Experts

Hello, TensorFlow :)

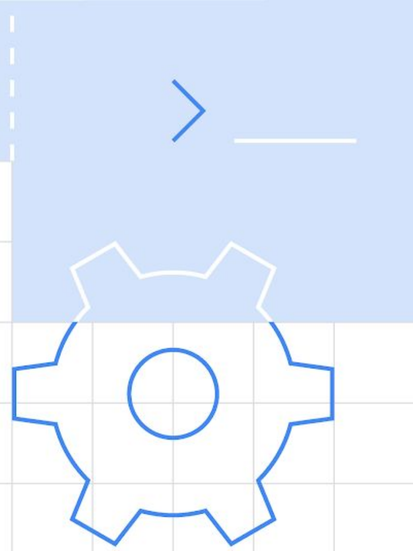
Basic Deep Learning recipes with TensorFlow

Sayak Paul
PyImageSearch
@RisingSayak

Google Developers

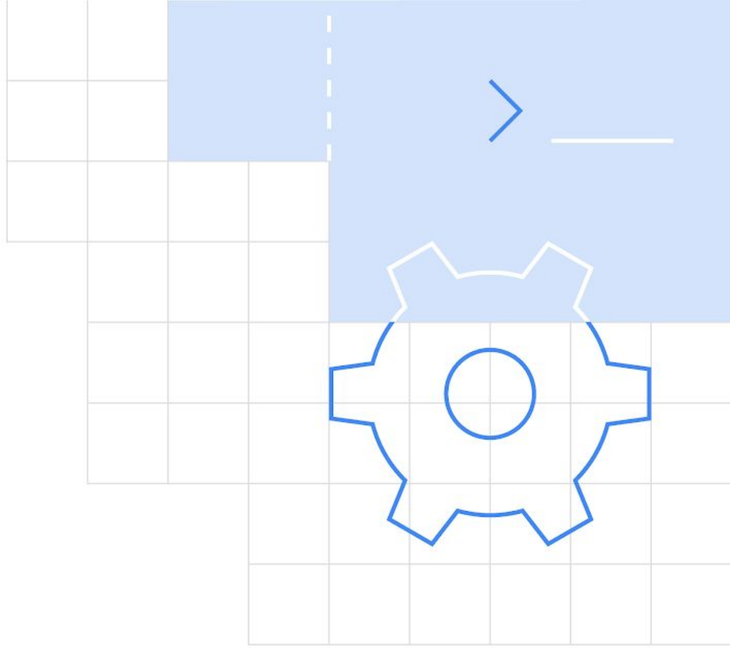
Acknowledgement

- The entire PyImageSearch team
- Laurence Moroney (Google)
- François Chollet (Google)
- Soonson Kwon (Google)



If terms like TensorFlow and Deep Learning scare you, don’t worry you’re not alone!

Experts



Prerequisites:

- Python (at least 6 months)
- Basic high school math

Experts

Agenda

- What is TensorFlow?
  - Tensor
  - Flow
- An introduction to Deep Learning
  - What is *learning* here?
  - Why the word *deep*?
- Let’s get coding!
- QA (if time permits)

What is TensorFlow?

As per tensorflow.org -

"The core open source library to help you develop and train ML models."

What is TensorFlow?

But at its core, ***TensorFlow is a library for doing numerical computation***.

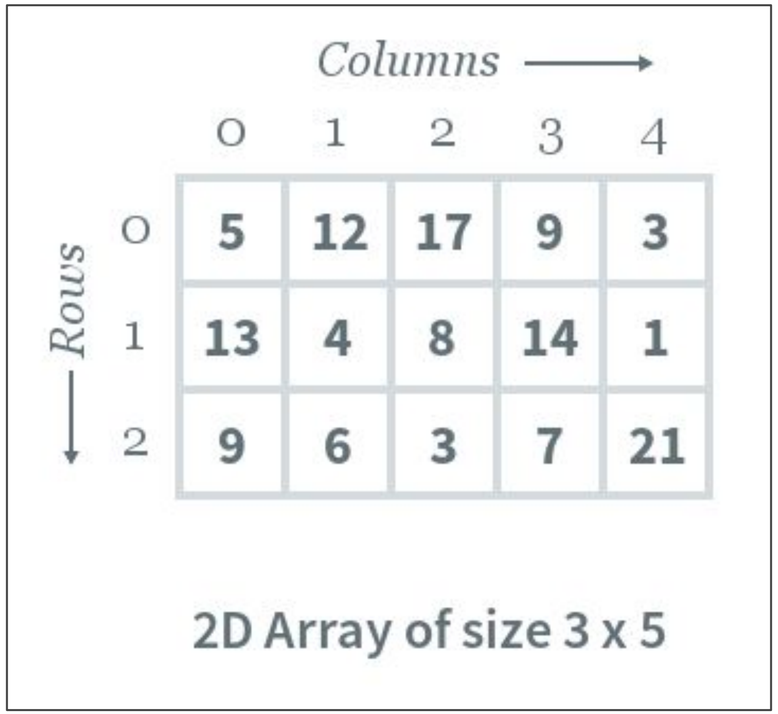
Why is it called TensorFlow?

Two components in there -

- Tensor
- Flow

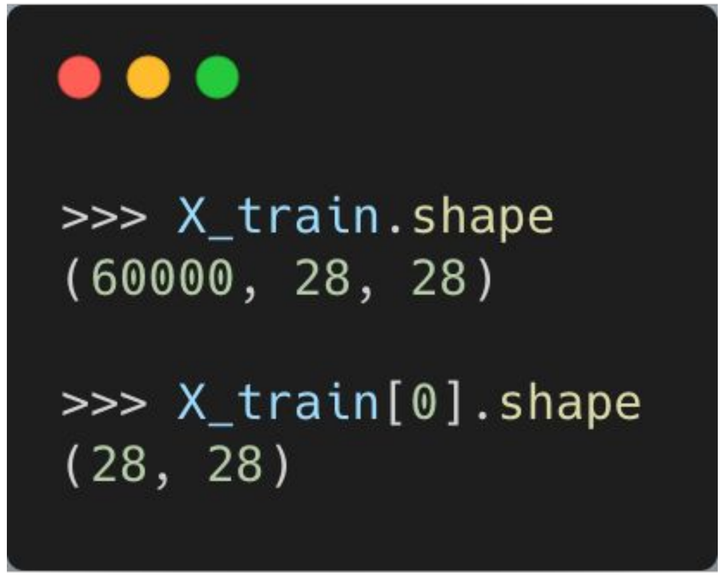
Tensors

Multi-dimensional arrays!



