Deep Learning Toolkit for Medical Imaging

Martin Rajchl
open science, open source
scope
comparison

**DLTK**
- hackable
- low-threshold access
- speciality ops for MI
- scalable
- tensorflow

**The DeepMedic**
- out-of-the-box
- image segmentation
- theano

**NiftyNet**
- full applications
- high-level config
- modules
- tensorflow

Logos for various deep learning frameworks and tools, including Caffe, Caffe2, Chainer, Cognitive Toolkit, MATLAB, mxnet, PaddlePaddle, PyTorch, TensorFlow, and more...
features
Overview

What is biomedical image analysis and why is it needed? Biomedical images are measurements of the human body on different scales (i.e. microscopic, macroscopic, etc.). They come in a wide variety of imaging modalities (e.g. a CT scanner, an ultrasound machine, etc.) and measure a physical property of the human body (e.g. radiodensity, the opacity to X-rays). These images are interpreted by domain experts (e.g. a radiologist) for clinical tasks (e.g. a diagnosis) and have a large impact on decision making of physicians.
tutorials & low entry threshold

https://github.com/DLTK/DLTK/tree/master/examples/applications/MRBrainS13_tissue_segmentation

https://github.com/DLTK/DLTK/blob/master/examples/tutorials/02_Building_a_model_fn.ipynb

https://github.com/DLTK/DLTK/blob/master/examples/tutorials/02_Building_a_model_fn.ipynb
### IXI Dataset

In this project we have collected nearly 600 MR images from normal, healthy subjects. The MR image acquisition protocol for each subject includes:

- T1, T2 and PD-weighted images
- MRA images
- Diffusion-weighted images (15 directions)

The data has been collected at three different hospitals in London:

- Hammersmith Hospital using a Philips 3T system (details of scanner parameters)
- Guy's Hospital using a Philips 1.5T system (details of scanner parameters)
- Institute of Psychiatry using a GE 1.5T system (details of the scanner parameters not available at the moment)

http://brain-development.org/ixi-dataset/

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download</td>
</tr>
<tr>
<td>Extraction</td>
</tr>
<tr>
<td>Parse info, labels, etc.</td>
</tr>
<tr>
<td>Validate</td>
</tr>
<tr>
<td>Resampling</td>
</tr>
</tbody>
</table>

### MRBrainS

Welcome to the MRBrainS website. On September 26th, 2013 we organized the Grand Challenge on MR Brain Image Segmentation workshop at the MICCAI in Nagoya, Japan, where we launched this evaluation framework. The aim of the MRBrainS evaluation framework is to compare (semi-)automatic algorithms for segmentation of grey matter, white matter and cerebrospinal fluid on multi-sequences (T1-weighted, T2-weighted inversion recovery and FLAIR) 3 Tesla MRI scans of the brain. More information about the workshop and the results of the workshop challenge can be found here.

http://mrbrains13.isi.uu.nl/

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register</td>
</tr>
<tr>
<td>Download</td>
</tr>
<tr>
<td>Copy to data/</td>
</tr>
<tr>
<td>Run example</td>
</tr>
</tbody>
</table>
full example applications
High-level training & deploy scripts

```python
print('Starting training...')
try:
    for i in range(MAX_STEPS // EVAL_EVERY_N_STEPS):
        nn.train(
            input_fn=train_input_fn,
            hooks=[train_qinit_hook, step_cnt_hook],
            steps=EVAL_EVERY_N_STEPS)
if args.run_validation:
    results_val = nn.evaluate(
        input_fn=val_input_fn,
        hooks=[val_qinit_hook, val_summary_hook],
        steps=EVAL_STEPS)
    print('Step = {}; val loss = {:.5f};'.format(
        results_val['global_step'],
        results_val['loss']))
```

