AGENDA

1. Introduction: An Edge to Cloud Solution
2. Framework
3. Understand the Basics: DeepStream SDK
4. Build with DeepStream: Example Applications
5. Getting Started Resources
6. DeepStream 3.0
7. DeepStream in Docker
8. Perception and Analytics for Global Understanding
9. DeepStream for IOT
10. An Overview of TensorRT 5.0
11. Transfer Learning Toolkit
12. Metropolis Partner Program
EDGE-TO-CLOUD SOLUTION
把海量的摄像头更有意义的信息提取出来，并做进一步分析，能让我们更高效的查询、统计、管理、报警、自动化处理等等。
SCALABILITY - AN EDGE TO CLOUD ARCHITECTURE

Where deploy intelligence?

<table>
<thead>
<tr>
<th>CLOUD</th>
<th>1000s of cameras</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic management</td>
<td></td>
</tr>
<tr>
<td>Public safety</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ON-PREM SERVER / APPLIANCE</th>
<th>10s-100s of cameras</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smart building</td>
<td></td>
</tr>
<tr>
<td>Airport security</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CAMERA</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Parking entrance</td>
<td></td>
</tr>
<tr>
<td>Law enforcement</td>
<td></td>
</tr>
</tbody>
</table>
NVIDIA METROPOLIS — EDGE TO CLOUD

EDGE AND ON-PREMISES

- Camera
- NVR
- Server

TENSORRT • DEEPSTREAM

CLOUD

- Data center

JETSON

TESLA

TESLA
NVIDIA T4 UNIVERSAL INFERENCE ACCELERATOR

320 Turing Tensor Cores
2,560 CUDA Cores
65 FP16 TFLOPS
130 INT8 TOPS | 260 INT4 TOPS
16GB | 320GB/s

Programmable Acceleration at Scale

Inference
Next-Generation Inference with Turing Tensor Cores for FP16, INT8, INT4, INT1

Video & Graphics
2x User Density vs P4
2x Video Decode Capability vs P4

DL Training
Entry Level Training SKU with Turing Tensor Cores

T4 and Turing performance projections and specifications are preliminary and subject to change without notice.
JETSON AGX XAVIER
World’s first AI computer for Autonomous Machines

AI Server Performance in 30W • 15W • 10W
512 Volta CUDA Cores • 2x NVDLA
8 Core CPU
32 DL TOPS
## JETSON AGX XAVIER

<table>
<thead>
<tr>
<th></th>
<th>JETSON TX2</th>
<th>JETSON AGX XAVIER</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GPU</strong></td>
<td>256 Core Pascal</td>
<td>512 Core Volta</td>
</tr>
<tr>
<td><strong>DL Accelerator</strong></td>
<td>-</td>
<td>NVDLA x 2</td>
</tr>
<tr>
<td><strong>Vision Accelerator</strong></td>
<td>-</td>
<td>VLA - 7 way VLIW Processor</td>
</tr>
<tr>
<td><strong>CPU</strong></td>
<td>6 core Denver and A57 CPUs</td>
<td>8 core Carmel CPUs</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td>8 GB 128 bit LPDDR4 58.4 GB/s</td>
<td>16 GB 256 bit LPDDR4x 137 GB/s</td>
</tr>
<tr>
<td><strong>Storage</strong></td>
<td>32 GB eMMC</td>
<td>32 GB eMMC</td>
</tr>
<tr>
<td><strong>Video Encode</strong></td>
<td>2x 4K @ 30 HEVC</td>
<td>2x 4K @ 60 / 4x 4K @ 30 HEVC</td>
</tr>
<tr>
<td><strong>Video Decode</strong></td>
<td>2x 4K @ 30 12 bit support</td>
<td>2x 8K @ 30 / 8x 4K @ 30 12 bit support</td>
</tr>
<tr>
<td><strong>Camera</strong></td>
<td>Up to 6 cameras CSI2 D-PHY 1.2 2.5Gbps/lane</td>
<td>Up to 8 cameras CSI2 D-PHY 1.2 2.5 Gbps/lane</td>
</tr>
<tr>
<td><strong>Mechanical</strong></td>
<td>50mm x 87mm 400 pin connector</td>
<td>100mm x 87mm 699 pin connector</td>
</tr>
</tbody>
</table>

*New!*

x2

x4

+2
FRAMEWORK

Local Understanding

Global Understanding

Present Rich Insights
DEEPSTREAM FOR AI APPLICATION PERFORMANCE AND SCALE

* Deepstream focus on Perception.
* Provide efficient way to build VA applications
* DS3.0 for tesla, DS3.0 for jetson, unification for tesla/jetson, python binding.

