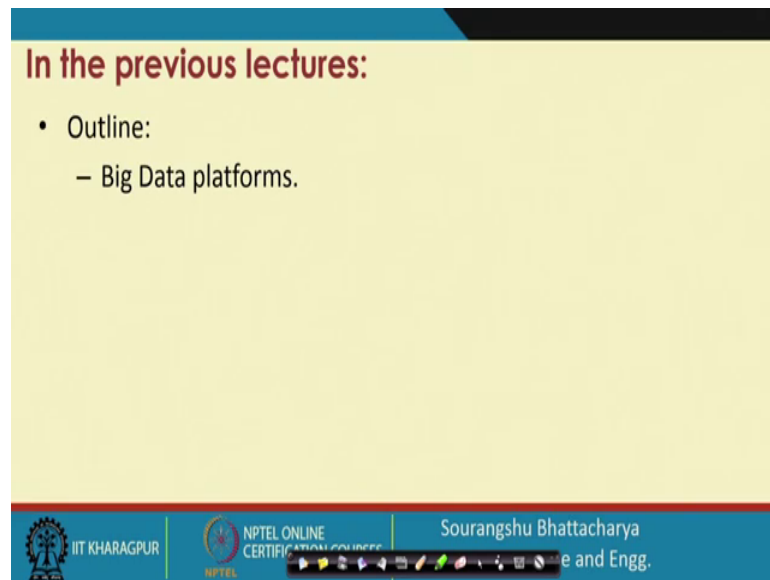


Scalable Data Science
Prof. Sourangshu Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 20
Distributed Machine Learning

Hello everyone welcome to the NPTEL course on Scalable Data Science, I am Professor Sourangshu Bhattacharya from Computer Science and Engineering at IIT Kharagpur. This is lecture number 20 and today we are going to introduce the Distributed Machine Learning so, yeah.

(Refer Slide Time: 00:39)



In the previous lectures:

- Outline:
 - Big Data platforms.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSE | Sourangshu Bhattacharya
Department of Computer Science and Engineering

So, in the previous lectures we have already covered the big data platforms that is Spark and we have covered various computations using the big data platform.

(Refer Slide Time: 00:51)

In this Lecture:

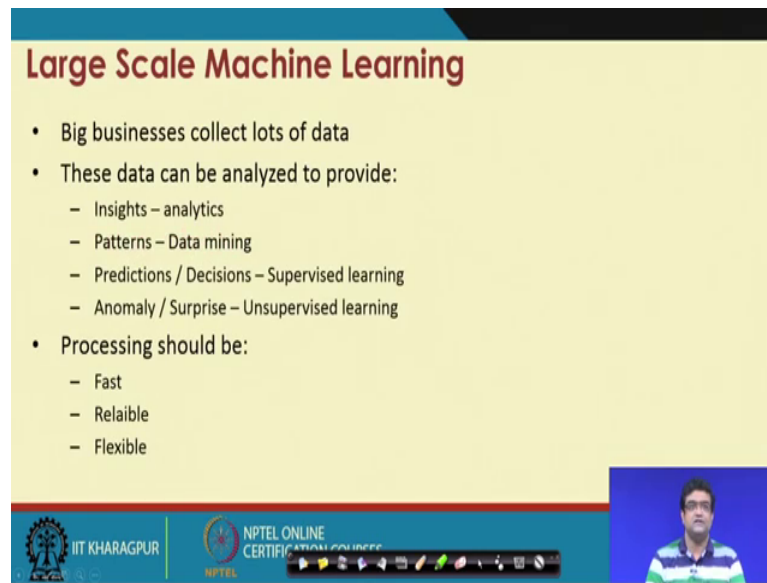
- Outline:
 - Motivation
 - Large scale machine learning -
 - Edge computing – autonomous vehicles
 - Architectures
 - Platforms
 - Tensorflow

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSE | Sourangshu Bhattacharya

So, in this lecture first we will cover the motivation of why distributed machine learning is important ok. So, we will cover two in two cases: one is large scale machine learning and other is edge computing which is autonomous vehicles. But, there are many many examples, but these two are representative of two different types of examples. Then we will discuss an over view of the types of architectures that we can have for distributed machine learning.

And then we will also cover in brief at a very high level the platform tensorflow which is which can also perform distributed machine learning. And, we will see how it is different from the Spark platform and then we will see some comparisons between the between these platforms ok:- [Soso](#), large scale machine learning.

(Refer Slide Time: 02:04)



Large Scale Machine Learning

- Big businesses collect lots of data
- These data can be analyzed to provide:
 - Insights – analytics
 - Patterns – Data mining
 - Predictions / Decisions – Supervised learning
 - Anomaly / Surprise – Unsupervised learning
- Processing should be:
 - Fast
 - Reliable
 - Flexible

The slide also features logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSE, along with a video inset of a speaker in the bottom right corner.

So, as we know big businesses collect lots of data. So, these big businesses could be in many many domains we shall see some of the domains. So, these could be in internet domain, these could be in finance domain, these could be in E-commerce domain, social networks all these domains ok. And so, ~~this these~~ these data can be analyzed. So, to provide first of all insights which is called the analytics then you can so, insights is basically just some trends or some monitoring kind of thing. So, this is the simplest outcome that you can get by analyzing this big data.

Second is patterns which is called data mining. So, you may find some correlations between some quantities which are not trivial to understand. Then third is you can so, so example of pattern could be something like you may see that there is a correlation between a certain product that your company is selling and the other product that your company is selling. So, the two may be either positively correlated or negatively correlated and then you can find predictions, you can do predictions with the lot of data that you have collected. So, this is normally called supervised learning.

So, if there are many many examples like you can try to predict stock prices, you can try to predict the click through rate of a user so on and so forth. And finally, you may want to find things like anomalies or clusters or surprises in the data which are also come in the realm of supervised learning:- ~~Something something~~ like clustering all these kinds of anomaly detection etcetera. Now, all these companies they have this requirement that

these are. So, these are business critical applications and hence all this kind of processing should be fast.

So, it should be over within a limited amount of time, let us say may be 1 hour or 1 day whatever it is, but it should be fast. So, speed is important and it should be reliable. So, it the computation should have happen reliably, it should not fail or something and the processing should be flexible. So, so you should be able to change. So, depending on the changing scenario of the business you should be able to change the computation little bit without having to incur very large overhead.

(Refer Slide Time: 05:27)

The slide is titled "Large scale Machine Learning" and features a table with two columns: "Industry" and "Use Case". The table lists four industries: Financial Services, eCommerce, Gaming, and Healthcare, each with specific use cases. At the bottom of the slide, there are logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSE, along with a small video inset of a speaker.

