JETSON ORIENTATION
A.I. LEARNING LAB

Dustin Franklin
Developer Relations

NVIDIA

FIRST Robotics Day

Thursday August 4, 2016

Workshop Agenda

Topic:

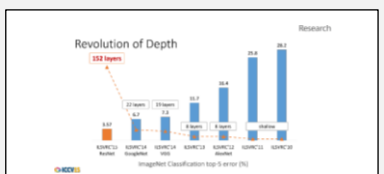
- Background on Jetson TX1
- Fundamentals of Deep Learning
- Deploying Realtime Networks with GPU Inference Engine

- Reinforcement Learning
  - Why?
  - How?
- Conclusion / Packing Up

MILESTONES IN AI

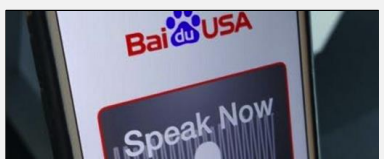
IMAGENET

Microsoft & Google
“Superhuman” Image
Recognition

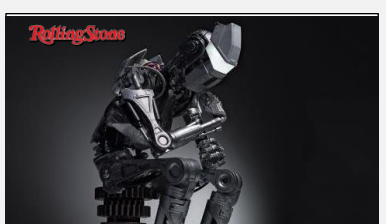


Microsoft
“Super Deep Network”

Berkeley’s Brett
End-to-End
Reinforcement Learning



Deep Speech 2
One network, 2 languages

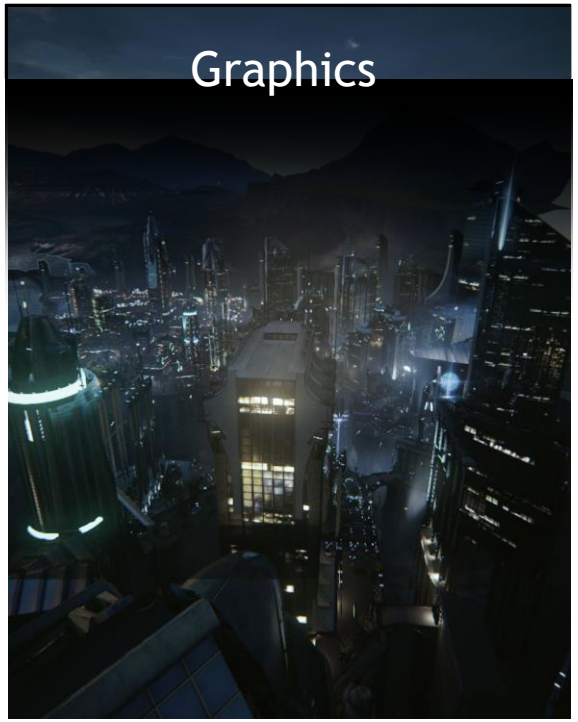


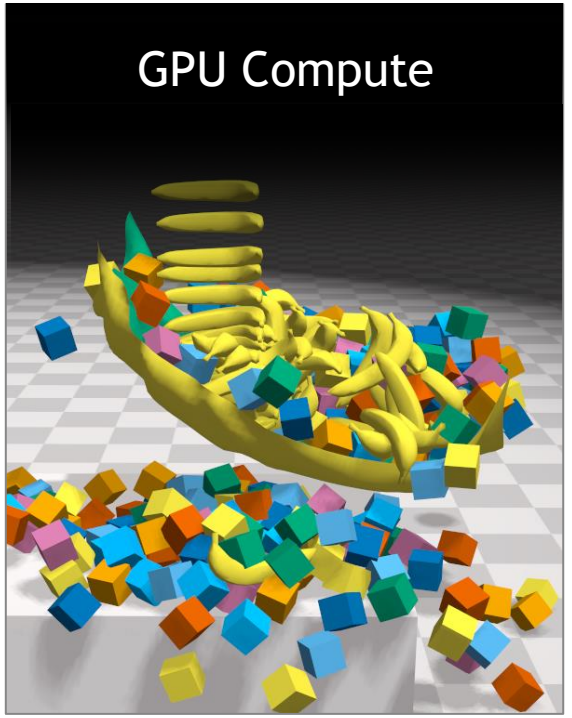
A New Computing Model
Hits Pop Culture

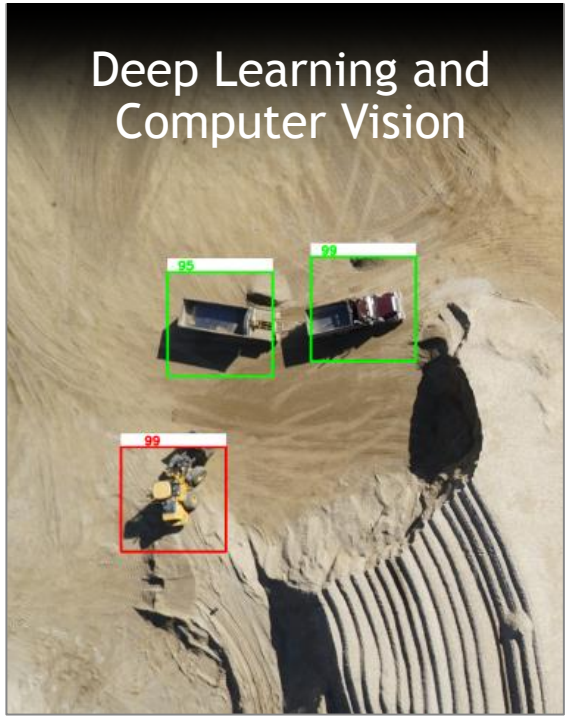
AlphaGo
Rivals a World Champion

TU Delft Deep-Learning
Amazon Picking Champion

NVIDIA GPU: MORE THAN GRAPHICS







DIRECTX

OpenGL

Vulkan

NVIDIA CUDA

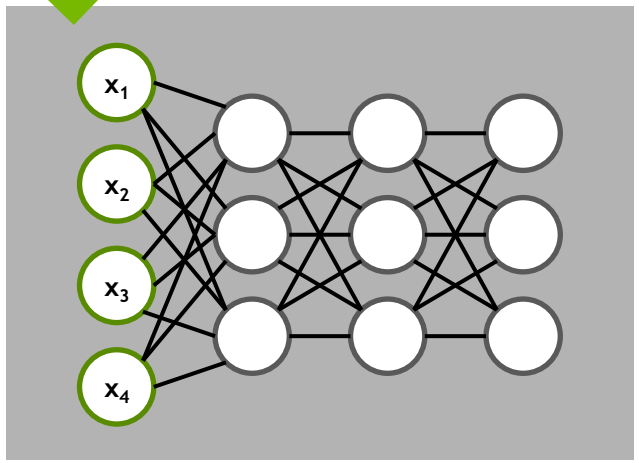
cuDNN

NVIDIA VISIONWORKS

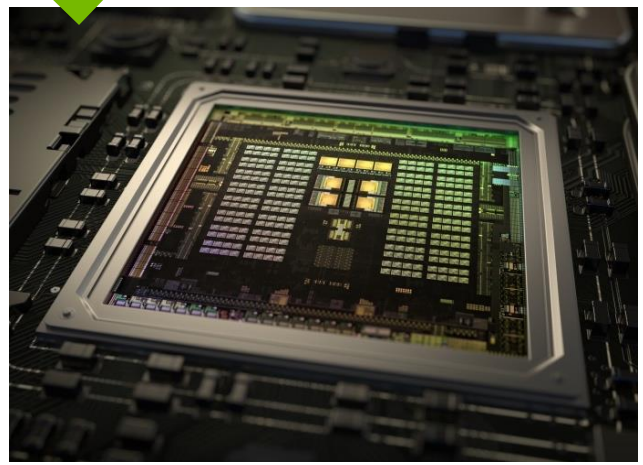
AT THE FRONTIER OF AUTONOMOUS MACHINES

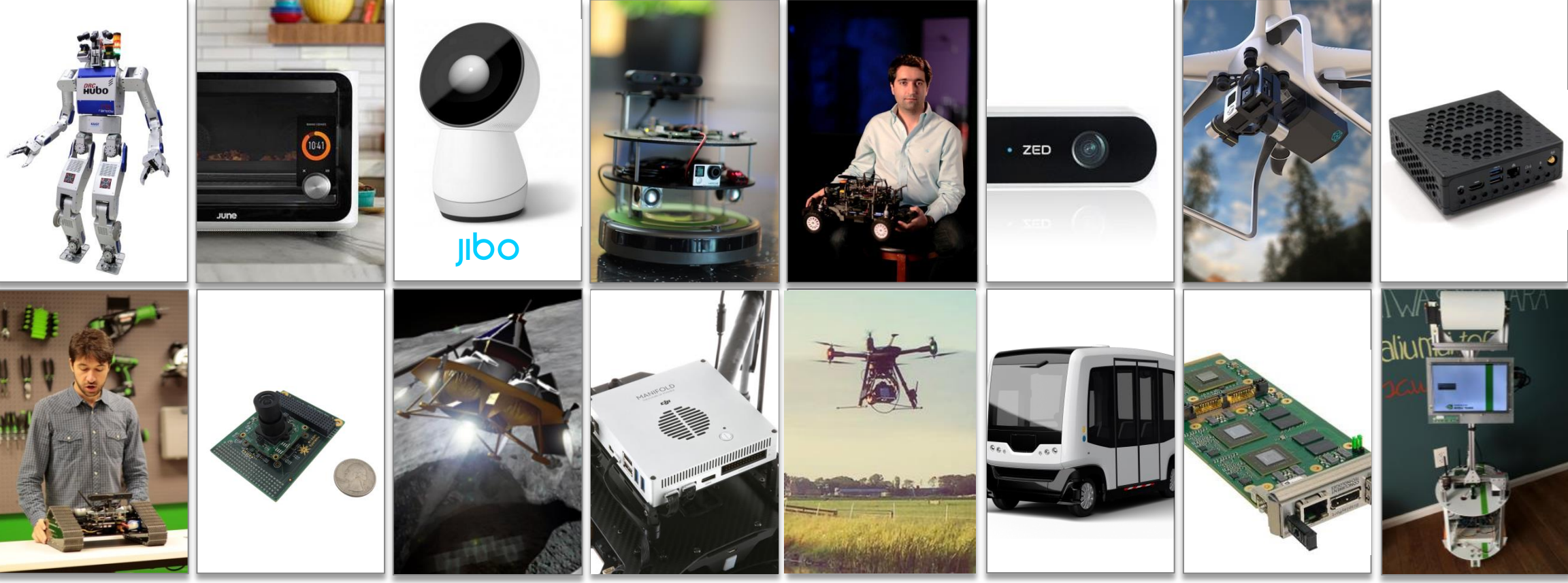
New use cases demand autonomy

Onboard sensing and deep learning, enable autonomy



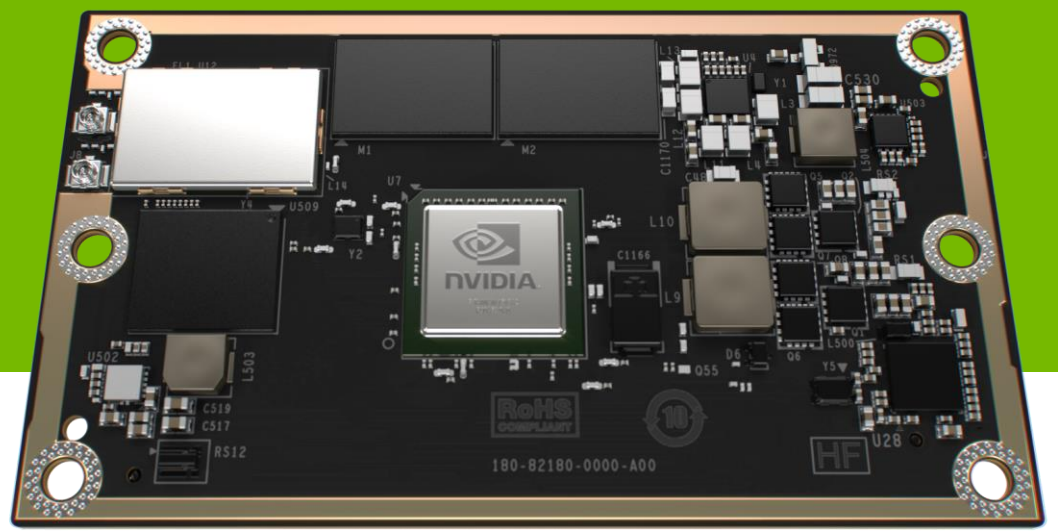
GPUs deliver superior performance and efficiency





JETSON AT THE LEADING EDGE

Powering the Next Generation of Autonomous Machines



Jetson TX1

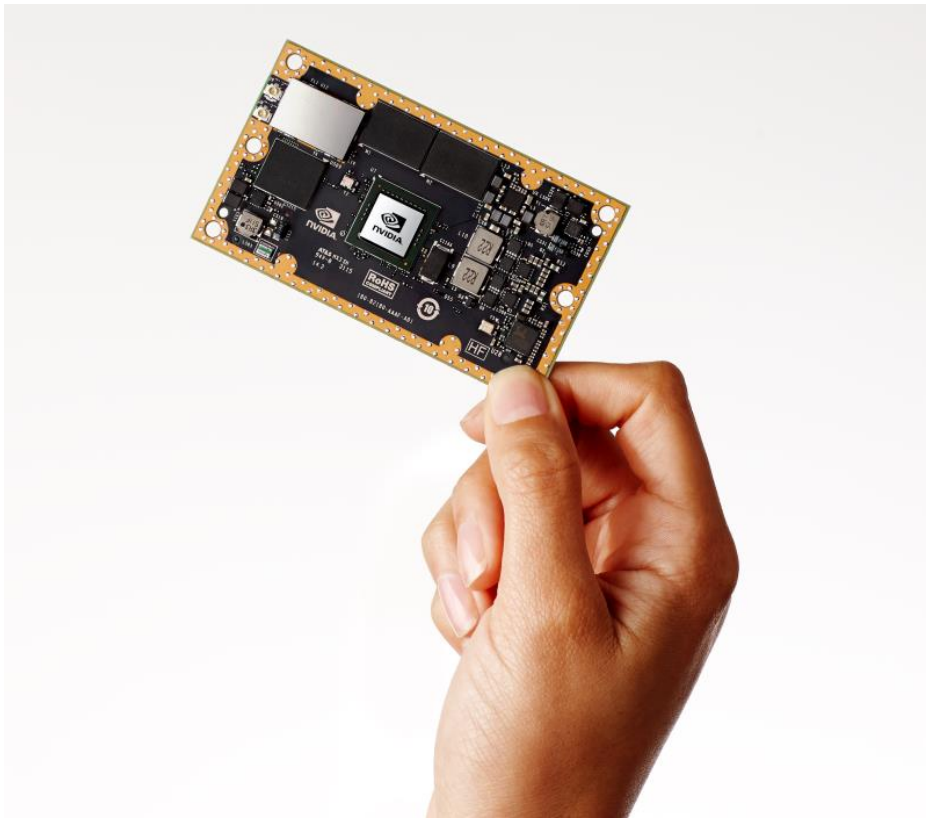
Supercomputer on a Module

Unmatched performance under 10W

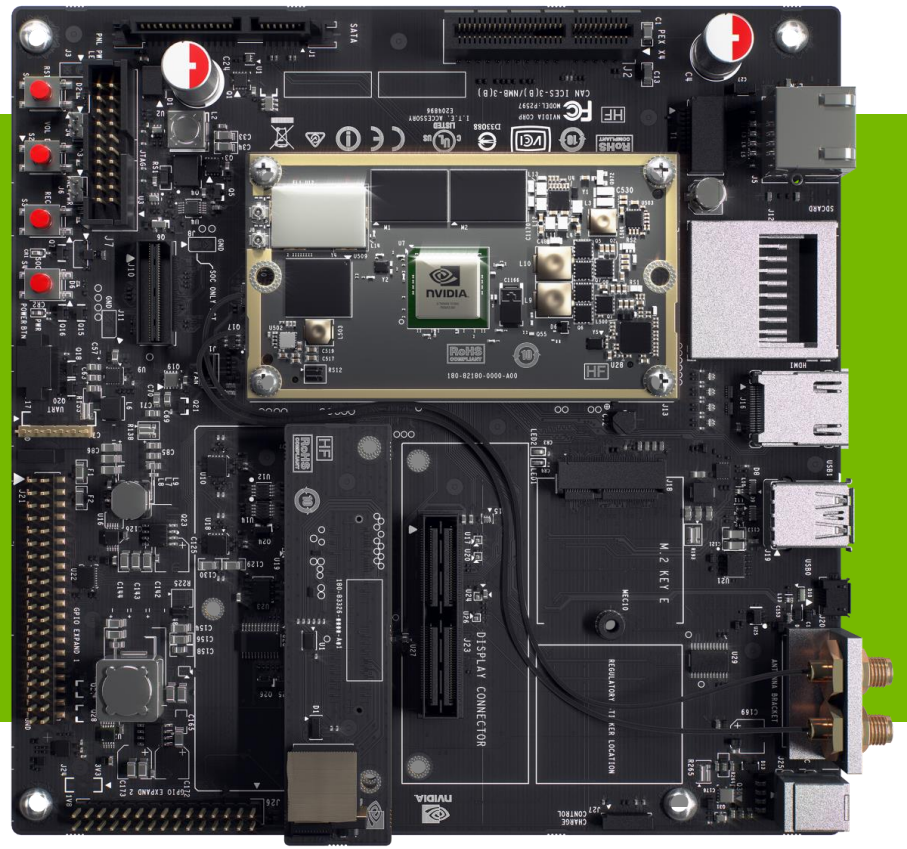
Advanced tech for autonomous machines

Smaller than a credit card

System on Module



	JETSON TX1
GPU	1 TFLOP/s 256-core Maxwell
CPU	4x 64-bit ARM A57 CPUs | 1.6 GHz
Memory	4 GB LPDDR4 | 25.6 GB/s
Video decode	4K 60Hz H.264
Video encode	4K 30Hz H.264
CSI	Up to 6 cameras | 1400 Mpix/s
Display	2x DSI, 1x eDP 1.4, 1x DP 1.2/HDMI
Wi-Fi	802.11 2x2 ac
Networking	1 Gigabit Ethernet
PCI-E	Gen 2 1x1 + 1x4
Storage	16 GB eMMC, SDIO, SATA
Other	3x UART, 3x SPI, 4x I2C, 4x I2S, GPIOs
Power	10-15W, 6.6V-19.5VDC
Size	50mm x 87mm

