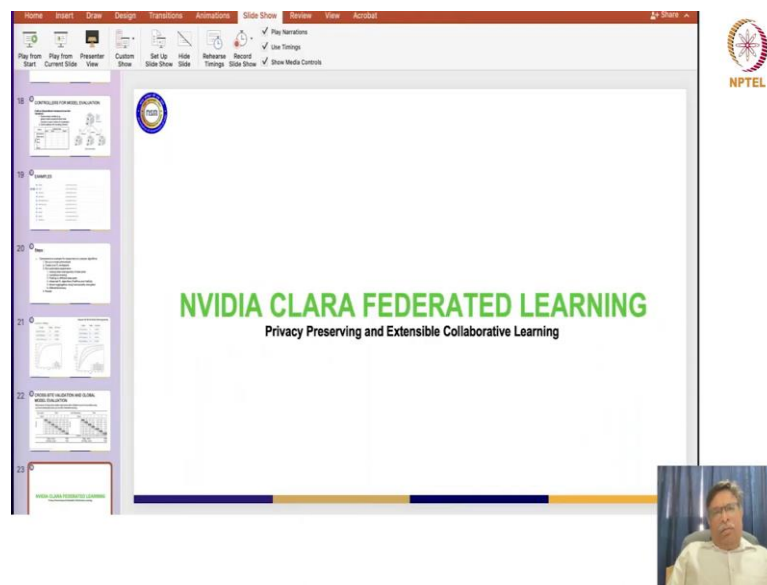


Applied Accelerated Artificial Intelligence
Prof. Satyadhyan Chickerur
School of Computer Science and Engineering
KLE Technological University
Indian Institute of Technology, Palakkad

Lecture - 61
Applied AI: Healthcare (Federated Learning, AI Assisted Annotation)
Session II - Part - 2

(Refer Slide Time: 00:17)



(Refer Slide Time: 00:26)

Introduction

- NVIDIA Clara is a healthcare application framework for AI-powered imaging, genomics, and the development and deployment of smart sensors.

IN CONTEXT TO FEDERATED LEARNING

- NVIDIA's latest release of Clara Train SDK, which features Federated Learning (FL), makes this possible with NVIDIA EGX, the edge AI computing platform.
- NVIDIA Clara's implementation is based on a server-client approach.
- The configurable MMAR (Medical Model ARchive) feature of Clara Train SDK makes it possible for developers to bring their own models and components to perform Federated Learning

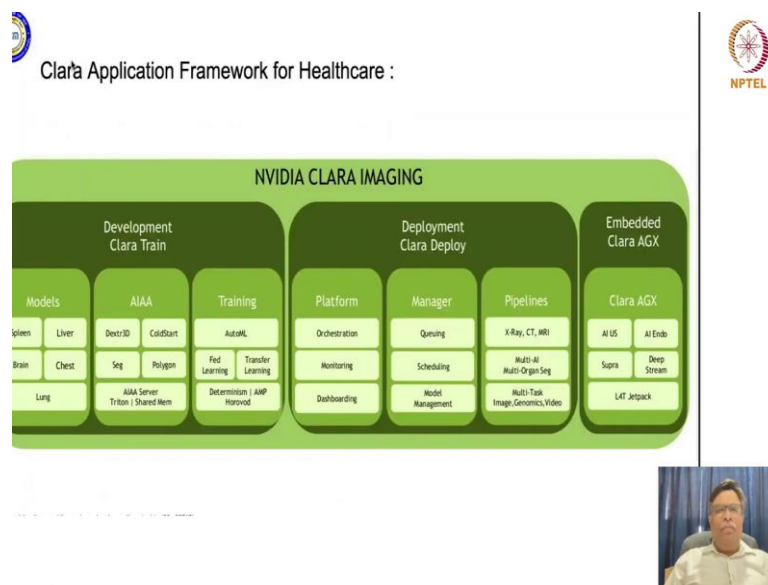
The diagram illustrates the Federated Learning architecture. It shows "Private Data" being processed by "NVIDIA Clara + EGX" to create a "Global Model". This model is then distributed to multiple server-client nodes for further processing.

A small video inset of the presenter is visible in the bottom right corner.

Clara Federated Learning. So now, that was NVFlare right. Now, this we are talking of something which is called as NVIDIA Clara. Now, this is a healthcare application framework which basically is related to any applications which are in the domain of imaging, genomics and which involves smart sensors right. So, in context to the federated learning thing which we are doing, there is the latest release of Clara Train SDK, which basically features federated learning and which helps us to basically run that on the edge AI platform ok.

So, NVIDIA edge GX or edge AI computing ok. Now, this again uses the same server client approach and then there is something which is called as MMAR which is Medical Model Archive which basically is used by this Clara Train SDK wherein you can fetch in your own model to perform federated learning right. So, this is basically a something which basically is the extension of to some extent NVIDIA FLARE which is actually a bigger SDK now which in which will help you to do imaging, genomics, and integrate various sensors also right.

(Refer Slide Time: 02:10)



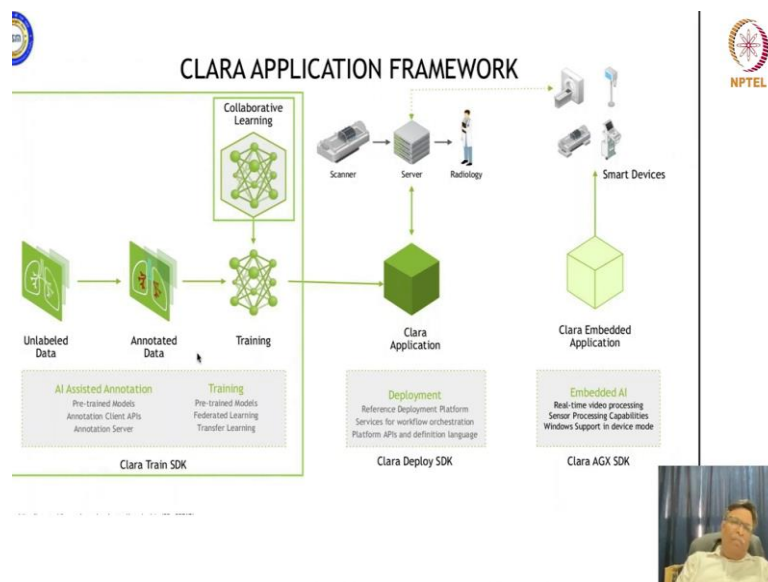
So, if you see this NVIDIA Clara from the aspect of healthcare and specifically imaging right, you can actually divide the whole thing for training or development, for deployment and embedded deployment as well ok. So, deployment embedded deployment and training. So, if you see this, you have got various models available with

you, since you are talking of imaging you have got models for spleen, liver, brain, chest and lungs and then you have AI assisted annotation here.

So, you can do federated learning, but we are talking of AI assisted annotation which we would be showing you in this part of the demo because we already told you of federated learning ok, as a example in NVIDIA FLARE. So, we will be showing you this AI assisted annotation which will have segmentation, which will have a polygon detection and then you have done actually link it with the inferencing Triton server and so on and so forth.

Then from the deployment aspect you have this orchestration monitoring dashboarding so on and so forth and you have got pipelines for X-Ray, CT, MRI then you have multi AI, multi organ segmentation, multi tasking imaging, genomics, video so on and so forth. You have a manager which helps you to do deployment queuing scheduling and so on and then you have this Clara AGX which runs on Linux for Tegra jet pack which deep stream which we showed you AI endo, AI US, supra and all of these are various imaging modalities and packages which you have right.

(Refer Slide Time: 04:20)



So, if you see the Clara application framework, you have got let us say Clara training, Clara deployment and then Clara AGX. So, let us try to understand what happens in training ok, this Clara training SDK is going to actually help you ok to do something called as AI assisted annotation. So, you have this unlabeled data, you have this

annotated data with you and then you have this training done ok and then using this collaborative learning approach you will develop a automated ok AI assisted annotation. So, that basically means you have trained a network which can help you to do some annotation right.

So, that is how this particular training SDK is going to work, the topic of deployment once you develop your application you can develop application and what to say and put it on to a server and then you can get the scans from the scanner, then you basically can get it from anywhere ok and then you can annotate it and your radiologist can actually see this ok. So, that is how it can be deployed.

And when you talk of embedded applications, you can put it on to smart devices and then you will get real time video processing capabilities, you can link with various sensors and then you can work in the device mode. So, that is how it is going to be useful the whole framework.

(Refer Slide Time: 06:20)

The slide is titled "Why use Clara Federated Learning?". It features the NPTEL logo in the top right corner. The content is organized into two main sections: "Let NVIDIA Engineers:" and "Allow Data Scientist to :".

- Let NVIDIA Engineers:**
 - Setup for training across multiple sites.
 - Secure connection, certification SSL.
 - Communication encryption.
 - Resolve deadlocks.
 - Clients join / die /rejoin.
- Out of the box**
 - Avoid data leakage through model parameters.
 - Weighted aggregate relative to data size.
 - Partial model sharing and differential privacy.
 - Provide off diagonal accuracy / metric.
- Allow Data Scientist to :**
 - Focus on the medical problem not coding.
 - BYO aggregation.
 - BYO Privacy logic.
 - Iterate with multiple model Arch, losses, augmentation, etc.
 - Fine tune hyper parameters.

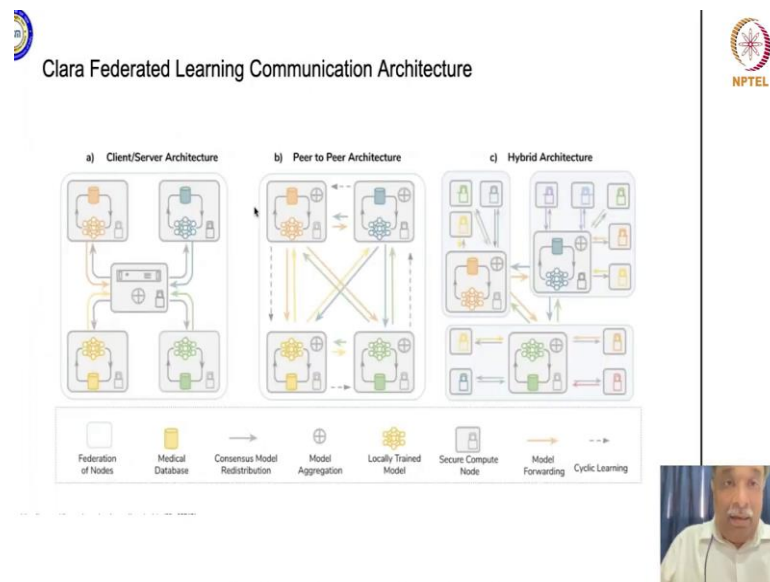
A small video inset in the bottom right corner shows a man speaking.

So, why use Clara? The so, what is going to basically help you in right. So, here NVIDIA will take care of let us say NVIDIA engineers or people who are setting up Clara for you are going to actually set up training across your multiple sites, you have got this certification SSL certification, you can do deadlock resolution and all so on and so forth, you can concentrate on medical problem not specifically coding ok.

And then you have aggregation, you have got privacy logic, you can basically concentrate more on architecture ok and various other tuning parameter so on and so forth. And then you basically can avoid data leakage through model parameters so on and so forth.

So, these are some advantages since this is open source everything is available, but when you talk and try to do it in the production environment right, you basically have to go to the NGC cloud and then there is something called as service provisions ok for this Clara then you will get lot of help as well.

(Refer Slide Time: 07:50)



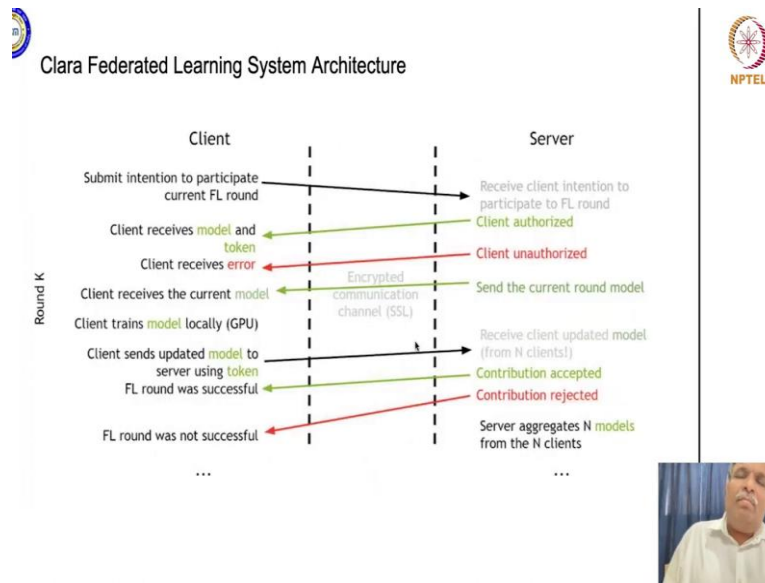
So, let us try to understand that client server architecture, ok then peer to peer architecture, and hybrid architecture I am not going into the details of any of these, but the basic idea is you have got a lot of nodes with you right. You have got medical databases then you have got distribution pathways, then you need a aggregator or aggregation needs to happen, then you have locally trained models ok at all these places, then you should basically be working with the secure compute nodes because we assume that all of these nodes are supposed to be secure.

And then you basically can do model forwarding in a sense that the models can be forwarded as well as the weights or the parameters or anything can be shared among various nodes right. That is how it is going to improve upon your model accuracy as well as so many things, then you have got this type of a cyclic learning wherein you can

basically try to basically cycle your parameters, values or results ok. So, that know everybody uses that for improving their own model so on and so forth.