High-level `tf.estimator` API for training

```python
def model_fn(features, labels, mode, params):
    
    Model function to construct a tf.estimator.EstimatorSpec. It creates a network given input features (e.g. from a dlk.io.abstract_reader) and training targets (labels). Further, loss, optimiser, evaluation ops and custom tensorboard summary ops can be added. For additional information, please refer to https://www.tensorflow.org/api_docs/python/tf/estimator/EstimatorSpec.

    
    # 5. Return EstimatorSpec object
    return tf.estimator.EstimatorSpec(mode=mode,
        predictions=net_output_ops,
        loss=loss,
        train_op=train_op,
        eval_metric_ops=eval_metric_ops)
```

Standardised `tf.estimator` interface for feed forward networks

Easy to read deploy scripts
scalable reference implementations

- Pre-built networks
- Validated performance of components
- Independent of input tenor shapes (future: rank)
- Quickly scale to new problem
a quick regression experiment

Toy task: Age regression from T1w MR images
IXI database, 2mm isotropic, random crops

IXI dataset, https://brain-development.org/
a quick regression experiment

JSON configs for experimental setups and high-level parameterisation

Residual Unit

Identity Mappings in Deep Residual Networks, He et al. 2016

Dense Unit

Densely Connected Convolutional Networks, Huang et al. 2016

ResNext Unit


Squeeze-Excitation Unit

Squeeze-and-Excitation Networks, Hu et al., 2018
a quick regression experiment

<table>
<thead>
<tr>
<th>Model</th>
<th>MAE</th>
<th>num weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>Residual</td>
<td>4.929 yrs</td>
<td>9.6m</td>
</tr>
<tr>
<td>Dense</td>
<td>5.189 yrs</td>
<td>1.8m</td>
</tr>
<tr>
<td>ResNext</td>
<td>4.764 yrs</td>
<td>2.3m</td>
</tr>
<tr>
<td>SE-ResNext</td>
<td>4.199 yrs</td>
<td>2.4m</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AGE</th>
<th>mean</th>
<th>std</th>
<th>min</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>47.35</td>
<td>16.76</td>
<td>20.17</td>
<td>81.94</td>
</tr>
<tr>
<td>train</td>
<td>48</td>
<td>17.14</td>
<td>20.17</td>
<td>81.94</td>
</tr>
<tr>
<td>test</td>
<td>43.89</td>
<td>14.02</td>
<td>25.53</td>
<td>71.21</td>
</tr>
</tbody>
</table>
validated performance

Kamnitsas, EMMA, BRATS challenge 1st, 2017

Pawlowski, synapse challenge, multi-organ segm., 1st place., 2017

Bai, Human-level performance on UK Biobank, 2018
### pre-processing & augmentation

<table>
<thead>
<tr>
<th>Pre-processing</th>
<th>Augmentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intensity normalisation (volume stats!):</td>
<td>Add noise:</td>
</tr>
<tr>
<td>-1 to 1 (CT, quantitative images)</td>
<td>- Gaussian noise</td>
</tr>
<tr>
<td>Whitening zero mean, unit std (weighted MRI)</td>
<td>- Random offsets</td>
</tr>
<tr>
<td>0 to 1</td>
<td>Flipping (where meaningful)</td>
</tr>
<tr>
<td>Spatial normalisation (computationally expensive, typically offline)</td>
<td>Random (elastic) deformations</td>
</tr>
<tr>
<td>- Resampling/Reslicing</td>
<td>Random cropping (ideally class-balanced, c.f. sampling)</td>
</tr>
<tr>
<td>- Registration (e.g. MNI)</td>
<td></td>
</tr>
</tbody>
</table>

Bias correction
class balancing

Under-represented classes (e.g. pathologies, small anatomical structures, etc) could either be undersampled (e.g. random sampling) or under-penalised (e.g. mean pixel-wise losses).