Tensors

You can call it a container of multi-dimensional arrays as well.



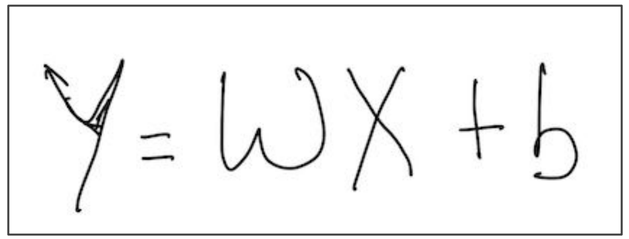
Flow

Flow represents a *directed computational graph*!

Flow

Flow represents a *directed computational graph*!

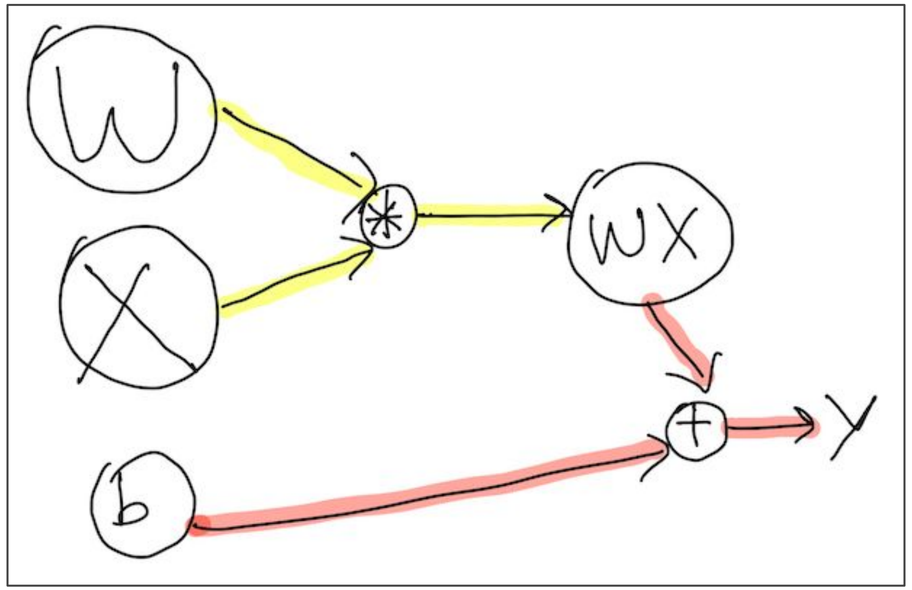
Can you represent the following equation in terms of a graph?

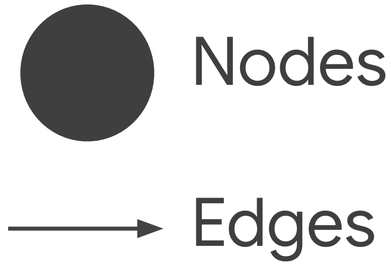


Flow

Flow represents a *directed computational graph*!

Yes, you *can*!





Flow

Flow represents a *computational graph*!

Long story cut short, TensorFlow represents all the variables and the operations in terms of a computation graph.*

*In *eager mode*, it *does not* construct a computation graph beforehand.

