

fastMONAI

Simplifying deep learning for medical image analysis



Western Norway
University of
Applied Sciences



MMIV

HELSE







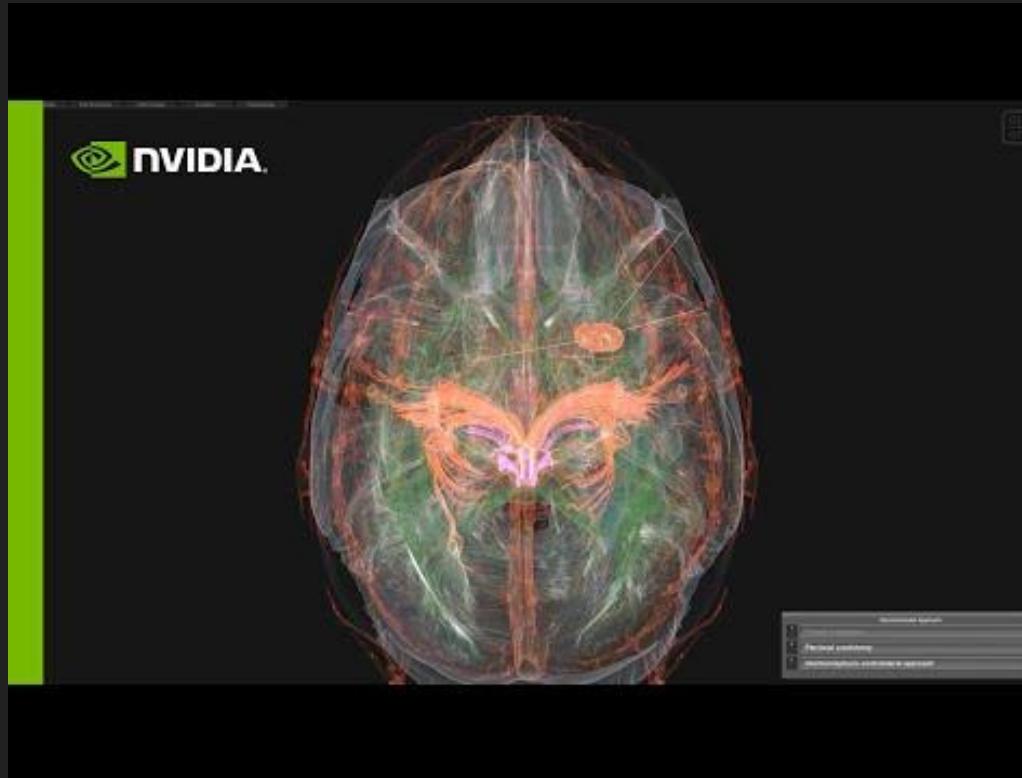




fastai + MONAI = fastMONAI

- **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 [1]: 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 CacheDataset  
from monai.losses import DiceLoss  
from monai.metrics import DiceMetric  
from monai.transforms import ToNumpy  
from monai.transforms import  
    Activations,  
    AsChannelFirstd,  
    AsDiscrete,  
    CenterSpatialCropd,  
    LoadImaged,  
    MapTransform,  
    NormalizeIntensityd,  
    Orientationd,  
    RandFlipd,  
    RandGaussianIntensityd,  
    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: 1.7.0+cu106e91  
MONAI Flags: HAS EXT = False, USE COMPILED = False  
MONAI rev id: 0563a4467fd60fece92d91cf47261868d171a1
```

```
Optional dependencies:
```

```
Pytorch Ignite version: 0.4.2
```

```
Mlib version: 3.2.1
```

```
scikit-image version: 0.15.0
```

```
Pillow version: 8.0.0
```

```
Tensorboard version: 2.1.15-0+nv
```

```
gdown version: 3.12.2
```

```
Torchvision version: 0.8.0+0
```

```
ITK version: 5.1.2
```

```
tqdm version: 4.54.1
```

```
lru-dict 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 no 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 0 is the background  
    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 = d[key]  
            # 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(result[-1], d[key] == 1))  
            # merge labels 0, 1, 2 and 3 to construct all  
            result.append(np.logical_or(result[-1], d[key] == 0))  
            # 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"),  
    CenterSpatialCropd(keys=["image", "label"], roi_size=[128, 128, 64], random_size=False),  
    RandFlipd(keys=["image", "label"], prob=0.5, spatial_axis=0),  
    RandGaussianIntensityd(keys="image", nonzero=True, channel_wise=True),  
    NormalizedIntensityd(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], random_size=False),  
    NormalizedIntensityd(keys="image", nonzero=True, channel_wise=True),  
    ToTensord(keys=["image", "label"]),  
])
```

Quickly load data with DecathlonDataset

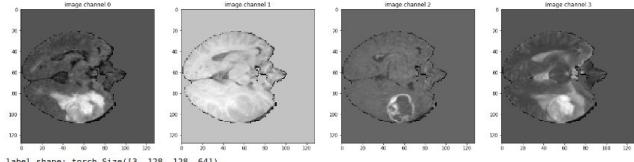
Here we use DecathlonDataset to automatically download and extract the dataset. It inherits MONAI CacheDataset, so we set cache_num=100 to cache 100 items for training and use the default args to cache all the items for validation.

