Deploying ML solutions with low latency in Python

Hello!

I am Aditya Lohia

You can find me at:

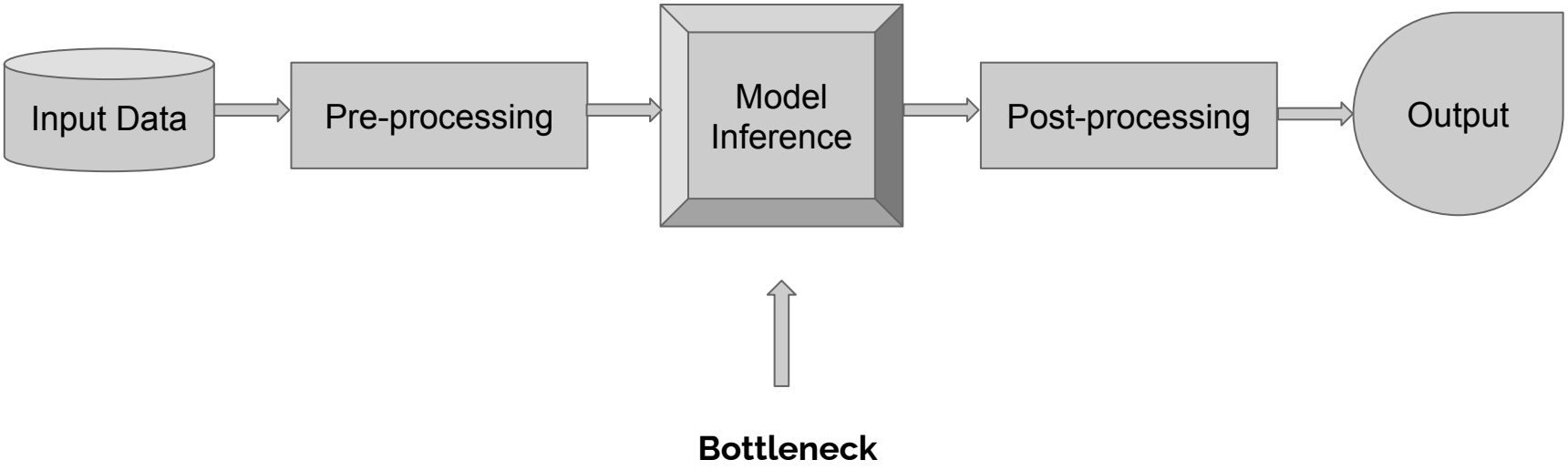
LinkedIn: https://www.linkedin.com/in/aditya-lohia-ml/

Website: http://www.adityalohia.me

As the world moves forward with the research in improving accuracies of deep learning algorithms, we face an imminent problem of deploying those algorithms.

1

What do I mean by low latency?