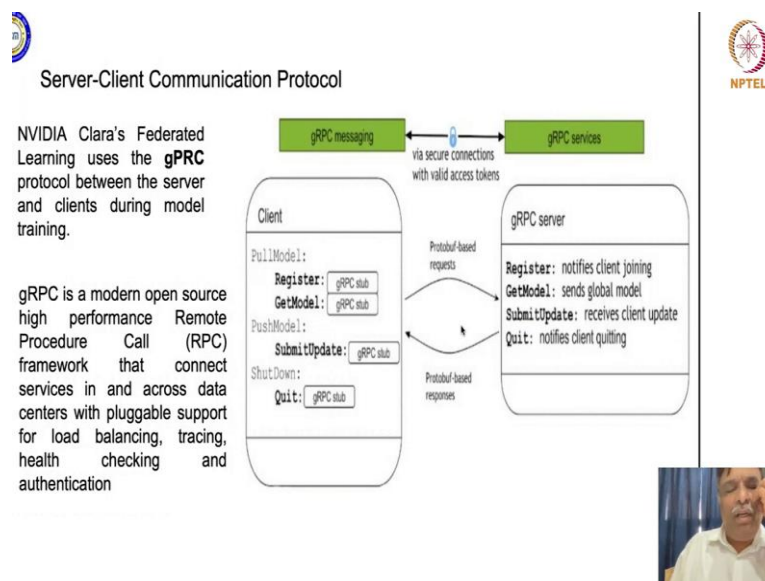
So, we have about three type's client server, peer to peer, and hybrid. This is in a brief thing of what are the various communication architectures available.

(Refer Slide Time: 09:23)



So, this is this client server mechanism, this is a very very easy thing which we know ok. So, I am not going into the details. So, any of these TCPIP communication models; obviously, we will follow the same thing. So, we are not going into the details.

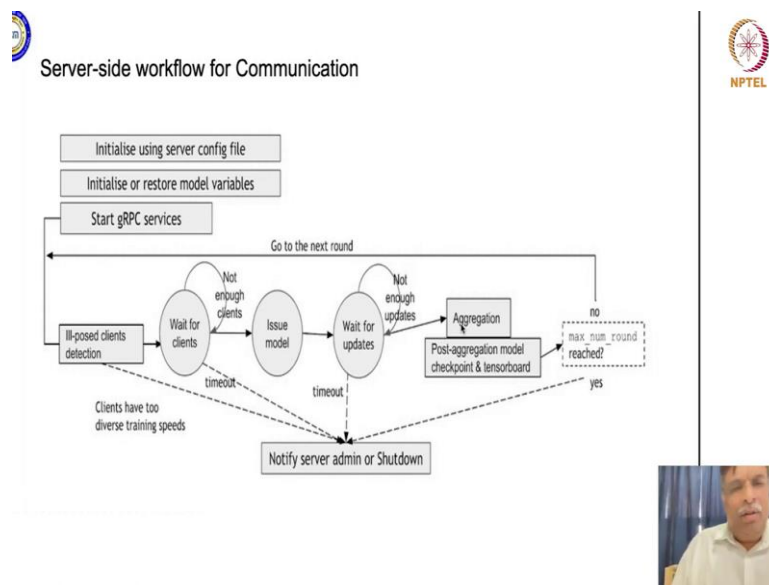
(Refer Slide Time: 09:40)



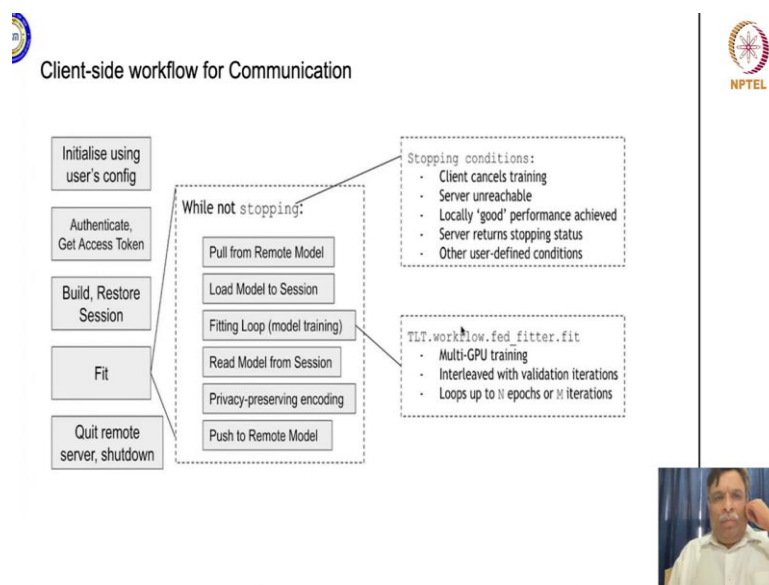
But there here we are using something which is called as open source gRPC ok. So, if you go to the gRPC website it is a open source high performance remote procedure call framework which basically uses secure connections via access tokens right.

So, the messaging as well as the services something of that sort, you are talking of Protobuf based responses.

(Refer Slide Time: 10:13)




(Refer Slide Time: 10:23)



So, this is how the server client communication protocol is there and this I suppose is the server side communication workflow not going into the details, but yeah this is how it is.


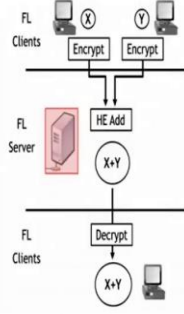
Then client side workflow for communications: again as I told you we will have to initialize and do some user configurations, then generating and accessing the tokens authentication, then how do you fit it remote model what are the stopping conditions, multi GPU thing all of this ok yeah.


(Refer Slide Time: 10:47)



Homomorphic Encryption(HE) in Clara

- NVIDIA Clara Train 4.0 offers homomorphic encryption (HE) tools for federated learning (FL).
- HE enables you to compute data while the data is still encrypted.
- HE is a form of encryption that permits users to perform computations on encrypted data.
- Secure Aggregation with Homomorphic Encryption
 - Protecting gradients/model inversion or attacks untrusted server
- Clients have symmetric key for encryption/decryption
 - Server can only save the encrypted model
 - Secret keys for decryption are owned by clients



So, homomorphic encryption you talk about. So, this gives you homomorphic encryption during federated learning ok. So, basically homomorphic encryption means that you can compute the data while the data is still encrypted right. So, basically it is something like doing computations on encrypted data or using encrypted data right, and how do you aggregate the data for your own use right and then all of these encryption and decryption other methods which you conventionally would be using would also be being used.

(Refer Slide Time: 11:34)



Federated Learning server configuration file

The Federated Learning server configuration file: *config_fed_server.json*.
Example:

```
{
  "servers": [
    {
      "name": "prostate_segmentation",
      "service": {
        "target": "localhost:8002",
        "options": [
          ["grpc.max_send_message_length", 1000000000],
          ["grpc.max_receive_message_length", 1000000000]
        ]
      },
      "ssl_private_key": "resources/certs/server.key",
      "ssl_cert": "resources/certs/server.crt",
      "ssl_root_cert": "resources/certs/rootCA.pem",
      "min_num_clients": 2,
      "max_num_clients": 100,
      "start_round": 0,
      "num_rounds": 300,
      "exclude_vars": "dummy",
      "num_server_workers": 100
    }
  ]
}
```



So, this is the server configuration file which basically is a json file which we are not going into the details. So, this is basically a server which basically gives the prostate segmentation something of that sort. So, what is the grpc maximums message length and receiving message length. So, what is the SSL private key certificate so on and so forth.

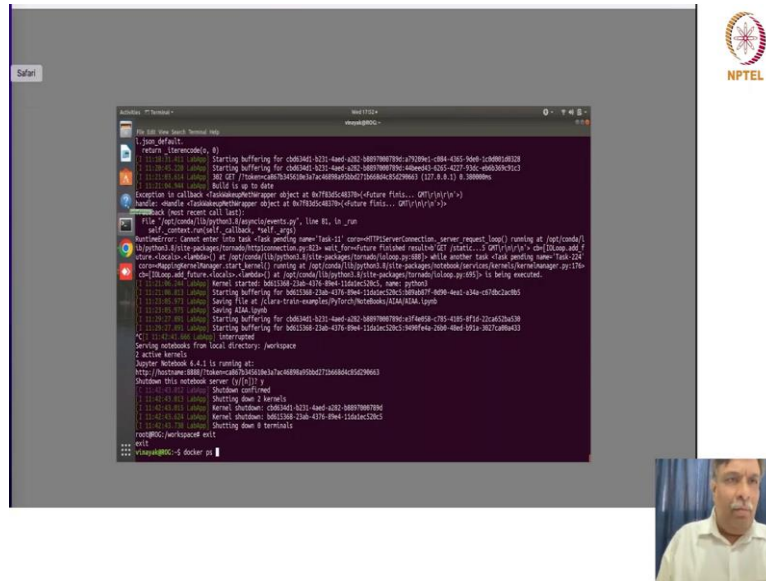
(Refer Slide Time: 11:58)



DEMO



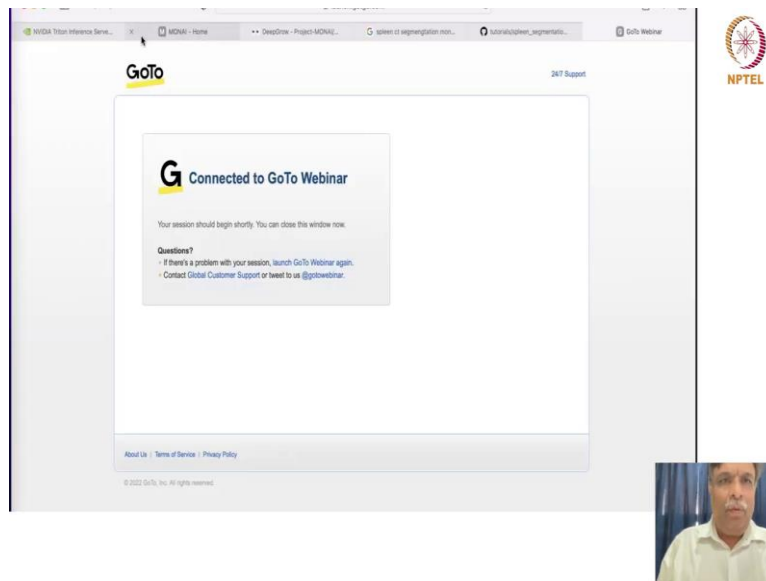
(Refer Slide Time: 12:04)



The terminal window displays the following error message:

```
Exception in callback <FutureWatcher object at 0x7f835c4877b0> (future finished...): OP(1)(1)(1)  
Traceback (most recent call last):  
  File <ipython-input-3-4e91a1e9e2e2.py>, line 81, in <run>  
    self._context.run(self._callback, *self._args)  
RuntimeError: Cannot enter into task <task pending name='Task-11' coro=<HTTPEngineConnection._server_request_loop() running at /opt/conda/...>> while another task <task pending name='Task-12' coro=<HTTPEngineConnection._server_request_loop() running at /opt/conda/...>> will be another task <task pending name='Task-13' coro=<HTTPEngineConnection._server_request_loop() running at /opt/conda/...>> (site-packages/notebook/services/kernel/manager.py:154: ch<ILoop.add_future<lambda> at /opt/conda/...>> (site-packages/tornado/ioloop.py:895)) is being executed.  
  File <ipython-input-3-4e91a1e9e2e2.py>, line 81, in <run>  
    kernel.startname: b6013368-23ab-4376-89e4-1516a1c326c5, name: python  
  File <ipython-input-3-4e91a1e9e2e2.py>, line 81, in <run>  
    Starting buffering for b6013368-23ab-4376-89e4-1516a1c326c5:0b8ab17f-4090-4e42-434a-cf78c2ac8b5  
  File <ipython-input-3-4e91a1e9e2e2.py>, line 81, in <run>  
    Saving file as /data/1440-encrypted/09/09/notebooks/02A4/02A4.ipynb  
  File <ipython-input-3-4e91a1e9e2e2.py>, line 81, in <run>  
    Saving J2M4.ipynb  
  File <ipython-input-3-4e91a1e9e2e2.py>, line 81, in <run>  
    Starting buffering for c06354d1-5231-4e4d-c282-86897887879d:1781-4188-8f4f-32ca453a4538  
  File <ipython-input-3-4e91a1e9e2e2.py>, line 81, in <run>  
    Starting buffering for b6013368-23ab-4376-89e4-1516a1c326c5:949f4e4-2604-46d4-952a-3627ca8a4433  
  File <ipython-input-3-4e91a1e9e2e2.py>, line 81, in <run>  
    I kernel upgrade  
Serving notebooks from local directory: /workspace  
2 active kernels  
Jupyter Notebook 6.4.1 is running at:  
http://hostname:8888/?token=ab7535d3363a7c46889a950d273a684c5c29663  
Shutdown this notebook server (y/[n])?  
y  
  File <ipython-input-3-4e91a1e9e2e2.py>, line 81, in <run>  
    Shut down conf/note  
  File <ipython-input-3-4e91a1e9e2e2.py>, line 81, in <run>  
    Kernel: Shut down: c06354d1-5231-4e4d-c282-86897887879d  
  File <ipython-input-3-4e91a1e9e2e2.py>, line 81, in <run>  
    Kernel: Shut down: b6013368-23ab-4376-89e4-1516a1c326c5  
  File <ipython-input-3-4e91a1e9e2e2.py>, line 81, in <run>  
    Shut down 8 terminals  
root@8888:~/workspace# exit  
... exit  
vishay@8888:~$ docker ps |
```

(Refer Slide Time: 12:09)



The screenshot shows a GoTo Webinar connection page with the following text:

GoTo 24/7 Support

G Connected to GoTo Webinar

Your session should begin shortly. You can close this window now.