```
train_ds = DecathlonDataset(  
    root_dir=root_dir,  
    task="Task03_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="Task03_BrainTumour",  
    transform=val_transform,  
    section="validation",  
    download=False,  
    num_workers=4,  
)  
  
val_loader = DataLoader(val_ds, batch_size=2, shuffle=False, num_workers=4)
```

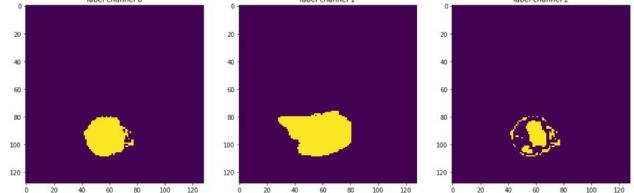
Check data shape and visualize

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

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

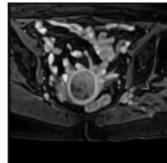
_, reorder = dataset_obj.suggestion()
resample = [1,1,1]

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)

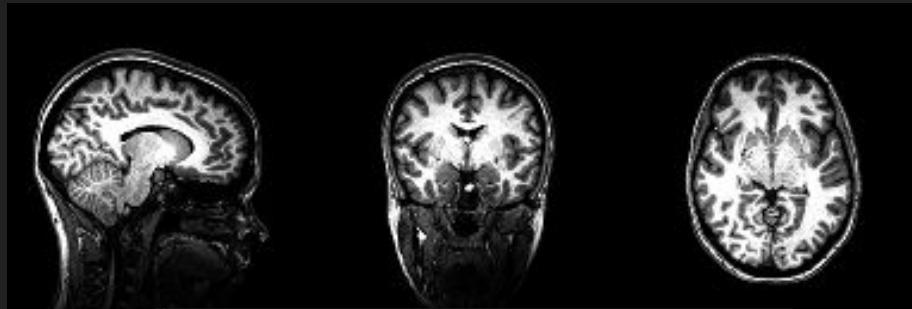
dls = dblock.dataloaders(df, bs=4)

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

```



Deep learning for 3D images?