Input Data
Pre-processing
Model Inference
Post-processing
Output
Bottleneck

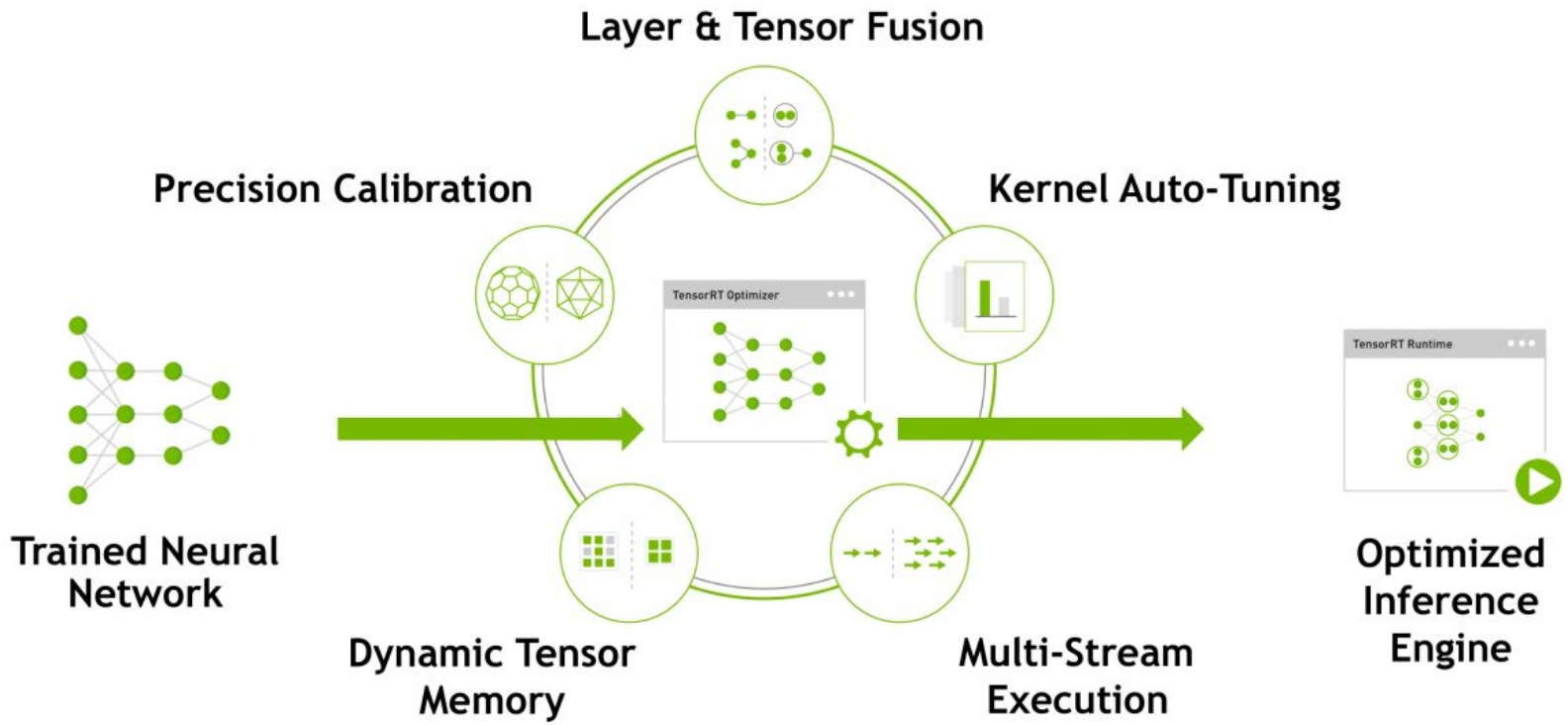
Methods to improve latency

1. Weight Quantization
2. Model Pruning
3. Knowledge Distillation
4. Framework based deployment

What is TensorRT?

- SDK for high-performance deep learning inference.
- Includes a deep learning inference optimizer and runtime.
- Delivers low-latency and high-throughput for deep learning inference applications.
- Supports both Python and C++
- Supports multiple deep learning frameworks such as TensorFlow, PyTorch, MXNet, Theano, etc.