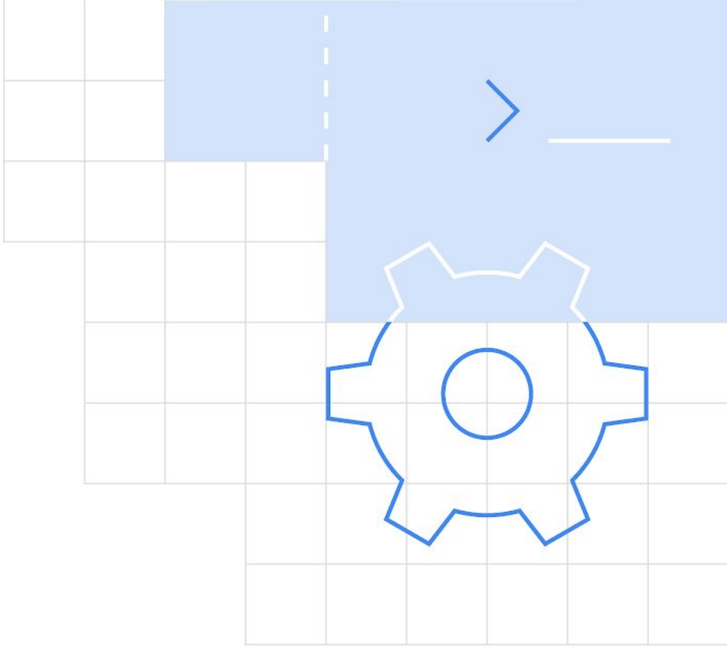
The promise of Machine Learning

Source: TensorFlow team

The promise of Machine Learning

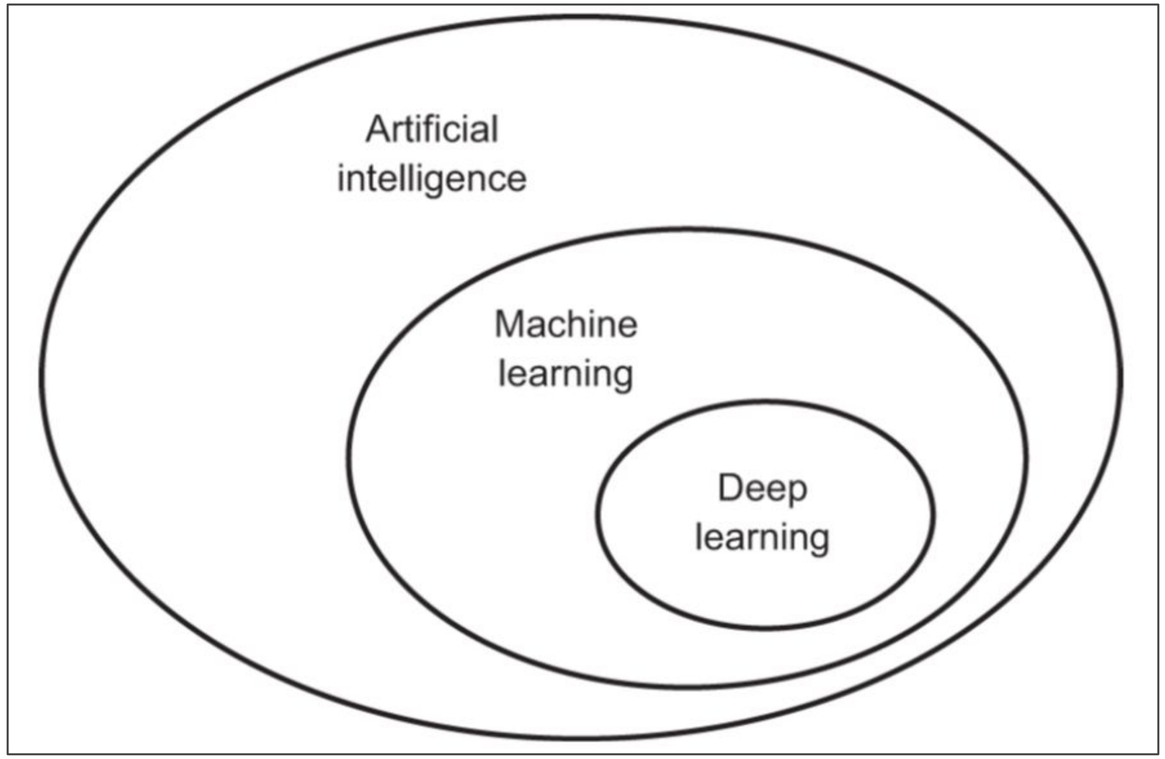
Source: TensorFlow team

Aren’t we doing **Deep Learning** today?

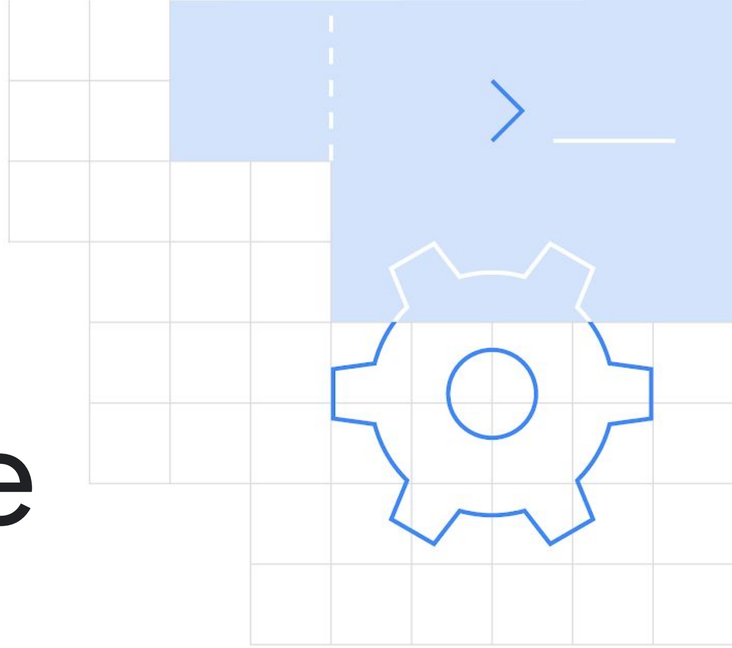


Experts

Yes, **of course**!



Source: [Deep Learning with Python]



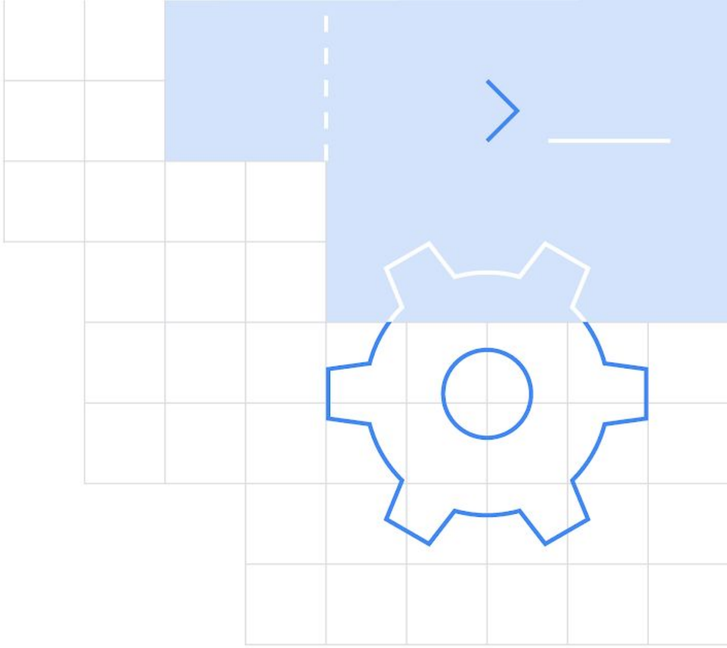
So, what conclusion can we draw from that figure?

Experts

Going back to this figure ...

Source: TensorFlow team

Why this might be **useful**?



Experts

Activity Recognition

```
if(speed<4){
    status=WALKING;
}
```

Activity Recognition

```
if(speed<4){
    status=WALKING;
}
```

```
if(speed<4){
    status=WALKING;
} else {
    status=RUNNING;
}
```

Activity Recognition

```
if(speed<4){
    status=WALKING;
}
```

```
if(speed<4){
    status=WALKING;
} else {
    status=RUNNING;
}
```

What now?

Our daily (digital) life

- Google Photos
- Next word suggestion while you're typing
- Tag suggestions in Facebook photos
- ...

Two fundamental elements

- Data
- Labels (Answers)

An example

age	secondary makrs	higher secondary marks	admission
20	356	471	0
18	421	405	1
19	321	300	0
18	423	400	1
...	...	...	...