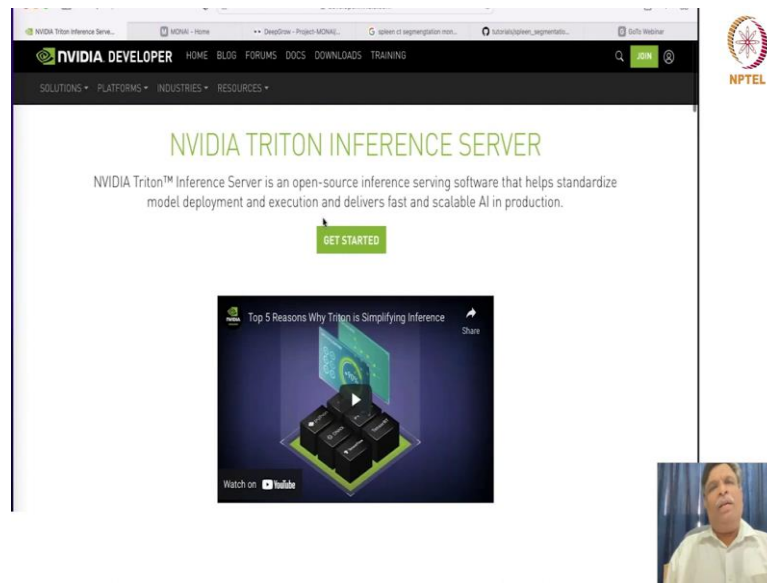
Questions?

- If there's a problem with your session, launch GoTo Webinar again.
- Contact Global Customer Support or tweet to us @gotowebinar.

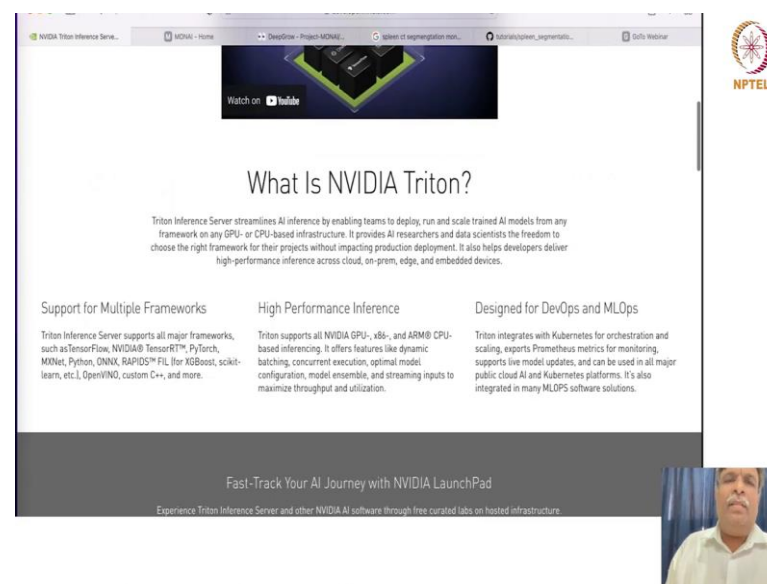
About Us | Terms of Service | Privacy Policy

© 2022 GoTo, Inc. All rights reserved.

(Refer Slide Time: 12:11)

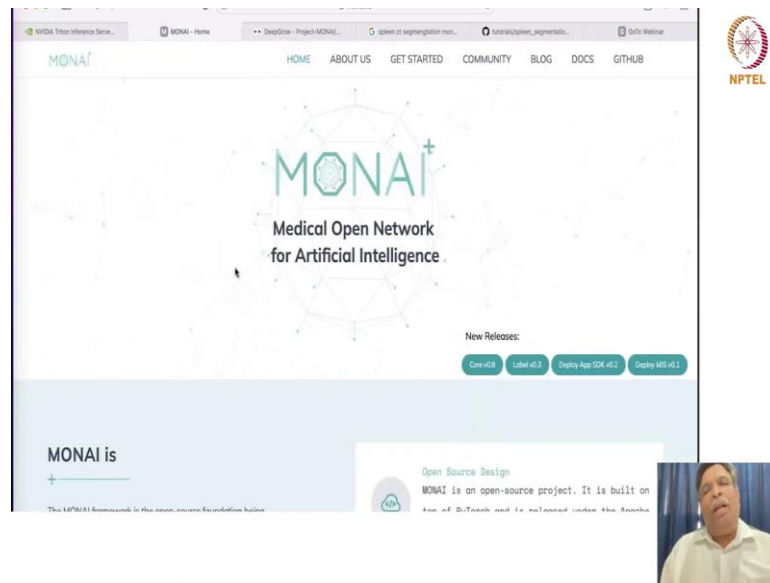


(Refer Slide Time: 12:17)

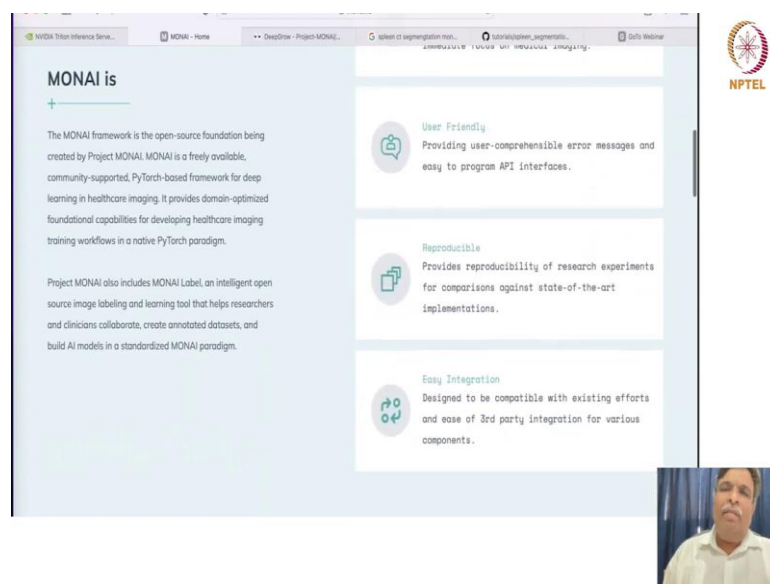


So, let us try to show some demo on that. So, before that let me just show you two or three things which we will be using there is something which is called as Triton inference server. So, this basically is a inference server which streamlines the AI inferencing right.

(Refer Slide Time: 12:30)



(Refer Slide Time: 12:38)



So, this is basically return inference server then you have something which I told you which is medical open network for artificial intelligence or MONAI right. So, it is a open source project. So, it is built on top of PyTorch and basically it is a very very good framework right.

(Refer Slide Time: 12:50)

VIEW PAGE ON GITHUB.COM

⚠ The indexable preview below may have rendering errors, broken links, missing images, and does not include the last updated date. Please view the original page on GitHub.com and not this indexable preview if you intend to use this content.

3D Volume → 2D Slices

2D Slice → U-net → Segmentation

Positive Guidance Map

Negative Guidance Map

MONAI Label employs DeepGrow for the annotation of 3D medical images (Magnetic Resonance (MR) or Computed Tomography (CT)).

About GitHub Wiki SEE: a search engine enabler for GitHub Wikis as GitHub blocks many GitHub Wikis from search engines.

So, then there is something which is called as deep grow which we will be using. So, I will show you that as well when we are using MONAI for labeling right. So, annotation. So, all of this is used. So, it uses a unit architecture for segmentation and this is one thing which I thought we will show you before we start. So, yes. So, these are three things MONAI, then Triton, and then deep grow ok.

(Refer Slide Time: 13:20)

Project-MONAI/tutorials

Code Issues 31 Pull requests 8 Discussions 0 Actions 0 Security 0 Insights

master tutorials/3d_segmentation/spleen_segmentation_3d.ipynb

JK-rez Typo removal (#662) Latest commit 4187a37 1 hour ago History

999 lines 1999 lines 579 KB

Spleen 3D segmentation with MONAI

This tutorial shows how to integrate MONAI into an existing PyTorch medical DL program. And easily use below features:

1. Transforms for dictionary format data.
2. Load NIfTI image with metadata.
3. Add channel dim to the data if no channel dimension.
4. Scale medical image intensity with expected range.
5. Crop out a batch of balanced images based on positive / negative label ratio.
6. Cache IO and transforms to accelerate training and validation.

(Refer Slide Time: 13:26)

Spleen 3D segmentation with MONAI

This tutorial shows how to integrate MONAI into an existing PyTorch medical DL program.

And easily use below features:

1. Transforms for dictionary format data.
2. Load Nifti image with metadata.
3. Add channel dim to the data if no channel dimension.
4. Scale medical image intensity with expected range.
5. Crop out a batch of balanced images based on positive / negative label ratio.
6. Cache IO and transforms to accelerate training and validation.
7. 3D UNet model, Dice loss function, Mean Dice metric for 3D segmentation task.
8. Sliding window inference method.
9. Deterministic training for reproducibility.

The Spleen dataset can be downloaded from <http://medicaldecathlon.com/>.

Target: Spleen
Modality: CT
Size: 61 3D volumes (41 Training + 20 Testing)
Source: Memorial Sloan Kettering Cancer Center

And then there is something which is called a spleen segmentation which will show you. So, this is spleen segmentation with MONAI. So, this is setting up the environment and so on and so forth. So, we will try to show you on our local machine ok instead of just trying to run it on this thing ok. So, Google Colab or something, but yeah.

(Refer Slide Time: 13:50)

For details about installing the optional dependencies, please visit:
<https://docs.monai.io/en/latest/installation.html#installing-the-recommended-dependencies>

Setup data directory

You can specify a directory with the `MONAI_DATA_DIRECTORY` environment variable. This allows you to save results and reuse downloads. If not specified a temporary directory will be used.

```
In [33]: directory = os.environ.get('MONAI_DATA_DIRECTORY')
root_dir = tempdir.mkdtemp() if directory is None else directory
print(root_dir)

/nctspace/data/medical/
```

Download dataset

Downloads and extracts the dataset. The dataset comes from <http://medicaldecathlon.com/>.

```
In [4]: resource = "https://med4c-monai.s3-us-west-2.amazonaws.com/task09_spleen.tar"
md5 = "41806a3f1d4e5b2f5f6ec3d8a524e"

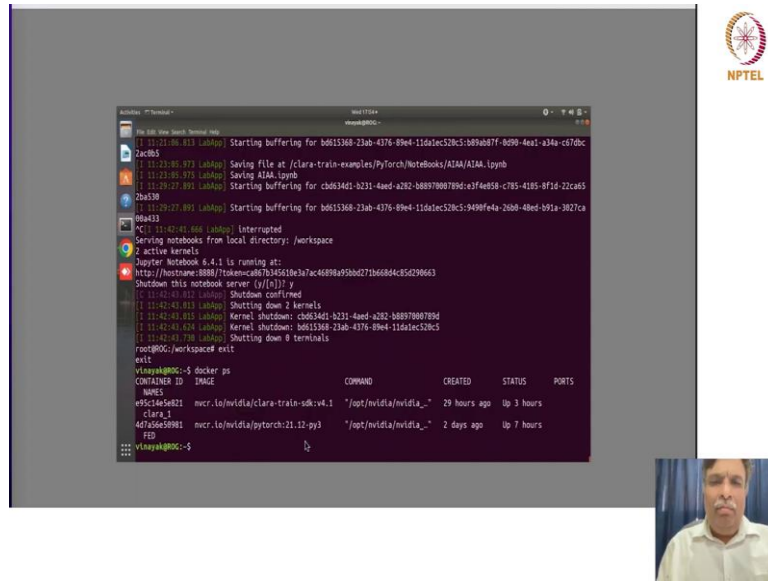
compressed_file = os.path.join(root_dir, "task09_spleen.tar")
data_dir = os.path.join(root_dir, "task09_spleen")
if not os.path.exists(data_dir):
    download_and_extract(resource, compressed_file, root_dir, md5)
```

Set MSD Spleen dataset path

```
In [53]: train_images = sorted(
glob.glob(os.path.join(data_dir, "images"), "**.nii.gz"))
train_labels = sorted(
```

So, this is how we are going to show you our local this thing. So, let me just show you that. So, we will again be using dockers. So, we will zoom in a bit yeah.

(Refer Slide Time: 14:07)



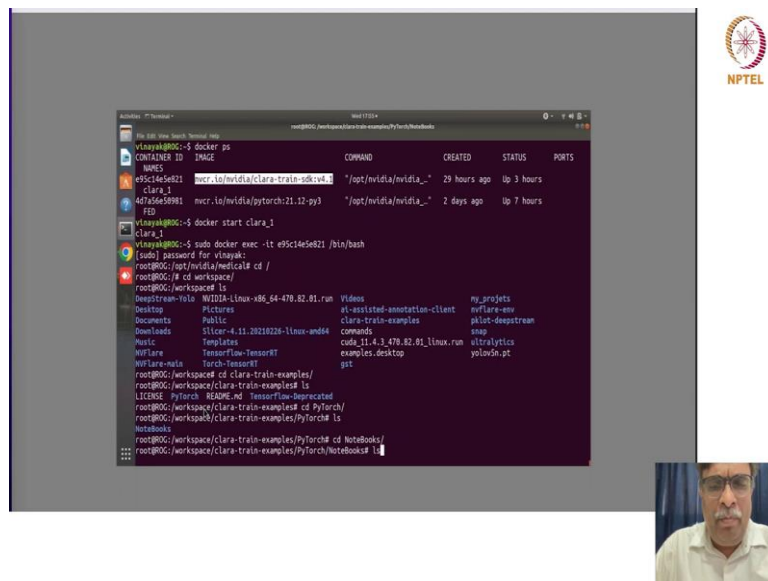
The terminal window shows the following output:

```
Starting buffering for b615368-23ab-4376-89e4-11da1ec52bc5:bb9bb7f-6d99-4ea2-e34a-c570bc
2ac0b5
11:21:20.873 LabApp Saving file at /clara-train-examples/PyTorch/notebooks/ADAM/ADAM.ipynb
11:21:20.873 LabApp Saving ADAM.ipynb
11:21:27.893 LabApp Starting buffering for cdb63461-b231-4aed-a282-b8897808789d:e3f4e58-c785-4185-9f5d-22ca65
2ba538
11:21:27.893 LabApp Starting buffering for b615368-23ab-4376-89e4-11da1ec52bc5:949ff4e-268d-b91a-3027ca
994431
[Ctrl] 11:42:41.668 LabApp Interrupted
Saving notebooks from local directory: /workspace
2 active kernels
Jupyter Notebook 6.4.1 is running at:
http://hostname:8888/?token=c6b7b34510e3a7ac4689ba95b0d271b61884c85c290663
Shutdown this notebook server [y|n]: y
11:42:41.812 LabApp Shutdown confirmed
11:42:41.813 LabApp Shutting down 2 kernels
11:42:41.813 LabApp Kernel shutdown: cdb63461-b231-4aed-a282-b8897808789d
11:42:41.824 LabApp Kernel shutdown: b615368-23ab-4376-89e4-11da1ec52bc5
11:42:41.778 LabApp Shutting down 0 terminals
root@ROC:/workspace# exit
vlayak@ROC:~$ docker ps
```

NAME	CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
e95c14e5e821	mucr.io/nvidia/clara-train-sdk:v1.1		"/opt/nvidia/nvidia_..."	29 hours ago	Up 3 hours	
clara_1	4d7a56e59881	mucr.io/nvidia/pytorch:21.12-py3	"/opt/nvidia/nvidia_..."	2 days ago	Up 7 hours	

The video inset shows a man with a mustache and glasses, wearing a light-colored shirt, speaking.

