DEVELOPER BLOG

BUILDING AN OBJECT DETECTION APPLICATION WITH NVIDIA NGC ON GPU-POWERED AWS INSTANCES

By: Shokoufeh Monjezi Kouchak, Ashish Sardana, Chintan Patel

Organizations across virtually every industry are taking advantage of AI to gather deeper insights, improve efficiencies, and increase customer satisfaction. Today, computer vision is used in medical industries to enhance physicians' decision making, in retail stores to get customer feedback by understanding their gestures, and in live-stock industry to track the wellbeing of farm animals. In this post, we will use computer vision to build a traffic analysis application.

LET'S GET STARTED

We will walk you through running an object detection model with NVIDIA Metropolis, an application framework that simplifies the development, deployment and scale of AI-enabled video analytics applications from edge to cloud.

While the instructions are specific to building a traffic analysis application, these instructions can be adopted to build other computer vision services using NVIDIA AI software such as models available from the NVIDIA NGC catalog.

The NGC catalog is the hub of GPU-optimized AI software including frameworks, industry-specific SDKs, pretrained models, and Jupyter notebooks. The software helps AI practitioners build, train, deploy, and run inference faster on any GPU-powered systems. To simplify the workflow for AWS users, we now offer the AI software through the NGC storefront in AWS Marketplace.

While building a traffic analysis software, there are three things to consider: 1. the volume of traffic, 2. the speed of movement, and 3. the type of vehicles. All three of the above heavily rely on two core techniques, object detection and tracking.

> Leverage Pretrained Models for Faster Development

Building a model from scratch is challenging and most organizations don't have the resources to build a model from the ground up because it requires GBs to TBs of labelled training data which can get very expensive. It also requires compute, training time, and AI expertise.

Pretrained models address this challenge. Using a method called transfer learning, you can retrain a pretrained model for your use case.

With transfer learning, you already have the core of the model to start with. You can train faster as you need to just retrain the output layers while leveraging the compute intensive core layers.

Retraining the output layers for your use cases requires just a fraction of the data and you benefit from the accuracy of the original model.

We will use the tlt_trafficcamnet model that detects one or more physical objects from four categories within an image and returns a box around each object, as well as a category label for each object.

The four categories of objects detected by this model are – cars, people, road signs and two-wheelers. The model is based on the NVIDIA DetectNet_v2 detector with ResNet18 as a feature extractor This architecture, also known as GridBox object detection, uses bounding-box regression on a uniform grid on the input image. The Gridbox system divides an input image into a grid which predicts four normalized bounding-box parameters (xc, yc, w, h) and confidence values per output class.

> NVIDIA DeepStream

The NVIDIA DeepStream SDK delivers a complete streaming analytics toolkit for AI-based video and image understanding and multi-sensor processing. DeepStream features hardware-accelerated building blocks, called plugins that bring deep neural networks and other complex processing tasks into a stream processing pipeline.

This container is for data center GPUs such as NVIDIA T4 running on x86 platform. There are data samples inside this container that you can use as an input of models including the above mentioned trafficcamnet. In this article, we use the DeepStream container hosted on AWS Marketplace.

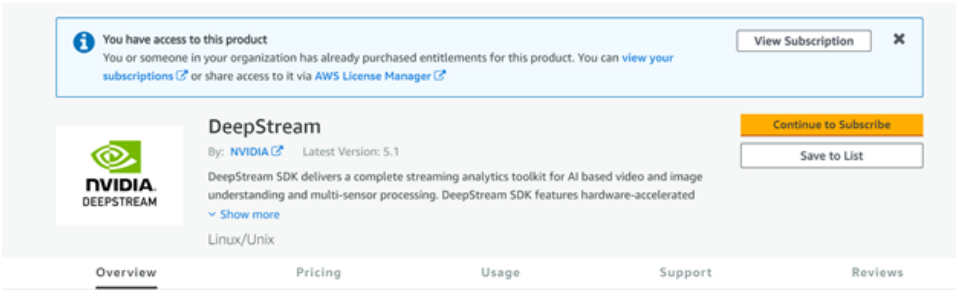
Requirements:

1. A server with 1 or more (preferably 8) NVIDIA A100's, either on cloud (AWS p4dn.24xlarge) or on-prem. A P3 instance works too.
2. NVIDIA Driver 460+

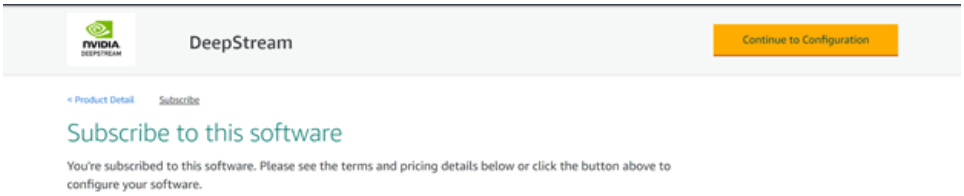
To run the model on AWS you can use an NVIDIA AMI which is optimized for running deep learning models and it comes with the NVIDIA driver installed. Follow the next steps to make an EC2 instance.