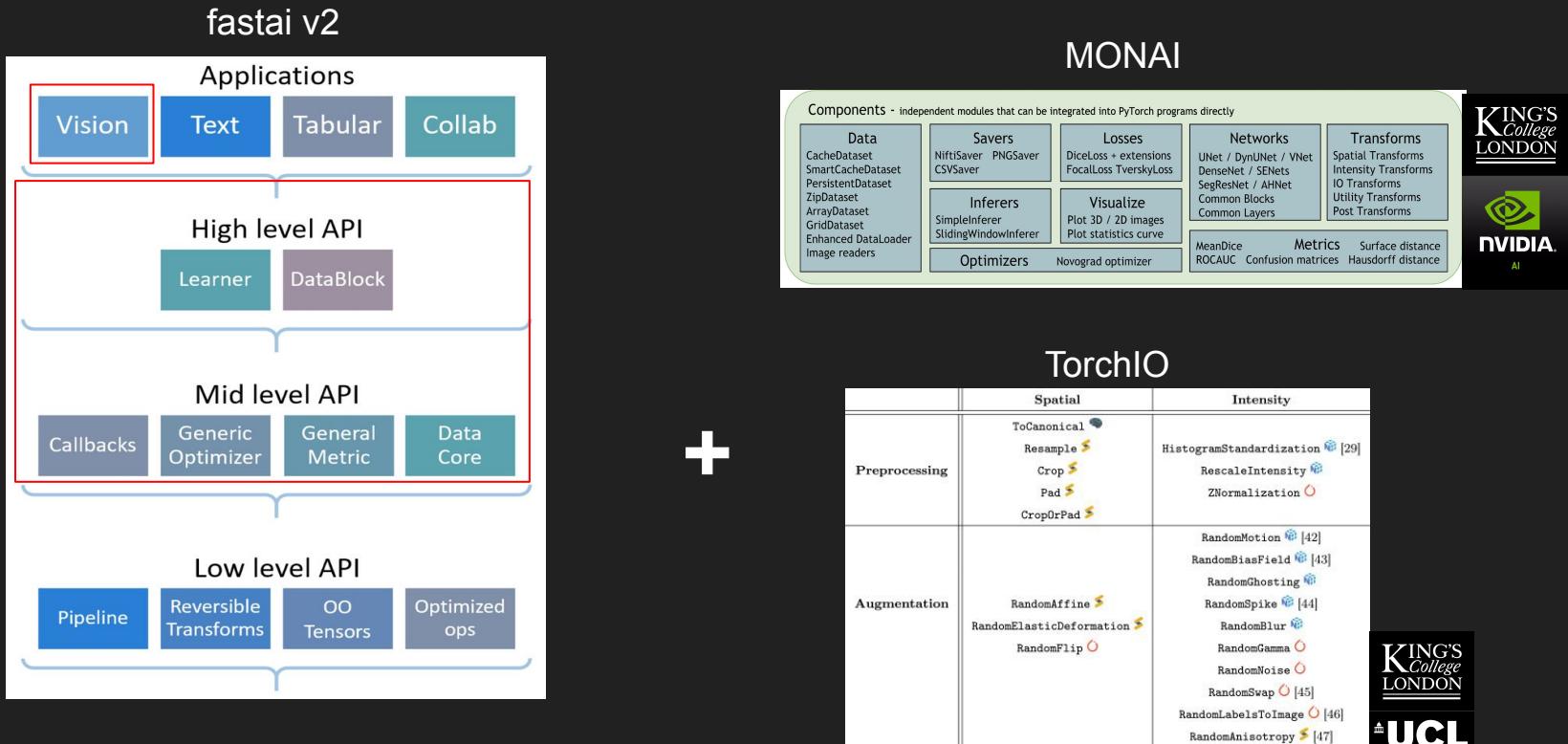


```
img.shape
```

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

Not supported in fastai

fastMONAI



fastMONAI



Learner

Callbacks

Datablock

Optimizers



Networks

Metrics



Transforms

Well-documented

The screenshot shows the MONAI documentation page for the UNETR model. It includes a search bar, a navigation menu with links like 'What's New', 'Highlights', 'API Reference', 'Installation Guide', 'Development', and 'More'. Below the menu, there's a section titled 'Return type: `Tensor`' with a note about the `UNet` instance referring to the former. The main content area displays the `class nn.Module` code for `UNET`, which defines a class with various layers and parameters. At the bottom, there are examples of code snippets for different use cases.

The screenshot shows the fastai documentation site. The top navigation bar includes links for 'Installing', 'About fastai', 'Migrating from other libraries', 'Windows Support', 'Tests', 'Contributing', 'Docker Containers', and a search bar. The main content area features a large heading 'Welcome to fastai' with the subtext 'fastai simplifies training fast and accurate neural nets using modern best practices'. Below this are sections for 'Quick start', 'Tutorials', 'Interpretation of Predictions', 'Data', 'Core', and 'Transforms'. A sidebar on the right lists 'On this page' sections such as 'Getting started', 'Data structures', 'Batch-based pipelines', 'Transforms', 'Preprocessing', 'Augmentation', 'Others', and 'Medical image datasets'. There are also sections for 'Additional interfaces', 'Examples gallery', and 'GitHub repository'.

The screenshot shows the TorchIO documentation site. The top navigation bar includes links for 'Installing', 'About fastai', 'Migrating from other libraries', 'Windows Support', 'Tests', 'Contributing', 'Docker Containers', and a search bar. The main content area features a large heading 'Welcome to fastai' with the subtext 'fastai simplifies training fast and accurate neural nets using modern best practices'. Below this are sections for 'Quick start', 'Tutorials', 'Interpretation of Predictions', 'Data', 'Core', and 'Transforms'. A sidebar on the right lists 'On this page' sections such as 'Getting started', 'Data structures', 'Batch-based pipelines', 'Transforms', 'Preprocessing', 'Augmentation', 'Others', and 'Medical image datasets'. There are also sections for 'Additional interfaces', 'Examples gallery', and 'GitHub repository'.

fastMONAI has step-by-step tutorials using open datasets

Classification

```
from fastMONAI.vision.core import *
from fastMONAI.vision.data import *
from fastMONAI.vision.loss import *
from fastMONAI.metrics import *
from fastMONAI.schedulers import *
from fastMONAI.utils import *

# NB 028 - Path config
LOCAL_DATA = MR_DCM_DIR + "/data"

STUDY_DIR = download_idx_data(LOCAL_DATA)

# 2022-05-31 10:40:41,127 - INFO - Expected mdf is None, skip mdf check for file /home/cathiezh/machine_learning/fastMONAI/rbrc.../data/123/123-T1.tar.
# 2022-05-31 10:40:41,127 - INFO - File exists: /home/cathiezh/machine_learning/fastMONAI/rbrc.../data/123/123-T1.tar, skipped downloading.
# 2022-05-31 10:40:41,128 - INFO - Writing into directory: /home/cathiezh/machine_learning/fastMONAI/rbrc.../data/123/123...
# 2022-05-31 10:40:41,128 - INFO - File exists: /home/cathiezh/machine_learning/fastMONAI/rbrc.../data/123/123-T1.vtk.
# 2022-05-31 10:40:41,128 - INFO - File exists: /home/cathiezh/machine_learning/fastMONAI/rbrc.../data/123/123-T1.xls, skipped downloading.
# 2022-05-31 10:40:41,128 - INFO - File exists: /home/cathiezh/machine_learning/fastMONAI/rbrc.../data/123/123-T1.xls, skipped downloading.

# df = pd.read_csv(STUDY_DIR + "dataset.csv")
df.head()
```

13 path subject_id gender age_scanner

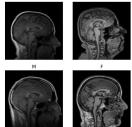
```
1 1 0 30.80
2 1 0 46.71
3 1 0 34.24
4 1 0 34.29
```

```
dataset = MedDataset(path=STUDY_DIR + "T1_images", max_workers=12)
```

```
resample, reorder = dataset._get_suspicion()
```

```
ds = resample_and_rescale(ds, slice_label, calib, label, calib, label, tmean2D(normalization), PadCrop(size=256), resample=resample, reorder=reorder, bsize=16)
```

```
ds = ds.batch(batch_size=8, num_anatomical_plane="A")
```



