













Machine Learning



Your robot can understand what it sees

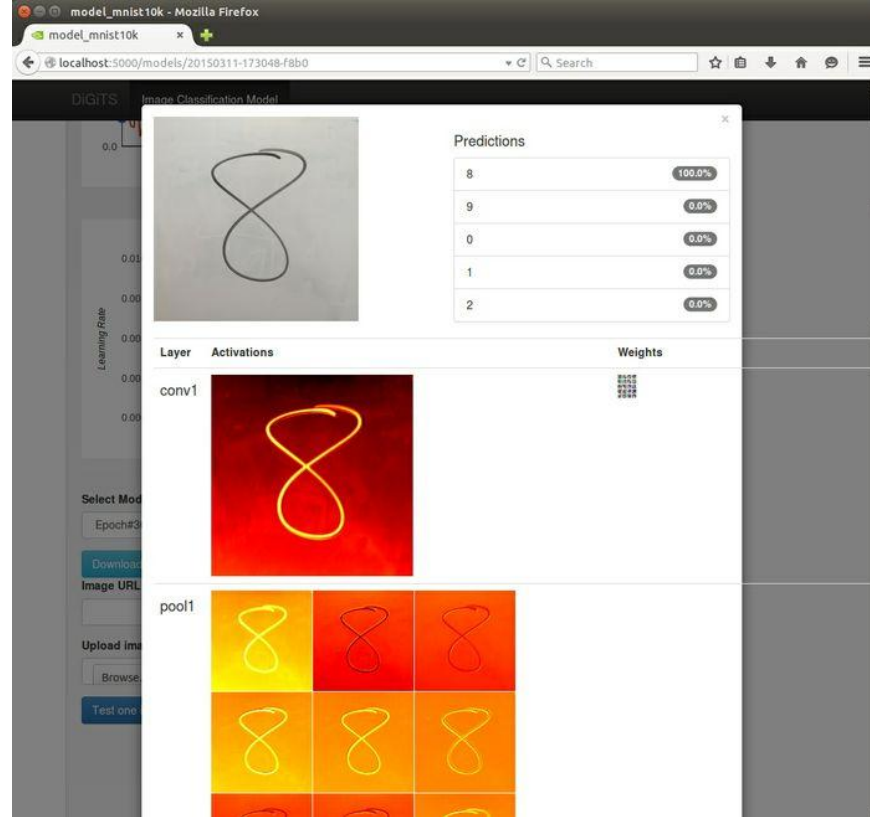
	err angle (degree)	-12		12		host car estimation
	err toMarking_ML (m)	-1		1		host car ground truth
	err toMarking_MR (m)	-1		1		traffic car estimation
	err toMarking_M (m)	-1		1		traffic car ground truth

Autonomous driving, 2.5x speed

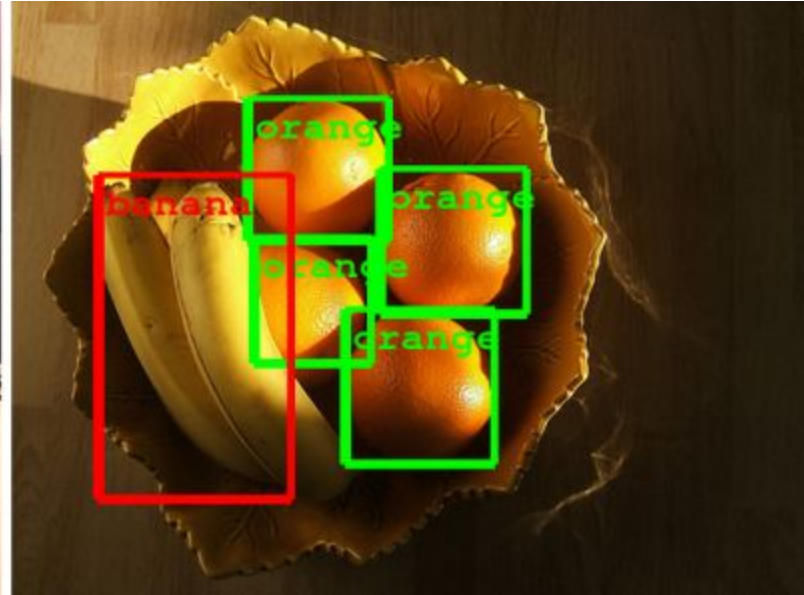
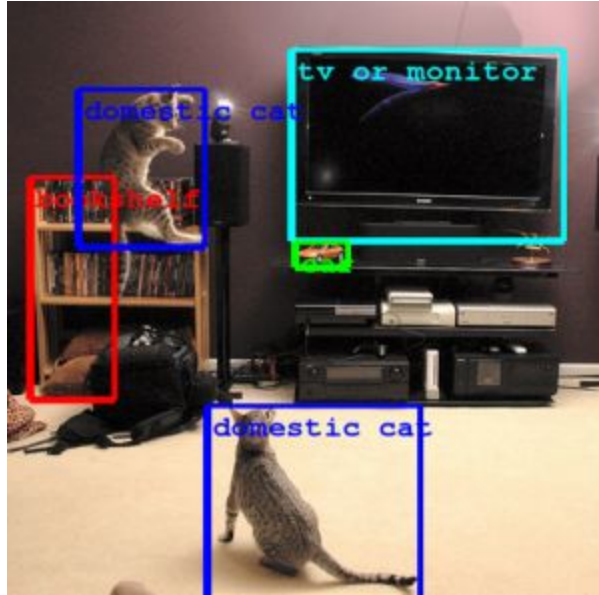