1. Launch an EC2 Instance:
    a. Log in to the AWS console
    b. Choose EC2 from Service
    c. Choose Marketplace from the menu on the left side of the page
    d. Choose an NVIDIA AMI from AWS Marketplace and make an EC2 instance with A100 GPUs (p4d.24xlarge) or P3 instance if you don't have access to the P4.

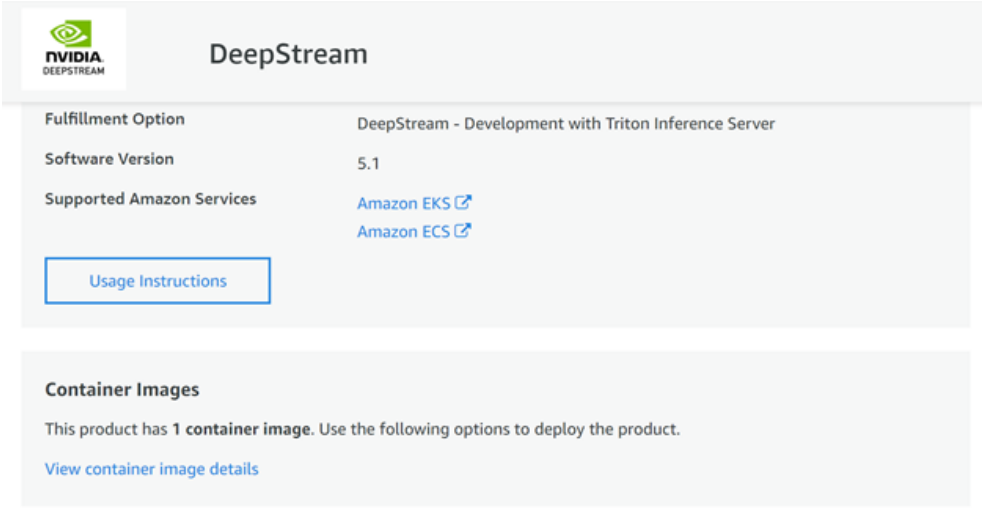
    e. Launch the instance
2. Pull an NVIDIA Container from AWS marketplace:
    a. Go to the DeepStream product page on the NGC storefront
    b. Subscribe to DeepStream on the top right side of the page

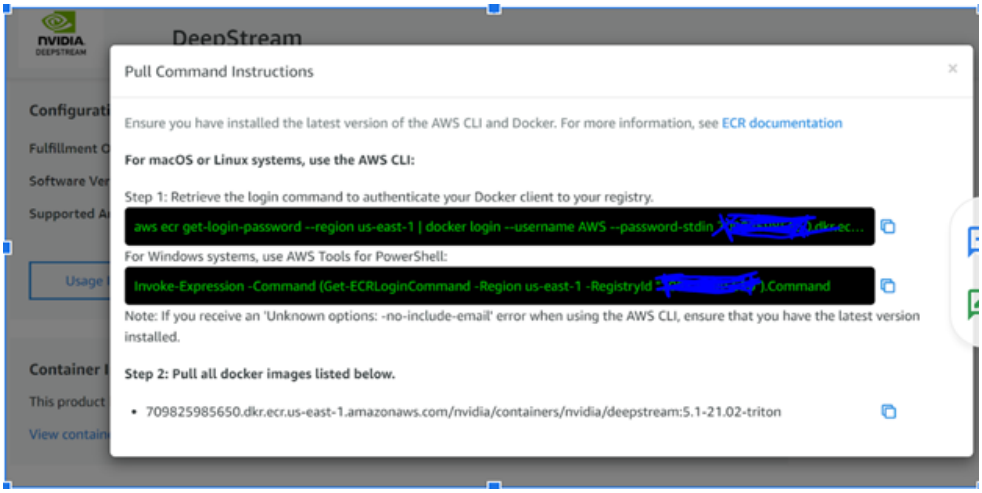


    c. Click Continue to Configure



d. You can find the pull command in the “View container image details” link





e. Retrieve the login command to authenticate your Docker client to your registry

```
aws ecr get-login-password --region us-east-1 | docker login
--username AWS --password-stdin
XXXXXXX.dkr.ecr.us-east-1.amazonaws.com
```

