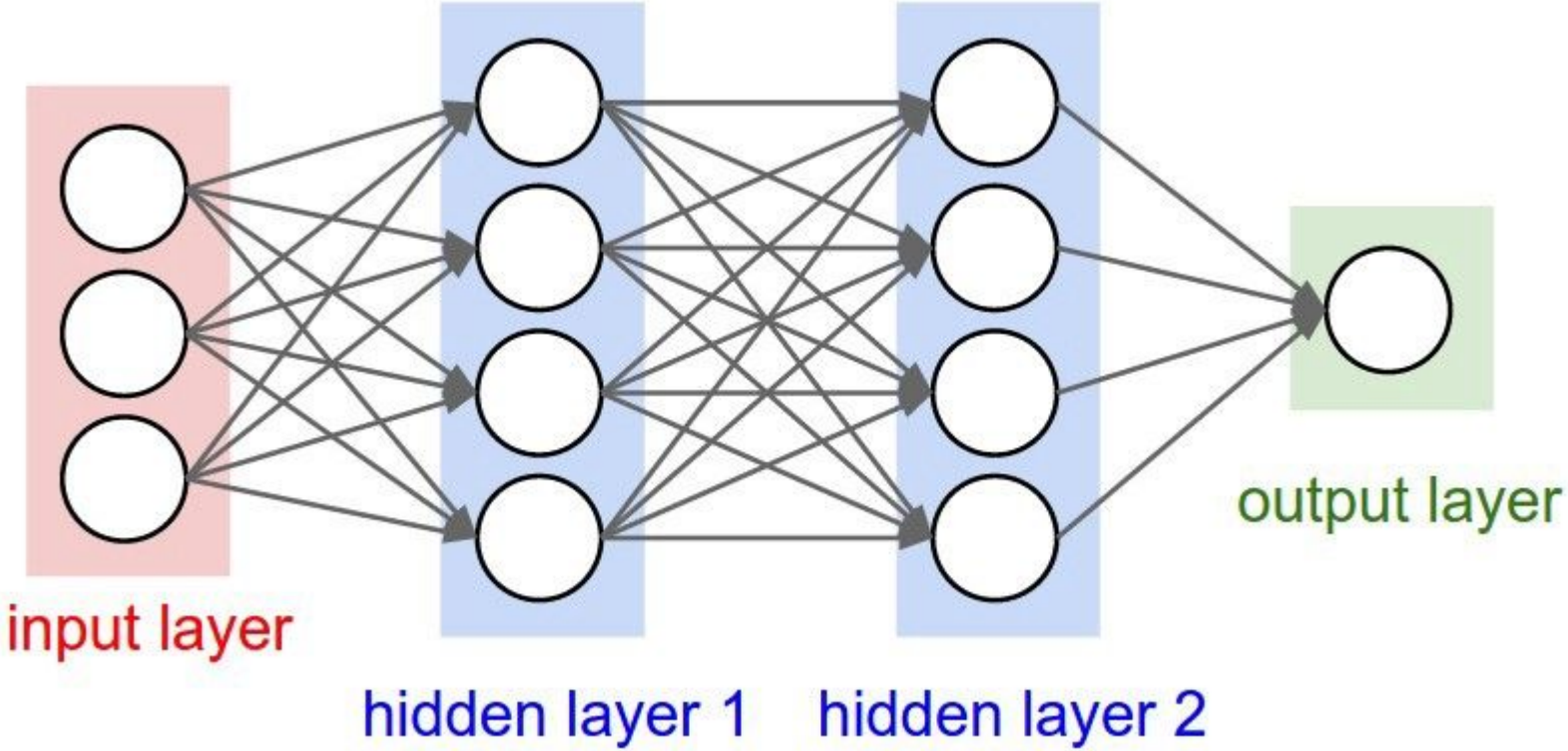
- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning
- Artificial Neural Networks

- and some more ...

