HELLO AI WORLD — MEET JETSON NANO
<table>
<thead>
<tr>
<th>WEBINAR AGENDA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Intro to Jetson Nano</strong></td>
</tr>
<tr>
<td>- AI for Autonomous Machines</td>
</tr>
<tr>
<td>- Jetson Nano Developer Kit</td>
</tr>
<tr>
<td>- Jetson Nano Compute Module</td>
</tr>
<tr>
<td><strong>Jetson Software</strong></td>
</tr>
<tr>
<td>- JetPack 4.2</td>
</tr>
<tr>
<td>- ML/DL Framework Support</td>
</tr>
<tr>
<td>- NVIDIA TensorRT</td>
</tr>
<tr>
<td>- Inferencing Benchmarks</td>
</tr>
<tr>
<td><strong>Application SDKs</strong></td>
</tr>
<tr>
<td>- DeepStream SDK</td>
</tr>
<tr>
<td>- Isaac Robotics SDK</td>
</tr>
<tr>
<td><strong>Getting Started</strong></td>
</tr>
<tr>
<td>- Jetson Nano Resources</td>
</tr>
<tr>
<td>- Hello AI World</td>
</tr>
<tr>
<td>- JetBot</td>
</tr>
<tr>
<td>- System Setup</td>
</tr>
<tr>
<td>- Tips and Tricks</td>
</tr>
</tbody>
</table>
JETSON POWERS AUTONOMOUS MACHINES

WAREHOUSE

DELIVERY

AGRICULTURE

RETAIL

INDUSTRIAL
JETSON NANO DEVELOPER KIT
$99 CUDA-X AI Computer

128 CUDA Cores | 4 Core CPU
4GB LPDDR4 Memory
472 GFLOPs
5W | 10W

Accessible and easy to use
# JETSON NANO DEVKIT SPECS

## Interfaces

<table>
<thead>
<tr>
<th>Interface</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>USB</td>
<td>(4x) USB 3.0 A (Host)</td>
</tr>
<tr>
<td>Camera</td>
<td>MIPI CSI-2 x2 (15-position Flex Connector)</td>
</tr>
<tr>
<td>Display</td>
<td>HDMI</td>
</tr>
<tr>
<td>Networking</td>
<td>Gigabit Ethernet (RJ45, PoE)</td>
</tr>
<tr>
<td>Wireless</td>
<td>M.2 Key-E with PCIe x1</td>
</tr>
<tr>
<td>Storage</td>
<td>MicroSD card (16GB UHS-1 recommended minimum)</td>
</tr>
<tr>
<td>40-Pin Header</td>
<td>UART</td>
</tr>
<tr>
<td>Power</td>
<td>5V DC (µUSB, Barrel Jack, PoE) - 5W</td>
</tr>
<tr>
<td>Size</td>
<td>80x100mm</td>
</tr>
</tbody>
</table>

## Processor

<table>
<thead>
<tr>
<th>Component</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>64-bit Quad-core ARM A57 @ 1.43GHz</td>
</tr>
<tr>
<td>GPU</td>
<td>128-core NVIDIA Maxwell @ 921MHz</td>
</tr>
<tr>
<td>Memory</td>
<td>4GB 64-bit LPDDR4 @ 1600MHz</td>
</tr>
<tr>
<td>Video Encoder</td>
<td>4Kp30</td>
</tr>
<tr>
<td>Video Decoder</td>
<td>4Kp60</td>
</tr>
</tbody>
</table>

## Distributors Include:
JETSON NANO
Compact AI Compute Module

128 CUDA Cores | 4 Core CPU
4GB LPDDR4 Memory
16GB eMMC 5.1
45x70mm
5W | 10W

$129 (1Ku)
Available June 2019
# JETSON NANO COMPUTE MODULE

## PROCESSOR

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>64-bit Quad-core ARM A57 @ 1.43GHz</td>
</tr>
<tr>
<td>GPU</td>
<td>128-core NVIDIA Maxwell @ 921MHz</td>
</tr>
<tr>
<td>Memory</td>
<td>4GB 64-bit LPDDR4 @ 1600MHz</td>
</tr>
<tr>
<td>Video Encoder</td>
<td>4Kp30</td>
</tr>
<tr>
<td>Video Decoder</td>
<td>4Kp60</td>
</tr>
</tbody>
</table>