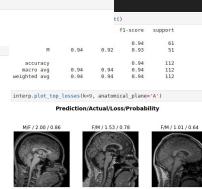
```
TrueModel = torch.nn.Sequential(
    nn.Conv2d(1, 32, kernel_size=3, stride=2),
    nn.ReLU(),
    nn.BatchNorm2d(32),
    nn.Conv2d(32, 64, kernel_size=3, stride=2),
    nn.ReLU(),
    nn.BatchNorm2d(64),
    nn.Conv2d(64, 128, kernel_size=3, stride=2),
    nn.ReLU(),
    nn.BatchNorm2d(128),
    nn.Conv2d(128, 256, kernel_size=3, stride=2),
    nn.ReLU(),
    nn.BatchNorm2d(256)
)
```

```
learn = Learner(ds, model, metrics=accuracy)
```

```
learn.fit_one_cycle(1)
```

```
epoch 0 loss 0.71002 acc 0.68707 time 0.03808 0.0311
```

```
epoch 1 loss 0.34020 acc 0.98982 time 0.02260 0.0311
```



```
from fastMONAI.vision.core import *
from fastMONAI.vision.data import *
from fastMONAI.vision.loss import *
from fastMONAI.metrics import *
from fastMONAI.schedulers import *
from fastMONAI.utils import *

# NB 028 - Path config
LOCAL_DATA = MR_DCM_DIR + "/data"

STUDY_DIR = download_idx_data(LOCAL_DATA)

# 2022-05-29 13:13:26,737 - INFO - Expected mdf is None, skip mdf check for file /home/cathiezh/machine_learning/fastMONAI/rbrc.../data/123/123-T1.tar.
# 2022-05-29 13:13:26,738 - INFO - File exists: /home/cathiezh/machine_learning/fastMONAI/rbrc.../data/123/123-T1.tar, skipped downloading.
# 2022-05-29 13:13:26,738 - INFO - Writing into directory: /home/cathiezh/machine_learning/fastMONAI/rbrc.../data/123/123...
# 2022-05-29 13:13:26,738 - INFO - File exists: /home/cathiezh/machine_learning/fastMONAI/rbrc.../data/123/123-T1.vtk.
# 2022-05-29 13:13:26,738 - INFO - File exists: /home/cathiezh/machine_learning/fastMONAI/rbrc.../data/123/123-T1.xls, skipped downloading.
# 2022-05-29 13:13:26,738 - INFO - File exists: /home/cathiezh/machine_learning/fastMONAI/rbrc.../data/123/123-T1.xls, skipped downloading.

# df = pd.read_csv(STUDY_DIR + "dataset.csv")
df.head()
```

13 path subject_id gender age_scanner

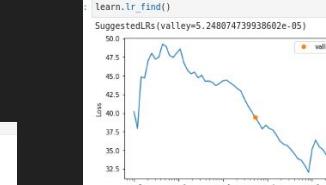
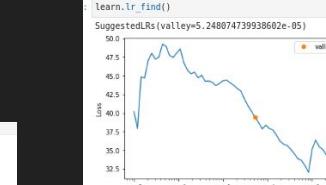
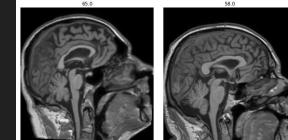
```
1 1 0 30.80
2 1 0 46.71
3 1 0 34.24
4 1 0 34.29
```

```
dataset = MedDataset(path=STUDY_DIR + "T1_images", max_workers=12)
```

```
resample, reorder = dataset._get_suspicion()
```

```
ds = resample_and_rescale(ds, slice_label, calib, label, calib, label, tmean2D(normalization), PadCrop(size=256), resample=resample, reorder=reorder, bsize=16)
```

```
ds = ds.batch(batch_size=8, num_anatomical_plane="A")
```



Regression

```
from fastMONAI.vision.core import *
from fastMONAI.vision.data import *
from fastMONAI.vision.loss import *
from fastMONAI.metrics import *
from fastMONAI.schedulers import *
from fastMONAI.utils import *

# NB 028 - Path config
LOCAL_DATA = MR_DCM_DIR + "/data"

STUDY_DIR = download_idx_data(LOCAL_DATA)

# 2022-05-29 13:13:26,737 - INFO - Expected mdf is None, skip mdf check for file /home/cathiezh/machine_learning/fastMONAI/rbrc.../data/123/123-T1.tar.
# 2022-05-29 13:13:26,738 - INFO - File exists: /home/cathiezh/machine_learning/fastMONAI/rbrc.../data/123/123-T1.vtk.
# 2022-05-29 13:13:26,738 - INFO - Writing into directory: /home/cathiezh/machine_learning/fastMONAI/rbrc.../data/123/123...
# 2022-05-29 13:13:26,738 - INFO - File exists: /home/cathiezh/machine_learning/fastMONAI/rbrc.../data/123/123-T1.xls.
# 2022-05-29 13:13:26,738 - INFO - File exists: /home/cathiezh/machine_learning/fastMONAI/rbrc.../data/123/123-T1.xls, skipped downloading.

# df = pd.read_csv(STUDY_DIR + "dataset.csv")
df.head()
```