In this codelab, we'll be focusing on Artificial Neural Network and generally, the umbrella term for such tasks is Deep Learning.

Deep Learning which is a subset of Machine Learning, simply put is a technique that teaches computers to do what comes naturally to humans. One of which is classifying images.

Basic Neural Network Architecture



Now that we've gotten a little backstory into the world of AI, it's time to get our hands dirty!!!

Our Goal:
To build a Machine Learning model to classify images such trucks, automobile, bird, cat, dog etc.

Tools we'll be working with:

- Colab
- CIFAR-10 Dataset
- Tensorflow
- Keras

https://github.com/ade-mola/gdg-ogbomoso

Special Service Announcement!!

Running a deep learning model on your PC without a GPU might make you lose your mind!!

Instead use a cloud service. Popular cloud services are Google Colab, Kaggle Kernels, Microsoft Azure, AWS Machine Learning, FloydHub, IBM Watson …

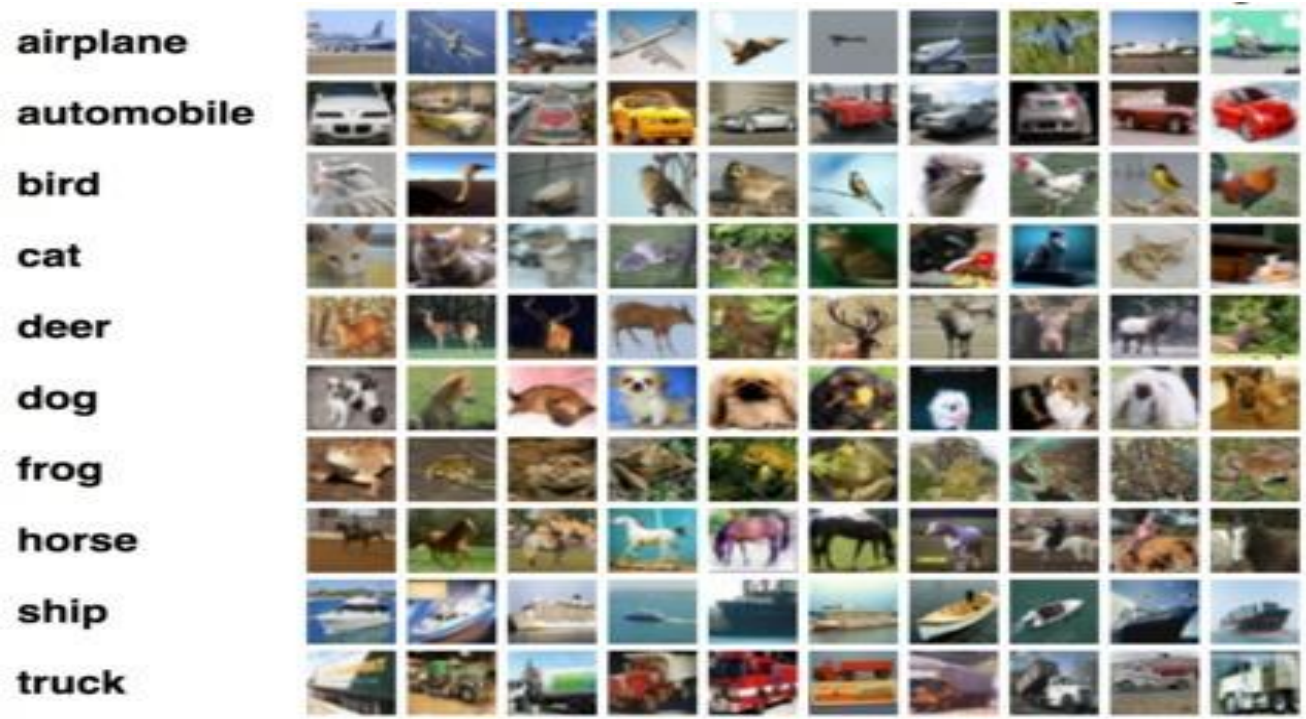
YOU DON'T WANNA BE THIS GUY

Import Libraries

```
[1]  import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     %matplotlib inline
```

```
[2]  import tensorflow as tf
     from tensorflow import keras
```