## INTERFACES

<table>
<thead>
<tr>
<th>Interface</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>USB</td>
<td>USB 3.0</td>
</tr>
<tr>
<td>Camera</td>
<td>12 lanes MIPI CSI-2 (up to 4 cameras)</td>
</tr>
<tr>
<td>Display</td>
<td>HDMI</td>
</tr>
<tr>
<td>Networking</td>
<td>Gigabit Ethernet</td>
</tr>
<tr>
<td>PCIe</td>
<td>PCIe Gen2 x1/x2/x4</td>
</tr>
<tr>
<td>Storage</td>
<td>16GB eMMC 5.1</td>
</tr>
<tr>
<td>Other I/O</td>
<td>(4x) I2C</td>
</tr>
<tr>
<td>Power</td>
<td>5V DC, 5W</td>
</tr>
<tr>
<td>Size</td>
<td>45x70mm, 260-pin SODIMM connector</td>
</tr>
</tbody>
</table>

Production module available June 2019
THE JETSON FAMILY
From AI at the Edge to Autonomous Machines

**JETSON NANO**
- 5–10W
- 0.5 TFLOPS (FP16)
- 45mm x 70mm
- $129 / $99 (Devkit)

**JETSON TX1 → JETSON TX2 4GB**
- 7–15W
- 1–1.3 TFLOPS (FP16)
- 50mm x 87mm
- $299

**JETSON TX2 8GB | Industrial**
- 7–15W
- 1.3 TFLOPS (FP16)
- 50mm x 87mm
- $399–$749

**JETSON AGX XAVIER**
- 10–30W
- 11 TFLOPS (FP16) | 32 TOPS (INT8)
- 100mm x 87mm
- $1099

Multiple Devices — Same Software

AI at the Edge

Fully Autonomous Machines
# JETSON SOFTWARE

## DeepStream SDK
- Depth Estimation
- Object Detection
- Pose Estimation
- Gesture Recognition

## Isaac Robotics Engine
- Path Planning
- Autonomous Navigation
- Ecosystem Modules

## JetPack SDK
- TensorRT
- cuDNN
- VisionWorks
- OpenCV
- cuBLAS
- cuFFT
- Vulkan
- OpenGL
- libargus
- Video API
- Drivers
- Ecosystem

## JetPack Developer Tools
- Graphics
- Multimedia
- Sensors

## CUDA-X • Linux for Tegra • ROS

## Hardware
- Jetson Nano
- Jetson TX1/TX2
- Jetson AGX Xavier

[developer.nvidia.com/jetpack](developer.nvidia.com/jetpack)
## Package Versions

<table>
<thead>
<tr>
<th>Package</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>L4T BSP</td>
<td>32.1</td>
</tr>
<tr>
<td>Linux Kernel</td>
<td>4.9.140</td>
</tr>
<tr>
<td>Vulkan</td>
<td>1.1.1</td>
</tr>
<tr>
<td>OpenGL</td>
<td>4.6</td>
</tr>
<tr>
<td>OpenGL-ES</td>
<td>3.2.5</td>
</tr>
<tr>
<td>EGL</td>
<td>1.5</td>
</tr>
<tr>
<td>GLX</td>
<td>1.4</td>
</tr>
<tr>
<td>X11 ABI</td>
<td>24</td>
</tr>
<tr>
<td>Wayland</td>
<td>1.14</td>
</tr>
<tr>
<td>L4T Multimedia API</td>
<td>32.1</td>
</tr>
<tr>
<td>Argus Camera API</td>
<td>0.97</td>
</tr>
<tr>
<td>GStreamer</td>
<td>1.14.1</td>
</tr>
<tr>
<td>Nsight Systems</td>
<td>2019.3</td>
</tr>
<tr>
<td>Nsight Graphics</td>
<td>2018.7</td>
</tr>
<tr>
<td>Nsight Compute</td>
<td>1.0</td>
</tr>
<tr>
<td>Jetson GPIO</td>
<td>1.0</td>
</tr>
<tr>
<td>Jetson OS</td>
<td>Ubuntu 18.04</td>
</tr>
<tr>
<td>Host OS</td>
<td>Ubuntu 16.04 / 18.04</td>
</tr>
</tbody>
</table>