An example

age	secondary makrs	higher secondary marks	admission
20	356	471	0
18	421	405	1
19	321	300	0
18	423	400	1
...	...	...	...

Multiple rows and multiple columns!

An example

age	secondary makrs	higher secondary marks	admission
20	356	471	0
18	421	405	1
19	321	300	0
18	423	400	1
...	...	...	...

Rows: **Data points/Instances**
Columns: **Features**

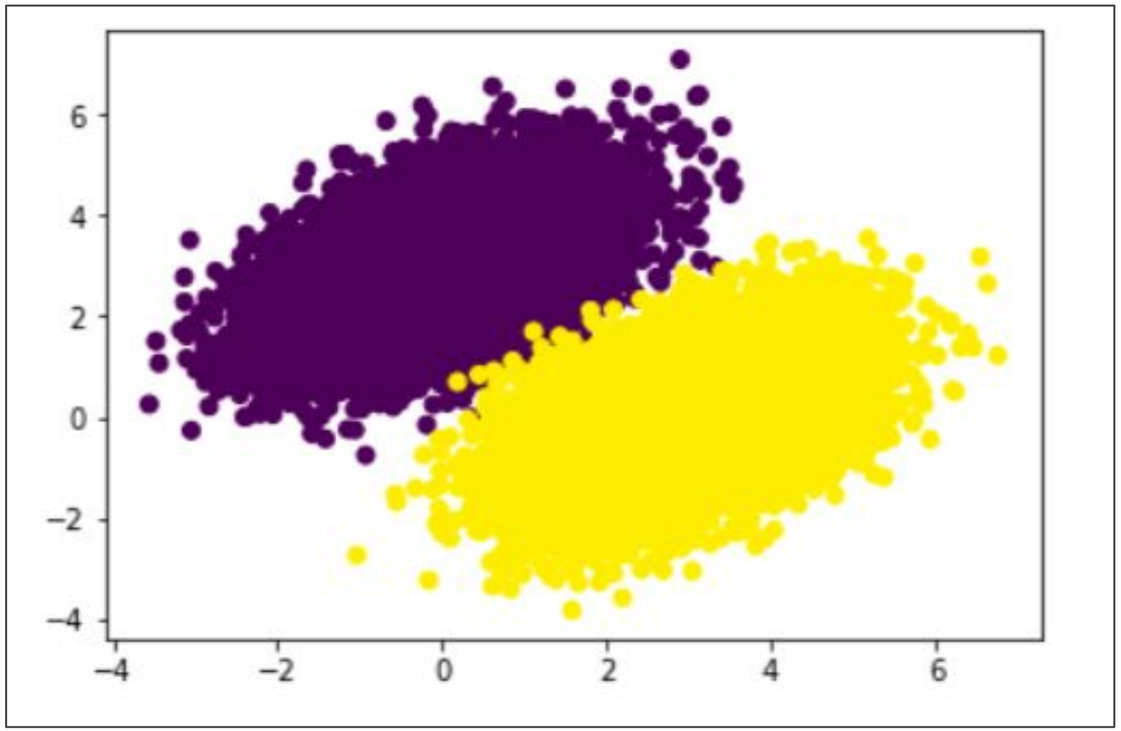
An example

age	secondary makrs	higher secondary marks	admission
20	356	471	0
18	421	405	1
19	321	300	0
18	423	400	1
...	...	...	...
Data			**Labels/Answers**

Rows: **Data points/Instances**
Columns: **Features (the last one is for *Labels*)**

Task at hand

How can we separate the following data points?



Task at hand

To separate the data points we need to know the ***underlying pattern of the data points*** that separates them.

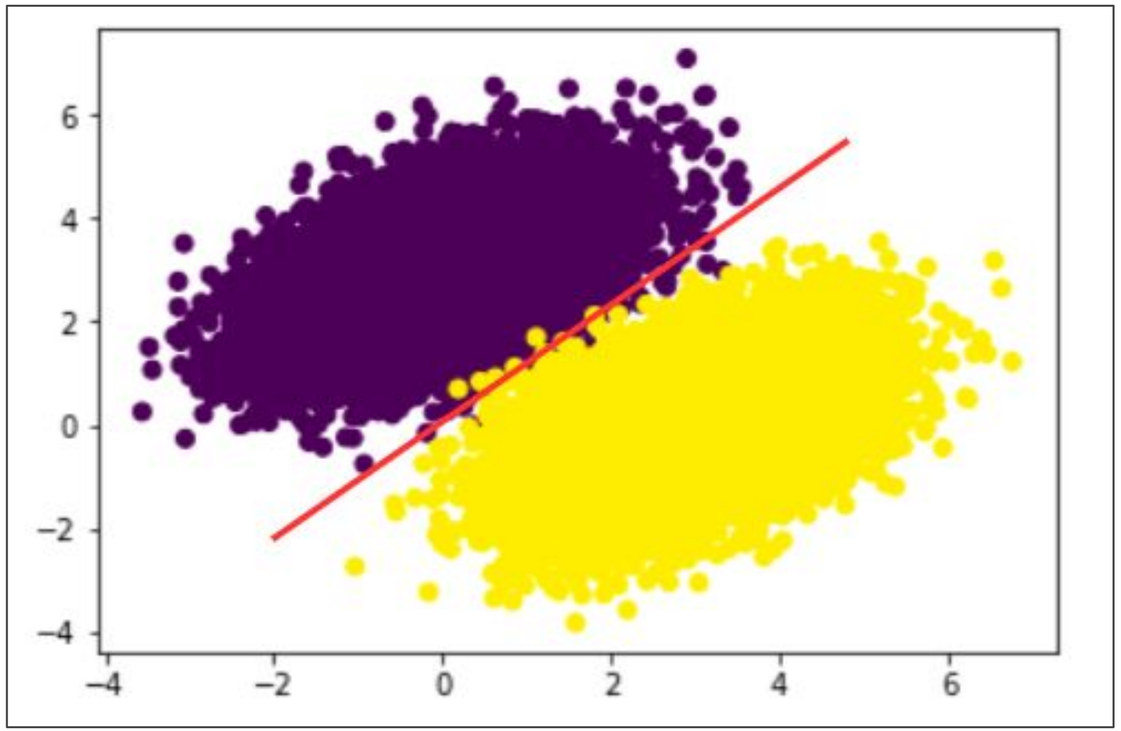
Task at hand

To separate the data points we need to know the ***underlying pattern of the data points*** that separates them.

