**Deep Learning Nodes for ROS/ROS2**

This repo contains deep learning inference nodes and camera/video streaming nodes for ROS/ROS2 with support for Jetson Nano/TX1/TX2/Xavier NX/AGX Xavier and TensorRT.

The nodes use the image recognition, object detection, and semantic segmentation DNN's from the [jetson-inference](https://github.com/dusty-nv/jetson-inference) library and NVIDIA [Hello AI World](https://developer.nvidia.com/embedded/twodaystoademo) tutorial, which come with several built-in pretrained networks for classification, detection, and segmentation and the ability to load customized user-trained models.

The camera/video streaming nodes support the following input/output interfaces:

* MIPI CSI cameras
* V4L2 cameras
* RTP / RTSP
* Videos & Images
* Image sequences
* OpenGL windows

ROS Melodic and ROS2 Eloquent are supported, and the latest version of JetPack is recommended.

**Installation**

First, install the latest version of [JetPack](https://developer.nvidia.com/embedded/jetpack) on your Jetson.

Then, follow the steps below to install the needed components on your Jetson.

**jetson-inference**

These ROS nodes use the DNN objects from the [jetson-inference](https://github.com/dusty-nv/jetson-inference) project (aka Hello AI World). To build and install jetson-inference, see [this page](https://github.com/dusty-nv/jetson-inference/blob/master/docs/building-repo-2.md) or run the commands below:

$ cd ~

$ sudo apt-get install git cmake

$ git clone --recursive https://github.com/dusty-nv/jetson-inference

$ cd jetson-inference

$ mkdir build

$ cd build

$ cmake ../

$ make -j$(nproc)

$ sudo make install

$ sudo ldconfig

Before proceeding, it's worthwhile to test that jetson-inference is working properly on your system by following this step of the Hello AI World tutorial:

* [Classifying Images with ImageNet](https://github.com/dusty-nv/jetson-inference/blob/master/docs/imagenet-console-2.md)

**ROS/ROS2**

Install the ros-melodic-ros-base or ros-eloquent-ros-base package on your Jetson following these directions:

* ROS Melodic - [ROS Install Instructions](http://wiki.ros.org/melodic/Installation/Ubuntu)
* ROS2 Eloquent - [ROS2 Install Instructions](https://index.ros.org/doc/ros2/Installation/Eloquent/Linux-Install-Debians/)

Depending on which version of ROS you're using, install some additional dependencies and create a workspace:

**ROS Melodic**

$ sudo apt-get install ros-melodic-image-transport ros-melodic-vision-msgs

For ROS Melodic, create a Catkin workspace (~/ros\_workspace) using these steps:  
<http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment#Create_a_ROS_Workspace>

**ROS Eloquent**

$ sudo apt-get install ros-eloquent-vision-msgs \

ros-eloquent-launch-xml \

ros-eloquent-launch-yaml \

python3-colcon-common-extensions

For ROS Eloquent, create a workspace (~/ros\_workspace) to use:

$ mkdir -p ~/ros2\_example\_ws/src

**ros\_deep\_learning**

Next, navigate into your ROS workspace's src directory and clone ros\_deep\_learning:

$ cd ~/ros\_workspace/src

$ git clone https://github.com/dusty-nv/ros\_deep\_learning

Then build it - if you are using ROS Melodic, use catkin\_make. If you are using ROS2 Eloquent, use colcon build:

$ cd ~/ros\_workspace/

# ROS Melodic

$ catkin\_make

$ source devel/setup.bash

# ROS2 Eloquent

$ colcon build

$ source install/local\_setup.bash

The nodes should now be built and ready to use. Remember to source the overlay as shown above so that ROS can find the nodes.

**Testing**

Before proceeding, if you're using ROS Melodic make sure that roscore is running first:

$ roscore

If you're using ROS2, running the core service is no longer required.

**Video Viewer**

First, it's recommended to test that you can stream a video feed using the [video\_source](https://github.com/dusty-nv/ros_deep_learning" \l "video-source-node) and [video\_output](https://github.com/dusty-nv/ros_deep_learning" \l "video-output-node) nodes. See [Camera Streaming & Multimedia](https://github.com/dusty-nv/jetson-inference/blob/master/docs/aux-streaming.md) for valid input/output streams, and substitute your desired input and output argument below. For example, you can use video files for the input or output, or use V4L2 cameras instead of MIPI CSI cameras. You can also use RTP/RTSP streams over the network.

# ROS Melodic

$ roslaunch ros\_deep\_learning video\_viewer.ros1.launch input:=csi://0 output:=display://0

# ROS2 Eloquent

$ ros2 launch ros\_deep\_learning video\_viewer.ros2.launch input:=csi://0 output:=display://0

**imagenet Node**

You can launch a classification demo with the following commands - substitute your desired camera or video path to the input argument below (see [here](https://github.com/dusty-nv/jetson-inference/blob/master/docs/aux-streaming.md) for valid input/output streams).