Install TensorFlow, PyTorch, Caffe, Caffe2, MXNet, ROS, and other GPU-accelerated libraries.

Available Now For Jetson
[developer.nvidia.com/jetpack](https://developer.nvidia.com/jetpack)
OPEN FRAMEWORK SUPPORT

MACHINE LEARNING
- Caffe
- Caffe2
- Keras
- mxnet
- PyTorch
- TensorFlow

ROBOTICS / IOT
- AWS Greengrass
- docker
- MPI
- ROS
- JETSON

JETSON
NVIDIA TensorRT

**TRAIN**
- TensorFlow
- Caffe2
- Keras
- mxnet
- PyTorch
- Caffe

**EXPORT**
- TF-TRT
- UFF
- ONNX
- .caffemodel

**OPTIMIZE**
- TensorRT Model Optimizer
  - Layer Fusion, Kernel Autotuning, GPU Optimizations, Mixed Precision, Tensor Layout, Batch Size Tuning

**DEPLOY**
- TensorRT Runtime Engine
  - C++ / Python
JETSON NANO RUNS MODERN AI

JETSON NANO RUNS MODERN AI

Inference

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-50</td>
<td>36</td>
<td></td>
<td></td>
<td>39</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inception-v4</td>
<td>11</td>
<td></td>
<td></td>
<td>10</td>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VGG-19</td>
<td>10</td>
<td></td>
<td></td>
<td>27</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSD MobileNet-v2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSD MobileNet-v2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSD MobileNet-v2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tiny YOLO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U-Net</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Super Resolution</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OpenPose</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

developer.nvidia.com/embedded/jetson-nano-dl-inference-benchmarks
JETSON NANO RUNS MODERN AI

Inference

- Coral Dev Board (Edge TPU)
- Raspberry Pi 3 + Intel Neural Compute Stick 2
- Jetson Nano
- Not supported/DNR

developer.nvidia.com/embedded/jetson-nano-dl-inference-benchmarks
NETWORK VIDEO RECORDER

- GigE x8 switch and POE injector
- Ethernet ports: 8
- Power Input: 48V
- AC/DC Power supply
- 5V
- JETSON NANO
- PCIE
- UART
- R5232
- R5485
- PCIe to SATA
- USB hub
- HDMI
- LAN (RJ45)
- USB 3.0 x 2
- Digital in x4
- Digital out x4
- NVR (Network Video Recorder)
# ISAAC SDK

## ISAAC OPEN TOOLBOX

<table>
<thead>
<tr>
<th>Sensor and Actuator Drivers</th>
<th>Core Libraries</th>
<th>GEMS</th>
<th>Reference DNN</th>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>KAYA (Nano)</td>
<td>CARTER (Xavier)</td>
<td>LINK (Multi Xavier)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## CUDA-X

- Jetson Nano
- Jetson TX2
- Jetson AGX Xavier

[developer.nvidia.com/isaac-sdk](developer.nvidia.com/isaac-sdk)
ISAAC ROBOTS

developer.nvidia.com/isaac-sdk
GETTING STARTED

Resources
Tutorials
System Setup
Tips and Tricks
Accessories
JETSON NANO RESOURCES

Tutorials

Projects

Developer Forums

Jetson Developer Zone
eLinux Wiki

Accessories
HELLO AI WORLD
Getting Started with Deep Learning

Pretrained Networks
NVIDIA Jetson JetPack | TensorRT
Realtime Inferencing

github.com/dusty-nv/jetson-inference
HELLO AI WORLD
Getting Started with Deep Learning

