Presentation of the proposed pipelines for fMRI preprocessing, multi-view representation learning, and regression task

May 28, 2020

1 Proposed Pipeline for resting-state fMRI preprocessing

The main goal of this pipeline is to process resting-state fMRI (rsfMRI) data, in order to extract a set of features vectors, which can be represented as a correlation matrix between the voxels V_i and regions of interest (ROIs). The rsfMRI are available in the following directory:

`https://openneuro.org/datasets/ds001771/versions/1.0.2`

These data contain 40 subjects: sub-03, ...sub-35, sub-37,...sub-42.
We used SPM software to process rsfMRI.
All script batch and the source code are available in the following links:

- Task-fmri preprocessing: `https://github.com/sellamiakrem/prep_tfmri`
- Resting-state fMRI preprocessing: `https://github.com/sellamiakrem/prep_rsfmri`
- Deep multi view representation learning: `https://github.com/sellamiakrem/deep-multi-view-representation-learning`
- Trace regression: `https://github.com/sellamiakrem/trace_regression`

In fact, in order to execute it, there are two possibilities: interactive or no interactive mode of Frioul:

- Interactive mode: it aims to process all subjects in a sequential processing, i.e. subject per subject. In order to execute the program, the following syntax it should be used:
  python preprocessing_rsfmri sub-xx, e.g.: python preprocessing_rsfmri 3 4 5 6...42

- No Interactive mode: it seeks to process all subjects at the same time (parallel processing). Therefore, the following syntax can be used:

/hpc/soft/anaconda3/bin/python batch_frioul.py

The main goal of the proposed pipeline is to process all rsfMRI data, in order to compute the correlation matrix between all voxels and ROIs. Therefore, as input, we have a bold images for each subject S_i, and we apply a batch script on SPM using Matlab software to get a corrected image in Gifti format. Then, we compute the correlation matrix between voxels and ROIs. Therefore, Figure 2 reports different steps of the proposed pipeline for rsfMRI preprocessing. For this pipeline, we have used SPM on MATLAB and Python softwares. The

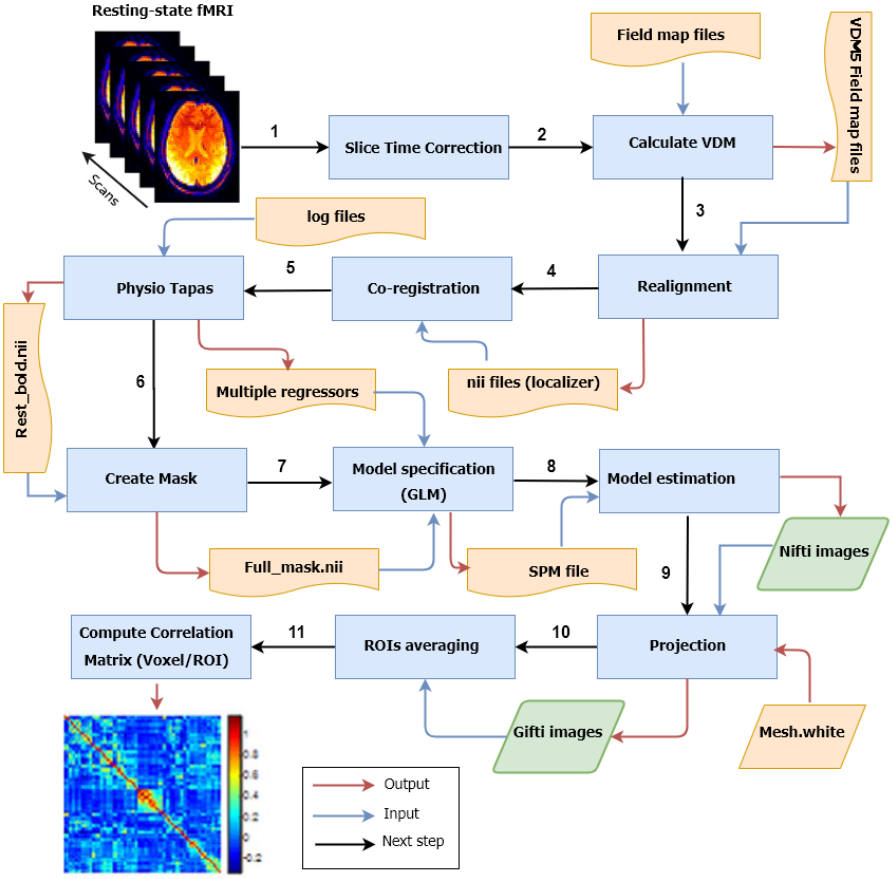

Figure 1: The proposed pipeline for resting state fMRI preprocessing

proposed batch script contains 8 steps: Slice time correction, calculate VDM,

realignment, co-registration, Physio-tapas, model specification, and model estimation. For the rest of steps: create mask, projection, ROIs averaging, and correlation matrix, we used Python software. Figure, shows the organization