Note that the imagenet node also publishes classification metadata on the imagenet/classification topic in a [vision\_msgs/Detection2DArray](http://docs.ros.org/melodic/api/vision_msgs/html/msg/Detection2DArray.html) message -- see the [Topics & Parameters](https://github.com/dusty-nv/ros_deep_learning#imagenet-node-1) section below for more info.

# ROS Melodic

$ roslaunch ros\_deep\_learning imagenet.ros1.launch input:=csi://0 output:=display://0

# ROS2 Eloquent

$ ros2 launch ros\_deep\_learning imagenet.ros2.launch input:=csi://0 output:=display://0

**detectnet Node**

To launch an object detection demo, substitute your desired camera or video path to the input argument below (see [here](https://github.com/dusty-nv/jetson-inference/blob/master/docs/aux-streaming.md) for valid input/output streams). Note that the detectnet node also publishes the metadata in a vision\_msgs/Detection2DArray message -- see the [Topics & Parameters](https://github.com/dusty-nv/ros_deep_learning#detectnet-node-1) section below for more info.

# ROS Melodic

$ roslaunch ros\_deep\_learning detectnet.ros1.launch input:=csi://0 output:=display://0

# ROS2 Eloquent

$ ros2 launch ros\_deep\_learning detectnet.ros2.launch input:=csi://0 output:=display://0

**segnet Node**

To launch a semantic segmentation demo, substitute your desired camera or video path to the input argument below (see [here](https://github.com/dusty-nv/jetson-inference/blob/master/docs/aux-streaming.md) for valid input/output streams). Note that the segnet node also publishes raw segmentation results to the segnet/class\_mask topic -- see the [Topics & Parameters](https://github.com/dusty-nv/ros_deep_learning#segnet-node-1) section below for more info.

# ROS Melodic

$ roslaunch ros\_deep\_learning segnet.ros1.launch input:=csi://0 output:=display://0

# ROS2 Eloquent

$ ros2 launch ros\_deep\_learning segnet.ros2.launch input:=csi://0 output:=display://0

**Topics & Parameters**

Below are the message topics and parameters that each node implements.

**imagenet Node**

| **Topic Name** | **I/O** | **Message Type** | **Description** |
| --- | --- | --- | --- |
| image\_in | Input | [sensor\_msgs/Image](http://docs.ros.org/melodic/api/sensor_msgs/html/msg/Image.html) | Raw input image |
| classification | Output | [vision\_msgs/Classification2D](http://docs.ros.org/melodic/api/vision_msgs/html/msg/Classification2D.html) | Classification results (class ID + confidence) |
| vision\_info | Output | [vision\_msgs/VisionInfo](http://docs.ros.org/melodic/api/vision_msgs/html/msg/VisionInfo.html) | Vision metadata (class labels parameter list name) |
| overlay | Output | [sensor\_msgs/Image](http://docs.ros.org/melodic/api/sensor_msgs/html/msg/Image.html) | Input image overlayed with the classification results |

| **Parameter Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| model\_name | string | "googlenet" | Built-in model name (see [here](https://github.com/dusty-nv/jetson-inference#image-recognition) for valid values) |
| model\_path | string | "" | Path to custom caffe or ONNX model |
| prototxt\_path | string | "" | Path to custom caffe prototxt file |
| input\_blob | string | "data" | Name of DNN input layer |
| output\_blob | string | "prob" | Name of DNN output layer |
| class\_labels\_path | string | "" | Path to custom class labels file |
| class\_labels\_HASH | vector<string> | class names | List of class labels, where HASH is model-specific (actual name of parameter is found via the vision\_info topic) |

**detectnet Node**

| **Topic Name** | **I/O** | **Message Type** | **Description** |
| --- | --- | --- | --- |
| image\_in | Input | [sensor\_msgs/Image](http://docs.ros.org/melodic/api/sensor_msgs/html/msg/Image.html) | Raw input image |
| detections | Output | [vision\_msgs/Detection2DArray](http://docs.ros.org/melodic/api/vision_msgs/html/msg/Detection2DArray.html) | Detection results (bounding boxes, class IDs, confidences) |
| vision\_info | Output | [vision\_msgs/VisionInfo](http://docs.ros.org/melodic/api/vision_msgs/html/msg/VisionInfo.html) | Vision metadata (class labels parameter list name) |
| overlay | Output | [sensor\_msgs/Image](http://docs.ros.org/melodic/api/sensor_msgs/html/msg/Image.html) | Input image overlayed with the detection results |

