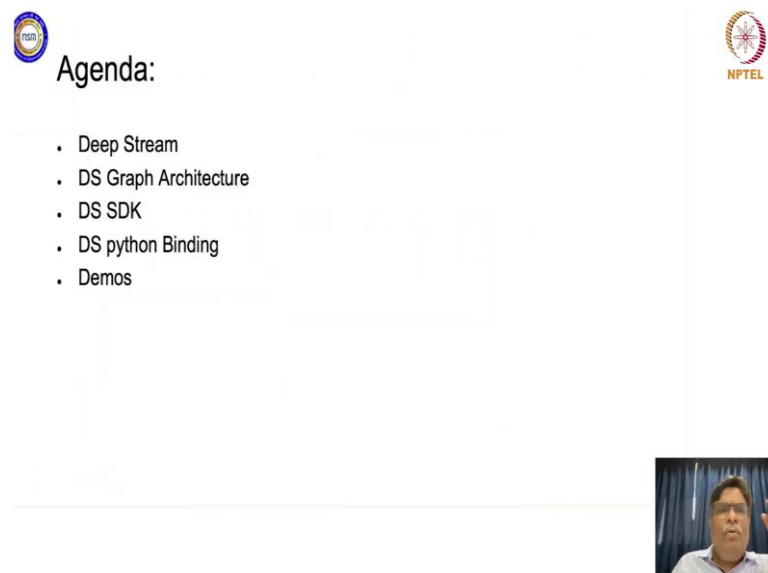


Applied Accelerated Artificial Intelligence
Prof. Satyadhyan Chickerur
Department of Computer Science and Engineering
Indian Institute of Technology, Palakkad

Lecture - 56
Applied AI: Smart City (Intelligent Video Analytics)
Session 2: DeepStream - Part 1

Yeah. So, today let us start the second session which specifically is on something which is called as DeepStream and this basically will be related to the previous session, which was related to Intelligent Video Analytics.

(Refer Slide Time: 00:36)



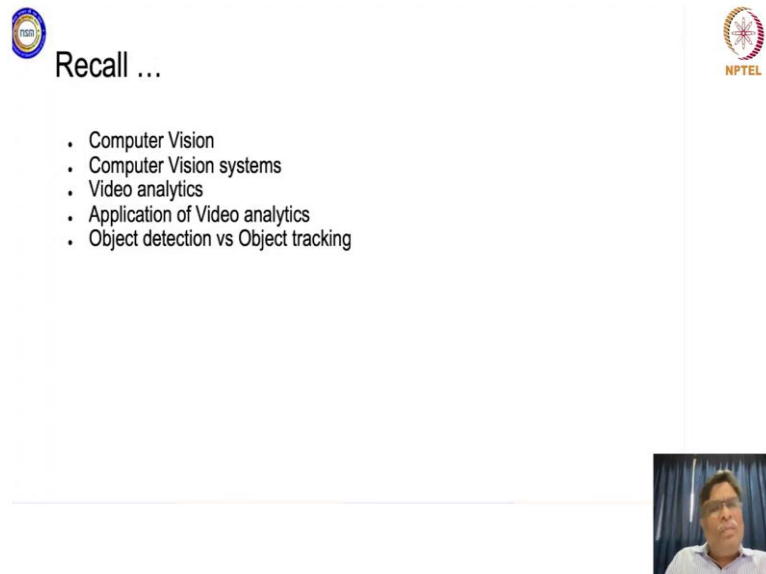
Agenda:

- Deep Stream
- DS Graph Architecture
- DS SDK
- DS python Binding
- Demos

So, let us try to see the agenda for the day, we will cover what DeepStream is, what is a DeepStream graph architecture, how will you use a DeepStream SDK. And since you are using Python, how is that the DeepStream Python binding is generically used and we will have our 3 to 4 demos today.

So, 2 demos we will be showing you on Dockers which basically will use DeepStream and run on a laptop cum EGX type of a system or anything. And then two demos which will show specifically on the inferencing portion, which runs on the edge device like a Jetson TX2 or Jetson Nano or whatever, right. So, this is the overall agenda.

(Refer Slide Time: 01:21)



The slide features two logos: a circular logo on the top left and the NPTEL logo on the top right. The text 'Recall ...' is positioned below the left logo. A bulleted list of topics is centered on the slide. A small video inset of a speaker is located in the bottom right corner of the slide area.

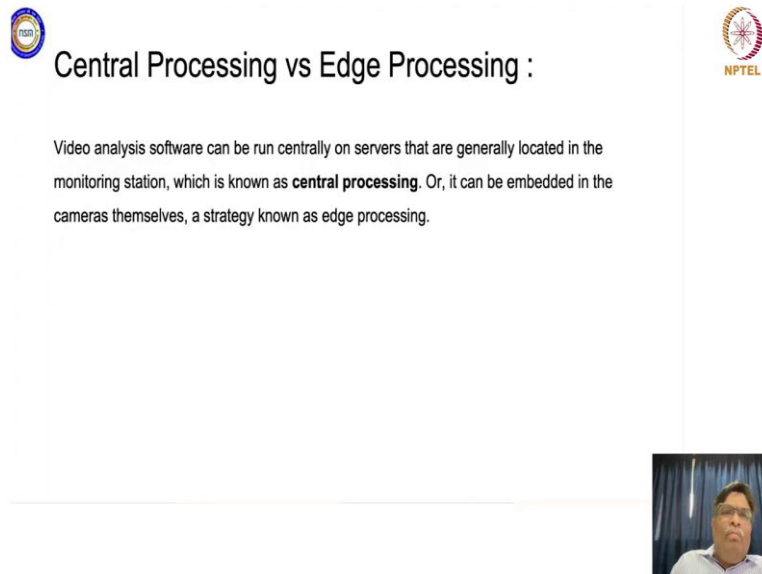
Recall ...

- Computer Vision
- Computer Vision systems
- Video analytics
- Application of Video analytics
- Object detection vs Object tracking

Now, if you recall, we actually in the previous session started with what a computer vision is, what are computer vision systems and we showed you some examples of video analytics. And we also understood the difference between what object detection does as against what object tracking is.


And we showed you the sample for a parking thing and today most probably we will show you the parking thing also which basically would involve GPUs and which will be more real time right, in processing as compared to the previous example which we showed in the previous session.

(Refer Slide Time: 02:06)



Central Processing vs Edge Processing :

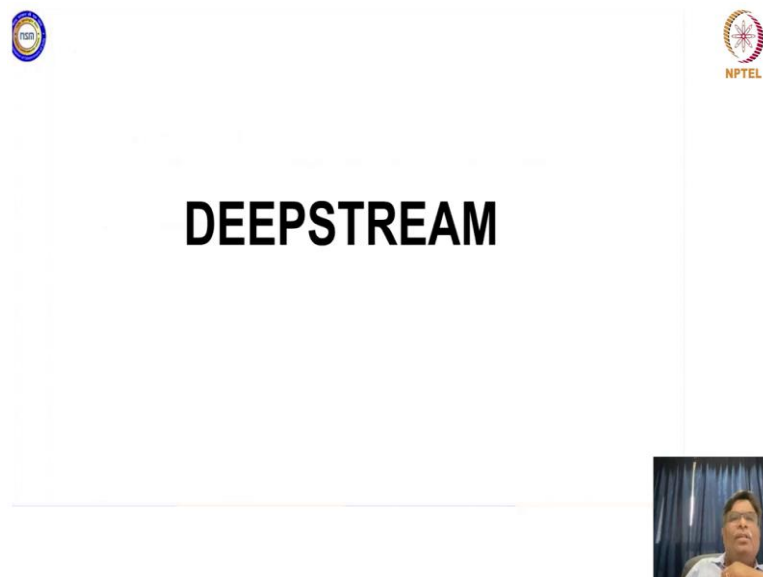
Video analysis software can be run centrally on servers that are generally located in the monitoring station, which is known as **central processing**. Or, it can be embedded in the cameras themselves, a strategy known as edge processing.