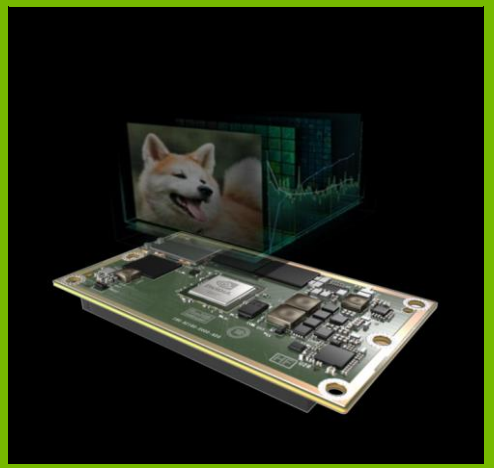


Jetson TX1 Developer Kit

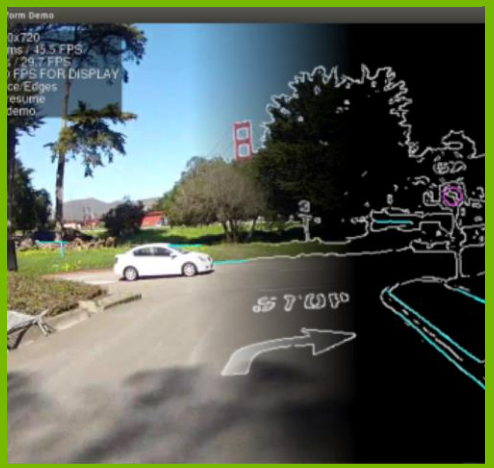
Jetson TX1
Developer Board
5MP Camera

NVIDIA JETPACK

Deep Learning SDK



DIGITS Workflow



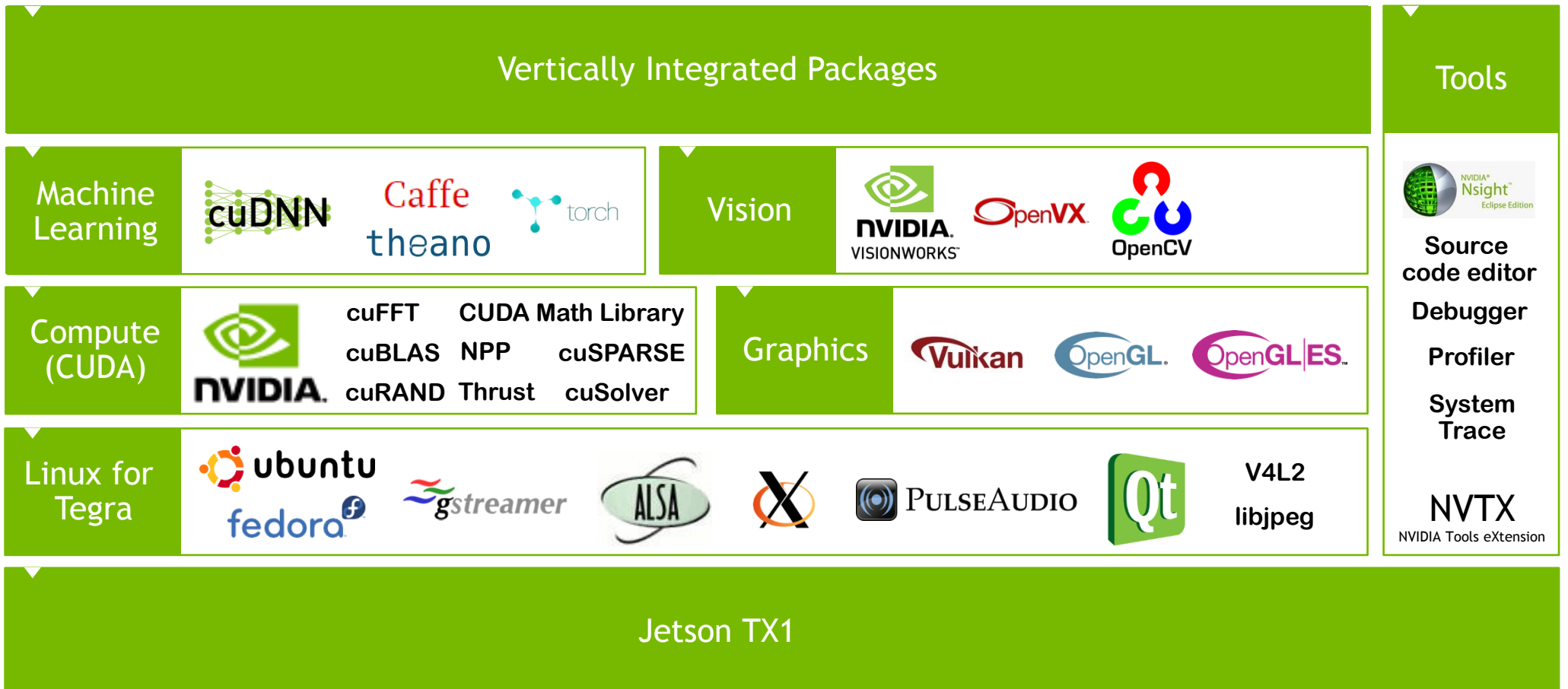
VisionWorks

Jetson Multimedia SDK

and other technologies:

CUDA, Linux4Tegra, NSIGHT EE, OpenCV4Tegra,
OpenGL, Vulkan, System Trace, Visual Profiler, Ubuntu 14.04

JETSON SDK: THE DETAILS



VISIONWORKS™

CUDA-accelerated Computer Vision Toolkit

- Full OpenVX 1.1 implementation
- Easy integration with existing CV pipelines
- Custom extensions

Robotics | Augmented Reality | Drones

Feature Tracking | Structure from Motion | Object Tracking | Dense Optical Flow

Example Applications

VisionWorks™ Toolkit

VisionWorks™ API + FrameWorks

IMAGE ARITHMETIC	GEOMETRIC TRANSFORMS	Features
Absolute Difference	Affine Warp +	Canny Edge Detector
Accumulate Image	Perspective Warp	Fast Corners +
Accumulate Squared	Flip Image	Fast Track
Accumulate Weighted	Gaussian Pyramid	Harris Corners +
Add / Subtract / Multiply	Remap	Harris Track
Channel Combine	Scale Image	Hough Circles
Channel Extract		Hough Lines

Applications

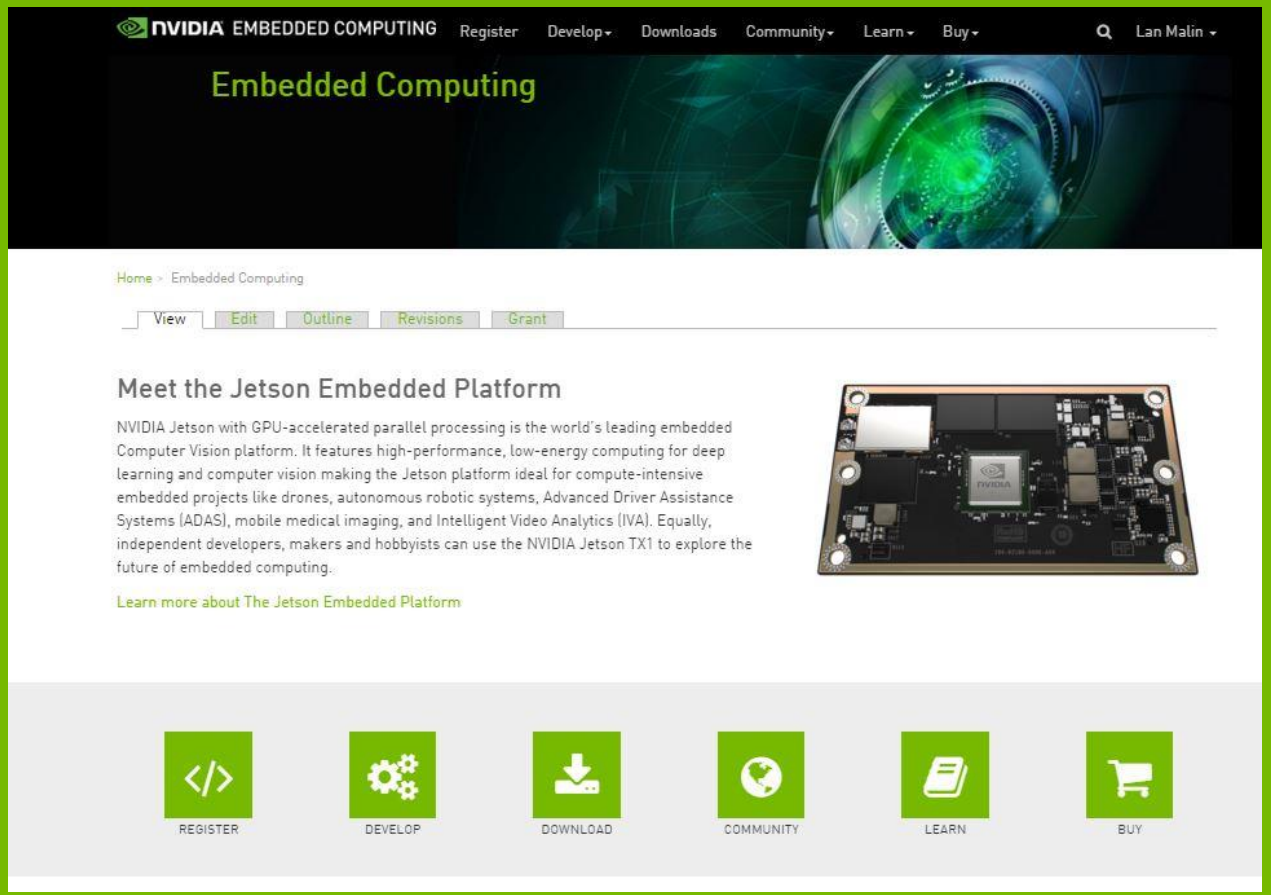
VisionWorks

CUDA

Jetson TX1

Comprehensive Developer Platform

- Jetpack SDK
- Libraries
- Developer tools
- Design collateral
- Developer Forum
- Training and Tutorials
- Ecosystem

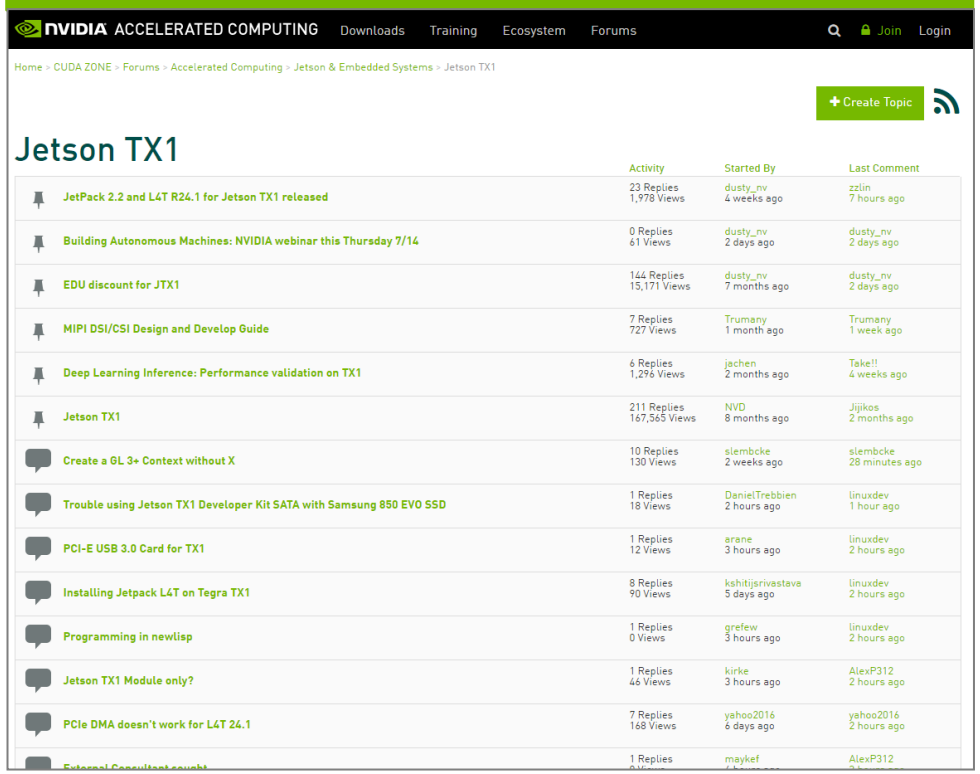


http://developer.nvidia.com/embedded-computing

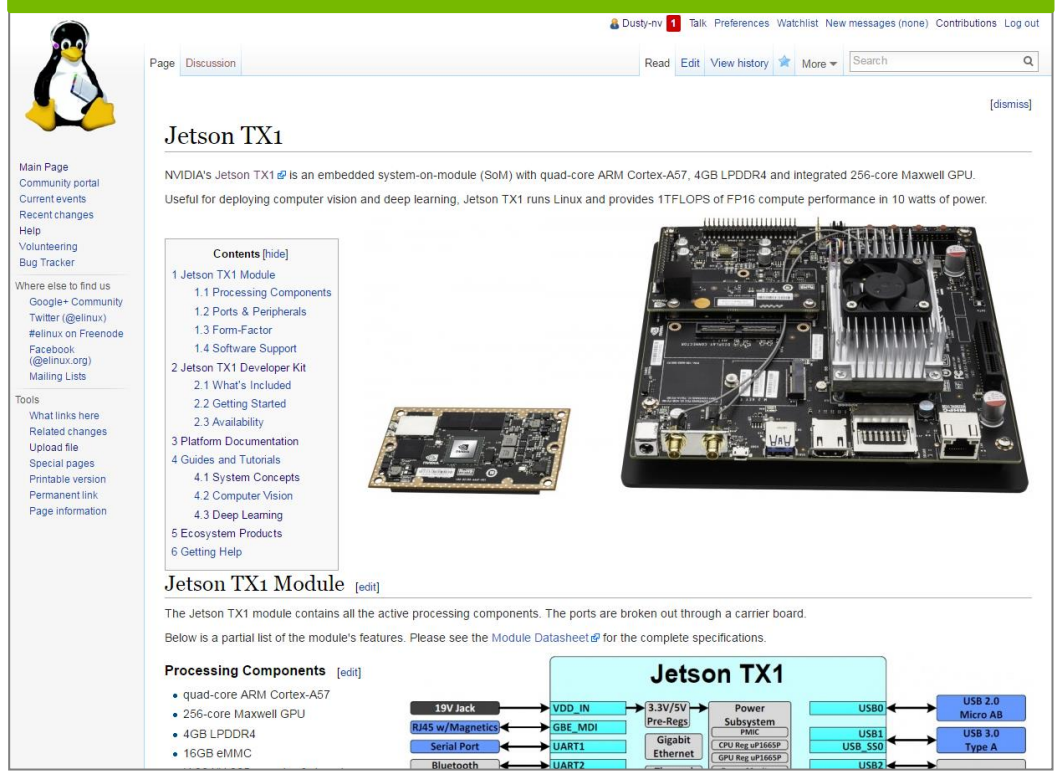
GETTING STARTED

JETSON COMMUNITY

Developer Forums devtalk.nvidia.com



eLinux Wiki eLinux.org/Jetson_TX1



THE PERIPHERALS JETSON CONNECTS WITH

including Community Contributions

- Infrared devices:
  - SICK LIDAR (LMS 200); Hokuyo; rpLIDAR
  - Asus Xtion Pro Live (PrimeSense)
  - Intel RealSense (mult. generations)
- Stereo and color cameras:
  - StereoLabs Zed (consumer-oriented)
  - Point Grey Research USB3 and GigE
  - e-con Systems CSI-MIPI Cameras
    with external ISP