1. Download and Build the GitHub Repo
   
git clone http://github.com/dusty-nv/jetson-inference

2. Classifying Images from Command Line
3. Coding Your Own Recognition Program
4. Realtime Recognition from Live Camera
5. Detecting Objects in Images from Disk
6. Object Detection from Live Camera

github.com/dusty-nv/jetson-inference
1. Download and Build the GitHub Repo

2. Classifying Images from Command Line
   
   ```bash
   ./imagenet-console bear_0.jpg output_0.jpg
   ```

3. Coding Your Own Recognition Program

4. Realtime Recognition from Live Camera

5. Detecting Objects in Images from Disk

6. Object Detection from Live Camera

[GitHub repository link]

**HELLO AI WORLD**

Getting Started with Deep Learning
HELLO AI WORLD
Getting Started with Deep Learning

1. Download and Build the GitHub Repo
2. Classifying Images from Command Line
3. Coding Your Own Recognition Program
   ./my-recognition test-image.jpg
4. Realtime Recognition from Live Camera
5. Detecting Objects in Images from Disk
6. Object Detection from Live Camera

   github.com/dusty-nv/jetson-inference

#include <jetson-inference/imageNet.h>
#include <jetson-utils/loadImage.h>

int main( int argc, char** argv )
{
    // load the image recognition network with TensorRT
    imageNet* net = imageNet::Create(imageNet::GOOGLENET);

    // this variable will store the confidence of the classification (between 0 and 1)
    float confidence = 0.0;

    // classify the image with TensorRT on the GPU (hence we use the CUDA pointer)
    // this will return the index of the object class that the image was recognized as
    const int classIndex = net->Classify(imgCUDA, imgWidth, imgHeight, &confidence);

    // make sure a valid classification result was returned
    if( classIndex >= 0 )
    {
        // retrieve the name/description of the object class index
        const char* classDescription = net->GetClassDesc(classIndex);

        // print out the classification results
        printf("image is recognized as '%s' (class #%d) with %f%% confidence\n", classDescription, classIndex, confidence * 100.0f);
    }

    // free the network's resources before shutting down
    delete net;
    return 0;
}
HELLO AI WORLD
Getting Started with Deep Learning

1. Download and Build the GitHub Repo
2. Classifying Images from Command Line
3. Coding Your Own Recognition Program
4. Realtime Recognition from Live Camera
   ./imagenet-camera googlenet
5. Detecting Objects in Images from Disk
6. Object Detection from Live Camera

github.com/dusty-nv/jetson-inference
HELLO AI WORLD
Getting Started with Deep Learning

1. Download and Build the GitHub Repo
2. Classifying Images from Command Line
3. Coding Your Own Recognition Program
4. Realtime Recognition from Live Camera

5. Detecting Objects in Images from Disk
   
   ./detectnet-console dogs.jpg output.jpg coco-dog
   ./detectnet-console peds.jpg output.jpg multiped

6. Object Detection from Live Camera

github.com/dusty-nv/jetson-inference
HELLO AI WORLD
Getting Started with Deep Learning

1. Download and Build the GitHub Repo
2. Classifying Images from Command Line
3. Coding Your Own Recognition Program
4. Realtime Recognition from Live Camera
5. Detecting Objects in Images from Disk
6. Object Detection from Live Camera

   ./detectnet-camera <model-name>

<table>
<thead>
<tr>
<th>Object Detection Models</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>facenet</td>
<td>(faces)</td>
<td>multiped</td>
<td>(humans)</td>
</tr>
<tr>
<td>coco-dog</td>
<td>(dogs)</td>
<td>coco-bottle</td>
<td>(bottles)</td>
</tr>
<tr>
<td>coco-chair</td>
<td>(chairs)</td>
<td>coco-airplane</td>
<td>(airplanes)</td>
</tr>
</tbody>
</table>

github.com/dusty-nv/jetson-inference
TWO DAYS TO A DEMO

Training + Inference

AI WORKFLOW
Train using DIGITS and cloud/PC
Deploy to the field with Jetson

TRAINING GUIDES
All the steps required to follow to train your own models, including the datasets.

