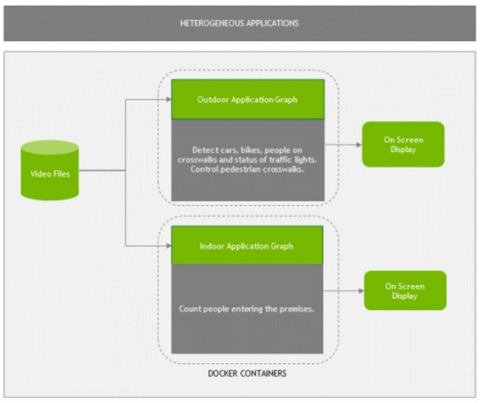
author:	Daniel Casota	to:	VMware Photon OS team
		cc:	

AI Inferencing Lab on Photon OS
with NVidia

Table of Contents

1 Introduction

Systems with more and more powerful GPUs are used for the desire to have additional information visualized in real time during video transmissions. These are referred to as AI inference systems.



HETEROGENEOUS APPLICATIONS
Video Files
Outdoor Application Graph
Detect cars, bikes, people on crosswalks and status of traffic lights. Control pedestrian crosswalks.
On Screen Display
Indoor Application Graph
Count people entering the premises.
On Screen Display
DOCKER CONTAINERS

Figure 1
Schema of an AI inference Systems[1]

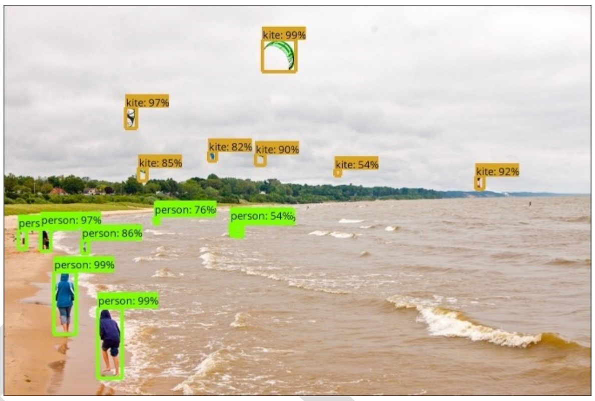


Figure 2 Example object detection[2] Outdoor

AI Artificial Intelligence describes how a series-ready object recognition, image classification, segmentation and many other properties are achieved from the aggregation of video streams and other sensors.

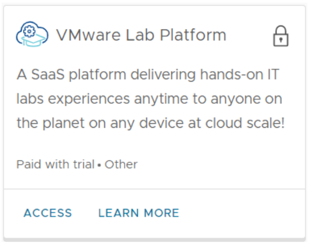
For this lab, the literature research on AI inference systems was limited exclusively to the so-called NVidia Triton Inference Server and NVidia examples materials. As the largest GPU maker, Nvidia has made significant hardware/software advances in AI.

The Triton Inference Server is part of the NVidia AI Enterprise Software Suite. It can be deployed both as a local system (edge) and as a hyperscaler system on AWS, Azure and Google Cloud. In both cases, an implementation based on docker containers was developed.

VMware Photon OS ist the preferred os for cloud-native applications. It runs on x86_64 + arm64 processors and on any hyperscaler cloud.
Out of this lab scope is a research about provisioning a docker swarm mesh e.g. hybrid and across multiple hyperscaler systems using VMware solutions.

In addition, it was not researched whether there are alternative products, e.g. for Intel or AMD GPUs.

The easiest way to setup a lab is using the VMware Lab platform. Go to https://console.cloud.vmware.com.



VMware Lab Platform

A SaaS platform delivering hands-on IT labs experiences anytime to anyone on the planet on any device at cloud scale!

Paid with trial • Other

ACCESS   LEARN MORE

This lab doc has been created in September 2022, using Photon OS on Azure as lab os.

[1] https://developer.nvidia.com/blog/intelligent-video-analytics-deepstream-sdk-3-0/
[2] Siehe https://github.com/tensorflow/models/tree/master/research/object_detection und https://github.com/mylesagray/tensorflow-anpr

The goal of AI inference systems is the lowest possible latency, real-time behavior, information diversity from object recognition and resilient operation.

The following Nvidia presentation of their AI Enterprise Software Suite includes the current status of achievements.

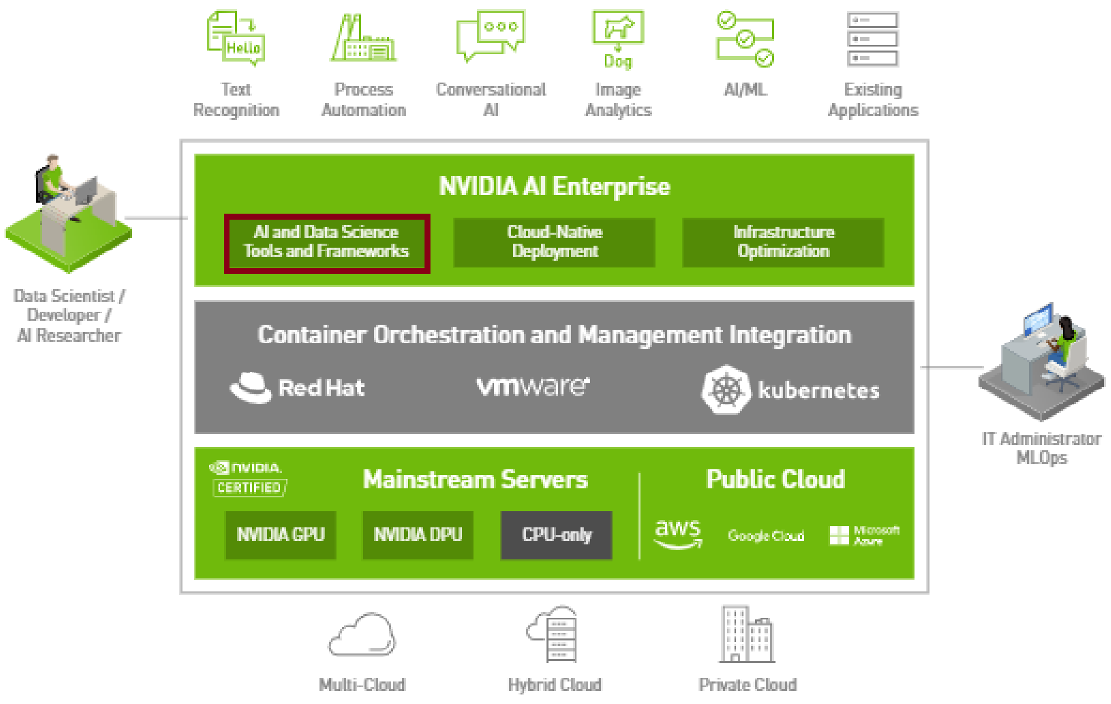


Figure 3 NVidia AI Enterprise[3]

AI achievements related to physical container solutions to support construction work in the rail network very important. How reliably and quickly are construction workers, other people, construction machinery, vehicles, objects, animals, etc. recognized during the day, at dusk and at night? How well are human gestures recognized? How safe can an AI conversation be reached?

For such "training" it is therefore important to find out what is already included in "AI and Data Science Tools and Frameworks" and how quickly and how easily requirements can be met with them.

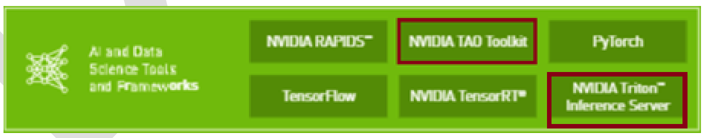

AI and Data Science Tools and Frameworks
NVIDIA RAPIDS™
NVIDIA TAO Toolkit
PyTorch
TensorFlow
NVIDIA TensorRT™
NVIDIA Triton™ Inference Server

Figure 4 NVidia AI Enterprise AI and Data Science Tools and Frameworks

The NVidia Triton Inference Server corresponds to the already mentioned AI Inference Service from figure Figure 1
Schema of an AI inference Systems. The so-called NVidia TAO Toolkit is designed for simplified training.

In the following sub-chapters, these and other components are briefly and concisely fanned out.

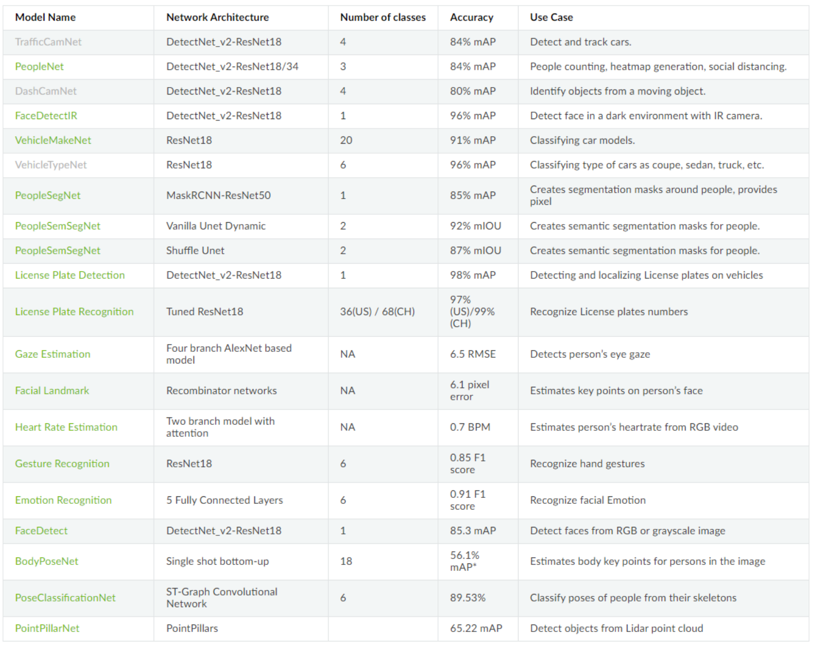
[3] https://resources.nvidia.com/en-us-ai-enterprise/en-us-nvidia-ai-enterprise/nvaie-overview?lb-mode=preview

1.1 NVidia TAO

How easily can a vehicle be recognized as a vehicle in an image and visualized by means of a rectangular border, and reused as code for other images? The NVidia Train - Adapt - Optimize (TAO) Framework is available to answer such questions. The NVidia TAO includes a variety of pre-trained models for later AI applications in the areas of speech and machine vision.
Specifically, there are two TAO versions.

The toolkit flavor, TAO Toolkit, is a low-code framework that enables enterprise application developers to optimize pre-trained NVidia models with custom data to build sufficient computer vision, speech, and language understanding models in hours instead of months , which should make extensive training unnecessary.
• According to the description, there is a no-code, GUI-based variant that combines the TAO Toolkit and other NVidia technologies in a single user interface. It is currently still under development.

Models can be downloaded individually or as a whole collection from the NVidia GPU Cloud (NGC) Marketplace https://catalog.ngc.nvidia.com.
The computer vision collection called "Computer Vision" includes a collection (="zoo") of pre-trained models. There are currently twenty models.



Model Name	Network Architecture	Number of classes	Accuracy	Use Case
TrafficCamNet	DetectNet_v2-ResNet18	4	84% mAP	Detect and track cars.
PeopleNet	DetectNet_v2-ResNet18/34	3	84% mAP	People counting, heatmap generation, social distancing.
DashCamNet	DetectNet_v2-ResNet18	4	80% mAP	Identify objects from a moving object.
FaceDetectIR	DetectNet_v2-ResNet18	1	96% mAP	Detect face in a dark environment with IR camera.
VehicleMakeNet	ResNet18	20	91% mAP	Classifying car models.
VehicleTypeNet	ResNet18	6	96% mAP	Classifying type of cars as coupe, sedan, truck, etc.
PeopleSegNet	MaskRCNN-ResNet50	1	85% mAP	Creates segmentation masks around people, provides pixel
PeopleSemSegNet	Vanilla Unet Dynamic	2	92% mIOU	Creates semantic segmentation masks for people.
PeopleSemSegNet	Shuffle Unet	2	87% mIOU	Creates semantic segmentation masks for people.
License Plate Detection	DetectNet_v2-ResNet18	1	98% mAP	Detecting and localizing License plates on vehicles
License Plate Recognition	Tuned ResNet18	36(US) / 68(CH)	97% (US)/99% (CH)	Recognize License plates numbers
Gaze Estimation	Four branch AlexNet based model	NA	6.5 RMSE	Detects person's eye gaze
Facial Landmark	Recombinator networks	NA	6.1 pixel error	Estimates key points on person's face
Heart Rate Estimation	Two branch model with attention	NA	0.7 BPM	Estimates person's heartrate from RGB video
Gesture Recognition	ResNet18	6	0.85 F1 score	Recognize hand gestures
Emotion Recognition	5 Fully Connected Layers	6	0.91 F1 score	Recognize facial Emotion
FaceDetect	DetectNet_v2-ResNet18	1	85.3 mAP	Detect faces from RGB or grayscale image
BodyPoseNet	Single shot bottom-up	18	56.1% mAP*	Estimates body key points for persons in the image
PoseClassificationNet	ST-Graph Convolutional Network	6	89.53%	Classify poses of people from their skeletons
PointPillarNet	PointPillars		65.22 mAP	Detect objects from Lidar point cloud

Figure 5 NVidia AI Models

If none of the pre-trained models can be used for target training from videos, images and audio, the effort is enormous. The scientific "creation of AI models" is the demarcation for another project work.

If an existing model can be adapted to your own needs, the TAO Toolkit has been a good choice. In the NVidia literature, a distinction is made between the phase of training a model and efficient provisioning for real-time evaluation from a (live) video stream.