Industry	Use Case
Financial Services	<ul style="list-style-type: none">• Show correlation between services purchased and investments/trades made• Identify customer segments• Recommendations for research articles to drive trading
eCommerce	<ul style="list-style-type: none">• Show types of events person will like• Decision tree based on likelihood to click through• Recommendations for a large "cold start" population
Gaming	<ul style="list-style-type: none">• Clustering for user profiles• Correlation between attributes of a game and behavior• Churn analysis
Healthcare	<ul style="list-style-type: none">• Recommend tests or other offerings• Identify factors/trends that lead to disease

~~ok~~ So, some examples:- So, for example, financial services industry you could show correlation between the services purchased and the investment made. So, how much investment they are making versus what all services they are purchasing from the company or you could identify the customer segment whether the customer is wealthy or not wealthy. You could recommend research article to people who are doing online trading. For example, you could recommend articles which will be useful for them in making decisions on online trading.

Then in the E-commerce domain you can do recommendation like you can show people, events or things that they like or you may try to predict the click through rate or conversion rate. So, you may try to predict something like if you show a particular ad what is the chance of this person taking on that ad and then you may want to do

recommendation for cold start:- ~~So~~so, people on which you do not have past data ok. Similarly, in the gaming industry there are lot of problem.

So, for example, you could try and cluster user profiles, you could find correlations between attributes of a game and behavior and you could try to predict churns. So, you could try to see whether a particular online gamer is leaving this particular game and going to some other game or not, similarly in health care. So, these are so, as you can see there are lots and lots of industries with large amounts of data. So, this data is collected from the user and basically doing machine learning large scale machine learning is a very important business for all of these companies ok.

(Refer Slide Time: 07:41)

Big Data

Search engine index: 10^{10} pages (10^{12} tokens)

Search engine logs: 10^{12} impressions and 10^9 clicks every year

Social networks: 10^9 nodes and 10^{12} edges

Tasks	Typical training data
Image classification	Millions of labeled images
Speech recognition	Thousands of hours of annotated voice data
Machine translation	Tens of millions of bilingual sentence pairs
Go playing	Tens of millions of expert moves

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSE

So, ~~so~~what are the things that you have ~~basically~~.basically? So, ~~so~~you have the big data. So, for example, the search engine has 10 to the power 10 pages or social networks of 10 to the power 9 nodes and 10 to the power 12 edges and then you have to do machine learning. So, you may have to do image classification on millions of labelled images.

So, this is the image made data set for example, or you can do speech recognition on again thousands of hours of annotated voice data. Or you can do machine translation tens of millions of bilingual sentence pairs and or something like you could try to teach an algorithm how to play go for example. And, for that again you have lots of expert moves already available ok.

(Refer Slide Time: 08:45)

Big Models

LightLDA: LDA with 10^6 topics (10^{11} parameters); More topics → better performance in ad selection and click predictions

DistBelief: DNN with 10^{10} weights; Deeper and larger networks → better performance in image classification.

Human brain: 10^{11} neurons and 10^{15} connections, much larger than any existing ML model.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSE

So, this is the first characteristics the second characteristic could be that you could have big models ok. So, so for example, the Light_LDA model which was published it had it is a basically LDA model. It will be a directly allocation model with 10 to the power 6 topics and 10 to the power 11 parameters. So, and what basically it is seen that the more topics you add the better performance you get in a downstream applications. So, for example, add selection or click prediction. So, so, and similarly for example, the DistBelief paper from Google they had 10 to the power 10 weights.

So, they were training deep neural network for in its classification and this is the order of parameters that you learn from the data. So, you have a big data and you also have big models that you want to train from data. And, the point here is that models are only going to grow bigger because, still we are not matching. So, so the human brain has something like 10 to the power 11 neuron 10 to the power 15 connections. So, we are ~~not~~ still not matching the size of the human brain. So, that number of parameters that the human brain might have.

(Refer Slide Time: 10:18)

The slide is titled "Big Compute" and features a bullet point: "Large computer clusters and highly parallel computational architectures". Below this, there are three images: a server room labeled "Cloud Computing", a rack of GPUs labeled "GPU Cluster", and a board of FPGAs labeled "FPGA Farm". The slide also includes logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSE, along with a small video inset of a speaker.

And, to handle of course, all this big data and big models we need big compute facilities. So, we have now cloud computing, we have large number of servers. On the other hand, we have highly parallel computers like GPUs which can execute simultaneously instructions on a thousands of data points at in 1 clock speed and then you can have clusters of GPUs. So, these are very popular now in compute hungry systems like deep learning etcetera. And, then there are other kinds of things like FPGA Farms, your Raspberry Pi kind of devices.

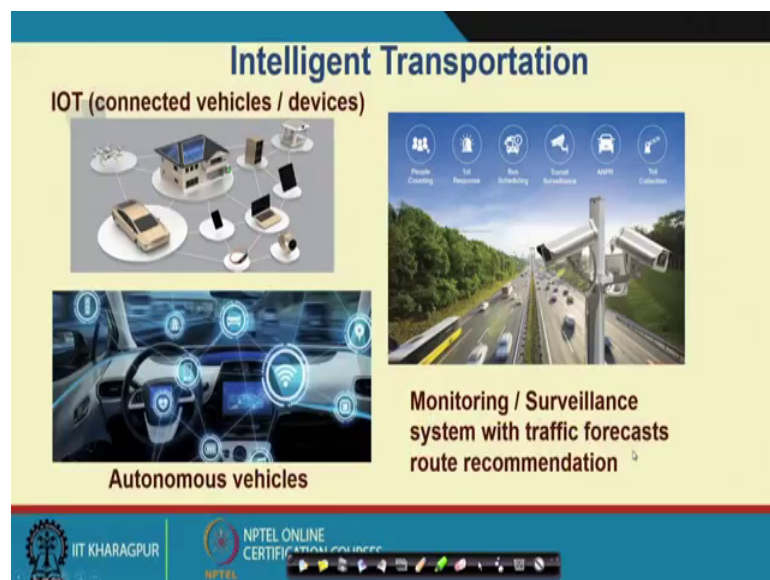
All of these could help you speedup and parallel rather speedup computation in a distributed or parallel manner very very fast. So, so this is one reason why you need distributed machine learning. Or so, in order to train for all the scenarios that I just describe you need to have distributed algorithms for machine learning ok. Now, I am going to describe another somewhat orthogonal scenario. So, in the previous scenario, that the data was coming to big company and then, these companies are processing them in a.

So, so because so the distributed computing is required mainly because of the large scale of data ok. So, the other paradigm of computing which is called edge computing's. So, in edge computing the idea is that our devices are becoming more and more smart. So, now your cell phone has as much processing power as your laptops would have even 5 years back ok. And, if you look at if you look at other devices like refrigerators, TV's,

cars etcetera all these devices are having you know a large amount of compute facility on board them.