DEPLOYING JETSON TX1

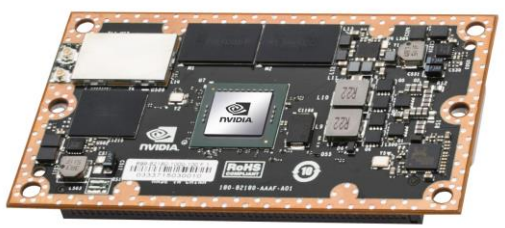
Modular Ecosystem

Connect Tech Inc.
Embedded Computing Experts

- Auvidea J120

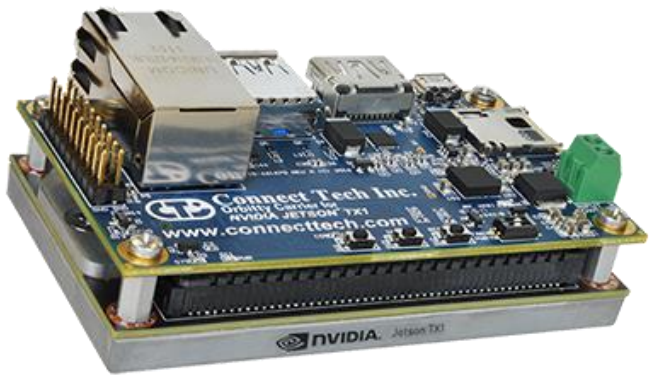
AUVIDEA
audio video media

NVIDIA

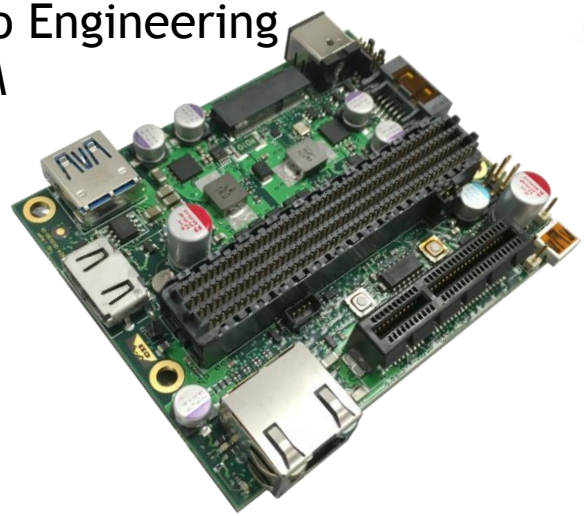


TX1 MODULE

- ConnectTech Orbitty



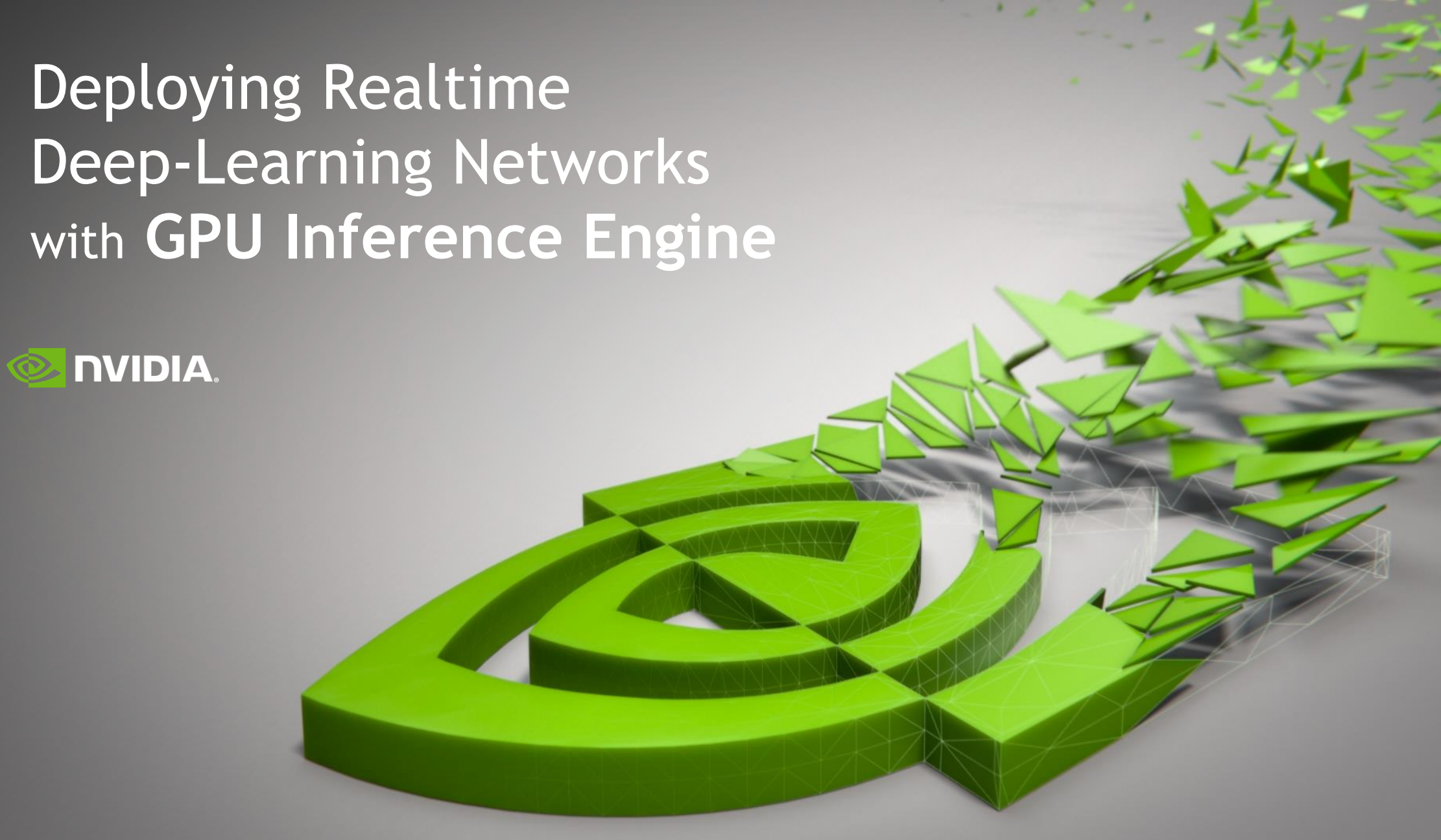
- Colorado Engineering TX1-SOM



COLORADO ENGINEERING, Inc.

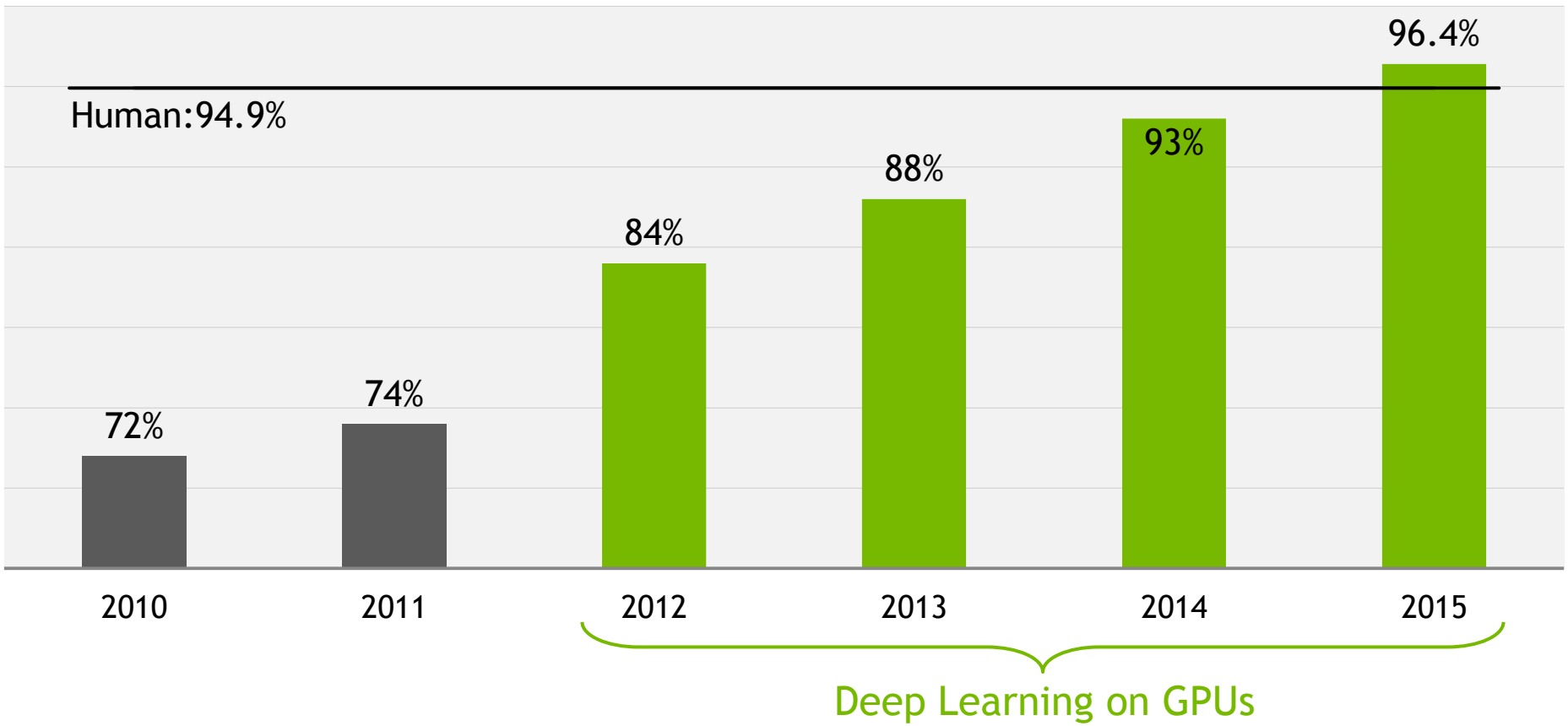
- ConnectTech Rosie

Deploying Realtime Deep-Learning Networks with **GPU Inference Engine**



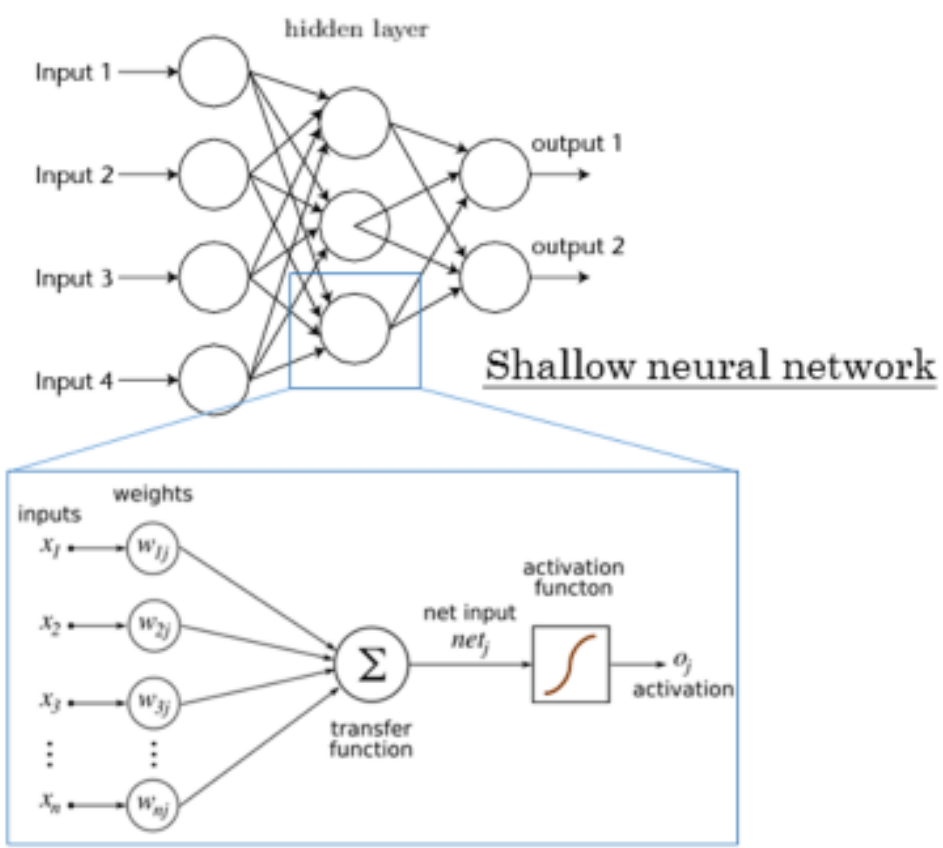
HOW FAR HAS AUTONOMY COME?

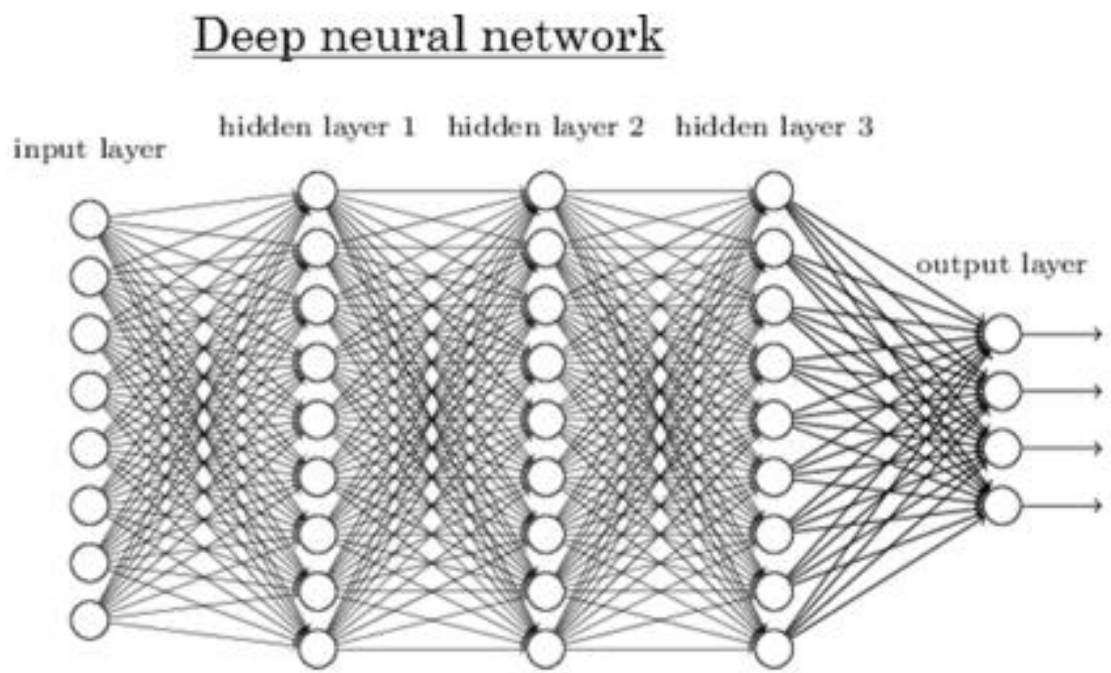
ImageNet classification accuracy



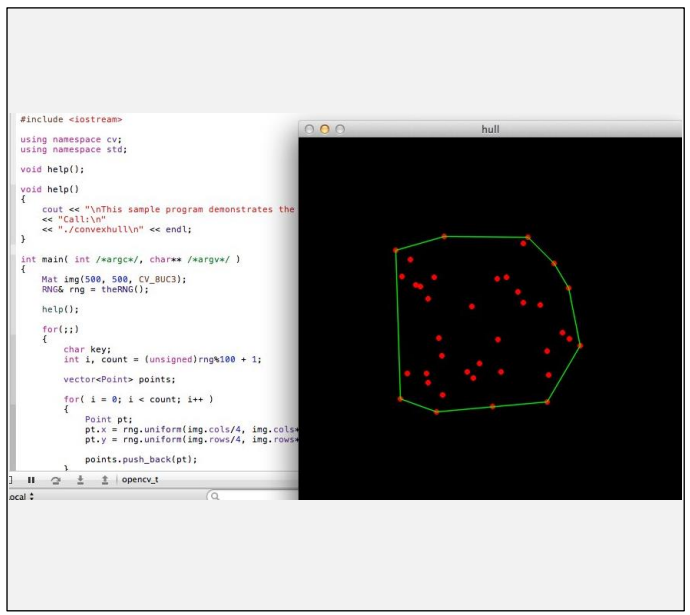
DEEP LEARNING

What's Different?