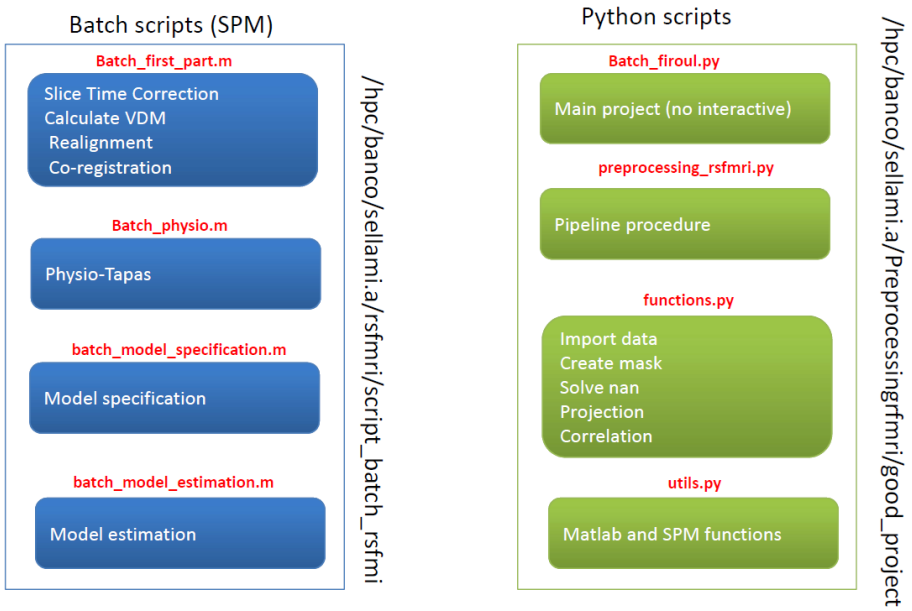

Figure 2: Description of source code files developed on SPM and Python softwares

2 Description of different steps of the proposed pipeline

2.1 Step 1: Slice Time Correction (STC)

Slice Time Correction (STC) generally improves our the statistical power of resting fMRI analyses. It aims to reduce the time gap between different slices. Table 1 shows the parameters settings for STC. To get an information about the echo times and time slicing, you can see the header files (*.json, *.txt).

2.2 Step 2: Realignment

The realignment is also known as 'motion correction'. The aim is to remove movement artifact in rsfMRI time-series. SPM does this by realigning a time-series of images acquired from the same subject using a least squares approach and a different parameters spatial transformation.

Number of slice	60
TR	0.955
TA	0.939
Slice timing	0.875 0.7975 0.7175 0.6375 0.5575 0.4775 0.3975 0.32 0.24 0.16 0.08 0 ($\times$ 5)
Slice order	1 $\times$ 60

Table 1: Parameters settings of STC

3 Step 3: Coregistration

A Process to overlay structural and functional images in a way that maximizes the mutual information. The mean functional image is coregistered to a high resolution anatomical image, and all of the other functional images are then resliced to align with the reference image.

3.1 Step 4: Physio-Tapas Toolbox

The general purpose of this Matlab toolbox is the model-based physiological noise correction of resting fMRI data using peripheral measures of respiration and cardiac pulsation.

3.2 Model specification (General linear Model)

General linear models (GLMs) are the most often used class of supervised methods for finding SPM of neural activation. For a given data set the target variable $y \in R^L$ (e.g. a single voxel time course of length L) is modeled as a linear combination of N given regressor time series x_i, each weighted by a coefficient stored in a vector $\beta \in R^N$, plus some gaussian, i.e., error $\epsilon \simeq (0, 1)$

$$\hat{y} = X\beta + \epsilon \tag{1}$$

where the $L \times N$ matrix $X = [x_1 x_2 ... x_N]$ contains the N regressors x_i of length L as column vectors and is called design matrix. A typical GLM analysis includes as regressors all experimentally controlled parameters and additionally so called nuisance regressors, which are not of interest in the analysis but explain considerable amounts of variance in the data.