(Refer Slide Time: 14:15)



The terminal window shows the following output:

```
vlayak@ROC:~$ docker ps
```

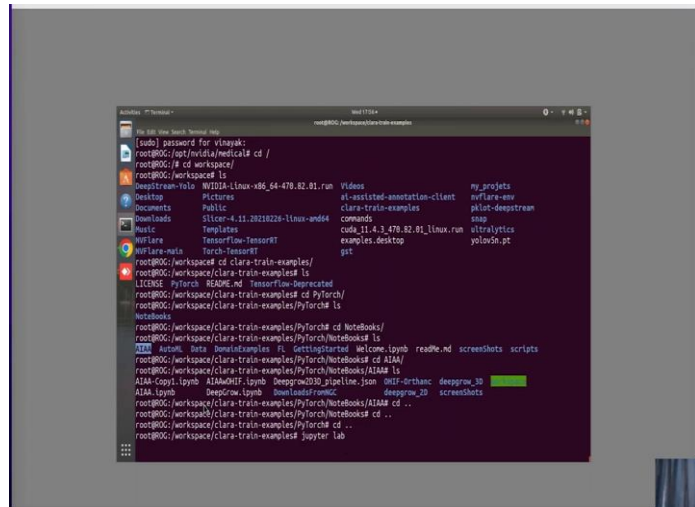
NAME	CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
e95c14e5e821	mucr.io/nvidia/clara-train-sdk:v1.1		"/opt/nvidia/nvidia_..."	29 hours ago	Up 3 hours	
clara_1	4d7a56e59881	mucr.io/nvidia/pytorch:21.12-py3	"/opt/nvidia/nvidia_..."	2 days ago	Up 7 hours	

```
vlayak@ROC:~$ docker start clara_1
clara_1
vlayak@ROC:~$ sudo docker exec -it e95c14e5e821 /bin/bash
[sudo] password for vlayak:
root@ROC:/workspace# cd /
root@ROC:~# cd workspace/
root@ROC:/workspace# ls
DeepStream-v1a  W10128-1linux-x86_64-478.82.01.run  Videos  py_projects
Desktop        Pictures                                  at-assisted-annotation-client  nvidia-env
Documents     Public                                   clara-train-examples          pilot-deepstream
Downloads     Sltcor-4.11.20210228-1linux-amd64      commands                       snap
Music          Templates                                 cudv_11.4.3_478.82.01_linux.run  ultralytics
WiFiLan-v1a   TensorFlow-TensorRT                       examples.desktop              yolov5n.pt
root@ROC:/workspace# cd clara-train-examples/
root@ROC:/workspace/clara-train-examples# ls
LICENSE  PyTorch  README.md  tensorflow-deepestest
root@ROC:/workspace/clara-train-examples# cd PyTorch/
root@ROC:/workspace/clara-train-examples/PyTorch# ls
notebooks/
root@ROC:/workspace/clara-train-examples/PyTorch# cd notebooks/
root@ROC:/workspace/clara-train-examples/PyTorch/notebooks# ls
```

The video inset shows the same man as in the previous slide, speaking.

So, we are using this Clara train SDK version full dot one. So, we will start this docker. So, the docker has started, yes password yes, yeah. So, many of them, but we will try to show you some examples which are there in the Clara train folder yeah. So, we will go to the PyTorch thing a lot of notebooks available. So, we will go them, yes.

(Refer Slide Time: 15:22)



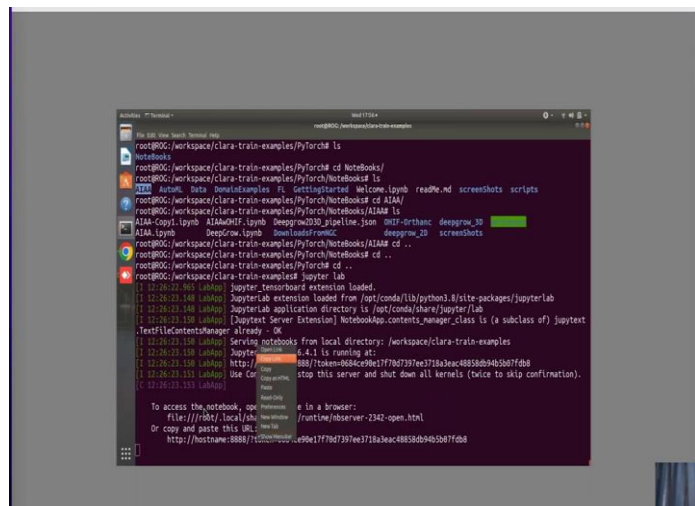
The terminal window shows the following commands and output:

```
root@ROC:~/workspace# cd /
root@ROC:~/workspace# cd workspace/
root@ROC:~/workspace# ls
Desktop  Pictures  vi-assisted-annotation-client  my_projects
Documents  public  clara-train-examples  nflare-env
Downloads  Slicer-4.11.2018228-linux-amd64  commands  plot-deepstream
nflare  Templates  cuda_11.4.3_478.82.01_linux.run  ultralytics
nflare-mia  Tensorflow-tensorflow  examples.desktop  yolov3n.pt
nflare-nia  torch-tensorflow  gpt

root@ROC:~/workspace# cd clara-train-examples/
root@ROC:~/workspace/clara-train-examples# ls
LICENSE  PyTorch  S2S2S2C.md  tensorflow-deprecated
root@ROC:~/workspace/clara-train-examples# cd PyTorch/
root@ROC:~/workspace/clara-train-examples/PyTorch# ls
Notebooks
root@ROC:~/workspace/clara-train-examples/PyTorch# cd Notebooks/
root@ROC:~/workspace/clara-train-examples/PyTorch/Notebooks# ls
AutofML  Beta  DownloadExamples  FL  GettingStarted  Welcome  Iyumb  readMe.md  screenshots  scripts
root@ROC:~/workspace/clara-train-examples/PyTorch/Notebooks# cd AIAA/
root@ROC:~/workspace/clara-train-examples/PyTorch/Notebooks/AIAA# ls
AIAA-Copy1.lyymb  AIAA-GHIF.lyymb  Deepgrow2D.pipeline.json  DMTF-Orthanc  deepgrow_3D
AIAA.lyymb  DeepGrow.lyymb  DownloadFromNGC  deepgrow_2D  screenshots
root@ROC:~/workspace/clara-train-examples/PyTorch/Notebooks/AIAA# cd ..
root@ROC:~/workspace/clara-train-examples/PyTorch/Notebooks# cd ..
root@ROC:~/workspace/clara-train-examples/PyTorch# cd ..
root@ROC:~/workspace/clara-train-examples# jupyter lab
```

So, we will go to this AIAA which is AI Assisted Annotation, see there is there are so many of these things ok. So, we have got deepgrow 2D, deepgrow 3D ok and since we are trying to use the docker from NGC we will try to use NGC a lot yeah.

(Refer Slide Time: 16:04)

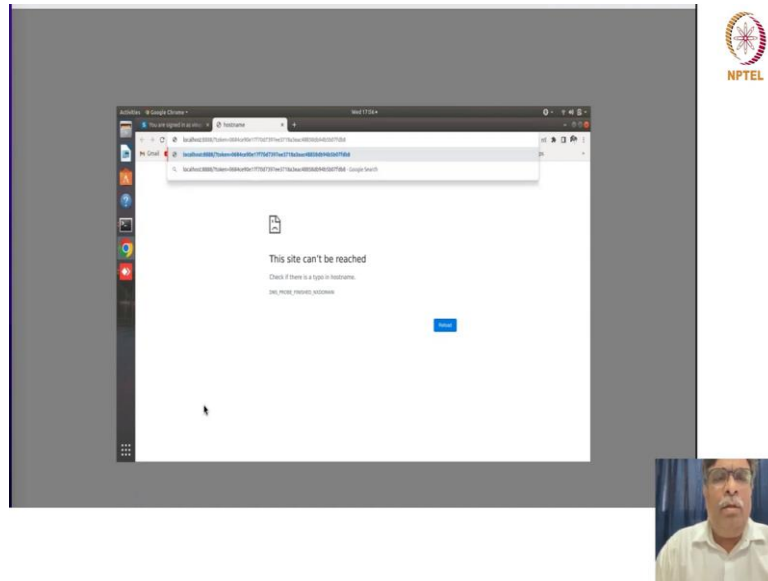


The terminal window shows the following output:

```
root@ROC:~/workspace/clara-train-examples/PyTorch# ls
Notebooks
root@ROC:~/workspace/clara-train-examples/PyTorch# cd Notebooks/
root@ROC:~/workspace/clara-train-examples/PyTorch/Notebooks# ls
AutofML  Beta  DownloadExamples  FL  GettingStarted  Welcome  Iyumb  readMe.md  screenshots  scripts
root@ROC:~/workspace/clara-train-examples/PyTorch/Notebooks# cd AIAA/
root@ROC:~/workspace/clara-train-examples/PyTorch/Notebooks/AIAA# ls
AIAA-Copy1.lyymb  AIAA-GHIF.lyymb  Deepgrow2D.pipeline.json  DMTF-Orthanc  deepgrow_3D
AIAA.lyymb  DeepGrow.lyymb  DownloadFromNGC  deepgrow_2D  screenshots
root@ROC:~/workspace/clara-train-examples/PyTorch/Notebooks/AIAA# cd ..
root@ROC:~/workspace/clara-train-examples/PyTorch/Notebooks# cd ..
root@ROC:~/workspace/clara-train-examples/PyTorch# cd ..
root@ROC:~/workspace/clara-train-examples# jupyter lab
[12-26-23 06:18:05] LabApp JupyterLab extension loaded.
[12-26-23 06:18:05] LabApp JupyterLab extension loaded from /opt/conda/lib/python3.8/site-packages/jupyterlab
[12-26-23 06:18:05] LabApp JupyterLab application directory is /opt/conda/share/jupyter/lab
[12-26-23 06:18:10] LabApp [Jupyter Server Extension] NotebookApp.contents_manager_class is (a subclass of) JupyterTextFileContentsManager already - OK
[12-26-23 06:18:10] LabApp Searching notebooks from local directory: /workspace/clara-train-examples
[12-26-23 06:18:10] LabApp JupyterLab 4.1.1 is running at:
[12-26-23 06:18:10] LabApp http://localhost:8888/?token=8684ce98e1779d7397ee3718a3eac4855d9455d7fd8
[12-26-23 06:18:10] LabApp Use Ctrl-C to stop this server and shut down all kernels. (twice to skip confirmation).

To access this notebook, open a browser:
  http://localhost:8888/?token=8684ce98e1779d7397ee3718a3eac4855d9455d7fd8
Or copy and paste this URL:
  http://hostname:8888/?token=8684ce98e1779d7397ee3718a3eac4855d9455d7fd8
```

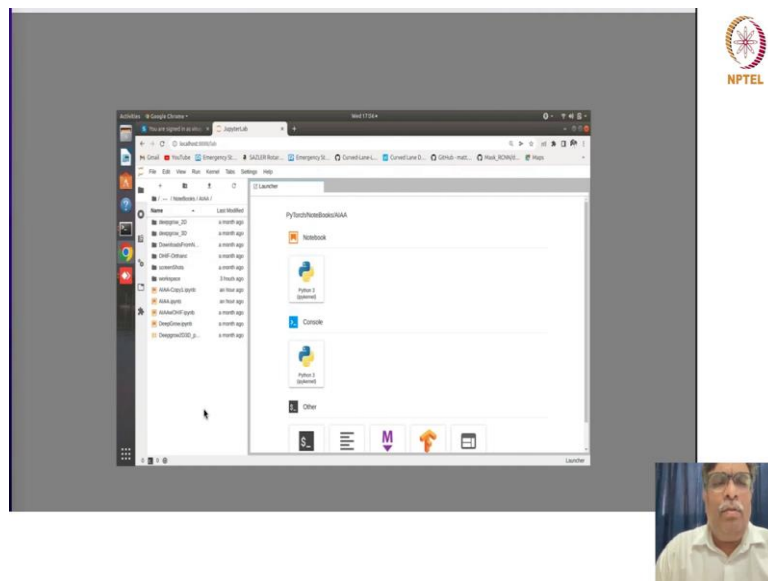

(Refer Slide Time: 16:06)



The screenshot shows a Google Chrome browser window with a white background and a blue header. The main content area displays the message "This site can't be reached" in black text, followed by "Check if there is a typo in the address." and "DNS_PROBE_FINISHED_NXDOMAIN" in smaller text. A blue "Refresh" button is centered below the text. The address bar shows a URL starting with "https://". A small inset video of a man is visible in the bottom right corner.