13 path subject_id gender age_scanner

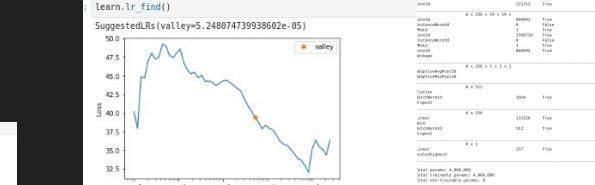
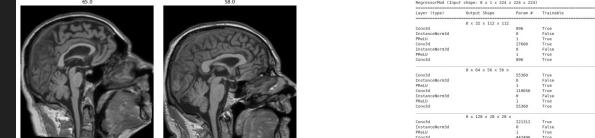
```
1 1 0 30.80
2 1 0 46.71
3 1 0 34.24
4 1 0 34.29
```

```
dataset = MedDataset(path=STUDY_DIR + "T1_images", max_workers=12)
```

```
resample, reorder = dataset._get_suspicion()
```

```
ds = resample_and_rescale(ds, slice_label, calib, label, calib, label, tmean2D(normalization), PadCrop(size=256), resample=resample, reorder=reorder, bsize=16)
```

```
ds = ds.batch(batch_size=8, num_anatomical_plane="A")
```



```
from fastMONAI.vision.core import *
from fastMONAI.vision.data import *
from fastMONAI.vision.loss import *
from fastMONAI.metrics import *
from fastMONAI.schedulers import *
from fastMONAI.utils import *

# NB 028 - Path config
LOCAL_DATA = MR_DCM_DIR + "/data"

STUDY_DIR = download_idx_data(LOCAL_DATA)

# 2022-05-29 13:13:26,737 - INFO - Expected mdf is None, skip mdf check for file /home/cathiezh/machine_learning/fastMONAI/rbrc.../data/123/123-T1.tar.
# 2022-05-29 13:13:26,738 - INFO - File exists: /home/cathiezh/machine_learning/fastMONAI/rbrc.../data/123/123-T1.vtk.
# 2022-05-29 13:13:26,738 - INFO - Writing into directory: /home/cathiezh/machine_learning/fastMONAI/rbrc.../data/123/123...
# 2022-05-29 13:13:26,738 - INFO - File exists: /home/cathiezh/machine_learning/fastMONAI/rbrc.../data/123/123-T1.xls.
# 2022-05-29 13:13:26,738 - INFO - File exists: /home/cathiezh/machine_learning/fastMONAI/rbrc.../data/123/123-T1.xls, skipped downloading.

# df = pd.read_csv(STUDY_DIR + "dataset.csv")
df.head()
```

13 path subject_id gender age_scanner

```
1 1 0 30.80
2 1 0 46.71
3 1 0 34.24
4 1 0 34.29
```

```
dataset = MedDataset(path=STUDY_DIR + "T1_images", max_workers=12)
```

```
resample, reorder = dataset._get_suspicion()
```

```
ds = resample_and_rescale(ds, slice_label, calib, label, calib, label, tmean2D(normalization), PadCrop(size=256), resample=resample, reorder=reorder, bsize=16)
```

```
ds = ds.batch(batch_size=8, num_anatomical_plane="A")
```



```
from fastMONAI.vision.core import *
from monai.apps import DecathlonDataset
from sklearn.model_selection import train_test_split

# We use the MONAI function DecathlonDataset to download the data and generate items for training.

path = Path("../data")
if not os.path.exists(path):
    DecathlonDataset(download=True, section="Task01_BrainTumour").download()

# Download external data
# We use the MONAI function DecathlonDataset to download the data and generate items for training.

path = Path("../data")
if not os.path.exists(path):
    DecathlonDataset(download=True, section="Task01_BrainTumour").download()

# training data = DecathlonDataset(download=True, section="Task01_BrainTumour", section="training", download=True,
#                                     transform=PadCrop((128, 128, 128), 0))
# validation data = DecathlonDataset(download=True, section="Task01_BrainTumour", section="validation", download=True,
#                                     transform=PadCrop((128, 128, 128), 0))

# 2022-09-01 17:36:43,299 - INFO - Verifying Task01_BrainTumour.tar: mdf = 240abf52f70e0301544901065072.
# 2022-09-01 17:36:43,300 - INFO - File exists: ../data/Task01_BrainTumour.tar, skipped downloading.
# 2022-09-01 17:36:43,301 - INFO - Non-empty folder exists in ../data/Task01_BrainTumour, skipped extracting.

df = pd.DataFrame(training_data)
df.info()

(388, 2)

Split the labeled data into training and test

train_df, test_df = train_test_split(df, test_size=0.1, random_state=42)
train_df.info()
test_df.info()

(340, 2), (39, 2)

Look at training data

med_dataset = MedDataset(img_list=train_df.label.tolist(), dtype=MEDMask, max_workers=12)

summary_df = med_dataset.summary()

summary_df.head()

dim_0 dim_1 dim_2 voxel_0 voxel_1 voxel_2 orientation
0 240 240 156 1.0 1.0 1.0 RAG+ /home/natheshchiluDev/fastMONAI/Task01_BrainTumour/Task01_002.nii.gz 349 example_per_total

resample, reorder = med_dataset.suspicion()
resample, reorder
((1.0, 1.0, 1.0), False)

img_size = med_dataset.get_largest_img(resample=reorder)
img_size
(240, 240, 156, 0)

b3d = b3d[224, 224, 128]

item_ids = [(Decompression, PadCrop((128, 128, 128), 0), RandomAffine(scales=0, degrees=5, isotropic=True))]

# b3d = MedDataset(download=True, transform=Compose([RandomAffine(scales=0, degrees=5, isotropic=True), PadCrop((128, 128, 128), 0)]), get_items_fn=MedMaskBlock, get_xColReader_fn=ImageReader, get_yColReader_fn=LabelReader, item_size=item_size, reorder=reorder, resample=reorder)

ds = b3d.dataset(ds.train, ds.valid)

# training and validation
# (len(ds.train), len(ds.valid), len(ds.valid_ds.items))
(280, 60)

ds.info_batch(anatomical_plane="A")
```



Scientific publications

Screenshot of a GitHub repository for "Pulmonary nodule classification in lung cancer from 3D thoracic CT scans".