3.3 Projection of nii files into gii files

In this step we use the following code to produce the gii files in LH and RH spaces: Example for sub-40 :

In order to visualize the obtained gifti files, we can use Anatomist software to combine the mesh lh.white or rh.white with the gifti file ($ctrl + f$). However, it's necessary to convert the mesh of surfer format to gifti format as follows :

```
$FREESURFER_HOME/bin/mri_vol2surf --src {} --o {}_rh.gii --out_type gii --regheader sub-40 --hemi /rh --projfrac-avg 0 1
0.1 --surf-fwhm 0 --sd /hpc/banco/cagna.b/my_intertva/surf/data/sub-40/fs --trgsubject sub-40 \;

$FREESURFER_HOME/bin/mri_vol2surf --src {} --o {}_lh.gii --out_type gii --regheader sub-40 --hemi /lh --projfrac-avg 0 1
0.1 --surf-fwhm 0 --sd /hpc/banco/cagna.b/my_intertva/surf/data/sub-40/fs --trgsubject sub-40 \;

freesurfer_setup

mris_convert /hpc/banco/cagna.b/my_intertva/surf/data/sub-40/fs/sub-40/surf/lh.white
/hpc/banco/sellami.a/InterTVA/rsfmri/sub-40/glm/noisefiltering/lh.white.gii

mris_convert /hpc/banco/cagna.b/my_intertva/surf/data/sub-40/fs/sub-40/surf/rh.white
/hpc/banco/sellami.a/InterTVA/rsfmri/sub-40/glm/noisefiltering/rh.white.gii
```

3.4 Correlation Matrix (V_i, ROI)

This step aims to compute the correlation matrix between two time series of voxels and ROIs. In fact, we have a set of ROIs obtained from annotations files, and a set of gii files extracted by the projection operation. Therefore, we compute the average matrix of All ROIS, and the correlation between the voxel and the mean of each ROI using a correlation coefficient such as Pearson coefficient or mutual information criterion.
All script batch and the source code are available in the following directory:

`https://github.com/sellamiakrem/prep_rsfmri`

The main python file is *preprocessing_tfmri.py*. The only parameter to give to this script is the subject ID. For example: *python3 preprocessing_rsfmri.py 1 2 10 15*. As output, you get a correalation matrix for each subject, which contains the perason correlation between each ROI and each vertex v_i.

4 Task-fMRI preprocessing

All task-fMRI data are preprocessed by Banco team. So, here our goal is to construct a activation matrix using gifti files of task-fMRI. matrix as input, we have lh and rh gifti files, e.g., *beta_0001.lh_fsaverage5.gii* and *beta_0001.rh_fsaverage5.gii*

All script batch and the source code are available in the following directory:

`https://github.com/sellamiakrem/prep_tfmri`

The main python file is *preprocessing_tfmri.py*. The only parameter to give to this script is the subject ID. For example: *python3 preprocessing_tfmri.py 1 2 10 15*. As output, you get a matrix for each subject, which contains the beta values β_j of each vertex v_i.

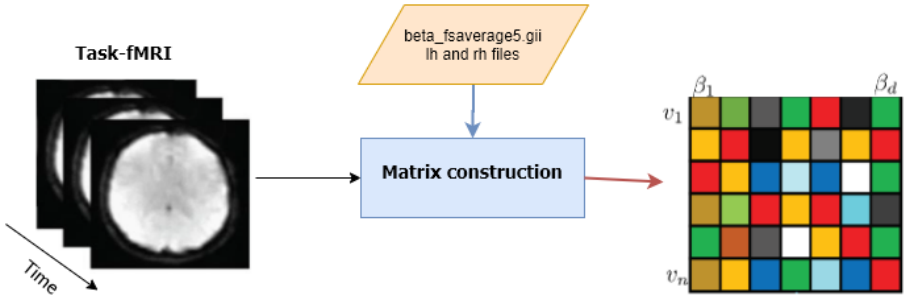

Figure 3: The proposed pipeline for task-fMRI preprocessing

5 Multi-view representation learning

This section presents the experimental protocol for the comparative study on representation learning. The main goal is to apply different dimensionality reduction techniques (PCA, ICA, Deep CCA, and Deep Autoencoder), on fMRI data sets, i.e., on task-fMRI (view 1), resting-state fMRI (view 2), and combined views (view 1 + view 2). Fig. 4 reports the main phases of the experimental protocol for the comparative study on dimensionality reduction methods. The