Useful web urls:
https://github.com/NVIDIA-AI-IOT/TAO-Toolkit-Whitepaper-use-cases
https://github.com/NVIDIA-AI-IOT/tao_toolkit_recipes

Each model is shipped "Unpruned" for the training phase and "Pruned" for efficient provisioning.

The "unpruned" model is intended for training with the TAO Toolkit and the user's own data set. This allows models to be provided that are adapted to the application. This also requires the source, the so-called Jupyter notebook. The Jupyter notebook available as part of the TAO container can be used for retraining.

Pruning removes parameters from the model to reduce model size without affecting the integrity of the model itself. The "pruned" i.e. cropped model is usually much more powerful than the unpruned model for deployment at the edge for real-time evaluation from a (live) video stream.

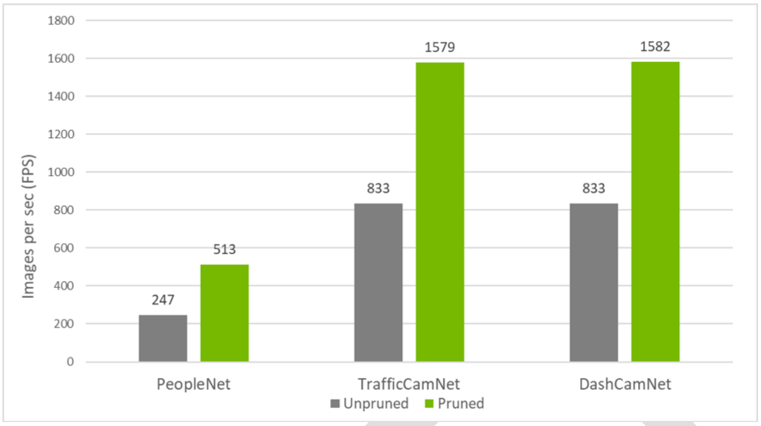


Figure 6 Comparison unpruned / pruned performance for various NVidia TAO Models

In addition to the pruning meccano, there is NVidia TensorRT, a programmable inference accelerator that can achieve significantly better real-time results when orchestrating multiple models for HGPU hardware.

1.1.1 The DashCamNet Model

The DashCamNet model is one of the twenty pre-trained computer vision models and could be a good starting point for the customer project. The model recognizes cars, people, road signals and bicycles.

It is good to know in advance that the model has some limitations. The DashcamNet was conditioned with good light source conditions - it is therefore still unclear how well it works in the twilight and at night. The model cannot recognize objects smaller than 20x20 pixels and at least 40% of a vehicle must be recognizable as such. The signage recognition only works for the USA.

There is only a single resnet18_dashcamnet.tlt file on the download site https://catalog.ngc.nvidia.com/orgs/nvidia/teams/tao/models/dashcamnet/files?version=unpruned_v1.0 .

Resnet18 refers to the underlying neural network that has classified images into object categories based on algorithms from the ImageNet database. It is also referred to as DetectNet_V2 in NVidia literature. The number 18 reflects the number of deep learning layers. See also https://ch.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html and https://frankdenneman.nl/2022/06/30/machine-learning-on-vmware- platform-part-3-training-versus-inference/ .

The .tlt extension stands for "Transfer Learning Toolkit" and was the original name for the NVidia TAO Toolkit.

The associated Jupyer notebook can be found at https://catalog.ngc.nvidia.com/orgs/nvidia/teams/tao/resources/cv_samples/files .

1.2 NVidia Triton Inference Server

The Triton Inference Server is open-source software that optimizes AI inference. With Triton, an AI model can be provided from frameworks for deep learning and machine learning. Triton supports inference across cloud, data center, edge, and embedded devices on NVIDIA GPUs, x86, and ARM CPUs. Triton offers optimized performance for many query types, including real-time, batch, ensembles, and audio/video streaming.

Useful web urls:
https://github.com/triton-inference-server
https://github.com/NVIDIA-AI-IOT/deepstream_tao_apps

https://github.com/NVIDIA-AI-IOT/tao-toolkit-triton-apps
https://github.com/NVIDIA-AI-IOT/deepstream_triton_migration
https://developer.nvidia.com/nvidia-triton-inference-server
https://developer.nvidia.com/deepstream-sdk
Deepstream samples: https://docs.nvidia.com/metropolis/deepstream/dev-guide/text/DS_ReadmeFirst.html
Intelligent Video Analytics: https://forums.developer.nvidia.com/c/accelerated-computing/intelligent-video-analytics/deepstream-sdk/15
https://github.com/sachinsharma9780/Build-ML-pipelines-for-Computer-Vision-NLP-and-Graph-Neural-Networks-using-Nvidia-Triton-Server

1.2.1 NVidia Deepstream

DeepStream is part of the Triton Inference Server functionality and provides cross-platform, scalable, TLS-encrypted security for computer vision-optimized AI applications that can be deployed on-premises, at the edge and in the cloud.

The technical article "NVidia DeepStream and Triton integration" clearly illustrates the inner, so-called Deep-Stream functions of the Inference Server. This should suffice as a brief and concise reference to Deep-Stream.

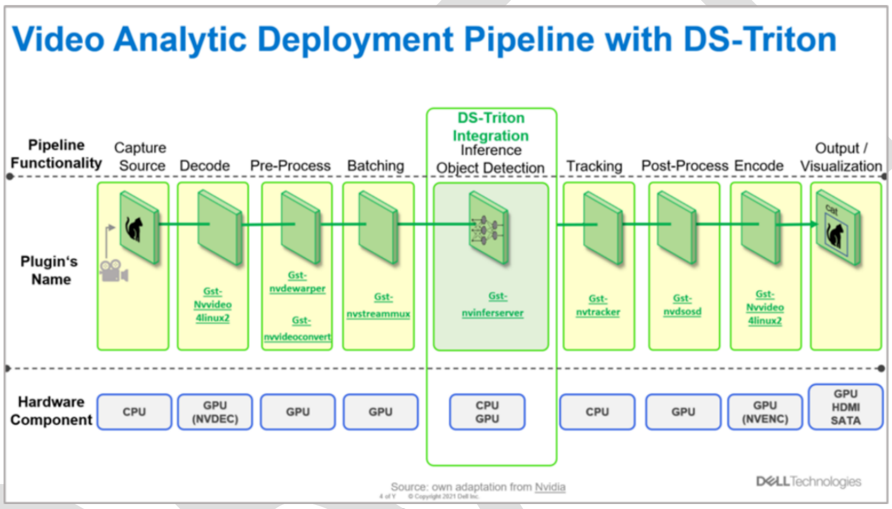
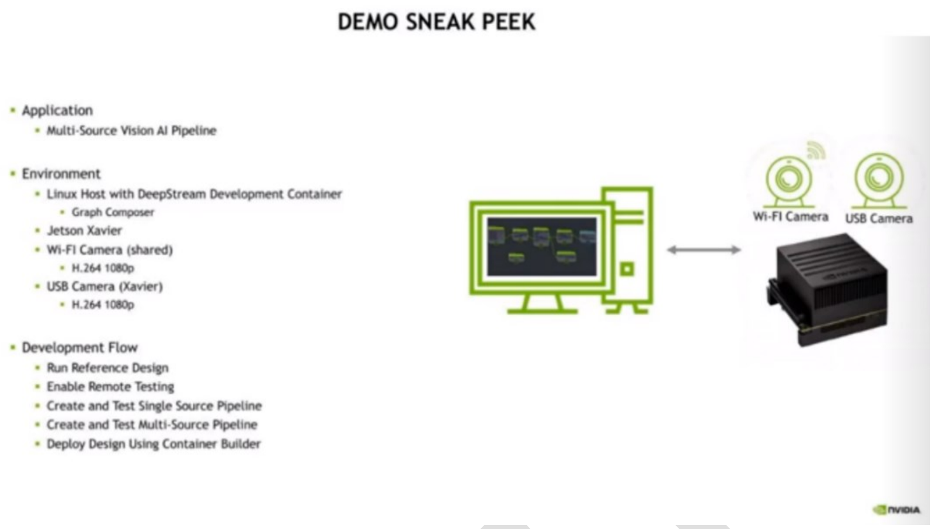


Figure 7 NVidia DeepStream with Triton integration

The Deepstream SDK provides examples for different scenarios.
For example, a sample app for Azure IoT Edge is described in the Deepstream SDK Release Notes.

1.3 NVidia Graph Composer

The NVidia Graph Composer enables system engineers to build real-time computer vision pipelines with DeepStream plugins and deploy them using containers - all without writing a single line of code. Graph Composer is therefore suitable for the process of developing, testing and deploying an AI-supported video stream system.
The content of the NVidia GTC session "Deep Dive into Jetson and Deep-Stream" serves as a development plan aid.

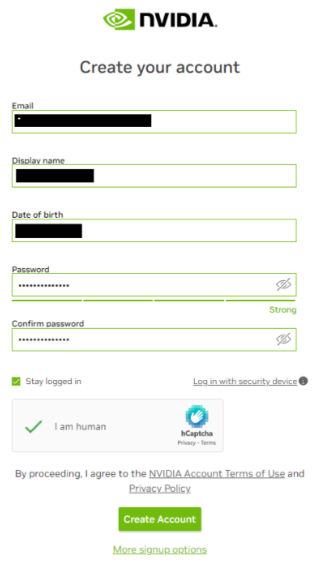

DEMO SNEAK PEEK

- Application
  - Multi-Source Vision AI Pipeline
- Environment
  - Linux Host with DeepStream Development Container
    - Graph Composer
  - Jetson Xavier
  - Wi-Fi Camera (shared)
    - H.264 1080p
  - USB Camera (Xavier)
    - H.264 1080p
- Development Flow
  - Run Reference Design
  - Enable Remote Testing
  - Create and Test Single Source Pipeline
  - Create and Test Multi-Source Pipeline
  - Deploy Design Using Container Builder

1.4 NVidia NGC Account

Some lab operations require an NVidia NGC user account.

The NVidia NGC user account can be set up free of charge at https://ngc.nvidia.com/signin. The first time, an email address must be entered in order to access the "Create your Account" website.


NVIDIA

Create your account

Email: [redacted]

Display name: [redacted]

Date of birth: [redacted]

Password: ••••••••••••••

Strong

Confirm password: ••••••••••••••

✓ Stay logged in   Log in with security device

✓ I am human   hCaptcha Privacy - Terms

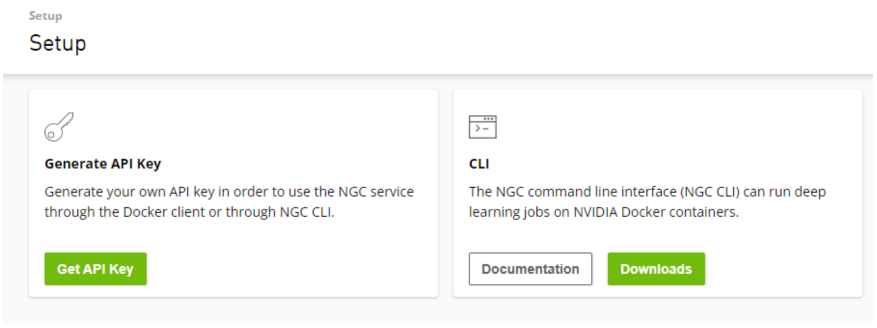
By proceeding, I agree to the NVIDIA Account Terms of Use and Privacy Policy

Create Account

More signup options

Figure 8 NVidia NGC user account

As soon as the user account is set up, a so-called API key can be generated. Here you have to log in with the user account on https://ngc.nvidia.com/setup .


Setup

Setup

Generate API Key

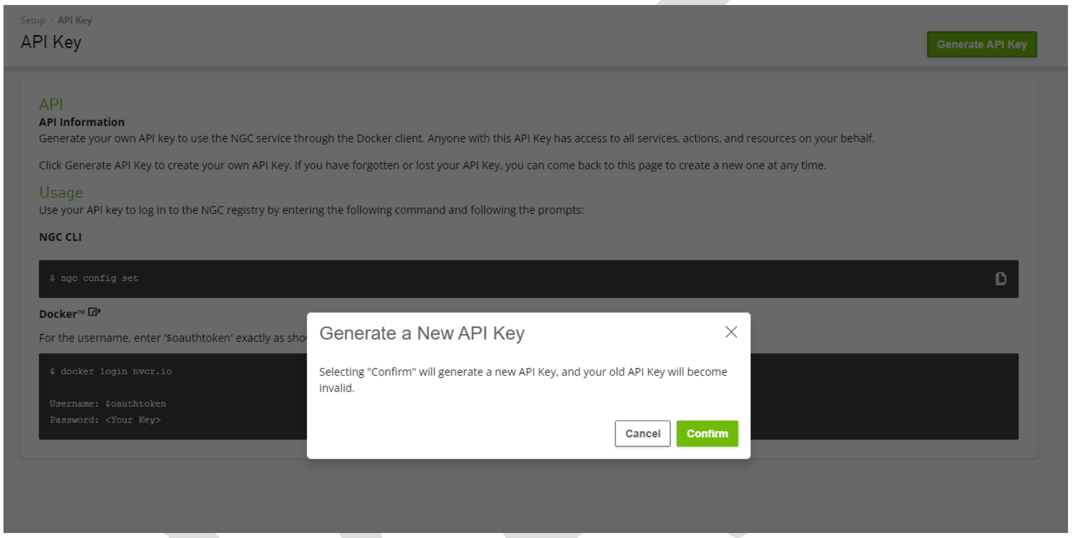
Generate your own API key in order to use the NGC service through the Docker client or through NGC CLI.

CLI

The NGC command line interface (NGC CLI) can run deep learning jobs on NVIDIA Docker containers.

Figure 9 NVidia NGC setup

Click Get API Key.


Setup > API Key

API Key

API

API Information

Generate your own API key to use the NGC service through the Docker client. Anyone with this API Key has access to all services, actions, and resources on your behalf.