Reference – NVIDIA TensorRT | NVIDIA Developer



Reference – NVIDIA TensorRT | NVIDIA Developer

Weight & Activation Precision Calibration

Maximizes throughput by quantizing models to INT8 while preserving accuracy

Layer & Tensor Fusion

Optimizes use of GPU memory and bandwidth by fusing nodes in a kernel

Kernel Auto-Tuning

Selects best data layers and algorithms based on target GPU platform

Dynamic Tensor Memory

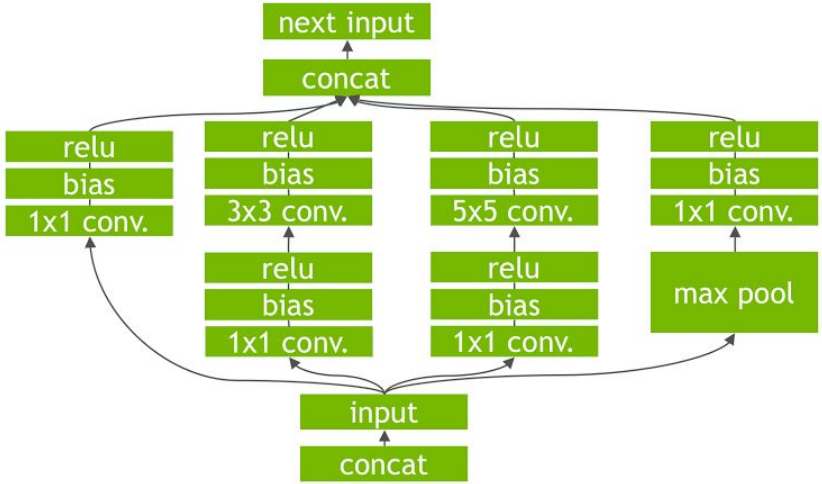
Minimizes memory footprint and re-uses memory for tensors efficiently

Multi-Stream Execution

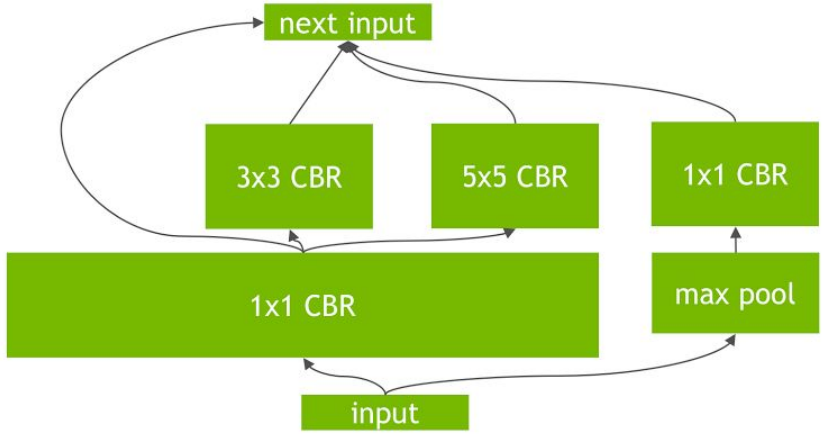
Scalable design to process multiple input streams in parallel

Reference – NVIDIA TensorRT | NVIDIA Developer

Un-Optimized Network



TensorRT Optimized Network



Network	Layers	Layers after Fusion
VGG19	43	27
Inception V3	309	113
ResNet-152	670	159

How does it work?

- Convert the pretrained PyTorch model into ONNX.
- Import the ONNX model into TensorRT.
- Apply optimizations and generate an engine.
- Perform inference on the GPU.

Metrics - RetinaFace

Models	Device	Batch Size	Mode	Input Shape (HxW)	FPS
RetinaFace (resnet50)	GTX1080	1	FP32	480x640	81
RetinaFace (resnet50)	GTX1080	1	INT8	480x640	190
RetinaFace (mobilenet0.25)	GTX1080	1	FP32	480x640	400

TensorRT

Metrics - YOLOv5

Models	Device	Batch Size	Mode	Input Shape (HxW)	FPS
YOLOv5-s	GTX1080	1	FP16	608x608	142
YOLOv5-s	GTX1080	4	FP16	608x608	170
YOLOv5-s	GTX1080	8	FP16	608x608	188
YOLOv5-m	GTX1080	1	FP16	608x608	67
YOLOv5-l	GTX1080	1	FP16	608x608	30

TensorRT

Best Practices

- Use Mixed Precision - FP32, FP16, and INT8
- Do not build engine for each inference (re-use the built engine)
- Change workspace size