So, the idea is that can we put some part of the computation and all these devices are collecting data and many of these devices are actually running machine learning algorithm. So, for example, your cell phone is running as speech algorithm, it is it is probably running many other machine learning algorithms. So, ~~so~~ the point is that can we have machine learning architecture which is truly distributed rather than centralized; so, that much of the computation can happen on edge devices.

(Refer Slide Time: 13:39)



–So, I will go in to this use case of intelligent transportation. So, basically what happens now is that you have lots of devices. So, we are in the era of internet of things. So, we have lots of connected devices and intelligent devices and these devices are collecting data and they also have processing power. So, for example, your car has a computer in between so, and it also has camera, it has LIDAR, it has probably some proximity sensors, it has a GPS receiver and all these things. So, that you know it can sense your position, it can see whatever the things that are around you, it can create a map of the terrain around you and so on and so forth.

So, all these things can be useful in for example, learning a system which will make the car drive by itself. So, this is the idea for autonomous vehicles that all these sensors they collect data and then we analyze this data and then we use some machine learning

techniques to try and predict. For example, what should be the steering angle? What should be the throttle? Should you break etcetera etcetera ok. So, this is one application another application could be monitoring a surveillance applications. So, now a days as you know there are CCTV cameras everywhere.

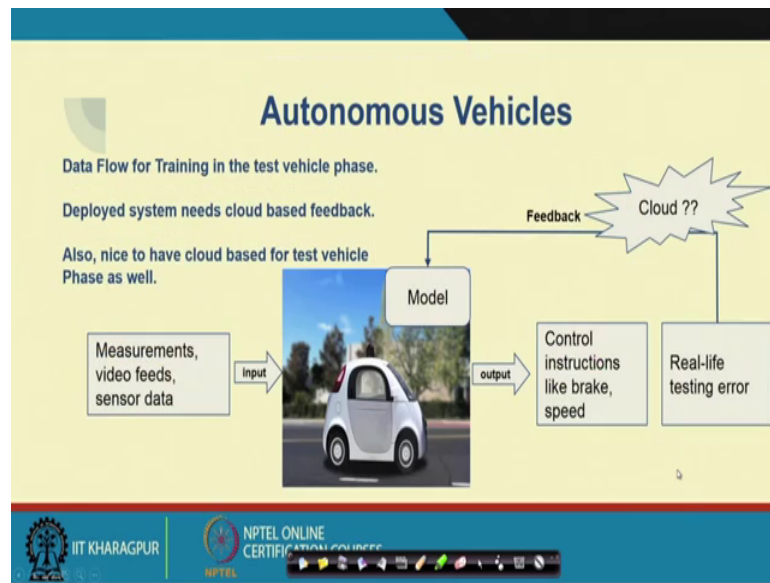
So, for example, in shopping malls, in public places, in roads etcetera ok. And for example, one application could be that can be monitor or can we do traffic forecasts ok. So, can we say using cameras mounted near different junctions almost all road intersections, now have ~~cameras-cameras?~~ So, ~~so~~-using those cameras can we predict or can we monitor the amount of traffic that is flowing through and then may be we can forecast the traffic and we can do road traffic recommendation ok. So, these are all machine learning ~~the these~~ these systems you use machine learning algorithms.

(Refer Slide Time: 16:18)

The slide is titled "Autonomous vehicles" and features a yellow background with a blue header. On the left, it states "Lots of sensor data are generated every second" and lists components of autonomous cars: Embedded Computers, GPS receivers, Short-range wireless network interfaces, and In-car sensors. It also mentions "Google's self driving car gathers 750Mb/s [Source]". On the right, there is a 3D visualization of a car's sensor range, showing a central car with various colored cones representing different sensors. Below the visualization, it says "Google Car 'sees' while making a turn". The slide footer includes the IIT KHARAGPUR logo and the NPTEL ONLINE CERTIFICATION COURSE logo.

So, the so, as I have already told that autonomous cars contain all these kind of data. So, so as you can see the Google's self driving car gathers around 750 mega bits per second ok. So, this is this is lot of data that the car is collecting.

(Refer Slide Time: 16:46)



So, the current model of training is that the car collects this data and it is in a offline mode ok. So, so basically what happens is in the training phase the car is asked to collect a lot of data. So, it collects the data from ~~this sensor~~ these sensors and then it also collects the human drivers feedback. So, how much is the human driver breaking? How much, what is this steering angle? Is it going left or right? And, all these things all these data it collects and then that data is sent to a to a system where we train the machine learning models and then the machine learning models are deployed to the car ok.

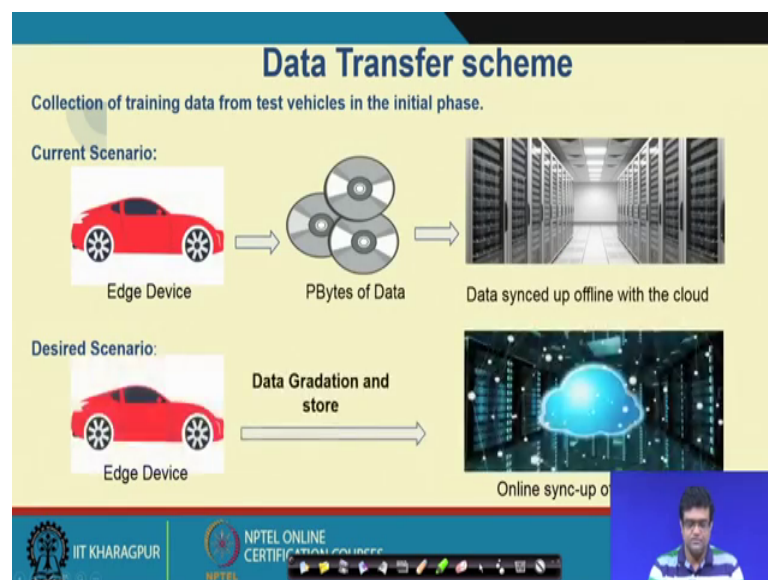
(Refer Slide Time: 17:52)



But as you can understand the so, all these vehicles are connected to the cloud ok. So, or at least in the future it is expected that the vehicles will be connected to the cloud ok. And, the question is can we transfer this kind of data in real time or at least one should not have to segregate out the training and the testing phase for autonomous vehicles. So, as you drive your car so, the autonomous vehicle should be able to learn how to drive the car or so, so that is the idea ok. And so, this is the form of distributed machine learning as you can understand because, there are many autonomous systems running each one running on one of the cars.-

And, then each one of this system they send the data to a may be a cloud in a real time manner. And, then the on the cloud the models get retrained or readjusted. And, maybe there is a slight variation in the way you prefer your car to be driven than somebody else then may be that or maybe there is a difference between how the traffic is ~~in a~~ in city 1 versus city 2. And, ~~may be maybe~~ you want to drive the cars somewhat differently in two different cities. So, this updated model may be deployed in the vehicle accordingly ok. So, this is another form of distributed machine learning that we will see more and more often in the future ok.