Click Generate API Key to create your own API Key. If you have forgotten or lost your API Key, you can come back to this page to create a new one at any time.

Usage

Use your API key to log in to the NGC registry by entering the following command and following the prompts:

NGC CLI

```
$ ngc config set
```

Docker™

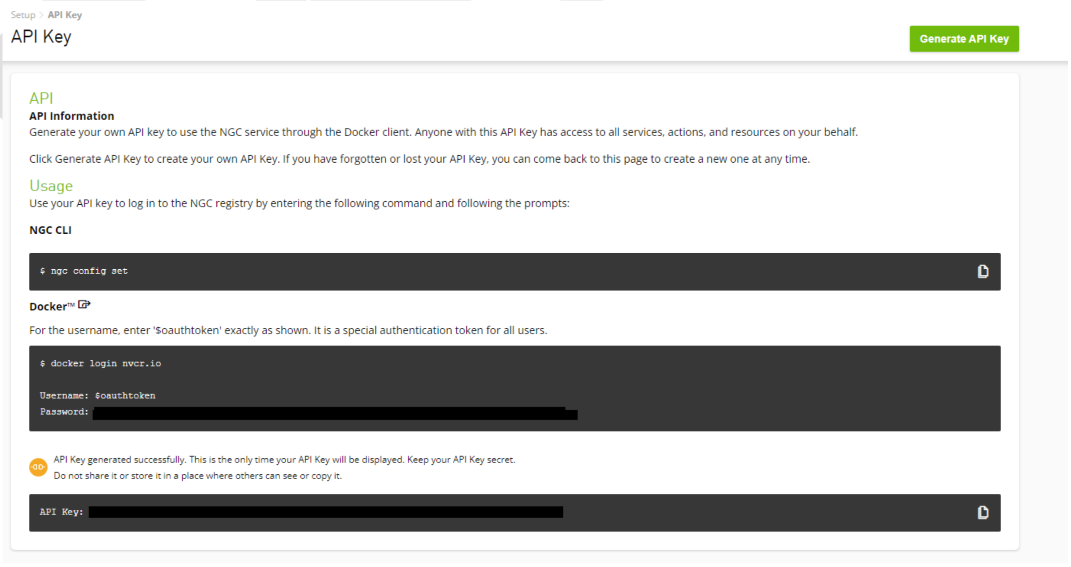
For the username, enter '$oauthtoken' exactly as sho

```
$ docker login nvcr.io

Username: $oauthtoken
Password: <Your Key>
```

Generate a New API Key

Selecting "Confirm" will generate a new API Key, and your old API Key will become invalid.

Click Generate API Key, and then Confirm.


Setup > API Key

API Key

API

API Information

Generate your own API key to use the NGC service through the Docker client. Anyone with this API Key has access to all services, actions, and resources on your behalf.

Click Generate API Key to create your own API Key. If you have forgotten or lost your API Key, you can come back to this page to create a new one at any time.

Usage

Use your API key to log in to the NGC registry by entering the following command and following the prompts:

NGC CLI

```
$ ngc config set
```

Docker™

For the username, enter '$oauthtoken' exactly as shown. It is a special authentication token for all users.

```
$ docker login nvcr.io

Username: $oauthtoken
Password: [redacted]
```

API Key generated successfully. This is the only time your API Key will be displayed. Keep your API Key secret. Do not share it or store it in a place where others can see or copy it.

```
API Key: [redacted]
```

Password and API Key can be regenerated on demand.

2 Lab Infrastruktur TAO Toolkit

The TAO Toolkit requires either hardware with NVidia GPU Ampere series or RTX, P, T, M series.

For the sake of simplicity, because an Azure subscription already exists, the TAO Toolkit is installed on an Azure VM. This may change and what exactly has to be set up in the Azure VM can vary. See also https://docs.nvidia.com/tao/tao-toolkit/text/running_in_cloud/running_tao_toolkit_on_azure.html .

2.1 Azure VM

The suggested Azure offer[4] is not used for an initial setup. A more minimal structure is sufficient. Accordingly to https://azureprice.net/ the offer[5] https://azureprice.net/vm/Standard_NV6 in East US2 Region with 1.0994 CHF per hour is actually the cheapiest, and contains a NVidia Tesla M60 GPU. It is important to know the details and limitations of the NV-series https://docs.microsoft.com/en-us/azure/virtual-machines/nv-series.

VMware Photon OS is installed in the Azure VM as the operating system. Although it is not one of the Linux distributions preferred by Azure, as a basic OS it is much slimmer than Ubuntu, and is already prepared for a zero trust fabric and enables simple Kubernetes hybrid cloud setups.

A VMware Photon OS Azure image was created for the installation and a virtual machine was instantiated. To do this, first download the two Powershell scripts published there under Use Case 3 from https://github.com/dcasota/photonos-scripts and place them on a windows laptop, for example.

Syntax to create a VMware Photon OS Azure image:

```
./create-AzImage-PhotonOS.ps1 -DownloadURL
hhttps://packages.vmware.com/photon/3.0/Rev2/azure/photon-azure-3.0-9355405.vhd.tar.gz
-ResourceGroupName PhotonOSTemplates -LocationName eastus2 -HyperVGeneration V1
```

Syntax to instantiate a virtual machine:

```
./create-AzVM_FromImage-PhotonOS.ps1 -Location eastus2
-ResourceGroupNameImage PhotonOSTemplates -ImageName photon-azure-3.0-9355405_V1.vhd
-ResourceGroupName NVidia -VMSize Standard_NV6 -VMName NVidia01
-VMLocalAdminCredential $(Get-credential -message 'Specify a Photon OS local admin username and password. Pass-
word must be 12-23 chars long.')
```

The partition should be large enough for later storage of files, e.g. 64GB. Unfortunately, this cannot be parameterized at the time of provisioning and must therefore be configured afterwards.

Azure Powershell Code:

```
# Source https://docs.ukcloud.com/articles/azure/azs-how-resize-disk-ps.html?tabs=tabid-1%2Ctabid-
a%2Ctabid-c
# Set your resource group and VM name
$Resourcegroupname = "NVidia"
$VMName = "NVidia01"

# Obtain your VM object
$VM = Get-AzVM -ResourceGroupName $Resourcegroupname -Name $VMName

# Stop the VM before resizing the disk
Stop-AzVM -ResourceGroupName $Resourcegroupname -Name $VMName -Force

# Resize managed OS disk
$Disk = Get-AzDisk -ResourceGroupName $Resourcegroupname -DiskName $VM.StorageProfile.OsDisk.Name
$Disk.DiskSizeGB = 64
Update-AzDisk -ResourceGroupName $Resourcegroupname -Disk $Disk -DiskName $Disk.Name

# Start the VM
Start-AzVM -ResourceGroupName $Resourcegroupname -Name $VMName
```

[4] https://azuremarketplace.microsoft.com/en-us/marketplace/apps/nvidia.ngc_azure_17_11?tab=PlansAndPrice
[5] Hierfür müssen u.U. die Regional-Quota https://docs.microsoft.com/en-us/azure/quotas/regional-quota-requests und die Per-VM-Quota https://docs.microsoft.com/en-us/azure/azure-supportability/per-vm-quota-requests angehoben werden.

For the sake of simplicity for the Jupyter Notebook later, port 8888 is opened in the Azure VM.

```
# network security rules configuration
$nsgname = "<ID>nsg"
$ResourcegroupName = "<resourcegroup>"
$LocationName="<location>"

$nsg=get-AzNetworkSecurityGroup -Name $nsgName -ResourceGroupName $ResourceGroupName -ErrorAction Si-
lentlyContinue
if ( -not $($nsg))
{
	$nsgRule1 = New-AzNetworkSecurityRuleConfig -Name nsgRule2 -Description "Allow 8888" `
	-Access Allow -Protocol Tcp -Direction Inbound -Priority 120 `
	-SourceAddressPrefix Internet -SourcePortRange * `
	-DestinationAddressPrefix * -DestinationPortRange 8888
	$nsg = New-AzNetworkSecurityGroup -Name $nsgName -ResourceGroupName $ResourceGroupName -Loca-
tion $LocationName -SecurityRules $nsgRule1
}
```

Photon OS Code:

```
# https://vmware.github.io/photon/docs/troubleshooting-guide/file-system-troubleshooting/expanding-
disk-partition/
# Mit fdisk -l prüfen um welche disk es sich handelt, z.B. /dev/sdb
echo 1 > /sys/class/block/sdb/device/rescan
tdnf install -y parted
parted --script /dev/sdb "resizepart 2 -1" quit
resize2fs /dev/sdb2
```

2.2 Initial configuration

It is a lab setup. In a second step, the installation must be checked without root rights.

In the lab base image, ssh access is allowed via username+password.

```
# sshd configuration
iptables -A INPUT -i eth0 -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -p tcp --dport 22 -j ACCEPT
sed "s/PasswordAuthentication no/PasswordAuthentication yes/" /etc/ssh/sshd_config >
/etc/ssh/sshd_config.new
cp /etc/ssh/sshd_config.new /etc/ssh/sshd_config
systemctl restart sshd
rm /etc/ssh/sshd_config.new
```

To check which NVidia graphics card has been detected, use `lspci`.

```
tdnf install -y pciutils
lspci
```

lspci-Output

```
# lspci
0000:00:00.0 Host bridge: Intel Corporation 440BX/ZX/DX - 82443BX/ZX/DX Host bridge (AGP disabled)
(rev 03)
0000:00:07.0 ISA bridge: Intel Corporation 82371AB/EB/MB PIIX4 ISA (rev 01)
0000:00:07.1 IDE interface: Intel Corporation 82371AB/EB/MB PIIX4 IDE (rev 01)
0000:00:07.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 02)
0000:00:08.0 VGA compatible controller: Microsoft Corporation Hyper-V virtual VGA
0001:00:00.0 VGA compatible controller: NVIDIA Corporation GM204GL [Tesla M60] (rev a1)
```

2.2.1 NVidia GPU drivers

Install the NVidia GPU[6] drivers.

```
# repo url
if [ `cat /etc/yum.repos.d/photon.repo | grep -o "packages.vmware.com/photon" | wc -l` -eq 0 ]; then
       cd /etc/yum.repos.d/
       sudo sed -i 's/dl.bintray.com\/vmware/packages.vmware.com\/photon\/$releasever/g' photon.repo
photon-updates.repo photon-extras.repo photon-debuginfo.repo
fi

# update components with impact to nvidia components
tdnf update -y docker

# install kernel api headers and devel
tdnf install -y build-essential wget tar
tdnf install -y linux-devel

reboot

wget https://us.download.nvidia.com/tesla/470.141.03/NVIDIA-Linux-x86_64-470.141.03.run
chmod a+x ./NVIDIA-Linux-x86_64-470.141.03.run
./NVIDIA-Linux-x86_64-470.141.03.run --kernel-source-path=/usr/lib/modules/`uname -r`/build --ui=none
--no-questions --accept-license



# Check if the graphics card was detected, sample output below
nvidia-smi
root@NVidia01 [ / ]# nvidia-smi
Wed Aug 24 07:02:55 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.141.03   Driver Version: 470.141.03   CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla M60           Off  | 00003130:00:00.0 Off |                  Off |
| N/A   34C    P0    36W / 150W |      0MiB /  8129MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

2.2.2 NVidia Container Toolkit

The NVidia Container Toolkit is a prerequisite to start the NVidia Docker Container.

```
# install NVidia Container toolkit to run NVidia docker container on Photon OS
tdnf install -y gpg
cd /etc/pki/rpm-gpg/
curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | gpg --dearmor -o /etc/pki/rpm-
gpg/nvidia-container-toolkit-keyring.gpg

cat << EOF >>/etc/yum.repos.d/nvidia-container-toolkit.repo
[libnvidia-container]
name=libnvidia-container
baseurl=https://nvidia.github.io/libnvidia-container/centos7/x86_64
gpgcheck=0
enabled=1
EOF
```

[6] https://github.com/vmware/photon/issues/1291

```
tdnf makecache
tdnf install nvidia-container-toolkit

systemctl restart docker

rm /etc/yum.repos.d/nvidia-container-toolkit.repo
```

2.2.3 NVidia Jupyterlab packages

Accordingly to the Nvidia description[7] a few packages are required for the jupyterlab functionality.

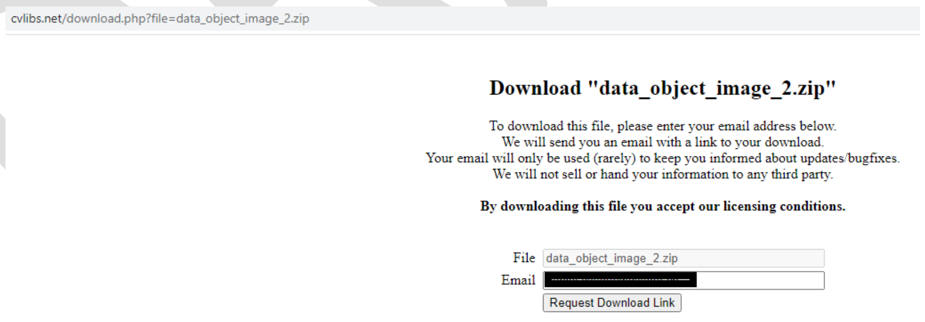
```
tdnf install -y wget unzip python3-pip
pip3 install virtualenvwrapper
export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
source /usr/bin/virtualenvwrapper.sh
mkvirtualenv -p /usr/bin/python3 launcher
pip3 install jupyterlab
pip3 install nvidia-tao

# NGC installation mit MD5Check-Ausgabe
wget --content-disposition https://ngc.nvidia.com/downloads/ngccli_linux.zip && unzip
ngccli_linux.zip && chmod u+x ngc-cli/ngc
find ngc-cli/ -type f -exec md5sum {} + | LC_ALL=C sort | md5sum -c ngc-cli.md5
echo "export PATH=\"\$PATH:$(pwd)/ngc-cli\"" >> ~/.bash_profile && source ~/.bash_profile

wget --content-disposition https://api.ngc.nvidia.com/v2/resources/nvidia/tao/cv_samples/ver-
sions/v1.2.0/zip -O    cv_samples_v1.2.0.zip
unzip -u cv_samples_v1.2.0.zip  -d ./cv_samples_v1.2.0 && cd ./cv_samples_v1.2.0
mkdir ./cv_samples_v1.2.0/detectnet_v2/data
```

For the examples in this chapter, two directories are pre-created.

