

# Open-Source Code, Research, Education and DeepReg, One or Four Things?

Yipeng Hu

[yipeng.hu@ucl.ac.uk](mailto:yipeng.hu@ucl.ac.uk)

Turing Workshop on Open-Source AI Software for Healthcare

21<sup>st</sup> November 2022, The Alan Turing Institute

## Software - Research - Education

Readability (train script)

Package or tool (network class)

Pre-defined workflow (data loader)

Manual, documentation or tutorial (docs vs papers)

Test (alternative codebase)

## Promotion

## Clinical deployment

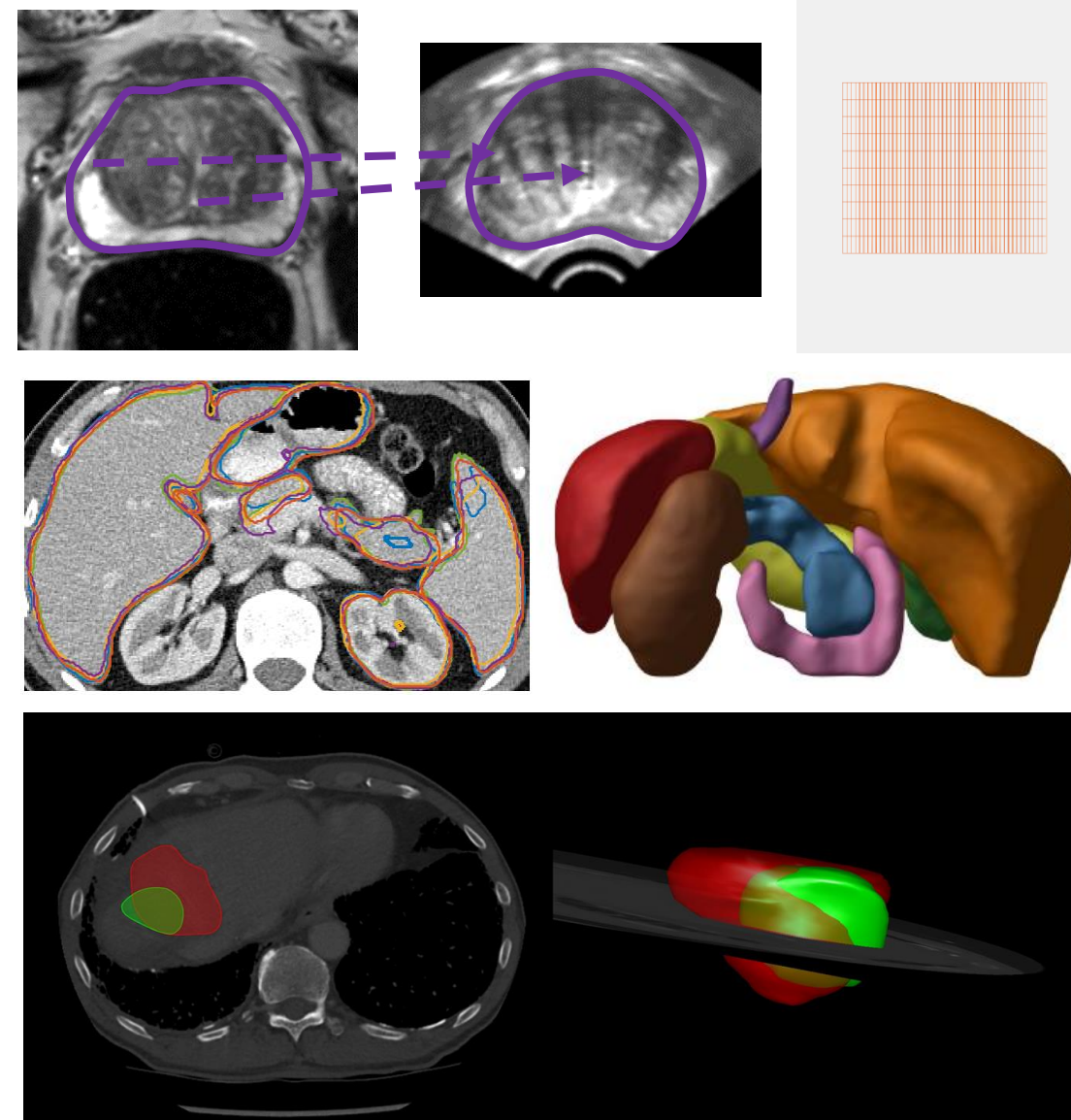
## Where are we heading

## Registration applications

- Multi-modal, e.g. image-guided interventions
- Inter-subject, e.g. atlas-based segmentation
- Medical image analysis, e.g. longitudinal analysis

## Why correspondence is a problem

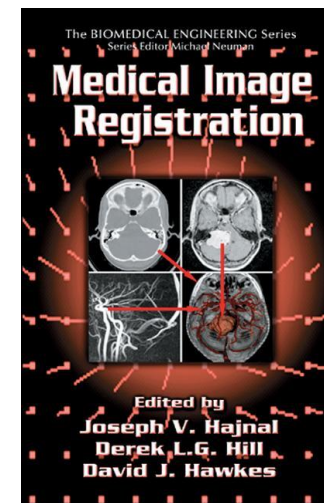
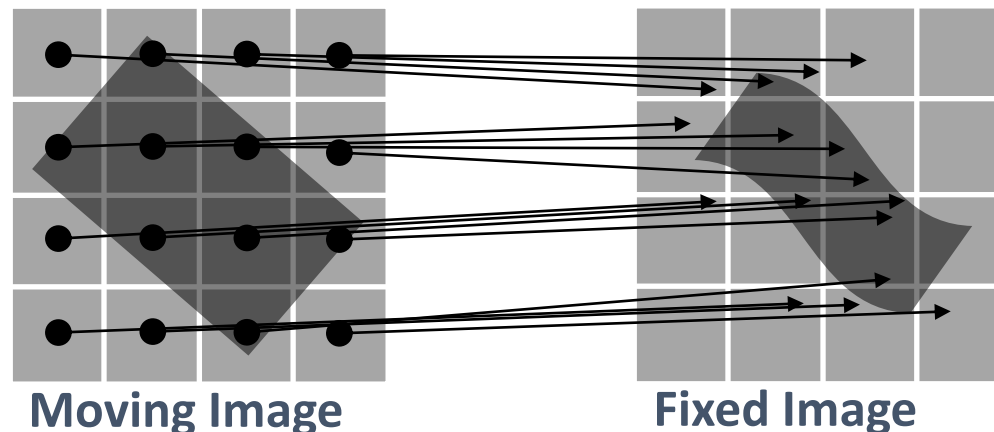
- Different imaging coordinates
- Motion and deformation
- Homology