(Refer Slide Time: 19:48)



(Refer Slide Time: 19:52)

Need for data reduction

Futuristic scenario: Realtime edge to core data transfer from autonomous vehicles.

- Raw data collected by an autonomous vehicle per hour: 1 TB
 - 1440 Mbits per second (2 megapixel * 24 bits per pixel * 30 frames per second)
- Data compression: 1 : 0.014
 - Compression ratio: 70 : 1 using MPEG (source: Wikipedia)
- Bandwidth needed per car for realtime data upload: 3.9 MBps
 - Assuming 100 cars connect to a base station: 390 MBps
- Target edge to core data transfer rate: ~ 10 MBps (Fast ethernet)
-
- Compression required **39 : 1** over and above the current MPEG compression

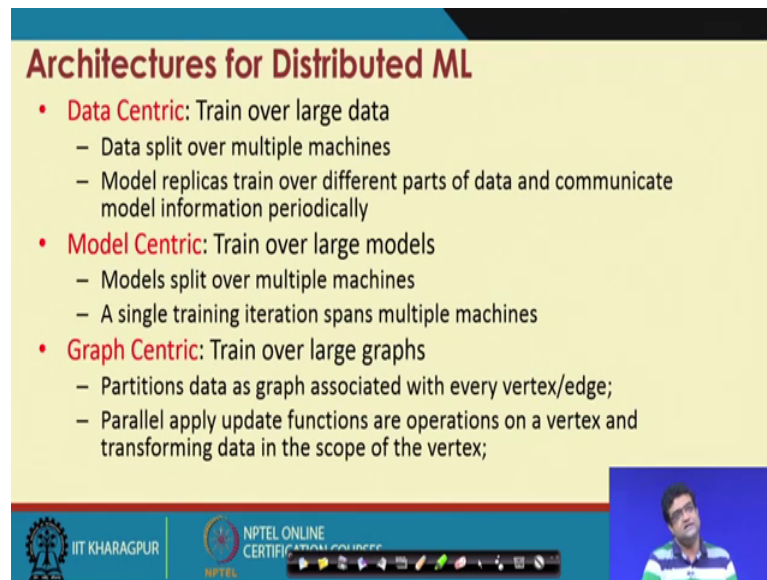
Reference: <https://devblogs.nvidia.com/training-self-driving-vehicles-challenge-scale/>

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATE COURSE | WAYMO

So, so this I have already described. So, this is some computation. So, basically the computation shows that we need large amounts of compression of the current data. Because, the rate at which the data is currently connected it goes to about 390 megabytes per second ok- Whereaswhereas, we should probably reinforce something like 10 megabytes per second of transfer, not more than that even that is actually high. So, then the question is can we do this compression using machine learning tools.

So, then the machine learning algorithm actually decides which machine learning algorithm which is running on the car actually decides which parts of the data to send to the cloud and which parts of the data not to send ok. So, so of course, there are many other scenarios where you need distributed machine learning. So, I have just described two scenarios which are very important in today's world. So, given this scenarios what are the architectures that we can use to do distributed machine learning.

(Refer Slide Time: 21:17)



Architectures for Distributed ML

- **Data Centric:** Train over large data
 - Data split over multiple machines
 - Model replicas train over different parts of data and communicate model information periodically
- **Model Centric:** Train over large models
 - Models split over multiple machines
 - A single training iteration spans multiple machines
- **Graph Centric:** Train over large graphs
 - Partitions data as graph associated with every vertex/edge;
 - Parallel apply update functions are operations on a vertex and transforming data in the scope of the vertex;

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSE

So, the first architecture which is the now the most commonly used architecture which is the one also we will be describing most in this course is the data centric architecture. So, the data centric architecture is mainly for the used case of training over large data:- [SoSo](#), training a machine learning model over large data. So, basically the idea here is that you use split the data over multiple machines and the model parameters that you have are replicated over all the machines. And, then you you compute the updates to the model on each machine based on the local data that you have on that machine.

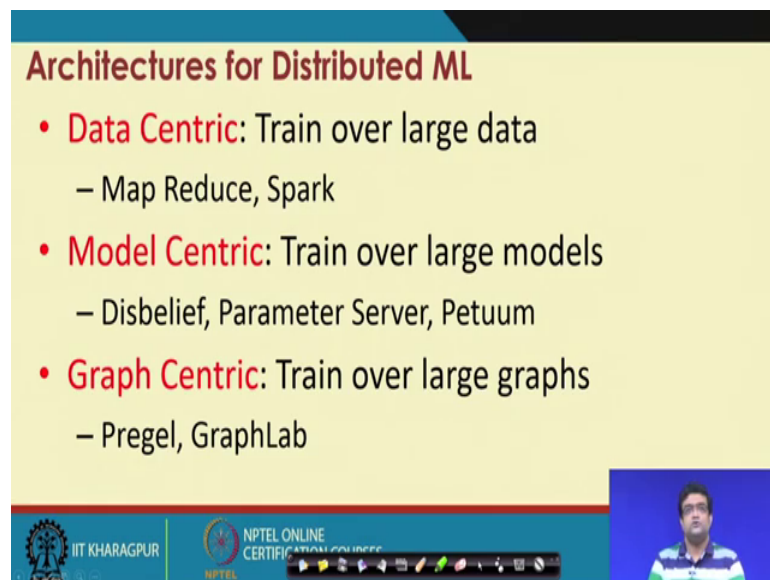
And, then you update the parameters on this machine and then you communicate the updated model or updated parameters with the other machines. So, you synchronize with the other machines either via central machine or via each other periodically. So, periodically you sort of synchronize this data with each other. Then the second approach which is still kind of it is being researched on and in certain cases it is very important is called the model centric approach. So, this is the situation where the model that you are trying to train is very large model. So, so what happens is that data is accessible to all the machines. So, whatever machines that you have you want to distribute the training of model over.

So, the data is accessible all the data is accessible from all the machines. but the model parameters are now split over different machines ok. And, the idea is that single machine updates model only a subset of model parameters on itself. And basically so, the single

update spans over multiple machines and in this case basically periodically you synchronize the parameters which are not updated in this machine from other machines. So, this is the second few of distributed machine learning second way of doing distributed machine learning. The third way is that your data itself comes in a graph. So, basically the, you train the model over large graphs so, your data itself comes in a graph. So, for example, it could be the web graph or it could be a large protein-protein interaction network in bioinformatics.

