fastMONAI

Simplifying deep learning for medical image analysis

Western Norway University of Applied Sciences

MMIV

HELSE VEST

Mohn Medical Imaging and Visualization Centre

MMIV

HELSE BERGEN

TROND MOHN STIFTELSE

UNIVERSITAS BERGENSIS

CMMIV

HELSE BERGEN

TROND MOHN STIFTELSE

UNIVERSITAS BERGENSIS

GMMIV HELSE BERGEN TROND MOHN STIFTELSE UNIVERSITAS BERGENSIS

MONAI

TorchIO

Western Norway University of Applied Sciences

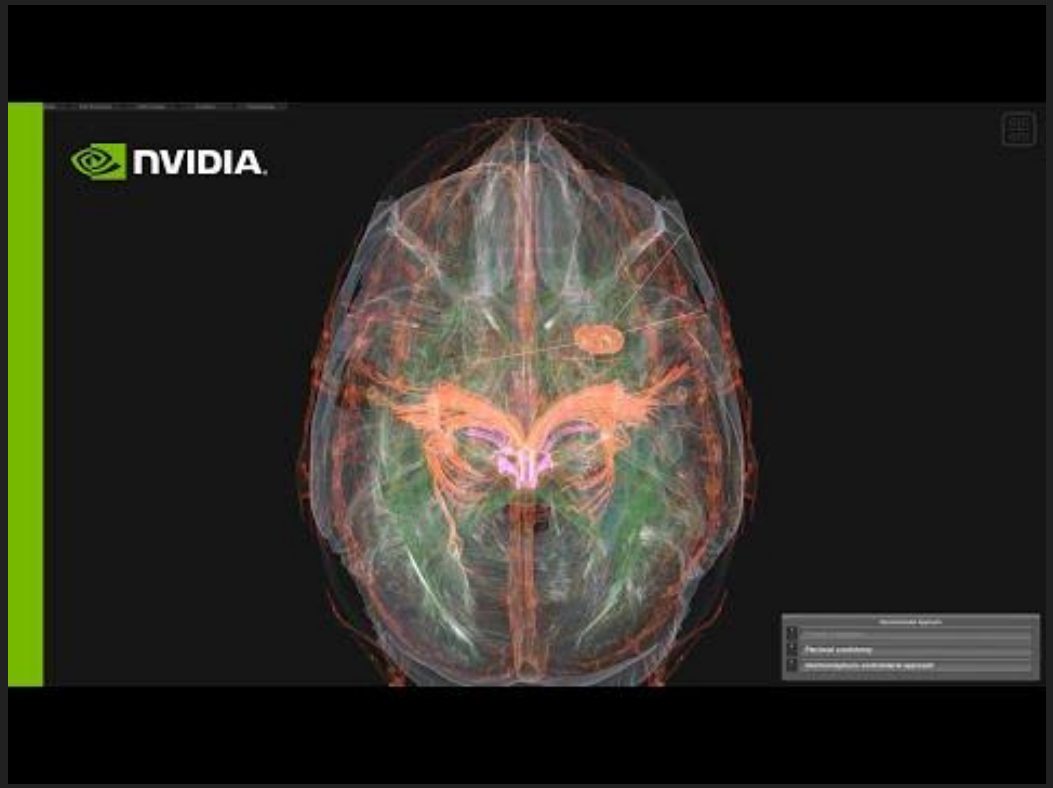
MMIV

HELSE VEST

fastai + MONAI = fastMONAI

- **MONAI**
  - Pytorch-based framework for deep learning in 2D and 3D medical imaging
- **TorchIO**
  - Python library for efficient loading, preprocessing and augmentation of 3D medical images
- **fastai**
  - Pytorch-based low-code library for 2D images utilizing state-of-the-art approaches in various domains
- **fastMONAI**
  - Utilizing multiple state-of-the-art techniques in the area of 3D medical image analysis

Why MONAI?



Announced MONAI Core v1.0 few days ago

Why an extension of MONAI?

Setup environment

```
In [ ]: !python -c "import monai" || pip install -q "monai[nibabel, tqdm]"
!python -c "import matplotlib" || pip install -q matplotlib
%matplotlib inline
```

Setup imports

```
In [1]: # Copyright 2020 MONAI Consortium
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#     http://www.apache.org/licenses/LICENSE-2.0
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

import os
import shutil
import tempfile

import matplotlib.pyplot as plt
import numpy as np
from monai.apps import DecathlonDataset
from monai.config import print_config
from monai.data import DataLoader
from monai.losses import DiceLoss
from monai.metrics import DiceMetric
from monai.networks.nets import UNet
from monai.transforms import (
    Activations,
    AsChannelFirstd,
    AsDiscrete,
    CenterSpatialCropd,
    Compose,
    LoadImaged,
    MapTransform,
    NormalizeIntensityd,
    Orientationd,
    RandFlipd,
    RandScaleIntensityd,
    RandShiftIntensityd,
    RandSpatialCropd,
    Spacingd,
    ToTensord,
)
from monai.utils import set_determinism

import torch

print_config()
```

```
MONAI version: 0.4.0
Numpy version: 1.19.1
Pytorch version: 1.7.0a0+7036e91
MONAI flags: HAS_EXT = False, USE_COMPILED = False
MONAI rev id: 0563a44671a602fcca92d91c7f47261868d171a1

Optional dependencies:
Pytorch Ignite version: 0.4.2
Nibabel version: 3.2.1
scikit-image version: 0.15.0
Pillow version: 8.0.1
Tensorboard version: 1.15.0+nv
gdown version: 3.12.2
TorchVision version: 0.8.0a0
ITK version: 5.1.2
tqdm version: 4.54.1
lmdb version: 1.0.0
psutil version: 5.7.2

For details about installing the optional dependencies, please visit:
    https://docs.monai.io/en/latest/installation.html#installing-the-recommended-dependencies
```

Setup data directory

You can specify a directory with the `MONAI_DATA_DIRECTORY` environment variable.
This allows you to save results and reuse downloads.
If not specified a temporary directory will be used.

```
In [3]: directory = os.environ.get("MONAI_DATA_DIRECTORY")
root_dir = tempfile.mkdtemp() if directory is None else directory
print(root_dir)
```

```
/workspace/data/medical
```

Set deterministic training for reproducibility

```
In [ ]: set_determinism(seed=0)
```

Define a new transform to convert brain tumor labels

Here we convert the multi-classes labels into multi-labels segmentation task in One-Hot format.

```
In [ ]: class ConvertToMultiChannelBasedOnBratsClassesd(MapTransform):
    """
    Convert labels to multi channels based on brats classes:
    label 1 is the peritumoral edema
    label 2 is the GD-enhancing tumor
    label 3 is the necrotic and non-enhancing tumor core
    The possible classes are TC (Tumor core), WT (Whole tumor)
    and ET (Enhancing tumor).

    """

    def __call__(self, data):
        d = dict(data)
        for key in self.keys:
            result = []
            # merge label 2 and label 3 to construct TC
            result.append(np.logical_or(d[key] == 2, d[key] == 3))
            # merge labels 1, 2 and 3 to construct WT
            result.append(
                np.logical_or(
                    np.logical_or(d[key] == 2, d[key] == 3), d[key] == 1
                )
            )
            # label 2 is ET
            result.append(d[key] == 2)
            d[key] = np.stack(result, axis=0).astype(np.float32)
        return d
```