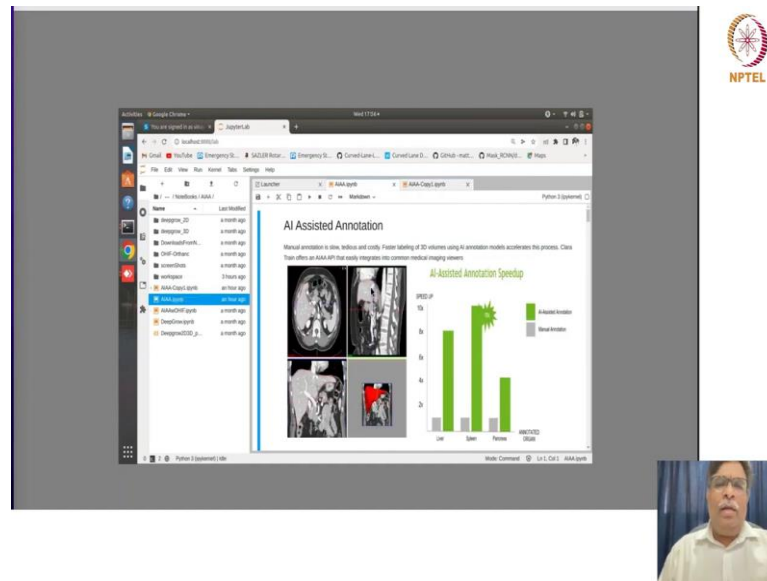
(Refer Slide Time: 16:23)



The screenshot shows a JupyterLab interface. On the left, there is a file browser showing a directory structure with files like "image_01.png" and "image_02.png". On the right, there is a notebook editor with a "Python 3" kernel and a "Notebook" icon. A small inset video of a man is visible in the bottom right corner.



(Refer Slide Time: 16:34)

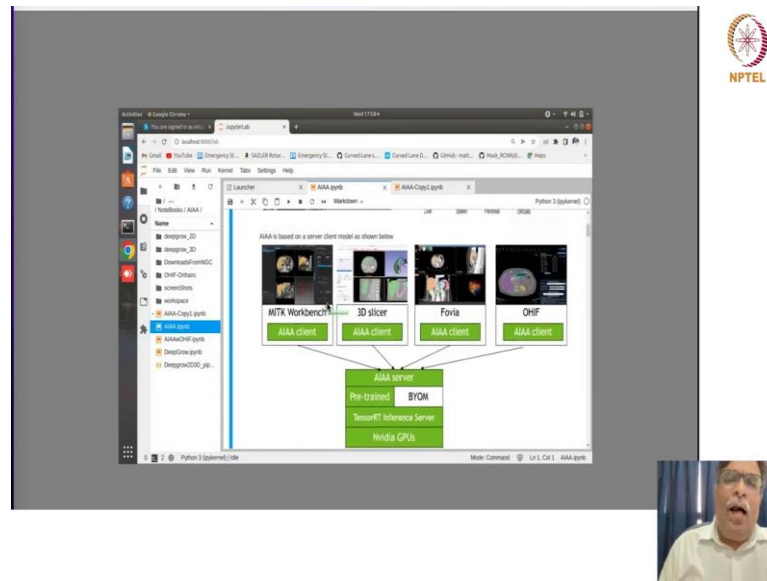


So, we have now opened Jupyter notebook, I will change the alphabets yes. So, if you see you got all the examples which I showed you. We will try to use AI assisted annotation yeah. So, 1 minute I will enlarge this a bit ok, I suppose this is visible to everyone a bit of it, people can scroll and see yeah I suppose its visible. So, one of the ideas of trying to come up with this AI assistant annotation is manual annotation is very very slow right everybody knows and it is a costly affair.

And then when you want to label 3D volumes right, it is still very costly here and very very domain specific right annotate yeah. So, when you are talking of 3D volumes, I will just increase the size again let me see yes. So, when you are talking of 3D volumes right that again is basically a very very difficult scenario right because it has to be very very domain specific and doing that annotation is a very very tedious job right.

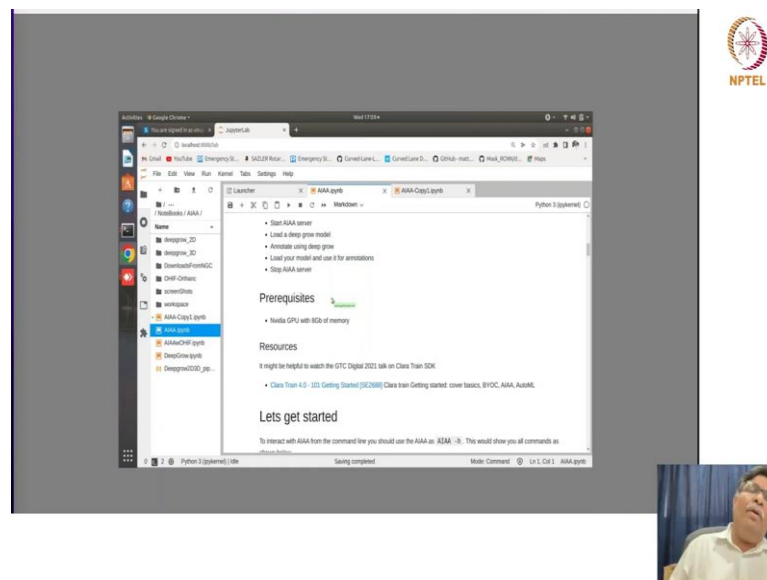
So, if you can get some trained model right which will help you to do annotation, which will assist at least to do annotation a bit and then it speeds up your annotation thing also right. So, yeah.

(Refer Slide Time: 18:17)



So, if you see this as I told you that this AIAA is basically a client server model. So, it has got a workbench, you have a 3D slicer, you have a 3D slicer and then you have this fovia client, OHIF client.

(Refer Slide Time: 18:42)

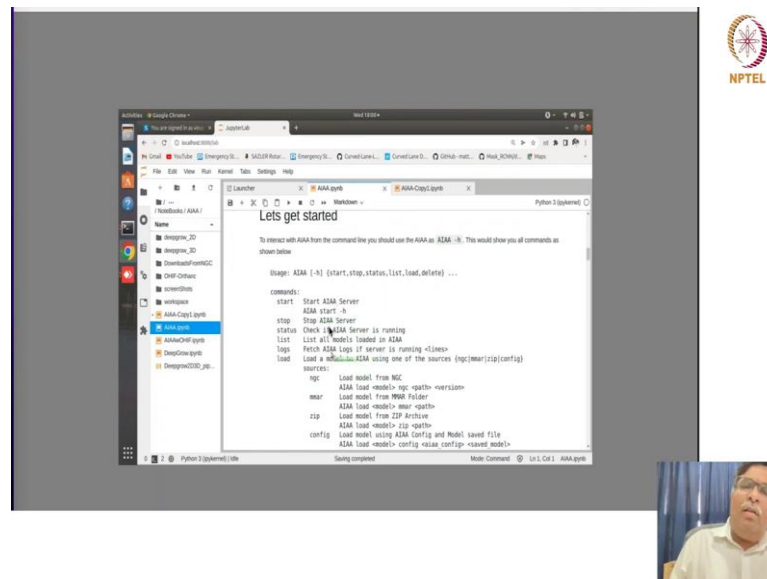


So, I am not going into the details of each and everything, but at least you will at least understand right, that you can basically be using this 3D slicer and this MITK workbench to some extent to do some annotation ok.

So, these are all clients and. So, basically you start your AIAA server. Now, you can start the server, put it on a local disk or you can directly connect it to the NVIDIA NGC cloud ok. So, we are trying to show you with our own local AIAA server, then you load a deep grow model, annotate using deep grow and then we can load our own model and use it for annotations as well and then we can stop the AIAA server right. So, this is helping you to actually do some automatic annotation right rather than doing anything else. So, yeah.

So, let us try to do things with the condition of course, the prerequisite is that NVIDIA GPU with 8 GB of memory is required. So, that is fine. So, yeah.

(Refer Slide Time: 19:47)



So, this is how we are going to do. So, we are going to start the server right and we are then going to load all the models which are there or whichever model we want, we load that model actually. We can load it from NGC as I told you or we can load it from a local server in both the cases.

(Refer Slide Time: 20:05)

The screenshot displays a JupyterLab environment with a Python 3 (ipykernel) kernel. The main window shows a script named 'ASAM.py' with the following code:

```

import os
import sys
import time
import subprocess

def start():
    print("Starting ASAM Server")
    status = check_if_server_is_running()
    list_models = list_all_models_loaded_in_ASAM()
    if status == 'running':
        print("ASAM Server is already running")
        return
    # Fetch ASAM logs if server is running
    if status == 'running':
        fetch_logs()
    # Load a model to ASAM using one of the sources (ngc|local|zip|config)
    sources = ['ngc', 'local', 'zip', 'config']
    for source in sources:
        if source == 'ngc':
            load_model_from_ngc()
        elif source == 'local':
            load_model_from_local_folder()
        elif source == 'zip':
            load_model_from_zip_archive()
        elif source == 'config':
            load_model_using_ASAM_Config_and_Model_saved_File()
    pipeline = load_pipeline_using_ASAM_Config()
    delete_model()

def stop():
    print("Stopping ASAM Server")
    # Delete a model from ASAM
    delete_model()

def check_if_server_is_running():
    # Before we get started let us check that we have an NVIDIA GPU available in the docker by running the call below
    # Following command should show all gpus available
    result = subprocess.run('nvidia-smi')
    return result

def list_all_models_loaded_in_ASAM():
    # Following command should show all gpus available
    result = subprocess.run('nvidia-smi')
    return result

def fetch_logs():
    # Following command should show all gpus available
    result = subprocess.run('nvidia-smi')
    return result

def load_model_from_ngc():
    # Following command should show all gpus available
    result = subprocess.run('nvidia-smi')
    return result

def load_model_from_local_folder():
    # Following command should show all gpus available
    result = subprocess.run('nvidia-smi')
    return result

def load_model_from_zip_archive():
    # Following command should show all gpus available
    result = subprocess.run('nvidia-smi')
    return result

def load_model_using_ASAM_Config_and_Model_saved_File():
    # Following command should show all gpus available
    result = subprocess.run('nvidia-smi')
    return result

def pipeline():
    # Following command should show all gpus available
    result = subprocess.run('nvidia-smi')
    return result

def delete_model():
    # Following command should show all gpus available
    result = subprocess.run('nvidia-smi')
    return result

if __name__ == '__main__':
    start()
    
```

The terminal output shows the execution of the 'start()' function, which checks if the server is running and then loads a model from the local folder.

(Refer Slide Time: 20:04)

The screenshot displays the output of the 'nvidia-smi' command in the terminal window. The output is as follows:

```

Wed Apr 20 12:38:38 2022
+-----+
| NVIDIA-SMI 470.38.01 | Driver Version: 470.38.01 | CUDA Version: 11.4 | | |
| GPU Name Persistence-M | Bus-Id | Disp-A | Part. # | Persistence |
| Fan Temp Perf. | Pwr.Usage | Temp | Perf. | Pwr.Usage |
|-----+-----+-----+-----+-----+-----+-----+-----+
| 0 | NVIDIA GeForce RTX 3090 | 00000000:01:00:03 | 0 | On | 100% | 100% |
| N/A | 55C | P5 | 13w / 100w | 47C/50C / 50/45C | 21% | Default | N/A |
+-----+-----+-----+-----+-----+-----+-----+
Processes:
| GPU | PID | Type | Process name | GPU Memory Usage |
+-----+-----+-----+-----+-----+-----+
| 0 | 104 | C | /usr/bin/python3 | 100% |
+-----+-----+-----+-----+-----+-----+

```

(Refer Slide Time: 20:22)

The screenshot shows a JupyterLab interface with a terminal window. The terminal output is as follows:

```

1. Start AIAA server

First lets set up AIAA path and change some permissions

AIAA_ROOT="/workspace/Clara-train-examples/PyTorch/NeTrekku/AIAA/workspace/"
AIAA_ROOT=$(pwd)
mkdir -p $AIAA_ROOT
chmod 777 $AIAA_ROOT
export "AIAA_ROOT=$AIAA_ROOT"

AIAA_ROOT is set to "/workspace/Clara-train-examples/PyTorch/NeTrekku/AIAA/workspace/"

1.2 AIAA Server with Triton BackEnd

Starting V4 Triton server was moved out of the class train container. In order to use Triton as the AIAA backend, you should use
/workspace/Clara-train-examples/PyTorch/NeTrekku/AIAA/workspace/
  
```

The NPTEL logo is visible in the top right corner of the slide.