Reference – NVIDIA TensorRT | NVIDIA Developer

Things to keep in mind

Engines generated are specific to the machine

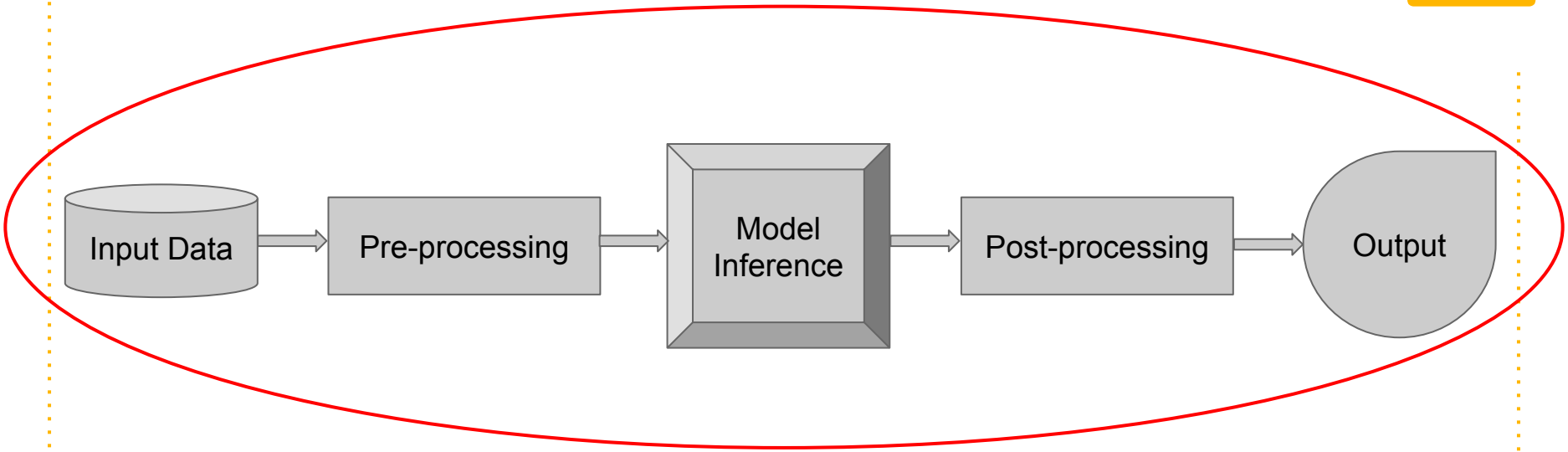
Installation takes time without docker

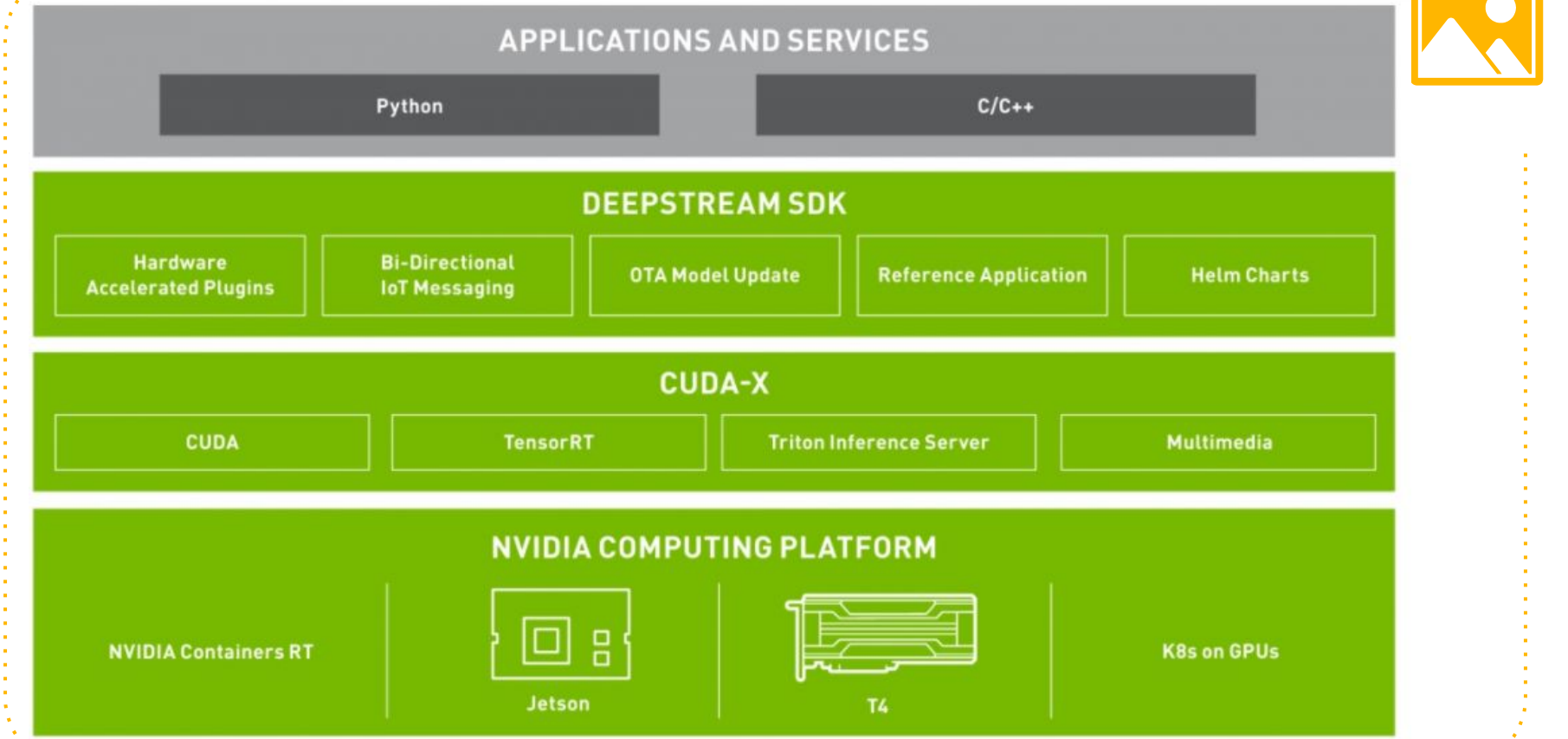
Multiple APIs for conversion

What is DeepStream?

