Summer school

Robotics Multimodal Deep Learning for Object Recognition

He Liu, Feng Wang

Department of Computer Science
Tsinghua University

Outline

1. **Background**
2. RGB-D Dataset
3. Related technology
4. Network architecture
5. Points and Bonus

Background

Tsinghua University

Robots



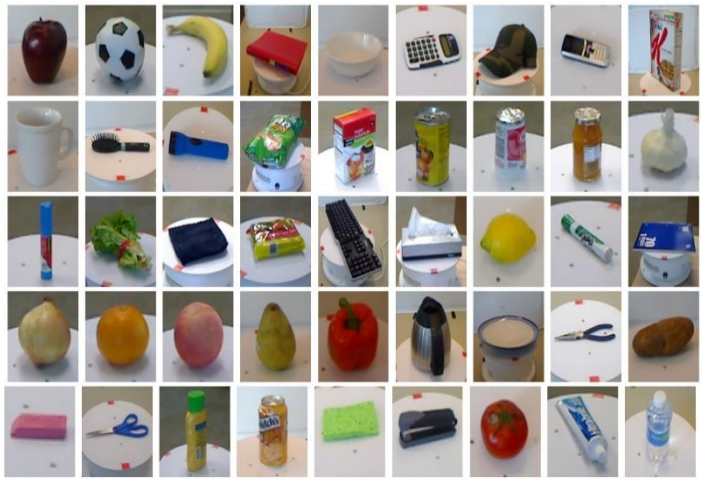



Background

Tsinghua University

KINECT sensor to obtain data

RGB-D dataset



RGB-D sensors are ubiquitous in many robotic systems. They are inexpensive, widely supported by open source software, do not require complicated hardware and provide unique sensing capabilities.

Background

Target:

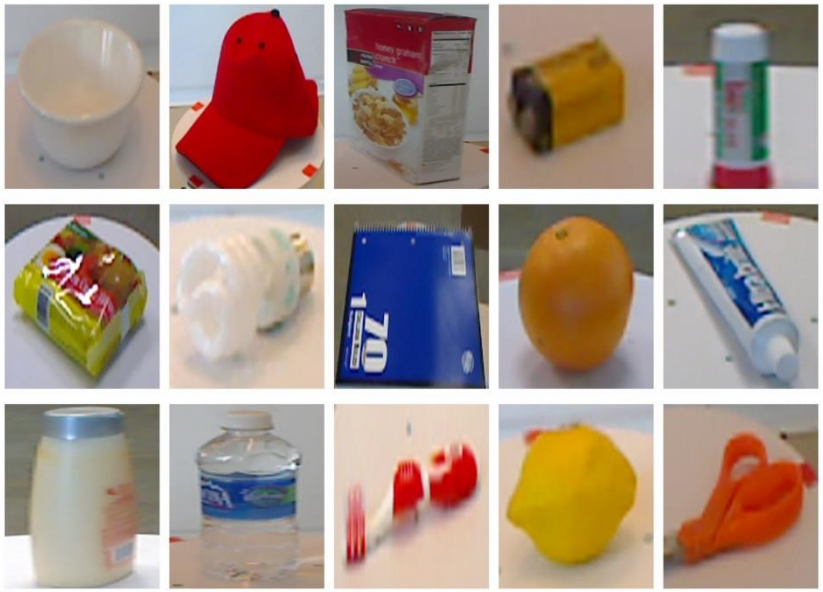
Design a multimodal (combine RGB and depth) network architecture classification