Repository details:

- Branch: master
- Commits: 22
- Last commit: c55bd2f on Feb 1, 2021

File list:

- figures
- notebooks
- src
- README.md
- environment.yml

README.md content:

Pulmonary nodule classification in lung cancer from 3D thoracic CT scans

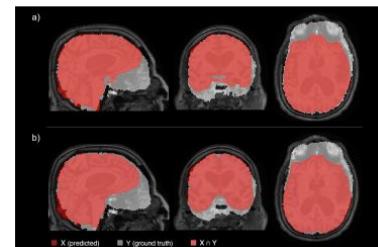
CNN model diagram:

Preprocessing: 3x3x3 dilation, Normalization and center of mass.

Data source: A1, A2, A3, A4.

Visual explanations: CAM, Grad CAM.

2D and 3D U-Nets for skull stripping in a large and heterogeneous set of head MRI using fastai



This code is written by Alexander Selvikvåg Lundervold and Satheshkumar Kalyugarasan .

V1

Screenshot of a GitHub repository for "Fully automatic whole-volume tumor segmentation in cervical cancer".

Repository details:

- Branch: main
- Commits: 3
- Last commit: f7161f3 on Mar 22

File list:

- figs
- notebooks
- src
- ignore
- README.md
- environment.yml

README.md content:

Fully automatic whole-volume tumor segmentation in cervical cancer

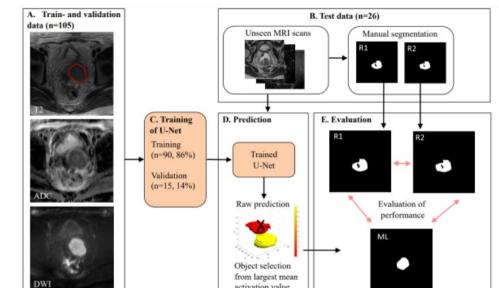
A. Train- and validation data (n=105)

B. Test data (n=26)

C. Training of U-Net

D. Prediction

E. Evaluation



V2

Other projects are currently being pursued

Let's look at how we developed fastMONAI

Nbdev: tool for exploratory programming

Create delightful software with Jupyter Notebooks

Write, test, document, and distribute software packages and technical articles — all in one place, your notebook.

[Get started](#)

Trusted in industry

NETFLIX transform Outerbounds NOVETTA

Netflix Technology Blog
Aug 16, 2018 · 13 min read · Listen

[...](#)

Beyond Interactive: Notebook Innovation at Netflix

By [Michelle Ufford](#), [M Pacer](#), [Matthew Seal](#), and [Kyle Kelley](#)

Notebooks have rapidly grown in popularity among data scientists to become the de facto standard for quick prototyping and exploratory analysis. At Netflix, we're pushing the boundaries even further, reimagining what a notebook can be, who can use it, and what they can do with it. And we're making big investments to help make this vision a reality.

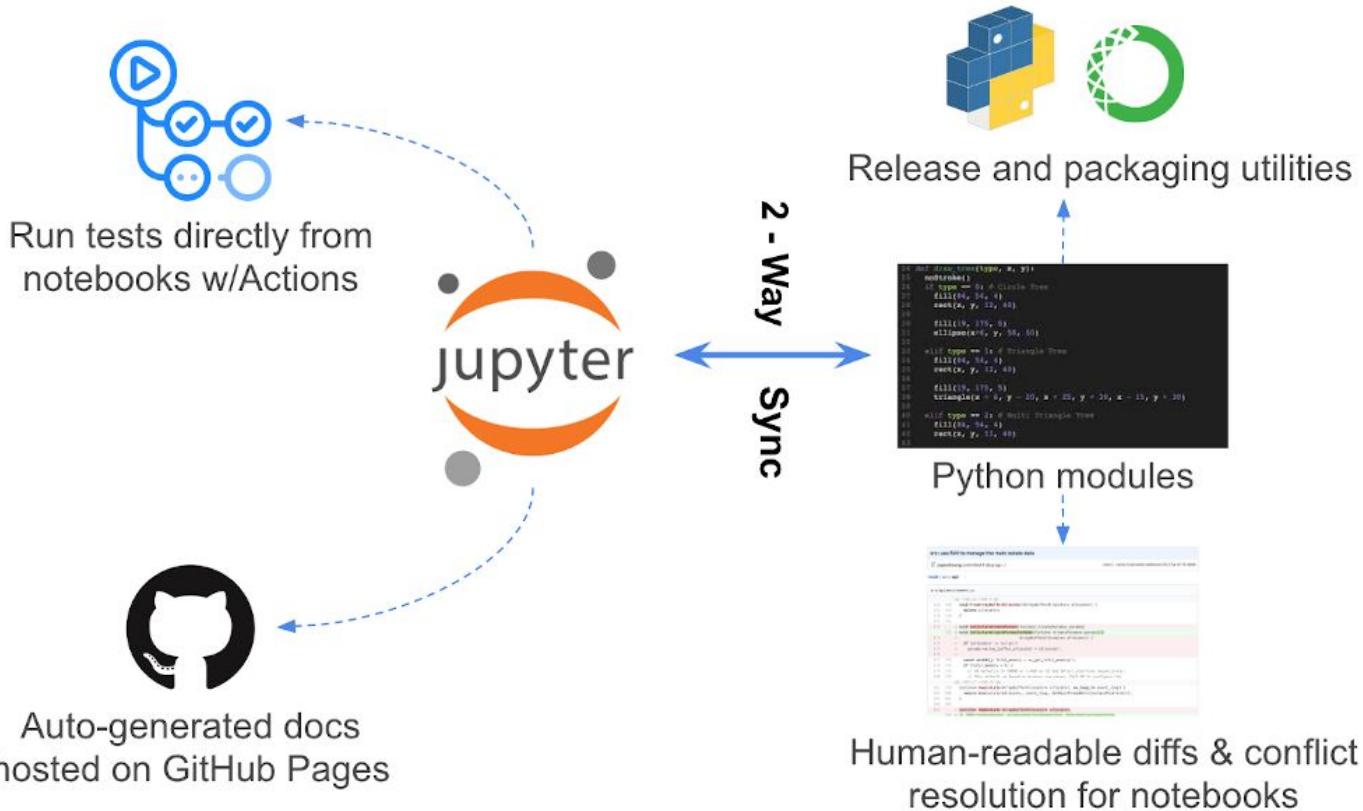
In this post, we'll share our motivations and why we find Jupyter notebooks so compelling. We'll also introduce components of our notebook infrastructure and explore some of the novel ways we're using notebooks at Netflix.

