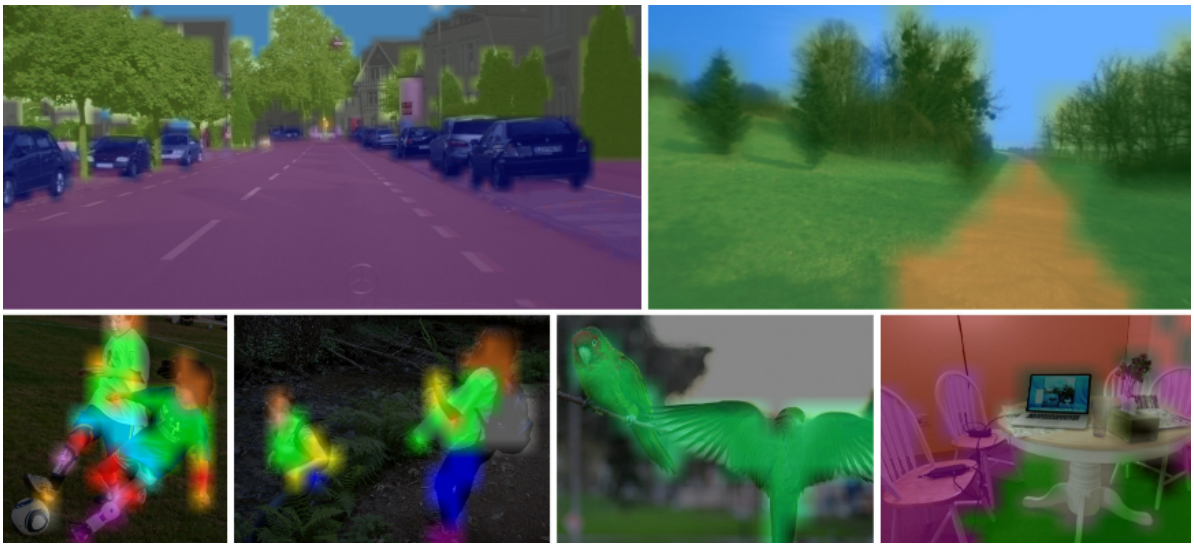
semantic segmentation

1.Introduction to Semantic Segmentation

The next deep learning feature we will introduce in this tutorial is semantic segmentation. Semantic segmentation is based on image recognition, except that classification occurs at the pixel level rather than the entire image. This is achieved by convolving the pre trained image recognition backbone, which converts the model into a fully convolutional network (FCN) that can be labeled by pixel. Segmentation is particularly useful for environmental perception, as it generates dense per pixel classifications of many different potential objects for each scene, including foreground and background.



SegNet accepts 2D images as input and outputs a second image with per pixel classification mask coverage. Each pixel of the mask corresponds to the classified object category. SegNet can be used from Python and C++.

Download other models

Various pre trained segmentation models for FCN-ResNet18 networks with real-time performance on Jetson. The following is a table of pre trained semantic segmentation models available for use, as well as the relevant network parameters used to load them. They are based on the 21 class FCN-ResNet18 network, trained using PyTorch on various datasets and resolutions, and exported to ONnano format to load TensorRT.

Dataset	Resolution	CLI Argument	Accuracy	Jetson Nano	Jetson Xavier
Cityscapes	512x256	`fcn-resnet18-cityscapes-512x256`	83.3%	48 FPS	480 FPS
Cityscapes	1024x512	`fcn-resnet18-cityscapes-1024x512`	87.3%	12 FPS	175 FPS
Cityscapes	2048x1024	`fcn-resnet18-cityscapes-2048x1024`	89.6%	3 FPS	47 FPS
DeepScene	576x320	`fcn-resnet18-deepscene-576x320`	96.4%	26 FPS	360 FPS
DeepScene	864x480	`fcn-resnet18-deepscene-864x480`	96.9%	14 FPS	190 FPS
Multi-Human	512x320	`fcn-resnet18-mhp-512x320`	86.5%	34 FPS	370 FPS
Multi-Human	640x360	`fcn-resnet18-mhp-640x360`	87.1%	23 FPS	325 FPS
Pascal VOC	320x320	`fcn-resnet18-voc-320x320`	85.9%	45 FPS	508 FPS
Pascal VOC	512x320	`fcn-resnet18-voc-512x320`	88.5%	34 FPS	375 FPS
SUN RGB-D	512x400	`fcn-resnet18-sun-512x400`	64.3%	28 FPS	340 FPS
SUN RGB-D	640x512	`fcn-resnet18-sun-640x512`	65.1%	17 FPS	224 FPS

Here, you can download the model you want based on the website: https://github.com/dusty-nv/jetson-inference

