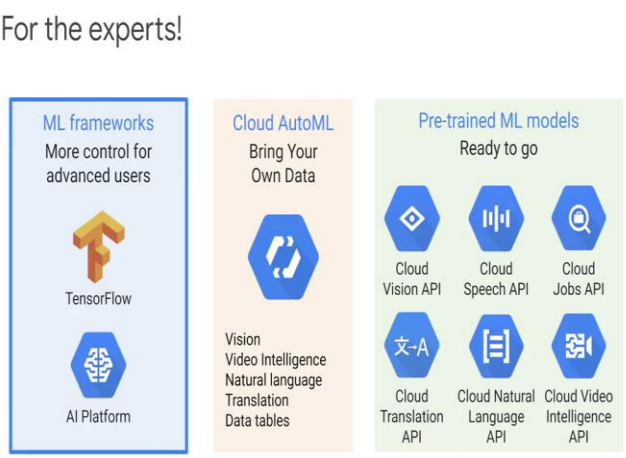
Google Cloud Computing Foundation Course
Evan Jones
Technical Curriculum Developer
Google Cloud

Lecture-79
Building Bespoke ML models

In this topic you will take a high-level look at the complexities behind developing Bespoke machine learning models but how also AI platform Google's managed machine learning service makes it easier for machine learning developers data scientists and data engineers to take their machine learning projects from ideation to production and deployment.

(Refer Slide Time: 00:22)

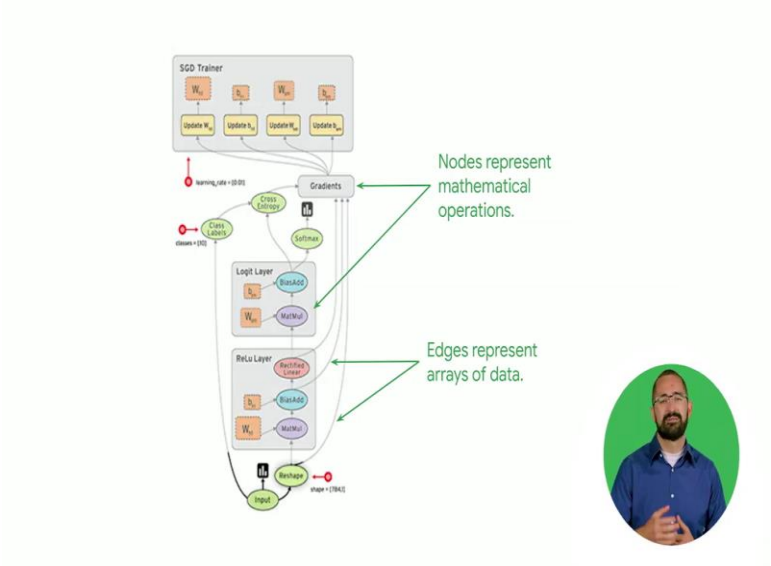


Earlier you are introduced to the idea that leveraging machine learning can split into three areas in this topic you look at the more complex but also the most adaptable option of those three.

(Refer Slide Time: 00:36)

As a starting point let us talk a little bit about tensor flow tensor flow is an open source high-performance library for a numerical computation not just about machine learning any numerical computation. In fact people have used tensor flow for all kinds of GPU computing for example you can use tensor flow to solve partial differential equations these are very useful in domains of like fluid dynamics. Tensor flow as a numeric programming library is appealing because you can write your own computation code in a high-level language like Python and have it be executed in a very, very fast way that matters at scale.