Setup transforms for training and validation

```
In [ ]: train_transform = Compose(
    [
        # load 4 Nifti images and stack them together
        LoadImaged(keys=["image", "label"]),
        AsChannelFirstd(keys="image"),
        ConvertToMultiChannelBasedOnBratsClassesd(keys="label"),
        Spacingd(
            keys=["image", "label"],
            pixdim=(1.5, 1.5, 2.0),
            mode=("bilinear", "nearest"),
        ),
        Orientationd(keys=["image", "label"], axcodes="RAS"),
        RandSpatialCropd(
            keys=["image", "label"], roi_size=[128, 128, 64], random_size=False
        ),
        RandFlipd(keys=["image", "label"], prob=0.5, spatial_axis=0),
        NormalizeIntensityd(keys="image", nonzero=True, channel_wise=True),
        RandScaleIntensityd(keys="image", factors=0.1, prob=0.5),
        RandShiftIntensityd(keys="image", offsets=0.1, prob=0.5),
        ToTensord(keys=["image", "label"]),
    ]
)
val_transform = Compose(
    [
        LoadImaged(keys=["image", "label"]),
        AsChannelFirstd(keys="image"),
        ConvertToMultiChannelBasedOnBratsClassesd(keys="label"),
        Spacingd(
            keys=["image", "label"],
            pixdim=(1.5, 1.5, 2.0),
            mode=("bilinear", "nearest"),
        ),
        Orientationd(keys=["image", "label"], axcodes="RAS"),
        CenterSpatialCropd(keys=["image", "label"], roi_size=[128, 128, 64]),
        NormalizeIntensityd(keys="image", nonzero=True, channel_wise=True),
        ToTensord(keys=["image", "label"]),
    ]
)
```

Quickly load data with DecathlonDataset

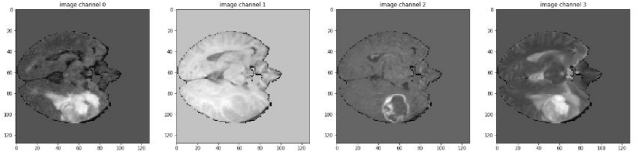
Here we use `DecathlonDataset` to automatically download and extract the dataset. It inherits MONAI `CacheDataset`, so we set `cache_num=100` to cache 100 items for training and use the defaut args to cache all the items for validation.

```
train_ds = DecathlonDataset(
    root_dir=root_dir,
    task="Task01_BrainTumour",
    transform=train_transform,
    section="training",
    download=True,
    num_workers=4,
    cache_num=100,
)
train_loader = DataLoader(train_ds, batch_size=2, shuffle=True, num_workers=4)
val_ds = DecathlonDataset(
    root_dir=root_dir,
    task="Task01_BrainTumour",
    transform=val_transform,
    section="validation",
    download=False,
    num_workers=4,
)
val_loader = DataLoader(val_ds, batch_size=2, shuffle=False, num_workers=4)
```

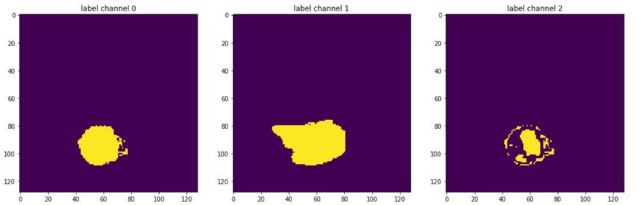
Check data shape and visualize

```
# pick one image from DecathlonDataset to visualize and check the 4 channels
print(f"image shape: {val_ds[2]['image'].shape}")
plt.figure("image", (24, 6))
for i in range(4):
    plt.subplot(1, 4, i + 1)
    plt.title(f"image channel {i}")
    plt.imshow(val_ds[2]["image"][i, :, :, 20].detach().cpu(), cmap="gray")
plt.show()
# also visualize the 3 channels label corresponding to this image
print(f"label shape: {val_ds[2]['label'].shape}")
plt.figure("label", (18, 6))
for i in range(3):
    plt.subplot(1, 3, i + 1)
    plt.title(f"label channel {i}")
    plt.imshow(val_ds[2]["label"][i, :, :, 20].detach().cpu())
plt.show()
```

```
image shape: torch.Size([4, 128, 128, 64])
```




```
label shape: torch.Size([3, 128, 128, 64])
```



```
from fastMONAI.vision_core import *
from fastMONAI.vision_data import *
from fastMONAI.vision_augmentation import *
from fastMONAI.vision_metrics import *
from fastai.vision.all import *
```

```
NB_DIR = Path.cwd()
STUDY_DIR = NB_DIR/'..'/'..'/'data'/'EC-data-Sat'
```

```
df = pd.read_csv(STUDY_DIR/'dataset.csv')
df.shape
```

```
(166, 6)
```

```
dataset_obj = MedDataset(img_list=df.mask_path.tolist(), dtype=NiiMask ,max_workers=12)
```

```
The volumes in this dataset have different orientations. Recommended to pass in the argument reorder=True when creating a MedDataset object for this dataset
```

```
data_info_df = dataset_obj.summary()
data_info_df.head()
```

nnU-Net: Self-adapting Framework

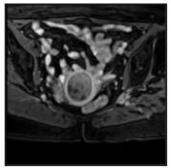
	dim_0	dim_1	dim_2	voxel_0	voxel_1	voxel_2	orientation	example_path	total
7	232	256	88	0.9766	0.9766	1.2000	LPS+	/home/sathiesh/machine_learning/fastMONAI/preprocess_nbs/../../data/EC-data-Sat/367/segmented/367segmentedJulie.nii.gz	28
4	192	192	48	1.3021	1.3021	2.0000	RAS+	/home/sathiesh/machine_learning/fastMONAI/preprocess_nbs/../../data/EC-data-Sat/011/segmented/011segmentedJulie.nii.gz	26
6	232	256	88	0.9766	0.9766	1.2000	LIP+	/home/sathiesh/machine_learning/fastMONAI/preprocess_nbs/../../data/EC-data-Sat/372/segmented/372segmentedJulie.nii.gz	25
3	192	192	48	1.3021	1.3021	2.0000	LPS+	/home/sathiesh/machine_learning/fastMONAI/preprocess_nbs/../../data/EC-data-Sat/017/segmented/017segmentedJulie.nii.gz	18
0	192	48	192	1.3021	2.0000	1.3021	RAS+	/home/sathiesh/machine_learning/fastMONAI/preprocess_nbs/../../data/EC-data-Sat/021/segmented/021segmentedJulie.nii.gz	17

