scikit-surgerytf Documentation

Matt Clarkson

Jul 28, 2022

Contents

CHAPTER 1

Features/Networks

- Liver Segmentation UNet: Based on https://doi.org/10.1007/978-3-319-24574-4_28.
- The usual FashionMNIST example, for learning purposes.

Source code is avaialble on GitHub.

1.1 scikit-surgerytf

Author: Matt Clarkson

scikit-surgerytf is part of the SNAPPY software project, developed at the Wellcome EPSRC Centre for Interventional and Surgical Sciences, part of University College London (UCL).

scikit-surgerytf supports Python 3.6+, and tensorflow >= 2.0.0.

The aim of scikit-surgerytf is to provide a home for various Tensor Flow examples and utilities and to show best practice. It's NOT meant to be a layer on-top of Tensor Flow or provide a new kind-of platform. The aim is that researchers can learn from examples, and importantly, learn how to deliver an algorithm that can be used by other people out of the box, with just a `pip install`, rather than a new user having to re-implement stuff, or struggle to

get someone else's code running. Researchers can commit their research to this repository, or use the PythonTemplate to generate their own project as a home for their new world-beating algorithm!

1.1.1 Features/Networks

- Liver Segmentation UNet: Based on https://doi.org/10.1007/978-3-319-24574-4_28.
- The usual FashionMNIST example, for learning purposes.

1.1.2 Design Principles

Each project herein should provide the following:

- Code that passes pylint.
- Unit testing, as appropriate. In all likelihood, testing will cover individual functions, not large training cycles.
- Sufficient logging, including date, time, software (git) version, runtime folder, machine name.
- A main class containing a network that can be run separately in train/test mode.
- Visualisation with TensorBoard.
- Saving of learned network weights at the end of training.
- Loading of pre-train weights, initialising the network ready for inference.
- The ability to be run repeatedly for hyper-parameter tuning via python scripting, not bash.
- The ability to be callable from within a Jupyter Notebook, and thereby amenable to weekly writup's for supervisions.
- One or more command line programs that are pip-installable, enabling a subsequent user to train and test your algorithm with almost-zero faff.
- Visualisation for debugging purposes, such as printing example image thumbnails etc. should be done in Jupyter notebooks, or in tensorboard, not in the same class as your algorithm.

Optional features could include:

- Small test projects that train quickly to completion won't need checkpointing, but large ones will.

1.1.3 Usage

Typical instructions for use:

First create a clean python environment, just installing tox:

```
# Create a clean conda environment
conda create -n myenv python=3.6
conda activate myenv
pip install tox
```

Then you get the code, and use tox to install all other dependencies:

```
git clone https://github.com/UCL/scikit-surgerytf
cd scikit-surgerytf
# edit requirements.txt, changing tensorflow to tensorflow-gpu.
# The default is the CPU version just for cross platform testing,
```

(continues on next page)

(continued from previous page)

```
# but for real use, you should swap it to GPU.
# Then run tox to install all dependencies.
tox
```

Then you can activate the tox created virtualenv and run top-level entry points directly from the root folder:

```
source .tox/py36/bin/activate
python sksurgeryrgbunet.py --help
```

Windows users would run:

```
.tox\py36\Scripts\activate
python sksurgeryrgbunet.py --help
```

So, for example, to run the sksurgeryrgbunet.py program and train on some data, you would do:

```
python sksurgeryrgbunet.py -d DATA -w working_dir -s output.hdf5
```

where DATA is a directory like:

```
DATA/P1/masks
DATA/P1/images
DATA/P2/masks
DATA/P2/images
.
.
DATA/PN/masks
DATA/PN/images
```

and P1,P2..PN just represents some patient identifier. Images and masks, though in different folders, must have the same name.

1.1.4 Developing

Cloning

You can clone the repository using the following command:

```
git clone https://github.com/UCL/scikit-surgerytf
```

Running tests

Pytest is used for running unit tests, but you should run using tox, as per the PythonTemplate instructions.

Linting

This code conforms to the PEP8 standard. Pylint is used to analyse the code. Again, follow the PythonTemplate instructions and run via tox.