A NEW COMPUTING MODEL



Traditional Computer Vision
Experts + Time

Deep Learning
DNN + Data + HPC

Autonomous Machines
Onboard Intelligence

Object Classification

Segmentation

Collision Avoidance

3D Reconstruction

Localization/Mapping


Light vehicles detected: 2
Delivery vehicles detected: 3
Construction vehicles detected: 5
Avg inference time per frame: 400ms
81
95
98
99
99
99
99
99
94
GPU: ON
Deep Learning: ON
NVIDIA

POWERING THE DEEP LEARNING ECOSYSTEM

NVIDIA SDK Accelerates Every Major Framework

Object Detection

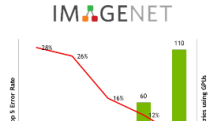


Image Classification

COMPUTER VISION

Voice Recognition

Language Translation

SPEECH AND AUDIO

Recommendation Engines

Sentiment Analysis

NATURAL LANGUAGE PROCESSING

Caffe

Chainer

DL4J

Deeplearning4j

Mocha.jl

julia

KERAS

Microsoft CNTK

MatConvNet

mxnet

MINERVA

Purine

OpenDeep

Pylearn2

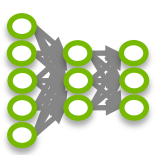
TensorFlow

torch

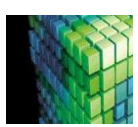
theano

DEEP LEARNING FRAMEWORKS

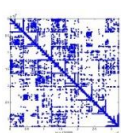
cuDNN



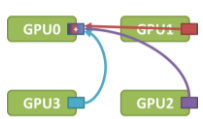
cuBLAS



cuSPARSE



NCCL

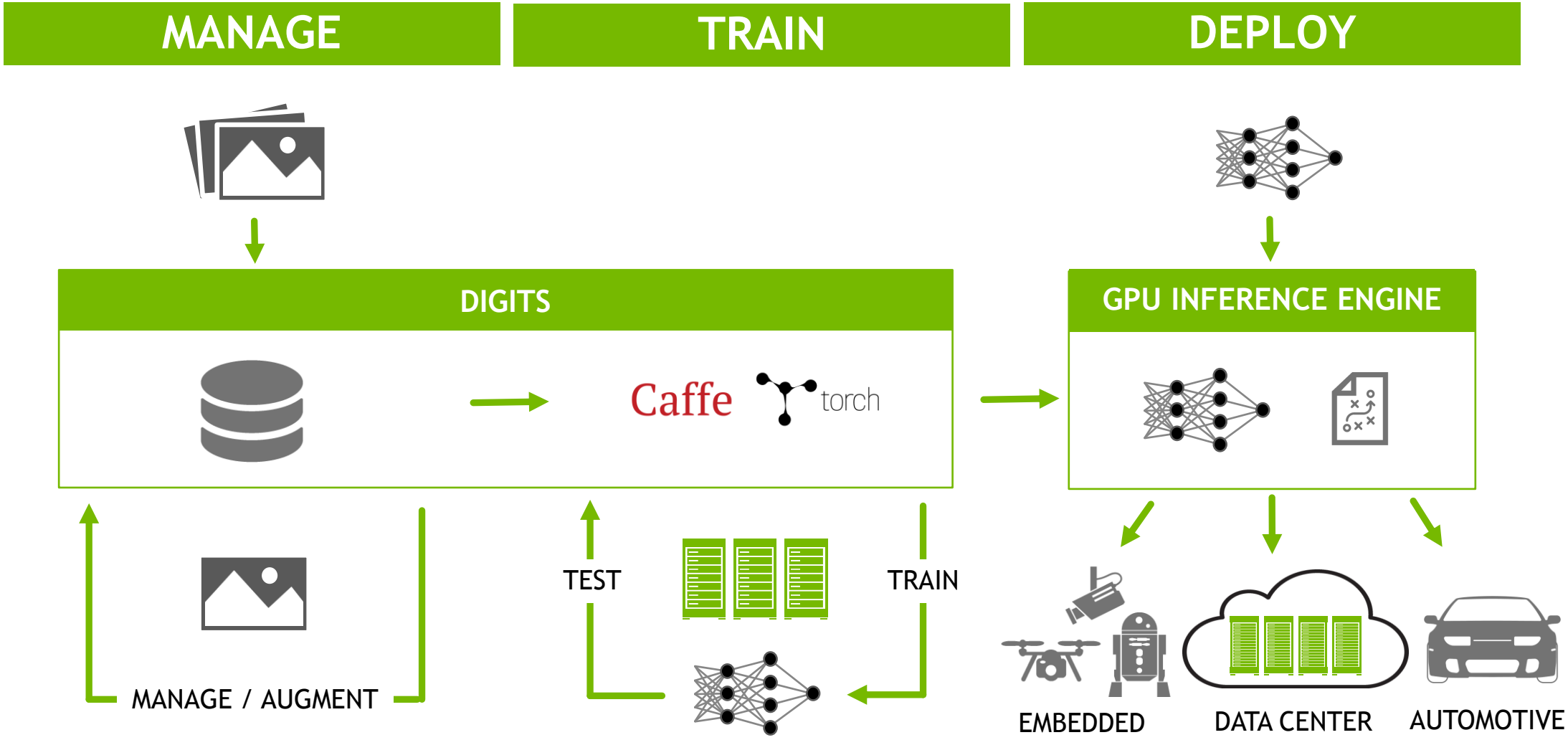


GIE

NVIDIA DEEP LEARNING SDK

developer.nvidia.com/deep-learning-software

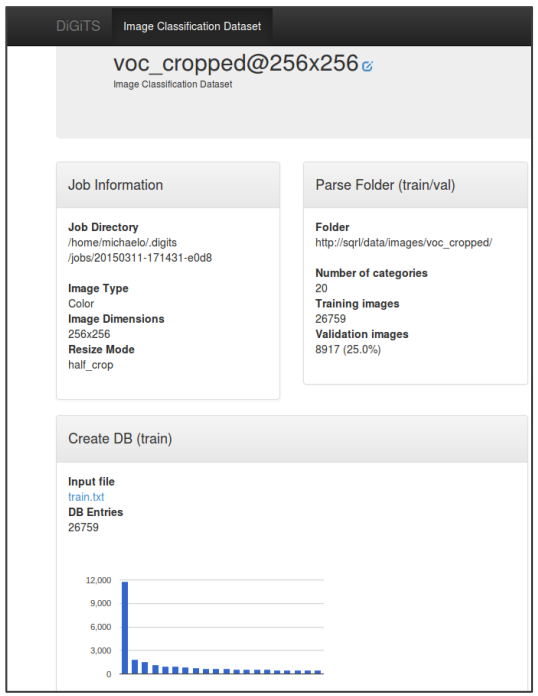
A COMPLETE COMPUTE PLATFORM



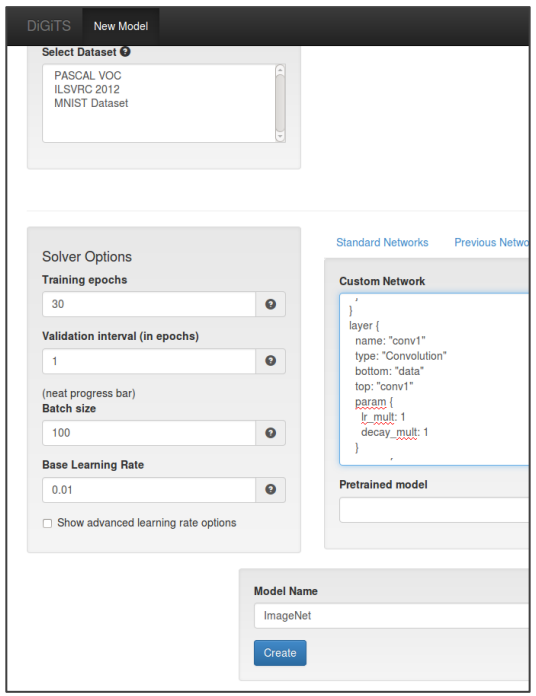
NVIDIA DIGITS

Interactive Deep Learning GPU Training System

Process Data

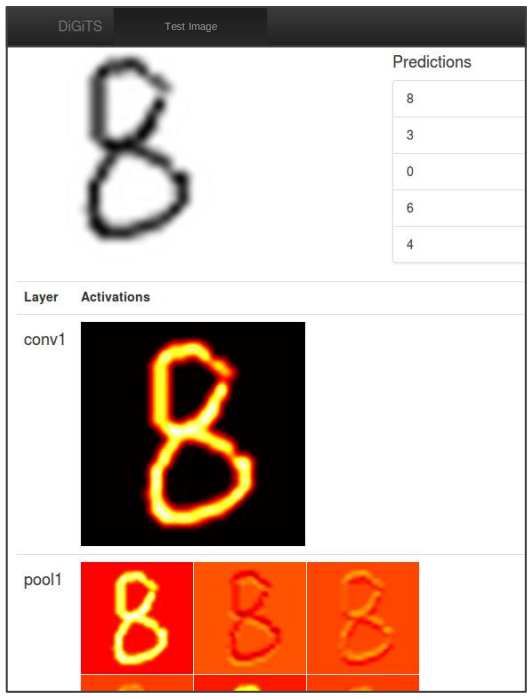


Configure DNN



Monitor Progress

Visualize Layers

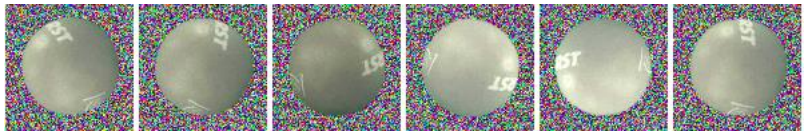


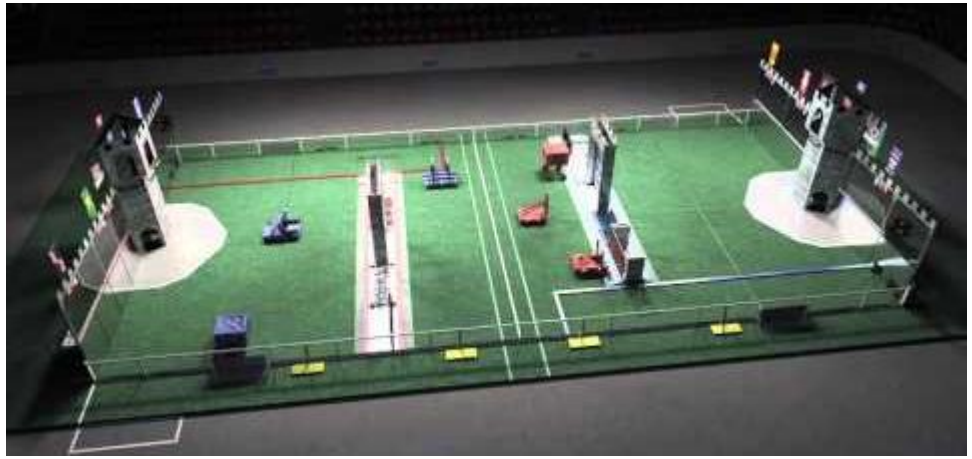
developer.nvidia.com/digits

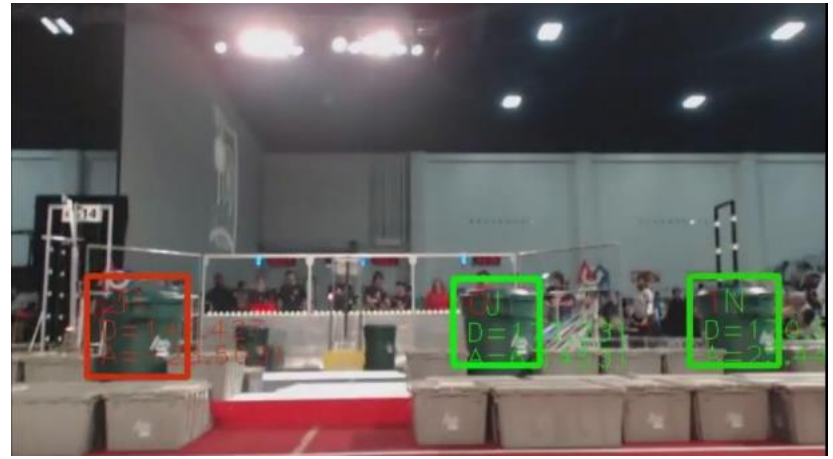
GitHub

FIRST Team 900

ROBUST DATA COLLECTION





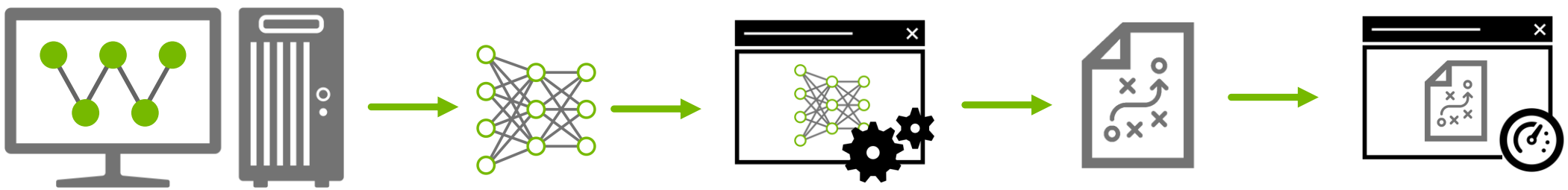


ZEBRACORNS

team900.org

GPU INFERENCE ENGINE

Workflow



developer.nvidia.com/gpu-inference-engine

Caffe features

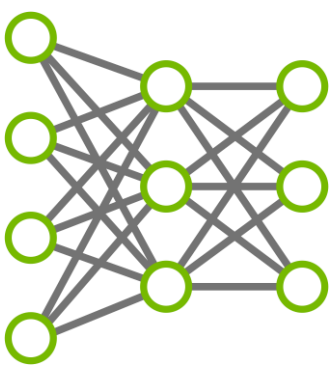
Deep Learning model definition

- Protobuf model format
  - Strongly typed format
  - Human readable
  - Auto-generates and checks Caffe code
  - Developed by Google
  - Used to define network architecture and training parameters
  - No coding required!

```
name: "conv1"
type: "Convolution"
bottom: "data"
top: "conv1"
convolution_param {
        num_output: 20
        kernel_size: 5
        stride: 1
        weight_filler {
                type: "xavier"
        }
}
```

GPU INFERENCE ENGINE

Optimizations



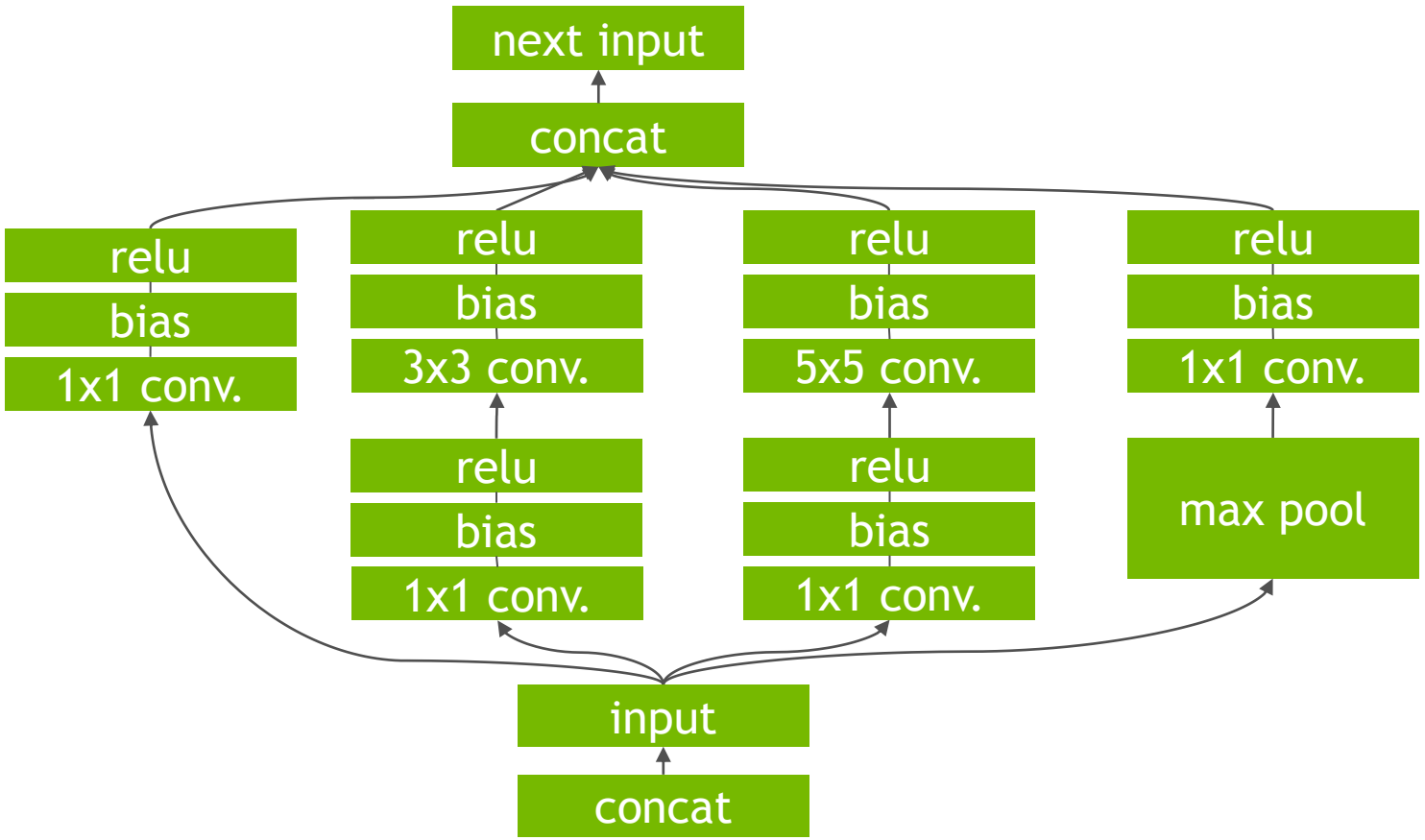
TRAINED
NEURAL NETWORK

- Fuse network layers
- Eliminate concatenation layers
- Kernel specialization
- Auto-tuning for target platform
- Select optimal tensor layout
- Batch size tuning