* Support multi-streams, multi-neural-network, memory management.
* Let user focus on network training, optimization
UNDERSTAND THE BASICS: DEEPSTREAM SDK
PERCEPTION
# DEEPSTREAM SDK

## User Applications

<table>
<thead>
<tr>
<th>Plugins (build with open source, 3rd party, NV)</th>
<th>Flexible and scalable graphs</th>
<th>Development Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNN inference/TensorRT plugins</td>
<td></td>
<td>End to end reference applications</td>
</tr>
<tr>
<td>Communications plugins</td>
<td></td>
<td>App building/configuration tools</td>
</tr>
<tr>
<td>Video/image capture and processing plugins</td>
<td></td>
<td>Plugin templates and adaptation guides</td>
</tr>
<tr>
<td>3rd party library plugins ... ...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DeepStream SDK

- TensorRT
- Multimedia APIs/ Video Codec SDK
- Imaging
- Metadata Description

- Linux, CUDA
- Jetson, Tesla
1. Base on components
2. Data flow, control flow
## Component | Function
--- | ---
PLUGINS | Basic building block connected through PADs
BINS | A container for a collection of plugins
PIPELINE | Top level bin providing a bus and managing the synchronization
DEEPPSTREAM BUILDING BLOCK

- A plugin model based pipeline architecture
- Graph-based pipeline interface to allow high-level component interconnect
- Heterogenous processing on GPU and CPU
- Hides parallelization and synchronization under the hood
- Inherently multi-threaded
DEEPSTREAM GRAPH ARCHITECTURE

1. **Capture**
   - RTSP/RAW

2. **Decoding and Camera Processing**
   - DECODE/ISP
     - DECODE, CAMERA PROCESS
     - SCALE, DEWARP, CROP, ...

3. **Image Processing**
   - IMAGE PROCESSING

4. **Batching**
   - BATCHING

5. **Deep Neural Network Processing**
   - DNN(s) ....

6. **Tracking**
   - TRACKING

7. **Visualization**
   - VIZUALIZATION

8. **Display/Storage**
   - DISPLAY/STORAGE

**Components:**
- GigE
- MM
- GPU
- CPU
- DLA
- VA
- MM
- GPU
- CPU
- HDMI
- SATA
- GPU
- CPU
- CUDA
- CPU
METADATA STRUCTURE

- **NvDSObjectParams** - Contains a subset of metadata information for an object detected in the frame.
- **GIE_Unique_ID** - Multiple neural networks get assigned a unique ID.
- **Num_rects** - Number of objects detected in the frame.
- **Stream_Id** - In case of multi-stream, to identify we need a stream id to associate which stream the data belongs to.
METADATA STRUCTURE

- **NvOSD_RectParams** - Bounding box coordinates
- **NvOSD_TextParams** - Label information required for display (white car, Mercedes, sedan)
- **NvDSAttribinfo** - Attributes of objects (type, color, make)
- **Tracking_ID** - Unique ID of that object from tracker
- **Class_ID** - Type of object (person, vehicle, two-wheeler, road sign)
MEMORY MANAGEMENT

Efficient Memory Management

DeepStream App

nvdec_h264

nvinfer (PGIE)

nvtracker

nvinfer (SGIE)

CUDA HW Buffer sharing

Gstreamer Plugin

Allocates CUDA HW Buffers (GPU Memory)

Uses shared CUDA HW (GPU Memory) buffers

Returns CUDA HW Buffer for reuse
MEMORY MANAGEMENT

GPU to CPU Copy

DeepStream App

gst-nvdecode

Gst-nvinfer

Gst-nvvidconv

Gst-xvimagesink

Efficient Buffer Copy

cudaMalloc

CUDA HW Buffer sharing

DtoH Buffer Copy

CPU Buffer sharing

Returns CUDA HW Buffer for reuse

Allocates CUDA HW (GPU memory) Buffers

Uses shared CUDA HW (GPU memory) buffers

Allocates CPU Buffers

Uses Shared CPU Buffer
**DECODER PLUGIN**

gst-nvvideocodecs

- **Encoded Stream**
- **gst-nvvideocodecs**
- **YUV**
- **NVDECODE API**
- **Low Level API**
- **NV12**
- **GPU**