You cannot distinguish between a cat and a dog if you don't know about them.

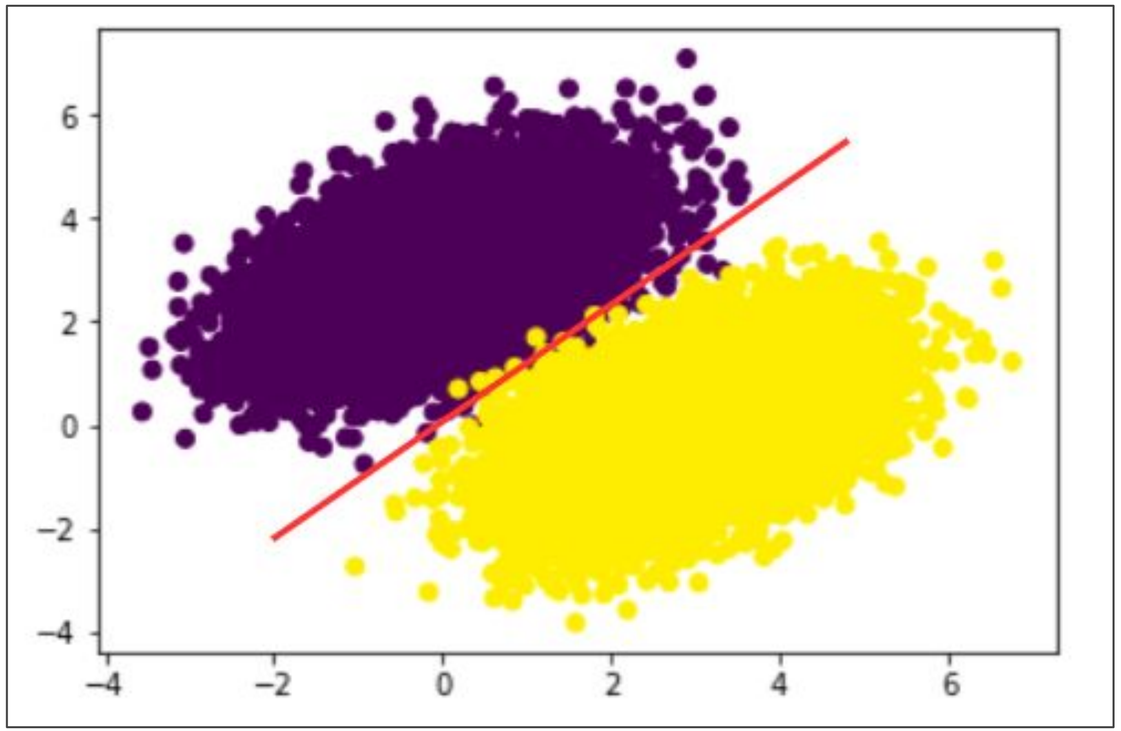
Task at hand

Quick thought: **The data points are linearly separable**



Task at hand

But how do we find that line?



Meet your new fellas - **parameters**

- Weights
- Biases

Meet your new fellas - **parameters**

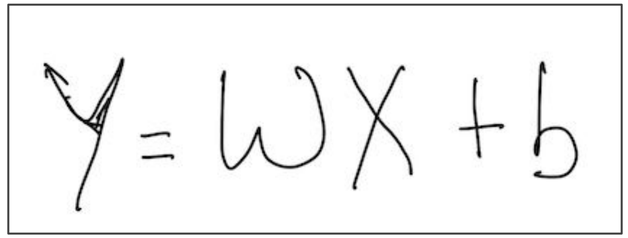
- We apply a series of **transformations** on our input data (the figure) with weights and biases.

Meet your new fellas - **parameters**

- We apply a series of **transformations** on our input data (the figure) with weights and biases.
- These transformations are needed to uncrumple our input data.

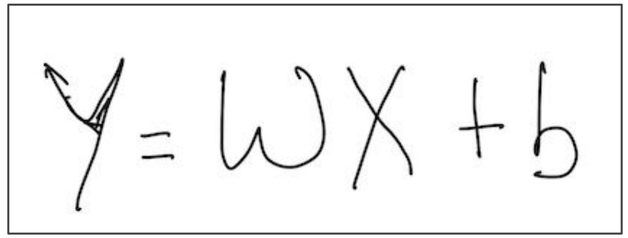
Meet your new fellas - **parameters**

You have already seen how these transformations look like -



Meet your new fellas - **parameters**

You have already seen how these transformations look like -



X = Data points
W = Weights
B = Bias

“Talk is cheap. Show me the code!”

Head over to this link 👉 bit.ly/ht-colab

- Transform our input data
- Calculate error
- Derivative 😢
- And more ...

From the code

The way we minimized the loss function (Mean Squared Error) comprises of the following -

- Linear regression
- Gradient-based learning
- Backpropagation

Linear Regression

The entire system that we built to minimize our loss function -

a. We applied transformations to our input data.

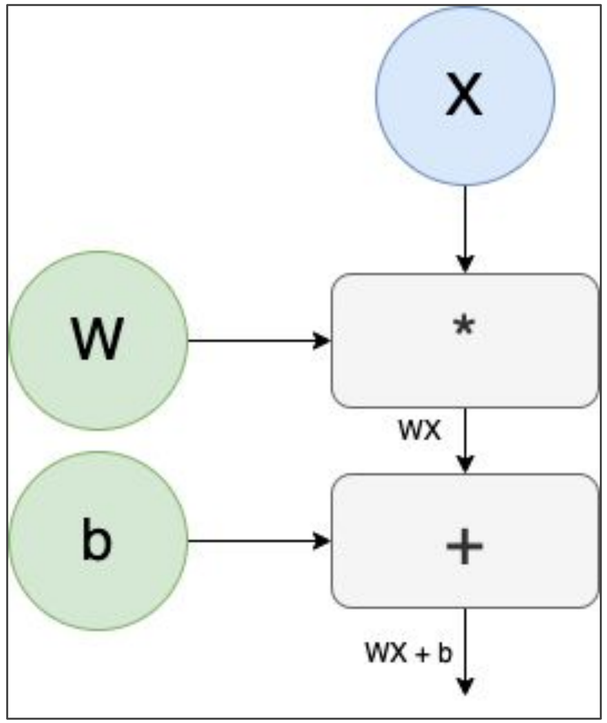


Figure adapted from François Chollet

Linear Regression

The entire system that we built to minimize our loss function -

b. We calculated the loss between the results of those transformations and our labels.