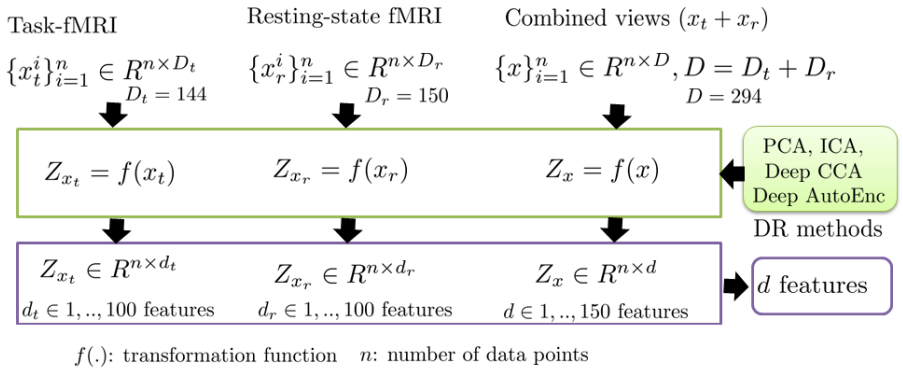

Figure 4: Experimental protocol for the comparative study on dimensionality reduction methods

main steps of the proposed protocol are:

1. **Data preprocessing**: The fMRI datasets, will be normalized with a MinMaxScaler to the range [0,1].

2. **Dimensionality reduction:** reduce the dimensionality of inputs (task-fMRI and resting-state fMRI), by extracting useful features of each vertex. We used PCA, ICA, Deep CCA, and Deep Autoencoder to perform dimensionality reduction. Formally, the dimensionality reduction step can

be defined as

$$Z = f(X) \tag{2}$$

where $Z \in \mathbb{R}^{n \times d}$ is the reduced data, d is the number of extracted features ($d \in 1, ..., 100$ features for task-fMRI and resting-state fMRI, and $d \in 1, .., 150$ for combined views, f is a transformation function, and $X \in \mathbb{R}^{n \times D}$ the input matrix.

The proposed pipeline is described in the figure .5. It contains two main steps:

1. **Intialization:** set all hyperparameters:
   - *Learning rate:* Learning rate controls how quickly or slowly a neural network model learns a problem. The default learning rate is 0.01 and no momentum is used by default.
   - *Epochs:* One Epoch is when an ENTIRE dataset is passed forward and backward through the neural network only ONCE The number of epoch in our case is of 300.
   - *Batch size:* Total number of training examples present in a single batch. Here, it equals to 500.
   - *Activation functions:* it determines the output of a deep neural network. For all the network models, we tested different pairs of activation functions for the hidden layers and output layer, respectively: (*linear, linear*), (*linear, sigmoid*), (*relu, linear*), and (*relu, sigmoid*).
   - *Number of layers:* we used 1 to 3 hidden layers for each model.
2. **Dimensionality reduction:** aims to reduce fMRI data and return latent space of the combined fMRIdata.

All script batch and the source code are available in the following directory:

`https://github.com/sellamiakrem/deep-multi-view-representation-learning`

The main python file are: *mdae_concat_three_layers.py*, *mdae_shared_three_layers.py*. As output, you get a trained model, which allow to reduce the dimensionality (features) of each subject.

6 Regression task

In this experiment, we will consider the trace regression model to predict the score of behavior $\hat{y}$. Fig. 6 shows the main phases of the proposed experimental protocol. Furthermore, the main steps can be summarized as follows:

1. The trace regression model will be estimated on the learned representation $Z\mathbb{R}^{n \times d}$. Moreover, we will compare the obtained results on the raw fMRI data X, i.e., without dimensionality reduction. A 10-fold cross validation will be implemented in order to assess the effectiveness of the designed model.