## The three-component formulation

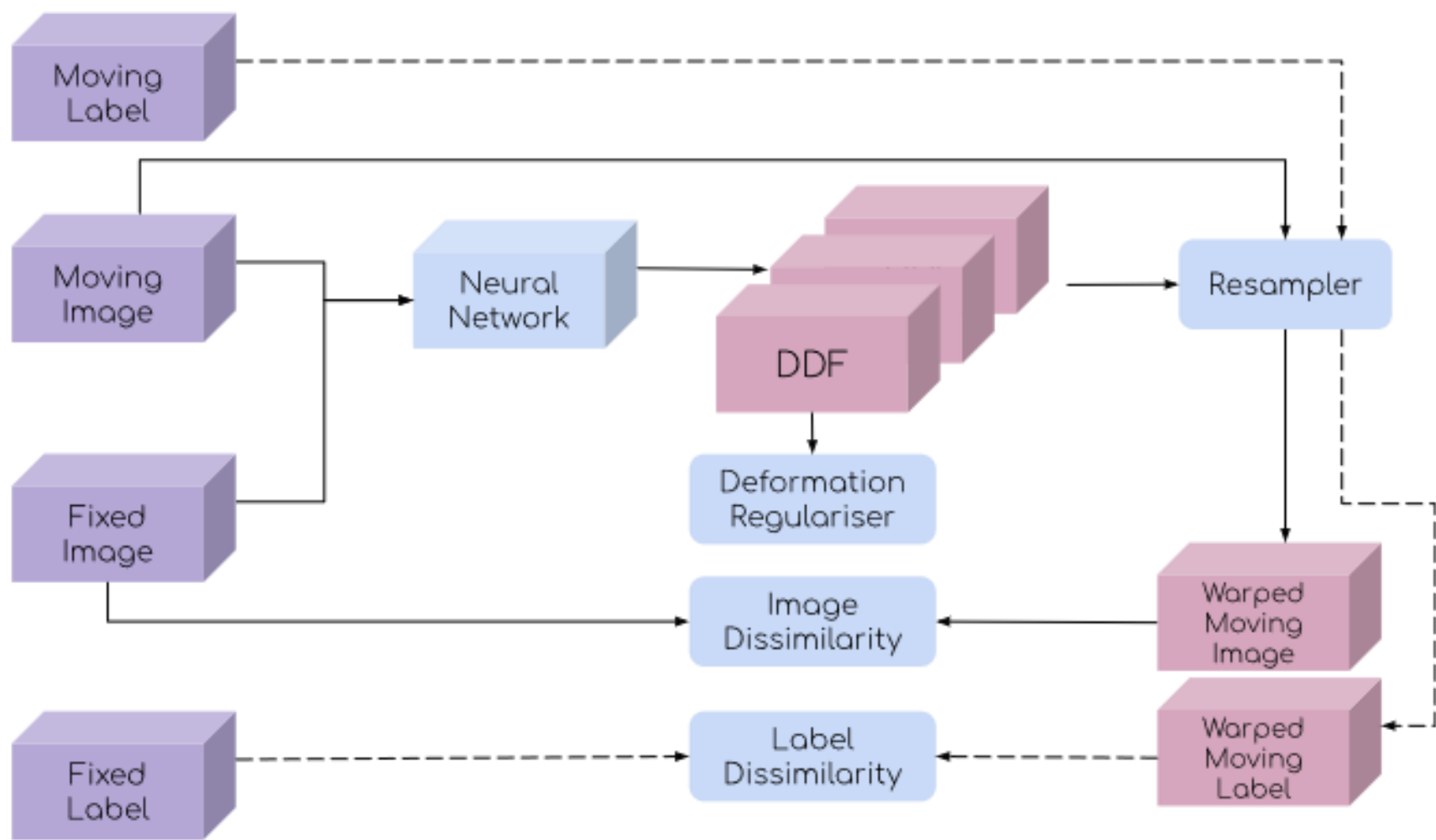
- **Regularised transformation models:**  $\mathcal{T}(I_A, \theta)$ , e.g. rigid, affine, TPS, FFD ...
  - **Image similarity measures:**  $\mathcal{S}(I_A, I_B)$ , e.g. SSD, CC, MI ...
  - **Optimisation:**  $\hat{\theta} = \max_{\theta} \mathcal{S}[I_{Fixed}, \mathcal{T}(I_{Moving}, \theta)]$
- e.g. [Rueckert et al 1999] and two-decades research

- **Intensity- and Feature-based methods**





# Medical image registration using deep learning



# Freely available, community-supported open-source tools for medical image registration using deep learning

DeepReg.net  
github.com/DeepRegNet/DeepReg  
DeepReg.readthedocs.io



DeepRegNet / DeepReg

Unwatch 11Unstar 162Fork 22

<> CodeIssues 32Pull requests 1ActionsProjectsWikiSecurityInsights

main6 branches3 tags

Go to fileAdd fileCode

mathpluscode Merge pull request #506 from DeepRegNet/505-stable-pypi-tag82d5742 15 hours ago1,797 commits

.github	Merge branch 'main' into 461-some-robots-to-improve-our-workflow	12 days ago
config	Issue #344: refactor grouped_mask_prostate_longitudinal demo	11 days ago
data/test	Issue #280: save outputs using nifti1 format	3 months ago
deepreg	Merge branch 'main' into 481-improve-error-check-when-data-path-is-wr...	yesterday
demos	Issue #489: correct unpaired_ct_abdomen test test-demo	2 days ago
docs	Issue #499: update docs regarding release v0.1.0	yesterday
test	Merge branch 'main' into 481-improve-error-check-when-data-path-is-wr...	yesterday
.dockerignore	reformat all files	4 months ago
.flake8	issue #10 fix black isort conflict, max len = 88	4 months ago
.gitignore	Issue #344: refactor classical demos	10 days ago
.isort.cfg	Issue #289: init of sphinx	3 months ago

About

Medical image registration using deep learning (beta)

image-registrationmedical-image-registrationimage-fusiondeep-learningdeep-neural-networksneural-networkconvolutional-neural-networkstensorflow2deepreg

ReadmeApache-2.0 License

Releases 3v0.1.0 (Latest)yesterday2 releases

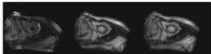
deep

Freely available, community-supported open-source tools for medical image registration using deep learning.

View on GitHubRead The Docs

deep

Unsupervised and weakly-supervised algorithms.  
High-quality software. Robust validation and documentation.  
Open-accessible data. Efficient training. Rapid deployment.



Get Started

Not sure where to start? Check out the documentation and demos.

DeepReg

DeepReg is a freely available, community-supported open-source toolkit for research and education in medical image registration using deep learning.

The current version is implemented as a TensorFlow2-based framework, and contains implementations for unsupervised, and weakly-supervised algorithms with their combinations and variants. DeepReg has a practical focus on growing and diverse clinical applications, as seen in the provided examples - DeepReg Demos.

Get involved and help make DeepReg better!

Features

DeepReg extends and simplifies workflows for medical imaging researchers working in TensorFlow 2, and can be easily installed and used for efficient training and rapid deployment of deep-learning registration algorithms.

DeepReg is designed to be used with minimal programming or scripting, owing to its built-in command line tools.

Our development and all related work involved in the project is public, and released under the Apache 2.0 license.

Contributors

DeepReg is maintained and led by a team of developers and researchers. People with significant contributions to DeepReg are listed below (in alphabetical order).

Name	Affiliation (at time of contribution)
Adria Casamitjana	University College London
Alexander Grimwood	University College London
Daniel C. Alexander	University College London
Dean C. Barratt	University College London
Esther Bonmati	University College London
Juan Eugenio Iglesias	University College London / Massachusetts Institute of Technology
Matt J. Clarkson	University College London
Nina Montalba Brown	University College London
Qianqian Yang	University College London / King's College London
Ravi Deledar	University College London
Shahar U. Saneif	University College London
Stefano B. Blumberg	University College London
Tom Vercauteren	King's College London

# Readability – the original training.py script

```

1  import tensorflow as tf
2  import sys
3  import random
4  import time
5
6  import labelreg.helpers as helper
7  import labelreg.networks as network
8  import labelreg.utils as util
9  import labelreg.losses as loss
10
11
12 # 0 - get configs
13 config = helper.ConfigParser(sys.argv, 'training')
14
15 # 1 - data
16 reader_moving_image, reader_fixed_image, reader_moving_label, reader_fixed_label = helper.get_data_readers(
17     config['Data']['dir_moving_image'],
18     config['Data']['dir_fixed_image'],
19     config['Data']['dir_moving_label'],
20     config['Data']['dir_fixed_label'])
21
22
23 # 2 - graph
24 ph_moving_image = tf.placeholder(tf.float32, [config['Train']['minibatch_size']+reader_moving_image.data_shape+[1]])
25 ph_fixed_image = tf.placeholder(tf.float32, [config['Train']['minibatch_size']+reader_fixed_image.data_shape+[1]])
26 ph_moving_affine = tf.placeholder(tf.float32, [config['Train']['minibatch_size']+[1, 12]])
27 ph_fixed_affine = tf.placeholder(tf.float32, [config['Train']['minibatch_size']+[1, 12]])
28 input_moving_image = util.warp_image_affine(ph_moving_image, ph_moving_affine) # data augmentation
29 input_fixed_image = util.warp_image_affine(ph_fixed_image, ph_fixed_affine) # data augmentation
30
31 # predicting ddf
32 reg_net = network.build_network(network_type=config['Network']['network_type'],
33                                 minibatch_size=config['Train']['minibatch_size'],
34                                 image_moving=input_moving_image,
35                                 image_fixed=input_fixed_image)