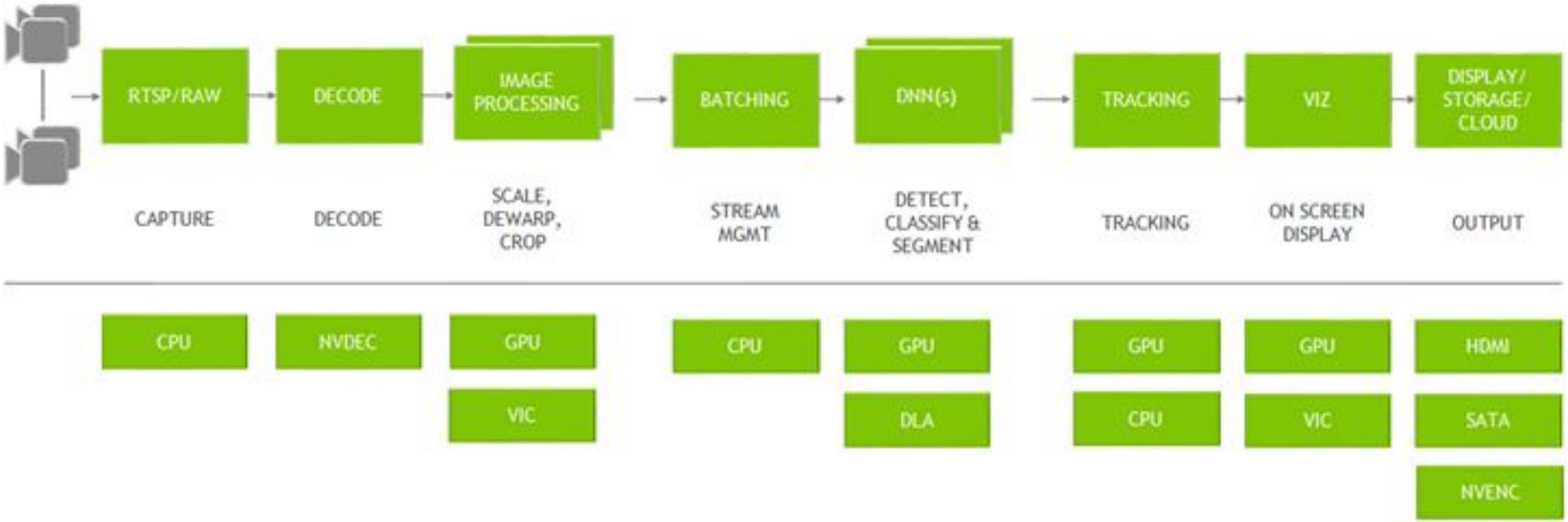
- Pipeline for deploying ML solutions.
- Multi-platform scalable framework with TLS security.
- Deploy on edge and connect to any cloud.
- Supports both Python and C++.
- Uses custom developed Gstreamer objects for GPU to speed-up pipelines.
- Supports deployment using multiple frameworks.