```
mkdir -p /workspace/tao-experiments/detectnet_v2
mkdir -p /workspace/tao-experiments/data/training
```

The sample data listed in the Jupyter Notebook can be obtained from the links below:

- http://www.cvlibs.net/download.php?file=data_object_image_2.zip



cvlibs.net/download.php?file=data_object_image_2.zip

Download "data_object_image_2.zip"

To download this file, please enter your email address below.
We will send you an email with a link to your download.
Your email will only be used (rarely) to keep you informed about updates/bugfixes.
We will not sell or hand your information to any third party.

By downloading this file you accept our licensing conditions.

File data_object_image_2.zip
Email [redacted]
Request Download Link

  Note: The file is about 11.7GB in size.

- http://www.cvlibs.net/download.php?file=data_object_label_2.zip
  Note: The file is about 5.5MB in size.

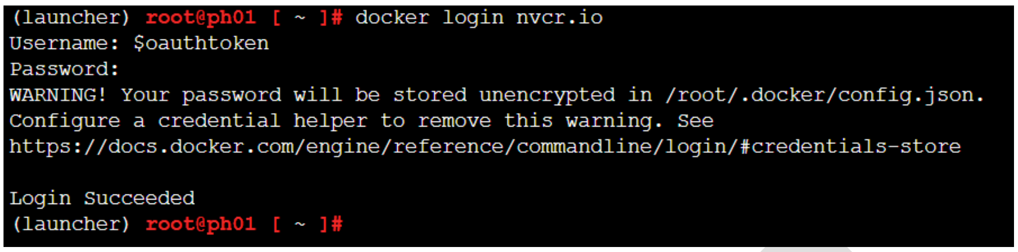
Copy the two zip files e.g. via WinSCP in the Azure VM in the directory
`./cv_samples_v1.2.0/detectnet_v2/data`

[7] https://docs.nvidia.com/tao/tao-toolkit/text/running_in_cloud/running_tao_toolkit_on_azure.html#setting-up-an-azure-vm

2.3 NVidia Public Registry

The next step is to ensure access to the NVidia Public Registry. NVidia hosts its own Public Docker Registry on nvcr.io. To log in, you need an NVidia NGC API key, see chapter *1.4 NVidia NGC Account*.

Login with `docker login nvcr.io` .



```
(launcher) root@ph01 [ ~ ]# docker login nvcr.io
Username: $oauthtoken
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
(launcher) root@ph01 [ ~ ]#
```

Start the docker daemon with `systemctl restart docker` if it isn't started already.

2.4 Jupyter Notebook

The jupyter notebook is published on port 8888. The port must be shared on both the Azure VM and Photon OS. Here is the code for Photon OS:

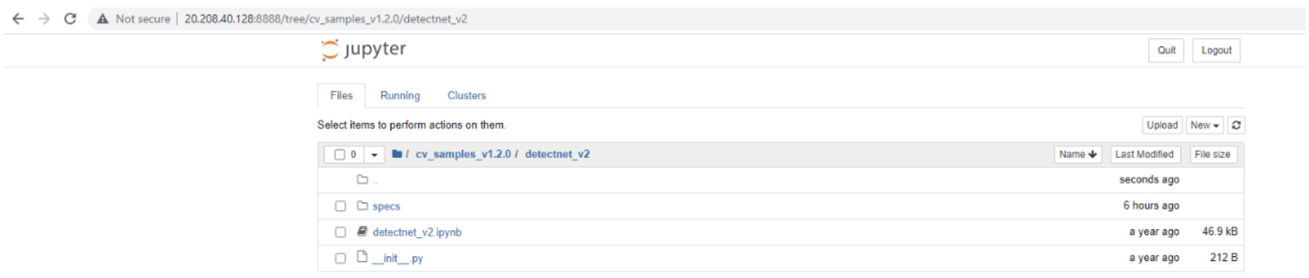
```
iptables -A INPUT -i eth0 -p tcp --dport 8888 -j ACCEPT
iptables -A OUTPUT -p tcp --dport 8888 -j ACCEPT
iptables-save >/etc/systemd/scripts/ip4save
```

Start the Jupyter Notebook.

```
Cd ./cv_samples_v1.2.0/
/root/.virtualenvs/launcher/bin/jupyter notebook --allow-root --ip 0.0.0.0 --no-browser

[W 2022-08-16 14:57:33.899 LabApp] 'ip' has moved from NotebookApp to ServerApp. This config will be
passed to ServerApp. Be sure to update your config before our next release.
[W 2022-08-16 14:57:33.906 LabApp] 'allow_root' has moved from NotebookApp to ServerApp. This config
will be passed to ServerApp. Be sure to update your config before our next release.
[W 2022-08-16 14:57:33.911 LabApp] 'allow_root' has moved from NotebookApp to ServerApp. This config
will be passed to ServerApp. Be sure to update your config before our next release.
[I 2022-08-16 14:57:33.925 LabApp] JupyterLab extension loaded from /root/.virtualenvs/laun-
cher/lib/python3.10/site-packages/jupyterlab
[I 2022-08-16 14:57:33.929 LabApp] JupyterLab application directory is /root/.virtualenvs/laun-
cher/share/jupyter/lab
[I 14:57:33.937 NotebookApp] Serving notebooks from local directory: /
[I 14:57:33.939 NotebookApp] Jupyter Notebook 6.4.12 is running at:
[I 14:57:33.942 NotebookApp] http://ph01:8888/?token=1471fe6c3147473cbb94f59b8071e1caccf80a655aeca50b
[I 14:57:33.945 NotebookApp]  or http://127.0.0.1:8888/?to-
ken=1471fe6c3147473cbb94f59b8071e1caccf80a655aeca50b
[I 14:57:33.948 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to
skip confirmation).

    To access the notebook, open this file in a browser:
        file:///root/.local/share/jupyter/runtime/nbserver-873-open.html
    Or copy and paste one of these URLs:
        http://ph01:8888/?token=1471fe6c3147473cbb94f59b8071e1caccf80a655aeca50b
     or http://127.0.0.1:8888/?token=1471fe6c3147473cbb94f59b8071e1caccf80a655aeca50b
```

Open the web browser and insert the public ip with the TokenID.
Beispiel: `http://20.208.40.1288888/?token=1471fe6c3147473cbb94f59b8071e1caccf80a655aeca50b`

Switch into directory CV_Samples_v1.2.0 > Detectnet_v2.

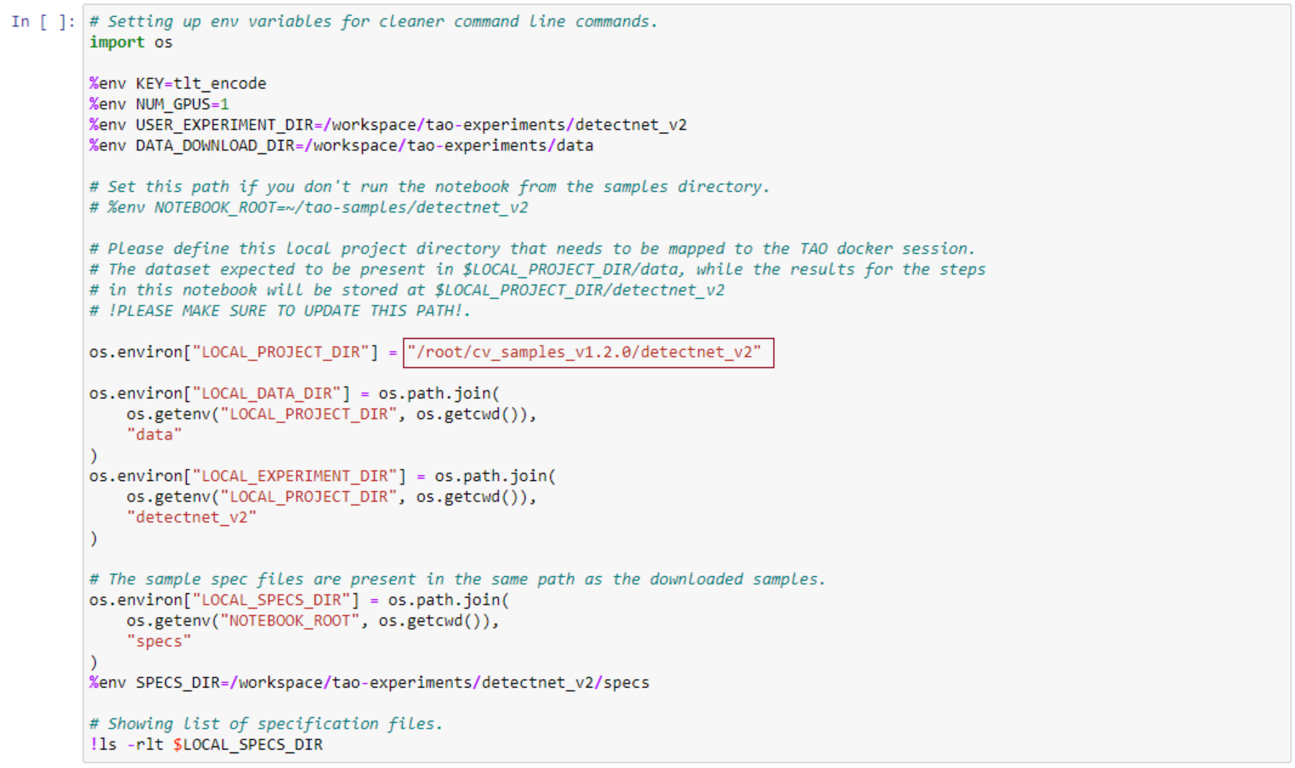


Select items to perform actions on them.

/ cv_samples_v1.2.0 / detectnet_v2	Last Modified	File size
..	seconds ago	
specs	6 hours ago	
detectnet_v2.ipynb	a year ago	46.9 kB
__init__.py	a year ago	212 B

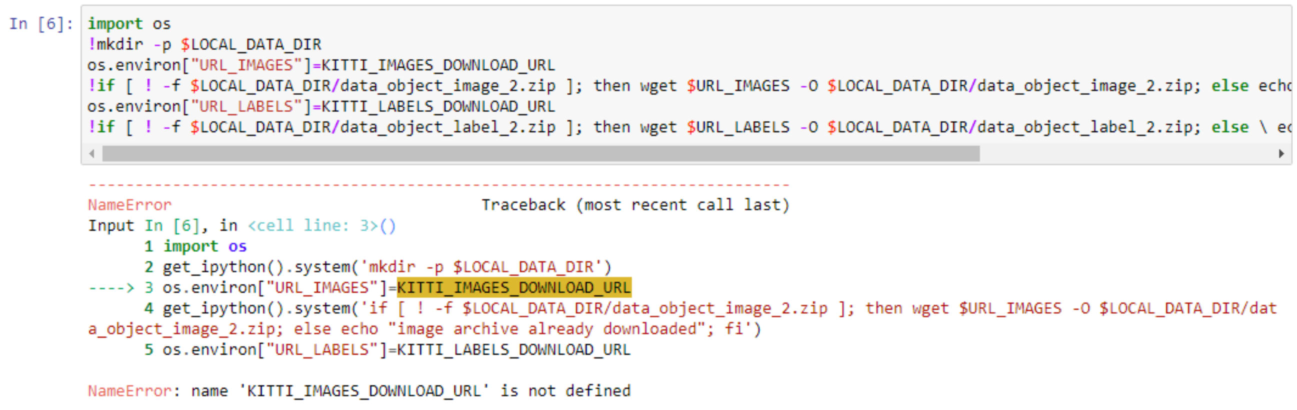
Open the file detectnet_v2.ipynb.

In section with the environment variables, set the path for LOCAL_PROJECT_DIR.



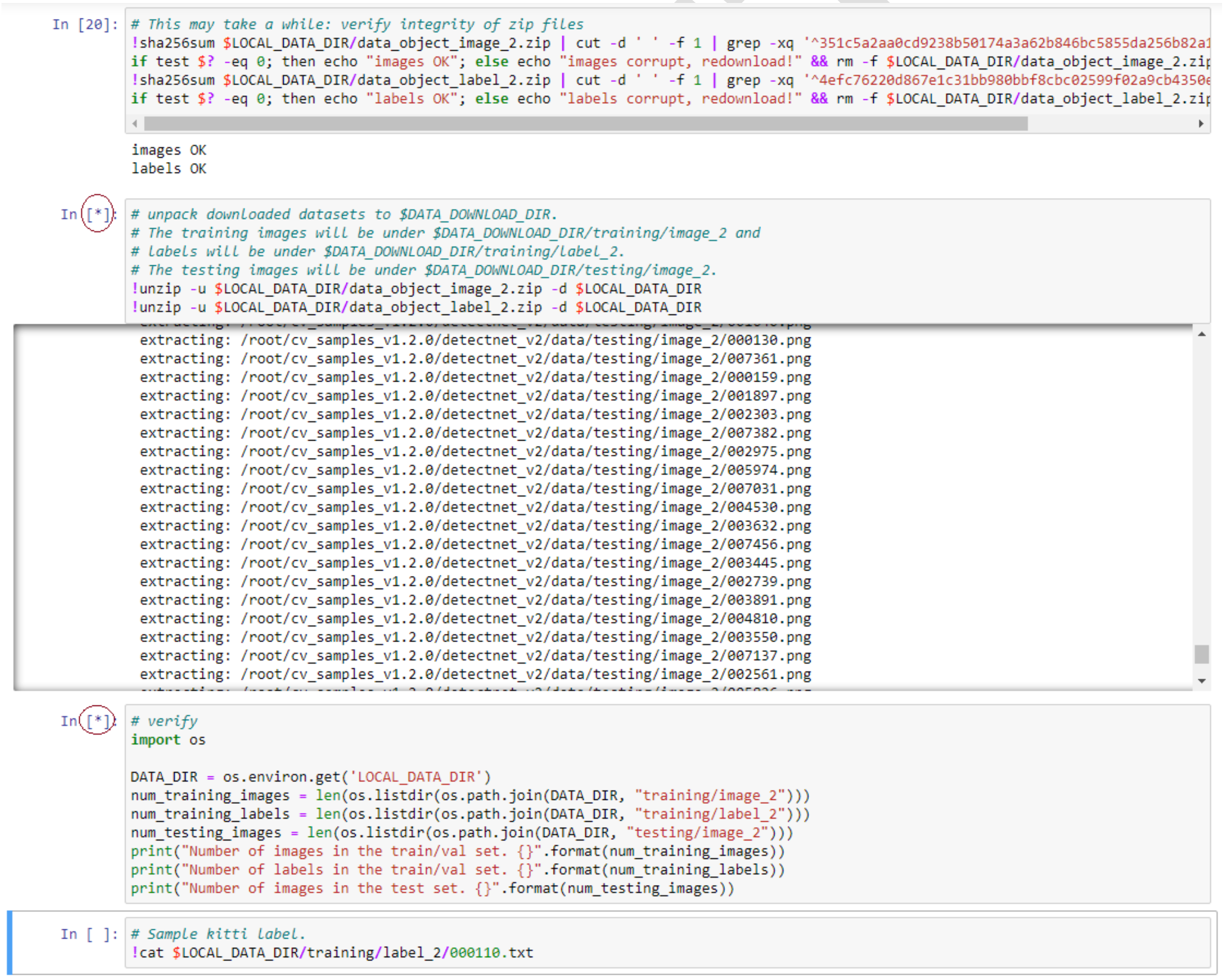
```
In [ ]: # Setting up env variables for cleaner command line commands.
        import os