So, now we all know the difference between the central processing, versus the edge processing. And if we are trying to do it on the edge device itself, ok. So, for example, when you talk of video analysis software, it can be run centrally on the server that are located actually in the monitoring stations or data centers and this is called as central processing.

Or that particular software could be embedded in the camera itself right and then camera would be there on the edge device and you can call it as edge processing, right. So, this is the basic distinction between when you try to develop a central processing type of a application versus a edge processing type of a application.

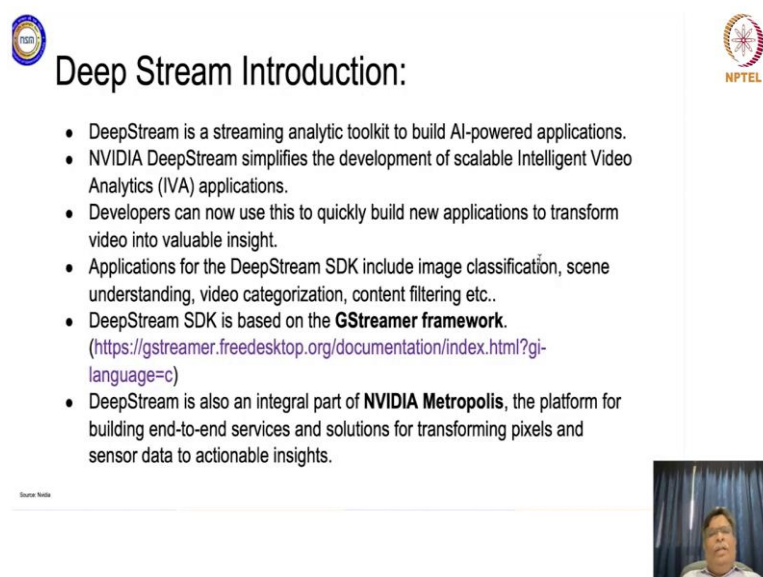
(Refer Slide Time: 02:57)



The slide features the IIT Bombay logo in the top left and the NPTEL logo in the top right. The word "DEEPSTREAM" is centered in a large, bold, black font. A small video inset in the bottom right corner shows a man speaking.

Now, let us try to go into a bit of a detail on what a DeepStream is.

(Refer Slide Time: 03:03)



The slide features the IIT Bombay logo in the top left and the NPTEL logo in the top right. The title "Deep Stream Introduction:" is followed by a bulleted list of points. A small video inset in the bottom right corner shows a man speaking.

- DeepStream is a streaming analytic toolkit to build AI-powered applications.
- NVIDIA DeepStream simplifies the development of scalable Intelligent Video Analytics (IVA) applications.
- Developers can now use this to quickly build new applications to transform video into valuable insight.
- Applications for the DeepStream SDK include image classification, scene understanding, video categorization, content filtering etc..
- DeepStream SDK is based on the **GStreamer framework**. (<https://gstreamer.freedesktop.org/documentation/index.html?glanguage=c>)
- DeepStream is also an integral part of **NVIDIA Metropolis**, the platform for building end-to-end services and solutions for transforming pixels and sensor data to actionable insights.

And if you see DeepStream, DeepStream is the open source toolkit which NVIDIA has designed and which is a streaming analytical tool kit to build AI powered application right. And what does it simplify actually? So, NVIDIA DeepStream simplifies the whole development process for developing a scalable intelligent video application.

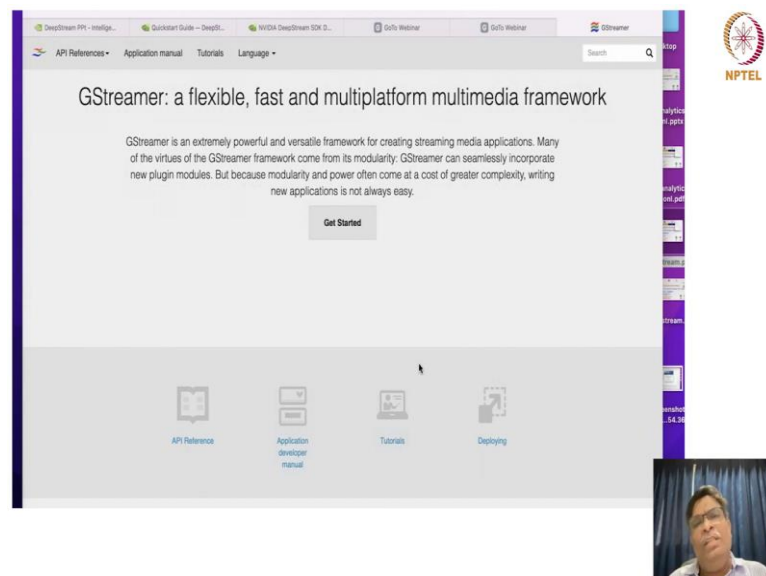
So, any intelligent video analytics application ok, development is simplified when you start using DeepStream. And why do developers actually need to use DeepStream type of

applications right? So, you can basically do this to quickly build new applications ok, and transform your video which will give you some good insights very fast right.

Because that particular pipeline developed is such a nice type of pipeline right, when we discuss this it is very very scalable and it is very easy for application development to happen right. And the SDK which is now there ok, you can do applications which will basically involve image classification, scene understanding, you talk of video categorization content filtering. So, many of them right.

And this DeepStream SDK is based on the GS or GStreamer framework, right. So, if you go to this is a very very important ok.

(Refer Slide Time: 04:58)



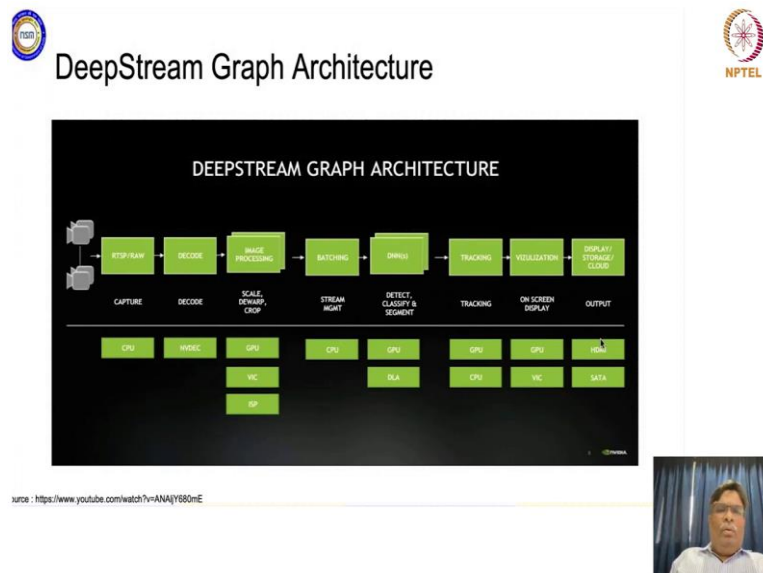
Flexible fast multi platform multimedia framework which is called as GStreamer ok and this is a versatile framework for creating streaming media applications ok. And one of the things which if people would have been working on multimedia application development, they would understand that one of the important things of this GStreamer is that it is very very modular, right.