Reference – NVIDIA DeepStream SDK | NVIDIA Developer


Input Data
Pre-processing
Model Inference
Post-processing
Output



Reference – NVIDIA DeepStream SDK | NVIDIA Developer



Reference – Welcome to the DeepStream Documentation — DeepStream DeepStream Version: 5.0 documentation (nvidia.com)

				Jetson Nano	Jetson Xavier NX			Jetson AGX Xavier			T4
Model Architecture	Inference Resolution	Precision	Model Accuracy	GPU (FPS*)	GPU (FPS)	DLA1 (FPS)	DLA2 (FPS)	GPU (FPS)	DLA1 (FPS)	DLA2 (FPS)	GPU (FPS)
PeopleNet-ResNet18	960x544	INT8	80%	14	218	72	72	384	94	94	1105
PeopleNet-ResNet34	960x544	INT8	84%	10	157	51	51	272	67	67	807
TrafficCamNet-ResNet18	960x544	INT8	84%	19	261	105	105	464	140	140	1300
DashCamNet-ResNet18	960x544	INT8	80%	18	252	102	102	442	133	133	1280
FaceDetect-IR-ResNet18	384x240	INT8	96%	95	1188	570	570	2006	750	750	2520

Reference – NVIDIA DeepStream SDK | NVIDIA Developer

TensorRT Resources?

- Developer Guide :: NVIDIA Deep Learning TensorRT Documentation
- TrojanXu/yolov5-tensorrt: A tensorrt implementation of yolov5: https://github.com/ultralytics/yolov5
- NVIDIA/TensorRT: TensorRT is a C++ library for high performance inference on NVIDIA GPUs and deep learning accelerators. (github.com)
- tensorflow/tensorrt: TensorFlow/TensorRT integration (github.com)
- Working With TensorRT Samples :: NVIDIA Deep Learning TensorRT Documentation
- Catalog | NVIDIA NGC
- Creating an Object Detection Pipeline for GPUs | NVIDIA Developer Blog
- Working With TensorRT Samples :: NVIDIA Deep Learning TensorRT Documentation
- wang-xinyu/tensorrtx: Implementation of popular deep learning networks with TensorRT network definition API (github.com)

DeepStream Resources?

- NVIDIA DeepStream SDK | NVIDIA Developer
- DeepStream Getting Started | NVIDIA Developer
- Welcome to the DeepStream Documentation — DeepStream DeepStream Version: 5.0 documentation (nvidia.com)
- DeepStream Development Guide — DeepStream DeepStream Version: 5.0 documentation (nvidia.com)
- NVIDIA-AI-IOT/deepstream_python_apps: A project demonstrating use of Python for DeepStream sample apps given as a part of SDK (that are currently in C,C++). (github.com)
- Tag: DeepStream | NVIDIA Developer Blog

Thanks!

Find the code at -> aditya-dl/RetinaFace-TensorRT-Python: Deploy RetinaFace algorithm using TensorRT in Python (github.com)

Any questions?