2.Image semantic segmentation

The following is an example of using urban landscape models to segment urban street scenes:After building the project, please ensure that your terminal is located in the aarch64/bin directory:

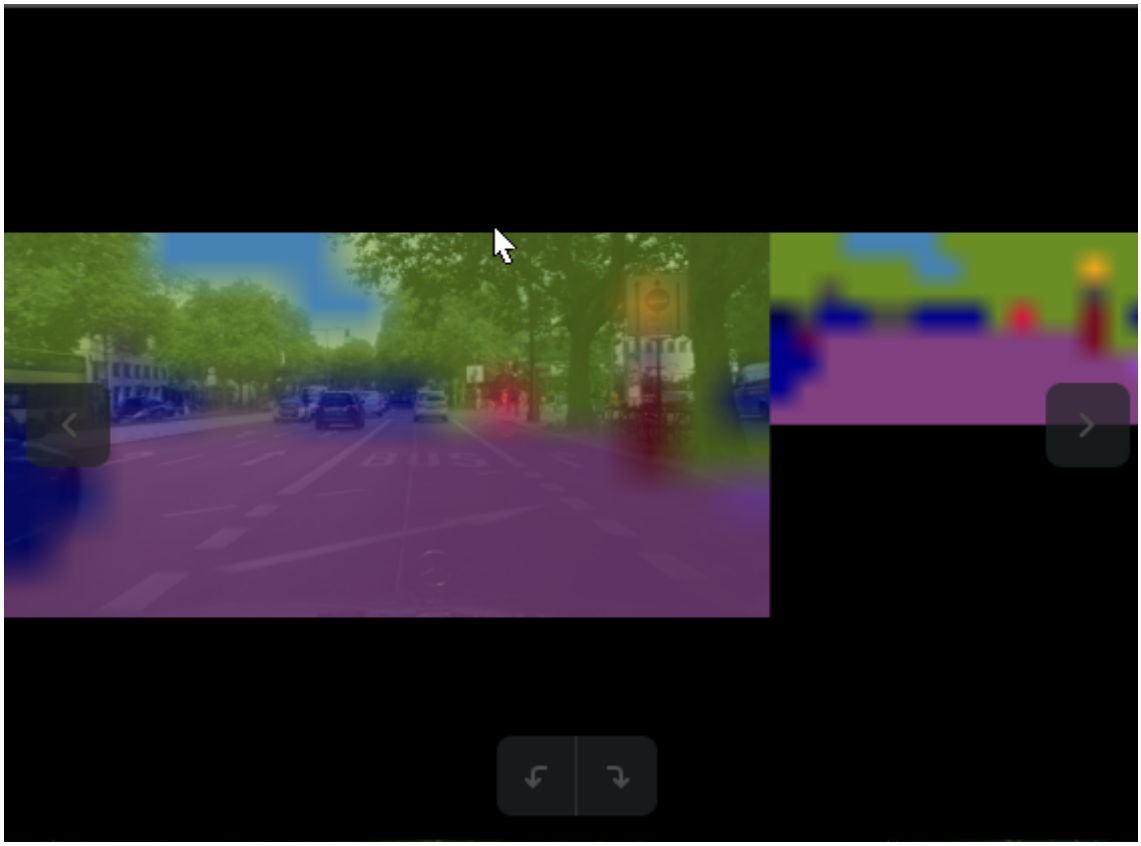
```
cd jetson-inference/build/aarch64/bin
```

Here are some examples of using the fcn resnet18 cityscapes model:

--Network=You can put your downloaded model files in this. For example, what I have here is fcn resnet18 cityscapes# C++

```
# C++
$ ./segnet --network=fcn-resnet18-cityscapes images/city_0.jpg
images/test/output.jpg

# Python
$ ./segnet.py --network=fcn-resnet18-cityscapes images/city_0.jpg
images/test/output.jpg
```



