What is tensorflow

It is a machine learning framework used to build models, like NLP natural language processing , computer vision , face recognition.

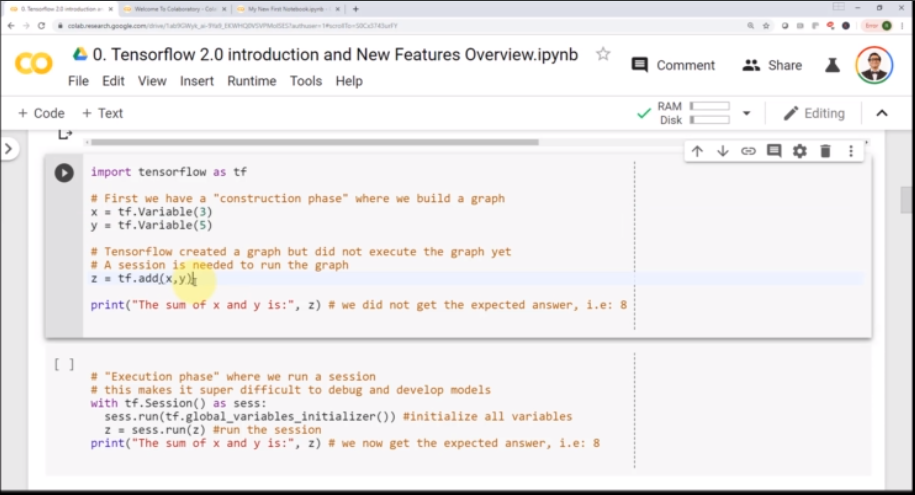
It is developed by google in 2015 by google brain team

GPU and TPU

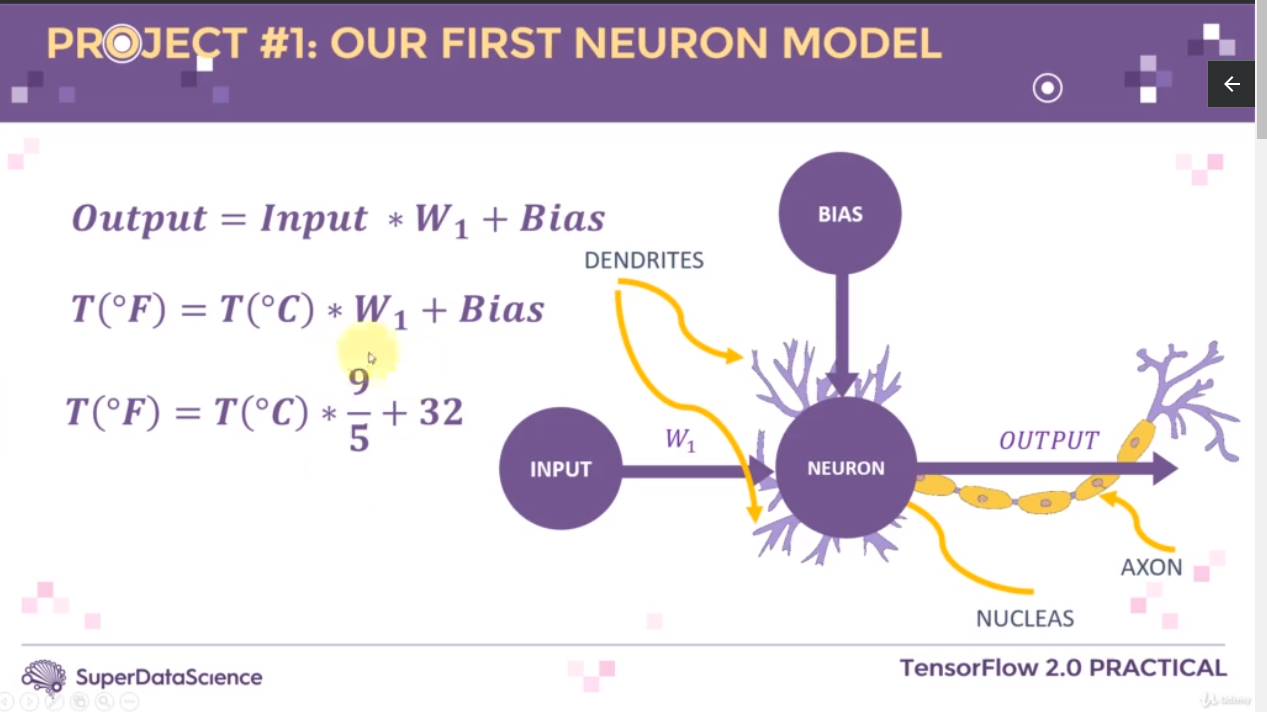
Graphics Processing unit – used in CAD drawing, video editing.