f. Pull the container

```
docker pull
XXXXXXXX.dkr.ecr.us-east-1.amazonaws.com/nvidia/containers/nvidia/
deepstream:5.1-21.02-triton
```

g. Run the container

```
docker run --gpus all --rm -it -p 8888:888
XXXXXXX.dkr.ecr.us-east-1.amazonaws.com/nvidia/containers/nvidia/d
eepstream:5.1-21.02-triton
```

h. Run the model

i. Download the TrafficCamNet model for object detection from the NGC catalog

```
cd
/opt/nvidia/deepstream/deepstream-5.1/samples/configs/tlt_pretrain
ed_models
```

```
# download trafficcamnet

mkdir -p ../../models/tlt_pretrained_models/trafficcamnet && \
    wget
https://api.ngc.nvidia.com/v2/models/nvidia/tlt_trafficcamnet/vers
ions/pruned_v1.0/files/resnet18_trafficcamnet_pruned.etlt \
       -O
../../models/tlt_pretrained_models/trafficcamnet/resnet18_traffic
amnet_pruned.etlt && \
    wget
https://api.ngc.nvidia.com/v2/models/nvidia/tlt_trafficcamnet/vers
ions/pruned_v1.0/files/trafficnet_int8.txt \
       -O
../../models/tlt_pretrained_models/trafficcamnet/trafficnet_int8.t
xt
```

i. Optimize the model for deployment

NVIDIA TensorRT is a model optimization engine and is designed to work in a complementary fashion with training frameworks such as TensorFlow, PyTorch, and MXNet. It focuses specifically on running an already-trained network quickly and efficiently on NVIDIA hardware. You can find detailed documentation here.

In order to optimize the traffic camnet model we downloaded, we will use trtexec, a tool which optimizes the network by combining layers and optimizing the kernel selection for improved latency, throughput, power efficiency, and memory consumption.

This tool is called internally by DeepStream before running the application. Thus, we would specify the precision (float16 or integer8) and the batch size (used to better manage the memory), and let DeepStream generate an optimized engine file.

j. Generate the TensorRT Engine

Run the next commands to move related config files to the config folder and make the TensorRT engine.

```
cd ~/..
git clone
https://github.com/shokoufeh-monjezi/deepstream_example.git
# move config files and run deepstream app for generating tensorrt
engines

cp
/deepstream_example/engine_bs/config_infer_primary_trafficcamnet.t
xt
/opt/nvidia/deepstream/deepstream-5.1/samples/configs/tlt_pretrain
ed_models/config_infer_primary_trafficcamnet.txt

cp
/deepstream_example/engine_bs/deepstream_app_source1_trafficcamnet
.txt
/opt/nvidia/deepstream/deepstream-5.1/samples/configs/tlt_pretrain
ed_models/deepstream_app_source1_trafficcamnet.txt

deepstream-app -c
/opt/nvidia/deepstream/deepstream-5.1/samples/configs/tlt_pretrain
ed_models/deepstream_app_source1_trafficcamnet.txt
```

k. Prepare sample videos

There are some video samples in the DeepStream container that you can use for running the model. The following commands show how to prepare these video samples.

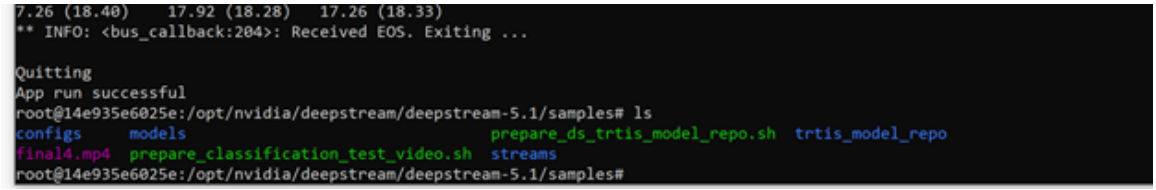
```
# prepare video samples
apt-get update && apt-get install -y ffmpeg

cd /opt/nvidia/deepstream/deepstream-5.1/samples/

./prepare_classification_test_video.sh

# move the engine, config.pbtxt and label files for trafficcamnet
cd /opt/nvidia/deepstream/deepstream-5.1/samples/

mkdir -p trtis_model_repo/trafficcamnet/1

cp
models/tlt_pretrained_models/trafficcamnet/resnet18_trafficcamnet_
pruned.etlt_b50_gpu0_int8.engine
trtis_model_repo/trafficcamnet/1/resnet18_trafficcamnet_pruned.etl
t_b50_gpu0_int8.engine

cp /deepstream_example/trafficcamnet_config.pbtxt
/opt/nvidia/deepstream/deepstream-5.1/samples/trtis_model_repo/tra
fficcamnet/config.pbtxt

cp /deepstream_example/labels_trafficcamnet.txt
/opt/nvidia/deepstream/deepstream-5.1/samples/trtis_model_repo/tra
fficcamnet/labels.txt

# move the model and app config file of the use-case
cp
/deepstream_example/config/config_infer_primary_trafficcamnet_trit
on.txt
/opt/nvidia/deepstream/deepstream-5.1/samples/configs/deepstream-a
pp-trtis/config_infer_primary_trafficcamnet_triton.txt
```