And you can plug in new modules as and when required and this modularity is a very very powerful thing about this GStreamer, which will help you to actually develop applications which can be very easily deployed, right. So, this is what is our interest and

this particular DeepStream SDK ok, is basically developed using this GStreamer framework right.

And if you see also that there is something which is called as NVIDIA Metropolis and this Metropolis is a platform which gives you end-to-end services and solutions for transforming pixel and sensor data to actionable insights right. We are not going into metropolis for the time being, but DeepStream is a central portion for NVIDIA Metropolis which is very very good for developing applications which are related to smart city, right. So, this is how the whole DeepStream overall right, looks like right.

(Refer Slide Time: 06:37)



Now, there is a specific pipeline, which we call it as a DeepStream graph architecture ok. If you understand this, at a level where in what does this architecture or a framework help you to do.

So, it helps you to do end-to-end development ok, of any streaming application which you are trying to develop. So, it starts with something called as capture and it ends with something wherein you will get some output, right. So, this is the total framework from starting to the end and these are the various hardwares on which each of these stages could be run.

So, let us go into a brief of one liners of what all of these modules do during your whole pipeline right. So, capture the first step is to capture your data from multiple cameras, it

can be through real time streaming protocols or it can be a raw data or whatever. This particular module captures the information; it can be done on a CPU.



Now, then there is something which you need to decode right, there is some portion which you need to decode. So, decoding is done using NVDEC and this basically is a module which does some decoding, there are details available. We will share the link with you if you want to go into the details of each modules. But for the time being in interest of time, we are just telling you certain things, which will help you to understand and appreciate the context under which we are trying to develop in some applications, right.

So, now this image processing portion, wherein you can do scaling you can do dewarping and you can do cropping ok, of some portions of your image or the frames or whatever you call can be done on GPUs or ISPs or some VICs ok. Now, this is Video Intelligent Codecs, ISPs and GPUs right.

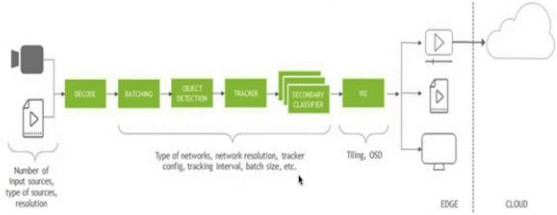
And then you can do some stream management, batching type of applications, this module is done on a CPU, then this is deep learning accelerator, this is a GPU which will help you to run DNNs, it will help you to detect classify and segment anything. Then you can do tracking, tracking can be done on a GPU or CPU. Then again visualization, it can be an on screen display again you can do GPU, you can use it for rendering, you can use your VICs for it.

And ultimately in the end you can either do a display or you can store it or you can basically put it into cloud and this particular output right can be viewed using a HDMI or SATA type of stuff or whatever. So, the basic idea is this whole of this particular architecture ok, would be useful when you basically would like to develop such an application; so yeah.

(Refer Slide Time: 10:13)

 DeepStream Reference Application - deepstream-app 

- The DeepStream reference application is a **GStreamer based solution and consists of set of GStreamer plugins encapsulating low-level APIs to form a complete graph.**
- App is fully configurable and also holds pre-built inference plugin.
- Good reference application to start learning the capabilities of DeepStream.



The diagram illustrates the DeepStream Reference Application architecture. It shows a pipeline starting with input sources (camera and video file) feeding into a 'RECEIVE' plugin. This is followed by 'BATCHING', 'OBJECT DETECTION', 'TRACKER', and 'SECURITY CLASSIFIER' plugins, which are collectively labeled as 'Type of networks, network resolution, tracker config, tracking interval, batch size, etc.'. The pipeline then goes through 'TILING, OSD' and 'VE' (Video Encoder) plugins. The output is split into two paths: one for 'EDGE' devices (displaying video and OSD) and another for 'CLOUD' storage. A source URL is provided at the bottom: https://docs.nvidia.com/metropolis/deepstream/dev-guide/text/DS_Overview.html. A small video inset of the presenter is visible in the bottom right corner.

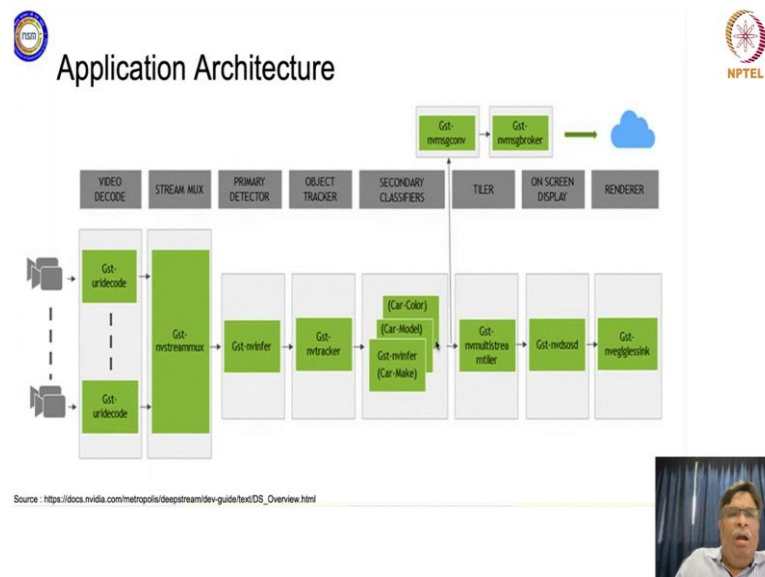
So, if you want to actually develop a DeepStream app, DeepStream reference application. So, the reference application as I told you is the GStreamer application, you would be using the GStreamer framework as I showed you; excuse me. So, this GStreamer based solution will consist of a set of GStreamer plugins, which are basically considered to be low level APIs and which will give you a complete graph or a complete end to end pipeline, right. So, this app is fully configurable and you can actually hold a pre-built inference plugin also.

So, as it is very very modular in nature, you can go on adding various plugins ok and the capabilities of this DeepStream ok, would be basically very very useful when you try to develop in some applications as we will see in some of the examples. Now, here the first thing is to capture.

So, you will have number of input sources, types of sources resolution, it can be video, it can be real live input from the camera then you do the decoding, there will be batching, then you have this object detection. Then you can track then there is a classifier which will help you to basically do many things and then you can do tiling and on screen display visualization and then it can be either done on the edge.