(Refer Slide Time: 01:11)



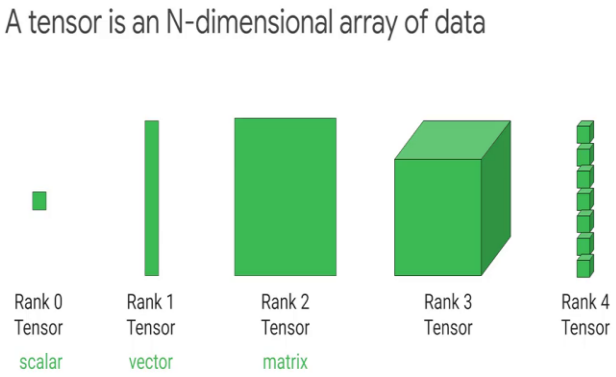
The way tensor flow works is that you create a directed acyclic graph or a dag to represent your computation directed means it has a direction of flow a cyclic means that it cannot feed into itself

meaning that it is not a circle and graph because it has those nodes and edges. In this schematic those nodes represent mathematical operations things like adding, subtracting and multiplying and also very more complex functions.

Here for example softmax, matrix multiplication and so on are mathematical operations that are part of this directed graph. Connecting the nodes are those edges the lines the input and the output of those mathematical operations. The edges represent arrays of data essentially the result of computing the cross entropy is one of those three inputs to this bias add operation and the output of the bias add operation is sent along to the matrix multiplication operation or mat goal.

The other input to the mat mill you need two inputs to multiply matrix together the other input is the variable or that is the weight.

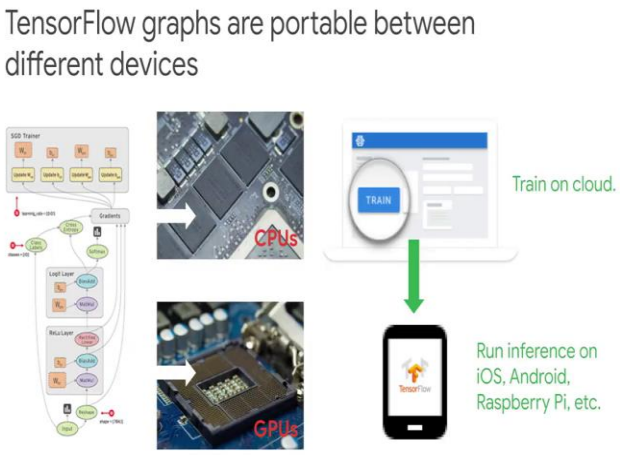
(Refer Slide Time: 02:26)



So, where does the name tensorflow come from in math a simple number like 3 or 5 is called a scalar a vector is a one-dimensional array of numbers in physics a vector I know everyone's trying to remember it right now is something that has what magnitude and direction but in computer science we as a vector to mean one dimensional or 1d arrays. A 2 dimensional array is called a matrix, a 3-dimensional array well we just call it a 3d tensor.

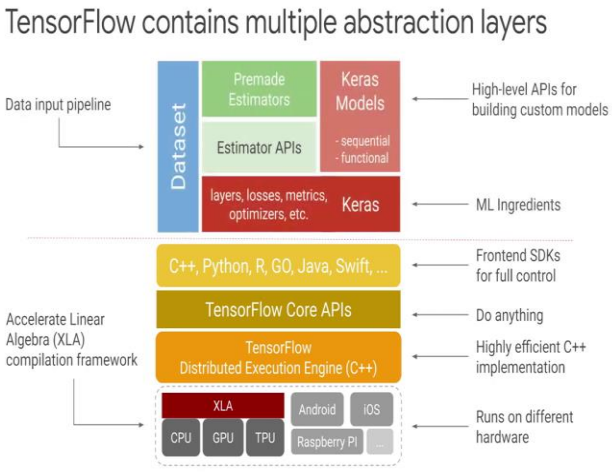
So a scalar vector matrix 3D tensor 4D tensor and so on. A tensor is therefore an n dimensional array of data so your data and tensorflow are tensors they flow through the graph hence tensor flow.

(Refer Slide Time: 03:23)



You can build a dag in Python stored in a saved model and restore it in a C++ program for low latency predictions, you can use the same Python code and execute it both on CPUs and GPUs. This provides language and hardware portability.