OPTIMIZED
INFERENCE
RUNTIME

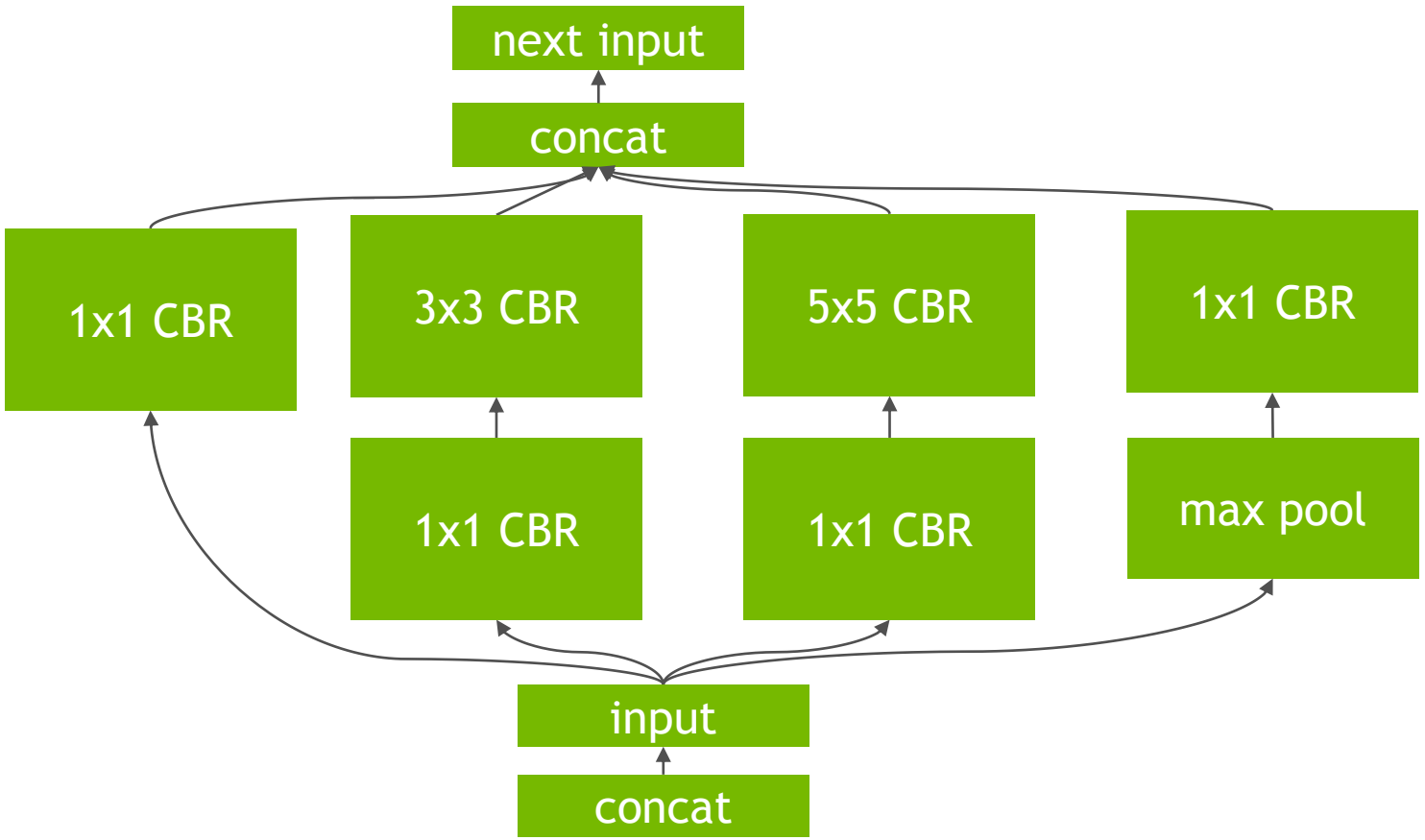
developer.nvidia.com/gpu-inference-engine

Graph Optimization



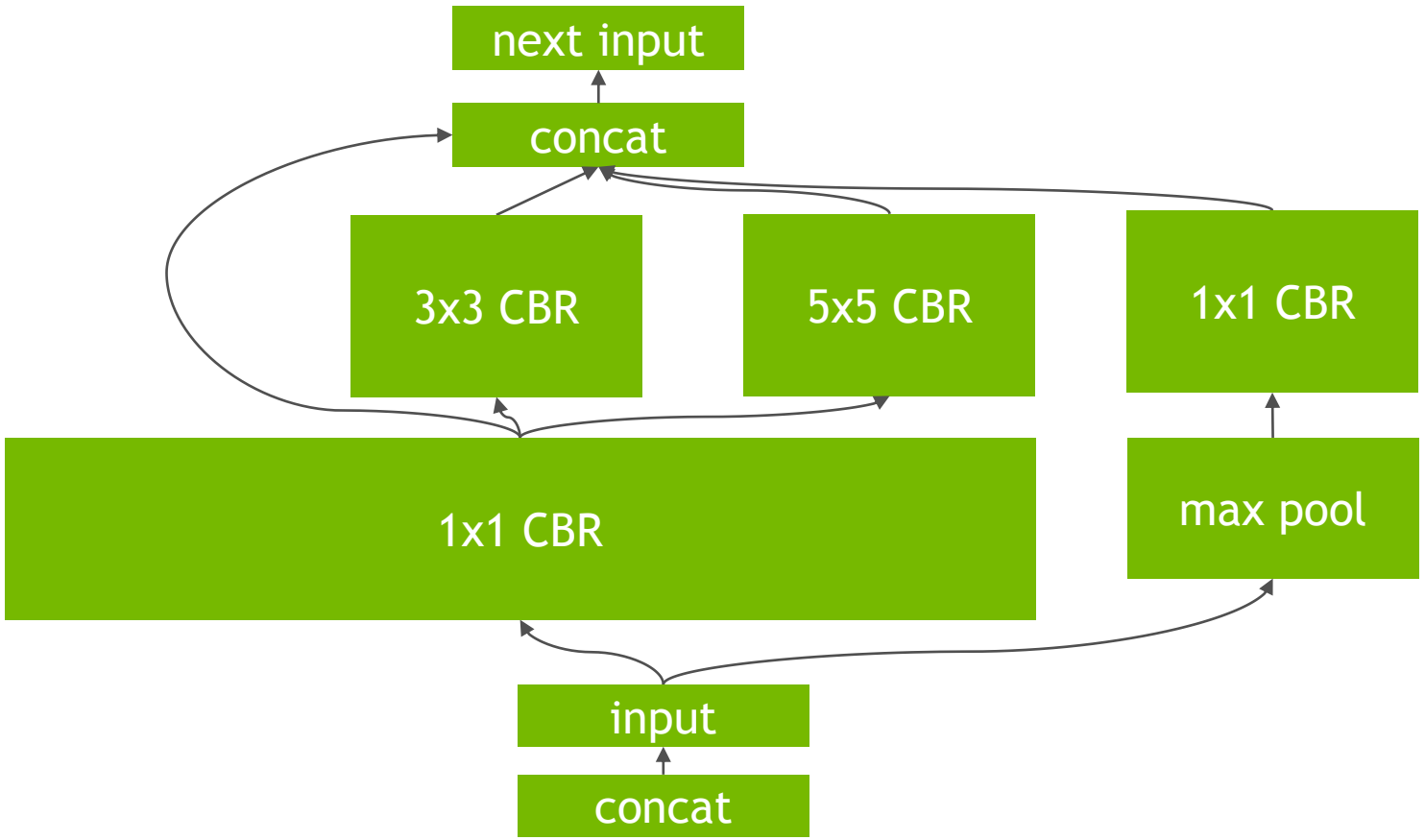
Graph Optimization

Vertical fusion



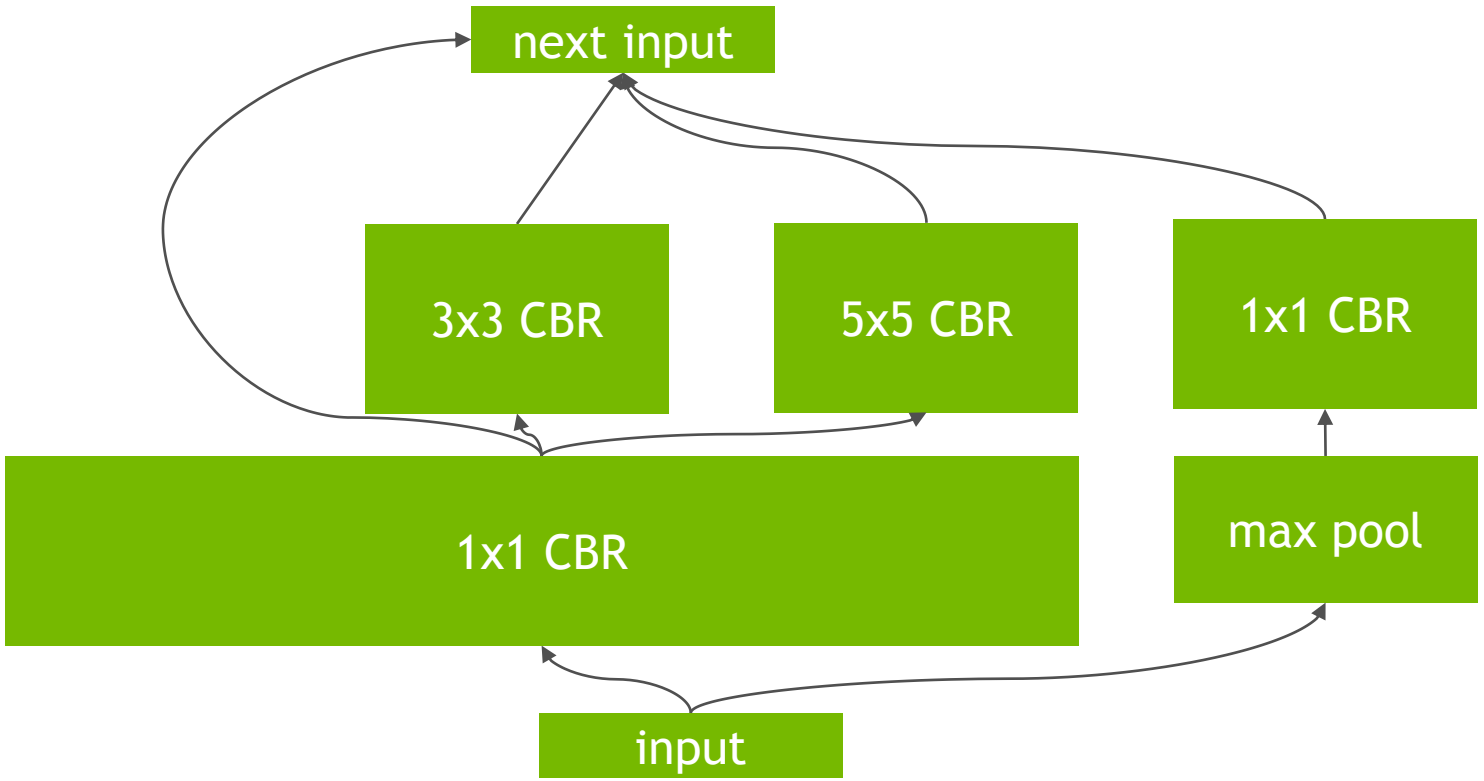
Graph Optimization

Horizontal fusion



Graph Optimization

Concat elision



Autotuning

Choose the fastest kernel for each layer

- Baseline is cuDNN / cuBLAS
- Direct convolution kernels for small batch
- Custom Winograd & Implicit GEMM for Half2
- Custom Deconvolution for filter size == stride case
- Weight pre-transform for Winograd
- Optimal T/N choice for BLAS
- Run cudnnFindForwardConvolutionEx() with multiple iterations

Build

Importing a Caffe Model

```
// create the network definition
INetworkDefinition* network = infer->createNetwork();

// create a map from caffe blob names to GIE tensors
std::unordered_map<std::string, infer1::Tensor> blobNameToTensor;

// populate the network definition and map
CaffeParser* parser = new CaffeParser;
parser->parse(deployFile, modelFile, *network, blobNameToTensor);

// tell GIE which tensors are required outputs
for (auto& s : outputs)
    network->setOutput(blobNameToTensor[s]);
```

Build

Engine Creation

```
// Specify the maximum batch size and scratch size
CudaEngineBuildContext buildContext;
buildContext.maxBatchSize = maxBatchSize;
buildContext.maxWorkspaceSize = 1 << 20;

// create the engine
ICudaEngine* engine =
   infer->createCudaEngine(buildContext, *network);

// serialize to a C++ stream
engine->serialize(gieModelStream);
```

Runtime

Binding Buffers

```
// get array bindings for input and output
int inputIndex = engine->getBindingIndex(INPUT_BLOB_NAME),
    outputIndex = engine->getBindingIndex(OUTPUT_BLOB_NAME);

// set array of input and output buffers
void* buffers[2];
buffers[inputIndex] = gpuInputBuffer;
buffers[outputIndex] = gpuOutputBuffer;
```

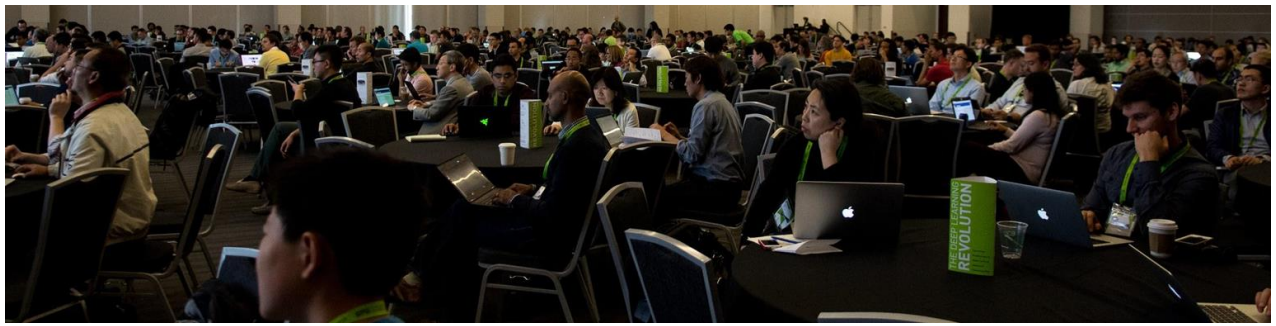
Runtime

Running the Engine

```
// Specify the batch size
CudaEngineContext context;
context.batchSize = batchSize;

// add GIE kernels to the given stream
engine->enqueue(context, buffers, stream, NULL);

<…>

// wait on the stream
cudaStreamSynchronize(stream);
```

NVIDIA Deep Learning Institute

Hands-on Training for Data Scientists and Software Engineers



Training organizations and individuals to solve challenging problems using Deep Learning

On-site workshops and online courses presented by certified experts

Covering complete workflows for proven application use cases

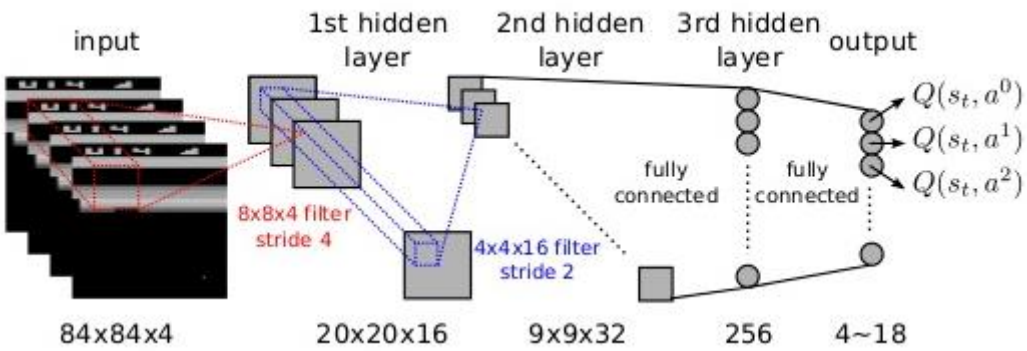
Image classification, object detection, natural language processing, recommendation systems, and more

www.nvidia.com/dli

Deep Reinforcement Learning

PLAYING ATARI WITH DEEPMIND

From Pixels to Actions: Human-level control through Deep Reinforcement Learning



AlphaGo wins match against Go champ Lee Se-dol 4-1

Historic Go match shows technological advances in AI research

AlphaGo beats world champion Lee Sedol in the second match of the Go tournament

Go-playing artificial intelligence program AlphaGo won a historic match against Korean Go grandmaster Lee Se-dol with a final score of 4-1.

Although Lee put up a good fight, he resigned after 280 moves in the final showdown, which lasted around five hours -- the longest in the five-game match.

"It is regrettable that I could not wrap up the match by winning the final round." said Lee, attending a press conference after the final game at Four Seasons Hotel in Gwanghwamun on Tuesday.

"I thought I got an upper hand in the earlier stage of the game today, but I eventually lost because I was not good enough," Lee said.

The Go-playing software boasted its almost impeccable capability to read and predict, and was able to maintain a slight lead throughout the final round.