We will show you this, as well as on premise through the cloud or on the data center or whatever. We will show you both some examples of this and then this is what basically a DeepStream application is going to help you to attain, ok.

(Refer Slide Time: 12:15)



So now, that whole application which we talked about, which are the various steps ok they may appear, this may appear the slides may appear to be repetitive, but it is going to give you a fair understanding of how basically or what total thing is involved when you want to develop an application using DeepStream right.

So, as I told you right this is Gstreams uridecode, Gstreams uridecode. So, this basically is a video decode step after you get the whole input from the camera, either its live or using RAW or RTSP or whatever then there is a mux. So, this uses something called as nvstreammux it is Gst-nvstreammux. So, the Gst or the GStreamers framework then they have added, the NVIDIA has added nvstreammux for it then it becomes a stream mux.

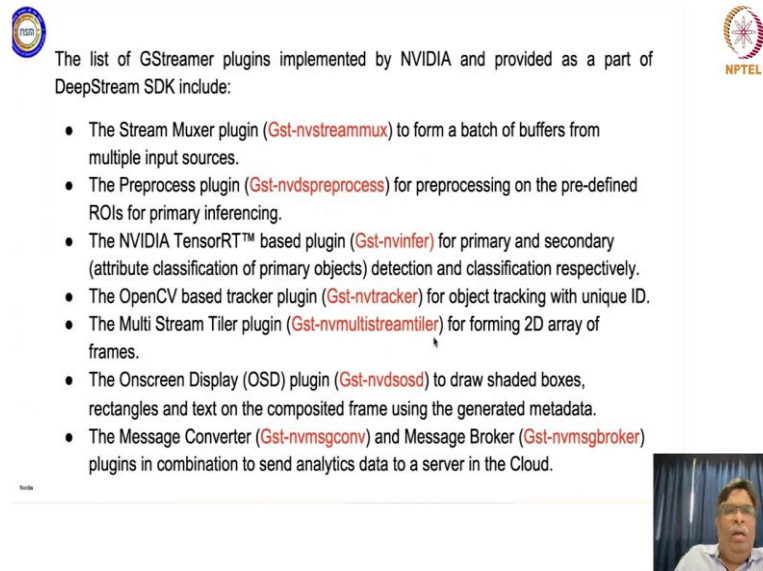
And then there is a primary detector which is basically a plugin, which is basically a inferencing module which involves something like TensorRT thing and then this is a tracker and then there are secondary classifiers. Like for example, the make of the car or the model of the car or the colour of the car.

And then there is a tiler which basically gives you as to how do you get the output displayed and then, this is something like on screen display then this is something which is called as a renderer, right.

So, if you see that way, the steps are almost the same, but we are trying to link each of these steps with certain modules, which are actually implemented using the GStream

framework, which is effectively implemented or developed using plugins which NVIDIA has given us as DeepStream and which efficiently runs on inferencing as well as on the GPUs, right. So, this is the overall application architecture.

(Refer Slide Time: 14:47)



The list of GStreamer plugins implemented by NVIDIA and provided as a part of DeepStream SDK include:

- The Stream Muxer plugin (**Gst-nvstreammux**) to form a batch of buffers from multiple input sources.
- The Preprocess plugin (**Gst-nvdspreprocess**) for preprocessing on the pre-defined ROIs for primary inferencing.
- The NVIDIA TensorRT™ based plugin (**Gst-nvinfer**) for primary and secondary (attribute classification of primary objects) detection and classification respectively.
- The OpenCV based tracker plugin (**Gst-nvtracker**) for object tracking with unique ID.
- The Multi Stream Tiler plugin (**Gst-nvmultistreamtiler**) for forming 2D array of frames.
- The Onscreen Display (OSD) plugin (**Gst-nvdsosd**) to draw shaded boxes, rectangles and text on the composited frame using the generated metadata.
- The Message Converter (**Gst-nvmsgconv**) and Message Broker (**Gst-nvmsgbroker**) plugins in combination to send analytics data to a server in the Cloud.

So, to go into the details of what each of these plugins provide to us, these plugins actually are implemented by NVIDIA and they are a part of DeepStream SDK. Now, for example, this Stream Muxer Plugin which is called as `Gst-nvstreammux` actually forms a batch of buffers from multiple input sources. So, what does this plugin do effectively? You are going to get a lot of buffers ok, based on how many multiple input sources you get ok. And for each of these input source this `Gst-nvstreammax` is going to create a buffer for you, right.

And then the pre process plugin or `Gst-nvdspreprocess` is for pre processing on the predefined region of interest for primary inferencing. So, the idea is you will have a pre processor plugin, which will help you to pre process the data based on the region of interest. As in the previous example, previous session we saw that we wanted to actually have the image of the center of the screen, right.

So, we cropped it from the center, so that was a region of interest. So, region of interest can change, right. So, the region of interest has to be selected. So, we can use this pre-processed plugin for the scene. Then there is something which is called as `Gst-nvinfer`

which I told that it is related to TensorRT. So, this is NVIDIA TensorRT based plugin, which will help you to do some primary detection and classification, right.

So, you can actually use this plugin for detecting and doing some classification. Now, then there is something which is called as Gstream-nvtracker. So, it is a plugin again which will help you to work with OpenCV ok and which helps you to do object tracking with a unique ID. So, you are using OpenCV for tracking and then you are generating a unique ID. Then this is this Multi Stream Tiler Plugin which basically does or gives you a 2D array of frames, right.

So, this is a tiler plug in and you can get multiple stream tilers. So, you get these tiles right and then you will get a 2D array of frames as each of these tiles. And then this on screen display plugin basically is Gst-nvdsosd is to draw shaded boxes rectangles and text on the composite frame using the generated metadata.

So, this basically means that your onscreen display is going to actually draw certain boxes, rectangles and you can display text also on this composited frame right which basically will help you to generate certain metadata if required or something of that sort.

Now, this message converter and message broker is a plugin which will help you to actually send in analytics data to a server in the cloud or this is a broker type of communication, which you, do ok. So, message broker type of communication and then there is a converter; obviously, needed excuse me, ok.

(Refer Slide Time: 19:02)

The slide features the NVIDIA logo on the top left and the NPTEL logo on the top right. The main title is "DeepStream SDK". A bullet point states: "NVIDIA's DeepStream SDK delivers a complete streaming analytics toolkit for AI-based multi-sensor processing, video, audio and image understanding." Below this, five key features are listed in a grid:

- Powerful & Flexible SDK**: A unified SDK suitable for a multitude of use-cases across a broad set of industries.
- Low-Code Programming**: Create powerful Vision AI applications using Graph Composer's simple and intuitive UI.
- Real-time Insights**: Understand rich and multimodal sensor data at the edge.
- Managed AI services**: Deploy AI services in cloud native containers and orchestrate using Kubernetes.
- Reduced TCO**: Train with TAO Toolkit and use DeepStream to increase stream density.

A small video inset in the bottom right corner shows a man with glasses speaking.