**Input**
- H.264, H.265, VP8, VP9, MPEG2/4

**Output**
- NV12

**Parameters**
- Bit rate control, i-frame decoding
## BATCHING

**GstNvStreamMux**

<table>
<thead>
<tr>
<th>Input</th>
<th>NV12 / RGBA streams</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>NV12/RGBA Batched Buffer Metadata containing information about input frames: original timestamps, frame numbers</td>
</tr>
</tbody>
</table>
| Parameters     | Batch size (int) - Number of buffers in a batch  
Batch timeout (int) - Time in microseconds to wait to form a batch  
Width, Height (int) - Scaling factor for source frames  
Frame padding (int) - Maintain source aspect ratio by padding with black bands |
BATCHING - GSTNVSTREAMMUX

UPSTREAM

Buffers Input

Decoder / Camera 1

Decoder / Camera 2

Decoder / Camera...

BATCHING

Batching Round robin

Batch of N buffers of size (W x H)

Scaling

DOWNSTREAM

NV12/RGBA Batched Buffer and NvStreamMeta
## NVIDIA-ACCELERATED PLUGINS

<table>
<thead>
<tr>
<th>Plugin Name</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>gst-nvvideocodecs</td>
<td>Accelerated video decoders</td>
</tr>
<tr>
<td>gst-nvstreammux</td>
<td>Stream aggregator - muxer and batching</td>
</tr>
<tr>
<td>gst-nvinfer</td>
<td>TensorRT based inference for detection &amp; classification</td>
</tr>
<tr>
<td>gst-nvtracker</td>
<td>Reference KLT tracker implementation</td>
</tr>
<tr>
<td>gst-nvosd</td>
<td>On-Screen Display API to draw boxes and text overlay</td>
</tr>
<tr>
<td>gst-tiler</td>
<td>Renders frames from multi-source into 2D grid array</td>
</tr>
<tr>
<td>gst-eglglesink</td>
<td>Accelerated X11 / EGL based renderer plugin</td>
</tr>
<tr>
<td>gst-nvvidconv</td>
<td>Scaling, format conversion, rotation</td>
</tr>
<tr>
<td>Gst-nvdewarp</td>
<td>Dewarping for 360 Degree camera input</td>
</tr>
<tr>
<td>Gst-nvmsgconv</td>
<td>Meta data generation</td>
</tr>
<tr>
<td>Gst-nvmsgbroker</td>
<td>Messaging to Cloud</td>
</tr>
</tbody>
</table>
BUILD WITH DEEPSTREAM: EXAMPLE APPLICATIONS
SMART PARKING SOLUTIONS WITH DEEPSTREAM

* 150 fish eye cameras      * identify open parking spots, LPR, tracking
* identify anomalies like a broken car in the aisle, overtime,
Detection and classification

PERCEPTION GRAPH

COMM PLUGIN | PREPROCESSING PLUGINS | DETECTION, CLASSIFICATION & TRACKING PLUGINS | COMMUNICATIONS PLUGINS
---|---|---|---
RTSP | Decoder | Dewarp library | Camera calibration | ROI calibration | Global positioning | Tracker | Transmit Metadata | Analytics server

**360d feeds**

Dewarping

**ROI: Lines**

Detection network: 2 classes, car(for object tracking) / car-front-back(for spot detection)
Gst-nvbboxfilter keeps only car-front-back objects
Gst-nvspot determines occupied or empty
Gst-nvmsgconv, gst-nvmsgbroker

**ROI: Polygon**
GETTING STARTED
RESOURCES
MULTI-STREAM REFERENCE APPLICATION
CONFIGURATION FILE

enable-perf-measurement=1  // To enable performance measurement
perf-measurement-interval-sec=10  // Sampling interval in seconds for performance metrics
flow-original-resolution=1  // Stream muxer flows original input frames in pipeline
#gie-kitti-output-dir=/home/ubuntu/kitti_data/  // Location of KITTI metadata files

[source0]
enable=1  // Enables source0 input
#Type - 1=CameraV4L2 2=URI 3=MultiURI  // 1) Input source can be USB Camera (V4L2)
  // 2) URI to the encoded stream. Can be a file, HTTP URI or an RTSP live source
  // 3) Select URL from multi-source input
type=3  // Type of input source is selected
uri=file:///..../streams/sample_720p.mp4  // Actual path of the encoded source.
num-sources=1  // Number of input sources.
gpu-id=0  // GPU ID on which the pipeline runs within a single system

PERFORMANCE ANALYSIS

- Measure KPIs & identify gaps
  - throughput, supported stream count

- Utilization information to understand bottlenecks
  - nvidia-smi

- Latency measurements to further debug and identify rate limiting processes
  - using gst-logs

- Kernel execution profiling for fine grained analysis
  - using nsight, nvvp
ONLINE RESOURCES

- Gstreamer Plugin and Application Development Guide
  - https://gstreamer.freedesktop.org/documentation/
- NVIDIA DeepStream SDK
- Blog
- Webinar:
DEEPSTREAM 3.0
# DEEPSTREAM SDK 3.0

## Plugins (build with open source, 3rd party, NV)
- DNN inference/TensorRT plugins
- Communications plugins
- Video/image capture and processing plugins
- 3rd party library plugins ...