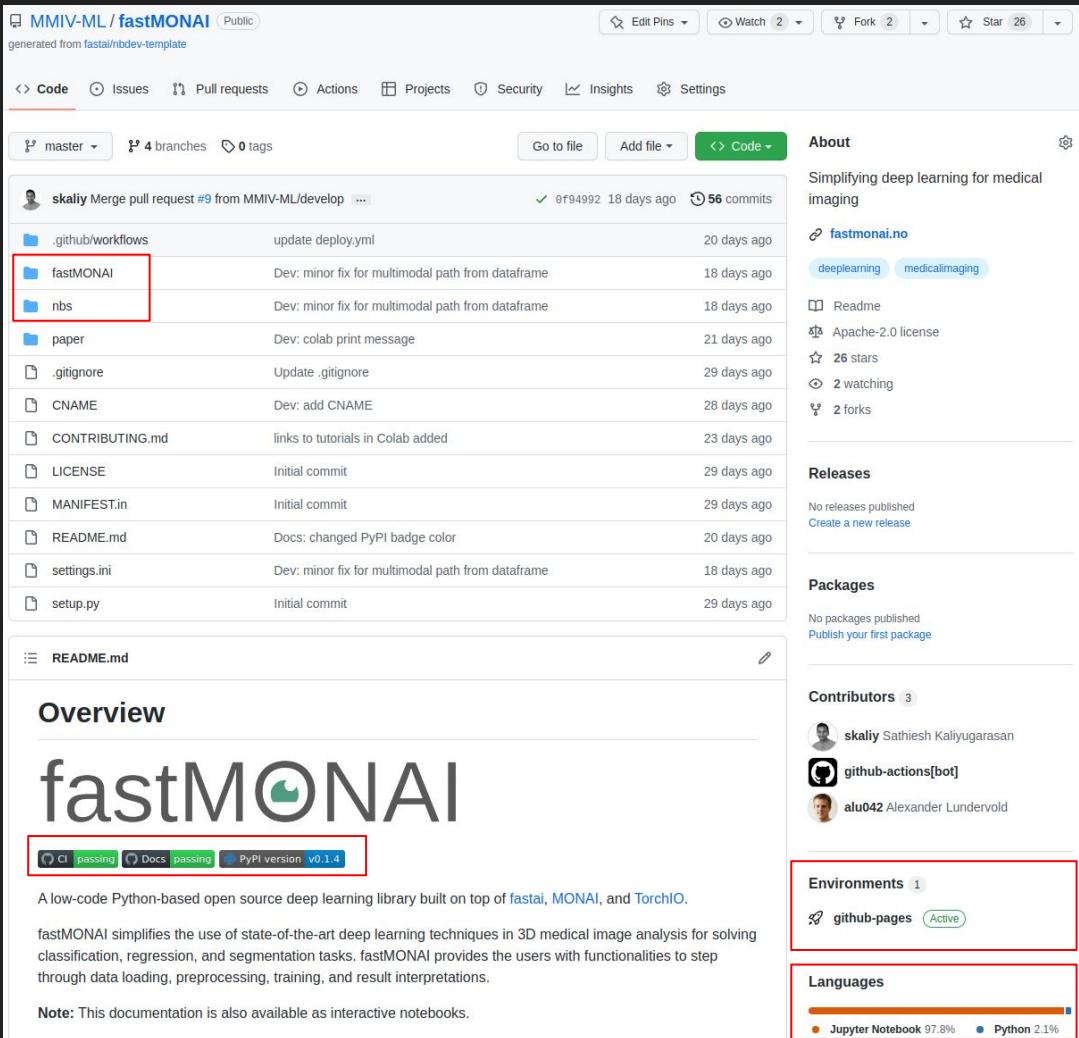
If you're short on time, we suggest jumping down to the Use Cases section.