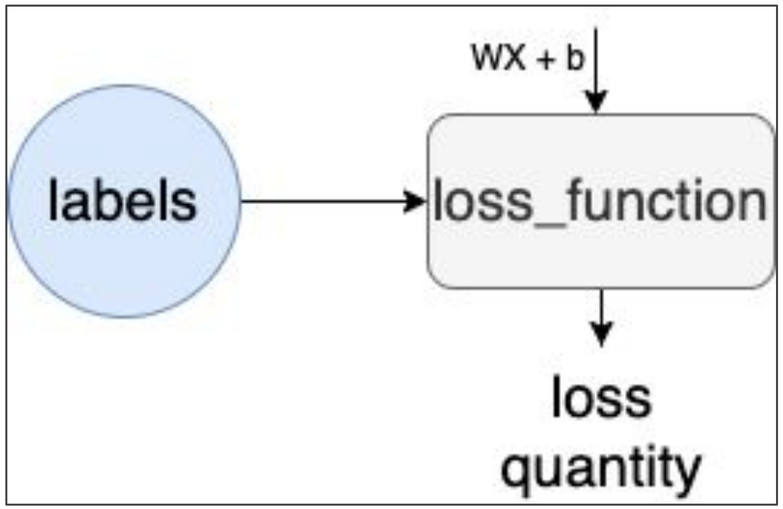


Figure adapted from François Chollet

Linear Regression

The entire system that we built to minimize our loss function -

c. We then *calculated derivatives of the loss function* w.r.t the parameters and *updated the parameters*.

Linear Regression

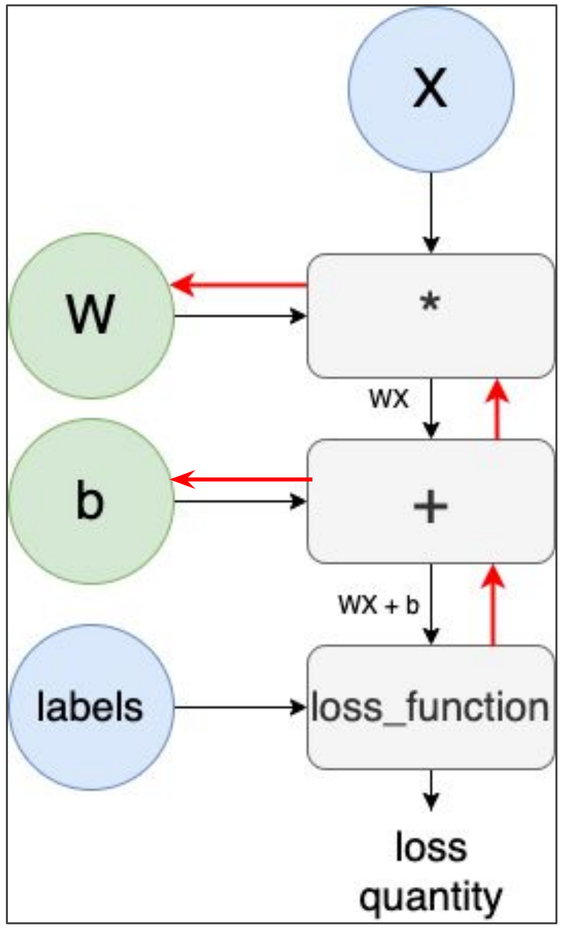


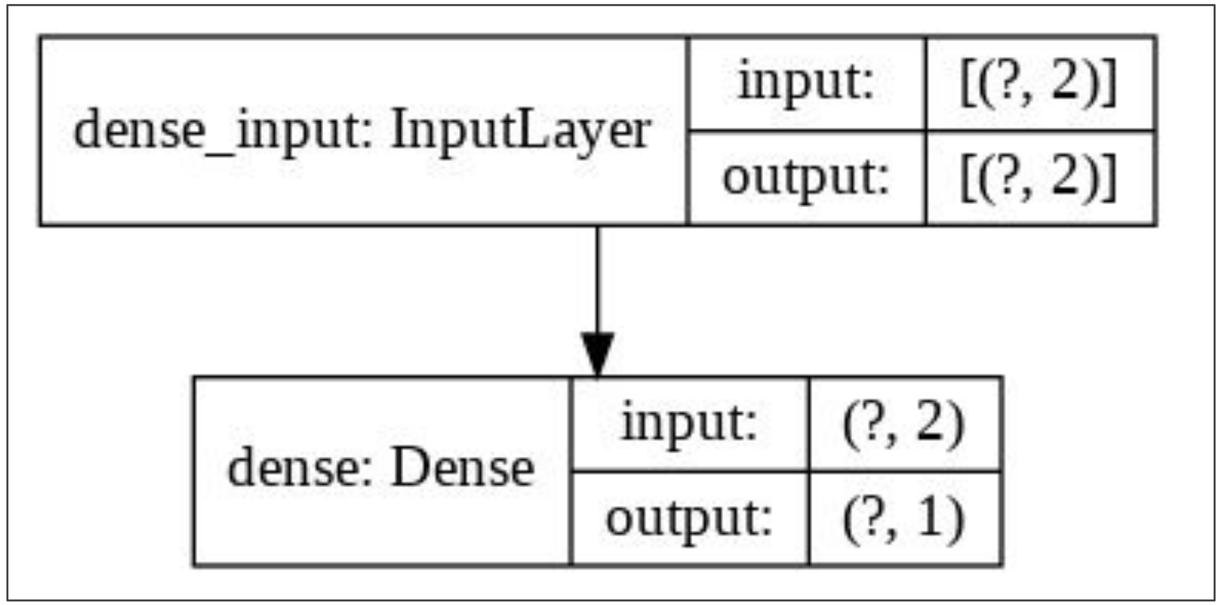
Figure adapted from François Chollet

Linear Regression

The entire system that we built to minimize our loss function -

a. We applied transformations to our input data.
b. We calculated the loss between the results of those transformations and our labels.
c. We then *calculated derivatives of the loss function* w.r.t the parameters and *updated the parameters*.
d. We repeated **a**,**b**, and **c**.