Too hot: Pixel segmentation (too much for FRC, for now)



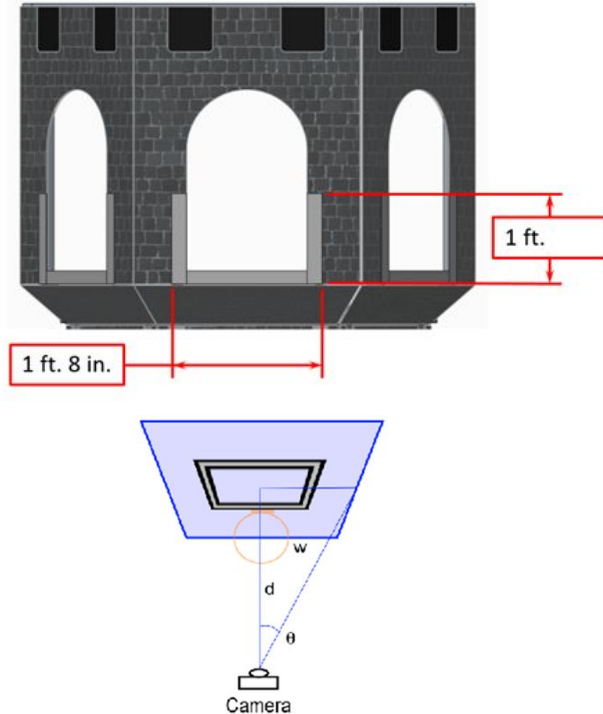
Too cold: identifying which category of object is in image



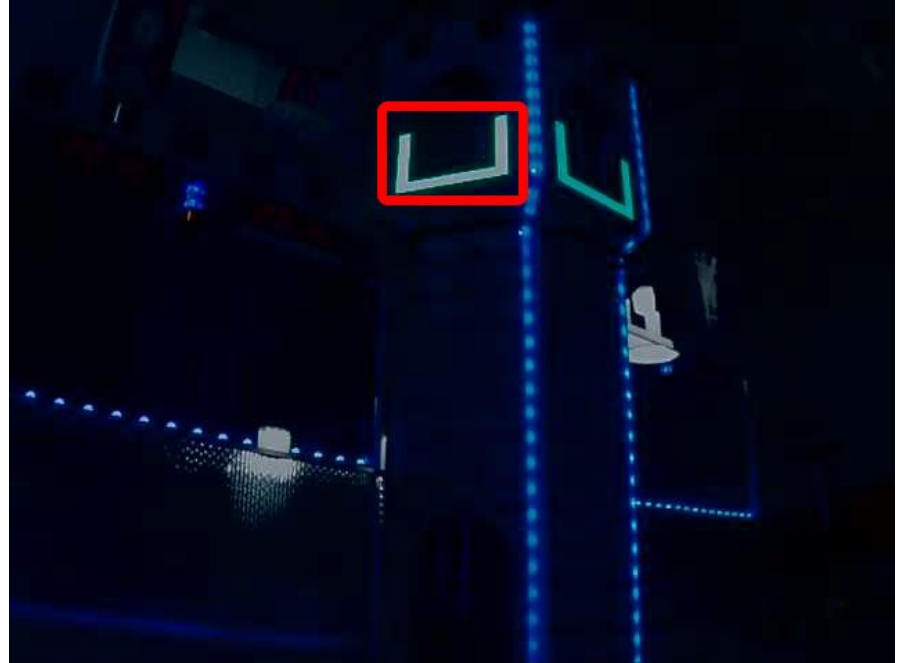
Just right: determine where in an image an object is

**Machines can
learn?**

How would you mathematically define this U shape?