```

```

# loss
ph_moving_label = tf.placeholder(tf.float32, [config['Train']['minibatch_size']+reader_moving_image.data_shape+[1]])
ph_fixed_label = tf.placeholder(tf.float32, [config['Train']['minibatch_size']+reader_fixed_image.data_shape+[1]])
input_moving_label = util.warp_image_affine(ph_moving_label, ph_moving_affine) # data augmentation
input_fixed_label = util.warp_image_affine(ph_fixed_label, ph_fixed_affine) # data augmentation

warped_moving_label = reg_net.warp_image(input_moving_label) # warp the moving label with the predicted ddf

loss_similarity, loss_regulariser = loss.build_loss(similarity_type=config['Loss']['similarity_type'],
                                                    similarity_scales=config['Loss']['similarity_scales'],
                                                    regulariser_type=config['Loss']['regulariser_type'],
                                                    regulariser_weight=config['Loss']['regulariser_weight'],
                                                    label_moving=warped_moving_label,
                                                    label_fixed=input_fixed_label,
                                                    network_type=config['Network']['network_type'],
                                                    ddf=reg_net.ddf)

train_op = tf.train.AdamOptimizer(config['Train']['learning_rate']).minimize(loss_similarity+loss_regulariser)

# utility nodes - for information only
dice = util.compute_binary_dice(warped_moving_label, input_fixed_label)
dist = util.compute_centroid_distance(warped_moving_label, input_fixed_label)

```

```

60 # 3 - training
61 num_minibatch = int(reader_moving_label.num_data/config['Train']['minibatch_size'])
62 train_indices = [i for i in range(reader_moving_label.num_data)]
63
64 saver = tf.train.Saver(max_to_keep=1)
65 sess = tf.Session()
66 sess.run(tf.global_variables_initializer())
67 for step in range(config['Train']['total_iterations']):
68
69     if step in range(0, config['Train']['total_iterations'], num_minibatch):
70         random.shuffle(train_indices)
71
72     minibatch_idx = step % num_minibatch
73     case_indices = train_indices[
74         minibatch_idx*config['Train']['minibatch_size']:(minibatch_idx+1)*config['Train']['minibatch_size']]
75     label_indices = [random.randrange(reader_moving_label.num_labels[i]) for i in case_indices]
76
77     trainFeed = {ph_moving_image: reader_moving_image.get_data(case_indices),
78                 ph_fixed_image: reader_fixed_image.get_data(case_indices),
79                 ph_moving_label: reader_moving_label.get_data(case_indices, label_indices),
80                 ph_fixed_label: reader_fixed_label.get_data(case_indices, label_indices),
81                 ph_moving_affine: helper.random_transform_generator(config['Train']['minibatch_size']),
82                 ph_fixed_affine: helper.random_transform_generator(config['Train']['minibatch_size'])}
83
84     sess.run(train_op, feed_dict=trainFeed)

```

## Readability – the original training.py script



```
1 import tensorflow as tf
2
3
4 def build_loss(similarity_type, similarity_scales, regulariser_type, regulariser_weight,
5               label_moving, label_fixed, network_type, ddf):
6     label_similarity = multi_scale_loss(label_fixed, label_moving, similarity_type.lower(), similarity_scales)
7     if network_type.lower() == 'global':
8         ddf_regularisation = tf.constant(0.0)
9     else:
10        ddf_regularisation = tf.reduce_mean(local_displacement_energy(ddf, regulariser_type, regulariser_weight))
11    return tf.reduce_mean(label_similarity), ddf_regularisation
12
13
14 def weighted_binary_cross_entropy(ts, ps, pw=1, eps=1e-6):
15     ps = tf.clip_by_value(ps, eps, 1-eps)
16     return -tf.reduce_sum(
17         tf.concat([ts*pw, 1-ts], axis=4)*tf.log(tf.concat([ps, 1-ps], axis=4)),
18         axis=4, keep_dims=True)
19
20
21 def dice_simple(ts, ps, eps_vol=1e-6):
22     numerator = tf.reduce_sum(ts*ps, axis=[1, 2, 3, 4]) * 2
23     denominator = tf.reduce_sum(ts, axis=[1, 2, 3, 4]) + tf.reduce_sum(ps, axis=[1, 2, 3, 4])+eps_vol
24     return numerator/denominator
25
```



# Readability – the DeepReg training.py script

```

48 train(
49     gpu="0",
50     config_path=config_path,
51     gpu_allow_growth=True,
52     ckpt_path="",
53     log_dir=log_dir,
54     exp_name=exp_name,
55 )

```

```

1 import argparse
2 from datetime import datetime
3
4 from deepreg.train import train
5
6 name = "paired_mrus_prostate"
7
8
9 # parser is used to simplify testing
10 # please run the script with --full flag to ensure non-testing mode
11 # for instance:
12 # python script.py --full
13 parser = argparse.ArgumentParser()
14 parser.add_argument(
15     "--test",
16     help="Execute the script with reduced image size for test purpose.",
17     dest="test",
18     action="store_true",
19 )
20 parser.add_argument(
21     "--full",
22     help="Execute the script with full configuration.",
23     dest="test",
24     action="store_false",
25 )
26 parser.set_defaults(test=True)
27 args = parser.parse_args()
28
29
30 print(
31     "\n\n\n\n\n"
32     "=====\n"
33     "The training can also be launched using the following command.\n"
34     "deepreg_train --gpu '0' "
35     f"--config_path demos/{name}/{name}.yaml "
36     f"--log_dir demos/{name} "
37     "--exp_name logs_train\n"
38     "=====\n"
39     "\n\n\n\n\n"
40 )
41
42 log_dir = f"demos/{name}"
43 exp_name = "logs_train/" + datetime.now().strftime("%Y%m%d-%H%M%S")
44 config_path = [f"demos/{name}/{name}.yaml"]
45 if args.test:
46     config_path.append("config/test/demo_paired.yaml")