(Refer Slide Time: 03:45)

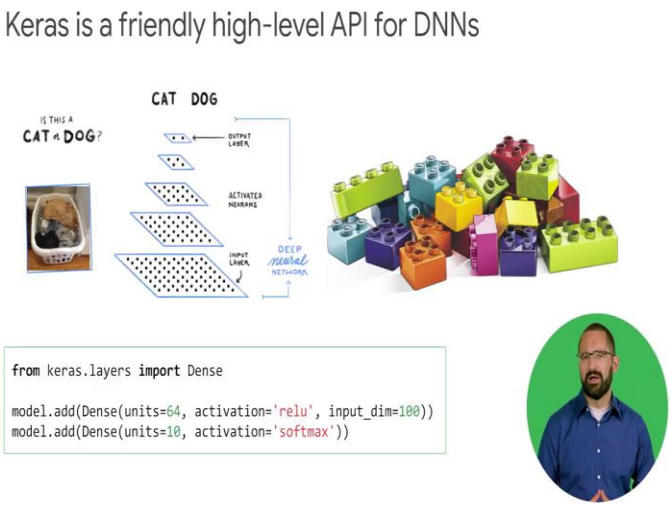


Like most software libraries tensorflow contains multiple abstraction layers the lowest level of abstraction is a layer that is implemented to target the different hardware platforms like running

ML on your mobile device. In the latest version of tensor flow you also get the accelerate linear algebra or excel lay framework which is a faster compiler for all the math that underpins your ML models. Again very low-level stuff here they should probably know exists but probably would nott interface with directly.

On top of that hardware and moving up to more abstraction is the execution engine for tensor flow written in C++ for highly efficient operations you could write me an entire tensorflow function if you wanted to in C++ and register it as a tensorflow operation. Generally data scientists will use the api's which are a bit more abstract and next on our list. Next are the familiar front-end SDKs or software development kits where we can use C++, Pytho, Go, Java etc to access your tenso flow operations. I will be honest with you though a lot of my work in tensorflow uses pre-built ML ingredients provided through the keras and dataset api's keras is a high-level neural networks api written in Python and it can run on top of tensorflow. Here is how friendly and abstract the Keras library is.

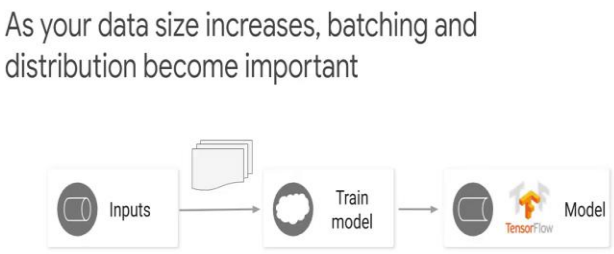
(Refer Slide Time: 05:13)



Remember the deep neural network before that found the dog that is hiding in the laundry basket if you are going to use Kares creating and adding those layers into that DNN would simply be model dog ad and how large you want to layer to be in units. There are some other concepts for image models like soft Max and Riilu layers that you will pick up when working with these

different algorithms but the code itself for building these layers is just like stacking block after block.

(Refer Slide Time: 05:40)

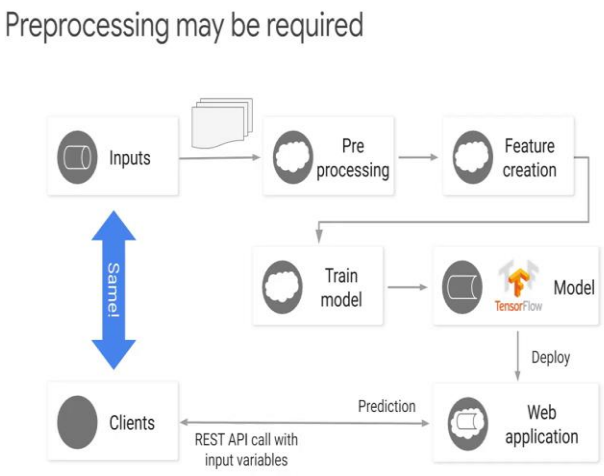


Now let us talk about data size if you have a small dataset like one that just fits in memory pretty much any ML framework will work or Python and so on they have these statistical packages that often in three or four lines of code and it will just work. But these are in memory data sets once your data sets get larger these packages would not work. You will need to split your data into batches and then train. However you are also needed to distribute your training over many, many machines.