Asynchronous Methods for Deep Reinforcement Learning

Volodymyr Mnih[1] VMNIH@GOOGLE.COM
Adrià Puigdomènech Badia[1] ADRIAP@GOOGLE.COM
Mehdi Mirza[1,2] MIRZAMOM@IRO.UMONTREAL.CA
Alex Graves[1] GRAVESA@GOOGLE.COM
Tim Harley[1] THARLEY@GOOGLE.COM
Timothy P. Lillicrap[1] COUNTZERO@GOOGLE.COM
David Silver[1] DAVIDSILVER@GOOGLE.COM
Koray Kavukcuoglu[1] KORAYK@GOOGLE.COM
[1] **Google DeepMind**
[2] **Montreal Institute for Learning Algorithms (MILA), University of Montreal**

Abstract

We propose a conceptually simple and lightweight framework for deep reinforcement learning that uses asynchronous gradient descent for optimization of deep neural network controllers. We present asynchronous variants of four standard reinforcement learning algorithms and show that parallel actor-learners have a stabilizing effect on training allowing all four methods to successfully train neural network controllers. The best performing method, an asynchronous variant of actor-critic, surpasses the current state-of-the-art on the Atari domain while training for half the time on a single multi-core CPU instead of a GPU. Furthermore, we show that asynchronous actor-critic succeeds on a wide variety of continuous motor control problems as well as on a new task involving finding rewards in random 3D mazes using a visual input.

http://arxiv.org/abs/1602.01783

AlphaGo wins match against Go champ Lee Se-dol 4-1

Comments 240 Shares

Inside Google's DeepMind AlphaGo GPU cluster

January 28, 2016

For the first time, a computer has beaten a human professional at the game of Go – an ancient board game that has long been viewed as one of the greatest challenges for Artificial Intelligence.

Google DeepMind's GPU-accelerated AlphaGo program beat Fan Hui, the European Go champion, five times out of five in tournament conditions.

Demis Hassabis, who oversees DeepMind, mentioned in a recent article that DeepMind's deep learning system works pretty well on a single computer equipped with a decent number of GPU accelerators, but for the match against Fan Hui, the researchers used a larger network of computers that spanned about 170 GPUs. This larger computer network both trained the system and played the actual game, drawing on the results of the training.

ronous Methods for Deep Reinforcement Learning

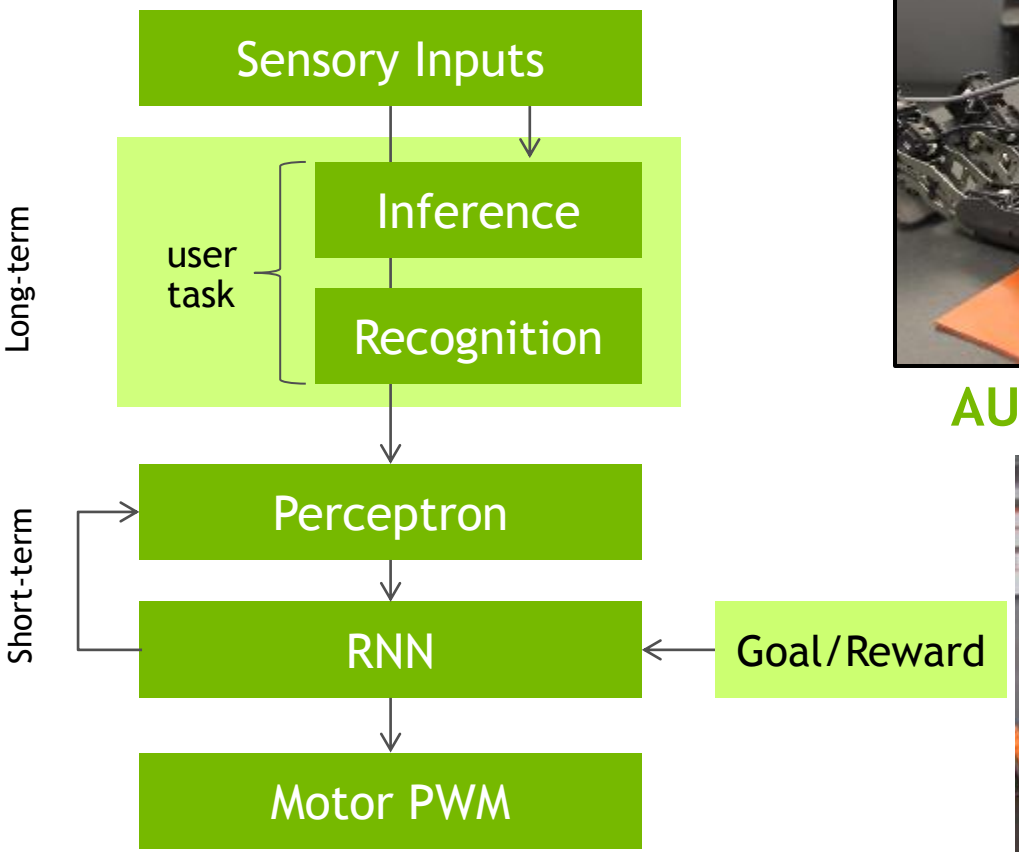
r Mnih[1] VMNIH@GOOGLE.COM
gdomènech Badia[1] ADRIAP@GOOGLE.COM
irza[1,2] MIRZAMOM@IRO.UMONTREAL.CA
ves[1] GRAVESA@GOOGLE.COM
ey[1] THARLEY@GOOGLE.COM
P. Lillicrap[1] COUNTZERO@GOOGLE.COM
ver[1] DAVIDSILVER@GOOGLE.COM
vukcuoglu[1] KORAYK@GOOGLE.COM
DeepMind
Institute for Learning Algorithms (MILA), University of Montreal

Abstract

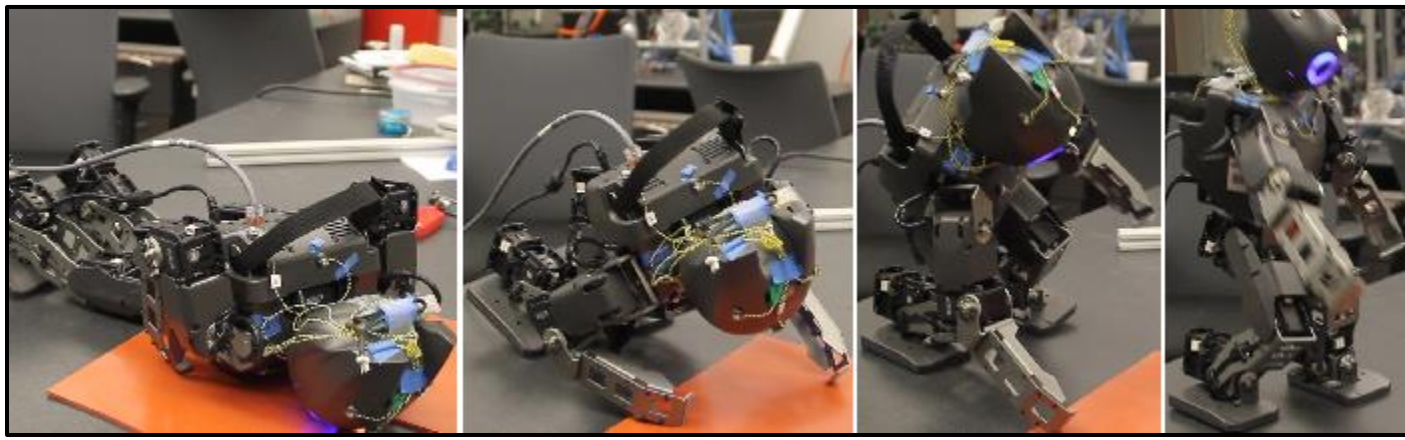
opose a conceptually simple and lightweight framework for deep reinforce-ning that uses asynchronous gradient descent for optimization of deep neural ontrollers. We present asynchronous variants of four standard reinforcement lgorithms and show that parallel actor-learners have a stabilizing effect on llowing all four methods to successfully train neural network controllers. The rming method, an asynchronous variant of actor-critic, surpasses the cur--of-the-art on the Atari domain while training for half the time on a single CPU instead of a GPU. Furthermore, we show that asynchronous actor-critic on a wide variety of continuous motor control problems as well as on a new ving finding rewards in random 3D mazes using a visual input.

http://arxiv.org/abs/1602.01783

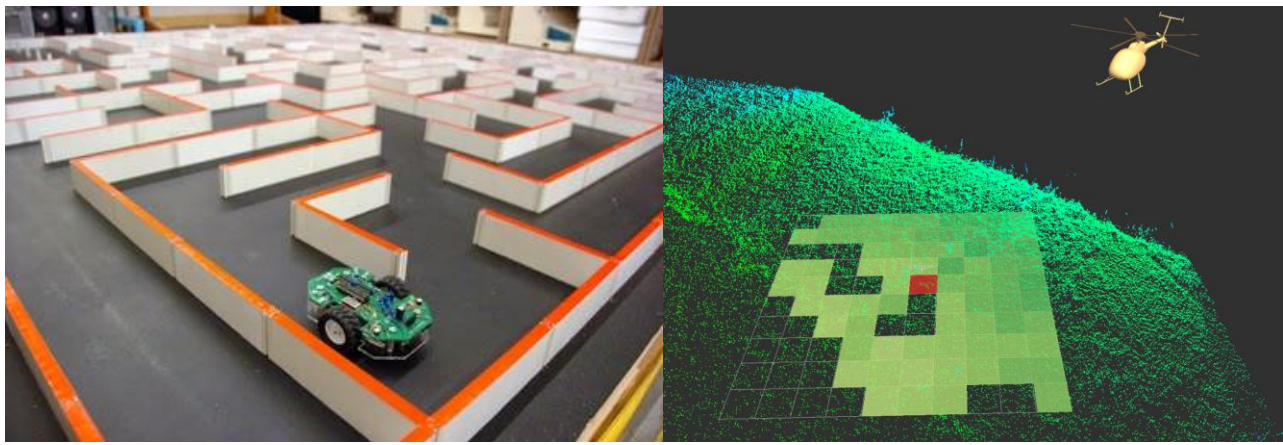
END-TO-END LEARNING



MOTION CONTROL



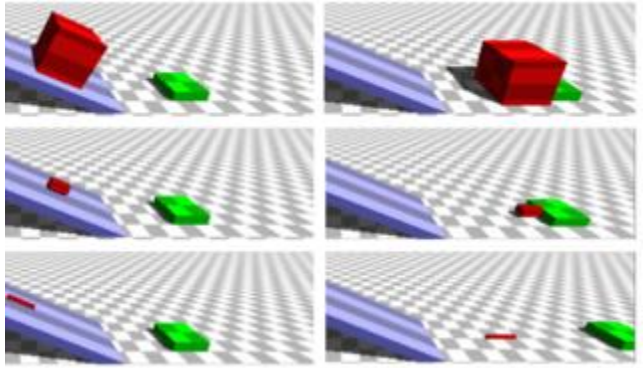
AUTONOMOUS NAVIGATION

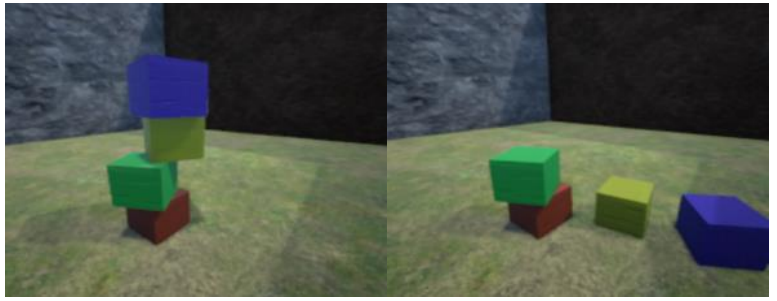


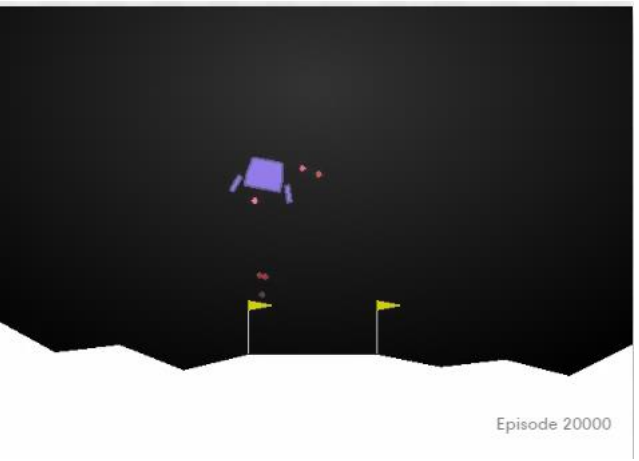
SIMULATION

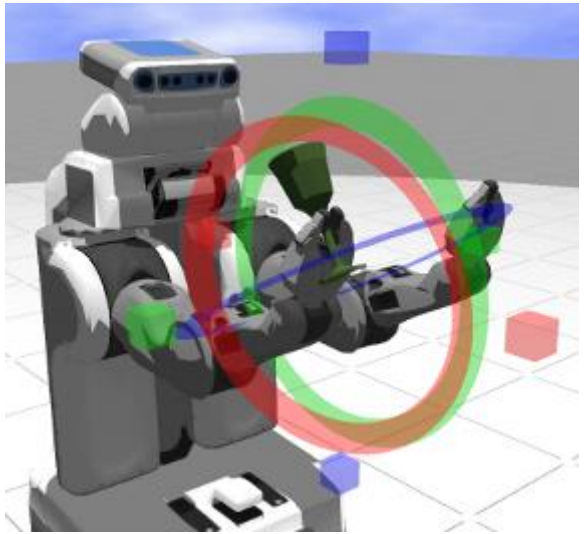
Physical Intuition

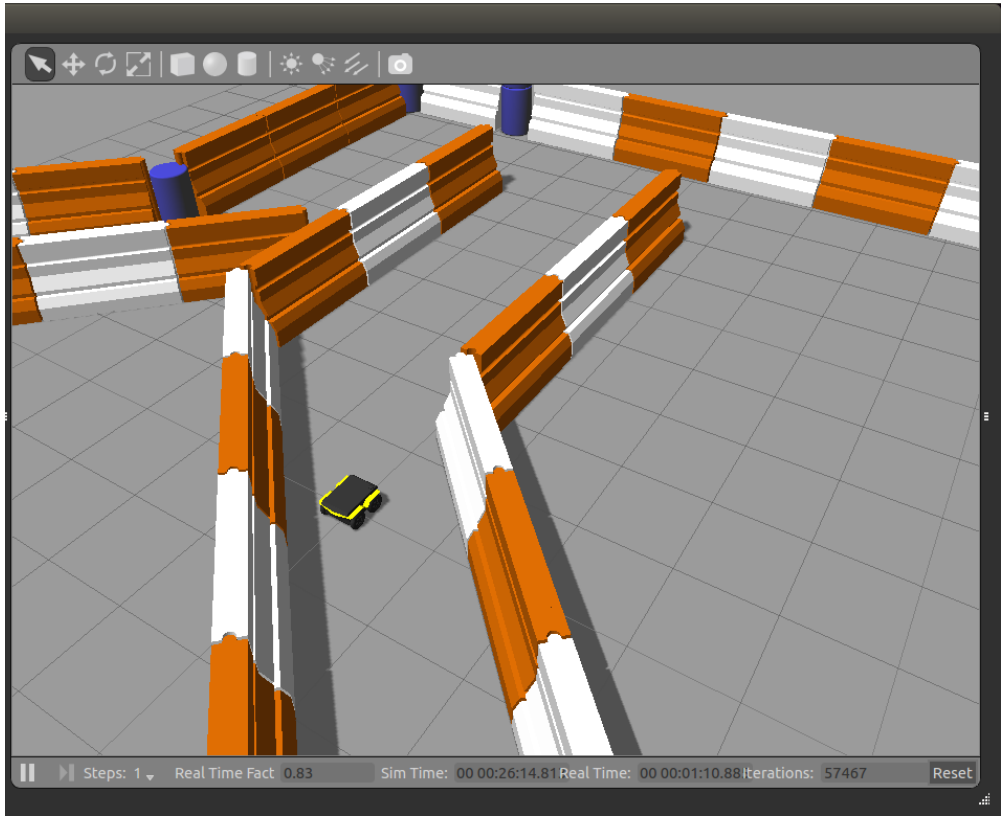
- OpenAI Gym
- Gazebo
- Unreal4Torch
- PhysX
- Others







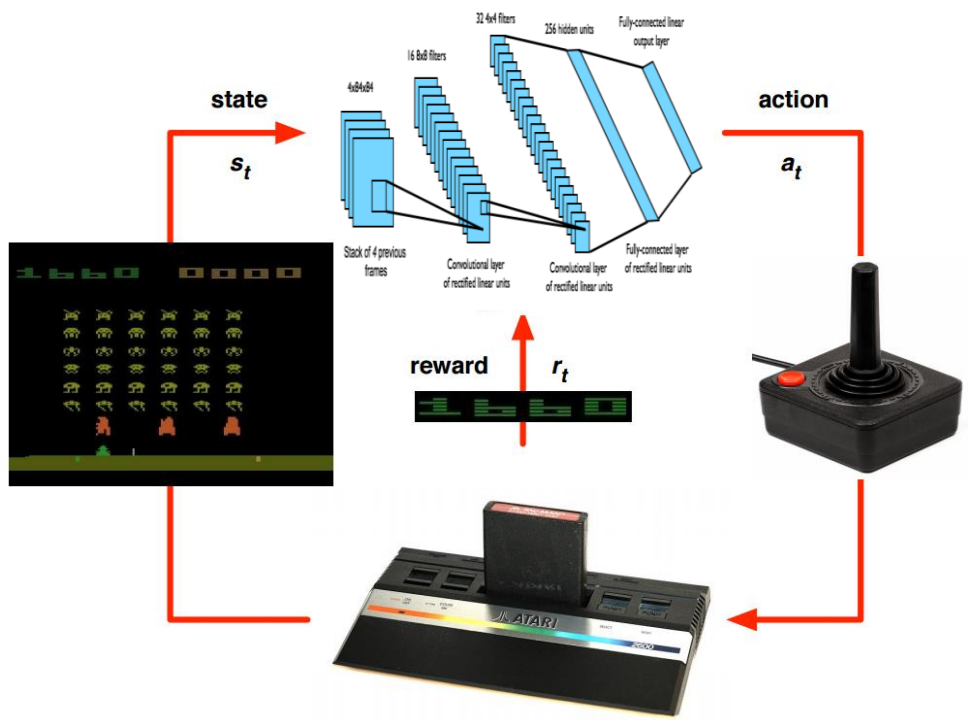




Q-LEARNING

How's it work?