And then basically the so, the data is partitioned according to the nodes of the graph ok. And the computations happen on the nodes of the graph and basically the operations on a vertex use the transforming data in the scope of that vertex. So, only the vertices which are connected to this particular vertex are able to update the or rather effect the computation that happens in this particular graph. So, one example could be page rank computation or there could be many other examples like a statistical relational learning etcetera. Also, where the graphs are important or the input there are large input graphs which are important for the underlined machine learning model.

(Refer Slide Time: 25:44)



Architectures for Distributed ML

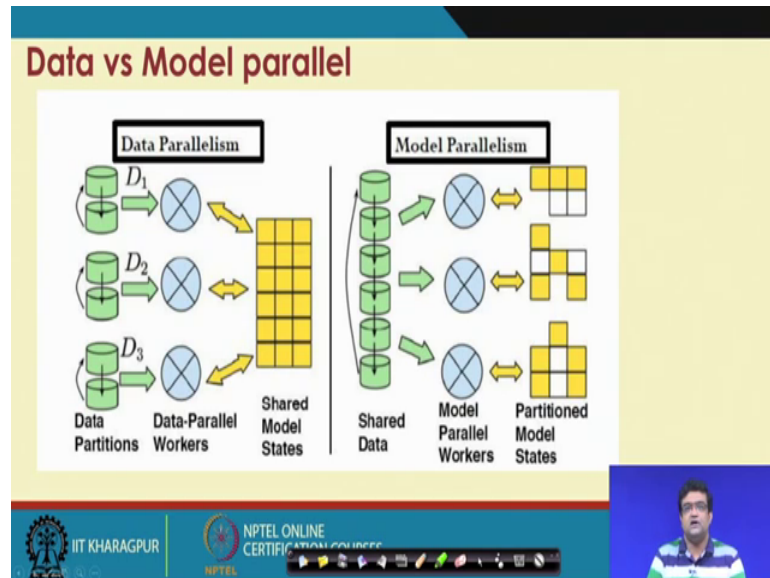
- **Data Centric:** Train over large data
 - Map Reduce, Spark
- **Model Centric:** Train over large models
 - Disbelief, Parameter Server, Petuum
- **Graph Centric:** Train over large graphs
 - Pregel, GraphLab

The slide also features logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, along with a small video inset of a presenter in the bottom right corner.

So, typically the data centric training, one would you use either Map Reduce or Spark. For model centric training typically one would you something like Disbelief or Parameter Server or Petuum. So, these are basically all these are some kinds of

parameter server and yeah. And for graph centric computation one might use something like Pregel or GraphLab which are basically graph based computation framework.

(Refer Slide Time: 26:26)



So, this is the difference between data parallelism and model parallelism that we have already discussed. So, here you can see that in data parallelism the data is partitioned and in model parallelism actually the model parameters are partitioned: [Andand](#), accordingly your update strategy and synchronization strategy changes ok.

(Refer Slide Time: 26:55)

TensorFlow

Tensors: n-dimensional arrays

Vector: 1-D tensor

Matrix: 2-D tensor

Deep learning process are flows of tensors

A sequence of tensor operations

Can represent also many machine learning algorithms

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, up till now we have so, so we have discussed the parameter for big data computation or rather scalable data mining, which is Spark which is as you can think that it is a data parallel or it is more suited towards data parallel computation. I will describe now another framework which is somewhat similar to Spark, but in its design, but it is very very it is much more suitable or much more widely used in machine learning and especially in deep learning ok. So, this framework is called tensor flows.

So, the basic idea here is that you have tensors which are n-dimensional arrays and then the process of learning or the algorithm of learning is about flow of tensors from one route to another in a graph. So, if you recall you also had computation graph in Spark where each node was RDD and data would flow from one node to another as you as the computation progress. So, here also the understanding is same except that the data that flows from one node to another is tensor ok.

(Refer Slide Time: 28:42)

With TensorFlow

```
import tensorflow as tf
```

$X \cdot W$

$$\begin{bmatrix} a_1 & b_1 & c_1 \end{bmatrix} \cdot \begin{bmatrix} W_{a_1} & W_{b_1} & W_{c_1} \\ W_{a_2} & W_{b_2} & W_{c_2} \\ W_{a_3} & W_{b_3} & W_{c_3} \end{bmatrix} = \begin{bmatrix} a_0 & b_0 & c_0 \end{bmatrix}$$

$y = \text{tf.matmul}(x, w)$

$a_0 = \text{relu}(a_0)$

$b_0 = \text{relu}(b_0)$

$c_0 = \text{relu}(c_0)$

$\text{out} = \text{tf.nn.relu}(y)$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSE

So, just to give you an example of how this works: So, you basically have in a neural network since situation like this where you have set of input layers here. And so, you have a set of input layers which are multiplied by this parameter w and then you get the output layer. And given many many examples you want to compute this output layer w. So, the way this is so, this is how you would mathematically represent the network.

So, you will have you can compute a 1 a 2 and a 1 b 1 and c 1 from so, these are the intermediate computation at this node, at this node. So, these are the intermediate

computation at this node and this is the input and when you multiply by w you get this and then you apply this relu function to get the actual output ok. Now, this you can do in tensor flow like this, that you can define the input as x, this parameter as w and then you can compute y which is `tf.matmul(x, w)`. And, then you can compute this out which is the output.

(Refer Slide Time: 30:21)

Define Tensors

x_{aa}	x_{ab}	x_{ac}
x_{ba}	x_{bb}	x_{bc}
x_{ca}	x_{cb}	x_{cc}

→ w

`Variable(<initial-value>, name=<optional-name>)`

```
import tensorflow as tf
w = tf.Variable(tf.random_normal([3, 3]), name='w')
y = tf.matmul(x, w)
relu_out = tf.nn.relu(y)
```

Variable stores the state of current execution

Others are operations

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSE

Now, so first thing is that how you define tensors. So, you define you can define two types of tensors, the first is first type is called variable. So, ~~this~~ these variables are tensors which store the state of the current execution ok. So, for example, in this case the w is a variable because, it is the parameter which will change as your training algorithm progresses. So, it will in some sense store the state of your computation as your learning algorithm progresses ok.