Geometry matters -- and why vision is tough to program, so why not let the machine solve the problem?

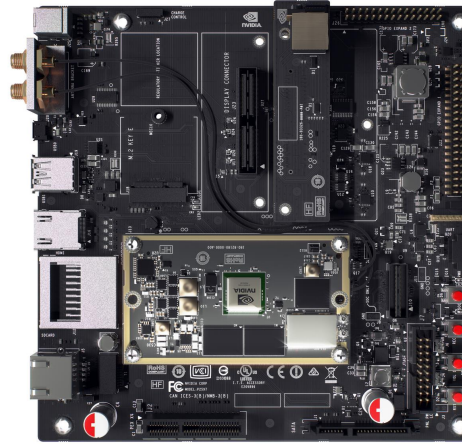


The Left U is very different than the right U. You and I can tell both are targets. Can the machine?

Software tools

- NVIDIA CUDA / CUDNN
Library for efficient math computation on GPUs
- NVIDIA DIGITS
Webpage tool to train and test a neural network
- Caffe (NVIDIA fork)
Neural network software that uses CUDA/CUDNN, and DIGITS runs for us with the correct settings
- NVIDIA TensorRT
Software to run the neural network efficiently on the Jetson TX1
Gets higher frame per second performance with other processor shortcuts

Hardware diagram



USB camera to either device, pro/cons with each.
Two cameras?

Robot Software flow

Show: Python, Camera, TensorRT, Python to networktables

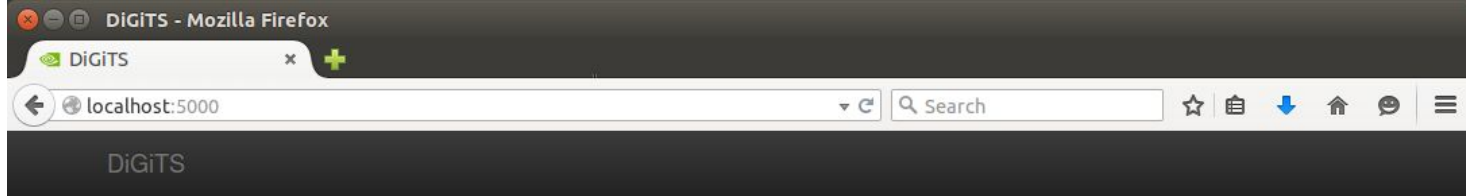
Gap

Then: C++ program on robot reads networktables values and moves robot

Building a neural network

You can train a neural network anywhere (go to your fastest computer). You can do this on your laptop with DIGITS and Caffe.

After you have a trained neural network, we then copy the file to the Jetson to run with TensorRT.



Home

Datasets

New Dataset
Images ▾

In progress

[dataset_imagenet@256x256](#)

Delete

Submitted: 05:29:57 PM (53 seconds ago)
Status: Running

Completed

[dataset_minst_10k@256x256](#)

Delete

Submitted: 05:21:27 PM
Status: Done after 31 seconds

[voc_cropped@256x256](#)

Delete

Submitted: 05:14:31 PM
Status: Done after 2 minutes, 26 seconds

Models

New Model
Images ▾

In progress

[model_mnist10k](#)

Delete

Submitted: 05:30:48 PM (2 seconds ago)
Status: Running

[LeNet_model_voc_cropped@256x256](#)