Outline

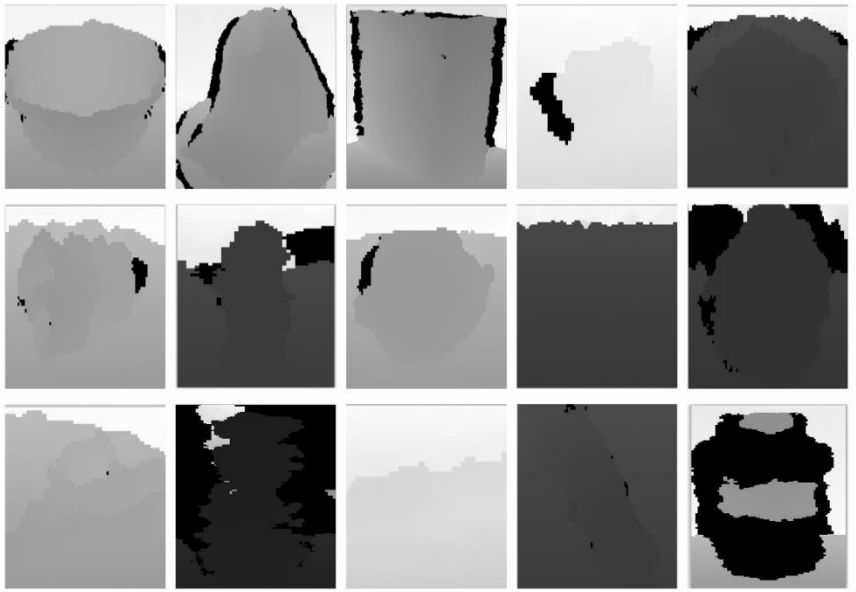
RGB-D dataset

Tsinghua University

RGB images



Depth images



15 classes totally

Outline

1. Background
2. RGB-D dataset
3. **Related technology**
4. Network architecture
5. Points and bonus

Related technology

Tensorlayer is a Deep Learning and Reinforcement Learning library extended from Google TensorFlow.

Advantages:

- **Easily customized**
- **Assembled for tackling real-world machine learning problems**

Outline

1. Background
2. RGB-D dataset
3. Related technology
4. **Experiments**
5. Points and bonus

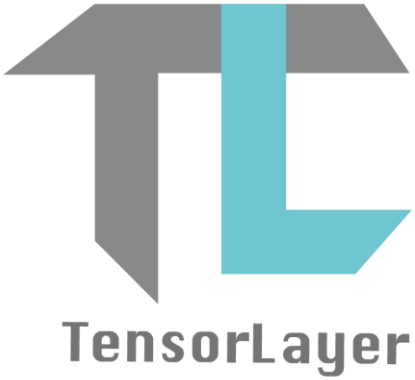
Experiment

- **Target:**
  - multimodal(combine RGB and depth) classification

- **Task:**
  - Design your own net for single model(RGB and depth apart) classification
  - Analysis the single model result(give your loss, acc…)
  - Design your own network for multimodal classification

- **Data:**
  - Dataset: Washington RGB-D
  - Classes: 15 classes of different objects, both have RGB and depth images

Experiment tools

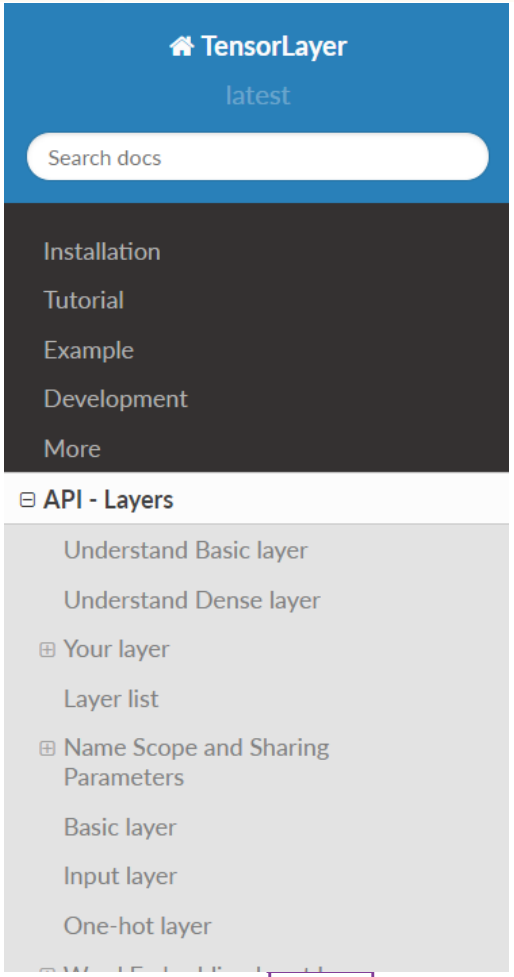


Install:
pip install tensorlayer

Reference:
http://tensorlayer.readthedocs.io/en/latest/index.html#

Experiment details

- **How to design your own net?**
  - Open tensorlayer website
  - Choose API-Layers， all layers supported by tensorlayer are listed
  - If you don’t know how to use them, see the example of mnist