The CIFAR-10 dataset



Next up is importing our CIFAR-10 dataset

```
[3] from keras.datasets import cifar10

    (x_train, y_train), (x_test, y_test) = cifar10.load_data()

    print('x_train shape: ', x_train.shape)  # num_of_samples * width * height * color_channel
    print('y_train shape: ', y_train.shape)  # sets of labels to data in x_train
    print('x_test shape: ', x_test.shape)    # num_of_samples * width * height * color_channel
    print('y_test shape: ', y_test.shape)    # sets of labels to data in x_test

    print(x_train.shape[0], " training samples")
    print(y_train.shape[0], " labels")
    print(x_test.shape[0], " test samples")
    print(y_test.shape[0], " labels")
```

```
Using TensorFlow backend.
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 42s 0us/step
x_train shape:  (50000, 32, 32, 3)
y_train shape:  (50000, 1)
x_test shape:  (10000, 32, 32, 3)
y_test shape:  (10000, 1)
50000  training samples
50000  labels
10000  test samples
10000  labels
```

Declare variables we'll work with

```
[7]  # declare variables to work with

     epochs = 50  # how many time we train on the dataset, one forward pass
     batch_size = 32  # number of samples propagated thru the network in one epoch

     class_names = ["airplane", "automobile", "bird", "cat",
                    "deer", "dog", "frog", "horse", "ship", "truck"]
```

Plot out some photos from our x_train

```
10]  plt.figure(figsize=(10,10))
     for i in range(25):
         plt.subplot(5,5,i+1)
         plt.xticks([])
         plt.yticks([])
         plt.grid(False)
         plt.imshow(x_train[i])
         plt.xlabel(class_names[y_train[i]])
```

RECAP

- We import in our libraries and datasets.
- We declared variables (batch_size, epochs, class_names) we'll be working with
- We plotted out random images to test we're on the right track

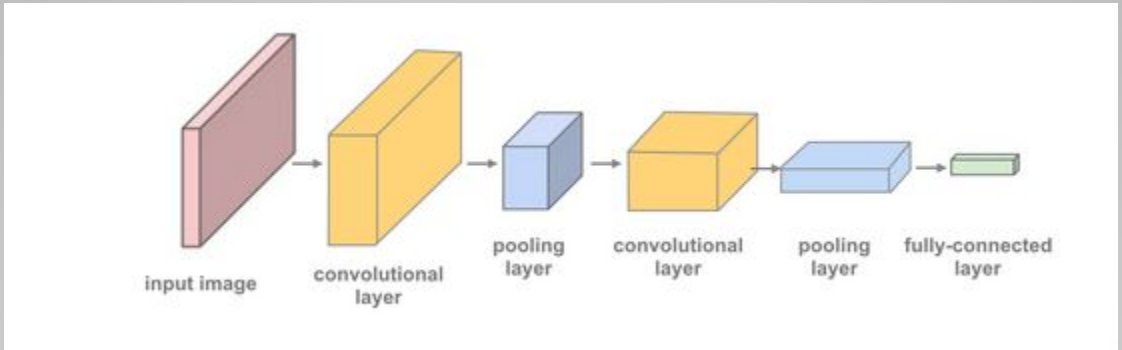
IT’S MODEL TIME!