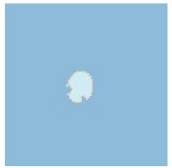
```
_, reorder = dataset_obj.suggestion()
resample = [1,1,1]
```

```
dblock = NiiDataBlock(blocks=(ImageBlock(cls=NiiImage), NiiMaskBlock), splitter=RandomSplitter(seed=42),get_x=ColReader('img_path'),get_y=ColReader('mask_path'),
                      item_tfms=[ZNormalizationTIO(), ImagePadOrCropTIO([256,256,256])] ,reorder=reorder,resample=resample)
```

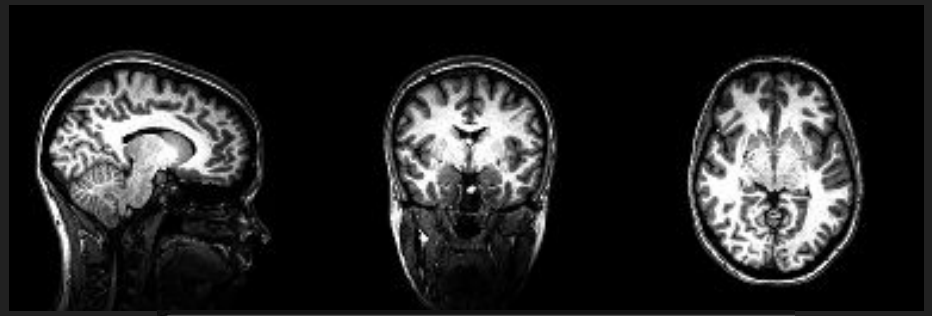
```
dls = dblock.dataloaders(df, bs=4)
```

```
dls.show_batch(figsize=(10, 10), anatomical_plane='A')
```





Deep learning for 3D images?



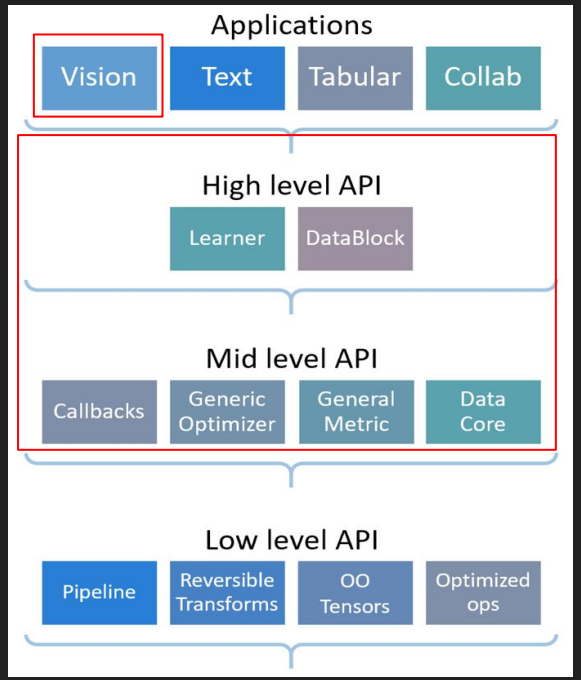
```
img.shape
```

```
torch.Size([1, 194, 256, 256])
```