(Refer Slide Time: 31:15)

TensorFlow

Code so far defines a data flow **graph**
Each **variable** corresponds to a node in the graph, not the **result**
Can be confusing at the beginning

```
import tensorflow as tf
w = tf.Variable(tf.random_normal([3, 3]), name='w')
y = tf.matmul(x, w)
relu_out = tf.nn.relu(y)
```

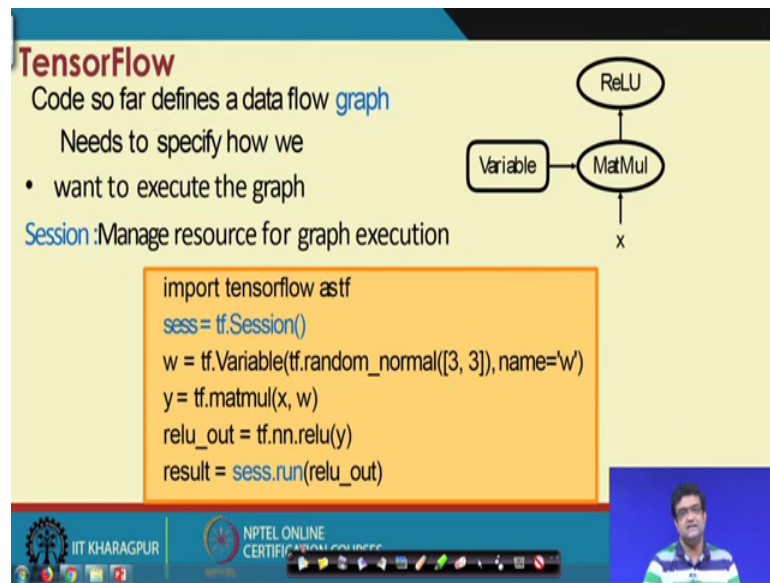
The diagram shows a computation graph with three nodes: 'Variable' (a rounded rectangle), 'MatMul' (an oval), and 'ReLU' (an oval). An arrow points from 'Variable' to 'MatMul'. Another arrow points from 'x' (a small 'x' below) to 'MatMul'. A third arrow points from 'MatMul' to 'ReLU'.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSE

So, if you write the code something like this which is basically encoding the computation that we just said then you [notice](#) that it is defining this kind of a graph ok. So, the node of [this graph](#) [these graphs](#) is unlike in case of Spark in this case the nodes are actually the operations ok. So, there are the input variable nodes and the most of the nodes in this graph are computation. So, this is the computation MatMul and this is the computation ReLU ok.

And, to this node and the data that flows from one node to another is a tensor. So, in from this node to this node basically the tensor x will flow and this node to this node the tensor which is stored in this variable will flow. And, then some computation will be done and the resultant will flow to this node and so on and so forth ok.

(Refer Slide Time: 32:34)



TensorFlow
Code so far defines a data flow graph
Needs to specify how we

- want to execute the graph

Session: Manage resource for graph execution

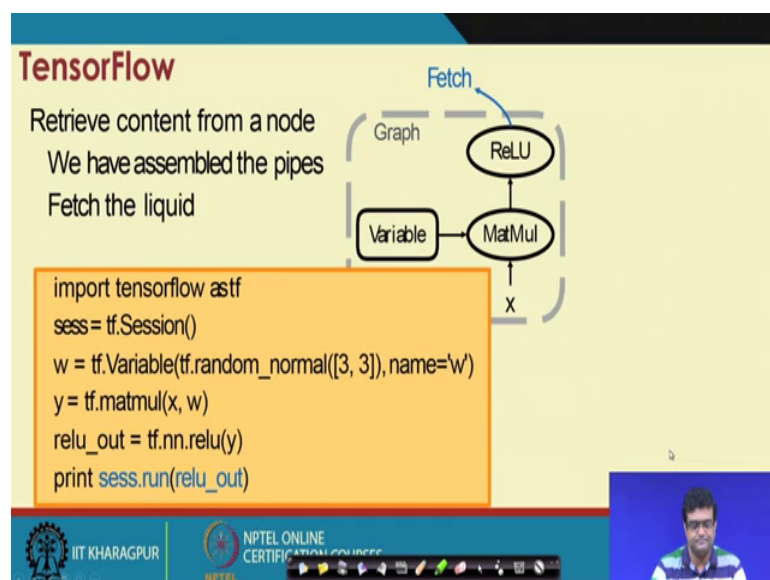
```
import tensorflow as tf
sess = tf.Session()
w = tf.Variable(tf.random_normal([3, 3]), name='w')
y = tf.matmul(x, w)
relu_out = tf.nn.relu(y)
result = sess.run(relu_out)
```

Diagram: A data flow graph with nodes Variable, MatMul, and ReLU. Variable and x are inputs to MatMul, and MatMul is an input to ReLU.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSE

So, ~~so~~ after you define the graph you have to learn the graph. So, you basically define a session. So, one way to think about the session is that this session is actually a graph, graph is the computation itself plus the data that you provide to the graph and then you start the session. So, so this is the so you define session and then you can run a graph in tensor flow using a session ok.

(Refer Slide Time: 33:17)



TensorFlow
Retrieve content from a node
We have assembled the pipes
Fetch the liquid

Diagram: A data flow graph with nodes Variable, MatMul, and ReLU. Variable and x are inputs to MatMul, and MatMul is an input to ReLU. A dashed box labeled 'Graph' encloses the MatMul and ReLU nodes. An arrow labeled 'Fetch' points to the ReLU node.

```
import tensorflow as tf
sess = tf.Session()
w = tf.Variable(tf.random_normal([3, 3]), name='w')
y = tf.matmul(x, w)
relu_out = tf.nn.relu(y)
print sess.run(relu_out)
```

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSE

Now, ~~now~~ what you have to do is you have to provide the data ok. So, how you provide this data x ok?

(Refer Slide Time: 33:36)

Variable

Variable is an empty node
Fill in the content of a
Variable node

```
import tensorflow as tf
sess = tf.Session()
w = tf.Variable(tf.random_normal([3, 3]), name='w')
y = tf.matmul(x, w)
relu_out = tf.nn.relu(y)
sess.run(tf.initialize_all_variables())
print sess.run(relu_out)
```

The diagram shows a computational graph with three nodes: 'Variable', 'MatMul', and 'ReLU'. The 'Variable' node is a blue box on the left. An arrow points from 'Variable' to 'MatMul'. Another arrow points from 'X' (input) to 'MatMul'. An arrow points from 'MatMul' to 'ReLU'. The 'ReLU' node is labeled 'Fetch'.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSE

So, first even before the data x so, you have this variable w. So, if you recall this was the variable w that we defined ok, right now the w is just an empty node. So, there is no data here. So, first you have to give that data. So, that is done by initializing the variable. So, you can for example, in this case randomly initialize the variable w ok. So, that is how you provide data to w.