(Refer Slide Time: 20:32)

The screenshot shows the terminal output for starting the AIAA server:

```

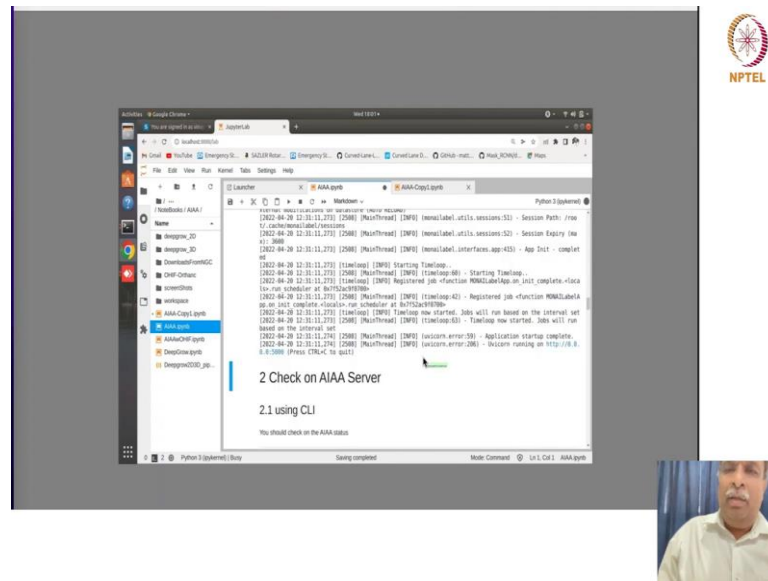
AIAA start = $AIAA_ROOT --engine TRITON --triton_ip tritonserver

Using PYTHONPATH=/opt/nvidia/medical/aiiaa --opt/nvidia/medical
[2022-08-28 11:21:18.824] [INFO] [MainThread] [INFO] (nameLabel_main-20) - ENGINE: workspace = /workspace
[2022-08-28 11:21:18.825] [INFO] [MainThread] [INFO] (nameLabel_main-20) - ENGINE: device = /dev
[2022-08-28 11:21:18.825] [INFO] [MainThread] [INFO] (nameLabel_main-20) - ENGINE: conf = None
[2022-08-28 11:21:18.825] [INFO] [MainThread] [INFO] (nameLabel_main-20) - ENGINE: No = 8.0.0
[2022-08-28 11:21:18.825] [INFO] [MainThread] [INFO] (nameLabel_main-20) - ENGINE: port = 5008
[2022-08-28 11:21:18.825] [INFO] [MainThread] [INFO] (nameLabel_main-20) - ENGINE: no_config = None
[2022-08-28 11:21:18.825] [INFO] [MainThread] [INFO] (nameLabel_main-20) - ENGINE: group = Full
[2022-08-28 11:21:18.825] [INFO] [MainThread] [INFO] (nameLabel_main-20) - ENGINE: ssl_keyfile = None
[2022-08-28 11:21:18.825] [INFO] [MainThread] [INFO] (nameLabel_main-20) - ENGINE: ssl_certfile = None
[2022-08-28 11:21:18.825] [INFO] [MainThread] [INFO] (nameLabel_main-20) - ENGINE: ssl_keyfile_password = None
[2022-08-28 11:21:18.825] [INFO] [MainThread] [INFO] (nameLabel_main-20) - ENGINE: ssl_ca_certfile = None
  
```

The NPTEL logo is visible in the top right corner of the slide.

So, let us try to understand this program and run it ok ah. So, this is your NVIDIA SMI which tells us that we are using NVIDIA GPU geforce and then this basically helps us start the AIAA server, we are trying to work with the server which is placed in our own workspace. So, we are starting that server, then this basically is trying to understand the inferencing server which is there in the back end. So, we will try to start that triton server as well. So, once it started.

(Refer Slide Time: 20:57)

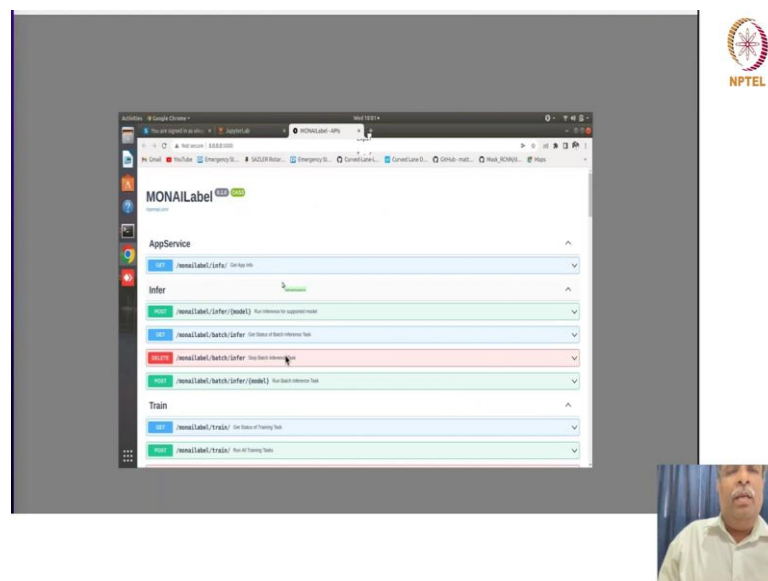


2 Check on AIAA Server

2.1 using CLI

You should check on the AIAA status

(Refer Slide Time: 20:58)



MONAILabel

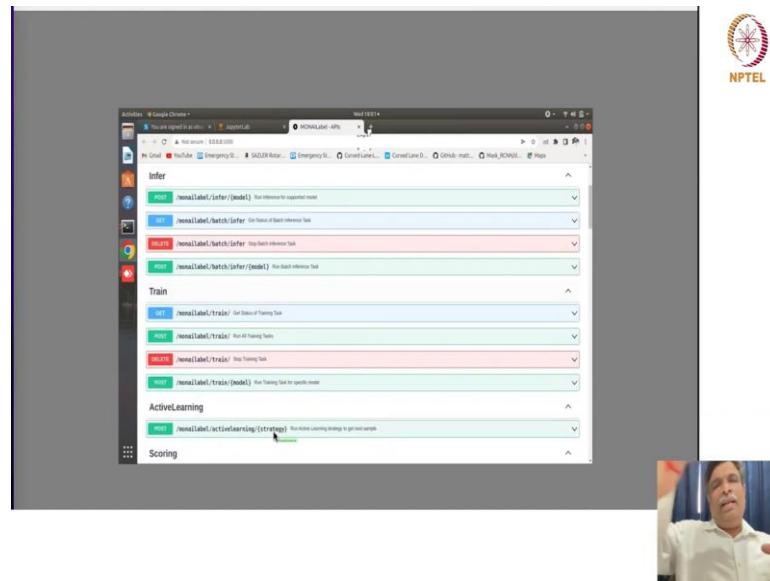
AppService

Infer

Train

So, we are using MONAI labeling there internally and then we got this right MONAI labeling, this is app service, this is for inferencing, this is for training right.

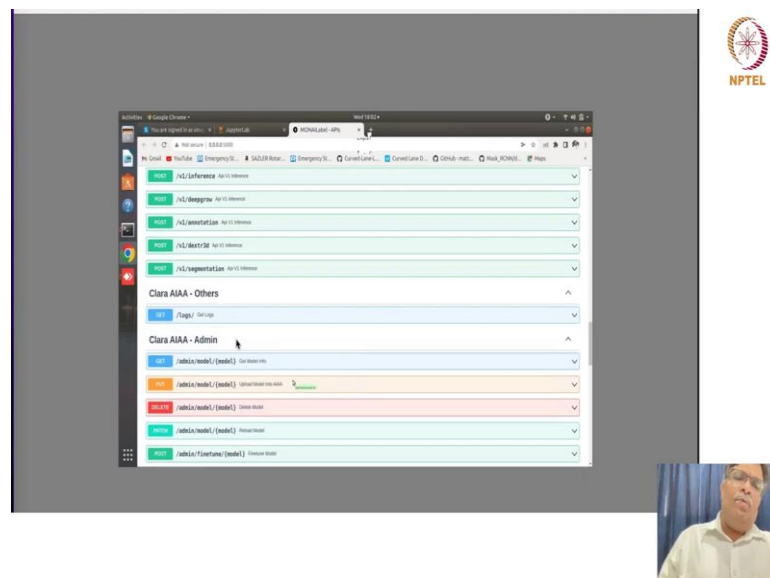
(Refer Slide Time: 21:10)



The screenshot displays a web application interface with a sidebar on the left and a main content area. The main content area is divided into several sections: 'Infer', 'Train', 'ActiveLearning', and 'Scoring'. Each section contains a list of tasks with their names and descriptions. The 'Infer' section has four tasks, 'Train' has three, 'ActiveLearning' has one, and 'Scoring' has one. The tasks are listed in a table-like format with columns for task name and description. An NPTEL logo is visible in the top right corner of the slide.

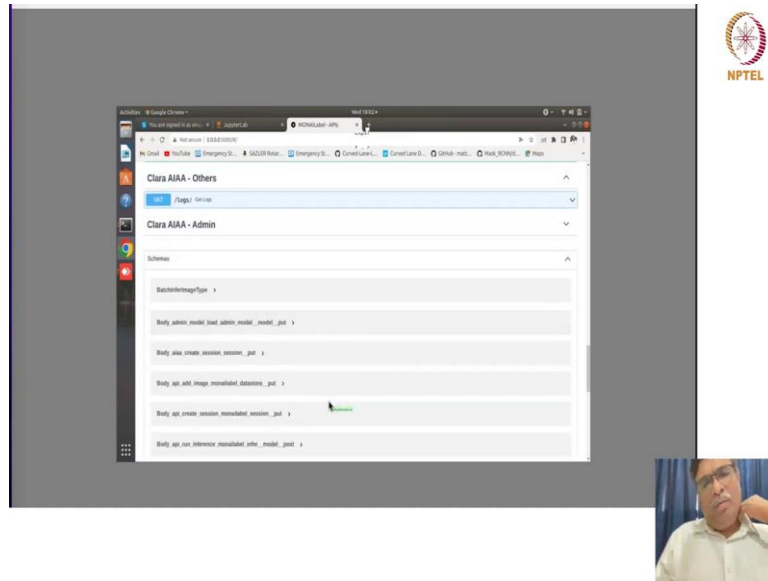
So, MONAI is integrated to Clara ok and then this basically has got a lot of things which you can do ok, yeah. So, you have got the data store, you can use it, you can do so many things right. So, yeah you have got sessions, you have got Clara AIAA sessions which are going which will be going on and which are present. So, all of this information is available in this local host right. So, many APIS which you can use and we are using yeah. So, yeah. So, you have admin thing also here right.

(Refer Slide Time: 21:40)



The screenshot displays a web application interface with a sidebar on the left and a main content area. The main content area is divided into two sections: 'Clara AIAA - Others' and 'Clara AIAA - Admin'. Each section contains a list of tasks with their names and descriptions. The 'Clara AIAA - Others' section has five tasks, and the 'Clara AIAA - Admin' section has five tasks. The tasks are listed in a table-like format with columns for task name and description. An NPTEL logo is visible in the top right corner of the slide.

(Refer Slide Time: 21:50)



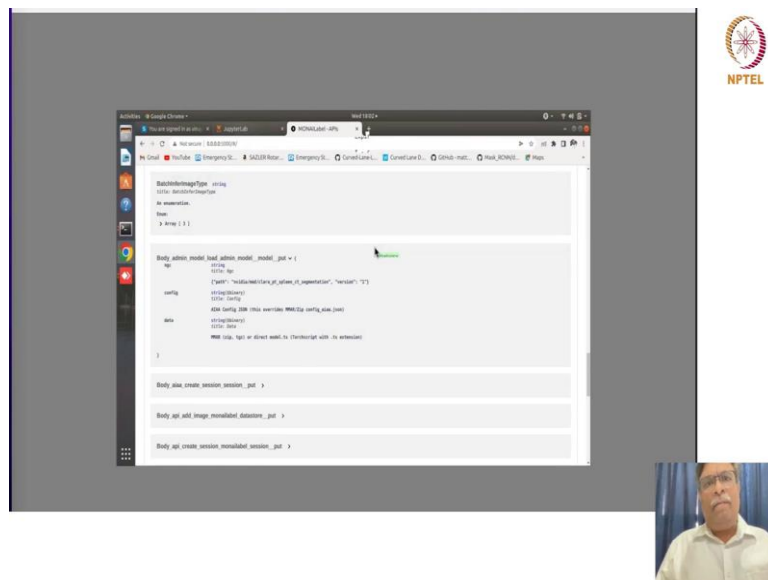
The screenshot shows a REST client interface in a browser window. The address bar indicates the URL is 'localhost:8080'. The page title is 'Clara AIAA - Admin'. Under the 'Schemas' section, there is a list of endpoints with expandable arrows:

- BackendImageType >
- Study admin model load admin model _post >
- Study admin create session _post >
- Study app add image memahabid database _post >
- Study app create session memahabid session _post >
- Study app run inference memahabid other_model _post >

The NPTEL logo is visible in the top right corner of the slide.

So, many schemas are available which you can use and yeah. So, many APIS which you can use. So, all of this thing is available that information.

(Refer Slide Time: 22:05)



The screenshot shows the same REST client interface, but now displaying the details of the selected schema: 'Study admin model load admin model _post = {}'. The details are as follows:

- Method: POST
- URL: localhost:8080/api
- Headers: Content-Type: application/json
- Body: {}

The NPTEL logo is visible in the top right corner of the slide.