Not supported in fastai

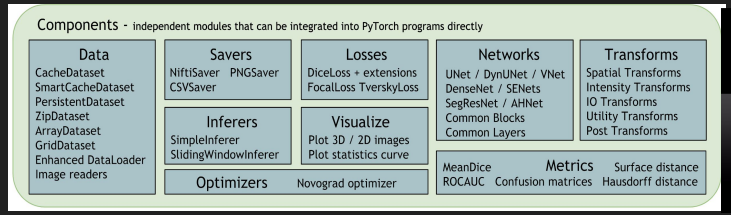
fastMONAI

fastai v2

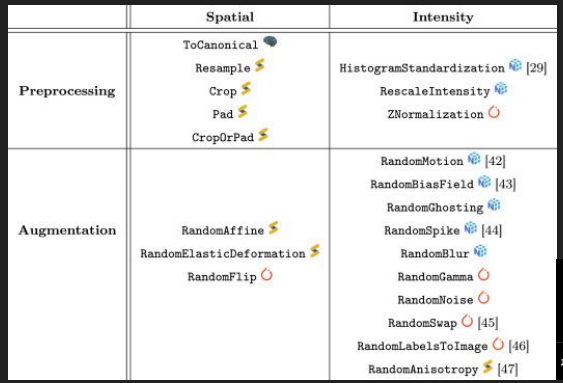


+

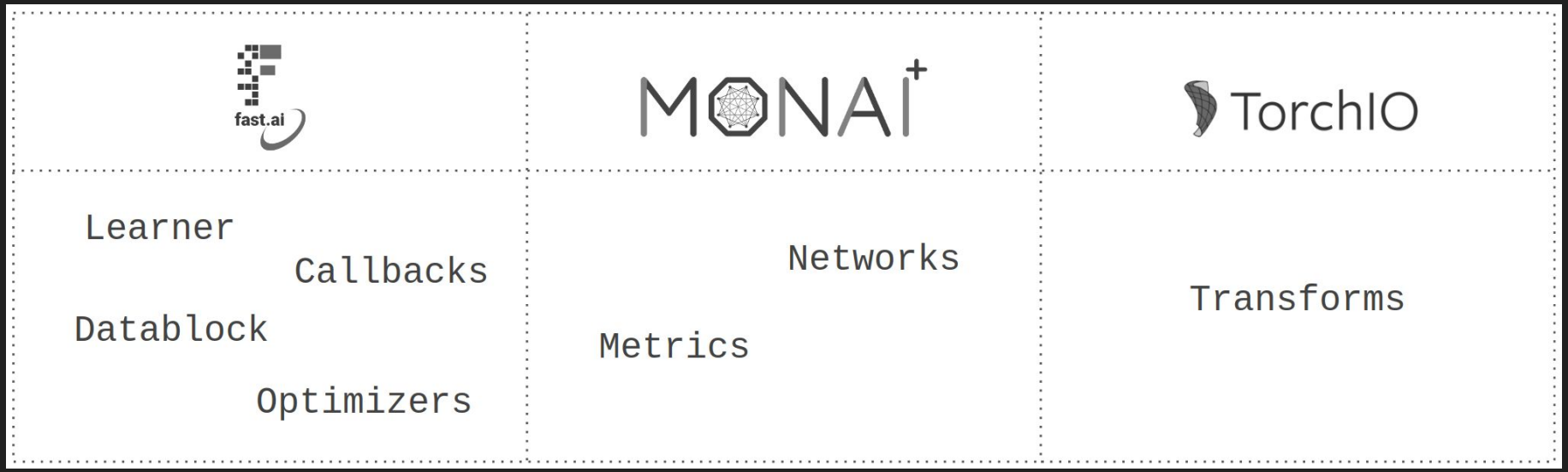
MONAI



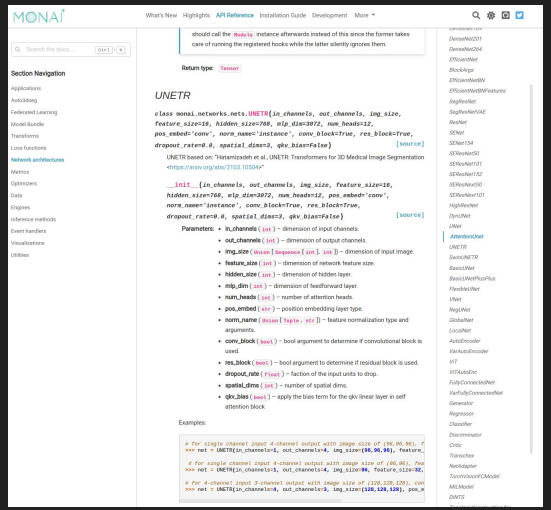
TorchIO

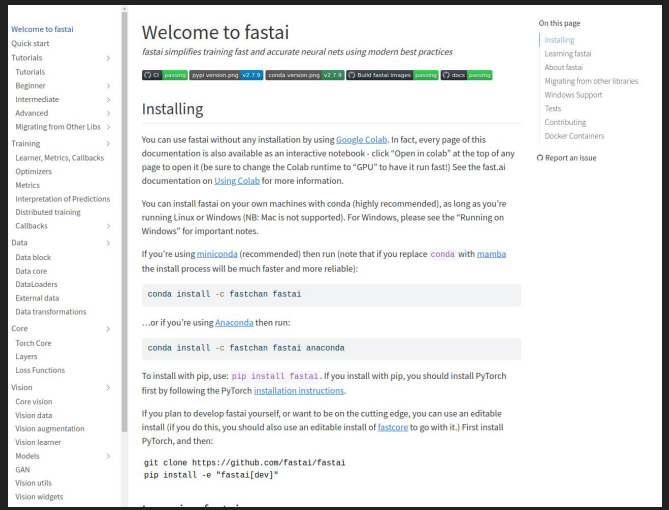


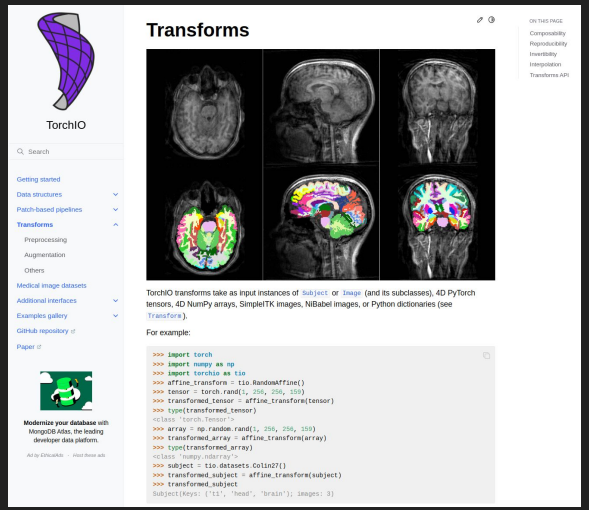
fastMONAI


fast.ai
MONAI+
TorchIO
Learner
Callbacks
Datablock
Optimizers
Networks
Metrics
Transforms