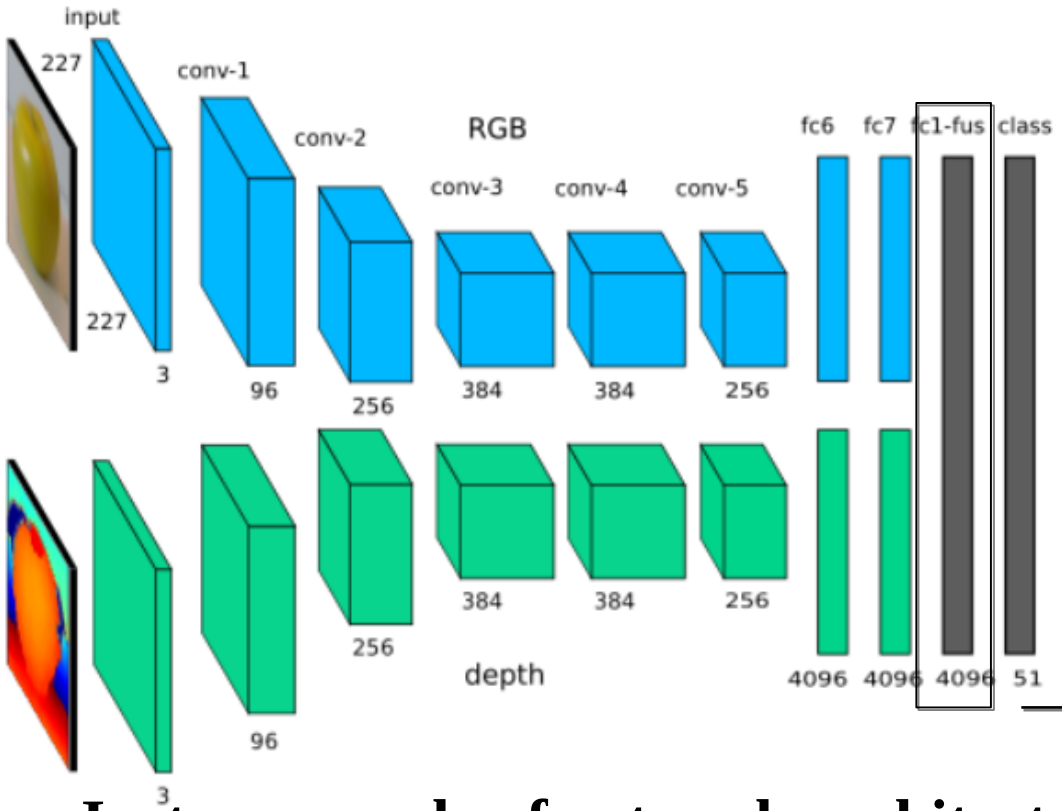


Experiment details

- **How to combine different modal together?**

RGB net

Depth net



Combine (concat layer)

Just a example of network architecture

Outline

1. Background
2. RGB-D dataset
3. Related technology
4. Experiment
5. **Points and bonus**

Points and Bonus

Points:

1.Train the two individual stream networks using RGB image and depth image and give the accuracy.

50 points

2.Train the fusion network and give the accuracy

45 points

3.Analyze the result and give the confusion matrix if time is enough.

5 points

4.Give the brief report to show your work.

Points and Bonus

Bonus:

The more tricks you use to arise your accuracy of the network, the higher points you will get.

Tricks:

1. Various ways of data augmentation
2. More reasonable Network

.....

