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 mm^3. 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.