Well-documented

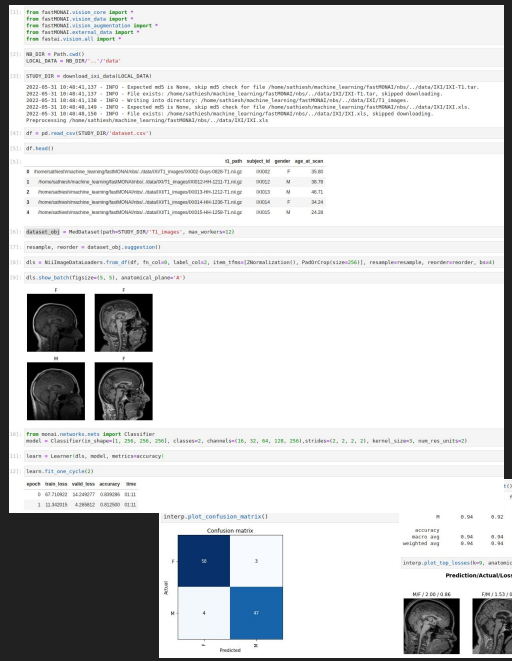




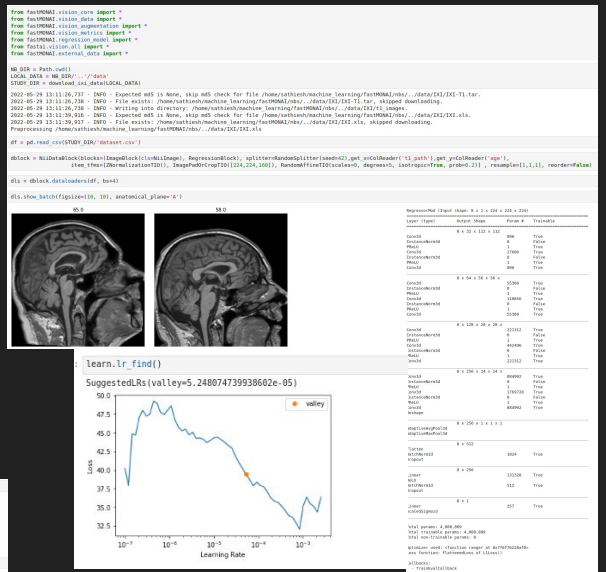


fastMONAI has step-by-step tutorials using open datasets

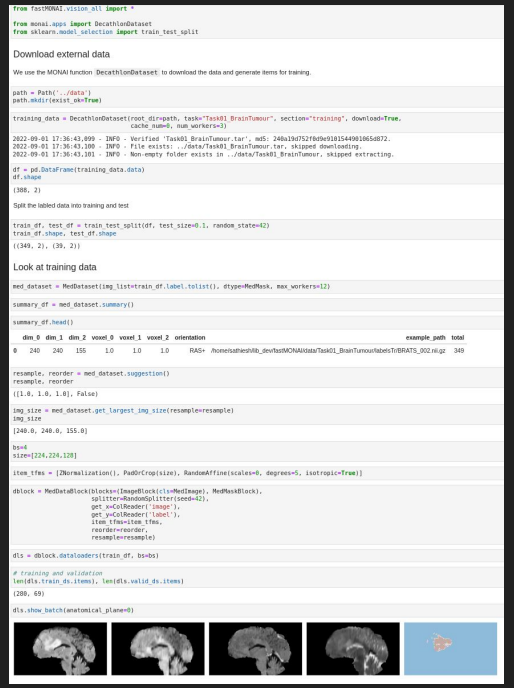
Classification



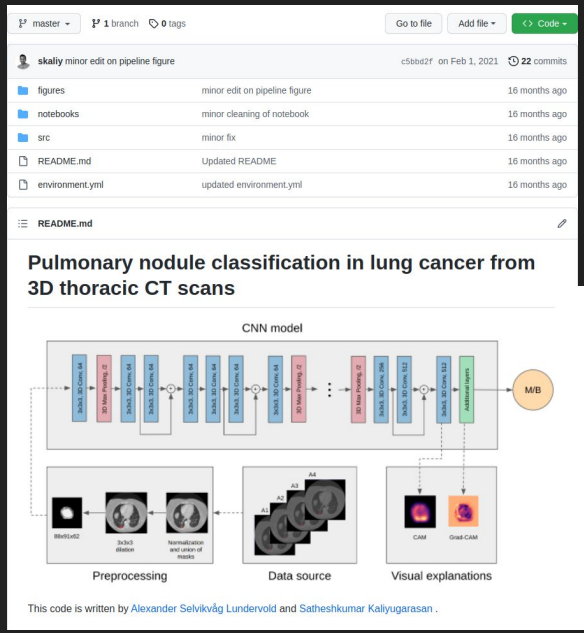
Regression

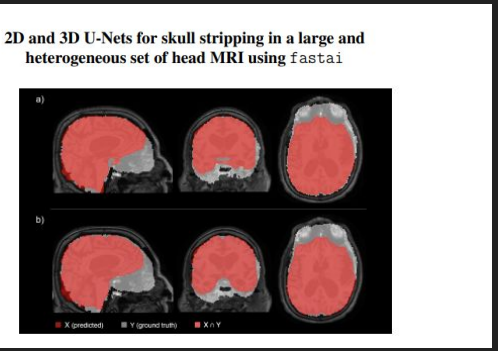


Segmentation

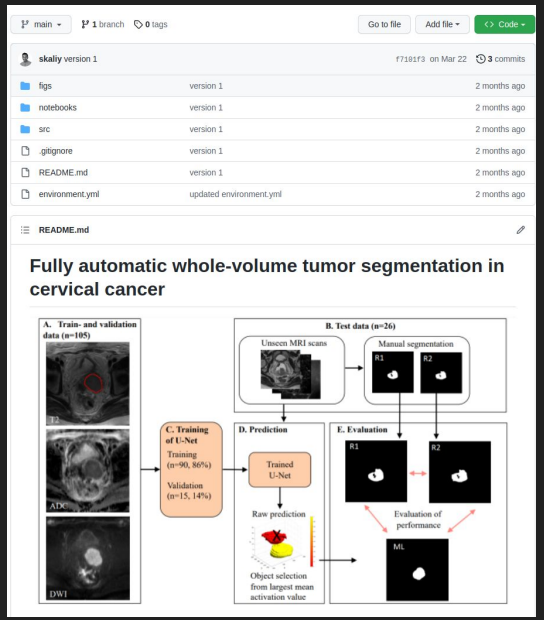


Scientific publications





V1

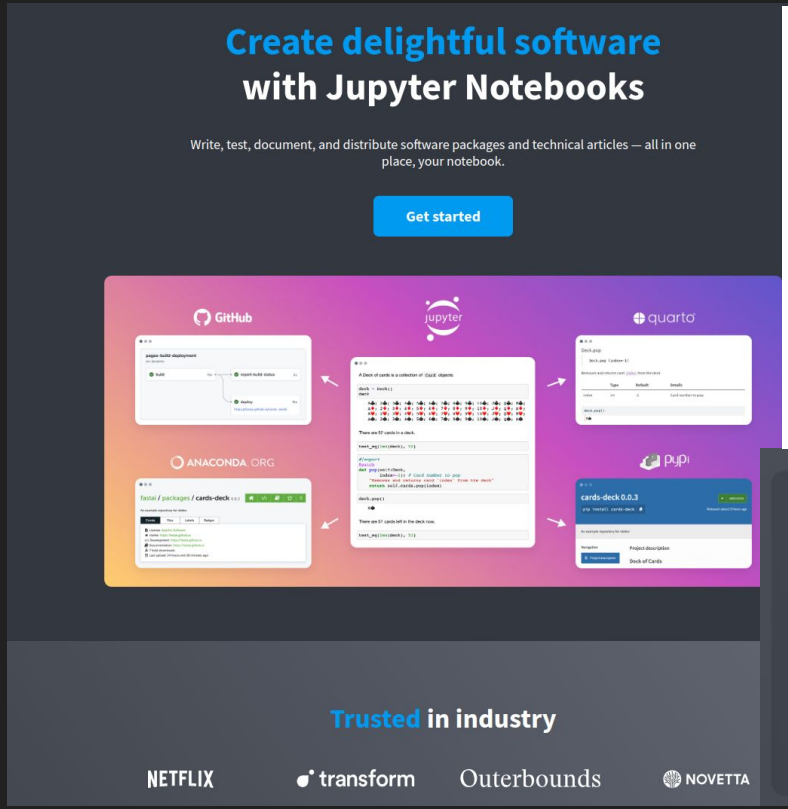


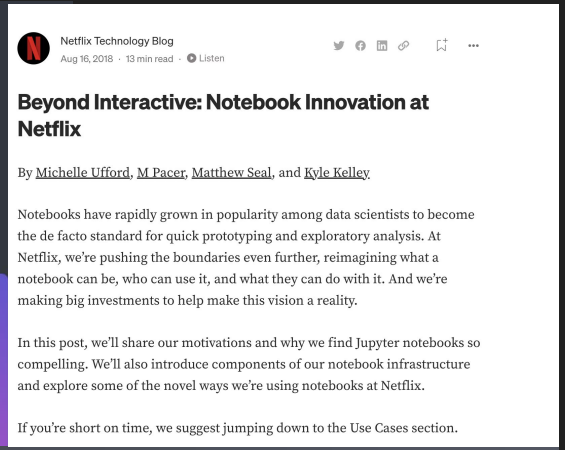
V2

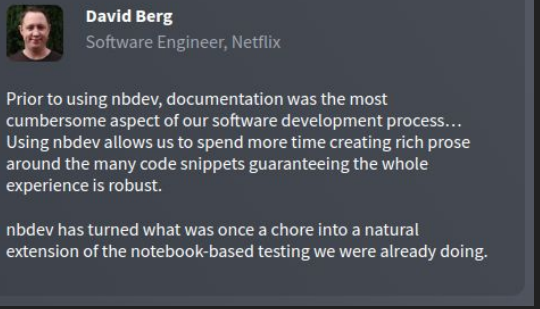
Other projects are currently being pursued