Points and Bonus

Results:

```
Epoch 492 of 500 took 6.878548s
   train loss: 0.000000
   train acc: 1.000000
   val loss: 1.772480
   val acc: 0.856875
Epoch 492 of 500 took 10.887568s, loss 0.001158
Epoch 493 of 500 took 6.863887s
   train loss: 0.000000
   train acc: 1.000000
   val loss: 1.677745
   val acc: 0.861250
Epoch 493 of 500 took 10.861353s, loss 0.000855
Epoch 494 of 500 took 6.857479s
   train loss: 0.000001
   train acc: 1.000000
   val loss: 1.574762
   val acc: 0.859687
Epoch 494 of 500 took 10.830899s, loss 0.000299
Epoch 495 of 500 took 6.870256s
   train loss: 0.000004
   train acc: 1.000000
   val loss: 2.085051
   val acc: 0.861875
Epoch 495 of 500 took 10.850719s, loss 0.004398
Epoch 496 of 500 took 6.883532s
   train loss: 0.000000
   train acc: 1.000000
   val loss: 2.260445
   val acc: 0.851250
Epoch 496 of 500 took 10.860419s, loss 0.000757
Epoch 497 of 500 took 6.886390s
   train loss: 0.000004
   train acc: 1.000000
   val loss: 2.514290
   val acc: 0.853125
Epoch 497 of 500 took 10.885096s, loss 0.002660
Epoch 498 of 500 took 6.885203s
   train loss: 0.000000
   train acc: 1.000000
   val loss: 2.284768
   val acc: 0.850938
Epoch 498 of 500 took 10.865541s, loss 0.001031
Epoch 499 of 500 took 6.877015s
   train loss: 0.000000
   train acc: 1.000000
   val loss: 2.218112
   val acc: 0.855625
Epoch 499 of 500 took 10.863680s, loss 0.000586
Epoch 500 of 500 took 6.870010s
   train loss: 0.000000
   train acc: 1.000000
   val loss: 2.246570
   val acc: 0.857187
Epoch 500 of 500 took 10.848072s, loss 0.000035
Total training time: 5405.666162s
```

```
I tensorflow/core/common_runtime/gpu/gpu_device.cc:906] DMA: 0
I tensorflow/core/common_runtime/gpu/gpu_device.cc:916] 0:   Y
I tensorflow/core/common_runtime/gpu/gpu_device.cc:975] Creating TensorFlow device (/gpu:0) -> (device: 0, name: GeForce GTX 1080 Ti, pci bus id: 0000:4b:00.0)
  [TL] InputLayer  input_layer: (?, 80, 80, 3)
  [TL] Conv2dLayer conv_1_1: shape:[3, 3, 3, 32] strides:[1, 1, 1, 1] pad:SAME act:relu
  [TL] Conv2dLayer conv_1_2: shape:[3, 3, 32, 32] strides:[1, 1, 1, 1] pad:SAME act:relu
  [TL] PoolLayer   pool_1_1: ksize:[1, 2, 2, 1] strides:[1, 2, 2, 1] padding:SAME pool:max_pool
  [TL] Conv2dLayer conv_2_1: shape:[3, 3, 32, 32] strides:[1, 1, 1, 1] pad:SAME act:relu
  [TL] Conv2dLayer conv_2_2: shape:[3, 3, 32, 32] strides:[1, 1, 1, 1] pad:SAME act:relu
  [TL] PoolLayer   pool_2_1: ksize:[1, 2, 2, 1] strides:[1, 2, 2, 1] padding:SAME pool:max_pool
  [TL] Conv2dLayer conv_3_1: shape:[3, 3, 32, 64] strides:[1, 1, 1, 1] pad:SAME act:relu
  [TL] Conv2dLayer conv_3_2: shape:[3, 3, 64, 64] strides:[1, 1, 1, 1] pad:SAME act:relu
  [TL] PoolLayer   pool_3_1: ksize:[1, 2, 2, 1] strides:[1, 2, 2, 1] padding:SAME pool:max_pool
  [TL] FlattenLayer flatten: 6400
  [TL] DenseLayer  fc1: 128 relu
  [TL] DropoutLayer drop1: keep:0.500000 is_fix:False
  [TL] DenseLayer  output_layer: 51 identity
Traceback (most recent call last):
  File "changeMain.py", line 79, in <module>
    tl.layers.initialize_global_variables(session)
NameError: name 'session' is not defined
unreal@unreal-Default-string:~/RGBD/liu$ python changeMain.py