Now sometimes people think that they can take a short code in order to keep the training simple by getting a bigger and bigger and bigger single machine with lots of GPUs but that is not the case scaling out is the answer not scaling up. Another common shortcut that people take is to just sample their data. So, that it is small enough to let the ML use on the hardware that they happen to have. They are limiting the effectiveness of ML by not using all of the data for the machine on a model.

From using all of that data and then devising a plan to collect 10 times that data that they currently have is often the difference between an ML that does not work an ML that appears magical.

(Refer Slide Time: 06:53)



Some of the other major improvements to ML happen with human insights come into the problem. In ML you bring human insights that's what your experts know about the use case of the data set in the form of refinements to existing features or the addition of new features in a process that is often called feature engineering you will also need to pre-process your raw data scale it and code it and these two things and a large data set also need to be distributed and then done in the cloud for scale.

Now that is just on the training side once you have completed and they successfully trained your model you want to deploy it for use in production at that point the performance characteristic changes instead of just thinking how long it takes to train on your training data set you must also think about how it is going to support the number of prediction queries per second or QPS that you are going to need.

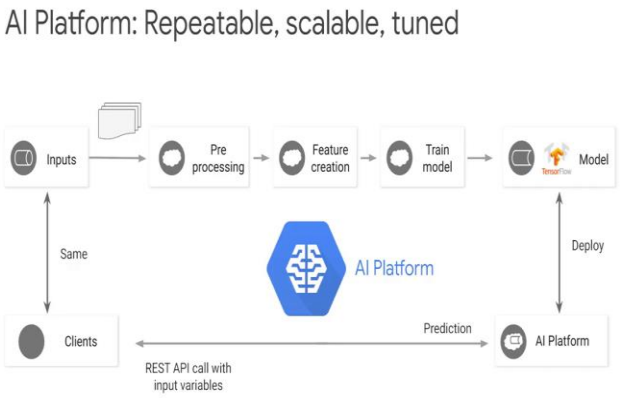
That requires your solution being able to scale the prediction code as necessary to support the users who need to make those timely predictions. Now the type of questions that you will ask yourself here in the serving of your model are what if the underlying model changes you have retrained it what if the parameters used in the model need to change what if the number of inputs or the data changes. Do you really want to expose all this to your users.

What if the client is not in the language that you used to train the model well you can invoke the tenorflow API from pretty much any programming language and you can use cloud server resources to automatically scale out to as many of QPS or queries per second that you need for those predictions. But these are things that need to be managed and that can be a challenge if you need to scale out that solution rapidly.

Earlier we mentioned feature engineering and how to build those pipelines to pre-process your training data before training. This same pre-processing must also happen at prediction time but beyond is cleaning up the data there is a variety of ways that your trained model could be a bit different than your prediction one. Using a standard like AI platform helps minimize these issues. Once you have rarely talked about your prediction inputs will often be systematically different than the ones that you use to train your model with.

In subtle in hard to detect ways maybe the average of some column has shifted or the variance has grown over time this is not a phenomenon that we call training, serving, skew and detecting it requires continual data collection and re-examination.

(Refer Slide Time: 09:36)