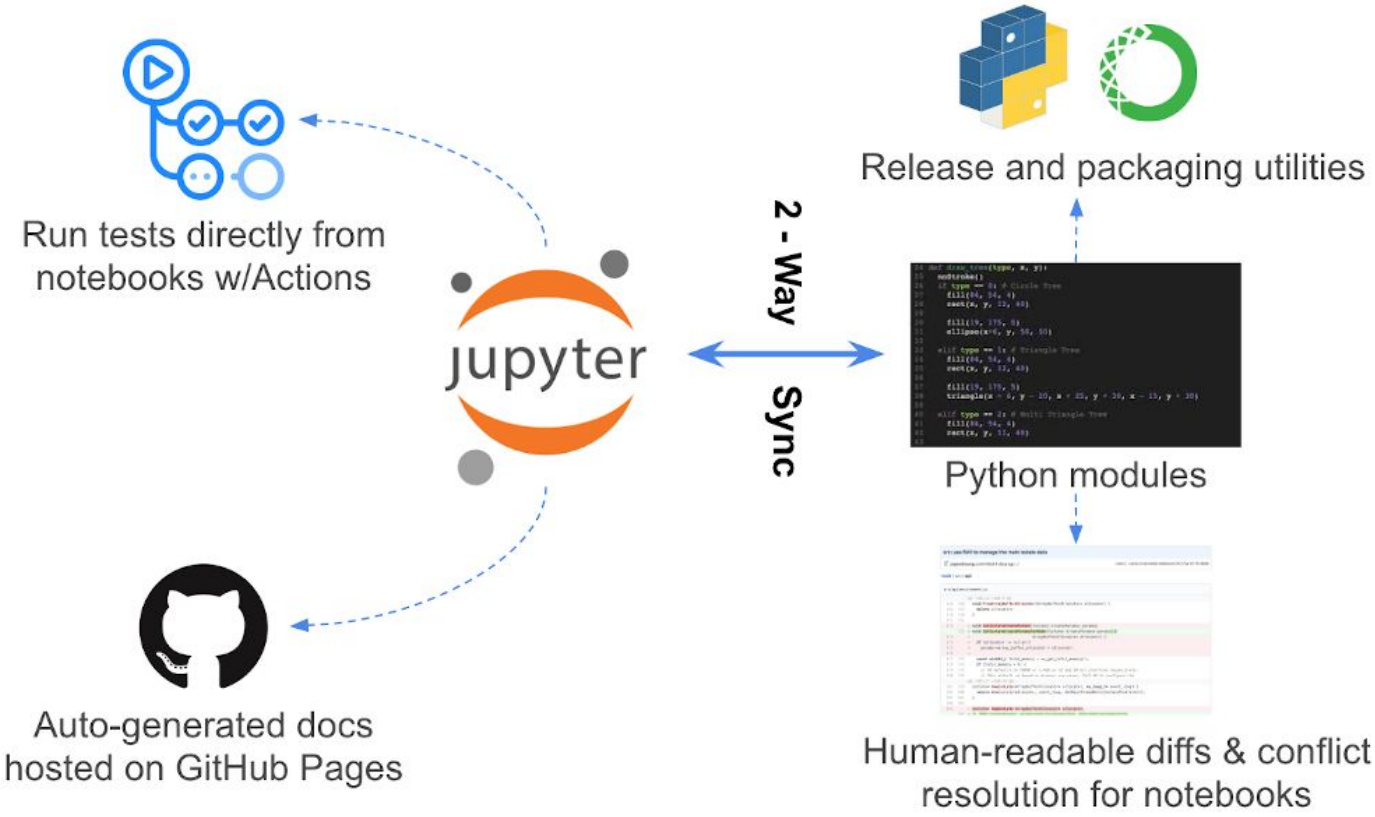
Let’s look at how we developed fastMONAI

Nbdev: tool for exploratory programming








Run tests directly from notebooks w/Actions
jupyter
2 - Way
Sync
Release and packaging utilities
Python modules
Auto-generated docs hosted on GitHub Pages
Human-readable diffs & conflict resolution for notebooks

MMIV-ML / **fastMONAI** Public

generated from fastai/nbdev-template

Edit Pins | Watch 2 | Fork 2 | Star 26

Code | Issues | Pull requests | Actions | Projects | Security | Insights | Settings

master | 4 branches | 0 tags | Go to file | Add file | Code

skaliy Merge pull request #9 from MMIV-ML/develop ... 0f94992 18 days ago 56 commits

.github/workflows	update deploy.yml	20 days ago
fastMONAI	Dev: minor fix for multimodal path from dataframe	18 days ago
nbs	Dev: minor fix for multimodal path from dataframe	18 days ago
paper	Dev: colab print message	21 days ago
.gitignore	Update .gitignore	29 days ago
CNAME	Dev: add CNAME	28 days ago
CONTRIBUTING.md	links to tutorials in Colab added	23 days ago
LICENSE	Initial commit	29 days ago
MANIFEST.in	Initial commit	29 days ago
README.md	Docs: changed PyPI badge color	20 days ago
settings.ini	Dev: minor fix for multimodal path from dataframe	18 days ago
setup.py	Initial commit	29 days ago

README.md

Overview

CI passing | Docs passing | PyPI version v0.1.4

A low-code Python-based open source deep learning library built on top of fastai, MONAI, and TorchIO.

fastMONAI simplifies the use of state-of-the-art deep learning techniques in 3D medical image analysis for solving classification, regression, and segmentation tasks. fastMONAI provides the users with functionalities to step through data loading, preprocessing, training, and result interpretations.

Note: This documentation is also available as interactive notebooks.

About

Simplifying deep learning for medical imaging

fastmonai.no

deeplearning | medicalimaging

Readme

Apache-2.0 license

26 stars

2 watching

2 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Contributors 3

skaliy Sathiesh Kaliyugarasan

github-actions[bot]

alu042 Alexander Lundervold

Environments 1

github-pages Active

Languages

Jupyter Notebook 97.8% **Python** 2.1%

Notebooks folder

master ▾ **fastMONAI / nbs /** Go to file Add file ▾ ⋯

skaliy Dev: minor fix for multimodal path from dataframe ✓ f685928 18 days ago **History**

[ ]:
```
#| default_exp vision_core
```

[ ]:
```
#| hide
from nbdev.showdoc import *
```

[ ]:
```
#| export
from fastMONAI.vision_plot import *
from fastai.data.all import *
from torchio import ScalarImage, LabelMap, ToCanonical, Resample
import pickle
import warnings
```

•••

•••

Load images

•••

•••

[ ]:
```
#| export
def med_img_reader(fn:(str, Path), # Image path
                   dtype=torch.Tensor, # Datatype (MedImage, MedMask, torch.Tensor)
                   resample:list=None, # Wheter to resample image to different voxel sizes and image dimensions.
                   reorder:bool=False, # Wheter to reorder the data to be closest to canonical (RAS+) orientation.
                   only_tensor:bool=True # Wheter to return only image tensor. If False, return ScalerImage or LabelMap object.
                  ):
    '''Load a medical image data. 4D tensor is returned as `dtype` if `only_tensor` is True, otherwise return ScalerImage or LabelMap.'''

    if ';' in fn:
        img_fns = fn.split(';')
        return _multi_channel(img_fns, reorder, resample, dtype=dtype)
        #add return metadata option to get original information

    o = _load(fn, dtype=dtype)
    o, org_metadata = _preprocess(o, reorder, resample)

    if only_tensor: return dtype(o.data.type(torch.float))

    return o, org_metadata
```