l. Deploy the model

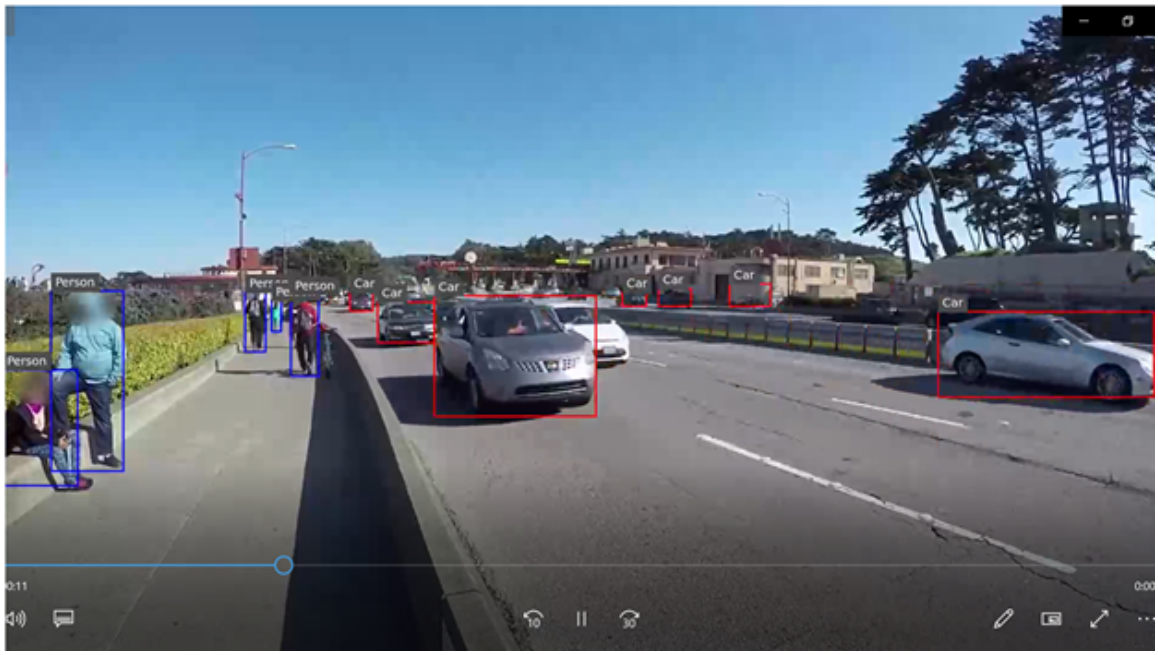
```
deepstream-app -c
/opt/nvidia/deepstream/deepstream-5.1/samples/configs/tlt_pretrain
ed_models/deepstream_app_source1_trafficcamnet.txt
```

RESULTS

The output of this model is a video named final4.mp4 that can be found in the samples directory at this address inside the running container:/opt/nvidia/deepstream/deepstream-5.1/samples.



The below is a frame of this video:



The TrafficCamNet model identifies objects on incoming video streams.

Click here to view the inference on the video stream.

SUMMARY

In this article, we walked you through steps of using NVIDIA resources on AWS to deploy an NVIDIA NGC model. Building AI solutions can get complicated, but the AI development tools and infrastructure available from NVIDIA and AWS can help simplify and speed up development.

Launch your EC2 instance and start building your object detection service with NVIDIA AI.

Ready to Get Started?

aws.amazon.com/marketplace/featured-seller/nvidia-ngc