| **Parameter Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| model\_name | string | "ssd-mobilenet-v2" | Built-in model name (see [here](https://github.com/dusty-nv/jetson-inference#object-detection) for valid values) |
| model\_path | string | "" | Path to custom caffe or ONNX model |
| prototxt\_path | string | "" | Path to custom caffe prototxt file |
| input\_blob | string | "data" | Name of DNN input layer |
| output\_cvg | string | "coverage" | Name of DNN output layer (coverage/scores) |
| output\_bbox | string | "bboxes" | Name of DNN output layer (bounding boxes) |
| class\_labels\_path | string | "" | Path to custom class labels file |
| class\_labels\_HASH | vector<string> | class names | List of class labels, where HASH is model-specific (actual name of parameter is found via the vision\_info topic) |
| overlay\_flags | string | "box,labels,conf" | Flags used to generate the overlay (some combination of none,box,labels,conf) |
| mean\_pixel\_value | float | 0.0 | Mean pixel subtraction value to be applied to input (normally 0) |
| threshold | float | 0.5 | Minimum confidence value for positive detections (0.0 - 1.0) |

**segnet Node**

| **Topic Name** | **I/O** | **Message Type** | **Description** |
| --- | --- | --- | --- |
| image\_in | Input | [sensor\_msgs/Image](http://docs.ros.org/melodic/api/sensor_msgs/html/msg/Image.html) | Raw input image |
| vision\_info | Output | [vision\_msgs/VisionInfo](http://docs.ros.org/melodic/api/vision_msgs/html/msg/VisionInfo.html) | Vision metadata (class labels parameter list name) |
| overlay | Output | [sensor\_msgs/Image](http://docs.ros.org/melodic/api/sensor_msgs/html/msg/Image.html) | Input image overlayed with the classification results |
| color\_mask | Output | [sensor\_msgs/Image](http://docs.ros.org/melodic/api/sensor_msgs/html/msg/Image.html) | Colorized segmentation class mask out |
| class\_mask | Output | [sensor\_msgs/Image](http://docs.ros.org/melodic/api/sensor_msgs/html/msg/Image.html) | 8-bit single-channel image where each pixel is a classID |

| **Parameter Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| model\_name | string | "fcn-resnet18-cityscapes-1024x512" | Built-in model name (see [here](https://github.com/dusty-nv/jetson-inference#semantic-segmentation) for valid values) |
| model\_path | string | "" | Path to custom caffe or ONNX model |
| prototxt\_path | string | "" | Path to custom caffe prototxt file |
| input\_blob | string | "data" | Name of DNN input layer |
| output\_blob | string | "score\_fr\_21classes" | Name of DNN output layer |
| class\_colors\_path | string | "" | Path to custom class colors file |
| class\_labels\_path | string | "" | Path to custom class labels file |
| class\_labels\_HASH | vector<string> | class names | List of class labels, where HASH is model-specific (actual name of parameter is found via the vision\_info topic) |
| mask\_filter | string | "linear" | Filtering to apply to color\_mask topic (linear or point) |
| overlay\_filter | string | "linear" | Filtering to apply to overlay topic (linear or point) |
| overlay\_alpha | float | 180.0 | Alpha blending value used by overlay topic (0.0 - 255.0) |

**video\_source Node**

| **Topic Name** | **I/O** | **Message Type** | **Description** |
| --- | --- | --- | --- |
| raw | Output | [sensor\_msgs/Image](http://docs.ros.org/melodic/api/sensor_msgs/html/msg/Image.html) | Raw output image (BGR8) |

| **Parameter** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| resource | string | "csi://0" | Input stream URI (see [here](https://github.com/dusty-nv/jetson-inference/blob/master/docs/aux-streaming.md#input-streams) for valid protocols) |
| codec | string | "" | Manually specify codec for compressed streams (see [here](https://github.com/dusty-nv/jetson-inference/blob/master/docs/aux-streaming.md#input-options) for valid values) |
| width | int | 0 | Manually specify desired width of stream (0 = stream default) |
| height | int | 0 | Manually specify desired height of stream (0 = stream default) |
| framerate | int | 0 | Manually specify desired framerate of stream (0 = stream default) |
| loop | int | 0 | For video files: 0 = don't loop, >0 = # of loops, -1 = loop forever |

**video\_output Node**

| **Topic Name** | **I/O** | **Message Type** | **Description** |
| --- | --- | --- | --- |
| image\_in | Input | [sensor\_msgs/Image](http://docs.ros.org/melodic/api/sensor_msgs/html/msg/Image.html) | Raw input image |

| **Parameter** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| resource | string | "display://0" | Output stream URI (see [here](https://github.com/dusty-nv/jetson-inference/blob/master/docs/aux-streaming.md#output-streams) for valid protocols) |
| codec | string | "h264" | Codec used for compressed streams (see [here](https://github.com/dusty-nv/jetson-inference/blob/master/docs/aux-streaming.md#input-options) for valid values) |
| bitrate | int | 4000000 | Target VBR bitrate of encoded streams (in bits per second) |