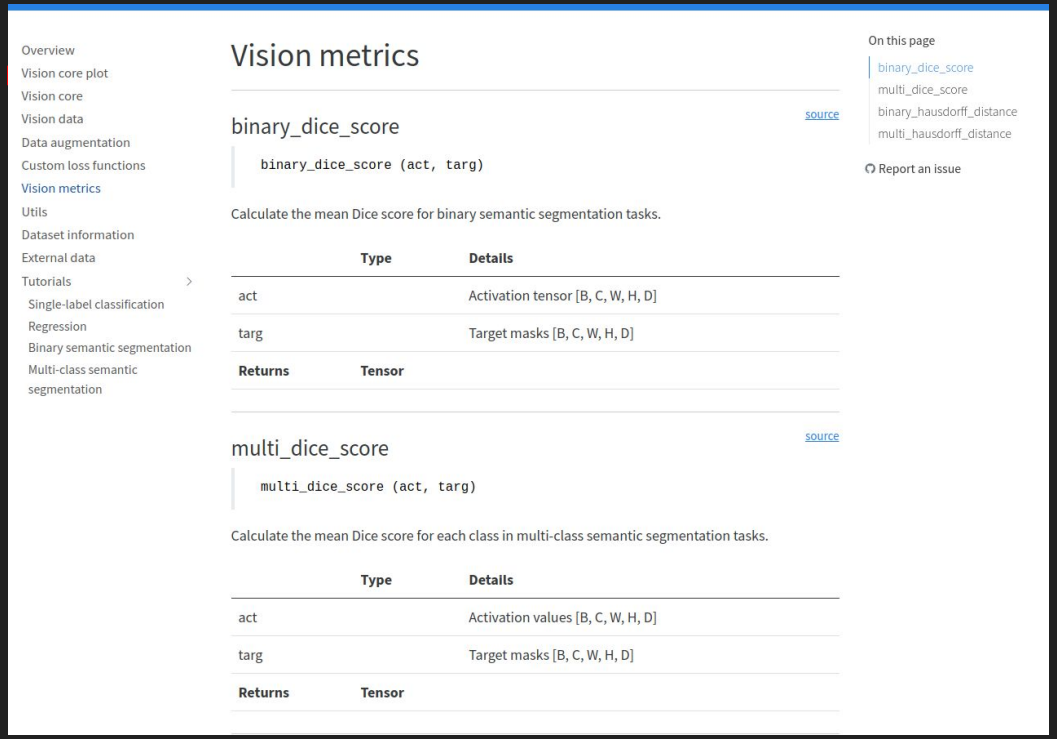
29 days ago
29 days ago
29 days ago
18 days ago
29 days ago
25 days ago
29 days ago
29 days ago
21 days ago
29 days ago
22 days ago
21 days ago
21 days ago
21 days ago
21 days ago
22 days ago

Source code

master ▾ fastMONAI / fastMONAI / Go to file Add file ▾ ···

```
# %% ../nbs/01_vision_core.ipynb 8
def med_img_reader(fn:(str, Path), # Image path
                   dtype=torch.Tensor, # Datatype (MedImage, MedMask, torch.Tensor)
                   resample:list=None, # Wheter to resample image to different voxel sizes and image dimensions.
                   reorder:bool=False, # Wheter to reorder the data to be closest to canonical (RAS+) orientation.
                   only_tensor:bool=True # Wheter to return only image tensor. If False, return ScalerImage or LabelMap object.
                  ):
    '''Load a medical image data. 4D tensor is returned as `dtype` if `only_tensor` is True, otherwise return ScalerImage or LabelMap.'''

    if ';' in fn:
        img_fns = fn.split(';')
        return _multi_channel(img_fns, reorder, resample, dtype=dtype)
        #add return metadata option to get original information

    o = _load(fn, dtype=dtype)
    o, org_metadata = _preprocess(o, reorder, resample)

    if only_tensor: return dtype(o.data.type(torch.float))

    return o, org_metadata
```

Documentation page: fastmonai.no



Write tests in notebooks

Vision metrics

[ ]:
```python
#| export
def _calculate_dsc(pred, targ):
    ''' MONAI `compute_meandice`:
    https://docs.monai.io/en/stable/_modules/monai/metrics/meandice.html#compute_meandice
    '''

    return torch.Tensor([compute_meandice(p[None], t[None]) for p, t in list(zip(pred,targ))])
```

[ ]:
```python
#| export
def _calculate_haus(pred, targ):
    ''' MONAI `compute_hausdorff_distance`:
    https://docs.monai.io/en/stable/_modules/monai/metrics/hausdorff_distance.html#compute_hausdorff_distance
    '''

    return torch.Tensor([compute_hausdorff_distance(p[None], t[None]) for p, t in list(zip(pred,targ))])
```

[ ]:
```python
#| export
def binary_dice_score(act, # Activation tensor [B, C, W, H, D]
                      targ # Target masks [B, C, W, H, D]
                     ) -> torch.Tensor:
    '''Calculate the mean Dice score for binary semantic segmentation tasks.'''

    pred = pred_to_binary_mask(act)
    dsc = _calculate_dsc(pred.cpu(), targ.cpu())

    return torch.mean(dsc)
```