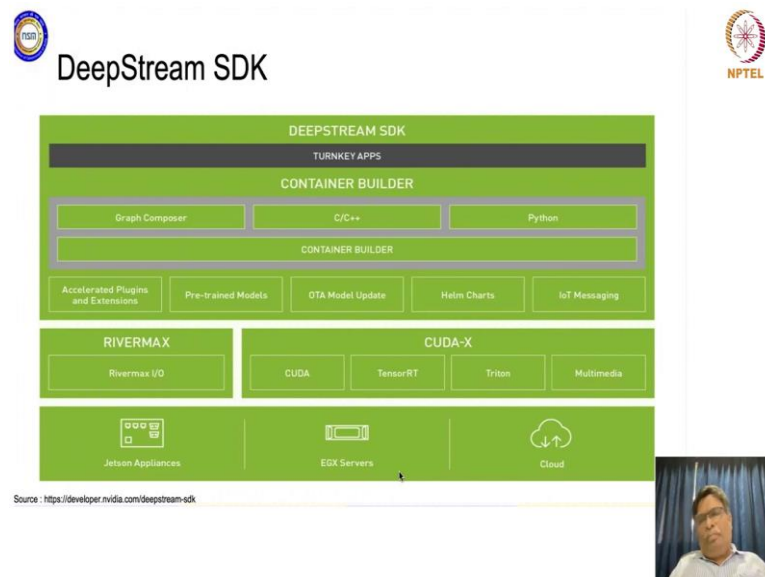
So, if you see what actually a DeepStream SDK gives you. So, it helps us to actually do a complete streaming of some AI based videos along with some analytics, right.

So, it is actually helping you to do streaming analytics for AI based sensor processing right. It can be data from the video sensor, audio sensor or image understanding whatever right. It is very powerful and flexible, it uses low code programming in the sense you have this graph composer ok, which is very very simple and intuitive UI.

We are not going to actually go into the details of graph composer today, but we will surely give you links you can just see them, but we are going into a bit of technical demos today, which will basically help you to appreciate the power of DeepStream SDK for inferencing as well as known on any of these GPUs, right.

So, it gives you real time insights, multi modal sensor data and you can develop managed AI services, you can deploy AI services on the cloud using containers and you can orchestrate using Kubernetes ok. And then you can train with the TAO Toolkit which uses DeepStream to increase stream density, right. So, this is the overall idea of why you should be going in for DeepStream applications, if you are doing video analytics. There is no other better framework ok, if you are working on video analytics or video processing for that matter ok.

(Refer Slide Time: 20:55)

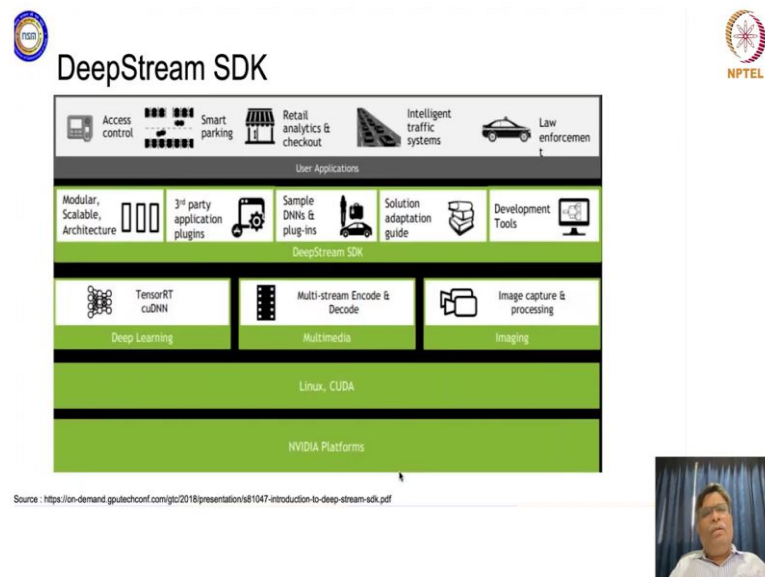


So, this is how basically it is going to pan out your total DeepStream SDK, will have turnkey apps you can develop in your application you will have container builder. And then this container builder basically you can use right with a graph composer or you can develop your applications using C, C++ or Python ok so that the whole container could be developed.

And then you can actually link that container of yours for developing certain accelerated plugins and extensions. You can use that with it, you can use that container with a pre trained model, you can use that with one time model update, you can use it for IoT messaging, you can use Helm charts and then all of this can run ok on CUDA or TensorRT or Triton or Multimedia and Rivermax I/O.

So, this is related to IO and then all of this could be either run on EGX servers Jetson for inferencing or on cloud. So, this is how the whole DeepStream SDK looks like, ok.

(Refer Slide Time: 22:19)




So, for example, some of the applications wherein DeepStream SDKs would be used all are being used ok, is like access control, smart parking we showed you an example of smart parking the other day. Today also we will show you something of smart parking ok.

And then these are used, these are used for retail analytics and checkout. You can use it for intelligent traffic systems, you can use it for law enforcement. So, modular scalable architecture we are talking of, we are talking of application integration, you can work with various DNNs, you can work with various developmental tools right. And then you can work with Multimedia, Imaging as well as Deep Learning.

So, you can work with TensorRT, cuDNN, you have facilities to work on Linux, CUDA and then each of this can run on any of these NVIDIA platforms, right. So, this is how it is going to happen.

(Refer Slide Time: 23:23)



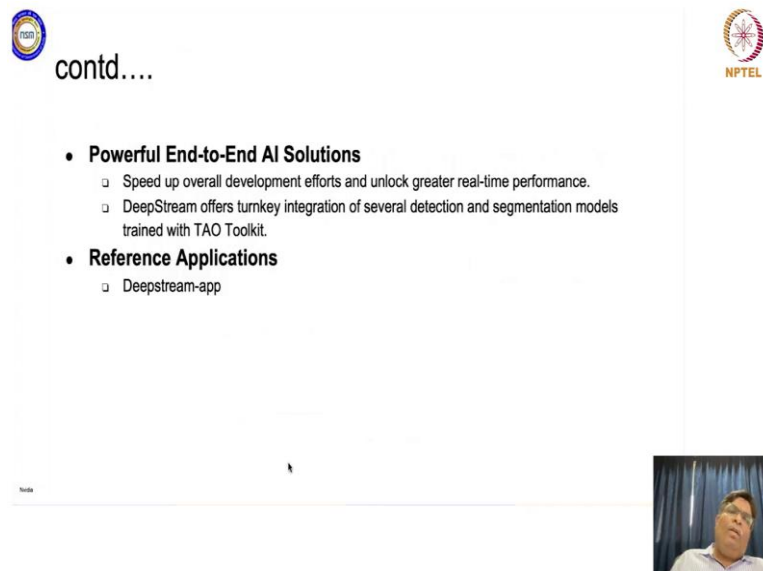
Why use DeepStream SDK?