```

# Readability – the DeepReg training.py script

```
48 train(
49     gpu="0",
50     config_path=config_path,
51     gpu_allow_growth=True,
52     ckpt_path="",
53     log_dir=log_dir,
54     exp_name=exp_name,
55 )
```

```
def train():
    """
    Main training loop.
    """
    # Parse command line arguments
    parser = argparse.ArgumentParser()
    parser.add_argument('--config_path', type=str, default='config.yaml')
    parser.add_argument('--gpu', type=str, default='0')
    parser.add_argument('--gpu_allow_growth', type=bool, default=True)
    parser.add_argument('--ckpt_path', type=str, default='')
    parser.add_argument('--log_dir', type=str, default='logs')
    parser.add_argument('--exp_name', type=str, default='')
    args = parser.parse_args()

    # Set GPU device
    os.environ['CUDA_VISIBLE_DEVICES'] = args.gpu

    # Load configuration
    config = yaml.load(open(args.config_path, 'r'), Loader=yaml.FullLoader)

    # Initialize logger
    log_dir = args.log_dir
    exp_name = args.exp_name
    os.makedirs(log_dir, exist_ok=True)
    os.makedirs(os.path.join(log_dir, exp_name), exist_ok=True)
    logger = logging.getLogger(__name__)
    logger.setLevel(logging.INFO)
    handler = logging.FileHandler(os.path.join(log_dir, exp_name, 'train.log'))
    handler.setLevel(logging.INFO)
    logger.addHandler(handler)

    # Initialize dataset
    dataset = Dataset(
        train_dir=config['train_dir'],
        valid_dir=config['valid_dir'],
        format=config['format'],
        labeled=config['labeled'],
        type=config['type'],
        moving_image_shape=config['moving_image_shape'],
        fixed_image_shape=config['fixed_image_shape'],
        num_channel_initial=config['num_channel_initial'],
        extract_levels=config['extract_levels'],
        method=config['method'],
        backbone=config['backbone'],
        loss=config['loss'],
        regularizer=config['regularization'],
        optimizer=config['optimizer'],
        learning_rate=config['learning_rate'],
        preprocess=config['preprocess'],
        data_augmentation=config['data_augmentation'],
        batch_size=config['batch_size'],
        shuffle_buffer_num_batch=config['shuffle_buffer_num_batch'],
        epochs=config['epochs'],
        save_period=config['save_period'],
    )

    # Train model
    model = train_model(dataset)

    # Save model
    save_model(model, args.ckpt_path)
```

```
1 dataset:
2 train:
3   dir:
4     - demos/paired_mrur_prostate/dataset/part00
5     - demos/paired_mrur_prostate/dataset/part01
6     - demos/paired_mrur_prostate/dataset/part02
7     - demos/paired_mrur_prostate/dataset/part03
8     - demos/paired_mrur_prostate/dataset/part04
9     - demos/paired_mrur_prostate/dataset/part05
10    - demos/paired_mrur_prostate/dataset/part06
11    - demos/paired_mrur_prostate/dataset/part07
12  format: nifti
13  labeled: true
14  valid:
15    dir:
16      - demos/paired_mrur_prostate/dataset/part08
17    format: nifti
18    labeled: true
19  test:
20    dir:
21      - demos/paired_mrur_prostate/dataset/part09
22      - demos/paired_mrur_prostate/dataset/part10
23    format: nifti
24    labeled: true
25  type: paired
26  moving_image_shape: [69, 69, 64]
27  fixed_image_shape: [47, 63, 44]
28  train:
29    # define neural network structure
30    method: "ddf" # the registration method, value should be ddf / dvf / conditional
31    backbone:
32      name: "local" # value should be local / global / unet
33      num_channel_initial: 16 # number of initial channel in local net, controls the size of the network
34      extract_levels: [0, 1, 2, 3]
35
36  # define the loss function for training
37  loss:
38    image:
39      name: "ssd"
40      weight: 0
41    label:
42      weight: 1.0
43      name: "dice"
44      scales: [0, 1, 2, 4, 8, 16]
45    regularization:
46      weight: 0.125 # weight of regularization loss
47      name: "bending" # options include "bending", "gradient"
48
49  # define the optimizer
50  optimizer:
51    name: "Adam"
52    learning_rate: 1.0e-5
53
54  # define the hyper-parameters for preprocessing
55  preprocess:
56    data_augmentation:
57      name: "affine"
58    batch_size: 4
59    shuffle_buffer_num_batch: 1 # shuffle_buffer_size = batch_size * shuffle_buffer_num_batch
60
61  # other training hyper-parameters
62  epochs: 5000 # number of training epochs
63  save_period: 1000 # the model will be saved every 'save_period' epochs.
```

```
def get_weighted_avg(imgs):
    """
    Get the weighted average of the input images.
    """
    # Parse command line arguments
    parser = argparse.ArgumentParser()
    parser.add_argument('--img', type=str, default='')
    parser.add_argument('--weight', type=float, default=1.0)
    args = parser.parse_args()

    # Load images
    img = load_image(args.img)

    # Calculate weighted average
    avg = get_weighted_avg(img, args.weight)

    # Save image
    save_image(avg, args.ckpt_path)
```

## Package or tool

- > For users who do not code
- > command line tool
- > custom config file
- > REGISTRY

Feedback from developers, testers, and users

```

353
354 @REGISTRY.register_model(name="ddf")
355 class DDFModel(RegistrationModel):
356     """
357     A registration model predicts DDF.
358
359     When using global net as backbone,
360     the model predicts an affine transformation parameters,
361     and a DDF is calculated based on that.
362     """
363
364     name = "DDFModel"
365
366     def _resize_interpolate(self, field, control_points):
367         resize = layer.ResizeCPTTransform(control_points)
368         field = resize(field)
369
370         interpolate = layer.BSplines3DTransform(control_points, self.fixed_image_size)
371         field = interpolate(field)
372
373         return field
374
375     def build_model(self):
376         """Build the model to be saved as self._model."""
377         # build inputs
378         self._inputs = self.build_inputs()
379         moving_image = self._inputs["moving_image"] # (batch, m_dim1, m_dim2, m_dim3)
380         fixed_image = self._inputs["fixed_image"] # (batch, f_dim1, f_dim2, f_dim3)
381
382         # build ddf
383         control_points = self.config["backbone"].pop("control_points", False)
384         backbone_inputs = self.concat_images(moving_image, fixed_image)
385         backbone = REGISTRY.build_backbone(
386             config=self.config["backbone"],
387             default_args=dict(
388                 image_size=self.fixed_image_size,
389                 out_channels=3,
390                 out_kernel_initializer="zeros",
391                 out_activation=None,
392             ),
393         )
394

```

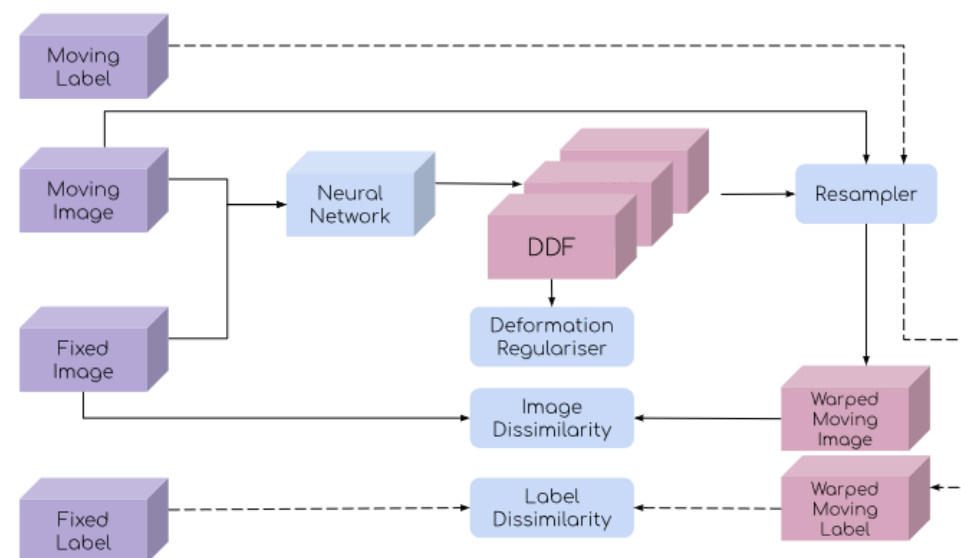
## Pre-defined workflow

Data loader script – for a two-stage, image/label feeding

```

35
36 class DataReader:
37
38     def __init__(self, dir_name):
39         self.dir_name = dir_name
40         self.files = os.listdir(dir_name)
41         self.files.sort()
42         self.num_data = len(self.files)
43
44         self.file_objects = [nib.load(os.path.join(dir_name, self.files[i])) for i in range(self.num_data)]
45         self.num_labels = [self.file_objects[i].shape[3] if len(self.file_objects[i].shape) == 4
46                             else 1
47                             for i in range(self.num_data)]
48
49         self.data_shape = list(self.file_objects[0].shape[0:3])
50
51     def get_num_labels(self, case_indices):
52         return [self.num_labels[i] for i in case_indices]
53
54     def get_data(self, case_indices=None, label_indices=None):
55         if case_indices is None:
56             case_indices = range(self.num_data)
57         # todo: check the supplied label_indices smaller than num_labels
58         if label_indices is None: # e.g. images only
59             data = [np.asarray(self.file_objects[i].dataobj) for i in case_indices]
60         else:
61             if len(label_indices) == 1:
62                 label_indices *= self.num_data
63             data = [self.file_objects[i].dataobj[..., j] if self.num_labels[i] > 1
64                     else np.asarray(self.file_objects[i].dataobj)
65                     for (i, j) in zip(case_indices, label_indices)]
66         return np.expand_dims(np.stack(data, axis=0), axis=4)
67