(Refer Slide Time: 34:12)

Placeholder

How about x?

```
placeholder(<data type>,
            shape=<optional-shape>,
            name=<optional-name>)
```

Its content will be fed

```
import tensorflow as tf
sess = tf.Session()
x = tf.placeholder("float", [1, 3])
w = tf.Variable(tf.random_normal([3, 3]), name='w')
y = tf.matmul(x, w)
relu_out = tf.nn.relu(y)
sess.run(tf.initialize_all_variables())
print sess.run(relu_out)
```

The diagram shows a computational graph with three nodes: 'Placeholder', 'MatMul', and 'ReLU'. The 'Placeholder' node is a blue box on the left. An arrow points from 'Placeholder' to 'MatMul'. Another arrow points from 'X' (input) to 'MatMul'. An arrow points from 'MatMul' to 'ReLU'. The 'ReLU' node is labeled 'Fetch'.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSE

The second thing that you need to provide is you provide you need to provide data to x. Now, this x is different from w because, it is not storing the state of the computation

rather it is fixed, it is the piece of data that you are providing from outside ok. So, this kind of data is provided using what are called place holders ok. So, ~~so~~ and this is how so, you define first x as the place holder.

(Refer Slide Time: 34:46)

Feed

Pump liquid into the pipe

```
import numpy as np
import tensorflow as tf
sess = tf.Session()
x = tf.placeholder("float", [1,3])
w = tf.Variable(tf.random_normal([3, 3]), name='w')
y = tf.matmul(x, w)
relu_out = tf.nn.relu(y)
sess.run(tf.initialize_all_variables())
print sess.run(relu_out, feed_dict={x:np.array([[1.0, 2.0, 3.0]])})
```

Graph

Fetch

ReLU

Variable

MatMul

x

Feed

IIT KHARAGPUR

NPTEL ONLINE CERTIFICATION COURSES

And then you can feed the values for the fed of place holder as you run the session. So, this is the statement for a feeding the value of the place holder x as you run the session and this is how the whole computation is done and as yeah.

(Refer Slide Time: 35:19)

Comparison

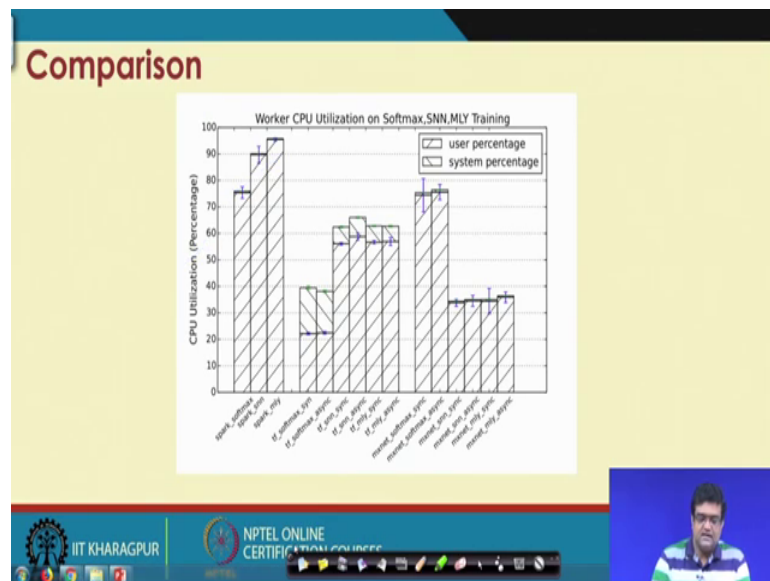
- Spark
- Tensorflow
- MXNet

IIT KHARAGPUR

NPTEL ONLINE CERTIFICATION COURSES

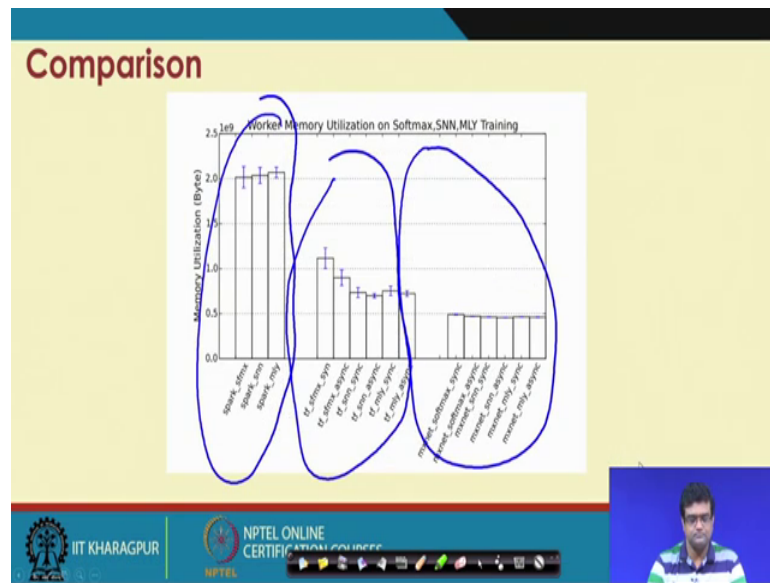
So, as we shall see or rather we will not go in to detail of exactly how a training is done. So, tensor flow provides also many other things like it provides optimizers and it provides gradient computation etcetera:- [Soso](#), that you can very easily train this kind of model ok. So, we just refer to a comparison of how these systems perform. So, this is three systems, first one is Spark. So, the comparison is between three systems Spark, TensorFlow and MXNet. MXNet is an also open source deep learning system which is especially designed for distributed computation.

(Refer Slide Time: 36:12)



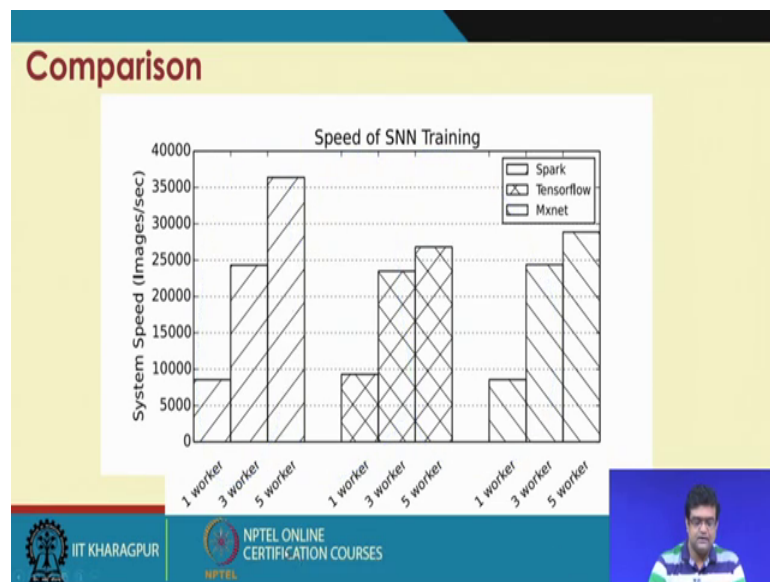
So, this result is over training of a single layered neural network using the MNIST dataset. So, this is the standard dataset in machine learning and you can see so, what we are showing here is the CPU utilization. So, the first group so, this group corresponds to Spark. So, these are three types of networks with Spark. This group corresponds to TensorFlow and this group corresponds to MXNet. And, what you can see is roughly the CPU utilization is slightly lower for TensorFlow and MXNet and slightly higher for Spark. This is because, this part is implemented in Java and that has its own overhead whereas, both TensorFlow and MXNet are implemented in C plus.