I tensorflow/stream_executor/dso_loader.cc:135] successfully opened CUDA library libcublas.so.8.0 locally
I tensorflow/stream_executor/dso_loader.cc:135] successfully opened CUDA library libcudnn.so.5 locally
I tensorflow/stream_executor/dso_loader.cc:135] successfully opened CUDA library libcufft.so.8.0 locally
I tensorflow/stream_executor/dso_loader.cc:135] successfully opened CUDA library libcuda.so.1 locally
I tensorflow/stream_executor/dso_loader.cc:135] successfully opened CUDA library libcurand.so.8.0 locally
['water_bottle', 'notebook', 'rubber_eraser', 'camera', 'glue_stick', 'food_bag', 'scissors', 'hand_towel', 'ball', 'pliers', 'onion', 'tomato', 'apple', 'dry_battery', 'calculato
r', 'toothpaste', 'banana', 'coffee_mug', 'food_can', 'keyboard', 'sponge', 'pitcher', 'marker', 'food_box', 'cereal_box', 'bell_pepper', 'binder', 'potato', 'flashlight', 'shampo
o', 'food_jar', 'instant_noodles', 'comb', 'soda_can', 'plate', 'mushroom', 'greens', 'lime', 'bowl', 'garlic', 'kleenex', 'lemon', 'food_cup', 'stapler', 'lightbulb', 'peach', 'c
ap', 'orange', 'pear', 'cell_phone', 'toothbrush']
W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE3 instructions, but these are available on your machine and could speed up CPU
 computations.
W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE4.1 instructions, but these are available on your machine and could speed up C
PU computations.
W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE4.2 instructions, but these are available on your machine and could speed up C
PU computations.
W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use AVX instructions, but these are available on your machine and could speed up CPU
computations.
W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use AVX2 instructions, but these are available on your machine and could speed up CPU
 computations.
W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use FMA instructions, but these are available on your machine and could speed up CPU
computations.
I tensorflow/core/common_runtime/gpu/gpu_device.cc:885] Found device 0 with properties:
name: GeForce GTX 1080 Ti
major: 6 minor: 1 memoryClockRate (GHz) 1.582
pciBusID 0000:4b:00.0
Total memory: 10.91GiB
Free memory: 10.57GiB
I tensorflow/core/common_runtime/gpu/gpu_device.cc:906] DMA: 0
I tensorflow/core/common_runtime/gpu/gpu_device.cc:916] 0:   Y
I tensorflow/core/common_runtime/gpu/gpu_device.cc:975] Creating TensorFlow device (/gpu:0) -> (device: 0, name: GeForce GTX 1080 Ti, pci bus id: 0000:4b:00.0)
  [TL] InputLayer  input_layer: (?, 80, 80, 3)
  [TL] Conv2dLayer conv_1_1: shape:[3, 3, 3, 32] strides:[1, 1, 1, 1] pad:SAME act:relu
  [TL] Conv2dLayer conv_1_2: shape:[3, 3, 32, 32] strides:[1, 1, 1, 1] pad:SAME act:relu
  [TL] PoolLayer   pool_1_1: ksize:[1, 2, 2, 1] strides:[1, 2, 2, 1] padding:SAME pool:max_pool
```

Points and Bonus

- **The location of dataset, loadData.py and main.py?**
  - **/home/share/rgbd or /home/share/rgbd.tar.gz**
  - **/home/share/Summer-school-of-THU**
- **Send your report to the Email address:**
  - **w-f17@mails.tsinghua.edu.cn before 5:00 p.m**

清華大學
Tsinghua University

THANK YOU