TPU –

Eager Execution – we can execute code line by line , in Tensor flow 1 we had to initiate a session to print or analyse the variables but from tensor flow 2 onwards we do not need it.



Keras – default API to interact with tensorflow



Regression

Regression works by predicting value of one variable Y based on another variable X , where X is called Independent variable and Y is called dependent variable.

Graph 1 when value of x increases the value of y is increases and graph 2 when y decreases if the value of x increases

graph 1 Y

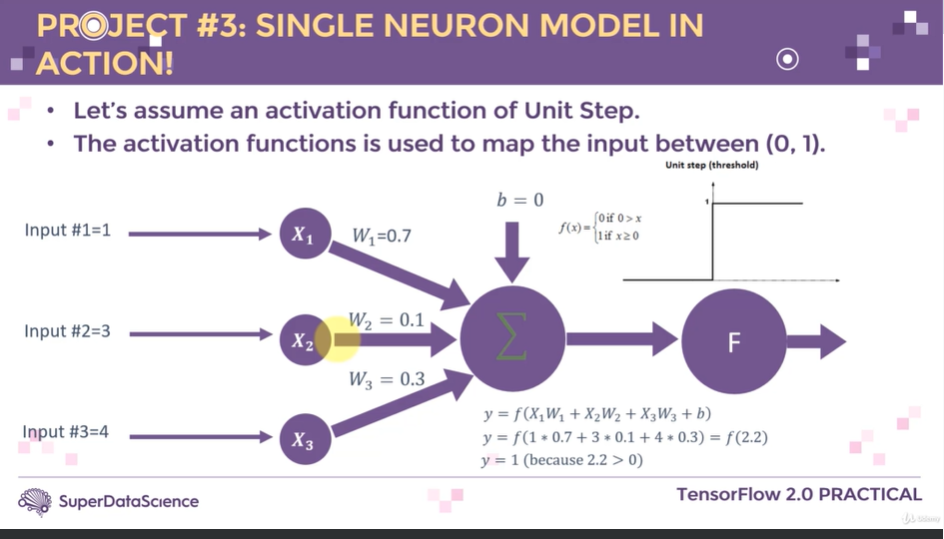
Y

X X Graph 2

Activation function

Activation function takes in inputs \* weigths + bias and makes a decision

For ex : the activation function when finds a positive value it sets the output as 1 and if it is a negative number it sets the value as 0 . this way the model is capable of taking decisions , like the object is a cat or not . i.e if we have range of values and when we want to to default them to some values we need activation function and this will help system to come to a decision



Activation function screen shot above.

Activation function overview

1.Sigmoid, 0,1 – output layer - for classifying data we can use this function

2.Relu – rectified linear units – in hidden layers. – for regression like predicting sales or linear regression problems we can use this function

3. Hyperbolic Tangent Activation - -1 to 1 - in out put layers

Multilayer perceptron network

Network which has hidden layers which connect input to output.

These are called dense network. Or deep learning.

Epochs

Updating the weights is known as epochs because there is an error in the predicted out to the desired output.

When we increase the epochs we minimize the errors and try to match the desired output.

// Create the Model

Model = tf.keras.models.Sequential() // initiate the model

Model.add(tf.keras.layers.Dense(units=100,activation=’relu’,input\_shape=(35, ))) // tell the model number of inputs along with activation function

Model.add(tf.keras.layers.Dense(units=100,activation=’relu’)) // hidden layer 1 or deep network , we don’t have to define input here as it is hidden

Model.add(tf.keras.layers.Dense(units=100,activation=’relu’)) // hidden layer 2 or deep network , we don’t have to define input here as it is hidden

Model.add(tf.keras.layers.Dense(units=1,activation=’linear’) // as we need only one output we have defined the unit as 1 and linear as output which means it will give linear co relation with the data.

// check the summary of the model

Model.summary() // will give you no of parameters to be trained including weights and biases.

/// compile the model.

Model.compile(optimizer=’Adam’ , loss=’mean\_squared\_error’)

// Train the model

Epochs\_hist = model.fit(X\_train,y\_train,epochs=20, batch\_size=50) // model.fit will train the data

**So mainly there are four steps that we need to perform**

Model = tf.keras.model.sequentials() // To create the model object

Model.add() // to create the model with the networks

Model.Summary() // optional but you can check the summary of the model

Model.compile() // compile the model , like compiling the code.

Model.fit() // train the model.