## Analytics - multi-camera, multi-sensor framework
- DeepStream in containers, Multi-GPU orchestration
- Tracking & analytics across large scale/ multi-camera
- Streaming and Batch Analytics
- Event fabric

## Development Tools
- End to end reference applications
- App building/configuration tools
- End-end orchestration recipes & adaptation guides
- Plugin templates, custom IP integration

## Linux, CUDA
- TensorRT
- Multimedia APIs/Video Codec SDK
- Imaging & Dewarping library
- Metadata & messaging
- NV containers
- Message bus clients
- Multi-camera tracking lib

## Perception infra - Jetson, Tesla server (Edge and cloud)
- Capability, scalability, and portability
- More plugins
- More samples, source code
- Support Docker, TRT5.0

## Analytics infra - Edge server, NGC, AWS, Azure
DEEPSTREAM 3.0 NEW FEATURES

- New model support: ONNX models, UFF models, and the models exported from Magnet SDK.
- Multistream batching example: This example shows how to run DeepStream SDK with multiple input streams.
- Support for TensorRT Iplugin Creator interface.
  - New SSD Example.
- Support for TensorRT IPluginFactory interface.
  - New faster RCNN example.
DEEPSTREAM 3.0 NEW FEATURES

➢ IOU based tracker algorithm.

➢ Dynamic stream management: Supports on-the-fly addition and deletion of input sources, variable frame rate streams, and dynamic resolution changes.

➢ Dewarper plugin: This new plugin dewarps the frames from 360D view cameras.

➢ New plugins gst-nvmsgconv (payload generation) and gst-nvmsgbroker (payload transmission).

➢ Message broker: Supports Meta generation and transmission to the Cloud.
ENABLING 360D CAMERA PROCESSING

NVWARP360 SDK

- Equirectangular
- Cylindrical
- Panini
- Perspective
- Rotated cylinder
- Pushbroom
SEAMLESS PLUG-AND-PLAY IN DEEPSTREAM

### nvdewarper

<table>
<thead>
<tr>
<th>Input</th>
<th>360D RGBA frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>RGBA buffer for each surface/projections. Projections: Pushbroom, Rotated cylinder, Perspective, Equirectangular, Panini, Cylindrical</td>
</tr>
<tr>
<td>Parameters</td>
<td>Number of Dewarping surface per frame, each surface width and height, projection type for each surface, dewarping angles like top, bottom, yaw, roll etc.</td>
</tr>
</tbody>
</table>
DYNAMIC STREAM MANAGEMENT

Application

1. Add/ Remove camera streams
2. Change FPS
3. Change resolutions
AN ALL NEW INFERENCE PLUGIN
gst-nvinfer

**Input**
- NV12/RGVA buffer, Model files - Caffe Model and Caffe Prototxt, ONNX, UFF file, TRT plugin-factory, plugin-creator

**Output**
- Original metadata + NvDsMeta attached by this element (class, bbox, etc)

**Parameters**
- Batch size, inference interval, clustering params, class threshold, Bbox color, width & height of Bbox to filterout some boxes for downstream component etc.
GSTNVINFER

PREPROCESSING
- Convert and/or Scale to network input resolution & color format
- Mean subtraction and scaling
- Optionally crop primary detected objects for secondary inferencing

TRT Inference

Bounding box parsing for clustering

TensorRT IPlugin implementation for custom layers

Custom BBOX parsing function

Lib containing IPlugin implementation

Unmodified input buffer
DEEPSTREAM IN DOCKER
DEEPSTREAM IN DOCKER

DeepStream Container
- DeepStream Applications
- DeepStream
- TensorRT
- Video Codec SDK
- CUDA
- Container OS User Space

CONTAINER 1
- Applications
- CUDA Toolkit
- Container OS User Space
- Docker Engine
- CUDA Driver
- Host OS
- NVIDIA GPUs