(Refer Slide Time: 23:48)

3. Load models in AIAA Server.

For this notebook we can download models from [HuggingFace](#). Models on HuggingFace are either:

- Annotation Models

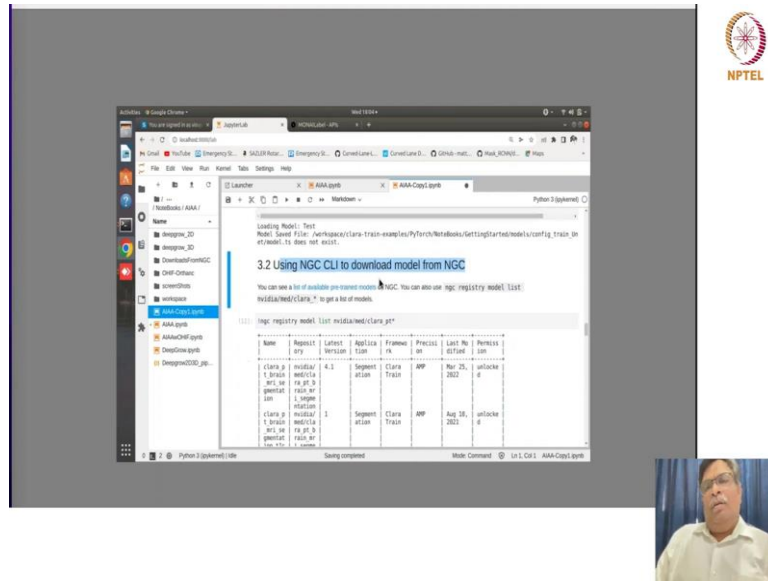
Deep Learning Gaze From Extreme Points to Object Segmentation

(Refer Slide Time: 23:47)

3.0 Loading options

In order to load a model to AIAA we will use the AIAA UI as

(Refer Slide Time: 23:56)



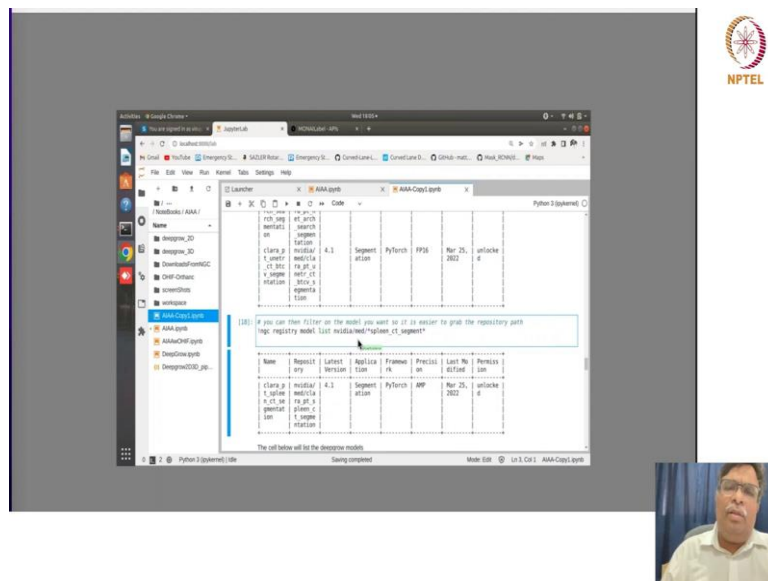
3.2 Using NGC CLI to download model from NGC

You can see a list of available pre-trained models on NGC. You can also use `ngc registry model list`.
`ngc registry model list mistral/med/clara_p*`

Name	Repository	Latest Version	Applica	Frameo	Precisio	Last Mo	Permiss
clara_p_medical	mistral/med/clara_p_medical	4.3	Segment	Clara	APP	Mar 25, 2022	unlocke
clara_p_medical	mistral/med/clara_p_medical	4.3	Segment	Clara	APP	Mar 25, 2022	unlocke
clara_p_medical	mistral/med/clara_p_medical	4.3	Segment	Clara	APP	Mar 25, 2022	unlocke
clara_p_medical	mistral/med/clara_p_medical	4.3	Segment	Clara	APP	Mar 25, 2022	unlocke
clara_p_medical	mistral/med/clara_p_medical	4.3	Segment	Clara	APP	Mar 25, 2022	unlocke

So, now we can use our own browser to see. So, load the models in AIAA server now. So, we will show you all that once it is loaded, we are trying to load it from the NGC directly; we are not connecting it to the NGC server which is available for Clara. We are directly putting that model downloading from the NGC cloud and putting it onto our local server and then showing you this.

(Refer Slide Time: 24:38)



If you can then filter on the model you want so it is easier to grab the repository path
`ngc registry model list mistral/med/splase_ct_segment*`

Name	Repository	Latest Version	Applica	Frameo	Precisio	Last Mo	Permiss
clara_p_medical	mistral/med/clara_p_medical	4.3	Segment	Clara	APP	Mar 25, 2022	unlocke
clara_p_medical	mistral/med/clara_p_medical	4.3	Segment	Clara	APP	Mar 25, 2022	unlocke
clara_p_medical	mistral/med/clara_p_medical	4.3	Segment	Clara	APP	Mar 25, 2022	unlocke
clara_p_medical	mistral/med/clara_p_medical	4.3	Segment	Clara	APP	Mar 25, 2022	unlocke
clara_p_medical	mistral/med/clara_p_medical	4.3	Segment	Clara	APP	Mar 25, 2022	unlocke

There are so many of things right, you have got segmentation, then Clara, spleen then pancreas and tumor city segmentation, all this annotation segmentation, annotation then

classification all of these are available actually. So, these models are available and then know we can filter the required model ok and then yeah. So, we have got these deep grown models deep grow models.

(Refer Slide Time: 25:01)

Name	Repository	Latest Version	Application	Framework	Precision	Last Modified	Permissions
clara_pt_spleen_ct_seg	nvista/med4cls_pt_spleen_ct_seg	4.1	Spleen	PyTorch	APP	Mar 25, 2022	unlock

The cell below will be the deepgrow model.

If you can then filter on the model you want to go to section to grab the repository path
 http://registry.nvidia.com/catalog/med4cls_pt_spleen_ct_seg

The cell below will download the deep grow model from NGC and load it to AIAA.

(Refer Slide Time: 25:17)

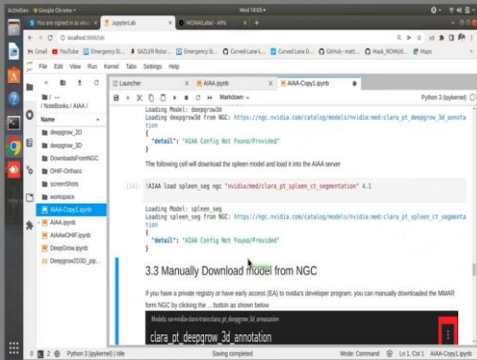
```

AIAA load deepgrow http://registry.nvidia.com/catalog/med4cls_pt_spleen_ct_seg 4.1
The following cell will download the spleen model and load it into the AIAA server

AIAA load spleen_seg http://registry.nvidia.com/catalog/med4cls_pt_spleen_ct_seg 4.1
Loading Model: spleen_seg
Loading spleen_seg from NGC: https://registry.nvidia.com/catalog/med4cls_pt_spleen_ct_seg
{"details": "AIAA Config Not Found/Provided"}
  
```

So, will be using. So, will be using spleen CT yeah then, what are we trying to do? We are trying to actually download the deep grow model from the NGC and load it into our AIAA annotation server right. So, we are loading this model, now it basically loads yeah.



(Refer Slide Time: 25:30)



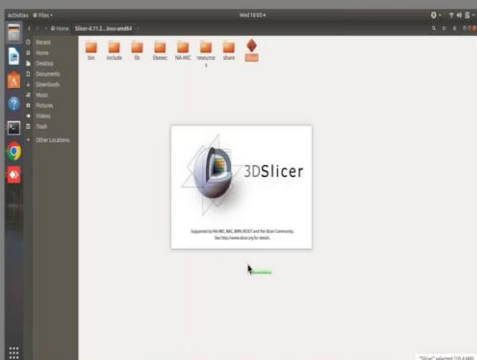
The screenshot shows a terminal window with the following content:

```
Loading Model: spleen_seg
Loading spleen_seg from NCC: https://ngc.nvidia.com/catalog/model/nvidia/med-clara_dt_deepgrow_3d_annotation
"default": "NCC Config Not Found/Provided"
}
The following cell will download the spleen model and load it into the AAA server
[1]: !NCCM_load spleen_seg ncc "nvidia/med-clara_dt_spleen_seg_annotation" 4.1
}
Loading Model: spleen_seg
Loading spleen_seg from NCC: https://ngc.nvidia.com/catalog/model/nvidia/med-clara_dt_spleen_seg_annotation
"default": "NCC Config Not Found/Provided"
}

3.3 Manually Download Model from NCC
If you have a private registry or have early access (EA) to NVIDIA's developer program, you can manually download the MBAR
from NCC by clicking the "action" as shown below
!NCCM_load spleen_seg ncc "nvidia/med-clara_dt_deepgrow_3d_annotation"
clara_dt_deepgrow_3d_annotation
```





(Refer Slide Time: 25:38)

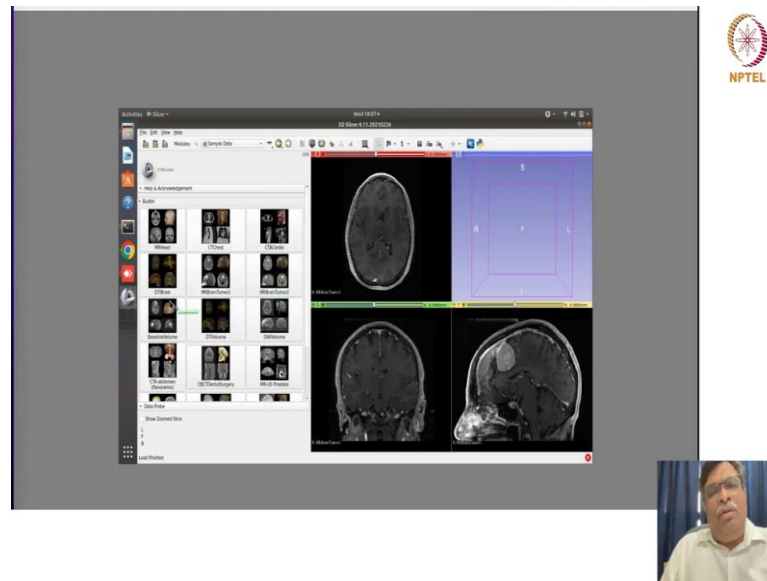


The screenshot shows a desktop environment with the following content:

3DSlicer

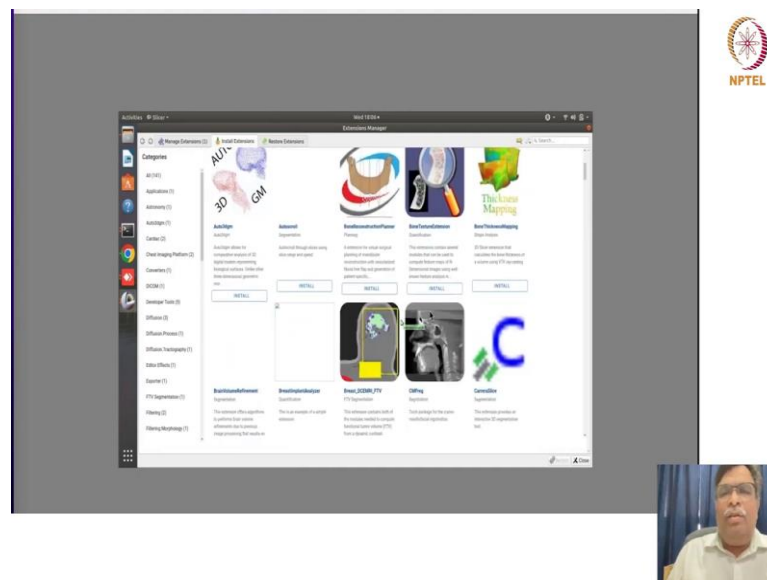


(Refer Slide Time: 25:46)



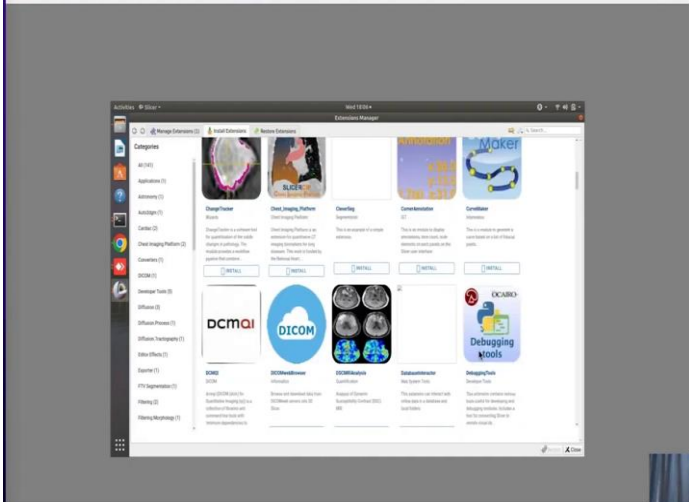
So, we will show you that the slicer. So, there is a 3D slicer which you would be using to see light know all of this. So, you have got this viewer which is a 3D slicer viewer which you would be using ok.

(Refer Slide Time: 26:01)



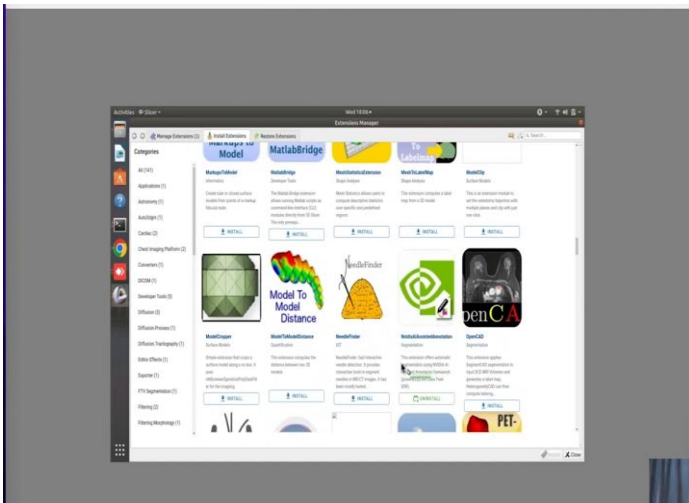
And then there are so many extensions which are available in this 3D slicer actually. So, you are trying to work with volumetric analysis. So, you basically would be requiring a 3D slicer.

(Refer Slide Time: 26:15)



The screenshot displays the MATLAB Education Manager window. On the left, there is a sidebar with a list of categories such as 'All (1)', 'Applications (1)', 'Accessibility (1)', 'CAD (2)', 'Cloud Imaging Patterns (2)', 'Connectivity (1)', 'DCM (1)', 'Developer Tools (3)', 'EPI (1)', 'EPI-Processing (1)', 'Image Effects (1)', 'Image (1)', 'Image Segmentation (1)', 'Imaging (1)', and 'Imaging Workflows (1)'. The main area shows a grid of plugin cards. Visible cards include 'ChargePacker', 'DICOM', 'DICOMViewer', 'DICOMWorkshop', 'Debugging Tools', 'DCM', 'DICOMViewer', 'DICOMWorkshop', and 'Debugging Tools'. Each card features a thumbnail image and a brief description of the plugin's functionality. The NPTEL logo is visible in the top right corner of the slide.