David Berg
Software Engineer, Netflix

Prior to using nbdev, documentation was the most cumbersome aspect of our software development process... Using nbdev allows us to spend more time creating rich prose around the many code snippets guaranteeing the whole experience is robust.

nbdev has turned what was once a chore into a natural extension of the notebook-based testing we were already doing.



A screenshot of a GitHub repository page for "MMIV-ML/fastMONAI". The repository is public and was generated from "fastai/nbdev-template". The main navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Security, Insights, and Settings. The repository has 4 branches and 0 tags. A pull request by skaliy titled "Merge pull request #9 from MMIV-ML/develop" has been merged, showing 56 commits. The commit history lists changes in .github/workflows, fastMONAI, and nbs. The fastMONAI folder is highlighted with a red box. Other files listed include .gitignore, CNAME, CONTRIBUTING.md, LICENSE, MANIFEST.in, README.md, settings.ini, and setup.py. The README.md file contains an overview of fastMONAI, which is described as a low-code Python-based open source deep learning library built on top of fastai, MONAI, and TorchIO. It highlights its use for state-of-the-art deep learning techniques in 3D medical image analysis. The page also shows CI status (CI passing), documentation (Docs passing), and PyPI version (v0.1.4). It includes sections for Environments (github-pages Active), Languages (Jupyter Notebook 97.8%, Python 2.1%), and Contributors (skaliy, github-actions[bot], alu042). The right sidebar provides information about the repository, such as simplifying deep learning for medical imaging, the Apache-2.0 license, and 26 stars.

MMIV-ML / fastMONAI Public

generated from fastai/nbdev-template

<> 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

fastMONAI

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 Kaliyugaran

github-actions[bot]

alu042 Alexander Lundervold

Environments 1

github-pages Active

Languages

Jupyter Notebook 97.8% Python 2.1%

Notebooks folder

master fastMONAI / nbs /

skaliv 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 ScalarImage or LabelMap object.
                  ):
    '''Load a medical image data. 4D tensor is returned as `dtype` if `only_tensor` is True, otherwise return ScalarImage 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
```

Go to file Add file ...

Source code

master fastMONAI/fastMONAI/

Go to file Add file ...

```
# %% ..../nbs/01_vision_core.ipynb 8
def med_img_reader(fn: str, 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

Overview
Vision core plot
Vision core
Vision data
Data augmentation
Custom loss functions
Vision metrics
Utils
Dataset information
External data
Tutorials >
Single-label classification
Regression
Binary semantic segmentation
Multi-class semantic segmentation

Vision metrics

binary_dice_score [source](#) [Report an issue](#)

Calculate the mean Dice score for binary semantic segmentation tasks.

Type	Details
act	Activation tensor [B, C, W, H, D]
targ	Target masks [B, C, W, H, D]

Returns **Tensor**

multi_dice_score [source](#)

Calculate the mean Dice score for each class in multi-class semantic segmentation tasks.

Type	Details
act	Activation values [B, C, W, H, D]
targ	Target masks [B, C, W, H, D]

Returns **Tensor**

Write tests in notebooks

Vision metrics

```
[ 1]: #| 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))])

[ 1]: #| 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))])

[ 1]: #| 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)

***  

***  

***  

[ 1]: #| 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

The screenshot shows a GitHub Actions CI run titled "Dev: minor fix for multimodal path from dataframe CI #64". The summary indicates a success status with a green checkmark icon. The "test" job is expanded, showing its history of steps:

- > ✓ Set up job
- > ✓ Run fastai/workflows/nbdev-ci@master
- > ✓ Post Run fastai/workflows/nbdev-ci@master
- > ✓ Complete job

Each step is marked with a green checkmark icon, indicating they all succeeded. The entire run was completed 18 days ago in 1m 50s.

Publish to PyPI

The screenshot shows the PyPI website interface. At the top, there is a search bar with the placeholder "Search projects" and a magnifying glass icon. To the right of the search bar are links for "Help", "Sponsors", "Log in", and "Register". Below the header, the title "fastMONAI 0.1.4" is displayed in large white text. To the right of the title is a green button with a checkmark icon and the text "Latest version". Below the title, there is a "pip install fastMONAI" button with a clipboard icon. On the left side of the page, there is a dark sidebar containing a configuration file snippet:

```
1 [DEFAULT]
2 # All sections below are required unless otherwise specified
3 # see https://github.com/fastai/nbdev/blob/master/settings.ini for examples
4
5 ### Python Library ###
6 lib_name = fastMONAI
7 min_python = 3.7
8 version = 0.1.4
9 ### OPTIONAL ###
10
11 requirements = fastai==2.7.9 monai==0.8.1 torchio==0.18.76 xlrd>=1.2.0
12 dev_requirements = ipywidgets nbdev tabulate
13
14 ### nbdev ###
15 nbs_path = nbs
16 doc_path = _docs
17 recursive = False
18 tst_flags = nostest
19
```

At the bottom right of the main content area, the text "Released: Sep 9, 2022" is visible.

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 Kaliyugaranan @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

Replies to [@skaliy3](#)

Awesome job!

Thank you!



Western Norway
University of
Applied Sciences



MMIV



TROND
MOHN
FOUNDATION

HELSE •• VEST

<https://skaliy.no>