CONTAINER N
- Applications
- CUDA Toolkit
- Container OS User Space
- Docker Engine
- CUDA Driver
- Host OS
- NVIDIA GPUs

Server
MULTIPLE PERCEPTION GRAPHS

HETEROGENEOUS APPLICATIONS

Video Files

16 streams 720p 30 fps

Traffic Graph
Primary Detector: Car, Bike, Road-sign, Person
3 Secondary Detectors: Vehicle Type, Make, Color

On Screen Display

Pedestrian Graph
Primary Detector: Person, Face, Bag
2 Secondary Detectors: Face and Bag attributes

On Screen Display

4 streams 720p 30 fps

DOCKER CONTAINERS
ARCHITECTURE FOR ANALYTICS

MESSAGING/ EVENTS
- Message Bus
- Extract & Transform

STREAM LAYER
- Counting
- Occupancy & State
- RT Alerts ...

BATCH LAYER
- Indexing & Search
- Heat maps ...

SERVING LAYER
- REST API

DATA STORAGE
- Raw data
- Results

Lambda architecture:兼顾可靠性实时性

REST API 实现一个后台server处理不同client端的http请求，比如手机、平板、PC等
DEEPSTREAM 3.0 END-TO-END APPLICATION

Kafka protocol: 是一个基于分布式的消息发布-订阅系统, 低耦合, 高性能, 数据不丢失
noSQL: 非关系数据库统称, 超大规模存储, 无固定模式, 可横向扩展, 快速
COMMAND CENTER UI

Browser-Based Interface

- Occupancy Map for Multiple Levels
- Sensor Fusion
- Anomalies
- Search Events and Anomalies
- Occupancy Stats and Flow Rates
REFERENCE ANALYTICS APPLICATION

STREAMING ANALYTICS

STORAGE

BATCH ANALYTICS

DATA STORE & INDEXING

VISUALIZATION

UI

Anomaly Detector

Movement Heatmap

User Defined 1

Anomaly Detector

RESTAPI / WEB SOCKETS

* Spark framework 是基于内存计算的大数据并行计算框架，能保证大数据环境下数据处理的实时性，容错性和可伸缩性

* Elasticsearch 是一个开源的 Java 开发的企业级搜索引擎，建立在全文搜索引擎库 Apache Lucene 基础之上
ANALYTICS DASHBOARD

基于 Kibana 一款开源的数据分析和可视化平台
PERCEPTION AND ANALYTICS IN CONTAINERS
SCALING AND PORTABILITY

PERCEPTION SERVER

DeepStream

Metadata

DeepStream

Metadata

Message Broker
Kafka

ANALYTICS SERVER

Multi Camera Tracking / Smoothing
Python Module

Anomaly Detection
Occupancy State

Apache Spark

ETL
Logstash

Parking Garage State Mgmt
Cassandra

Search Indexer
ElasticSearch

API

Analytics Dashboards
Kibana

Browser Apps

ETL
PERCEPTION AND ANALYTICS
Open source code


-> DeepStream 360-degree smart parking reference application

Analytics code:
https://github.com/NVIDIA-AI-IOT/deepstream_360_d_smart_parking_application

Blog:
https://devblogs.nvidia.com/multi-camera-large-scale-iva-deepstream-sdk/
DEEPSSTREAM FOR IOT
METADATA TO MESSAGE BROKERS

**gst-nvmsgtransform & gst-nvmsgbroker**

<table>
<thead>
<tr>
<th>Input</th>
<th>Metadata</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>Message sent over supported protocol</td>
</tr>
<tr>
<td>Parameters</td>
<td>Path to shared library implementing message generation from metadata based on schema</td>
</tr>
<tr>
<td></td>
<td>Protocol, URL, port, topic for message destination</td>
</tr>
<tr>
<td></td>
<td>Path to shared library implementing adaptor for desired protocol</td>
</tr>
</tbody>
</table>
DEEPSTREAM WITH AWS GREENGRASS

AWS Greengrass 提供了可部署于本地执行的应用程序的云的管理
DEEPSTREAM WITH AZURE IOT

EDGE APPLIANCE

DeepStream Application

DeepStream graph

Azure CLOUD

Azure Services

IOT Hub

CNTL ML & DL

Data Lake

Apache Kafka

Apache Spark

NpSQL

Storage Adapters

Search Indexer

Search & Query

Web Client

Web Client
METADATA DESCRIPTION SCHEMA

- Interoperability, orchestration & integration for intelligent video analytics
  - Describes observations of “Things” perceived in video using JSON messages and an extensible ontology
  - Things can either be “objects”, “places” or “events” at the highest level
- Also allows description of sensors and analytical engines used
- Time, location as primary attributes
- Supports relationships
  - HAS-A relationship, IS-A relationship...