        %env KEY=tlt_encode
        %env NUM_GPUS=1
        %env USER_EXPERIMENT_DIR=/workspace/tao-experiments/detectnet_v2
        %env DATA_DOWNLOAD_DIR=/workspace/tao-experiments/data

        # Set this path if you don't run the notebook from the samples directory.
        # %env NOTEBOOK_ROOT=~/tao-samples/detectnet_v2

        # Please define this local project directory that needs to be mapped to the TAO docker session.
        # The dataset expected to be present in $LOCAL_PROJECT_DIR/data, while the results for the steps
        # in this notebook will be stored at $LOCAL_PROJECT_DIR/detectnet_v2
        # !PLEASE MAKE SURE TO UPDATE THIS PATH!.

        os.environ["LOCAL_PROJECT_DIR"] = "/root/cv_samples_v1.2.0/detectnet_v2"

        os.environ["LOCAL_DATA_DIR"] = os.path.join(
            os.getenv("LOCAL_PROJECT_DIR", os.getcwd()),
            "data"
        )
        os.environ["LOCAL_EXPERIMENT_DIR"] = os.path.join(
            os.getenv("LOCAL_PROJECT_DIR", os.getcwd()),
            "detectnet_v2"
        )

        # The sample spec files are present in the same path as the downloaded samples.
        os.environ["LOCAL_SPECS_DIR"] = os.path.join(
            os.getenv("NOTEBOOK_ROOT", os.getcwd()),
            "specs"
        )
        %env SPECS_DIR=/workspace/tao-experiments/detectnet_v2/specs

        # Showing list of specification files.
        !ls -rlt $LOCAL_SPECS_DIR
```

Because the download url for the Images.zip and Labels.zip is missing, the Jupyter Notebook shows an error. However, the .zip files were already copied beforehand, so the error can be ignored.



```
In [6]: import os
        !mkdir -p $LOCAL_DATA_DIR
        os.environ["URL_IMAGES"]=KITTI_IMAGES_DOWNLOAD_URL
        !if [ ! -f $LOCAL_DATA_DIR/data_object_image_2.zip ]; then wget $URL_IMAGES -O $LOCAL_DATA_DIR/data_object_image_2.zip; else echo
        os.environ["URL_LABELS"]=KITTI_LABELS_DOWNLOAD_URL
        !if [ ! -f $LOCAL_DATA_DIR/data_object_label_2.zip ]; then wget $URL_LABELS -O $LOCAL_DATA_DIR/data_object_label_2.zip; else \ ec

        ---------------------------------------------------------------------------
        NameError                                 Traceback (most recent call last)
        Input In [6], in <cell line: 3>()
              1 import os
              2 get_ipython().system('mkdir -p $LOCAL_DATA_DIR')
        ----> 3 os.environ["URL_IMAGES"]=KITTI_IMAGES_DOWNLOAD_URL
              4 get_ipython().system('if [ ! -f $LOCAL_DATA_DIR/data_object_image_2.zip ]; then wget $URL_IMAGES -O $LOCAL_DATA_DIR/dat
        a_object_image_2.zip; else echo "image archive already downloaded"; fi')
              5 os.environ["URL_LABELS"]=KITTI_LABELS_DOWNLOAD_URL

        NameError: name 'KITTI_IMAGES_DOWNLOAD_URL' is not defined
```

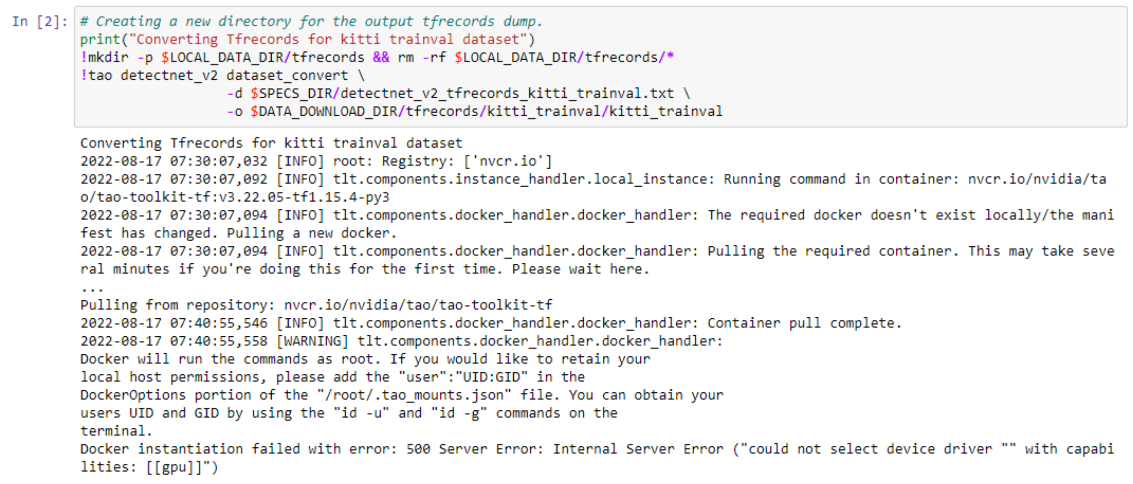
Unpacking the images can take some time. In Jupyter Notebook, partial steps in progress are marked with *.



```
In [20]: # This may take a while: verify integrity of zip files
         !sha256sum $LOCAL_DATA_DIR/data_object_image_2.zip | cut -d ' ' -f 1 | grep -xq '^351c5a2aa0cd9238b50174a3a62b846bc5855da256b82a1
         if test $? -eq 0; then echo "images OK"; else echo "images corrupt, redownload!" && rm -f $LOCAL_DATA_DIR/data_object_image_2.zip
         !sha256sum $LOCAL_DATA_DIR/data_object_label_2.zip | cut -d ' ' -f 1 | grep -xq '^4efc76220d867e1c31bb980bbf8cbc02599f02a9cb4350e
         if test $? -eq 0; then echo "labels OK"; else echo "labels corrupt, redownload!" && rm -f $LOCAL_DATA_DIR/data_object_label_2.zip

         images OK
         labels OK

In [*]:  # unpack downloaded datasets to $DATA_DOWNLOAD_DIR.
         # The training images will be under $DATA_DOWNLOAD_DIR/training/image_2 and
         # labels will be under $DATA_DOWNLOAD_DIR/training/label_2.
         # The testing images will be under $DATA_DOWNLOAD_DIR/testing/image_2.
         !unzip -u $LOCAL_DATA_DIR/data_object_image_2.zip -d $LOCAL_DATA_DIR
         !unzip -u $LOCAL_DATA_DIR/data_object_label_2.zip -d $LOCAL_DATA_DIR

          extracting: /root/cv_samples_v1.2.0/detectnet_v2/data/testing/image_2/000130.png
          extracting: /root/cv_samples_v1.2.0/detectnet_v2/data/testing/image_2/007361.png
          extracting: /root/cv_samples_v1.2.0/detectnet_v2/data/testing/image_2/000159.png
          extracting: /root/cv_samples_v1.2.0/detectnet_v2/data/testing/image_2/001897.png
          extracting: /root/cv_samples_v1.2.0/detectnet_v2/data/testing/image_2/002303.png
          extracting: /root/cv_samples_v1.2.0/detectnet_v2/data/testing/image_2/007382.png
          extracting: /root/cv_samples_v1.2.0/detectnet_v2/data/testing/image_2/002975.png
          extracting: /root/cv_samples_v1.2.0/detectnet_v2/data/testing/image_2/005974.png
          extracting: /root/cv_samples_v1.2.0/detectnet_v2/data/testing/image_2/007031.png
          extracting: /root/cv_samples_v1.2.0/detectnet_v2/data/testing/image_2/004530.png
          extracting: /root/cv_samples_v1.2.0/detectnet_v2/data/testing/image_2/003632.png
          extracting: /root/cv_samples_v1.2.0/detectnet_v2/data/testing/image_2/007456.png
          extracting: /root/cv_samples_v1.2.0/detectnet_v2/data/testing/image_2/003445.png
          extracting: /root/cv_samples_v1.2.0/detectnet_v2/data/testing/image_2/002739.png
          extracting: /root/cv_samples_v1.2.0/detectnet_v2/data/testing/image_2/003891.png
          extracting: /root/cv_samples_v1.2.0/detectnet_v2/data/testing/image_2/004810.png
          extracting: /root/cv_samples_v1.2.0/detectnet_v2/data/testing/image_2/003550.png
          extracting: /root/cv_samples_v1.2.0/detectnet_v2/data/testing/image_2/007137.png
          extracting: /root/cv_samples_v1.2.0/detectnet_v2/data/testing/image_2/002561.png

In [*]:  # verify
         import os

         DATA_DIR = os.environ.get('LOCAL_DATA_DIR')
         num_training_images = len(os.listdir(os.path.join(DATA_DIR, "training/image_2")))
         num_training_labels = len(os.listdir(os.path.join(DATA_DIR, "training/label_2")))
         num_testing_images = len(os.listdir(os.path.join(DATA_DIR, "testing/image_2")))
         print("Number of images in the train/val set. {}".format(num_training_images))
         print("Number of labels in the train/val set. {}".format(num_training_labels))
         print("Number of images in the test set. {}".format(num_testing_images))