```



# Pre-defined workflow

DeepReg data loaders

Possibilities and permutations

Compromises

Readability

- Unpaired Images
- Grouped Images
- Classical Registration
- DOCUMENTATION**
- Command Line Tools
- Logging
- Configuration File

## Dataset Loader

- Dataset type
- Dataset requirements
- Paired images**
  - Sampling
  - Configuration
- File loader**
  - Nifti
  - H5

## Unpaired images

## Grouped images

## Registry

## Experimental Features

## API REFERENCE

## Entry Point

## Dataset Loader

## File Loader

## Registry

## Network

## Backbone

## Layer

## Loss

## Optimizer

## CONTRIBUTING TO DEEPREG

## Configuration

An example configuration for paired dataset is provided as follows.

```
dataset:
  train:
    dir: "data/test/h5/paired/train" # folder containing data
    format: "h5" # nifti or h5
    labeled: true # true or false
  valid:
    dir: "data/test/h5/unpaired/test"
    format: "h5"
    labeled: true
  test:
    dir: "data/test/h5/unpaired/test"
    format: "h5"
    labeled: true
  type: "paired" # value should be paired / unpaired / grouped
  moving_image_shape: [16, 16, 16] # value should be like [dim1, dim2, dim3]
  fixed_image_shape: [8, 8, 8] # value should be like [dim1, dim2, dim3]
```

where, the configuration can be split into common configurations that shared by all dataset types and specific configurations for paired images:

- Common configurations
  - `dir/train` gives the directory containing training data. Same for `dir/valid` and `dir/test`.
  - `format` can only be Nifti or h5 currently.
  - `type` can be paired, unpaired or grouped, corresponding to the dataset type described above.
  - `labeled` is a boolean indicating if the data is labeled or not.
- Paired images configurations
  - `moving_image_shape` is the shape of moving images, a list of three integers.
  - `fixed_image_shape` is the shape of fixed images, a list of three integers.

Optionally, multiple dataset directories can be specified, such that the data will be sampled from several directories, for instance:

```
dataset:
  train:
    dir: # folders containing data
    - "data/test/h5/paired/train1"
    - "data/test/h5/paired/train2"
```

## Nifti

Nifti data are stored in files with suffix `.nii.gz`. Each file should contain only one 3D or 4D tensor, corresponding to an image or a label.

`obs` is short for one observation of a data sample - a 3D image volume or a 3D/4D label volume - and the name can be any string.

All image data should be placed under `moving_images/`, `fixed_images/` with respect to the provided directory. The label data should be placed under `moving_labels/`, and `fixed_labels/`, if available. These are top directories.

File names should be consistent between top directories, e.g.:

- `moving_images/`
  - `obs1.nii.gz`
  - `obs2.nii.gz`
  - ...
- `fixed_images/`
  - `obs1.nii.gz`
  - `obs2.nii.gz`
  - ...
- `moving_labels/`
  - `obs1.nii.gz`
  - `obs2.nii.gz`
  - ...
- `fixed_labels/`
  - `obs1.nii.gz`
  - `obs2.nii.gz`
  - ...

Check [test paired Nifti data](#) as an example.

Optionally, the data may not be all saved directly under the top directory. They can be further grouped in subdirectories as long as the data paths are consistent.

## H5

H5 data are stored in files with suffix `.h5`. Hierarchical multi-level indexing is not used. Each file should contain multiple key-value pairs and values are



# CLT user manual

(for a standalone software)

Classical Registration
<b>DOCUMENTATION</b>
Command Line Tools
Train
Required arguments
Optional arguments
Output
Predict
Warp
Visualise
Logging
Configuration File
Dataset Loader
Registry
Experimental Features
<b>API REFERENCE</b>
Entry Point
Dataset Loader
File Loader
Registry
Network
Backbone
Layer
Loss
Optimizer
<b>CONTRIBUTING TO DEEPREG</b>
Guidelines
Unit Test
DeepReg demo
Documentation
Release
<a href="#">Read the Docs</a>
v: latest ▼

## Required arguments

### • GPU:

`--gpu` or `-g`, specifies the index or indices of GPUs for training.

Example usage:

- `--gpu ""` for CPU only
- `--gpu "0"` for using only GPU 0
- `--gpu "0,1"` for using GPU 0 and 1.

### • Configuration:

`--config_path` or `-c`, specifies the configuration file for training.

The path must end with `.yaml`.

Optionally, multiple paths can be specified, and the configuration will be merged. In case of conflicts, values are overwritten by the last config file defining them.

Example usage:

- `--config_path config1.yaml` for using one single configuration file.
- `--config_path config1.yaml config2.yaml` for using multiple configuration files.

## Optional arguments

### • CPU allocation:

`--num_workers`, if given, TensorFlow will use limited CPUs.

By default, it uses only 1 CPUs. Setting it to non-positive values will be using all CPUs.

Example usage:

- `--num_workers 2` for using at most 2 CPUs.

### • GPU memory allocation:

`--gpu_allow_growth` or `-gr`, if given, TensorFlow will only grow the memory usage as is needed.

By default, it allocates all available GPU memory.

Example usage:

- `--gpu_allow_growth`, no extra argument is needed.

### • Load checkpoint:

`--ckpt_path` or `-k`, specifies the path of the saved model checkpoint, so that the training will be resumed from the given checkpoint.

# Tutorials

(for research, education)

Cluster)

## DEEPREG DEMO

Introduction to DeepReg Demos

Paired Images

Unpaired Images

Grouped Images

Classical Registration

## DOCUMENTATION

Command Line Tools

Logging

Configuration File

Dataset Loader

Registry

Experimental Features

## API REFERENCE

Entry Point

Dataset Loader

File Loader

Registry

Network

Backbone

Layer

Loss

Optimizer

## CONTRIBUTING TO DEEPREG

Guidelines

Unit Test

DeepReg demo

Documentation

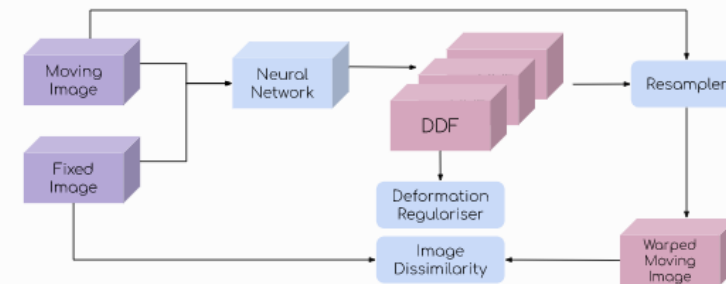
Release

## Learning

Depending on the availability of the data labels, registration networks can be trained with different approaches:

### Unsupervised

When the data label is unavailable, the training can be driven by the unsupervised loss. The loss function often consists of the intensity based loss and deformation loss. The following is an illustration of an unsupervised DDF-based registration network.



### Weakly-supervised

When there is no intensity based loss that is appropriate for the image pair one would like to register, the training can take a pair of corresponding moving and fixed labels (in addition to the image pair), represented by binary masks, to compute a label dissimilarity (feature based loss) to drive the registration.