Schematically our system is



Gradient-based *learning*

```
def minimize_loss(x, y):
    with tf.GradientTape() as tape:
        # Transform data and compute the loss
        predictions = compute_predictions(x)
        loss = compute_loss(y, predictions)

        # Compute the derivative/gradients
        dloss_dw, dloss_db = tape.gradient(loss, [w, b])

    # Update the parameters and return loss
    w.assign_sub(LEARNING_RATE * dloss_dw)
    b.assign_sub(LEARNING_RATE * dloss_db)
    return loss
```

Gradient-based *learning*

- **Learning, wait what?**
- ***Learning*** is a fancy term for **function approximation**.
- In our case, we tried to *find the underlying pattern* of the input data points *that separates them one another*.

Gradient-based *learning*

- **Learning, wait what?**
- ***Learning*** is a fancy term for **function approximation**.
- In our case, we tried to *find the underlying pattern* of the input data points *that separates them one another*.

These underlying patterns are nothing but functions and **finding these patterns = function approximation**

Gradient-based *learning*

- Remember we minimized the loss function?
- In other words, we ***trained*** a system to minimize the loss function.
- We did so with the help of ***gradient-based learning***.
- As we *updated the parameters* in this process, they are also called ***trainable variables/learnable parameters***.

Gradient-based ***learning***

- Remember we minimized the loss function?
- In other words, we ***trained*** a system to minimize the loss function.
- We did so with the help of ***gradient-based learning***.
- As we *updated the parameters* in this process, they are also called ***trainable variables/learnable parameters***.

This form of learning/training is also known as ***Supervised Learning***.

Backpropagation

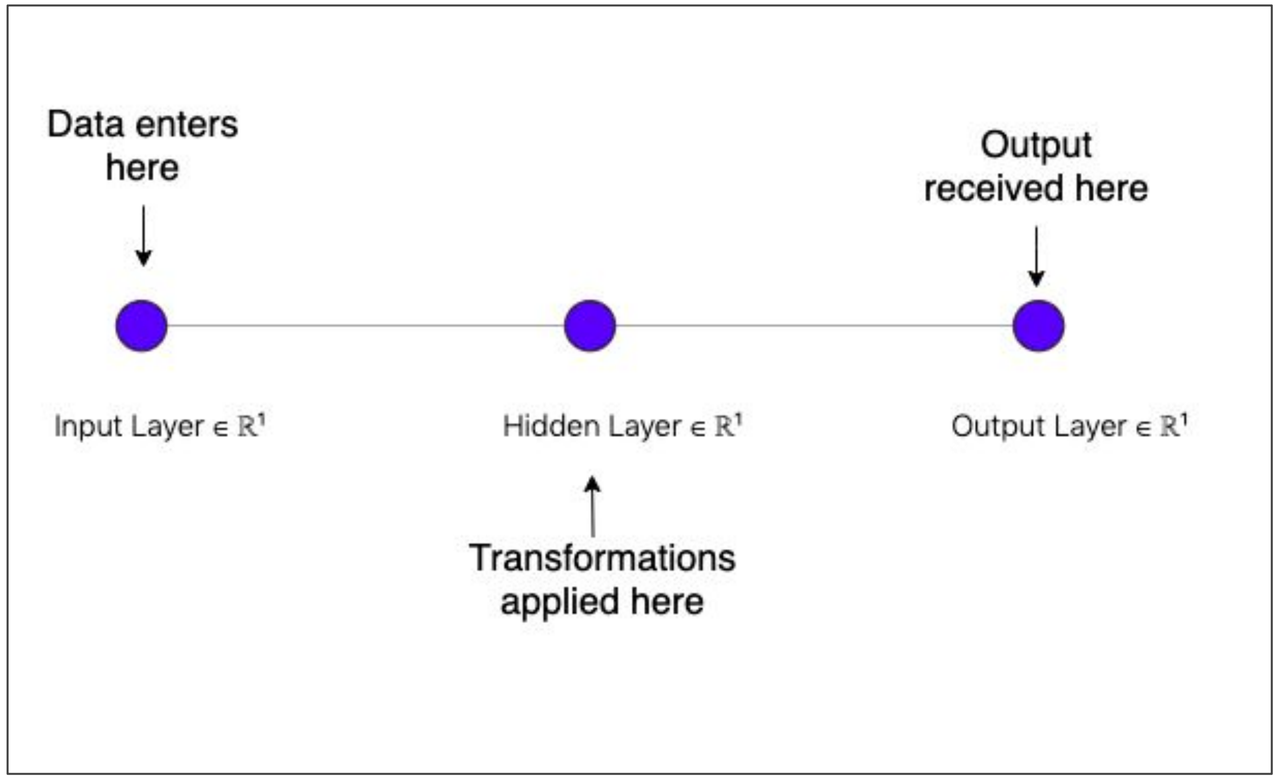
```
def minimize_loss(x, y):
    with tf.GradientTape() as tape:
        # Transform data and compute the loss
        predictions = compute_predictions(x)
        loss = compute_loss(y, predictions)

        # Compute the derivative/gradients
        dloss_dw, dloss_db = tape.gradient(loss, [w, b])

    # Update the parameters and return loss
    w.assign_sub(LEARNING_RATE * dloss_dw)
    b.assign_sub(LEARNING_RATE * dloss_db)
    return loss
```