In [ ]:  # Sample kitti label.
         !cat $LOCAL_DATA_DIR/training/label_2/000110.txt
```

When the partial step is completed, the * mark is replaced with a step number.

The "Converting Tfrecords for kitti trainval dataset " substep may also take a while. This is where the GPU query takes place for the first time.

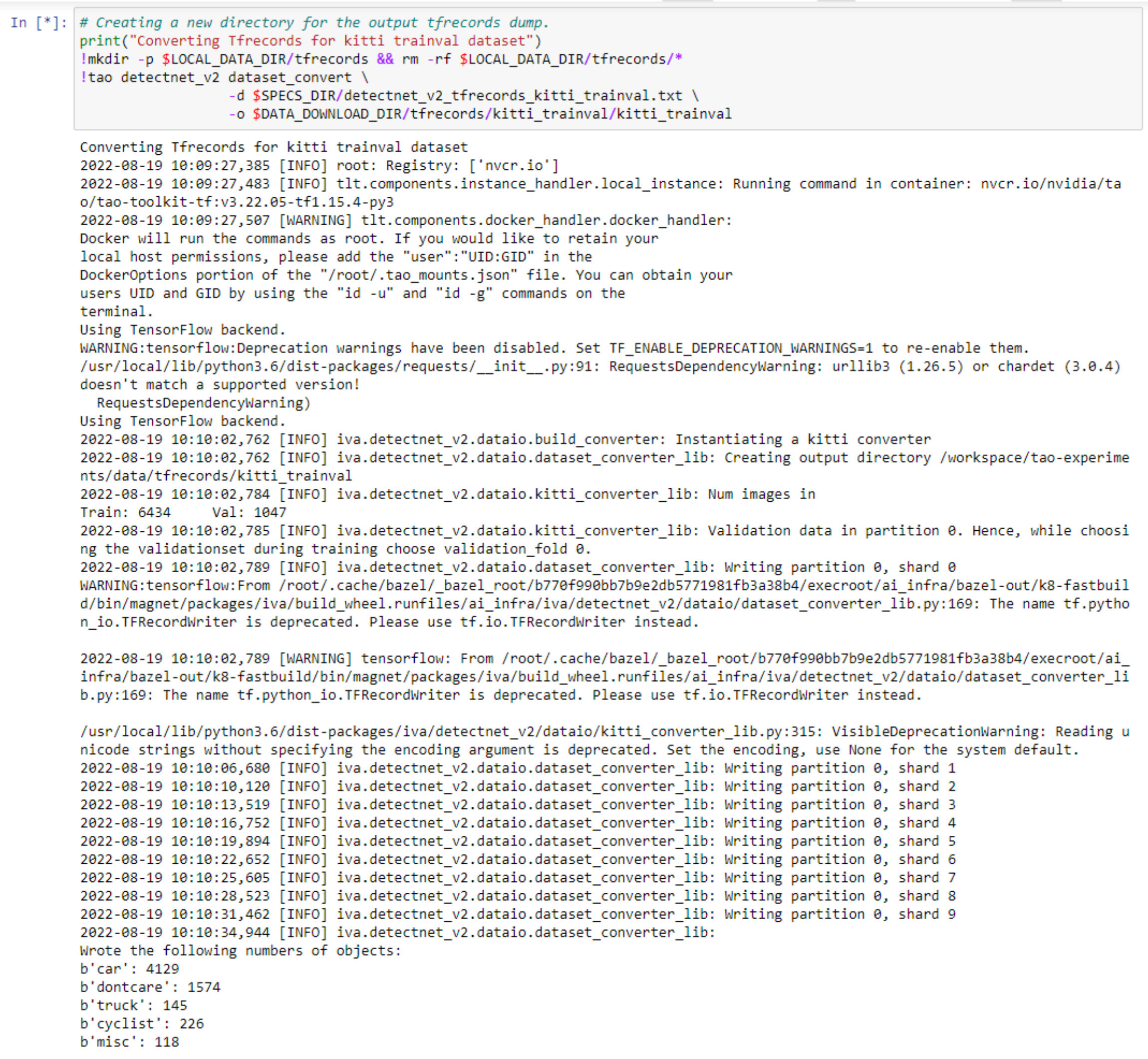


```
In [2]: # Creating a new directory for the output tfrecords dump.
        print("Converting Tfrecords for kitti trainval dataset")
        !mkdir -p $LOCAL_DATA_DIR/tfrecords && rm -rf $LOCAL_DATA_DIR/tfrecords/*
        !tao detectnet_v2 dataset_convert \
                          -d $SPECS_DIR/detectnet_v2_tfrecords_kitti_trainval.txt \
                          -o $DATA_DOWNLOAD_DIR/tfrecords/kitti_trainval/kitti_trainval
```

```
Converting Tfrecords for kitti trainval dataset
2022-08-17 07:30:07,032 [INFO] root: Registry: ['nvcr.io']
2022-08-17 07:30:07,092 [INFO] tlt.components.instance_handler.local_instance: Running command in container: nvcr.io/nvidia/ta
o/tao-toolkit-tf:v3.22.05-tf1.15.4-py3
2022-08-17 07:30:07,094 [INFO] tlt.components.docker_handler.docker_handler: The required docker doesn't exist locally/the mani
fest has changed. Pulling a new docker.
2022-08-17 07:30:07,094 [INFO] tlt.components.docker_handler.docker_handler: Pulling the required container. This may take seve
ral minutes if you're doing this for the first time. Please wait here.
...
Pulling from repository: nvcr.io/nvidia/tao/tao-toolkit-tf
2022-08-17 07:40:55,546 [INFO] tlt.components.docker_handler.docker_handler: Container pull complete.
2022-08-17 07:40:55,558 [WARNING] tlt.components.docker_handler.docker_handler:
Docker will run the commands as root. If you would like to retain your
local host permissions, please add the "user":"UID:GID" in the
DockerOptions portion of the "/root/.tao_mounts.json" file. You can obtain your
users UID and GID by using the "id -u" and "id -g" commands on the
terminal.
Docker instantiation failed with error: 500 Server Error: Internal Server Error ("could not select device driver "" with capabi
lities: [[gpu]]")
```

This message does not appear if the NVidia Container Toolkit is installed correctly.
See chapter *NVidia Container Toolkit*.

If the GPU is detected, the following message appears.



```
In [*]: # Creating a new directory for the output tfrecords dump.
        print("Converting Tfrecords for kitti trainval dataset")
        !mkdir -p $LOCAL_DATA_DIR/tfrecords && rm -rf $LOCAL_DATA_DIR/tfrecords/*
        !tao detectnet_v2 dataset_convert \
                          -d $SPECS_DIR/detectnet_v2_tfrecords_kitti_trainval.txt \
                          -o $DATA_DOWNLOAD_DIR/tfrecords/kitti_trainval/kitti_trainval
```

```
Converting Tfrecords for kitti trainval dataset
2022-08-19 10:09:27,385 [INFO] root: Registry: ['nvcr.io']
2022-08-19 10:09:27,483 [INFO] tlt.components.instance_handler.local_instance: Running command in container: nvcr.io/nvidia/ta
o/tao-toolkit-tf:v3.22.05-tf1.15.4-py3
2022-08-19 10:09:27,507 [WARNING] tlt.components.docker_handler.docker_handler:
Docker will run the commands as root. If you would like to retain your
local host permissions, please add the "user":"UID:GID" in the
DockerOptions portion of the "/root/.tao_mounts.json" file. You can obtain your
users UID and GID by using the "id -u" and "id -g" commands on the
terminal.
Using TensorFlow backend.
WARNING:tensorflow:Deprecation warnings have been disabled. Set TF_ENABLE_DEPRECATION_WARNINGS=1 to re-enable them.
/usr/local/lib/python3.6/dist-packages/requests/__init__.py:91: RequestsDependencyWarning: urllib3 (1.26.5) or chardet (3.0.4)
doesn't match a supported version!
  RequestsDependencyWarning)
Using TensorFlow backend.
2022-08-19 10:10:02,762 [INFO] iva.detectnet_v2.dataio.build_converter: Instantiating a kitti converter
2022-08-19 10:10:02,762 [INFO] iva.detectnet_v2.dataio.dataset_converter_lib: Creating output directory /workspace/tao-experime
nts/data/tfrecords/kitti_trainval
2022-08-19 10:10:02,784 [INFO] iva.detectnet_v2.dataio.kitti_converter_lib: Num images in
Train: 6434     Val: 1047
2022-08-19 10:10:02,785 [INFO] iva.detectnet_v2.dataio.kitti_converter_lib: Validation data in partition 0. Hence, while choosi
ng the validationset during training choose validation_fold 0.
2022-08-19 10:10:02,789 [INFO] iva.detectnet_v2.dataio.dataset_converter_lib: Writing partition 0, shard 0
WARNING:tensorflow:From /root/.cache/bazel/_bazel_root/b770f990bb7b9e2db5771981fb3a38b4/execroot/ai_infra/bazel-out/k8-fastbuil
d/bin/magnet/packages/iva/build_wheel.runfiles/ai_infra/iva/detectnet_v2/dataio/dataset_converter_lib.py:169: The name tf.pytho
n_io.TFRecordWriter is deprecated. Please use tf.io.TFRecordWriter instead.

2022-08-19 10:10:02,789 [WARNING] tensorflow: From /root/.cache/bazel/_bazel_root/b770f990bb7b9e2db5771981fb3a38b4/execroot/ai_
infra/bazel-out/k8-fastbuild/bin/magnet/packages/iva/build_wheel.runfiles/ai_infra/iva/detectnet_v2/dataio/dataset_converter_li
b.py:169: The name tf.python_io.TFRecordWriter is deprecated. Please use tf.io.TFRecordWriter instead.

/usr/local/lib/python3.6/dist-packages/iva/detectnet_v2/dataio/kitti_converter_lib.py:315: VisibleDeprecationWarning: Reading u
nicode strings without specifying the encoding argument is deprecated. Set the encoding, use None for the system default.
2022-08-19 10:10:06,680 [INFO] iva.detectnet_v2.dataio.dataset_converter_lib: Writing partition 0, shard 1
2022-08-19 10:10:10,120 [INFO] iva.detectnet_v2.dataio.dataset_converter_lib: Writing partition 0, shard 2
2022-08-19 10:10:13,519 [INFO] iva.detectnet_v2.dataio.dataset_converter_lib: Writing partition 0, shard 3
2022-08-19 10:10:16,752 [INFO] iva.detectnet_v2.dataio.dataset_converter_lib: Writing partition 0, shard 4
2022-08-19 10:10:19,894 [INFO] iva.detectnet_v2.dataio.dataset_converter_lib: Writing partition 0, shard 5
2022-08-19 10:10:22,652 [INFO] iva.detectnet_v2.dataio.dataset_converter_lib: Writing partition 0, shard 6
2022-08-19 10:10:25,605 [INFO] iva.detectnet_v2.dataio.dataset_converter_lib: Writing partition 0, shard 7
2022-08-19 10:10:28,523 [INFO] iva.detectnet_v2.dataio.dataset_converter_lib: Writing partition 0, shard 8
2022-08-19 10:10:31,462 [INFO] iva.detectnet_v2.dataio.dataset_converter_lib: Writing partition 0, shard 9
2022-08-19 10:10:34,944 [INFO] iva.detectnet_v2.dataio.dataset_converter_lib:
Wrote the following numbers of objects:
b'car': 4129
b'dontcare': 1574
b'truck': 145
b'cyclist': 226
b'misc': 118
```

The assignments and number of existing images are interesting.

```
b'car': 4129
b'dontcare': 1574
b'truck': 145
b'cyclist': 226
b'misc': 118
b'pedestrian': 638
b'tram': 67
b'van': 377
b'person_sitting': 23
```

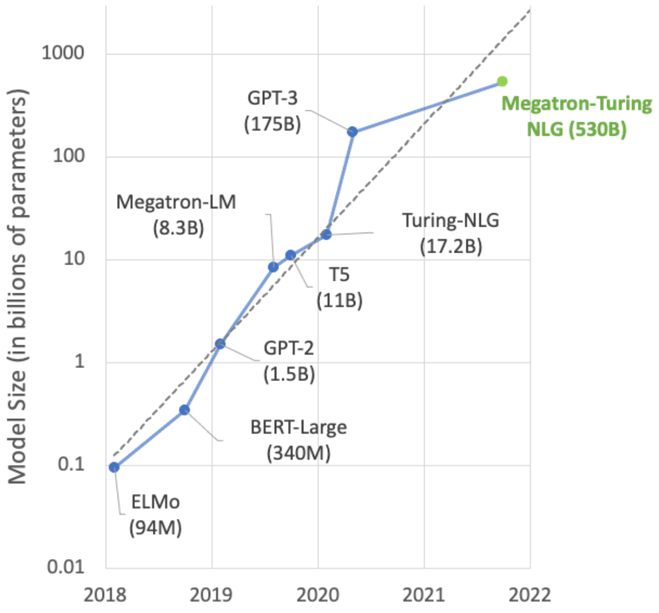
2.5 First finding

With the existing Jupyter Notebook, the training-validation-testing for the existing classifications can be carried out in a clearly defined process. The sub-phases for "Unpruned" and "Pruned" show that the accuracy factor is given high priority.

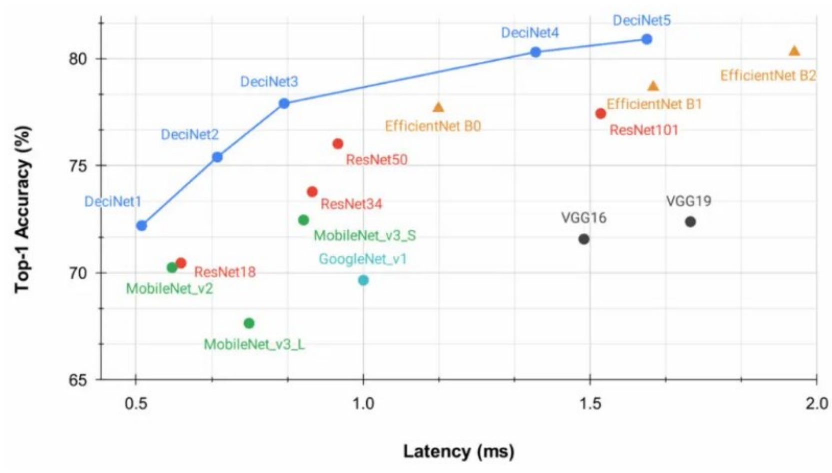
Higher accuracy is a prerequisite for using the trained models in production to solve real-world problems reliably and effectively.
Do you want to create and train desired properties, e.g. other classes "train", "excavator", crane" yourself?

Building high-precision models from scratch is time-consuming and capital-intensive, states NVidia on their blog. The blog refers to which neural networks are supported in the TAO Toolkit, which public datasets have been used for training, and which pre-trained models with which classifications have been created and how they have been validated.

In recent years, the number of parameters per deep neural network (DNNs)[8] increased logarithmically.
A downside is the enormous amount of energy[9] required for the training phase.



[8] https://developer.nvidia.com/blog/using-deepspeed-and-megatron-to-train-megatron-turing-nlg-530b-the-worlds-largest-and-most-powerful-generative-language-model/
[9] https://ekamperi.github.io/machine%20learning/2019/08/14/energy-considerations-dnn.html



Another challenge seems to be the choice of the state-of-the-art (SOTA) DNN depending on a specified hardware in order to ensure high accuracy and at the same time low latency later in production.

With regard to Azure Stack Edge Hub models suitable for DNN, there are no publications from the Microsoft Deep Learning Group[10].

The lab infrastructure using Azure VM with included M60 GPU was successfully set up. The TAO training with a suitable photo collection from step 4 in the Jupyter notebook was not carried out in favor of setting up an inference server beforehand.

In the next lab, an inference server is to be provisioned. The goal is to then use it to validate from the DashcamNet model and use inference from a video source.

2.5.1 Notes for optimizations on the lab infrastructure

The lab setup has so far been developed with root rights on Photon OS. This should be done with a dedicated local service account user. Secure storage of credentials for the Public NVidia Registry should be considered.
The NVidia installer asks for text input. It is to be examined how these can be eliminated for the purpose of higher automation.
The Python Grid Visualizer is built on top of the Matplotlib component. It is to be investigated how the Python import in the matplotlib_inline-backend can be done without errors ("ModuleNotFoundError").

[10] https://www.microsoft.com/en-us/research/group/deep-learning-group/

3 Lab Infrastructure Triton Inference Server

In this lab, the NVidia Docker container variant - without Kubernetes[11] provisioning - is used for the Triton Inference Servers benutzt. Download the Triton Inference Server from https://catalog.ngc.nvidia.com/orgs/nvidia/containers/tritonserver/tags.

3.1 Provisioning of Tritonserver without Deepstream on Photon OS

In this example, the NVidia Triton server without deepstream is provisioned. Photon OS serves as the host system. Docker and the NVidia Container Toolkit are already installed.