Combined with the regularisation on the predicted displacement field, this forms a weakly-supervised training. An illustration of an weakly-supervised DDF-based registration network is provided below.

When multiple labels are available for each image, the labels can be sampled during the training iteration, such that only one label per image is used in each iteration of the data set (epoch).

# Tutorials

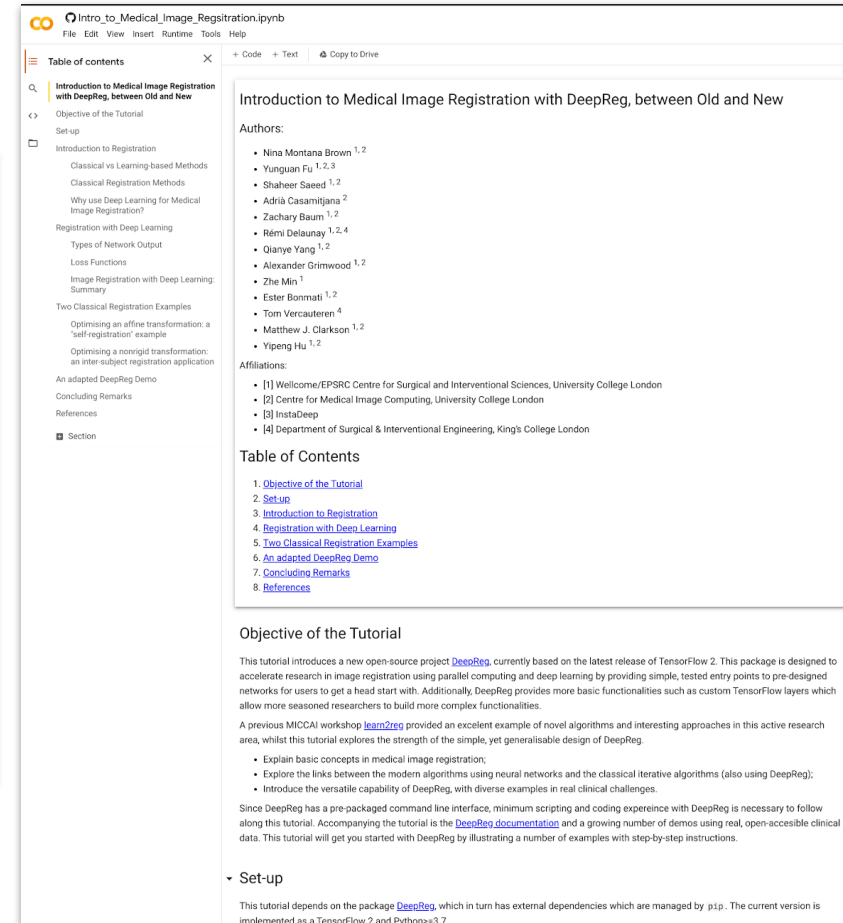
(for research, education)

1st Place Winner in MICCAI Educational Challenge 2020

## MEC 2020 Materials

### View Teaser Videos For 2020 Finalists

- 1st Place! Introduction to Medical Image Registration with DeepReg, Between Old and New (Teaser Video)**  
 Nina Montana Brown, Yunguan Fu, Shaheer Saeed, Adrià Casamitjana, Zachary Baum, Rémi Delaunay, Qianye Yang, Alexander Grimwood, Zhe Min, Ester Bonmati, Tom Vercauteren, Matthew J. Clarkson and Yipeng Hu - University College London, InstaDeep & King's College London
- 2nd Place! An Introduction to SimpleITK (Teaser Video)**  
 Ziv Yaniv, Bradley C. Lowekamp and David T. Chen - National Institute of Allergy and Infectious Diseases, National Institutes of Health & Medical Science and Computing LLC.
- 3rd Place! How to Organize a Challenge -- Behind the Scenes (Teaser Video)**  
 Annika Reinke, Matthias Eisenmann, Laura Aguilera Saiz, Sinan Onogur, Leonardo Antonio Ayala Menjivar and Peter Maximilian Full - German Cancer Research Center (DKFZ) & University of Heidelberg
- Finalist Guitar and Musical MRI Concepts (Part 2) (Part 3) (Teaser Video)**  
 Jerry J. Battista - Western University
- Finalist Hands on Machine Learning Training -- HaMLeT (Teaser Video)**  
 Leon Weninger, Laxmi Gupta and Philipp Gräbel - RWTH Aachen University
- Finalist The Basic Augmented Reality Demo -- BARD (Teaser Video)**  
 Matt Clarkson, Steve Thompson, Ester Bonmati, Tom Dowrick, Yipeng Hu and Ann Blandford - University College of London



Intro\_to\_Medical\_Image\_Registration.ipynb

File Edit View Insert Runtime Tools Help

Table of contents

Introduction to Medical Image Registration with DeepReg, between Old and New

Authors:

- Nina Montana Brown<sup>1,2</sup>
- Yunguan Fu<sup>1,2,3</sup>
- Shaheer Saeed<sup>1,2</sup>
- Adrià Casamitjana<sup>2</sup>
- Zachary Baum<sup>1,2</sup>
- Rémi Delaunay<sup>1,2,4</sup>
- Qianye Yang<sup>1,2</sup>
- Alexander Grimwood<sup>1,2</sup>
- Zhe Min<sup>1</sup>
- Ester Bonmati<sup>1,2</sup>
- Tom Vercauteren<sup>4</sup>
- Matthew J. Clarkson<sup>1,2</sup>
- Yipeng Hu<sup>1,2</sup>

Affiliations:

- [1] Wellcome/EPSRC Centre for Surgical and Interventional Sciences, University College London
- [2] Centre for Medical Image Computing, University College London
- [3] InstaDeep
- [4] Department of Surgical & Interventional Engineering, King's College London

Table of Contents

- Objective of the Tutorial
- Set-up
- Introduction to Registration
- Registration with Deep Learning
- Two Classical Registration Examples
- An adapted DeepReg Demo
- Concluding Remarks
- References

Objective of the Tutorial

This tutorial introduces a new open-source project [DeepReg](#), currently based on the latest release of TensorFlow 2. This package is designed to accelerate research in image registration using parallel computing and deep learning by providing simple, tested entry points to pre-designed networks for users to get a head start with. Additionally, DeepReg provides more basic functionalities such as custom TensorFlow layers which allow more seasoned researchers to build more complex functionalities.

A previous MICCAI workshop [learn2reg](#) provided an excellent example of novel algorithms and interesting approaches in this active research area, whilst this tutorial explores the strength of the simple, yet generalisable design of DeepReg.

- Explain basic concepts in medical image registration;
- Explore the links between the modern algorithms using neural networks and the classical iterative algorithms (also using DeepReg);
- Introduce the versatile capability of DeepReg, with diverse examples in real clinical challenges.

Since DeepReg has a pre-packaged command line interface, minimum scripting and coding experience with DeepReg is necessary to follow along this tutorial. Accompanying the tutorial is the [DeepReg documentation](#) and a growing number of demos using real, open-accessible clinical data. This tutorial will get you started with DeepReg by illustrating a number of examples with step-by-step instructions.

Set-up

This tutorial depends on the package [DeepReg](#), which in turn has external dependencies which are managed by `pip`. The current version is implemented as a TensorFlow 2 and Python>=3.7.