- A reinforcement learning agent includes:
  - **state** (environment)
  - **actions** (controls)
  - **reward** (feedback)
- A **value function** predicts the future reward of performing actions in the current state
  - Given the recent state, action with the maximum estimated future reward is chosen for execution
- For agents with complex state spaces, deep networks are used as **Q-value** approximator
  - Numerical solver (**gradient descent**) optimizes the network on-the-fly based on reward inputs

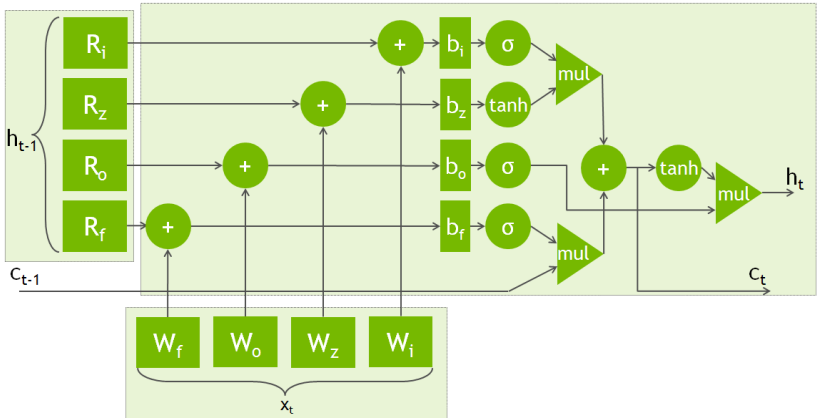


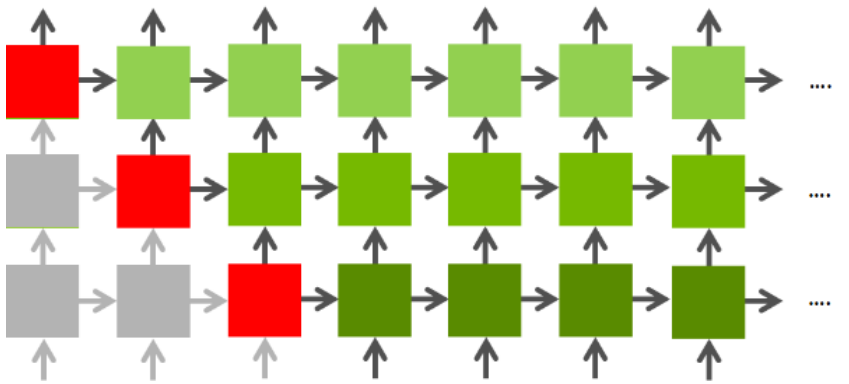
Example NN in Lua

```
 1: -- parameters
 2: nstates = {16,256,128}
 3: fanin = {1,4}
 4: filtsize = 5
 5: poolsize = 2
 6: normkernel = image.gaussian1D(7)
 7:
 8: -- Container:
 9: model = nn.Sequential()
10:
11: -- stage 1 : filter bank -> squashing -> L2 pooling -> normalization
12: model:add(nn.SpatialConvolutionMap(nn.tables.random(nfeats, nstates[1], fanin[1]), filtsize, filtsize))
13: model:add(nn.Tanh())
14: model:add(nn.SpatialLPPooling(nstates[1],2,poolsize,poolsize,poolsize,poolsize))
15: model:add(nn.SpatialSubtractiveNormalization(16, normkernel))
16:
17: -- stage 2 : filter bank -> squashing -> L2 pooling -> normalization
18: model:add(nn.SpatialConvolutionMap(nn.tables.random(nstates[1], nstates[2], fanin[2]), filtsize, filtsize))
19: model:add(nn.Tanh())
20: model:add(nn.SpatialLPPooling(nstates[2],2,poolsize,poolsize,poolsize,poolsize))
21: model:add(nn.SpatialSubtractiveNormalization(nstates[2], normkernel))
22:
23: -- stage 3 : standard 2-layer neural network
24: model:add(nn.Reshape(nstates[2]*filtsize*filtsize))
25: model:add(nn.Linear(nstates[2]*filtsize*filtsize, nstates[3]))
26: model:add(nn.Tanh())
27: model:add(nn.Linear(nstates[3], noutputs))
```

LSTM ACCELERATION

- Launch a 2D grid of RNN cells
- Multiple layers in a single call are faster
- Doesn't suffer from vanishing gradient
  - Able to adopt long-term strategy
- Supports:
  - Partially-observable environments
  - Uni/Bidirectional RNNs
  - Non-uniform length minibatches
  - Dropout between layers





DEEP-LEARNING RESEARCH ROVER

TURBO 2.0

github.org/dusty-nv

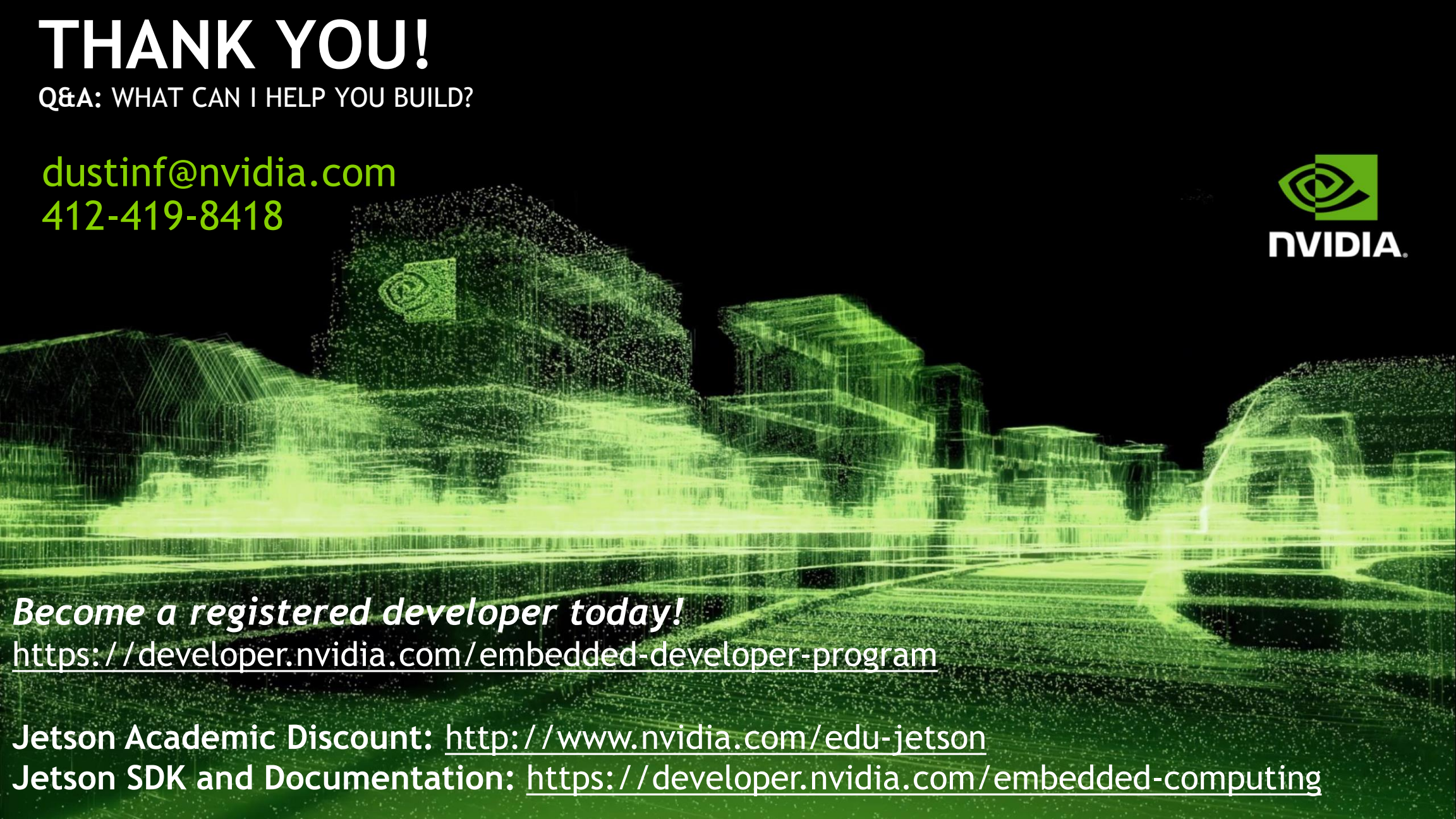
NVIDIA

GitHub

THANK YOU!

Q&A: WHAT CAN I HELP YOU BUILD?

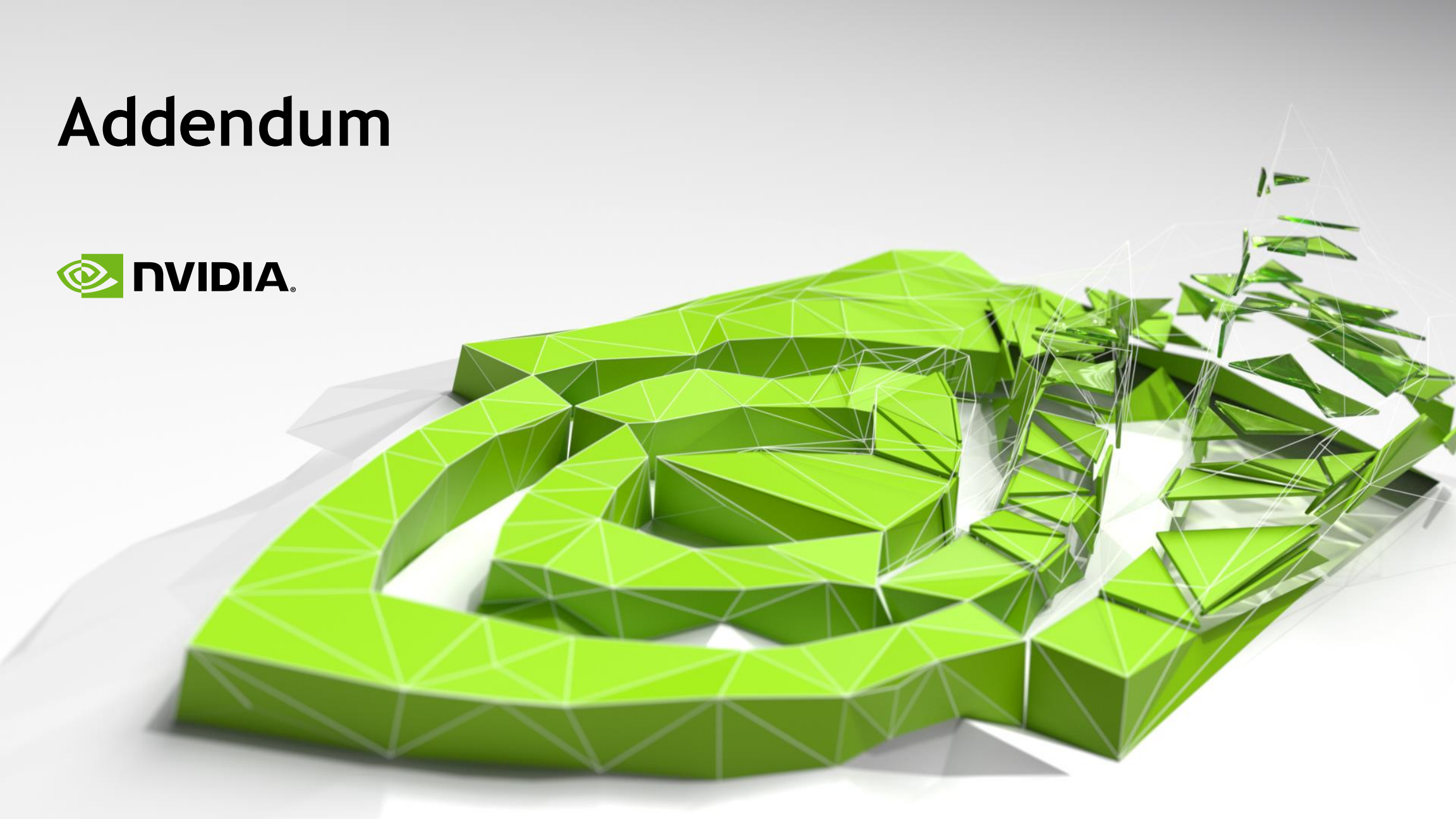
dustinf@nvidia.com
412-419-8418



Become a registered developer today!
https://developer.nvidia.com/embedded-developer-program

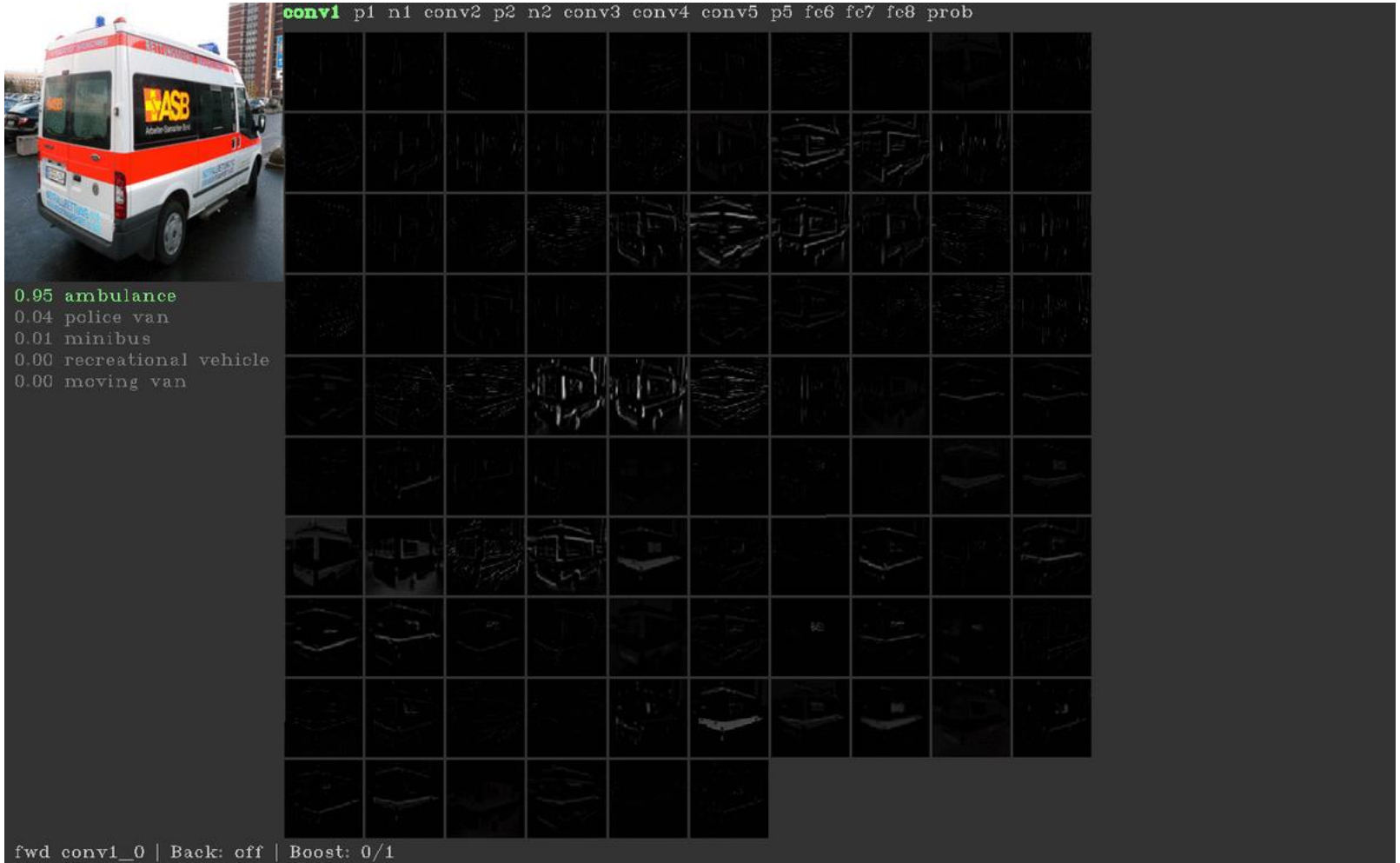
Jetson Academic Discount: http://www.nvidia.com/edu-jetson
Jetson SDK and Documentation: https://developer.nvidia.com/embedded-computing