```
cd /root

tdnf install -y git curl
git clone https://github.com/triton-inference-server/server.git

export EXAMPLESPATH=/root/server/docs/examples

cd $EXAMPLESPATH
./fetch_models.sh

docker pull nvcr.io/nvidia/tritonserver:22.07-py3

docker run --gpus=1 --rm -p8000:8000 -p8001:8001 -p8002:8002 -v/$EXAMPLESPATH/model_repository:/mo-
dels nvcr.io/nvidia/tritonserver:22.07-py3 tritonserver --model-repository=/models
```

Connected to the host system, e.g. using Putty, the status of the inference server can be checked.

```
# Check auf return status 200 dass Triton korrekt läuft
LocalAdminUser@NVidia01 [ ~ ]$ curl -v localhost:8000/v2/health/ready
*   Trying 127.0.0.1:8000...
* Connected to localhost (127.0.0.1) port 8000 (#0)
> GET /v2/health/ready HTTP/1.1
> Host: localhost:8000
> User-Agent: curl/7.83.1
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Content-Length: 0
< Content-Type: text/plain
<
* Connection #0 to host localhost left intact
LocalAdminUser@NVidia01 [ ~ ]$
```

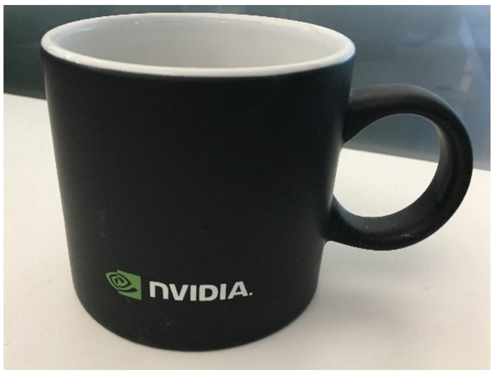
3.1.1 Installationskontrolle mittels Built-in Bildtesterkennung

A test detection can now be carried out.

```
# Abholen der Clientbeispiele
docker pull nvcr.io/nvidia/tritonserver:22.07-py3-sdk
docker run -it --rm --net=host nvcr.io/nvidia/tritonserver:22.07-py3-sdk
```

The following image serves as a built-in NVidia example.

[11] https://github.com/triton-inference-server/server/tree/main/deploy/k8s-onprem



```
# test detection
root@NVidia01:/workspace# /workspace/ins-
tall/bin/image_client -m densenet_onnx -c 3 -s
INCEPTION /workspace/images/mug.jpg
Request 0, batch size 1
Image '/workspace/images/mug.jpg':
    15.349563 (504) = COFFEE MUG
    13.227461 (968) = CUP
    10.424893 (505) = COFFEEPOT
root@NVidia01:/workspace#
```

The Densenet_ONNX model allows clear recognition for the specific image.

3.1.2 Image recognition of own pictures

For a non-built-in image test detection, three images were copied into the Docker container.

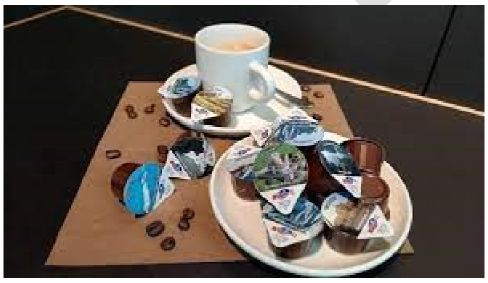
- datwyler-mug-picture01.jpg
- datwyler-mug-picture02.jpg
- datwyler-kaffee-kapseln.jpg

Port 8000 has been enabled for access to the inference server (Azure + Photon OS).

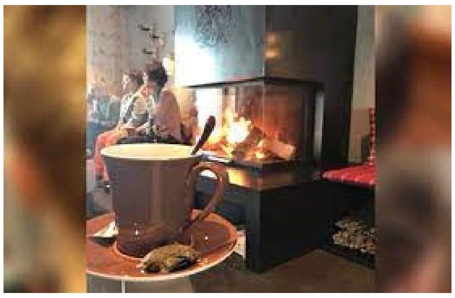
In the test, the inference server got the IP address `20.114.169.209`.

```
root@NVidia01 [ ~ ]# docker run -d --net=host nvcr.io/nvidia/tritonserver:22.07-py3-sdk
41c61826af630362d742f4f66e2759a5af624d431fffee5bf02e1314ed30bb34
root@NVidia01 [ ~ ]# docker cp /root/datwyler-mug-picture01.jpg
41c61826af630362d742f4f66e2759a5af624d431fffee5bf02e1314ed30bb34:/workspace/images/datwyler-mug-
picture01.jpg
root@NVidia01 [ ~ ]# docker cp /root/datwyler-mug-picture02.jpg
41c61826af630362d742f4f66e2759a5af624d431fffee5bf02e1314ed30bb34:/workspace/images/datwyler-mug-
picture02.jpg
root@NVidia01 [ ~ ]# docker cp /root/datwyler-kaffee-kapseln.jpg
41c61826af630362d742f4f66e2759a5af624d431fffee5bf02e1314ed30bb34:/workspace/images/datwyler-kaffee-
kapseln.jpg
root@NVidia01 [ ~ ]# docker commit 41c61826af630362d742f4f66e2759a5af624d431fffee5bf02e1314ed30bb34
sha256:e53607580c16a70c22c97c0615c7c5aa880742272f76cb62aa9251995795e137
root@NVidia01 [ ~ ]# docker run --entrypoint=bash -it
sha256:e53607580c16a70c22c97c0615c7c5aa880742272f76cb62aa9251995795e137
root@bfb844abacf9:/workspace# cd images
root@bfb844abacf9:/workspace/images# ls
datwyler-kaffee-kapseln.jpg  datwyler-mug-picture01.jpg  datwyler-mug-picture02.jpg  mug.jpg
root@bfb844abacf9:/workspace/images#
```

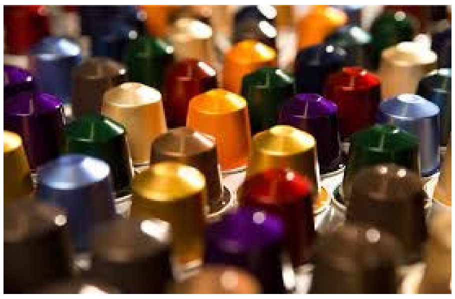
Here the recognition output:



```
root@bfb844abacf9:/workspace# /workspace/in-
stall/bin/image_client -u 20.114.169.209:8000 -m
densenet_onnx -c 3 -s INCEPTION /work-
space/images/datwyler-mug-picture01.jpg
Request 0, batch size 1
Image '/workspace/images/datwyler-mug-pic-
ture01.jpg':
    8.462603 (868) = TRAY
    7.758639 (923) = PLATE
    7.508104 (532) = DINING TABLE
```



```
root@bfb844abacf9:/workspace# /workspace/in-
stall/bin/image_client -u 20.114.169.209:8000 -m
densenet_onnx -c 3 -s INCEPTION /work-
space/images/datwyler-mug-picture02.jpg
Request 0, batch size 1
Image '/workspace/images/datwyler-mug-pic-
ture02.jpg':
    8.293711 (470) = CANDLE
    8.132739 (550) = ESPRESSO MAKER
    7.686849 (827) = STOVE
```



```
root@bfb844abacf9:/workspace# /workspace/in-
stall/bin/image_client -u 20.114.169.209:8000 -m
densenet_onnx -c 3 -s INCEPTION /work-
space/images/datwyler-kaffee-kapseln.jpg
Request 0, batch size 1
Image '/workspace/images/datwyler-kaffee-kap-
seln.jpg':
    10.763636 (878) = TYPEWRITER KEYBOARD
    9.352649 (494) = CHIME
    9.134264 (810) = SPACE BAR
root@bfb844abacf9:/workspace#
```

In contrast to the reference image, the image quality is very poor. The cup was not recognized in all three examples.

One finding is that images with low resolution are also classified as metadata.

3.2 Provisioned Docker Container with Deepstream and Triton Inference Server on Photon OS

With respect to https://catalog.ngc.nvidia.com/orgs/nvidia/containers/deepstream the docker container `nvcr.io/nvidia/deepstream:6.1-triton` is ideal for inferencing server application with deepstream sdk.

Open the ports needed (also on for the Azure vm).

```
iptables -A INPUT -i eth0 -p udp --dport 5400 -j ACCEPT
iptables -A OUTPUT -p udp --dport 5400 -j ACCEPT
iptables -A INPUT -i eth0 -p tcp --dport 8000 -j ACCEPT
iptables -A OUTPUT -p tcp --dport 8000 -j ACCEPT
iptables -A INPUT -i eth0 -p tcp --dport 8001 -j ACCEPT
iptables -A OUTPUT -p tcp --dport 8001 -j ACCEPT
iptables -A INPUT -i eth0 -p tcp --dport 8002 -j ACCEPT
iptables -A OUTPUT -p tcp --dport 8002 -j ACCEPT
iptables-save >/etc/systemd/scripts/ip4save
```

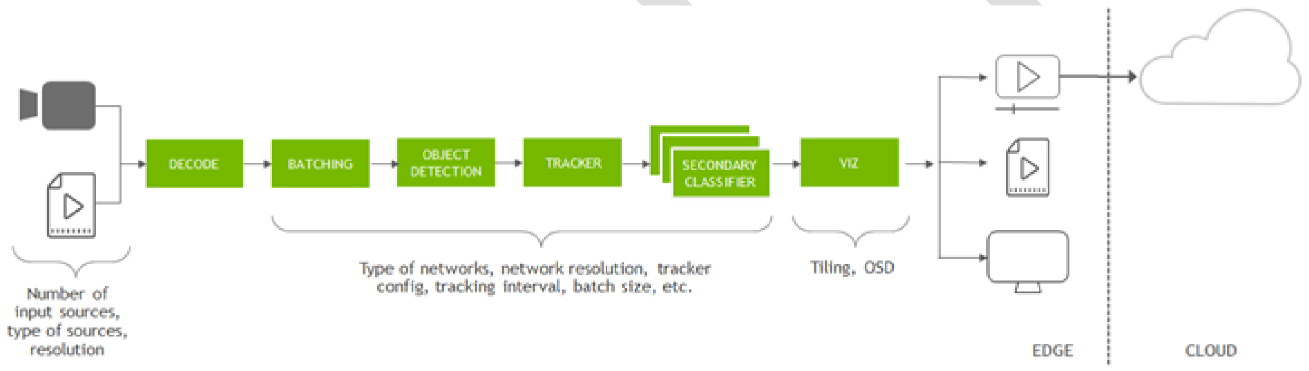
Provision the docker container.

```
docker run --gpus all -it --rm --net=host nvcr.io/nvidia/deepstream:6.1-triton -p 8000:8000/tcp -p
8001:8001/tcp -p 8002:8002/tcp -p 5400:5400/udp -p 8554:8554
```

output:

```
Status: Downloaded newer image for nvcr.io/nvidia/deepstream:6.1-triton

===============================
   DeepStreamSDK 6.1.0
===============================

*** LICENSE AGREEMENT ***
By using this software you agree to fully comply with the terms and conditions
of the License Agreement. The License Agreement is located at
/opt/nvidia/deepstream/deepstream/LicenseAgreement.pdf. If you do not agree
to the terms and conditions of the License Agreement do not use the software.


=============================
== Triton Inference Server ==
=============================

NVIDIA Release 22.03 (build 33743047)
Triton Server Version 2.20.0

Copyright (c) 2018-2022, NVIDIA CORPORATION & AFFILIATES.  All rights reserved.

Various files include modifications (c) NVIDIA CORPORATION & AFFILIATES.  All rights reserved.

This container image and its contents are governed by the NVIDIA Deep Learning Container License.
By pulling and using the container, you accept the terms and conditions of this license:
https://developer.nvidia.com/ngc/nvidia-deep-learning-container-license

NOTE: CUDA Forward Compatibility mode ENABLED.
  Using CUDA 11.6 driver version 510.47.03 with kernel driver version 470.141.03.
  See https://docs.nvidia.com/deploy/cuda-compatibility/ for details.

root@NVidia01:/opt/nvidia/deepstream/deepstream-6.1#
```

In ***Fehler! Verweisquelle konnte nicht gefunden werden.*** you can see the components of the sequence for a successful inferencing. In order for this to work, however, the configurations of the components must be coordinated. The Reference Apps[12] offers a good start to this modular principle.


DECODE
BATCHING
OBJECT DETECTION
TRACKER
SECONDARY CLASSIFIER
VIZ
Number of input sources, type of sources, resolution
Type of networks, network resolution, tracker config, tracking interval, batch size, etc.
Tiling, OSD
EDGE
CLOUD

Picture 1 NVidia DeepStream-App[13]

Inside the docker container, run the app with `deepstream-app`.