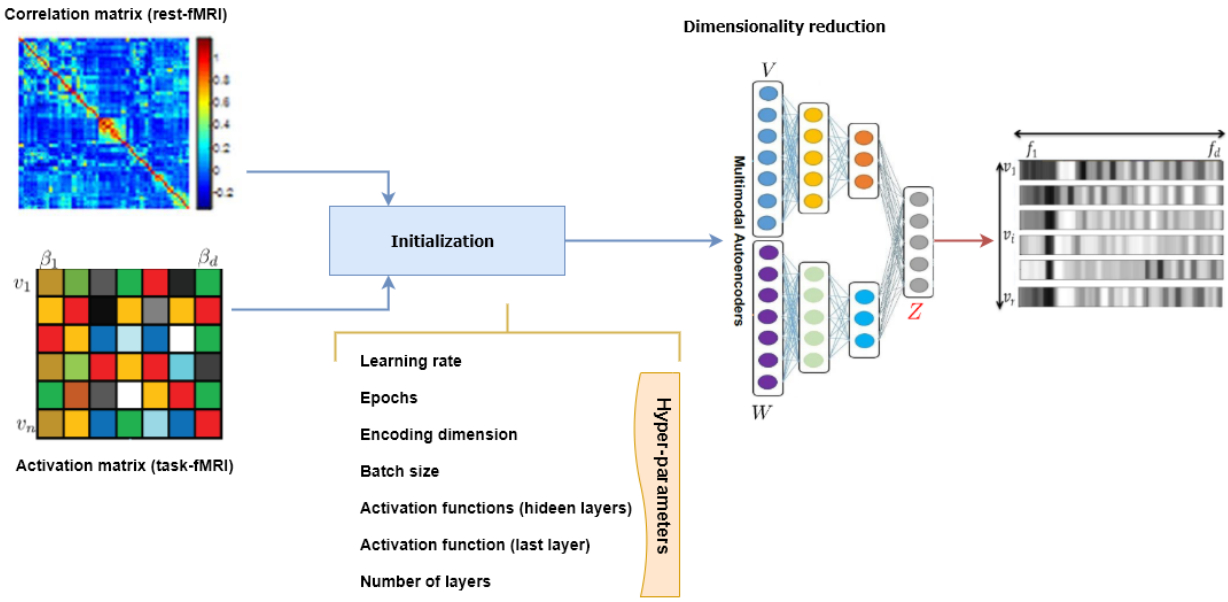

Figure 5: The proposed pipeline of multi-view representation learning

2. The trace model regression will be evaluated using manifold and parsimonious regularization, and without regularization.

3. Make a quantitative of the predictive model using MSE, MAE, and R^2.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{3}$$

$$R^2 = 1 - \frac{MSE(model)}{MSE(baseline)} = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y}_i)^2} \tag{4}$$

where N is the number of subjects, y is the true values (score of behavior), $\hat{y}$ is the predicted values, and $\bar{y}$ is the mean of the true values.

4. Make and qualitative evaluation of the obtained weight maps β^*

All scripts are available at the link: `https://github.com/sellamiakrem/trace_regression`. The main program is called *trace_regression.py*, which aims to predict the score of behavior using the latent representation Z_i.