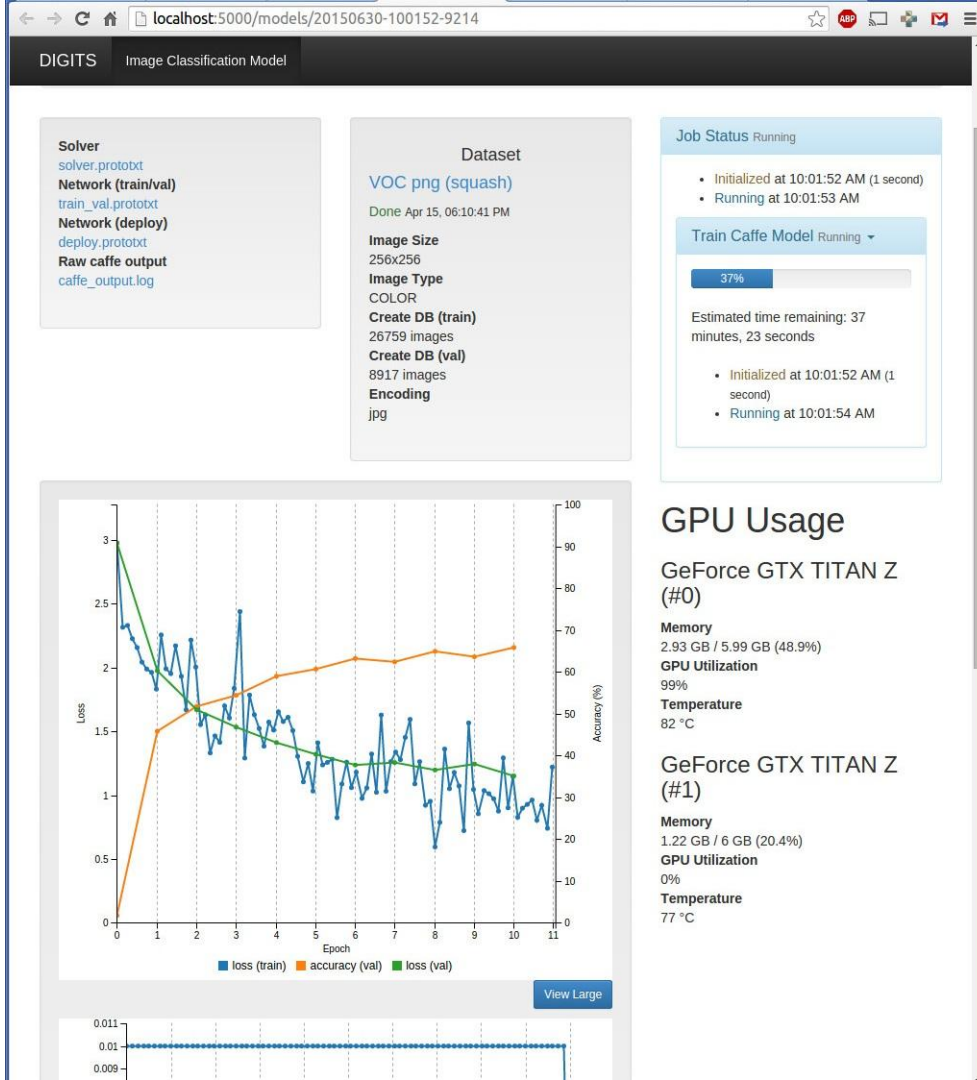
Delete

Submitted: 05:18:43 PM (12 minutes, 7 seconds ago)
Status: Running

Completed

Datasets: large collection of example pictures of the target U shape.

Models: A neural network layout trained for a specific task in your dataset (find the U)



Accuracy close to 70%
and only 37% finished
training.

Solver
solver.prototxt
Network (train/val)
train_val.prototxt
Network (deploy)
deploy.prototxt

Download
network files

Dataset

Database1
Done 01:00:29 PM
Image Size
256x256
Image Type
COLOR
Create DB (train)
137500 images
Create DB (val)
6066 images

Job Status (Hover)

- Initialized at 08:04:29 PM (1 second)
- Running at 08:04:30 PM

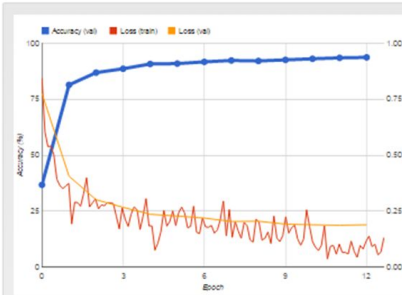
Train Caffe Model (Hover)

40%

Estimated time remaining: 9 hours, 41 minutes

- Initialized at 08:04:29 PM (1 second)
- Running at 08:04:31 PM

Training Status



Accuracy and loss
values during training



Learning rate

Select Model
Epoch: 12

Download

Image URL

Upload Image

Test one image

Upload Image List
 No file chosen

Accept a list of filenames or urls (you can use your val set)

Number of images use from the file
100

Learn Rate (1 to use all)

Number of images to show per category
9

Test several images (This takes a while, be patient)

Classification on the
fly with most recent
or previous network
snapshots

After training, this lower
area lets you try out
your NN with a new
picture

**How does one
write in this
“Python”?**

Before we start...

- No main method/function (executes from the top down)
- No semicolons to end lines
- Use of classes is not necessary (as in Java)
- Variables do not require a type
- No use of brackets, imposes strict indentation requirements instead

A sample

```
def count_numbers(amount):  
    num = 0  
    while num <= amount:  
        print('Number is ' + str(num))  
        num += 1  
    return True  
  
if count_numbers(20):  
    print('Number counting was successful')
```

Food for thought

- The names of the functions have underscores instead of camelCase
- Bools are capitalized
- There's no increment by one (++) operator
- Variables do not have a type specifier