- median-frequency reweighted
- batch stats reweighted
- mean smooth Dice/MIOU loss

- fixed count sampling
- moving stats sampling

For a performance comparison, see Pawlowski et al. NIPS WS 2017.
model zoo

NeuroNet, Rajchl, 2018

Pawlowski, 2017

Bai, 2018

Rajchl, 2016
design choices
seamless integration into Tensorflow

Data handling
- Data
- Preprocessing
- Augmentation & Sampling
- Queueing & Batching

Model Definition
- Network
- Layers

Training
- Loss
- Optimiser
- Metrics

Evaluation & Prediction
- Model Distribution
- Custom Inference
- Metrics

Within DLTK
- TF
- DLTK & TF
Data I/O

- **Load fn**
- **Memory**
- **tf.placeholder**

**Training Data**
- **Write tf.records**
- **Database**
- **Pre-fetch**
- **Decode from tf.records**

**Python generator w/ pre-processing and augmentation**
- **tf.dataset**
- **Pre-fetch**

**Queue**

**Network graph**
dltk reader

python generator w/ pre-processing and augmentation

- tf.dataset
- pre-fetch
- queue

training data
read fn
reader
training examples
exemplary research with DLTK
neuronet: fast and robust reproduction of multiple brain image segmentation pipelines
Rajchl et al. MIDL 2018
UK Biobank case: Raw T1w MR image, FSL First, MALP-EM, SPM tissue, FSL fast, MALP-EM tissue segmentations.

Exemplary failure cases due to head rotation (mid), CNN (right)

Rajchl, MIDL, 2018
neuronet
neuronet

Random UK Biobank case: Raw T1w MR image, FSL First, MALP-EM, SPM tissue, FSL fast, MALP-EM tissue segmentations.

Comparative DSC accuracy for single networks, nn_all and nn_tissue

Rajchl, MIDL, 2018
Multi-Modal Learning from Unpaired Images: Application to Multi-Organ Segmentation in CT and MRI

Valindria et al. IEEE WACV 2018
learning from unpaired images

Valindria et al., IEEE WACV, 2018
learning from unpaired images
learning from unpaired images

Valindria et al., IEEE WACV, 2018
implicit weight uncertainty in neural networks

Pawlowski et al. arXiv 2018
implicit weight uncertainty

Toy example: model accuracy versus model uncertainty of compared methods
implicit weight uncertainty

(a) Hypernetwork $G$

(b) Main Network

Bayes by Hypernet

Pawlowski et al, arXiv, 2018
implicit weight uncertainty

<table>
<thead>
<tr>
<th>Method</th>
<th>Error [%]</th>
<th>MNIST AUC</th>
<th>Outlier AUC</th>
<th>Runtime [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>0.80</td>
<td>0.99</td>
<td>0.71</td>
<td>710</td>
</tr>
<tr>
<td>Ensemble</td>
<td>0.49</td>
<td>0.99</td>
<td>0.65</td>
<td>2721</td>
</tr>
<tr>
<td>BbB</td>
<td>0.72</td>
<td>0.97</td>
<td>0.41</td>
<td>2892</td>
</tr>
<tr>
<td>Dropout</td>
<td>0.47</td>
<td>0.99</td>
<td>0.58</td>
<td>1224</td>
</tr>
<tr>
<td>MNF</td>
<td>0.63</td>
<td>0.99</td>
<td>0.58</td>
<td>21811</td>
</tr>
<tr>
<td>BbH (ours)</td>
<td>0.56</td>
<td>0.97</td>
<td>0.48</td>
<td>11504</td>
</tr>
</tbody>
</table>
future developments
TensorFlow Hub

Introduction

TensorFlow Hub is a library to foster the publication, discovery, and consumption of reusable parts of machine learning models. A module is a self-contained piece of a TensorFlow graph, along with its weights and assets, that can be reused across different tasks in a process known as transfer learning.

Modules contain variables that have been pre-trained for a task using a large dataset. By reusing a module on a related task, you can:

- train a model with a smaller dataset,
- improve generalization, or
- significantly speed up training.

```python
import tensorflow as tf
import tensorflow_hub as hub

with tf.Graph().as_default():
    embeddings = embed(["A long sentence.", "single-word", "http://example.com"])

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    sess.run(tf.tables_initializer())

print(sess.run(embeddings))
```
Conclusion

Try it out!
acknowledgements
resources

dltk.github.io

Martin Rajchl
m.rajchl@imperial.ac.uk
@m_rajchl
@mrajchl

medium.com/tensorflow