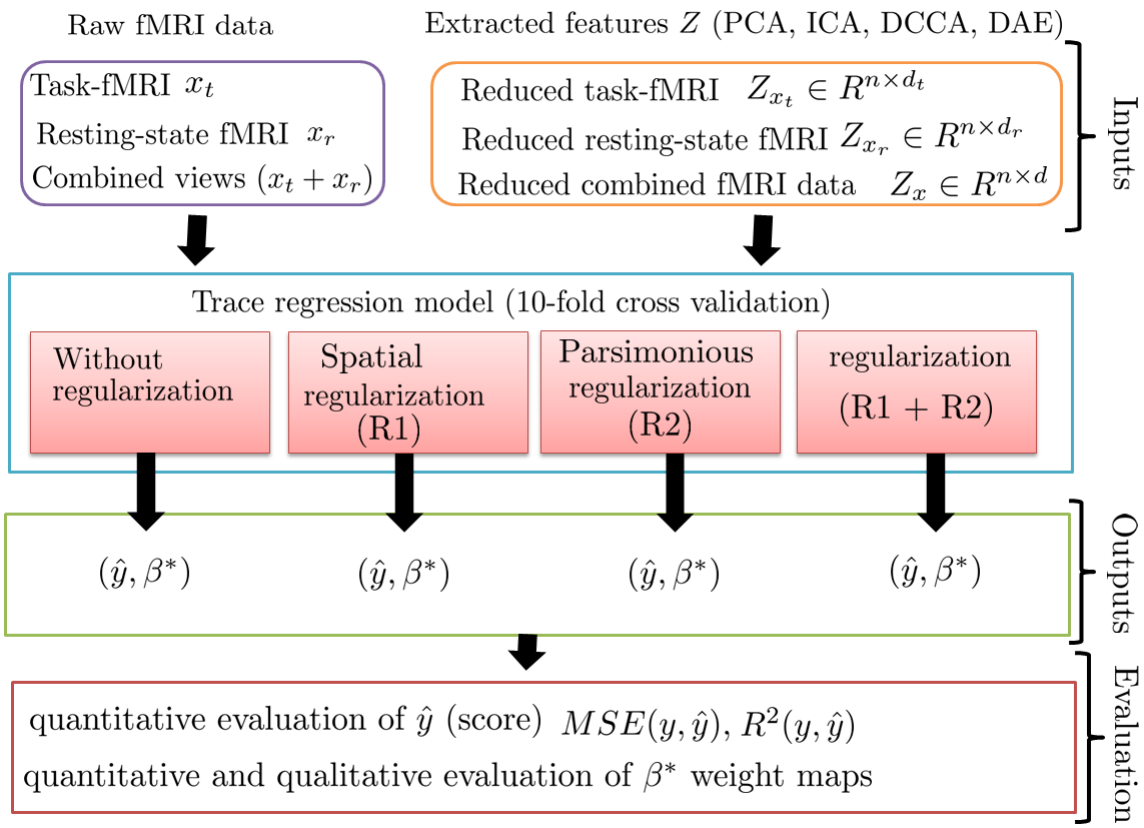

Figure 6: Experimental protocol for the predictive model

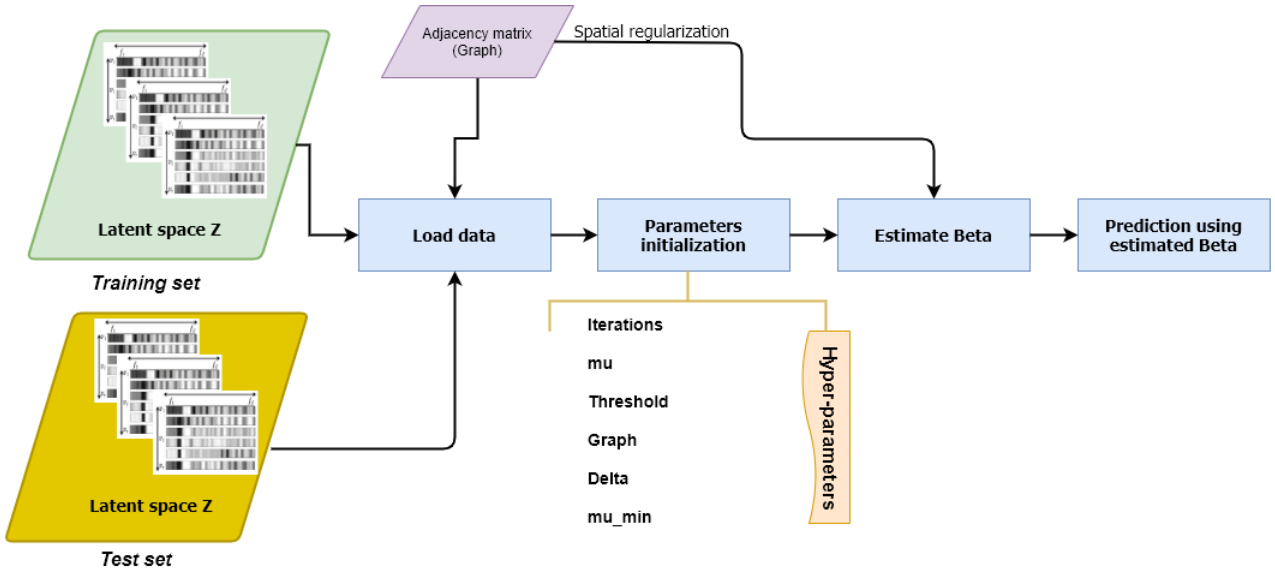

Figure 7: The proposed pipeline for the regression problem