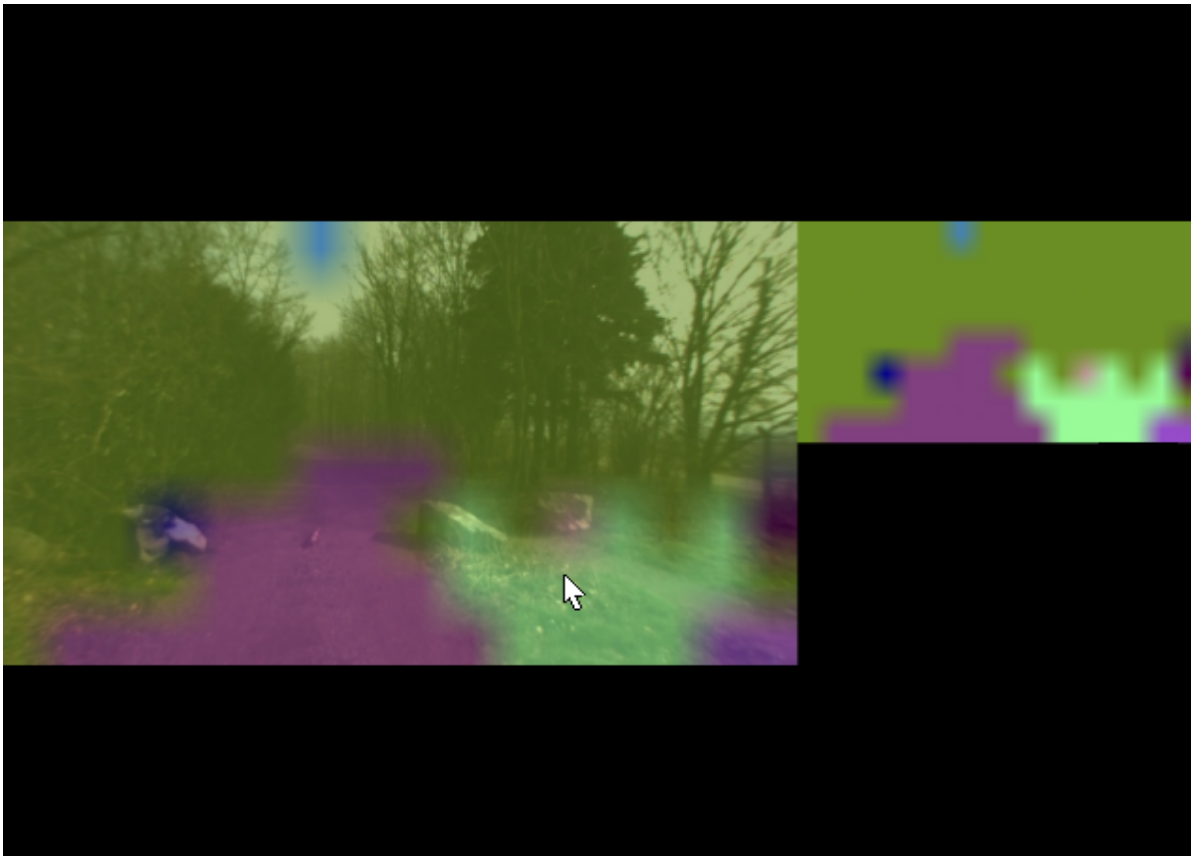
The following example is a DeepScene dataset composed of off-road forest paths and vegetation, which helps with path tracking for outdoor robots.

C++

```
$ ./segnet --network=fcn-resnet18-deepscene images/trail_0.jpg
images/test/output_overlay.jpg                # overlay
$ ./segnet --network=fcn-resnet18-deepscene --visualize=mask images/trail_0.jpg
images/test/output_mask.jpg  # mask
```

python

```
$ ./segnet.py --network=fcn-resnet18-deepscene images/trail_0.jpg
images/test/output_overlay.jpg               # overlay
$ ./segnet.py --network=fcn-resnet18-deepscene --visualize=mask
images/trail_0.jpg images/test/output_mask.jpg # mas
```



Note: If you are building your own environment, you must download the model file to the network folder to run the above program. By using the image we provide, you can directly input the above program

3.Run real-time camera segmentation demonstration

The semaet.cpp/semaet.py samples we previously used can also be used for real-time camera streaming. The supported camera types include:

- MIPI CSI cameras (csi://0)
- V4L2 cameras (/dev/video0)
- RTP/RTSP streams (rtsp://username:password@ip:port)

Here are some typical scenarios for starting the program - related to available models

C++

```
$ ./segnet --network=<model> csi://0                    # MIPI CSI camera
$ ./segnet --network=<model> /dev/video0                # V4L2 camera
$ ./segnet --network=<model> /dev/video0 output.mp4     # save to video file
```

python

```
$ ./segnet.py --network=<model> csi://0                 # MIPI CSI camera
$ ./segnet.py --network=<model> /dev/video0             # V4L2 camera
$ ./segnet.py --network=<model> /dev/video0 output.mp4  # save to video file
```

The model we can choose from here is fcn resnet18 mhp.The OpenGL window displays a real-time camera stream overlaid with segmented outputs, as well as a solid segmentation mask for clarity. Here are some examples to try using with different models:

```
# C++
$ ./segnet --network=fcn-resnet18-deepscene csi://0

# Python
$ ./segnet.py --network=fcn-resnet18-deepscene csi://0
```