# Papers

(for research)



## DeepReg: a deep learning toolkit for medical image registration

**Yunguan Fu<sup>1, 2, 3</sup>, Nina Montaña Brown<sup>1, 2</sup>, Shaheer U. Saeed<sup>1, 2</sup>, Adrià Casamitjana<sup>2</sup>, Zachary M. C. Baum<sup>1, 2</sup>, Rémi Delaunay<sup>1, 4</sup>, Qianye Yang<sup>1, 2</sup>, Alexander Grimwood<sup>1, 2</sup>, Zhe Min<sup>1</sup>, Stefano B. Blumberg<sup>2</sup>, Juan Eugenio Iglesias<sup>2, 5, 6</sup>, Dean C. Barratt<sup>1, 2</sup>, Ester Bonmati<sup>1, 2</sup>, Daniel C. Alexander<sup>2</sup>, Matthew J. Clarkson<sup>1, 2</sup>, Tom Vercauteren<sup>4</sup>, and Yipeng Hu<sup>1, 2</sup>**

**1** Wellcome/EPSRC Centre for Surgical and Interventional Sciences, University College London, London, UK **2** Centre for Medical Image Computing, University College London, London, UK **3** InstaDeep, London, UK **4** Department of Surgical & Interventional Engineering, King's College London, London, UK **5** Martinos Center for Biomedical Imaging, Massachusetts General Hospital and Harvard Medical School, Boston, USA **6** Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Boston, USA

DOI: [10.21105/joss.02705](https://doi.org/10.21105/joss.02705)

### Software

- [Review](#) ↗
- [Repository](#) ↗
- [Archive](#) ↗

Editor: [Kevin M. Moerman](#) ↗

## Summary

Image fusion is a fundamental task in medical image analysis and computer-assisted intervention. Medical image registration, computational algorithms that align different images together (Hill, Batchelor, Holden, & Hawkes, 2001), has in recent years turned the research attention towards deep learning. Indeed, the representation ability to learn from population

# Docstring, API documentation

[Paired Images](#)  
[Unpaired Images](#)  
[Grouped Images](#)  
[Classical Registration](#)  
[DOCUMENTATION](#)  
[Command Line Tools](#)  
[Logging](#)  
[Configuration File](#)  
[Dataset Loader](#)  
[Registry](#)  
[Experimental Features](#)

## API REFERENCE

### Entry Point

[Train](#)  
[Predict](#)  
[Warp](#)

[Dataset Loader](#)  
[File Loader](#)  
[Registry](#)  
[Network](#)  
[Backbone](#)  
[Layer](#)  
[Loss](#)  
[Optimizer](#)

## CONTRIBUTING TO DEEPREG

[Guidelines](#)  
[Unit Test](#)  
[DeepReg demo](#)  
[Documentation](#)  
[Release](#)

[Read the Docs](#)

v: latest

[» Entry Point](#)

[Edit on GitHub](#)

## Entry Point

### Train

Module to train a network using init files and a CLI.

```
deepreg.train.build_config(config_path: Union[str, List[str]], log_dir: str,
exp_name: str, ckpt_path: str, max_epochs: int = - 1) → Tuple[Dict, str, str]
```

Function to initialise log directories, assert that checkpointed model is the right type and to parse the configuration for training.

- Parameters:
- `config_path` – list of str, path to config file
  - `log_dir` – path of the log directory
  - `exp_name` – name of the experiment
  - `ckpt_path` – path where model is stored.
  - `max_epochs` – if `max_epochs > 0`, use it to overwrite the configuration

- Returns:
- `config`: a dictionary saving configuration
  - `exp_name`: the path of directory to save logs

```
deepreg.train.main(args=None)
```

Entry point for train script.

- Parameters:
- `args` – arguments

```
deepreg.train.train(gpu: str, config_path: Union[str, List[str]], ckpt_path: str,
num_workers: int = 1, gpu_allow_growth: bool = True, exp_name: str = "", log_dir: str =
'logs', max_epochs: int = - 1)
```

Function to train a model.

- Parameters:
- `gpu` – which local gpu to use to train.
  - `config_path` – path to configuration set up.
  - `ckpt_path` – where to store training checkpoints.
  - `num_workers` – number of cpu cores to be used,



# Docker

DeepReg is a freely available, community-supported open-source toolkit for research and education in medical image registration using deep learning.

**License:** Apache Software License (apache-2.0)

deep

Papers JOSS 10.21105/joss.02705 DOI 10.5281/zenodo.4281095

DeepReg is a freely available, community-supported open-source toolkit for research and education in medical image registration using deep learning.

# DeepReg Demos

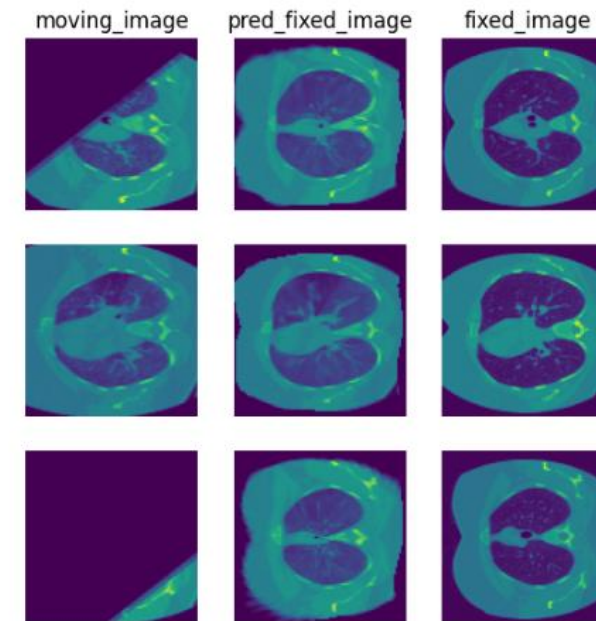
Reference
Paired brain MR-ultrasound registration
Paired prostate MR-ultrasound registration
Unpaired Images
Grouped Images
Classical Registration
DOCUMENTATION
Command Line Tools
Logging
Configuration File
Dataset Loader
Registry
Experimental Features
API REFERENCE
Entry Point
Dataset Loader
File Loader
Registry
Network
Backbone
Layer
Loss
Optimizer
CONTRIBUTING TO DEEPREG
Guidelines
Unit Test
DeepReg demo
Documentation
Release

## Visualise

The following command can be executed to generate a plot of three image slices from the the moving image, warped image and fixed image (left to right) to visualise the registration. Please see the visualisation tool docs [here](#) for more visualisation options such as animated gifs.

```
deepreg_vis -m 2 -i 'demos/paired_ct_lung/logs_predict/<time-stamp>/test/<pair-1
```

Note: The prediction must be run before running the command to generate the visualisation. The `<time-stamp>` and `<pair-number>` must be entered by the user.



## Contact

README.md

deepg

Package	License Apache 2.0 pypi v0.1.2 python 3 Downloads 10409
Documentation	docs failing
Code	Unit Test passing Integration Test failing codecov 100% code style black Scrutinizer 9.03 maintainability B
Papers	JOSS 10.21105/joss.02705 DOI 10.5281/zenodo.4281095

## DeepReg

DeepReg is a freely available, community-supported open-source toolkit for research and education in medical image registration using deep learning.


- TensorFlow 2-based for efficient training and rapid deployment;
- Implementing major unsupervised and weakly-supervised algorithms, with their combinations and variants;
- Focusing on growing and diverse clinical applications, with all DeepReg Demos using open-accessible data;
- Simple built-in command line tools requiring minimal programming and scripting;
- Open, permissible and research-and-education-driven, under the Apache 2.0 license.

---

## Getting Started

- [DeepReg.Net](#)
- [Documentation, Tutorials, and Quick Start](#)
- [Medical Image Registration Demos using DeepReg](#)
- [Issue Tracker](#)

## Contributing



dyhan316 added some stuff for change (#491)

92af0bd · 5 days ago · 540 commits

github/ISSUE_TEMPLATE	Update issue templates	3 years ago
data	update readme with https	13 months ago
scripts	added some stuff for change (#491)	5 days ago
voxmorph	Add option for unet likelihood	13 days ago
.gitignore	fixing git commit	4 years ago
LICENSE.md	switching to apache 2	2 years ago
README.md	Complete SynthMorph reference. (#401)	9 months ago
citations.bib	Complete SynthMorph reference. (#401)	9 months ago
setup.py	make sure packaging is required during setup	6 months ago

☰

README.md

## voxmorph: Learning-Based Image Registration

voxmorph is a general purpose library for learning-based tools for alignment/registration, and more generally modelling with deformations.

