**Image Manipulation using Nilearn:**

This Jupyter notebook provides an introduction to using the \*\*Nilearn\*\* library for neuroimaging data manipulation and visualization. Below are the key points and steps covered in the notebook:

## \*\*Overview\*\*

The notebook focuses on:

1. \*\*Image Manipulation\*\*: Resampling, smoothing, cleaning, masking, and extracting signals from neuroimaging data.

2. \*\*Image Visualization\*\*: Plotting various types of brain images using Nilearn's visualization functions.

3. \*\*Advanced Techniques\*\*: Performing Independent Component Analysis (ICA) and Dictionary Learning on fMRI data.

### \*\*1. Setup\*\*

- \*\*Imports\*\*: Essential libraries such as `nilearn`, `numpy`, and `matplotlib` are imported to handle image processing and visualization.

```python

from nilearn import plotting, image as nli

import numpy as np

import pylab as plt

%matplotlib inline

```

- \*\*Loading Data\*\*: The anatomical (T1) and functional (BOLD) images of a subject are loaded using `nli.load\_img()`. The first 5 volumes of the BOLD image are removed to account for steady-state issues.

```python

t1 = nli.load\_img('/data/ds000114/sub-01/ses-test/anat/sub-01\_ses-test\_T1w.nii.gz')

bold = nli.load\_img('/data/ds000114/sub-01/ses-test/func/sub-01\_ses-test\_task-fingerfootlips\_bold.nii.gz').slicer[..., 5:]

```

### \*\*2. Image Manipulation with Nilearn\*\*

#### \*\*Mean Image Calculation\*\*

- \*\*Mean Image\*\*: The mean image of the BOLD data is computed in one line using `nli.mean\_img()`.

```python

img = nli.mean\_img(bold)

plotting.view\_img(img, bg\_img=img)

```

#### \*\*Resampling Images\*\*

- \*\*Resample to Template\*\*: The T1 image is resampled to match the dimensions of the mean BOLD image using `nli.resample\_to\_img()`.

```python

resampled\_t1 = nli.resample\_to\_img(t1, img)

plotting.plot\_anat(resampled\_t1)

```

#### \*\*Smoothing Images\*\*

- \*\*Smoothing\*\*: The mean image is smoothed with different full-width half maximum (FWHM) values using `nli.smooth\_img()`.

```python

for fwhm in range(1, 12, 5):

smoothed\_img = nli.smooth\_img(img, fwhm)

plotting.plot\_epi(smoothed\_img, title=f"Smoothing {fwhm}mm")

```

#### \*\*Cleaning Images\*\*

- \*\*Cleaning Functional Images\*\*: The functional BOLD image is cleaned by detrending and standardizing it using `nli.clean\_img()`. Motion parameters can also be removed as confounds.

```python

func\_ds\_c = nli.clean\_img(bold, detrend=True, standardize=True, t\_r=2.5,

confounds='data/sub-01\_ses-test\_task-fingerfootlips\_bold\_mcf.par')

```

#### \*\*Masking and Signal Extraction\*\*

- A mask is created by thresholding the mean image and keeping only clusters larger than 1000 mm³. The average signal from the masked regions is extracted.

```python

mask = nli.math\_img('np.mean(img,axis=3) > 0', img=cluster)

all\_timecourses = apply\_mask(bold, mask)

mean\_timecourse = all\_timecourses.mean(axis=1)

plt.plot(mean\_timecourse)

```

### \*\*3. Independent Component Analysis (ICA)\*\*

- \*\*CanICA\*\*: Independent Component Analysis (ICA) is performed on the BOLD data using the `CanICA` module. This extracts independent components representing brain networks.

```python

from nilearn.decomposition import CanICA

canica = CanICA(n\_components=5, smoothing\_fwhm=6., standardize=True)

canica.fit(bold)

components\_img = canica.masker\_.inverse\_transform(canica.components\_)

```

- The ICA components are visualized on the T1 anatomical image.

```python

from nilearn.image import iter\_img

for i, cur\_img in enumerate(iter\_img(components\_img)):

plot\_stat\_map(cur\_img, bg\_img=t1, title=f"IC {i}")

```

### \*\*4. Dictionary Learning\*\*

- \*\*DictLearning\*\*: Similar to ICA but with better stability and sparser maps. Dictionary learning is applied to extract meaningful temporal elements from the fMRI data.

```python

from nilearn.decomposition import DictLearning

dict\_learning = DictLearning(n\_components=5, alpha=1., smoothing\_fwhm=6.)

dict\_learning.fit(bold)

components\_img = dict\_learning.masker\_.inverse\_transform(dict\_learning.components\_)

```

### \*\*5. Image Visualization with Nilearn\*\*

#### \*\*Glass Brain Visualization\*\*

- A glass brain plot shows significant voxels overlaid on a transparent MNI brain template.

```python

plotting.plot\_glass\_brain(localizer\_tmap, threshold=3, colorbar=True)

```

#### \*\*Overlay Functional Image onto Anatomical Image\*\*

- Functional images can be overlaid onto anatomical images with customizable cut coordinates.

```python

plotting.plot\_stat\_map(localizer\_tmap, display\_mode='z', cut\_coords=5, threshold=2)

```

#### \*\*3D Surface Plot\*\*

- A statistical map is projected onto a cortical mesh for surface-based visualization using `vol\_to\_surf` and `plot\_surf\_stat\_map`.

```python

texture = surface.vol\_to\_surf(localizer\_tmap, fsaverage['pial\_right'])

plotting.plot\_surf\_stat\_map(fsaverage['infl\_right'], texture)

```

### \*\*Conclusion\*\*

This notebook demonstrates how to use Nilearn for:

- Neuroimaging data manipulation (resampling, smoothing, cleaning).

- Advanced techniques like ICA and Dictionary Learning for extracting brain networks.

- Visualization techniques such as glass brain plots and 3D surface projections.