DEEP VISION PRIMITIVES
Image Recognition, Object Detection and Segmentation

github.com/dusty-nv/jetson-inference
JETBOT

~$250 DIY Autonomous Deep Learning Robotics Kit
Programmable through Jupyter IPython Notebooks
Trainable DNNs for obstacle detection, object following, path planning, and navigation
ROS support and Gazebo simulator available

Join our upcoming JetBot webinar, May 16 2019

github.com/NVIDIA-AI-IOT/JetBot
SYSTEM SETUP

- Device is booted from a MicroSD card
  - 16GB UHS-1 recommended minimum
- Download the SD card image from NVIDIA.com
- Flash the SD card image with Etcher program
  - From a Windows/Mac/Linux PC
  - You can also flash JetPack with NV SDK Manager
- Insert the MicroSD card into the slot located on the underside of the Jetson Nano module
- Connect keyboard, mouse, display, and power supply
- Board will automatically boot when power is applied
  - Green power LED will light

NVIDIA.com/JetsonNano-Start
POWER SUPPLIES

- 5V=2A Micro-USB charger
  - Adafruit #1995
- 5V=4A DC barrel jack adapter
  - Adafruit #1466
  - 5.5mm OD x 2.1mm ID x 9.5mm length
  - Place a jumper on header J48
- J41 Expansion Header, pins 2/4
  - Up to 5V=3A per pin (5V=6A total)
- Power over Ethernet (PoE)
  - Standard PoE supply is 48V
  - Use a PoE hat or 5V regulator

- J40 Button Header can disable Auto Power-On
  - Manual Power-On / Reset
  - Enter Recovery Mode
Different power mode presets: 5W and 10W

Default mode is 10W

Users can create their own presets, specifying clocks and online cores in /etc/nvpmodel.conf

```
< POWER_MODEL ID=1 NAME=5W >
  CPU_ONLINE CORE_0 1
  CPU_ONLINE CORE_1 1
  CPU_ONLINE CORE_2 0
  CPU_ONLINE CORE_3 0
  CPU_A57 MAX_FREQ 918000
  GPU MAX_FREQ 640000000
  EMC MAX_FREQ 1600000000
```

NVIDIA Power Model Tool

```
sudo nvpmodel -q  (for checking the active mode)
sudo nvpmodel -m 0  (for changing mode, persists after reboot)
sudo jetson_clocks  (to disable DVFS and lock clocks to max for active mode)
```
Run `sudo tegrastats` to launch the performance/utilization monitor:

<table>
<thead>
<tr>
<th>Memory</th>
<th>Memory Used / Total Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Utilization / Frequency (MHz)</td>
</tr>
<tr>
<td>GPU</td>
<td>Utilization / Frequency (MHz)</td>
</tr>
<tr>
<td>Thermal</td>
<td>Current Consumption (mW) / Average (mW)</td>
</tr>
</tbody>
</table>

Refer to the [L4T Developer Guide](docs.nvidia.com/jetson) for more options and documentation on the output.
USING GPIO

- Similar 40-pin header to rPI, 3.3V logic levels
- Adafruit Blinka + SeeedStudio Grove support
- Jetson.GPIO Python library
  - Compatible API with rPI.GPIO
  - Docs & samples in /opt/nvidia/jetson-gpio/
- sysfs I/O access from /sys/class/gpio/
  - Map GPIO pin: echo 38 > /sys/class/gpio/export
  - Set direction: echo out > /sys/class/gpio/gpio38/direction
  - Bit-banging: echo 1 > /sys/class/gpio/gpio38/value
  - Unmap GPIO: echo 38 > /sys/class/gpio/unexport
  - Query status: cat /sys/kernel/debug/gpio
- C/C++ programs (and other languages) can use same sysfs files
- I²C - libi2c for C/C++ and Python
## JETSON NANO ACCESSORIES