(Refer Slide Time: 37:17)



Same pattern follows when we try to plot the memory utilization. So, Spark users much more memory followed by TensorFlow and MXNet uses much less memory.

(Refer Slide Time: 37:32)

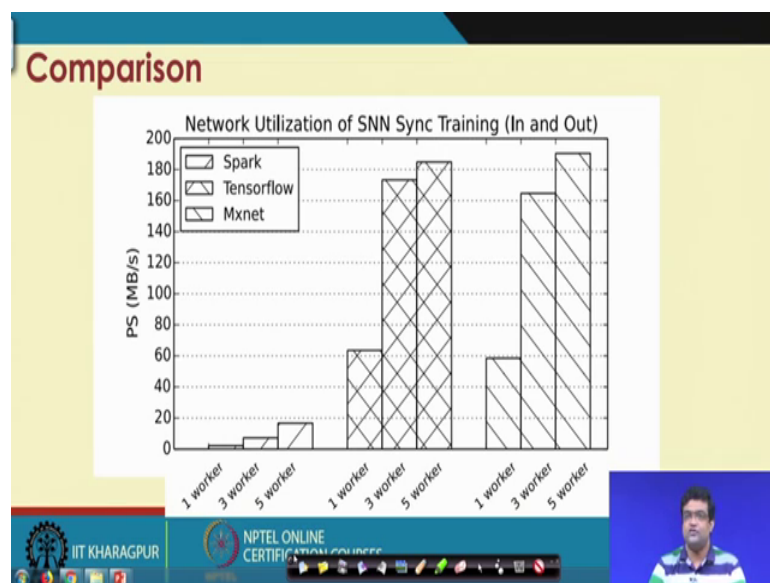


But you see now, when we look at the scalar so, on x axis for each of these tensor Spark, TensorFlow and MXNet, now Now we are showing the number of so, so the scalar as the number of workers. So, the first bar is for 3 workers, the second bar is for as the first one is for 1 worker, the second bar is for 3 workers and the last one is for 5 workers. On the y axis what we are showing is how many images or how many training images is this

system able to process per second ok. So, we see obviously, if we give it less number of workers it will be able to process less number of images; so, that that pattern we always see.

So, it is always increasing for so, this is this one is for Spark, this one is for TensorFlow and this one is for MXNet. Now, what we see here is that the scaling of Spark is much better. So, as the number of workers increases Spark system is able to scale much better than the other two [systems](#). This is because Spark is more designed most specifically designed for distributed computation. So, even though TensorFlow and MXNet can also perform distributed computation and they are in fact, much faster in case that you do single node computation, but as the number of nodes increases your Spark becomes somewhat faster in scaling.

(Refer Slide Time: 39:28)



So, this you see also in the network utilization. So, here now we are showing the network utilization for in the in the y axis for the same set. So, you can see that Spark utilizes much less network compared to TensorFlow or MXNet ok.

(Refer Slide Time: 40:00)

Conclusion:

- We have seen:
 - Motivation
 - Large scale machine learning -
 - Edge computing – autonomous vehicles
 - Architectures
 - Platforms
 - Tensorflow
 - Comparison

Handwritten notes on the slide:

Spark	TF
- DAG	- Cycles
- Immutable	- Variable

The slide footer includes logos for IIT KHARAGPUR, NPTEL ONLINE CERTIFICATION COURSE, and the name Sourangshu Bhattacharya.

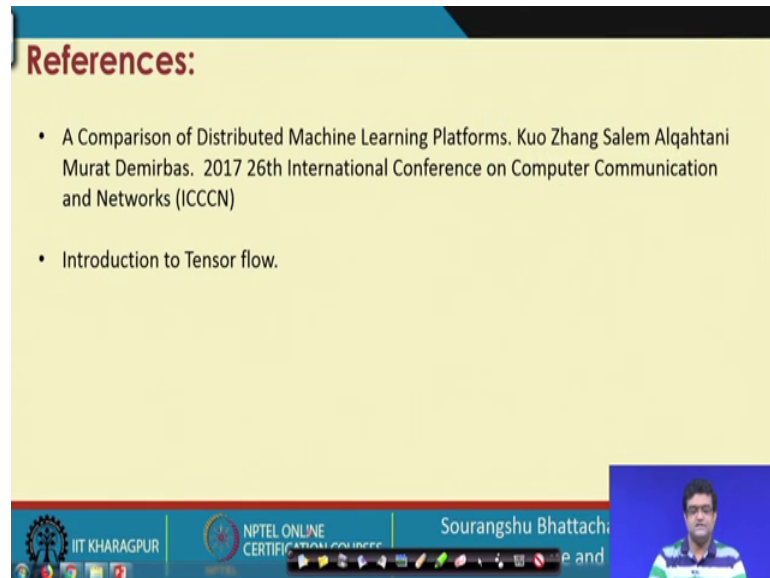
So, so just to conclude we have seen in two cases where distributed machine learning is very very important. And, we have seen the three possible architecture in which you can design distributed machine learning algorithms. We will see some specific algorithms in the in the following lectures and we have seen some platforms like Spark and TensorFlow and we have seen some comparisons.

So, basically what we have seen is that a Spark computation is a Spark platform is more designed for distributed computation whereas, TensorFlow is more designed for training machine learning model in a fast manner. So, these to have their own advantages both can do both, but they have their distinct advantages. And, also we have seen that the underlying philosophy of both Spark and machine learning is to actually have computation graphs. They have different convocations on the computation graph, they have different constraints.

So, for example, in the Spark the computation graph is a directed acyclic graph, but in TensorFlow that is not the case. So, these are the two main differences between Spark and TensorFlow. So, in for example, in Spark you have a DAG as a computation graph. So, you have a Directed Acyclic Graph whereas, whereas in case of TensorFlow you need not your graph can have cycles. And, the second differences is difference is that in Spark the RDD's are immutable whereas, in case of TensorFlow you can see that you can have variables. So, this these variables are actually mutable quantities. So, which

basically means that TensorFlow cannot support something like fault tolerance and things like that, but on the other hand it is more suited for machine learning computation.

(Refer Slide Time: 42:35)



References:

- A Comparison of Distributed Machine Learning Platforms. Kuo Zhang Salem Alqahtani Murat Demirbas. 2017 26th International Conference on Computer Communication and Networks (ICCCN)
- Introduction to Tensor flow.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSE | Sourangshu Bhattacharya | e and

So, these are the references, the comparison is taken from this particular paper and some of the things are taken from Introduction to Tensor flow.