\*\*\* There are tools which can identify the correct and optimized models for us --- Need to do research on this.

Normalization

why do we normalize the data in AI

ChatGPT

**ChatGPT**

Normalizing data in AI, specifically in machine learning and deep learning, serves several purposes:

1. **Improving convergence**: Many optimization algorithms used in training machine learning models converge faster when the data is normalized. Normalizing the data helps prevent large gradients that can slow down the convergence process.
2. **Equalizing scales**: Features in the dataset may have different scales. For example, one feature might range from 0 to 1 while another might range from 0 to 1000. Such differences in scale can affect the performance of certain machine learning algorithms, particularly those based on distances or gradients. Normalization scales all features to a similar range, typically between 0 and 1 or with a mean of 0 and a standard deviation of 1.
3. **Improving numerical stability**: Normalizing data can help prevent numerical instability issues during computation. Large values in features may cause overflow or underflow in computations, leading to numerical instability. Normalizing the data reduces the likelihood of such issues.
4. **Handling outliers**: Normalization can mitigate the impact of outliers in the dataset. Outliers can disproportionately influence the behavior of certain algorithms, particularly those sensitive to scale. By scaling the data, outliers are often brought within a similar range as other data points, reducing their influence.
5. **Assisting interpretability**: Normalization can sometimes improve the interpretability of the model. When features are on different scales, it can be challenging to interpret the importance of each feature. Normalizing the data can make it easier to interpret the relative importance of different features in the model.

Overall, normalization is a preprocessing step that helps to prepare the data for training machine learning models, making the optimization process more efficient and improving the model's generalization and performance.

**You**

what is normalizing the data can you explain me assuming i am 8 year kid

ChatGPT

**ChatGPT**

Sure! Imagine you have a bunch of toys that you want to organize neatly in your room. Some toys are big, some are small, and some are in between. Now, let's say you want to make sure they all fit nicely into a box.

Normalizing data is a bit like making sure all your toys are a similar size before you put them into the box. Here's how you might do it:

1. **Same Size Box**: First, you pick a box that's big enough for all your toys. This box represents the range of values you want your data to fit into.
2. **Making Toys Similar Sizes**: Next, you look at all your toys. If one toy is really big and another is really small, you might want to make them closer in size. So, you might decide to shrink the big toy a little bit and maybe stretch the small toy a bit until they're all about the same size.
3. **Putting Toys in the Box**: Now that all your toys are similar sizes, you can put them neatly into the box. This way, they all fit nicely together without any one toy taking up too much space or being squished in a tiny corner.

In the same way, when we normalize data in AI, we're making sure all the numbers in our dataset are on a similar scale, so they can work together nicely when we use them to train our AI models. This makes it easier for the AI to learn from the data and make good predictions or decisions.

**You**

but while normalizing the data we loose the actual values right how do you explain that ?

ChatGPT

**ChatGPT**

That's a great question! When we normalize data, we're not necessarily losing the actual values themselves, but we're changing their scale or range. Let's go back to our toy example to explain this:

Imagine you have a toy car that is 10 inches long and a toy dinosaur that is 20 inches long. If we decide to normalize these toys by scaling them down to fit in a smaller box, we might shrink the car to 5 inches and the dinosaur to 10 inches.

So, while the actual values changed (the lengths of the toys), we didn't lose the fact that the toy car was originally twice as long as the toy dinosaur. We've simply made them both smaller so they fit into our smaller box.

Similarly, in AI, when we normalize data, we're changing the scale of the numbers, but we're not losing the relationships between them. For example, if one feature in our dataset originally ranged from 0 to 100 and another ranged from 0 to 1, normalizing them might make both ranges between 0 and 1 or with a mean of 0 and a standard deviation of 1.

So, even though the actual values are different after normalization, the relationships between the values within each feature and across different features remain intact. This allows our AI models to still learn from the data effectively.

**Data set can be divided into three parts**

Training – train the model with this data.

Tesing – for testing the model , this data model has never seen before

Validation – for generalizing the model, otherwise the model will not give proper out when it see’s the new values.

How to over come this.

While training the data split the data for calculating validation loss aswell, so in the below graph you can see the orange line and blue at one point started going parllel to each other , so we should collect the x an y corrdinate details from where both the lines start to go parllel to each other.

Blue line – form where it goes parallel to orange line whe should not collect those points as it overfits the training and it will not understand the new values.

You should only collect those points from where they both splits.