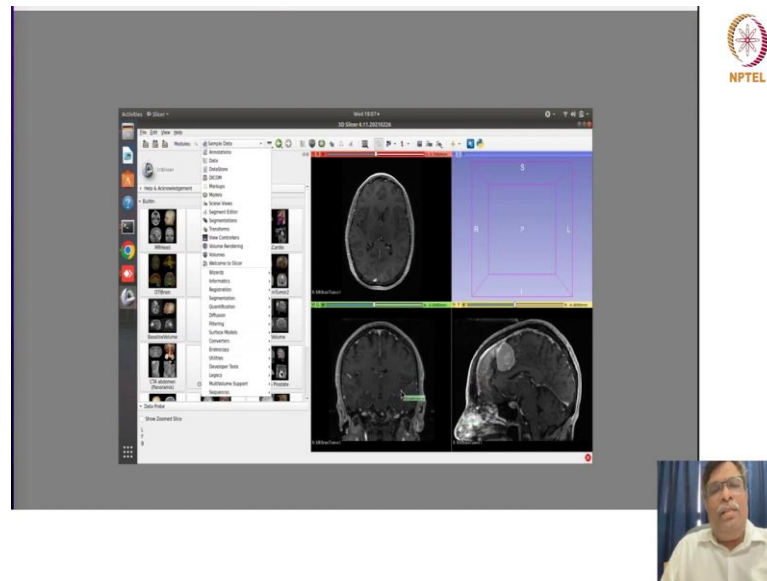
(Refer Slide Time: 26:20)



The screenshot displays the MATLAB Education Manager window with a different set of plugins. The sidebar categories are the same as in the previous slide. The main area shows a grid of plugin cards. Visible cards include 'Model', 'MatlabBridge', 'Model To Model Distance', 'NvidiaBridge', 'OpenCA', 'Model To Model Distance', 'NvidiaBridge', 'OpenCA', and 'PET'. Each card features a thumbnail image and a brief description of the plugin's functionality. The NPTEL logo is visible in the top right corner of the slide.

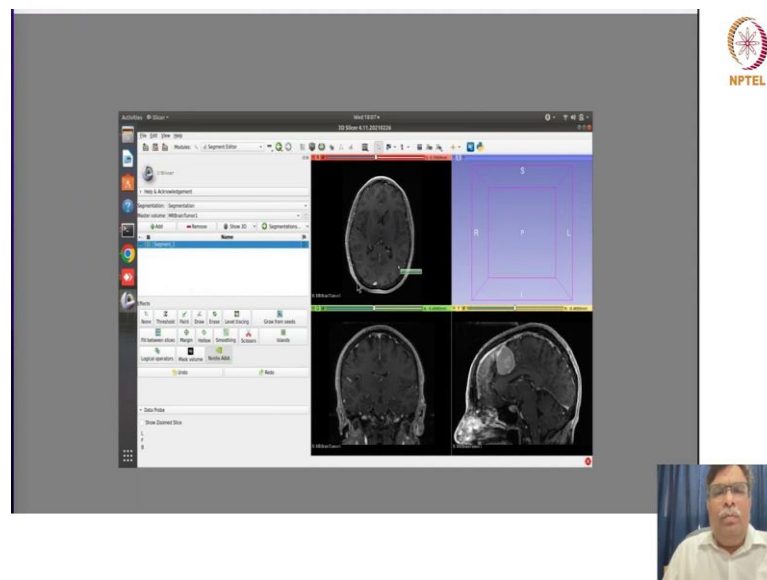
And there are so many plugins which you can use right. So, you have DICOM plugin here. Similarly, you have this NVIDIA AI assisted annotation segmentation plug in right. So, we will have to actually install this also yeah.

(Refer Slide Time: 26:49)



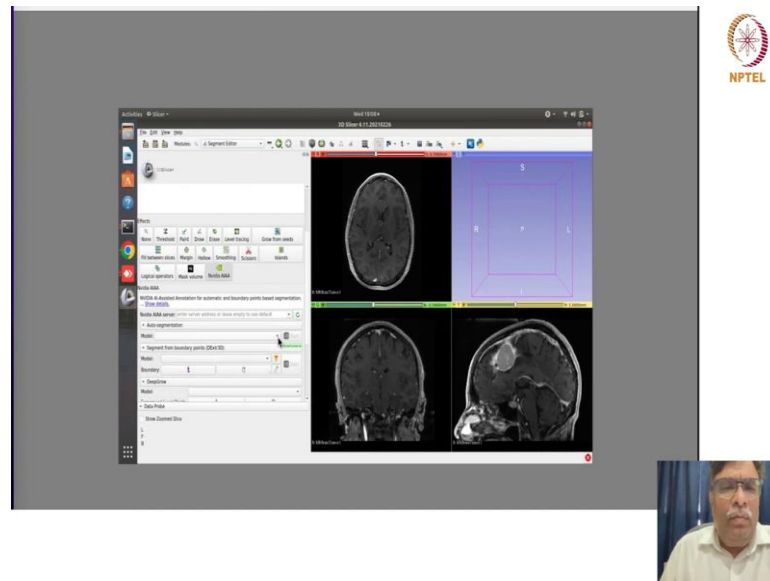
So, if you see this these are built in models which you are having right MR head, CT chest, CT cardio this that everything and then these are your various views right in the 3D slicer.

(Refer Slide Time: 26:57)



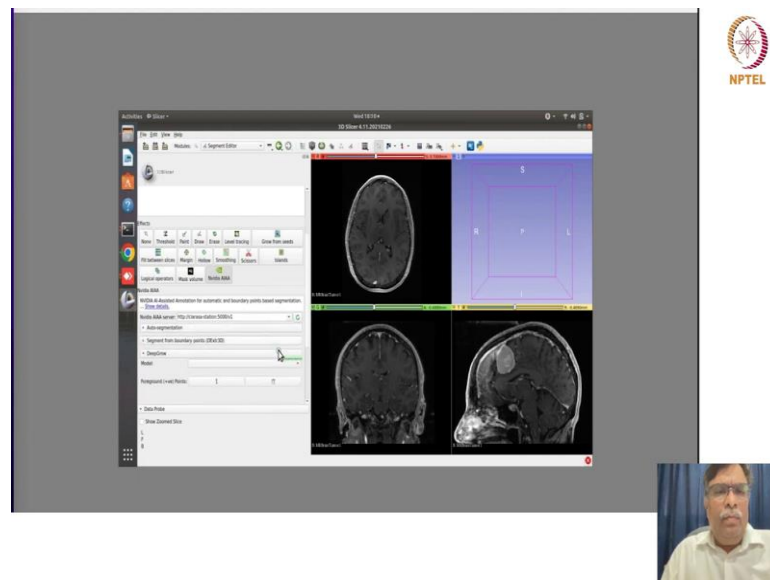
So, yeah. So, you have the segment editor which you can use to segment manually yeah.

(Refer Slide Time: 27:33)



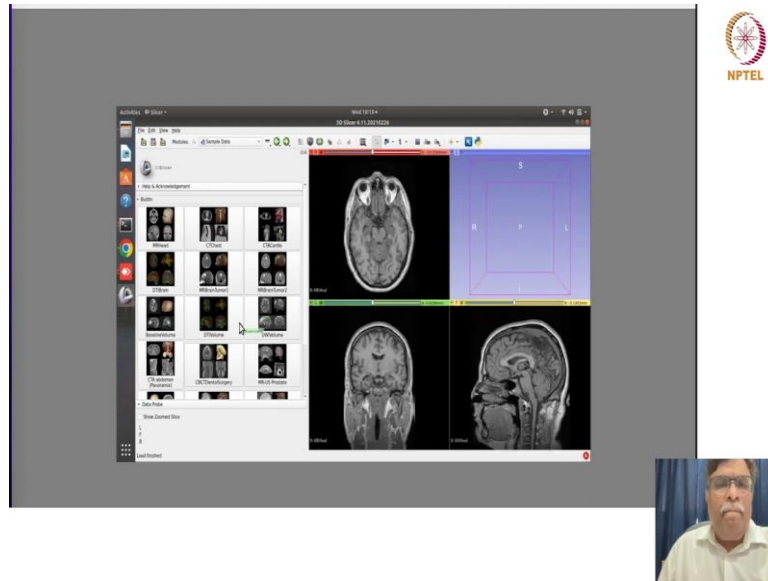
So, we will show you that as well and here you have got a lot of things, you have got mass, volume, you have got NVIDIA AIAA effect right see here; struck again sorry we have to, we have to pull in the model, got struck again, 1 minute. It got stuck again I suppose.

(Refer Slide Time: 29:01)

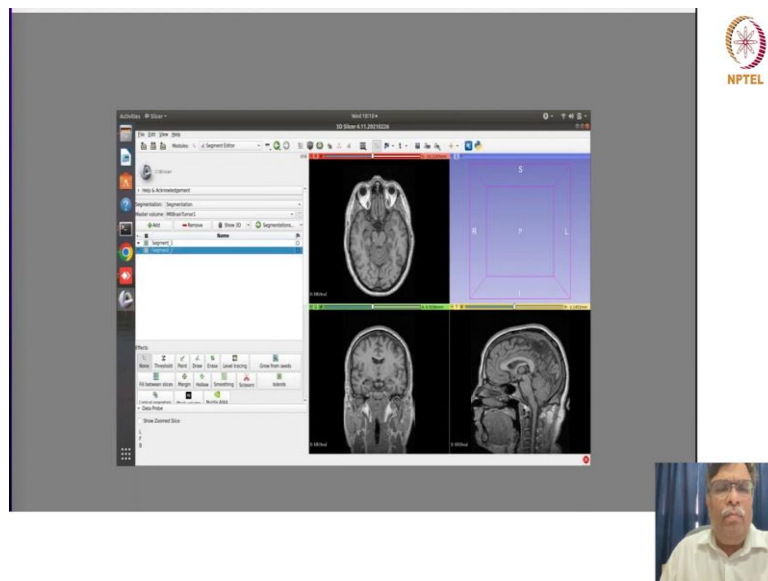


There is an option of selecting the server, we have to select this. I do not know what, yeah 1 minute AIAA server not getting selected with this remote, yeah.

(Refer Slide Time: 29:56)

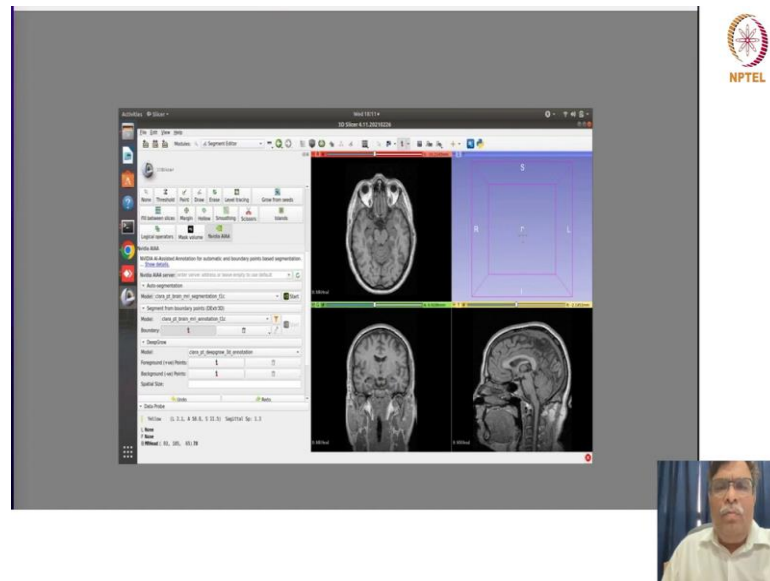


(Refer Slide Time: 30:10)



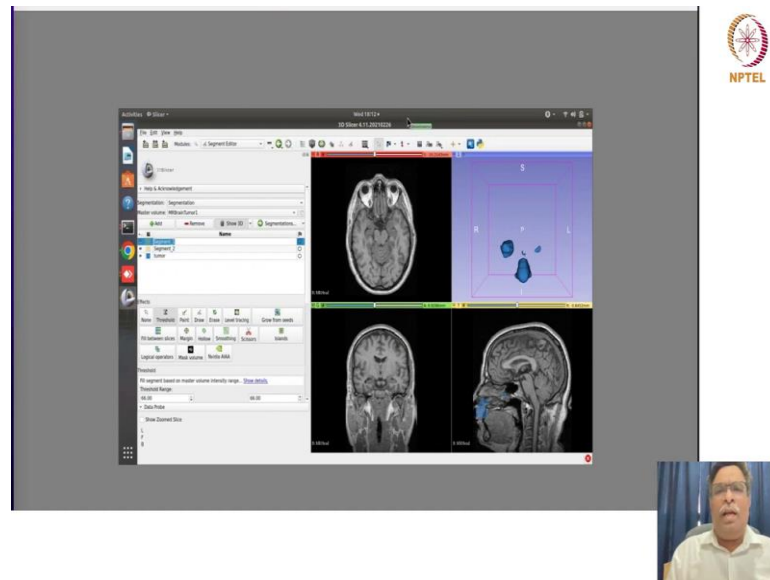
So, you can use some model and do segmentation, then you can generate some segments, I will try to show you that, yes.

(Refer Slide Time: 30:31)



So, it will I will first do auto segmentation using that server point and then we can do that annotation. All things which are available you have to auto select things yeah. So, you have to click some three boundary points here then, you can do the auto segmentation. So, we are uploading that volume to the server.

(Refer Slide Time: 31:38)



So, these are the areas which are detected with tumor can be selected right. So, this is how basically whatever you are trying to do is trying to basically be used for auto segmentation and automatic boundary point based segmentation and something and so

forth right. We just wanted to give a fair idea of how this type of a software could be used right this is just to show you that there are a lot of things available which you can use it for your research and projects yeah.

So, just the facility right, how accurate and how this thing is there we just wanted to show you in brief about all of this right. So, yeah. So, many facilities are available that is what we thought we will show it to you right. We are not going into each and every step very very accurately and correctly so that you come up with full automated automation or something like that, but we just wanted to tell you all of these facilities are available right. So, I suppose I am done.