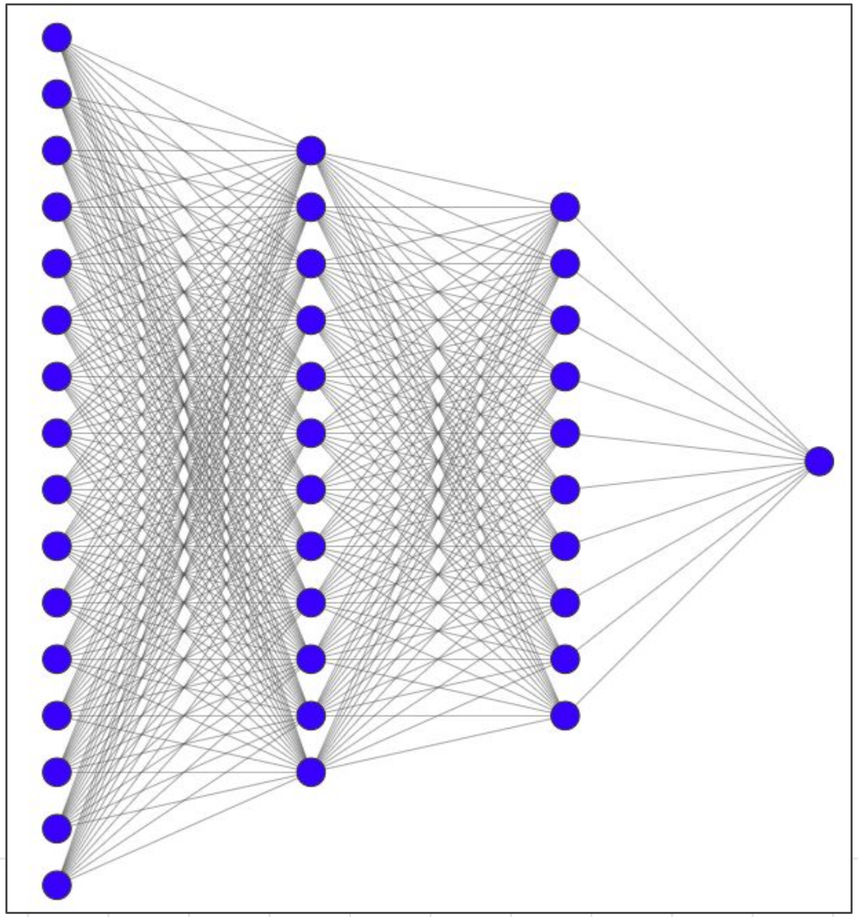
Meaning of *deep*

Another diagram of our system could be -



Meaning of *deep*

What if a diagram looked like so?



Meaning of ***deep***

- We can add *depth* to our system in a similar manner.

Meaning of ***deep***

- We can add *depth* to our system in a similar manner.
- A quick question -

Did you find any similarity in between the previous figure and the ***neurons*** inside our brain?

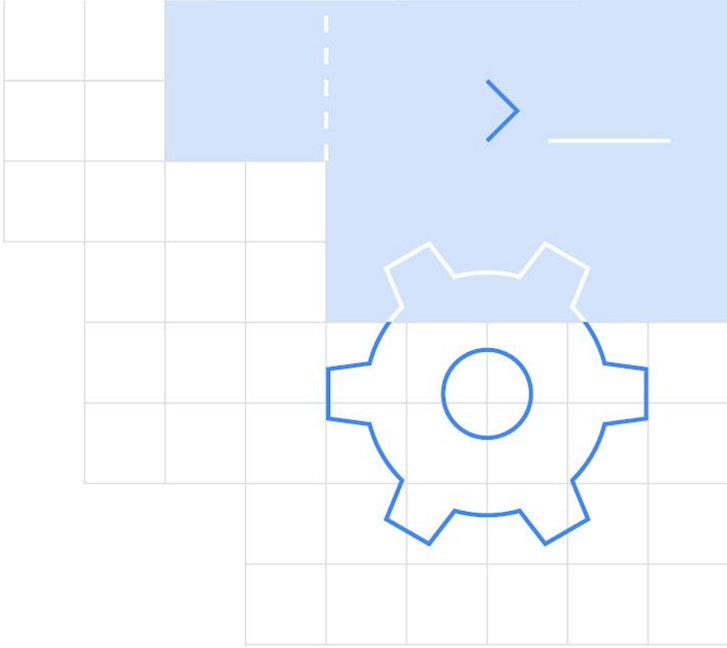
Meaning of ***deep***

- We can add *depth* to our system in a similar manner.
- A quick question -

Did you find any similarity in between the previous figure and the ***neurons*** inside our brain?

That is how the powerful **neural networks** look like

This is no way even the scratch of the surface!



Experts

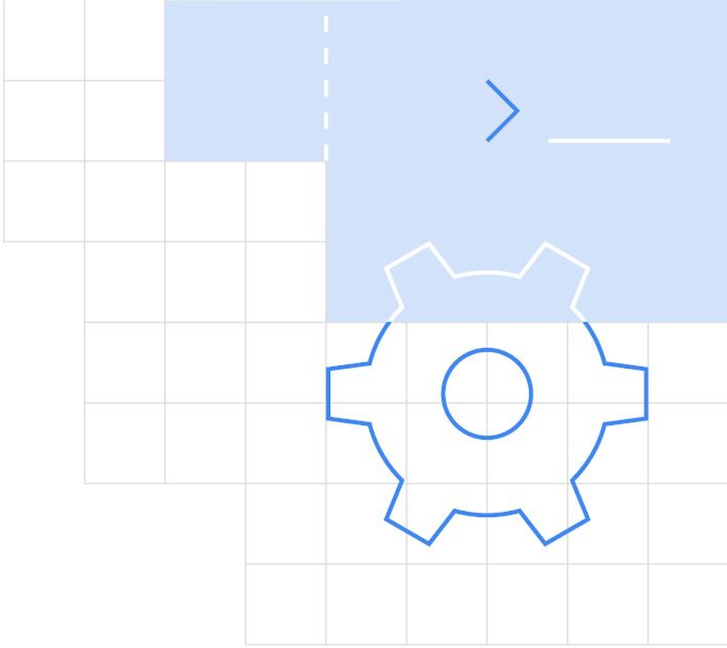
If this seemed interesting

Consider the following resource to jump in this ocean -

- Deep Learning With Python by **François Chollet**
- TensorFlow in Practice Specialization by **deeplearning.ai** & **Laurence Moroney**
- Neural Networks and Deep Learning by **Michael Nielsen**
- TensorFlow, Keras and deep learning, without a PhD by **Martin Gorner**
- Cannot fit more in one single slide, sorry :(

Slides available here -

bit.ly/gdg-goa-20



Experts

Experts

Thank You!

Sayak Paul
PyImageSearch
@RisingSayak