- **Seamless Development**
 - ❑ Developers can build seamless streaming pipelines for AI-based video, audio, and image analytics using DeepStream.
 - ❑ Offers extensive AI model support.
 - ❑ Offers the flexibility from rapid prototyping to full production level solutions.
- **Low-Code Programming with Graph Composer**
 - ❑ Developers can quickly construct complex pipelines and applications using graphical programming.
- **Securely Manage Apps & Service**
 - ❑ DeepStream SDK can run in any cloud and at the edge.
 - ❑ Bi-directional messaging between edge and cloud.
 - ❑ Smart record feature.
 - ❑ Seamless Over-the-Air (OTA) update for the entire app or individual AI models.
 - ❑ Secure IoT device communication




So, why you should use DeepStream SDK I have told you various reasons of trying to use it. So, these slides will be shared, you can have a look at these as well.

(Refer Slide Time: 23:36)



contd....

- **Powerful End-to-End AI Solutions**
 - ❑ Speed up overall development efforts and unlock greater real-time performance.
 - ❑ DeepStream offers turnkey integration of several detection and segmentation models trained with TAO Toolkit.
- **Reference Applications**
 - ❑ Deepstream-app



So, one of the things is it is a powerful end-to-end AI solution development toolkit, that is very very important, right.

you have DeepStream Python app. So, you can do this and then you can do certain bindings and wrappers and then you can use it, you can use C, C++ as well as Python both for it.


(Refer Slide Time: 24:41)



Platform and OS Compatibility

Jetson model Platform and OS Compatibility

D5 release	D5 3.0	D5 4.0.2 (Unified)	D5 5.0 GA, 5.0.1, 5.1 (Unified)	D5 6.0	D5 6.0.1
Jetson platforms	AGX Xavier	Nano, AGX Xavier, TX2, TX1	Nano, AGX Xavier, TX2, TX1, Jetson NX	Nano, AGX Xavier, TX2, TX1, Jetson NX	Nano, AGX Xavier, TX2, TX1, Jetson NX
OS	L4T Ubuntu 18.04/16.04	L4T Ubuntu 18.04	L4T Ubuntu 18.04	L4T Ubuntu 18.04	L4T Ubuntu 18.04
JetPack release	4.1.1	4.3	4.4 GA (4.5.1 GA for D5 5.1)	4.6 GA	4.6.1 GA
L4T release	31.1	32.3.1	32.4.3 (32.5.1 for D5 5.1)	32.6.1	32.7.1
CUDA release	CUDA 10.0	CUDA 10.0	CUDA 10.2	CUDA 10.2	CUDA 10.2
cuDNN release	cuDNN 7.3	cuDNN 7.6.3	cuDNN 8.0.0.x	cuDNN 8.2.1.32	cuDNN 8.2.1.32
TensorRT release	TRT 5.0	TRT 6.0.1	TRT 7.1.3	TRT 8.0.1	TRT 8.2.1
OpenCV release	OpenCV 3.3.1	OpenCV 4.1	OpenCV 4.1.1	OpenCV 4.1.1	OpenCV 4.1.1
VisionWorks	VisionWorks 1.6	VisionWorks 1.6	VisionWorks 1.6	VisionWorks 1.6.502	VisionWorks 1.6.502
GStreamer	GStreamer 1.8.3	GStreamer 1.14.1	GStreamer 1.14.1	GStreamer 1.14.5	GStreamer 1.14.5



And if you see the platform and the OS compatibility right, there are so many Jetson platforms, you are not going into the details of it, but these are various platforms on which inferencing can be done. And majority of this OS is which you work with would be Ubuntu ok, and then you have to use JetPack, you will use Linux for Tegra because we are trying to use with use these inferencing devices which are Jetson kits, right.

So, that is why you will work with CUDA. So, this is one thing which you should understand as to how is the compatibility of which OS, with which Jetson model and which DeepStream release, right.

(Refer Slide Time: 25:30)



Platform and OS Compatibility




DS release	DS 3.0	DS 4.0.2 (Unified)	DS 5.0 GA, 5.0.1 (Unified), 5.1	DS 6.0	DS 6.0.1
GPU platforms	P4, P40, V100, T4	P4, T4, V100	P4, T4, V100, GA100 (DS 5.1)	P4, T4, V100, GA100	P4, T4, V100, GA100
OS	Ubuntu 16.04	Ubuntu 18.04	Ubuntu 18.04 RHEL 8.x	Ubuntu 18.04 RHEL 8.x	Ubuntu 18.04 RHEL 8.x
GCC	GCC 5.4	GCC 7.3.0	GCC 7.3.0	GCC 7.3.0	GCC 7.3.0
CUDA release	CUDA 10.0	CUDA 10.1	CUDA 10.2 (Cuda 11.1 for DS 5.1)	CUDA 11.4.1	CUDA 11.4.1
cuDNN release	cuDNN 7.3	cuDNN 7.6.5+	cuDNN 7.6.5+ (CuDNN 8.0+ for DS 5.1)	cuDNN 8.2+	cuDNN 8.2+
TRT release	TRT 5.0	TRT 6.0.1	TRT 7.0.0 (TRT 7.2.2 for DS 5.1)	TRT 8.0.1	TRT 8.0.1
Display Driver	R410+	R418+	R450.51 (R460.32 for DS 5.1)	R470.63.01	R470.63.01
VideoSDK release	SDK 8.2	SDK 9.0	SDK 9.1	SDK 9.1	SDK 9.1
QSDK release	Not available	1.0.10	1.0.10	2.0.23	2.0.23
GStreamer release	GStreamer 1.8.3	GStreamer 1.14.1	GStreamer 1.14.1	GStreamer 1.14.5	GStreamer 1.14.5
OpenCV release	OpenCV 3.4.x	OpenCV 3.3.1	OpenCV 3.4.0	OpenCV 3.4.0	OpenCV 3.4.0
Docker image	deepstream:3.0	deepstream:4.0.2	deepstream:5.0, deepstream:5.0.1, deepstream:5.1	deepstream:6.0-ga	deepstream:6.0.1




So, the same thing holds good for various GPU platforms. So, this was just Jetson platform, this was GPU platforms. So, we will show you both the examples today.


(Refer Slide Time: 25:42)



Docker Containers Available: For dGPU



Container	Container pull commands
base docker (contains only the runtime libraries and GStreamer plugins. Can be used as a base to build custom dockers for DeepStream applications)	<code>docker pull nvr.io/vidia/deepstream:6.0.1-base</code>
devel docker (contains the entire SDK along with a development environment for building DeepStream applications and graph composer)	<code>docker pull nvr.io/vidia/deepstream:6.0.1-devel</code>
Triton Inference Server docker with Triton Inference Server and dependencies installed along with a development environment for building DeepStream applications	<code>docker pull nvr.io/vidia/deepstream:6.0.1-triton</code>
DeepStream IoT docker with deepstream-test5-app installed and all other reference applications removed	<code>docker pull nvr.io/vidia/deepstream:6.0.1-iot</code>
DeepStream samples docker (contains the runtime libraries, GStreamer plugins, reference applications and sample streams, models and configs)	<code>docker pull nvr.io/vidia/deepstream:6.0.1-samples</code>




And then when you are trying to work with something like discrete GPUs ok. So, there are docker containers available, we will try to work with dockers as well. We will be using the dockers only. So, now, the docker containers there will be base container, there will be development, there will be Triton, there will be IoT and there will be samples, right.


