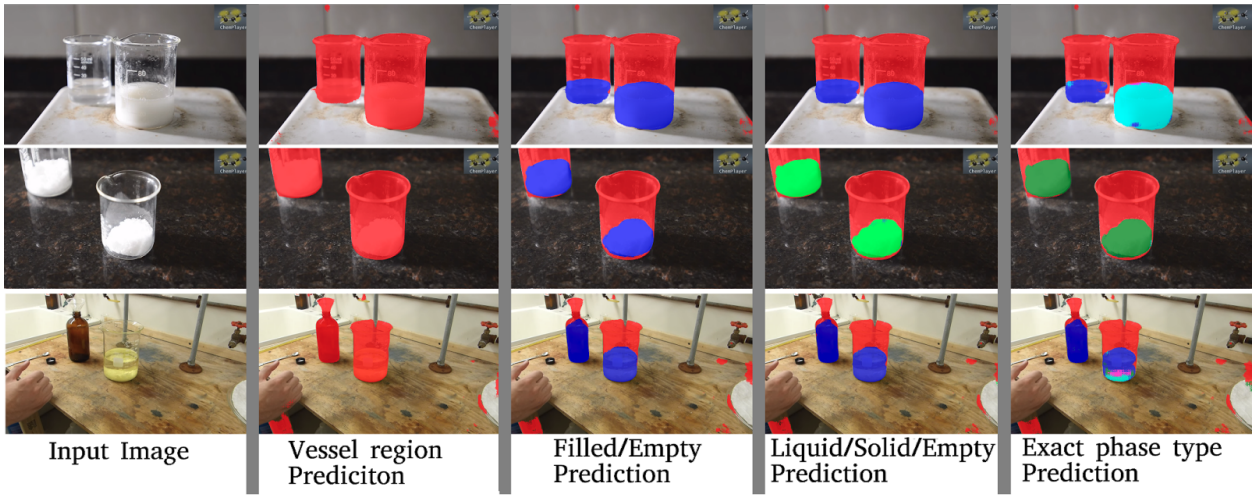
Classification and segmentation of materials in transparent vessel using fully convolutional neural network (FCN)

Neural nets for the purpose of recognition and segmentation of materials in transparent vessels, with emphasis on chemistry glassware. This task include labeling of each pixel in the image according to several level of classes shown in the image:

1) Vessel regions and background regions.
2) Filled vessel regions and empty vessel regions.
3) Liquid phase and solid phase regions.
4) Exact phase pixelwise classification (Solid,Liquid,Powder,Suspension,Foam…)

Dataset of annotated images of materials in glass vessels and their pixelwise semantic segmentation, is supplied to support this task and can be download from:
https://drive.google.com/file/d/0B6njwynsu2hXelJJOFdqRjhGWWM/view?usp=sharing



Details input/output

The fully convolutional neural network receive an image with material in a vessel and perform semantic segmentation of the image, such that that each pixel the in the image is assigned several labels. The network performed the pixelwise labeling in several level of class granularity and return for each level an image where the value of each pixel is the phase/material/object of this pixel in the image. All the predictions are generated simultaneously in one iteration of the net.

The output segmentation maps/images are as following (See image):

a) **Vessel/Background:** For each pixel assign value of 1 if it in the vessel and 0 otherwise.

b) **Filled/Empty:** similar to above but also distinguish between filled and empty region of the vessel. For each pixel assign one of the 3 values: 0) Background, 1) Empty vessel. 2) Filled vessel
c) **Phase type:** Similar to above but distinguish between liquid and solid regions of the filled vessel. For each pixel assign one of the 4 values: 0) Background, 1) Empty vessel. 2) Liquid. 3) Solid.
d) **Fine grain phase type:** Similar to above but distinguish between fine grain specific phases regions of the filled vessel. For each pixel assign one of 15 values: 1) BackGround. 2) Vessel. 3) Liquid. 4) Liquid Phase two. 5) Suspension. 6) Emulsion. 7) Foam. 8) Solid. 9) Gel. 10) Powder. 11) Granular. 12) Bulk. 13) Bulk Liquid. 14) Solid Phase two. 15) Vapor.

Requirements

This network was run and trained with Python 2.7 Anaconda package and Tensorflow 1.1.
The training was done using Nvidia GTX 1080, on Linux Ubuntu 16.04.

Setup

1) Download and install python anaconda 2.7 and Tensorflow 1.1
2) Download the code from the repository.
3) Download pretrained vgg16 net and put in the */Model_Zoo* subfolder in the main code folder. A pretrained vgg16 net can be download from: ftp://mi.eng.cam.ac.uk/pub/mttt2/models/vgg16.npy or https://drive.google.com/file/d/0B6njwynsu2hXZWcwX0FKTGJKRWs/view?usp=sharing
4) Download dataset of images materials in transparent vessel and extract in */Data_Zoo* folder in the main code dir. The dataset can be download from: https://drive.google.com/file/d/0B6njwynsu2hXelJJOFdqRjhGWWM/view?usp=sharing

5) If you not interested in training the net then download a pretrained model and extract to /log folder in the main code dir. The pretrained model could be download from: https://drive.google.com/file/d/0B6njwynsu2hXMjdLcjNfb2tNSUE/view?usp=sharing

Tutorial

Predicting pixel wise labels:

Run: *Inference.py*

Notes: Make sure Image_dir variable a point to a valid image folders, and /log folder contain a trained net.

Training network:

Run: Train.py

Note: Make sure you downloaded the dataset to /Data_Zoo folder

Evaluating net performance using intersection over union (IOU):

Run: *Evaluate_Net_IOU.py*

Notes: Make sure you downloaded the dataset to /Data_Zoo and the /log folder contain a trained net.

Background

The net is based on fully convolutional neural net described here:
https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf
The main modification is that the last prediction layer is split to give prediction in several level of granularity for each pixel. Similarly training of the network was done with several loss function simultaneously one for each set of classes. See BuildNetVgg16.py for the network structure.

The code is based on the pretrained VGG16 model by Marvin Teichmann