<table>
<thead>
<tr>
<th>Category</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Printable Enclosures</td>
<td><img src="image1.jpg" alt="Image" /></td>
</tr>
<tr>
<td>Battery Packs</td>
<td><img src="image2.jpg" alt="Image" /></td>
</tr>
<tr>
<td>5V Fans</td>
<td><img src="image3.jpg" alt="Image" /></td>
</tr>
<tr>
<td>Sensors &amp; Cameras</td>
<td><img src="image4.jpg" alt="Image" /></td>
</tr>
<tr>
<td>Carriers</td>
<td><img src="image5.jpg" alt="Image" /></td>
</tr>
<tr>
<td>GPIO Hats</td>
<td><img src="image6.jpg" alt="Image" /></td>
</tr>
</tbody>
</table>

[elinux.org/Jetson_Nano](https://elinux.org/Jetson_Nano)
CAMERA CAPTURE

- NVIDIA Argus (libargus)
  - Low-overhead offloaded ingest & ISP for MIPI CSI sensors
  - Docs & samples in /usr/src/tegra_multimedia_api/argus/
  - argus_camera - C++/Python wrapper library on GitHub

- GStreamer
  - nvarguscamerasrc element uses Argus internally
  - gst-launch-1.0 nvarguscamerasrc ! 'video/x-raw(memory:NVMM), \n    width=(int)1920, height=(int)1080, format=(string)NV12, \n    framerate=(fraction)30/1' ! nvoverlaysink -e
  - nvgstcapture camera viewer application

- V4L2
  - Interface with USB cameras and MIPI CSI YUV sensors (/dev/video)
  - libv4l (C/C++), pip install v4l2 (Python), v4l2src (GStreamer)

Up to three MIPI CSI-2 x4 cameras or four cameras in x4/x2 configurations (12 MIPI CSI-2 lanes total)
**VIDEO CODECS**

- Multi-stream HW encoder and decoder engines
- GStreamer
  - NV Encoder elements: `omxh265enc`, `omxh264enc`, ect.
    - `gst-launch-1.0 videotestsrc ! 'video/x-raw, format=(string)I420, width=(int)1920, height=(int)1080' ! omxh265enc ! matroskamux ! filesink location=test.mkv -e`
  - NV Decoder elements: `omxh265dec`, `omxh264dec`, ect.
    - `gst-launch-1.0 filesrc location=test.mkv ! matroskadecmux ! h265parse ! omxh265dec ! noverlaysink -e`
  - More pipelines in [L4T Accelerated GStreamer User Guide](#)
- V4L2 Extensions
  - NV Encoder: `/dev/nvhost-msenc` (YUV in, H.264/H.265 out)
  - NV Decoder: `/dev/nvhost-nvdec` (Bitstream in, NV12/YUV out)
  - Documentation + samples included with [L4T Multimedia API](#)
Zero Copy

- Shared memory fabric allows processor engines to access the same memory, without needing to copy between them

- CUDA Mapped Memory API’s
  - cudaHostAlloc(&cpuPtr, size, cudaHostAllocMapped);
  - cudaMemcpy(&gpuPtr, cpuPtr, 0);
  - No cudaMemcpy() required

- CUDA Unified Memory
  - cudaMallocManaged()
  - Coherent synchronization and caching
  - Disregards data movement on Jetson

- EGLStreams - graphics API interoperability

- Argus, NV V4L2 extensions, and DeepStream libraries are optimized for using ZeroCopy

[Diagram showing memory controller fabric with various interfaces such as PCIe x4, USB 3.0, DSI, eDP, HDMI, CSI x4, and more.]

docs.nvidia.com/cuda/cuda-for-tegra-appnote/
Thank you!

Developer Site
Getting Started
Hello AI World
DevTalk Forums
Visit the Wiki
developer.nvidia.com/jetson
nvidia.com/JetsonNano-Start
github.com/dusty-nv
devtalk.nvidia.com
eLinux.org/Jetson_Nano

Q&A: What can I help you build?

Dev Blog Jetson Nano Brings AI Computing to Everyone