1.1.5 Installing

You can pip install directly from the repository as follows:

```
pip install git+https://github.com/UCL/scikit-surgerytf
```

1.1.6 Contributing

Please see the contributing guidelines.

1.1.7 Useful links

- Source code repository
- Documentation

1.1.8 Licensing and copyright

1.1.9 Acknowledgements

Supported by Wellcome and EPSRC.

1.2 Segmentation

1.2.1 Liver Segmentation UNet

Module to implement a semantic (pixelwise) segmentation using UNet on 512x512.

class sksurgerytf.models.rgb_unet.**RGBUNet**(*logs='logs/fit'*, *data=None*, *working=None*, *omit=None*, *model=None*, *learning_rate=0.0001*, *epochs=50*, *batch_size=2*, *input_size=(512, 512, 3)*, *patience=20*)

Class to encapsulate RGB UNet semantic (pixelwise) segmentation network.

Thanks to Zhixuhao, and ShawDa for getting me started, and 'Harshall Lamba <https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47>_, for further inspiration.

predict(*rgb_image*)

Method to test a single image. Image resized to match network, segmented and then resized back to match the input size.

Parameters **rgb_image** – 3 channel RGB, [0-255], uchar.

Returns single channel, [0=bgl255=fg].

save_model(*filename*)

Method to save the whole trained network to disk.

Parameters **filename** – file to save to.

train()
Method to train the neural network. Writes each epoch to tensorboard log files.

Returns output of self.model.evaluate on validation set, or None.

sksurgerytf.models.rgb_unet.**run_rgb_unet_model**(*logs*, *data*, *working*, *omit*, *model*, *save*, *test*, *prediction*, *epochs*, *batch_size*, *learning_rate*, *patience*)
Helper function to run the RGBUnet model from the command line entry point.

Parameters

- **logs** – directory for log files for tensorboard.
- **data** – root directory of training data.
- **working** – working directory for organising data.
- **omit** – patient identifier to omit, when doing Leave-One-Out.
- **model** – file of previously saved model.
- **save** – file to save model to.
- **test** – input image to test.
- **prediction** – output image, the result of the prediction on test image.
- **epochs** – number of epochs.
- **batch_size** – batch size.
- **learning_rate** – learning rate for optimizer.
- **patience** – number of steps to tolerate non-improving accuracy

1.3 Fashion MNIST Example Classifer

Module to implement a basic classifier for the Fashion MNIST dataset. The aim of this module is to demonstrate how to create a class that can be developed, tested and re-used effectively. It is not a demonstration on how to do deep learning, or classification per se.

Inspired by TensorFlow tutorials.

class sksurgerytf.models.fashion.**FashionMNIST**(*logs='logs/fit'*, *model=None*, *learning_rate=0.001*, *epochs=1*)
Class to encapsulate a classifier for the Fashion MNIST dataset.

extract_failures(*number_to_fetch*)
Returns incorrectly classified test images. :param number_to_fetch: int, the number to find.

This method is slow, its only for demo purposes.

Returns indexes, images, predicted, labels

get_class_names()
Returns a copy of the valid class names. We return copies to stop external people accidentally editing the internal copies. It's safer in the long run, although in Python easy to work around.

Returns list of strings

get_test_image(*index*)
Extracts an image from the test data. Useful for unit testing, as the original data comes packaged up in a zip file.

Parameters **index** – int [0-9999], unchecked

Returns image, (28 x 28), numpy, single channel, [0-255], uchar.

save_model(*filename*)
Method to save the whole trained network to disk.

Parameters **filename** – file to save to.

test(*image*)
Method to test a single (28 x 28) image.

Parameters **image** – (28 x 28), numpy, single channel, [0-255], uchar.

Returns (class_index, class_name)

train()
Method to train the neural network. Writes each epoch to tensorboard log files.

Returns output of self.model.evaluate on test set.

sksurgerytf.models.fashion.**run_fashion_model**(*logs*, *model*, *save*, *test*)
Helper function to run the Fashion MNIST model from the command line entry point.

Parameters

- **logs** – directory for log files for tensorboard.
- **model** – file of previously saved model.
- **save** – file to save weights to
- **test** – image to test

Index