## Tutorial

Visit the [VoxelMorph tutorial](#) to learn about VoxelMorph and Learning-based Registration. Here's an [additional small tutorial](#) on warping annotations together with images, and another on [template \(atlas\) construction](#) with VoxelMorph.

## Instructions

To use the VoxelMorph library, either clone this repository and install the requirements listed in `setup.py` or install directly with pip.

Unsupervised Learning for Image Registration

machine-learning

deep-learning

diffeomorphism

optical-flow

unsupervised-learning

probabilistic

image-registration

image-alignment

📖 Readme

📄 Apache-2.0 license

🔖 Cite this repository ▾

★ 1.7k stars

👁 44 watching

🍴 499 forks

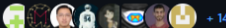
Releases

🏷 1 tags

Packages


No packages published

Used by 22



+ 14

Contributors 10



*We have no clue...!*





Trained models

Open data sets, e.g. learn2reg challenges

Clinical data pre-processing and curation

Learning Registration Components

UCL

Learning Transformation Models

Learning generative models from motion simulations

Hu, Y., Gibson, E., Vercauteren, T., Ahmed, H.U., Emberton, M., Moore, C.M., Noble, J.A. and Barratt, D.C., 2017, September. Intraoperative organ motion models with an ensemble of conditional generative adversarial networks. In *MICCAI 2017*.

Approximating finite-element analysis

Tonetti, M., Gao, G. and Yang, G.Z., 2017. A machine learning approach for real-time modelling of tissue deformation in image-guided neurosurgery. In *Artificial Intelligence in Medicine*.

Learning Registration Components

UCL

Learning Similarity Measures

Learning unsupervised features using a two-layer CNN

Wu, G., Kim, M., Wang, Q., Gao, Y., Liao, S. and Shen, D., 2013, September. Unsupervised deep feature learning for deformable registration of MR brain images. In *MICCAI 2013*.

Learning patch similarity measure from aligned images

Cheng, X., Zhang, L. and Zheng, Y., 2018. Deep similarity learning for multimodal medical images. In *Computer Methods in Biomechanics and Biomedical Engineering*.

Simonovsky, M., Gutiérrez-Becker, B., Mareus, D., Novak, H. and Komodakis, N., 2016. A deep metric for multimodal registration. In *MICCAI 2016*.

Learning Registration Components

UCL

Learning Optimisation

Learning from the Registered Images

Yang, X., Kwitt, R., Styner, M. and Niethammer, M., 2017. Quicksilver: Fast predictive image registration—a deep learning approach. In *NeuroImage*.

Learning From the Simulated (spatially-warped) Images

Miao, S., Wang, Z.L. and Liao, R., 2016. A CNN regression approach for real-time 2D/3D registration. In *IEEE transactions on medical imaging*.

The Roles of Generative Modelling

UCL

GANs and Autoencoders

Hu, Y., Gibson, E., Ghavami, N., Bonmati, E., Moore, C.M., Emberton, M., Vercauteren, T., Noble, J.A. and Barratt, D.C., 2018. Adversarial Deformation Regularization for Training Image Registration Neural Networks. In *MICCAI 2018*.

Yan, P., Xu, S., Rastinehad, A.R. and Wood, B.J., 2018. Adversarial Image Registration with Application for MR and TRUS Image Fusion. In *MICCAI 2018*.

Fan, J., Cao, X., Xie, Z., Yap, P.T. and Shen, D., 2018. Adversarial Similarity Network for Evaluating Image Alignment in Deep Learning Based Registration. In *MICCAI 2018*.

Krebs, J., Mani, T., Mallik, B., Ayache, N. and Delingette, H., 2018. Unsupervised Probabilistic Deformation Modeling for Robust Diffeomorphic Registration. In *OLMA 2018*.

Using Reinforcement Learning

UCL

Q-Learning

$$loss = \left( \frac{r + \gamma \max_{a'} Q(s, a') - Q(s, a)}{\gamma} \right)^2$$

Rigid

Liao, R., Miao, S., de Tournemine, P., Girib, S., Kamen, A., Manol, T. and Comaniciu, D., 2017, February. An Artificial Agent for Robust Image Registration. In *AAAI 2017*.

Non-Rigid via SSM

Krebs, J., Mani, T., Delingette, H., Zhang, L., Ghesu, F.C., Miao, S., Males, A.K., Ayache, N., Liao, R. and Kamen, A., 2017, September. Robust non-rigid registration through agent-based action learning. In *MICCAI 2017*.

### Reinforcement Learning for Rigid Registration

- The MDP tuple:
- State: current transformation, transformed image and the difference
  - Action: "adjusting" discrete transformation
  - Reward: difference to ground-truth transformation
  - Learning Algorithm: Q-agent with supervised registration path

Plenty research, no clinical application yet



As is and achieved?

Next interested PhD students?

MONAI?

Paper-based? e.g.

Yang 2019 MICCAI (DeepReg branch)

Yang 2021 ISBI (using DeepReg)

Yang 2022 TMI (PyTorch)

Baum 2021 MedIA (standalone repo)

Baum 2022 ASMUS (using DeepReg)

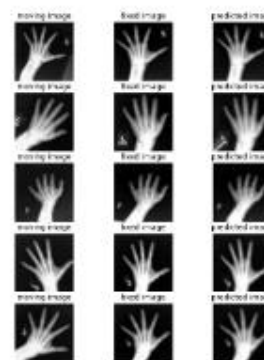
Baum 2022 TMI (using DeepReg)

Shen 2022 MICCAI (PyTorch)

UCL Modules? e.g.

MPHY0041 (DeepReg code)

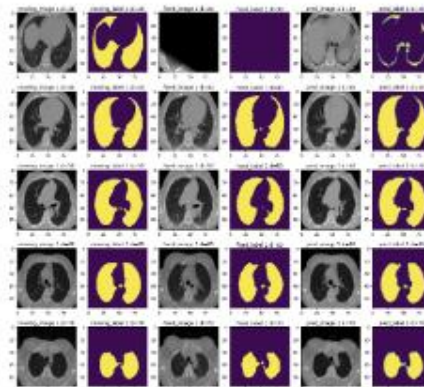
MPHY0043 (MONAI code)



## 3D intra-subject lung CT registration

The 3D intra-subject lung CT registration tutorial is an example of registration between 3D lung CT images acquired at inspiration and expiration from a single patient. This type of intra-subject registration is helpful in tracking anatomical features of interest like airways in airflow analysis or compensating motion during radiotherapy.

The tutorial showcases several features described above, including unsupervised and weakly-supervised losses, deformation regulariser, non-rigid transformation based on BDFs, and 3D volumetric registration using eval clinical images. To learn more, check out the provided notebook and the original [DeepReg demo](#) using the same open-accessible data set.



MONAI Medical Open Network for AI

Follow

Apr 23, 2021 · 4 min read · Listen



## Monai Explores Learning-Based Medical Image Registration

[MONAI](#) has been working closely with [DeepReg](#) on learning-based medical image registration using PyTorch. In the latest release, [MONAI v0.5.0](#), we are delighted to provide a set of essential tools for developing registration pipelines.

### Medical Image Registration

[Image registration](#), a process that spatially aligns one image with another, is important to many medical imaging applications. The two images that register are often referred to as moving and fixed images. An image registration algorithm generates a spatial transformation that can transform, or “warp”, the moving image onto the fixed image coordinates.

The ability to align two or more images together allows the complementary information acquired from different imaging modalities (multimodal registration, e.g., MR and ultrasound), at different time points (longitudinal registration), or from different patients (inter-subject registration). For example, track growth from a