To Be Open-Sourced with the Release
NVIDIA TensorRT
From Every Framework, Optimized For Each Target Platform
TENSORRT OVERVIEW

High-performance Deep Learning Inference Engine for Production Deployment

Data
Data Tools
Labeler

Training
Training Frameworks
Caffe
Theano
TensorFlow

Deployment
Inference Frameworks
TensorRT
Optimized Runtime Engine

We Are Here

Manage / Augment
Deep Neural Network
Data Center / Automotive / Embedded
TENSORRT DEPLOYMENT WORKFLOW

Step 1: Optimize trained model

Trained Neural Network → Import Model → TensorRT Optimizer → Serialize Engine → Optimized Plans

Step 2: Deploy optimized plans with runtime

Optimized Plans → De-serialize Engine → TensorRT Runtime Engine → Deploy Runtime
TENSORRT OPTIMIZATIONS

➢ Optimizations are completely automatic
➢ Performed with a single function call

```python
engine = trt.utils.uff_to_trt_engine(G_LOGGER,
    uff_model,
    parser,
    INFERENCES_BATCH_SIZE,
    1<<20,
    trt.infer.DataType.FLOAT)
```
TensorRT INTEGRATED WITH TensorFlow
Delivers 8x Faster Inference with TensorFlow + TRT

Available in TensorFlow 1.7
https://github.com/tensorflow

Images/sec @ 7ms Latency
ResNet-50 on TensorFlow

* Best CPU latency measured at 83 ms

CPU: Skylake Gold 6140, 2.5GHz, Ubuntu 16.04; 18 CPU threads.
Volta V100 SXM; CUDA (384.111; v9.0.176);
Batch size: CPU=1, TF_GPU=2, TF-TRT=16 w/ latency=6ms
TRANSFER LEARNING TOOLKIT

PRE-TRAINED MODEL

DATA

PRE-TRAINED MODEL ➔

PYTHON API

EVALUATION

PRUNING

RE-TRAINING

EXPORT

ADD CLASSES

SCENE ADAPTATION

PRUNE

OUTPUT MODEL

TRANSFER LEARNING WORKFLOW

1. Download docker container
2. Pull Model
3. Train with your data
4. Prune
5. Retrain
6. Evaluation
7. Export

Training config: Network choice, sgd/adam, batch-size, epochs, learning rate step decay, regularize, BatchNorm, dropout, bbox cluster/dbscan
Data augmentation and Dataloader
1. Reduce model size and increase throughput
2. Incrementally retrain model after pruning to recover accuracy

PRUNING

Prune  Retrain

$tlt$-prune  $tlt$-train
PRUNING EXAMPLE

6.5x reduction in model size

Network - ResNet18 4-class (Car, Person, Bicycle, Roadsign)
Memory size - 46.2 MB to 6.7 MB
FPS - 16fps to 30 fps

2x increase in throughput
SCENE ADAPTATION

Camera location vantage point

Train with new data from another vantage point, camera location, or added attribute
ADDING A NEW CLASS

Easy to edit models to add new class

Pre-trained 6 class GoogleNet classifying sedan, suv, truck, van, coupe, large vehicles

Add Emergency vehicle as another class simply by editing the spec file

Train, Prune and Retrain new model using new spec file

New network with 7 classes classifying sedan, suv, truck, van, coupe, large vehicles, emergency vehicles

tlt-train classification -e new_spec -r results -m model_file -n model_name
AVAILABILITY ON NVIDIA GPU CLOUD

Built on top of Tensorflow Docker Container
THE COMPLETE WORKFLOW
THE COMPLETE AI SOLUTION WORKFLOW FOR SMART CITIES

Train with TRANSFER LEARNING TOOLKIT

Optimize with TENSORRT

Deploy with DEEPSTREAM
FRAMEWORK
PERCEPTION -> ANALYTICS

Applications

Analytics for situational awareness
- Anomaly Detection
- Trajectory Analysis
- Metadata Search
- Spatio-temporal Analysis
- Sensor Fusion

Perception for scene understanding
- Object Detection
- Object Classification
- Object Tracking
- Actions and Behavior

Cameras & Sensors

Global view

Local view
谢谢大家！