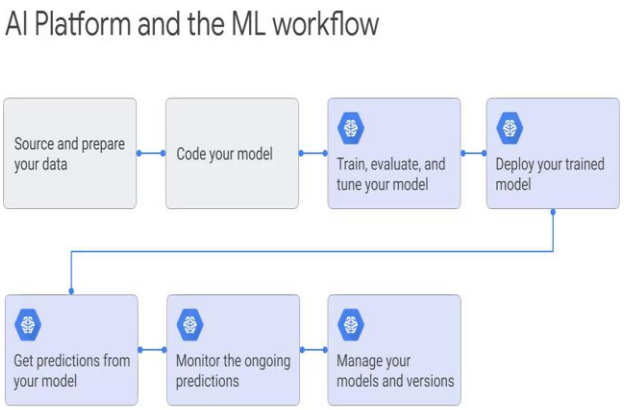
AI platforms simplifies this for you ensuring that the train model is what you actually run in other words it helps you handle that training serving skew. A platform will keep track of all those pre-processing and feature engineering steps for you as well as allow you to version your model

over time. In training a platform will help you distribute that pre-processing and train your model multiple times iteratively and then help you deploy your train model to the cloud for predictions.

Now when we talk about predictions the ML model is accessible through something like a simple REST API and it includes all the pre-processing and feature transformation that you did so the client code can simply supply the raw input variables and get back a prediction. A AI platform can also distribute the model as needed to supply a high number of queries per second those are the people that want to make predictions with your trained model with ml you need both high quality execution at training and prediction time.

If you have got a great model but it is super slow for those timely predictions no one's going to use it. While the computation of a tensorflow model once is relatively cheap the point of an ML model is to make those predictions for lots of incoming requests.

(Refer Slide Time: 10:56)



Let us look at a diagram that provides a high-level overview of the stages in the ML workflow. The blue fill boxes will indicate where a I apply it form provides managed services and api's for you use. You must have access to a large data set of training data that includes the attribute that lets the label the correct answer for machine learning and that is what you want to be able to infer or predict for the future based on all the other data that you have your other inputs are called features.

For example assume that you want your model to predict the sale price of a house begin with a large data set describing the characteristics of houses in a given area including things like the sale price of each house and when it was sold on so on and so forth. After you have sourced your data you have got to analyze it and understanding the data and pre-processing to get it ready for machine learning. In this pre-processing step you transform valid clean data into the format that best suits the needs of your model.

Tensorflow has already has a lot of pre-processing libraries that you can use automatically with AI platform in addition consider other GCP services that we talked about like bigquery cloud dataproc cloud data fusion and cloud data prep to help you with those transformations. A lot of the work and machine learning is just getting that clean data ready for machine learning. Then you can develop your model using established ML techniques or by defining new operations and approaches. You can start learning how to create your model by working through the documentation provided by tensorflow sukuk learn and XG boosts.

AI platform provides the services that you need to train and then evaluate your model all in the cloud. When training your model you feed it that data or those input features for which you already know the value of that target data attribute the historical right answer again that is called the label. You then run the model to predict those target values for your training data so that your model can adjust its settings to better fit itself to the data and predict the target value more accurately this is the whole learning part of machine learning.

Similarly when evaluating your trained model you feed it data that includes the target values you compare the results of your models predictions to the actual values for the evaluated data and then you use statistical techniques appropriate to your model to gauge its success that is how well it learned. You can then tune the model by changing the operations or the settings that you use to control for training purposes these are called hyper parameters such as the number of training steps to run in training.

This technique of adjusting those model knobs is called hyper parameter tuning. AI platform provides tools to upload your trained ML model to the cloud so then you can send prediction requests to the model. In order to deploy your trained model on AI platform you first must save your model using the tools provided by your machine learning framework. This involves serializing the information that represents your trained model into a file then you can then simply deploy for prediction in the cloud.

On GCP you would upload the saved model to a Google cloud storage bucket and then create a model resource on AI platform and you just specified that cloud storage path to where your saved model is located. AI platform provides the services you need to request predictions from your model in the cloud. There are two ways to get predictions from trained models online prediction sometimes called HTTP prediction and batch prediction in both cases you pass input data to your cloud hosted machine learning model and you get those inferences for each data instance.

And you can monitor the predictions on an ongoing basis and AI platform provides API is to examine all your running jobs in addition various GCP tools support the operation of your deployed model such as the entire suite of tools within stackdriver. AI platform provides various interfaces from managing your model and model versions including a REST API the G cloud AI platform command line tool and through the GCP console.