```
Usage:
  deepstream-app [OPTION?] Nvidia DeepStream Demo

Help Options:
  -h, --help                        Show help options
  --help-all                        Show all help options
  --help-gst                        Show GStreamer Options

Application Options:
  -v, --version                     Print DeepStreamSDK version
  -t, --tiledtext                   Display Bounding box labels in tiled mode
  --version-all                     Print DeepStreamSDK and dependencies version
  -c, --cfg-file                    Set the config file
  -i, --input-uri                   Set the input uri (file://stream or rtsp://stream)
```

The directory `/opt/nvidia/deepstream/deepstream-6.1/samples/configs/deepstream-app` contains a few sample configurations, although without the DashCamNet-Model from chapter 1.1.1.

[12] https://docs.nvidia.com/metropolis/deepstream/6.1/dev-guide/text/DS_Overview.html#deepstream-reference-app
[13] https://docs.nvidia.com/metropolis/deepstream/6.1/dev-guide/_images/DS_overview_reference_app.png

The original literature for the following description comes from https://wiki.seeedstudio.com/Traffic-Management-DeepStream-SDK/ .

In order to use the example with the included DashCamNet model, some things have to be installed within the Docker container.

```
# Avoid issue https://forums.developer.nvidia.com/t/deepstream-execution-fails/217419/4#
# apt-get update -y
apt-get install -y cuda-toolkit-11-6

# Download configuration files
git clone https://github.com/NVIDIA-AI-IOT/deepstream_reference_apps.git
cd /opt/nvidia/deepstream/deepstream-6.1/deepstream_reference_apps/deepstream_app_tao_configs
cp -a * /opt/nvidia/deepstream/deepstream-6.1/samples/configs/tao_pretrained_models/
# Download models
apt-get install -y wget zip
cd /opt/nvidia/deepstream/deepstream-6.1/samples/configs/tao_pretrained_models/
./download_models.sh
```

Executing `deepstream-app -c deepstream_app_source1_dashcamnet_vehiclemakenet_vehicletypenet.txt` results in an issue.

```
** INFO: <bus_callback:194>: Pipeline ready

Error String : Feature not supported on this GPUError Code : 801
ERROR from nvv4l2decoder0: Failed to process frame.
Debug info: gstv4l2videodec.c(1747): gst_v4l2_video_dec_handle_frame (): /GstPipeline:pipeline/Gst-
Bin:multi_src_bin/GstBin:src_sub_bin0/GstURIDecodeBin:src_elem/GstDecodeBin:decodebin0/nvv4l2deco-
der:nvv4l2decoder0:
Maybe be due to not enough memory or failing driver
ERROR from qtdemux0: Internal data stream error.
Debug info: qtdemux.c(6605): gst_qtdemux_loop (): /GstPipeline:pipeline/GstBin:multi_src_bin/Gst-
Bin:src_sub_bin0/GstURIDecodeBin:src_elem/GstDecodeBin:decodebin0/GstQTDemux:qtdemux0:
streaming stopped, reason error (-5)
Quitting
[NvMultiObjectTracker] De-initialized
App run failed
```

An issue case https://forums.developer.nvidia.com/t/deepstream-sdk-6-1-deepstream-app-example-issue-nvv4l2decoder0-failed-to-process-frame has been opened in the NVidia forum, and the response came promptly that the M60 NVidia card does not support hardware assisted h265 decoding.

At https://developer.nvidia.com/video-encode-and-decode-gpu-support-matrix-new each graphics card is listed with capabilities. In order to set up the visualization for an M60 GPU, a stream or a file in h264 format must be available.

The deepstream app offers the possibility to configure the pipeline components and thus also the output. Modifications have to be applied on the configuration file `deepstream_app_source1_dashcamnet_vehiclemakenet_vehicletypenet.txt`.

Specifically, the [sink0] section must be deactivated and the [sink2] section activated, see *Chapter Configuration file deepstream_app_source1_dashcamnet_vehiclemakenet_vehicletypenet.txt* . The adjustments are marked in bold. Sink numbers 0 and 2 have been adjusted for simplicity.

To prepare a RTSP stream for a simple display in a browser, an open source product, e.g. https://github.com/deepch/RTSPtoWeb can be used.

3.3 Bidirectional messaging with Kafka

As soon as objects or people are detected, they receive a tracking ID. This makes the deep stream app. In addition, one would like to further process such events and, for example, achieve a connection to a monitoring system. For this, DeepStream supports several IoT functions. The so-called Kafka protocol adapter allows bi-directional messaging, and intelligent recording based on an anomaly is supported via API. In addition, DeepStream also supports "over the air" updates of AI models while the application is running.

The following figure shows the two-way messaging architecture:

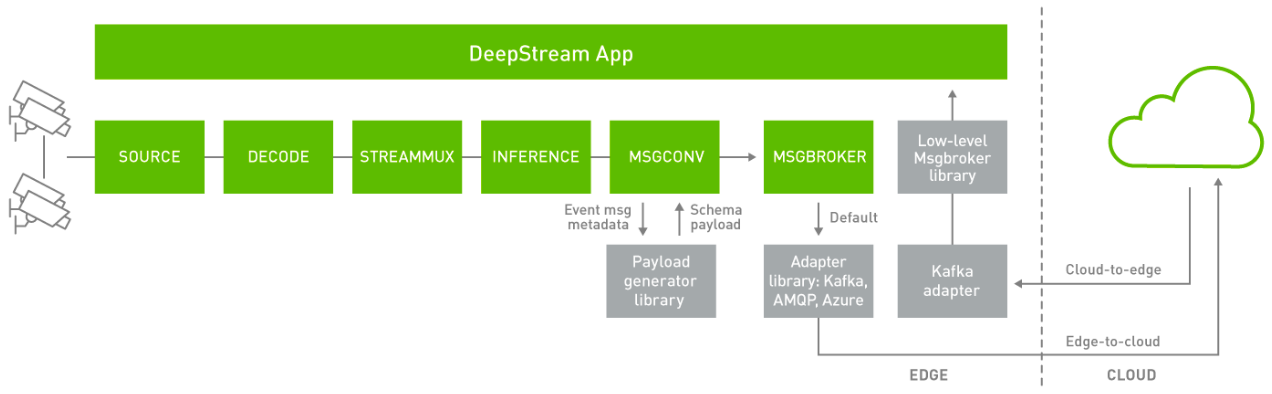


Figure 10 Deepstream IoT-Features[14]

Further literature on this topic is available under the following web links:
https://github.com/NVIDIA-AI-IOT/deepstream-occupancy-analytics/blob/master/README.md
https://docs.nvidia.com/metropolis/deepstream/5.0DP/dev-guide/index.html
https://docs.nvidia.com/metropolis/deepstream/6.1/dev-guide/text/DS_IoT.html#secure-edge-to-cloud-messaging
https://copyfuture.com/blogs-details/202206160503484598

3.3.1 Kafka Messages

According to the developer description on https://docs.nvidia.com/metropolis/deepstream/dev-guide/text/DS_ref_app_deepstream.html#sink-group, type=6 for message converters can be set up in a sink group. The corresponding plugin gst-nvmsgbroker supports the Kafka Protocol Adapter, AMQP Protocol Adapter, REDIS Protocol Adapter and Azure MQTT Protocol Adapter Libraries.

According to the somewhat older example from https://edgestreamsdk-tutorial.readthedocs.io/en/latest/sample_app.html, rules with snmp actions can be configured in the json file.
In order to process Kafka messages converted to SNMP, you may have to develop your own plugin.

[14] https://docs.nvidia.com/metropolis/deepstream/6.1/dev-guide/text/DS_IoT.html#secure-edge-to-cloud-messaging

4 Configuration file deepstream_app_source1_dashcamnet_vehiclemakenet_vehicletypenet.txt

```
################################################################################
# Copyright (c) 2020, NVIDIA CORPORATION. All rights reserved.
#
# Permission is hereby granted, free of charge, to any person obtaining a
# copy of this software and associated documentation files (the "Software"),
# to deal in the Software without restriction, including without limitation
# the rights to use, copy, modify, merge, publish, distribute, sublicense,
# and/or sell copies of the Software, and to permit persons to whom the
# Software is furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in
# all copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.  IN NO EVENT SHALL
# THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
# FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
# DEALINGS IN THE SOFTWARE.
################################################################################

[application]
enable-perf-measurement=1
perf-measurement-interval-sec=1

[tiled-display]
enable=1
rows=1
columns=1
width=1280
height=720
gpu-id=0

[source0]
enable=1
#Type - 1=CameraV4L2 2=URI 3=MultiURI
type=3
num-sources=1
uri=file://../../streams/sample_1080p_h264.mp4
gpu-id=0

[streammux]
gpu-id=0
batch-size=1
batched-push-timeout=40000
## Set muxer output width and height
width=1920
height=1080

[sink2]
enable=0
#Type - 1=FakeSink 2=EglSink 3=File
type=2
sync=1
source-id=0
gpu-id=0

[osd]
enable=1
gpu-id=0
border-width=3
text-size=15
text-color=1;1;1;1;
text-bg-color=0.3;0.3;0.3;1
font=Arial

[primary-gie]
enable=1
```

```
gpu-id=0
# Modify as necessary
model-engine-file=../../models/tao_pretrained_models/dashcamnet/resnet18_dash-
camnet_pruned.etlt_b1_gpu0_int8.engine
batch-size=1
#Required by the app for OSD, not a plugin property
bbox-border-color0=1;0;0;1
bbox-border-color1=0;1;1;1
bbox-border-color2=0;0;1;1
bbox-border-color3=0;1;0;1
gie-unique-id=1
config-file=config_infer_primary_dashcamnet.txt

[sink1]
enable=0
type=3
#1=mp4 2=mkv
container=1
#1=h264 2=h265 3=mpeg4
codec=1
#encoder type 0=Hardware 1=Software
enc-type=0
sync=0
bitrate=2000000
#H264 Profile - 0=Baseline 2=Main 4=High
#H265 Profile - 0=Main 1=Main10
profile=0
output-file=out.mp4
source-id=0
```

```
[sink0]
enable=1
```

```
#Type - 1=FakeSink 2=EglSink 3=File 4=RTSPStreaming 5=Overlay
type=4
#1=h264 2=h265
codec=1
#encoder type 0=Hardware 1=Software
enc-type=0
sync=0
bitrate=4000000
#H264 Profile - 0=Baseline 2=Main 4=High
#H265 Profile - 0=Main 1=Main10
profile=0
# set below properties in case of RTSPStreaming
rtsp-port=8554
udp-port=5400

[tracker]
enable=1
# For NvDCF and DeepSORT tracker, tracker-width and tracker-height must be a multiple of 32, respec-
tively
tracker-width=640
tracker-height=384
ll-lib-file=/opt/nvidia/deepstream/deepstream/lib/libnvds_nvmultiobjecttracker.so
# ll-config-file required to set different tracker types
# ll-config-file=../deepstream-app/config_tracker_IOU.yml
ll-config-file=../deepstream-app/config_tracker_NvDCF_perf.yml
# ll-config-file=../deepstream-app/config_tracker_NvDCF_accuracy.yml
# ll-config-file=../deepstream-app/config_tracker_DeepSORT.yml
gpu-id=0
enable-batch-process=1
enable-past-frame=1
display-tracking-id=1

[secondary-gie0]
enable=1
model-engine-file=../../models/tao_pretrained_models/vehiclemakenet/resnet18_vehiclemake-
net_pruned.etlt_b4_gpu0_int8.engine
gpu-id=0
batch-size=4
gie-unique-id=4
operate-on-gie-id=1
operate-on-class-ids=0;
config-file=config_infer_secondary_vehiclemakenet.txt
```

```
[secondary-gie1]
enable=1
model-engine-file=../../models/tao_pretrained_models/vehicletypenet/resnet18_vehiclety-
penet_pruned.etlt_b4_gpu0_int8.engine
gpu-id=0
batch-size=4
gie-unique-id=5
operate-on-gie-id=1
operate-on-class-ids=0;
config-file=config_infer_secondary_vehicletypenet.txt

[tests]
file-loop=1
```

5 Web urls (unsorted)

https://www.nvidia.com/de-de/data-center/products/ai-enterprise/vmware/
https://docs.microsoft.com/en-us/learn/modules/deploy-batch-inference-pipelines-with-azure-machine-learning/
https://developer.nvidia.com/vst-early-access
https://developer.nvidia.com/blog/building-multi-camera-media-server-ai-processing-jetson/
https://docs.microsoft.com/en-us/azure/virtual-machines/sizes-gpu
https://docs.microsoft.com/en-us/azure/databox-online/azure-stack-edge-gpu-overview
https://github.com/NVIDIA-AI-IOT/deepstream_python_apps/blob/master/apps/deepstream-rtsp-in-rtsp-out/README
https://github.com/NVIDIA-AI-IOT/deepstream_python_apps/tree/master/apps/deepstream-rtsp-in-rtsp-out
https://www.edge-ai-vision.com/2022/02/managing-video-streams-in-runtime-with-the-nvidia-deepstream-sdk/
https://docs.monai.io/projects/stream/en/latest/installation.html
https://forums.developer.nvidia.com/t/deepstream-sdk-faq/80236/15
https://forums.developer.nvidia.com/t/problem-while-attempting-use-gui-in-headless-mode/167307/34
https://github.com/latonaio/dashcamnet-on-deepstream
https://github.com/latonaio/ai-modeling-learning-materials
https://dream-soft.mydns.jp/blog/developper/smarthome/2020/12/2625/