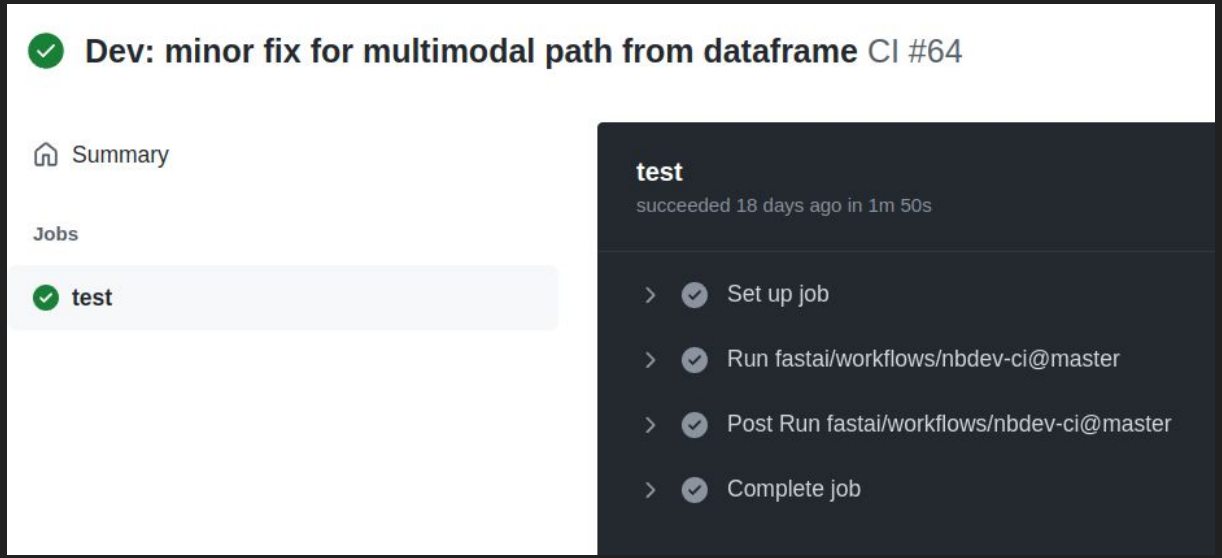
•••

•••

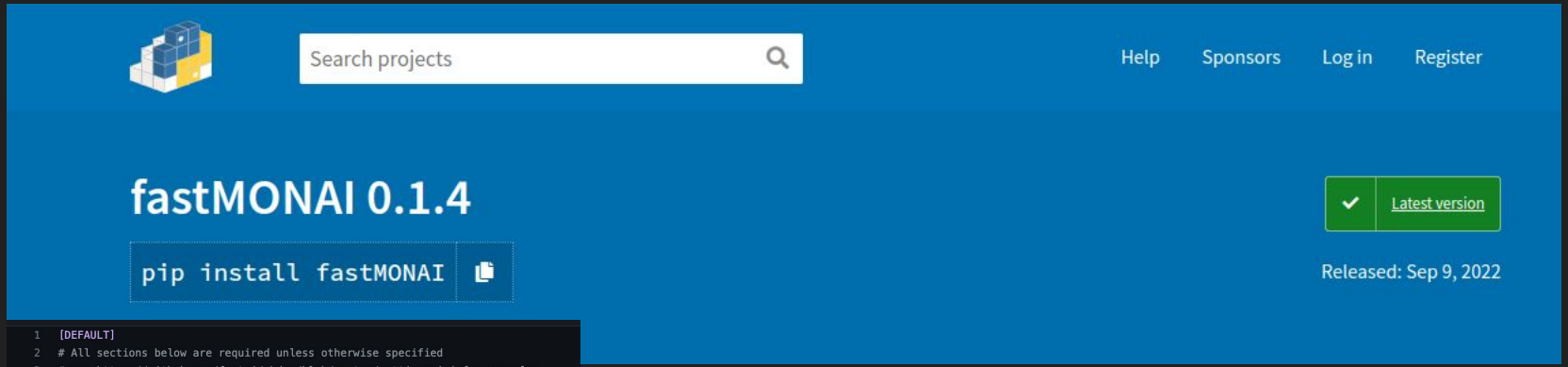
•••

[ ]:
```python
#| hide

# Test Dice score and Hausdorff distance
pred = torch.zeros((1,1,10,10,10))
pred[:,:,:5, :5, :5] = 1

targ = torch.zeros((1,1,10,10,10))
targ[:,:,:5, :5, :5] = 1

dsc = float(_calculate_dsc(pred, targ))
haus = float(_calculate_haus(pred,targ))

assert dsc == 1.0
assert haus == 0.0
```

Continuous integration with GitHub Actions



Publish to PyPI



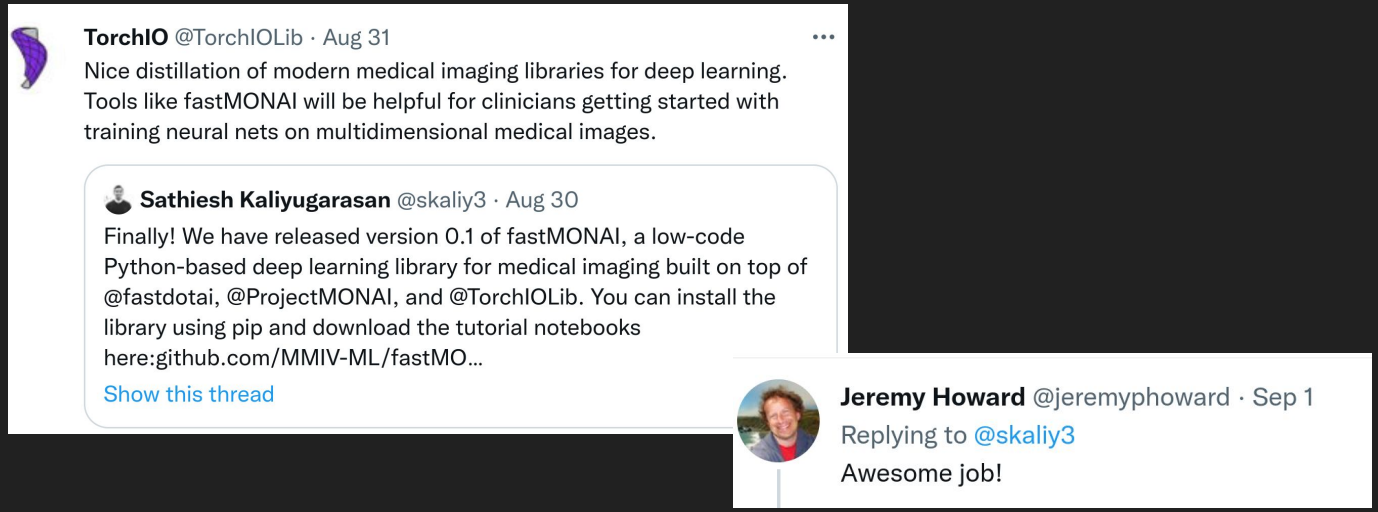

```
[DEFAULT]
# All sections below are required unless otherwise specified
# see https://github.com/fastai/nbdev/blob/master/settings.ini for examples

### Python Library ###
lib_name = fastMONAI
min_python = 3.7
version = 0.1.4
### OPTIONAL ###

requirements = fastai==2.7.9 monai==0.8.1 torchio==0.18.76 xlrd>=1.2.0
dev_requirements = ipywidgets nbdev tabulate

### nbdev ###
nbs_path = nbs
doc_path = _docs
recursive = False
tst_flags = notest
```

1-click Colab notebooks

Getting started

The best way to get started using fastMONAI is to read the paper and look at the step-by-step tutorial-like notebooks to learn how to train your own models on different tasks (e.g., classification, regression, segmentation). See the docs at https://fastmonai.no for more information.

Notebook	1-Click Notebook
09a_tutorial_classification.ipynb shows how to construct a binary classification model based on MRI data.	Open in Colab
09b_tutorial_regression.ipynb shows how to construct a model to predict the age of a subject from MRI scans ("brain age").	Open in Colab
09c_tutorial_binary_segmentation.ipynb shows how to do binary segmentation (extract the left atrium from monomodal cardiac MRI).	Open in Colab
09d_tutorial_multiclass_segmentation.ipynb shows how to perform segmentation from multimodal MRI (brain tumor segmentation).	Open in Colab


TorchIO @TorchIOLib · Aug 31
Nice distillation of modern medical imaging libraries for deep learning. Tools like fastMONAI will be helpful for clinicians getting started with training neural nets on multidimensional medical images.
Sathiesh Kaliyugarasan @skaliy3 · Aug 30
Finally! We have released version 0.1 of fastMONAI, a low-code Python-based deep learning library for medical imaging built on top of @fastdotai, @ProjectMONAI, and @TorchIOLib. You can install the library using pip and download the tutorial notebooks here:github.com/MMIV-ML/fastMO...
Show this thread
Jeremy Howard @jeremyphoward · Sep 1
Replying to @skaliy3
Awesome job!

Thank you!

https://skaliy.no