**Here from the above graph we can say that we don’t have train the model more than 20 epochs it exactly where the blue line and orange line are split.**

**When the model is trained only 20 epochs , you can see exactly where it is splitting or deviating**

****

RootMeanSquareError

MeanSquareError

MeanAbsoluteError

Rsquare

Adjusted Rsquare

Regression problems , Classification problems ,

Regression use cases

can you give me a list of regression use cases whose values we can predict using AI

ChatGPT

**ChatGPT**

Certainly! Regression analysis is a statistical technique used to model the relationship between a dependent variable and one or more independent variables. Here's a list of regression use cases where values can be predicted using AI:

1. **Sales Forecasting**: Predicting future sales based on historical sales data, marketing expenditures, seasonality, etc.
2. **Stock Price Prediction**: Forecasting stock prices based on historical price movements, trading volumes, market sentiment, and other relevant factors.
3. **Demand Forecasting**: Estimating future demand for products or services based on factors such as historical sales, economic indicators, and marketing campaigns.
4. **Real Estate Price Prediction**: Predicting property prices based on factors such as location, size, amenities, and market trends.
5. **Healthcare Outcome Prediction**: Forecasting patient outcomes or disease progression based on medical history, demographics, lab results, and treatment plans.
6. **Customer Lifetime Value Prediction**: Estimating the future value of customers based on their past purchase behavior, demographics, and engagement with the brand.
7. **Churn Prediction**: Predicting customer churn or attrition based on factors such as usage patterns, customer satisfaction scores, and interaction history.
8. **Energy Consumption Forecasting**: Forecasting energy consumption for residential or commercial buildings based on historical usage data, weather patterns, and occupancy.
9. **Credit Risk Assessment**: Predicting the likelihood of default on loans or credit cards based on credit history, income levels, and other financial indicators.
10. **Employee Attrition Prediction**: Forecasting the likelihood of employee turnover based on factors such as job satisfaction, tenure, and performance metrics.
11. **Crop Yield Prediction**: Predicting agricultural crop yields based on factors such as weather conditions, soil quality, and crop management practices.
12. **Insurance Claim Prediction**: Forecasting the likelihood and severity of insurance claims based on historical claims data, demographics, and policy details.
13. **Customer Segmentation**: Using regression analysis to segment customers based on their purchasing behavior, preferences, and demographics.
14. **Traffic Flow Prediction**: Predicting traffic congestion and flow patterns on roads based on historical traffic data, weather conditions, and special events.
15. **Website Traffic Prediction**: Forecasting website traffic based on historical visitor data, marketing campaigns, and seasonality.

These are just a few examples of regression use cases where AI techniques can be applied to make predictions and drive decision-making in various domains.

**Libraries need for regression and classification**

* import tensorflow as tf
* import pandas as pd - data frame manipulation (pd.concat(data,column))
* import numpy as np – numerical analysis
* import seaborn as sns – data visulaization
* import matplotlib.pyplot as plt – data visualization

how do you not overtrain the model and what is the problem ?

if we over train the model we will not get accurate results, we need to generalize the training model, that means if we over train the model , the AI will think only that particular training is correct or only the image used for training the model is the correct one rest of the other images which are part of testing data set are not correct though they are similar, and this model is of no use, so we should avoid over train the model.

How to generalize the model then ?

Try to have a validation data set , divide the training data into validation data set and while training the model do cross verification with the validation data set

Confusion matrix ?

**Convolution layer in Artificial Intelligence**

Of course! Imagine you have a magical magnifying glass that you can use to look at pictures really closely. Now, let's say you have a picture of a dog. When you use the magnifying glass, you can focus on one small part of the picture at a time, like the nose or the ear.

In artificial intelligence, a convolutional layer is like that magical magnifying glass. But instead of looking at pictures, it looks at different parts of an image to understand what's in it. Just like you focus on different parts of the picture with your magnifying glass, a convolutional layer looks at small chunks of the image at a time.

These small chunks are called "filters" or "kernels". They're like tiny templates that the convolutional layer uses to see if there's anything interesting in that part of the image. For example, one filter might be looking for edges, while another might be looking for colors or patterns.

The convolutional layer moves these filters across the entire image, scanning it bit by bit, just like you move your magnifying glass around the picture. Each time it scans a part of the image, it combines the information it finds there to create a new, simplified version of the image. This new version helps the computer understand what's in the picture better.

So, in simple terms, a convolutional layer in artificial intelligence is like a magical magnifying glass that helps a computer understand pictures by looking at small parts of them at a time.

How to we change the picture to an input of the neural networks , ?

We apply convulation layer (it contains subsamples , filters, kernal) and followed by flattening the network which becomes input to the neural network.

Lenet Architecture