Building the model

Building the Model

```
[11] model = tf.keras.Sequential()

     model.add(tf.keras.layers.Conv2D(32, kernel_size=(3,3), activation='relu', input_shape=(32,32,3)))
     model.add(tf.keras.layers.Conv2D(64, kernel_size=(3,3), activation='relu'))
     model.add(tf.keras.layers.MaxPool2D(pool_size=(2,2))) # this is used to extract tangible features

     model.add(tf.keras.layers.Conv2D(64, kernel_size=(3,3), activation='relu'))
     model.add(tf.keras.layers.MaxPool2D(pool_size=(2, 2)))

     model.add(tf.keras.layers.Conv2D(128, kernel_size=(3,3), activation='relu'))
     model.add(tf.keras.layers.MaxPool2D(pool_size=(2,2)))

     model.add(tf.keras.layers.Dropout(0.25))
     model.add(tf.keras.layers.Flatten())
     model.add(tf.keras.layers.Dense(1024, activation='relu'))
     model.add(tf.keras.layers.Dropout(0.5))
     model.add(tf.keras.layers.Dense(10, activation='softmax'))
```

Let’s explain some of the terms in the next slide.



Conv2D: This is a 2-dimension convolution layer which can be thought of as stack of filtered images gotten from the input image.

MaxPool2D: reduces the x-y size of an input, keeping only the most active pixels from the previous layer

ReLU Activation Function: f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero; f(z) = max(0, z)

Dropout: regularization technique used to reduce overfitting.

Now we train our model. Yay!

Training the Model

```
[10]  # compile model
      model.compile(loss='categorical_crossentropy',
                    optimizer=tf.keras.optimizers.Adam(lr=0.001, decay=1e-6),
                    metrics=['accuracy'])

      # train model
      model.fit(x_train, tf.keras.utils.to_categorical(y_train),
                batch_size=batch_size,
                shuffle=True,
                epochs=epochs,
                validation_data=(x_test, tf.keras.utils.to_categorical(y_test)))

      # evaluate model
      score = model.evaluate(x_test, tf.keras.utils.to_categorical(y_test))
      print(score)

      print('Accuracy: %.3f' % score[1])
```

```
[21]  predictions = model.predict(x_test)
```

Define helper functions to help with visualisations

```
[22] def plot_image(i, predictions_array, true_label, img):
       predictions_array, true_label, img = predictions_array[i], true_label[i], img[i]
       plt.grid(False)
       plt.xticks([])
       plt.yticks([])

       plt.imshow(img, cmap=plt.cm.binary)

       predicted_label = np.argmax(predictions_array)
       if predicted_label == true_label:
         color = 'blue'
       else:
         color = 'red'

       plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                    100*np.max(predictions_array),
                                    class_names[true_label]),
                                    color=color)
```

```
[17] def plot_value_array(i, predictions_array, true_label):
       predictions_array, true_label = predictions_array[i], true_label[i]
       plt.grid(False)
       plt.xticks([])
       plt.yticks([])
       thisplot = plt.bar(range(10), predictions_array, color="#777777")
       plt.ylim([0, 1])
       predicted_label = np.argmax(predictions_array)

       thisplot[predicted_label].set_color('red')
       thisplot[true_label].set_color('blue')
```

Let’s Visualise our Result (1)

```
[16] y_test = y_test.reshape(-1)
     y_test

     num_rows = 5
     num_cols = 3
     num_images = num_rows*num_cols
     plt.figure(figsize=(2*2*num_cols, 2*num_rows))
     for i in np.arange(num_images):
       plt.subplot(num_rows, 2*num_cols, 2*i+1)
       plot_image(i, predictions, y_test, x_test)
       plt.subplot(num_rows, 2*num_cols, 2*i+2)
       plot_value_array(i, predictions, y_test)
```

And VOILA!


cat 80% (cat)
ship 93% (ship)
ship 98% (ship)
airplane 99% (airplane)
frog 100% (frog)
frog 100% (frog)
automobile 53% (automobile)
frog 97% (frog)
cat 95% (cat)
truck 62% (automobile)
airplane 71% (airplane)
truck 100% (truck)

And with that, we can classify Images with our Model. This model can be incorporated as an API to web or mobile apps to identify Images on the GO!

THANK YOU!!