Addendum



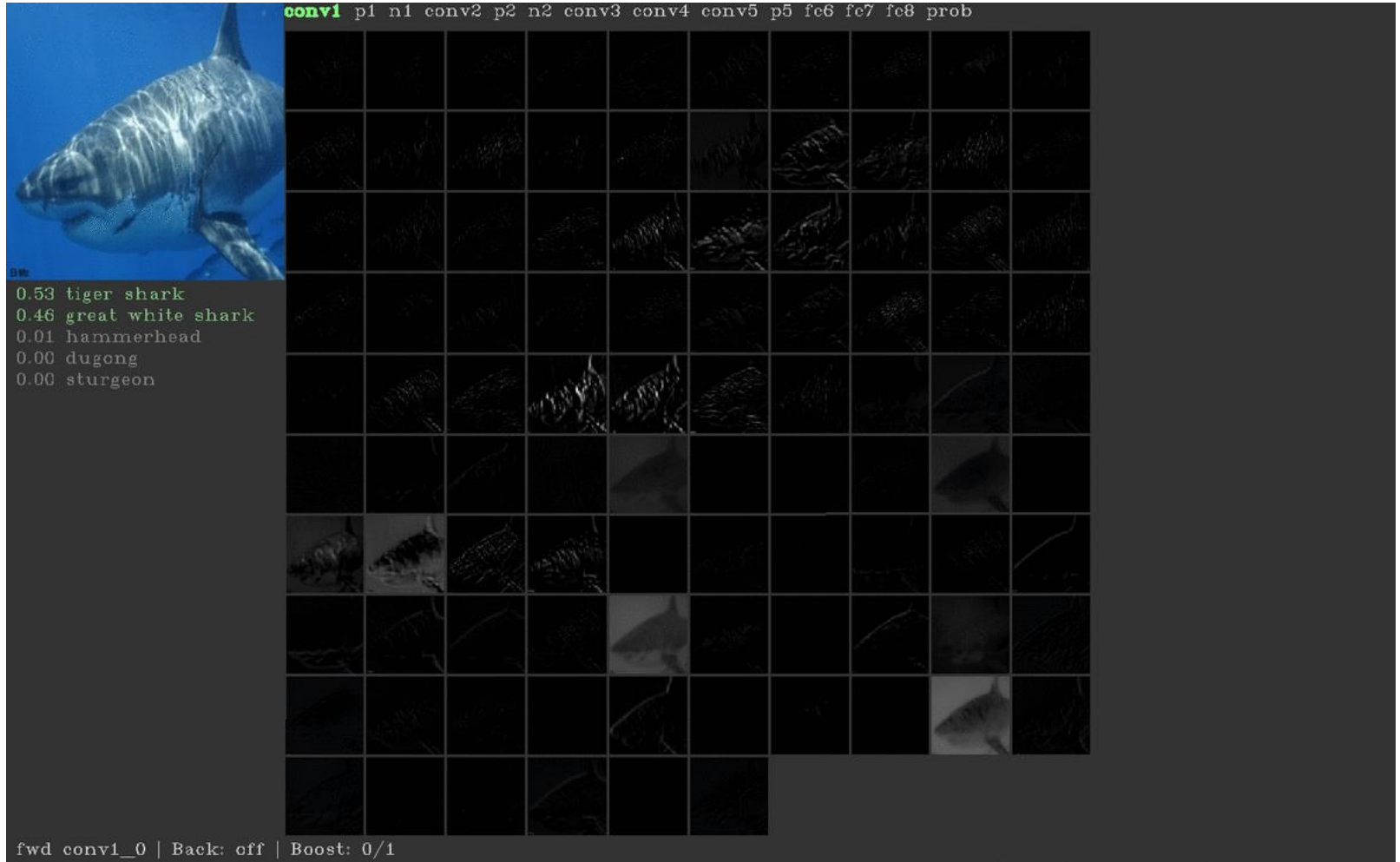
DEEP VISUALIZATION TOOLBOX

IMAGE RECOGNITION



DEEP VISUALIZATION TOOLBOX

IMAGE RECOGNITION



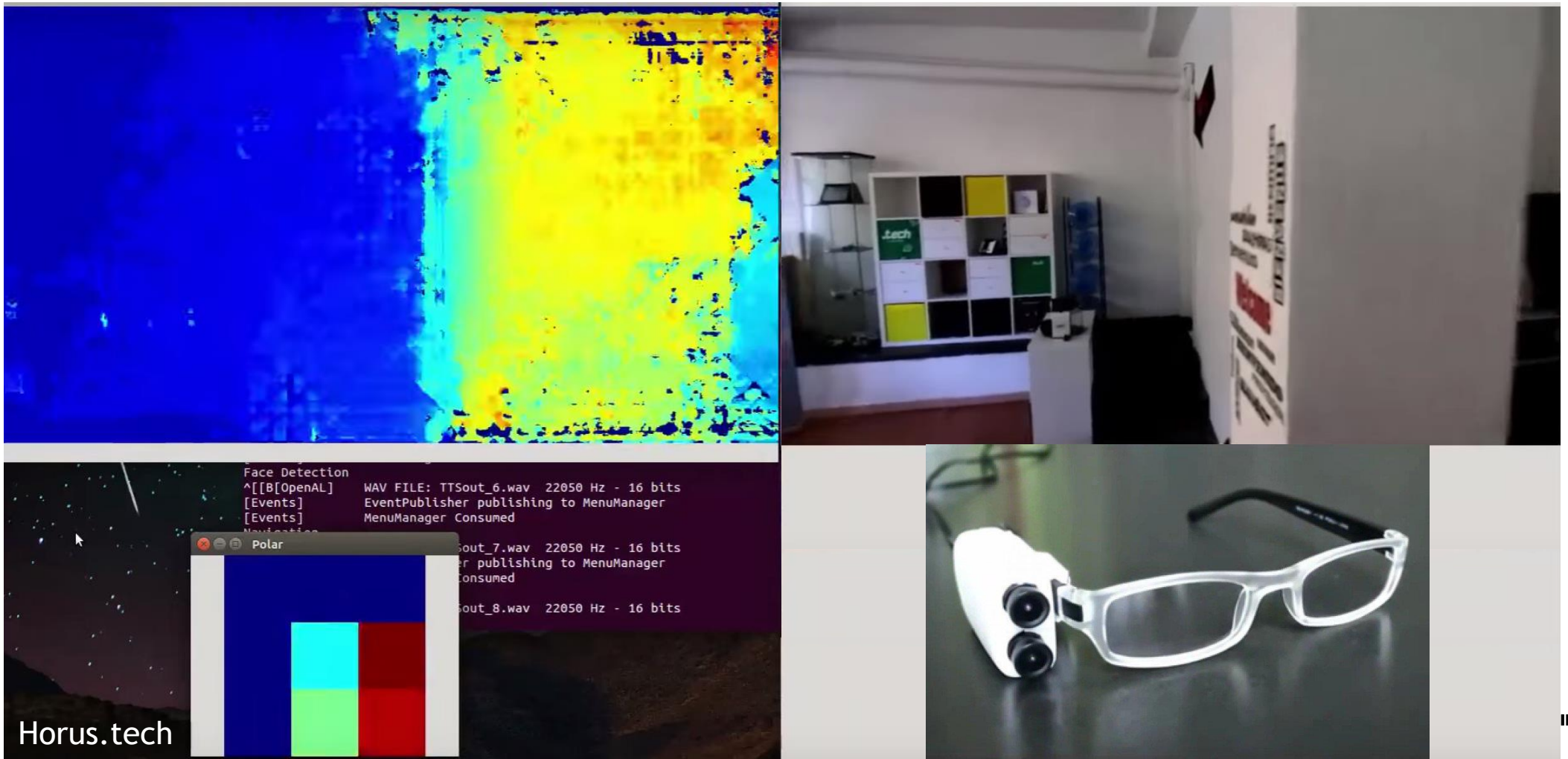
Horus Technology

HORUS

BLIND ASSIST DEVICE

EMERGING COMPANIES SUMMIT

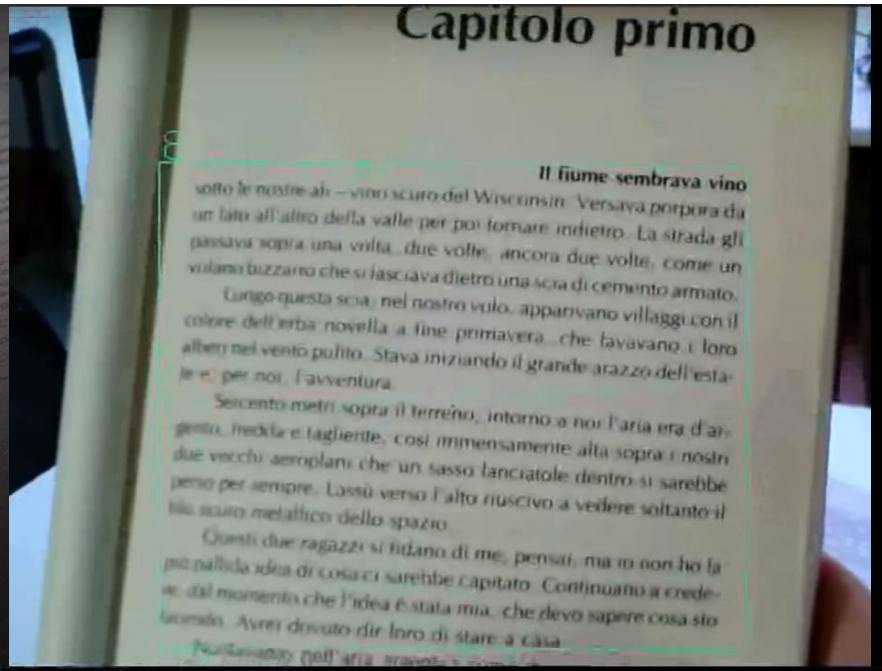
SOCIAL INNOVATION AWARD WINNER

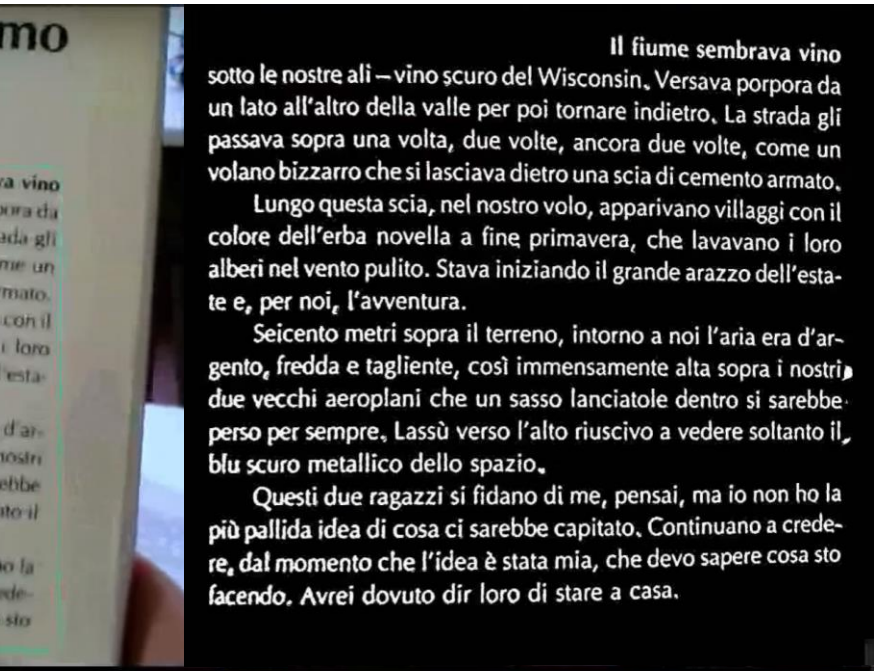


Horus Technology

HORUS
BLIND ASSIST DEVICE

EMERGING COMPANIES SUMMIT
SOCIAL INNOVATION AWARD WINNER





Horus.tech

IMAGE CAPTIONING WITH
NEURALTALK2


a man riding a bike on a beach with a dog in the water


a man sitting at a table with a laptop

a street sign on a pole in front of a building

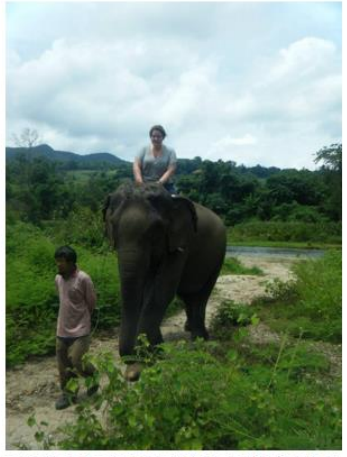

a plate with a sandwich and a salad


a black and white cat sitting in a bathroom sink


a woman is playing tennis on a tennis court


a little boy standing in a field with a kite


a man is sitting on a bench in front of a building


a man and a woman riding on the back of an elephant

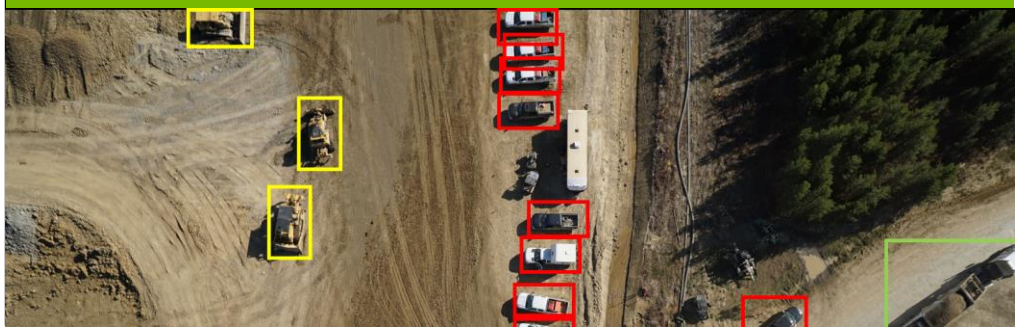

a laptop computer sitting on top of a wooden desk


a laptop computer sitting on top of a wooden desk


a bathroom with a sink and a toilet

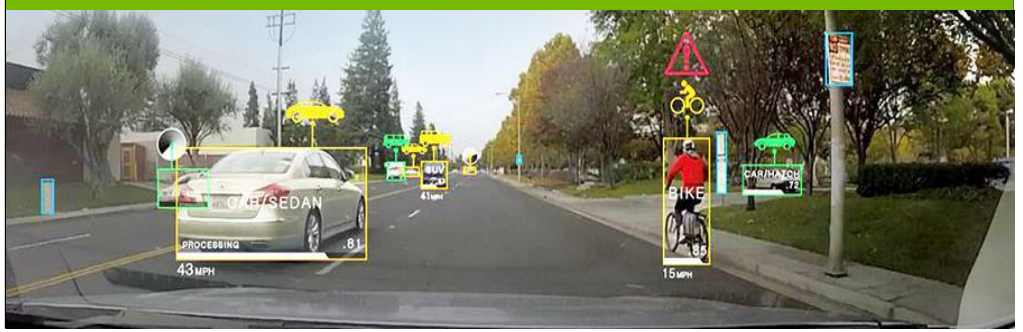
OBJECT DETECTION

REMOTE SENSING

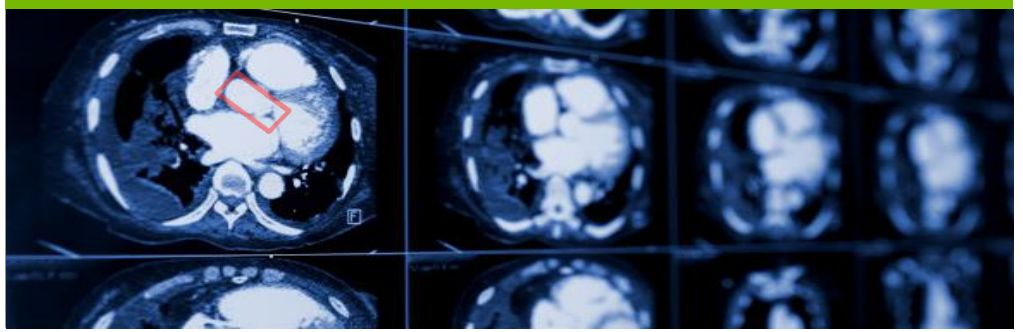


PICKING AND GRASPING

ADVANCED DRIVER ASSISTANCE SYSTEMS (ADAS)



MEDICAL DIAGNOSTICS



ROBERT BOND

DIY INTELLIGENCE

Installing the Particle Photon

The Photon was easy to setup. See the software section below for the code. It sits in the irrigation control box like this:

The black box on the left with the blue LED is a 24VAC to 5VDC converter from EBay. You can see the white relay on the relay board and the blue connector on the

The converter's 5V output is connected to the Photon's VIN pin. The relay board is basically analog: it has an open-collector NPN transistor with a nominal 3.3V input

Here's the way it is wired:

24VAC converter 24VAC	<------	Control box 24VAC OUT
24VAC converter +5V	<------	Photon VIN, resistor to relay board +3.3V
24VAC converter GND	<------	Photon GND, Relay GND
Photon D0	<------	Relay board signal input
Relay COM	<------	Control box 24VAC OUT
Relay NO	<------	Front yard water valve

Installing the Jetson

The only hardware bits added to the Jetson are a SATA SSD drive and a small Belkin USB hub. The hub has two wireless dongles that talk to the keyboard and mouse.

The SSD came up with no problem. I reformatted it as EXT4 and mounted it as /caffe. I strongly recommend moving all of your project code, git repos and application

The Wireless Access Point setup was pretty easy (really!) if you follow this guide. Just use the Ubuntu menus as directed and make sure you add this config setting

I installed vsftpd as the FTP server. The configuration is pretty much stock. I did not enable anonymous FTP. I gave the camera a login name and password that is not v

I installed Caffe using the JetsonHacks recipe. I believe current releases no longer have the LMDB_MAP_SIZE issue so try building it before you make the change. \
PYTHONPATH.

myplace.frontier.com/~r.bond