So, each of these containers ok, will have some specific things right. For example, base docker could be used to build custom dockers for DeepStream applications. Development docker will give you the graph composer. Triton Inference Server docker you will get a inference server and certain dependencies which are installed for building deep streaming applications.

Deep stream IoT docker will give you certain specific facilities where in your DeepStream program could be basically linked to some IoT based application. And then there are sample Dockers which will give you right, runtime libraries, samples and various other things. So, this is for discrete GPU right, discrete GPUs ok are GPUs which are specifically not integrated as a part of a processor; these are separate GPUs right; so yeah.


(Refer Slide Time: 27:12)



Docker Containers Available: For Jetson Devices

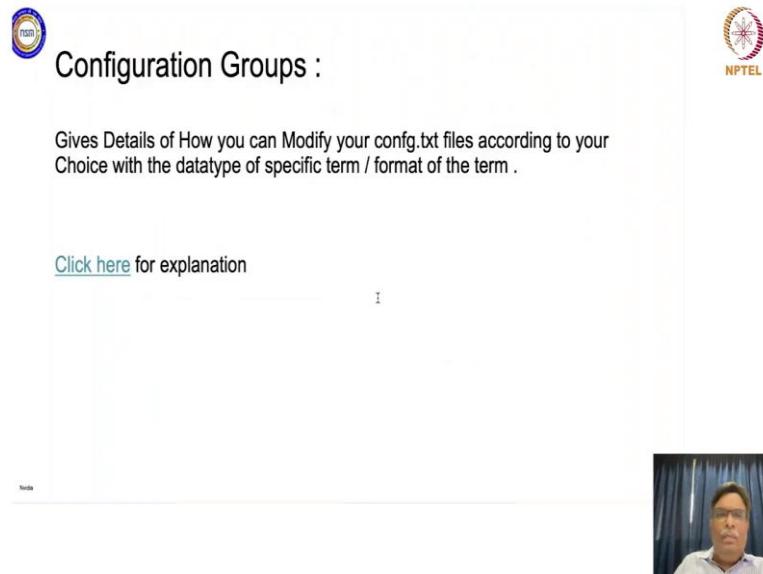


Container	Container pull commands
Base docker (contains only the runtime libraries and GStreamer plugins. Can be used as a base to build custom dockers for DeepStream applications)	<code>docker pull nvr.io/nvidia/deepstream-1rt:8.8.1-base</code>
DeepStream IoT docker with deepstream-test5-app installed and all other reference applications removed.	<code>docker pull nvr.io/nvidia/deepstream-1rt:8.8.1-iot</code>
DeepStream samples docker (contains the runtime libraries, GStreamer plugins, reference applications and sample streams, models and configs)	<code>docker pull nvr.io/nvidia/deepstream-1rt:8.8.1-samples</code>
DeepStream Triton docker (contains contents of the samples docker plus devel libraries and Triton Inference Server backends)	<code>docker pull nvr.io/nvidia/deepstream-1rt:8.8.1-triton</code>



So, then there are docker containers which are available for Jetson devices also. So, the idea is, these are for discrete GPUs, these are for Jetson devices. So, you understand the now difference between these two now. So, these jetson devices require Dockers which are less in size right, which are small in size as compared to docker containers which require to be run for discrete GPUs. You again have a base docker. You have a IoT docker, you have a Triton docker and a sample docker here as well, ok.

(Refer Slide Time: 27:54)



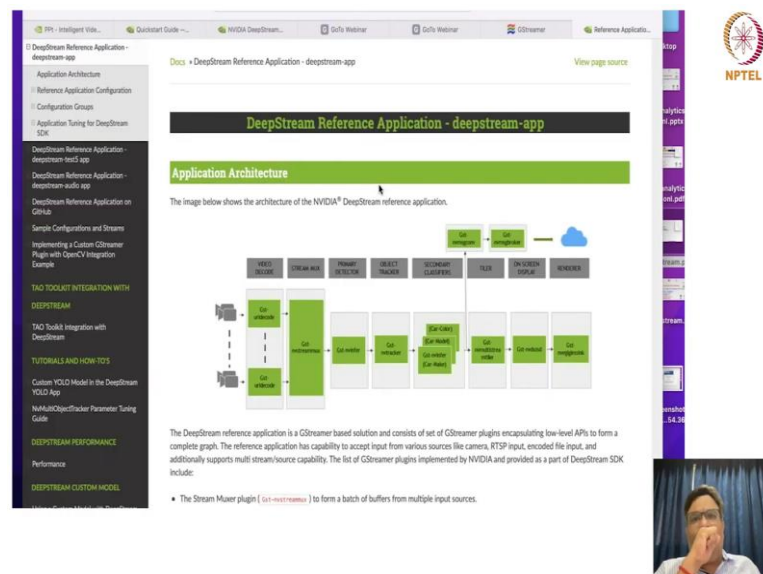
Configuration Groups :

Gives Details of How you can Modify your config.txt files according to your Choice with the datatype of specific term / format of the term .

[Click here](#) for explanation

So, there is one thing which I would like to specifically talk about is these configuration groups ok. Now, each of these applications when you develop, you have got various plugins right and these plugins are readily available to you. The only thing which you need to understand is each of these plugins can be configured based on our needs ok, and then these configuration things are to be maintained as config dot text files and these files need to be used by the DeepStream SDK for taking appropriate inputs for various plugins right.

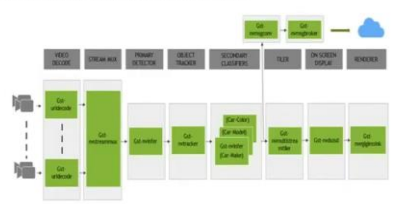
(Refer Slide Time: 28:47)



DeepStream Reference Application - deepstream-app

Application Architecture

The image below shows the architecture of the NVIDIA® DeepStream reference application.

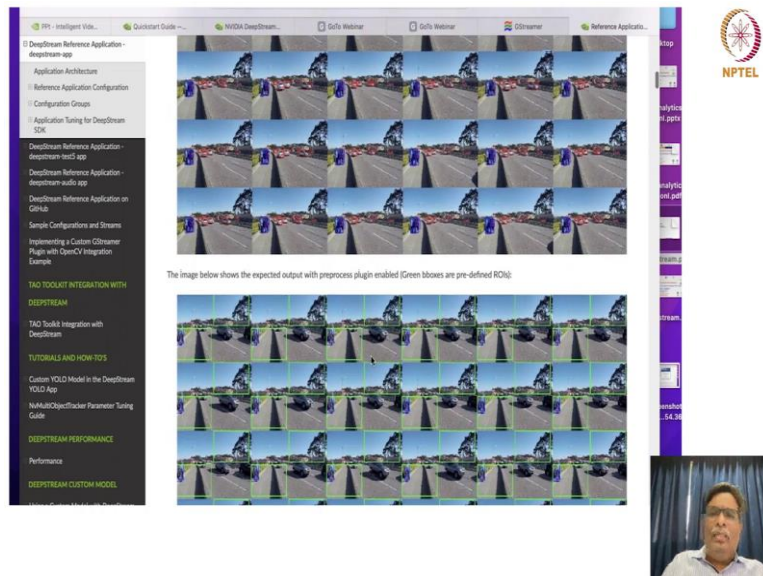


The DeepStream reference application is a GStreamer based solution and consists of set of GStreamer plugins encapsulating low-level APIs to form a complete graph. The reference application has capability to accept input from various sources like camera, RTSP input, encoded file input, and additionally supports multi stream/source capability. The list of GStreamer plugins implemented by NVIDIA and provided as a part of DeepStream SDK include:

- The Stream Muxer plugin (`gst-hls-muxer`) to form a batch of buffers from multiple input sources.

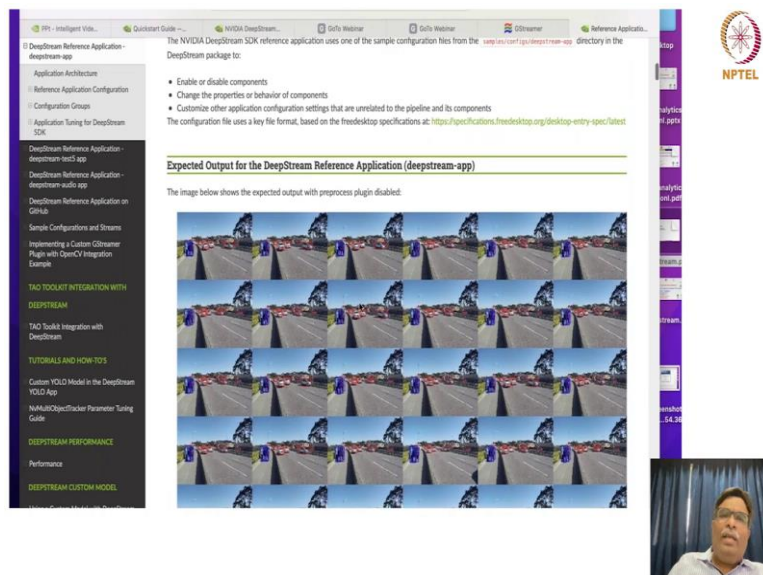
So, there is this document which talks of this reference application.

(Refer Slide Time: 28:54)



And this is how basically you can actually predefine ROI ok pre-processor plugin getting enabled.

(Refer Slide Time: 29:06)



And something like pre processor plugin getting disabled ok.

(Refer Slide Time: 29:11)

The screenshot shows the 'Configuration Groups' page in the NVIDIA DeepStream configuration tool. The page title is 'Configuration Groups' and it states: 'The application configuration is divided into groups of configurations for each component and application-specific component. The configuration groups are:'. Below this, there is a table titled 'Configuration Groups - deepstream app'.

Group	Configuration Group
Application Group	Application configurations that are not related to a specific component.
Tiled-display Group	Tiled display in the application.
Source Group	Source properties. There can be multiple sources. The groups must be named as: [source0], [source1] ...
Streammux Group	Specify properties and modify behavior of the streammux component.
Preprocess Group	Specify properties and modify behavior of the preprocess component.
Primary GIE and Secondary GIE Group	Specify properties and modify behavior of the primary GIE. Specify properties and modify behavior of the secondary GIE. The groups must be named as: [secondary-gie0], [secondary-gie1] ...
Tracker Group	Specify properties and modify behavior of the object tracker.
Message Converter Group	Specify properties and modify behavior of the message converter component.
Message Consumer Group	Specify properties and modify behavior of message consumer components. The pipeline can contain multiple message consumer components. Groups must be named as [message-consumer0], [message-consumer1] ...
OSD Group	Specify properties and modify the on-screen display (OSD) component that overlays text and rectangles on the frame.
Sink Group	Specify properties and modify behavior of sink components that represent outputs such as displays and files for rendering, encoding, and file saving. The pipeline can contain multiple sinks. Groups must be named as: [sink0], [sink1] ...
Tests Group	Diagnostics and debugging. This group is experimental.
NvInAnalytics Group	Specify nvinanalytics plugin configuration file, and to add the plugin in the application.

So many examples are there and these are the various groups right, pre-processing groups, stream mux groups. So, you know that there is a stream mux available.

Now, this stream mux group will specify the properties and behaviour of the stream mux component. So, if you click on this, it will tell you as to what particular type of properties and behaviours right you need to actually give the inputs to.

(Refer Slide Time: 29:34)

The screenshot shows the 'Streammux Group' page in the NVIDIA DeepStream configuration tool. The page title is 'Streammux Group' and it states: 'The [streammux] group specifies and modifies properties of the [nvstreammux] plugin.'. Below this, there is a table titled 'Streammux group'.

Key	Meaning	Type and Value	Example	Platforms
gpu-id	GPU element is to use in case of multiple GPUs.	Integer, +0	gpu-id-1	iGPU, Jetson
live-source	Indicates the muxer that sources are live.	Boolean	live-source=0	iGPU, Jetson
buffer-pool-size	Number of buffers in Muxer output buffer pool.	Integer, +0	buffer-pool-size=4	iGPU, Jetson
batch-size	Muxer batch size.	Integer, +0	batch-size=4	iGPU, Jetson
batched-push-timeout	Timeout in microseconds after to push the batch after the first buffer is available, even if the complete batch is not formed.	Integer, +-1	batched-push-timeout=40000	iGPU, Jetson
width	Muxer output width in pixels.	Integer, +0	width=1280	iGPU, Jetson
height	Muxer output height in pixels.	Integer, +0	height=720	iGPU, Jetson
enable-padding	Indicates whether to maintain source aspect ratio when scaling by adding black bands.	Boolean	enable-padding=0	iGPU, Jetson
mem-alloc	Type of CUDA memory the element is to allocate for output buffers. 0 [buf-mem-default, a platform-specific default. 1 [buf-mem-cuda-pinned] pinned/host CUDA memory.	Integer 0, 1, 2		

So, in this, when you are talking of this nvstreammux plug in, you will have all of these values right you have GPU ID, what is the key right, live source whether what is the

buffer pool size, what is the batch size, what is the width height right. What are the various platforms you can use and how do you give these parameters into the plugin as inputs ok so, all of this.

(Refer Slide Time: 30:09)

Key	Meaning	Type and Value	Example	Platforms
enable	Enables or disables the message converter	Boolean	enable=1	dGPU, Jetson
msg-conv-config	Pathname of the configuration file for the Gie-convmsgconv element.	String	msg-conv-config=dstest5_msgconv_sample_config.txt	dGPU, Jetson
msg-conv-payload-type	Type of payload. 0: PAYLOAD_DEEPSTREAM: Deepstream schema payload. 1: PAYLOAD_DEEPSTREAM_MINIMAL: Deepstream schema payload minimal. 254: PAYLOAD_RESERVED: Reserved type.	Integer 0, 1, 254, or 257	msg-conv-payload-type=0	dGPU, Jetson

(Refer Slide Time: 30:18)

Demo 1 :

Running Deepstream-app from docker container in dGPU system

So, what we are going to do is we can show you some demos now and in those demos we will try to show you the config file as well ok. So, we will try to show you the config file as well and we will start the demos now. So, the first thing is let me start it on this particular 1 minute, let me start this on 1 minute; yes.

(Refer Slide Time: 30:50)

