FUNCTIONAL NEURAL CONNECTIVITY ANALYSIS

FOR SCHIZOPHRENIC PATIENTS

WITH THE DEEP AUTOENCODER

AND

VARIATIONAL AUTOENCODER

Authored by:

OMID REZANIA

Table of contents:

Introduction:

Functional Neural Connectivity: Understanding Brain Dynamics and Implications in Schizophrenia

Functional neural connectivity explores the temporal correlations between spatially distant neurophysiological events, revealing how different brain regions interact, coactivate and cooperate to support various cognitive functions and behaviors (Friston, 2011; Bassett & Sporns, 2017). This field has revolutionized neuroscience by providing insights into both normal brain function and disorders such as schizophrenia, where disruptions in connectivity play a crucial role (Fornito et al., 2012; Pettersson-Yeo et al., 2011). Functional neural connectivity represents a crucial aspect of brain function, revealing how neural circuits organize and synchronize to support cognition, behavior, and disease pathology. Advances in imaging technology and analytical methods continue to deepen our understanding of brain networks and their role in health and disease. Schizophrenia is characterized by disruptions in thought processes, perceptions, emotions, and behaviors. Research into functional neural connectivity in schizophrenia has revealed significant alterations in how different brain regions communicate, shedding light on the neurobiological underpinnings of the disorder.

Advances in imaging technology and analytical methods continue to deepen our understanding of brain networks and their role in health and disease.

Functional MRI (fMRI) is pivotal in studying functional neural connectivity. It measures changes in blood flow and oxygenation, offering insights into brain activity at high spatial resolution (Biswal et al., 1995). Resting-state fMRI (rs-fMRI) captures spontaneous fluctuations in the blood oxygen level-dependent (BOLD) signal, revealing intrinsic functional networks like the default mode network (DMN) that are crucial for understanding brain organization (Raichle, 2015; Smith et al., 2009).

The Default Mode Network (DMN) is a prominent and well-studied functional brain network that is active when an individual is not focused on the external environment or engaged in specific tasks. Instead, it becomes active during introspective thoughts, daydreaming, spontaneous cognition, and self-referential processing. The term "default mode" arose because initially, it was observed that this network was most active during rest or when the brain was not engaged in goal-directed tasks.

Resting-state functional MRI (rs-fMRI) has been pivotal in mapping and studying the DMN. It identifies synchronized fluctuations in the BOLD (Blood Oxygen Level Dependent) signal across DMN regions, revealing its functional connectivity dynamics.

Alterations in DMN activity and connectivity have been implicated in various psychiatric and neurological conditions. For example, disruptions in the DMN are observed in disorders such as Alzheimer's disease, autism spectrum disorders, and schizophrenia (Buckner et al., 2008; Whitfield-Gabrieli & Ford, 2012).

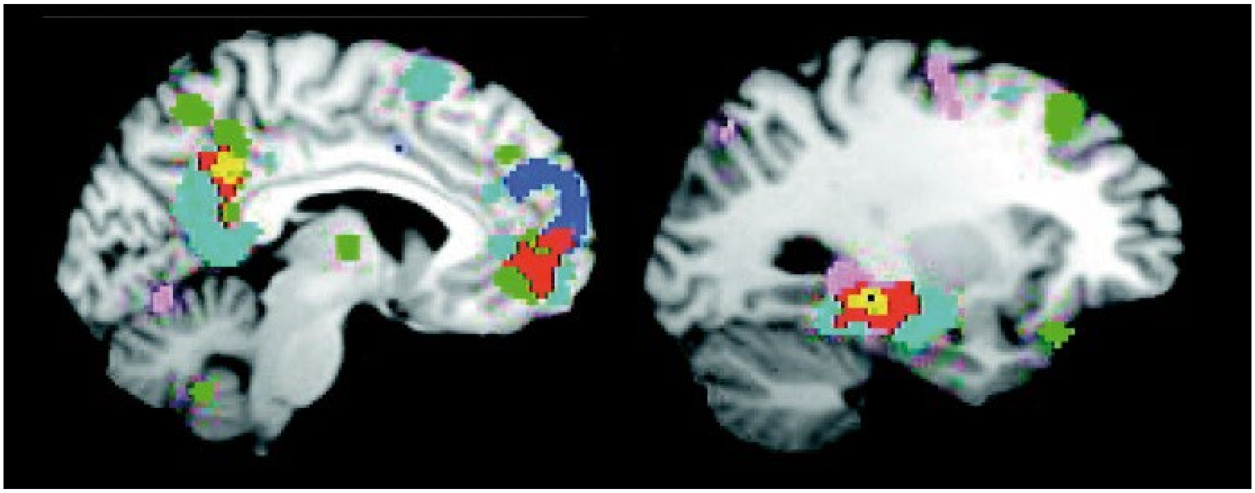


Figure 1 – The fMRI scan delineating the DMN of someone remembering, thinking about the future. The coloured regions represent the Medial Prefrontal Cortex(mPFC), Posterior Cingulate Cortex (PCC) and Inferior Parietal Lobule (IPL). (Buckner et al.,2013)

Extensive research has considered whether disruption of the default network may contribute to disease. A recent study from Whitfield-Gabrieli and colleagues35 found that patients with schizophrenia display a hyperactive default network and aberrant connectivity of the default network. (Buckner et al.,2013)

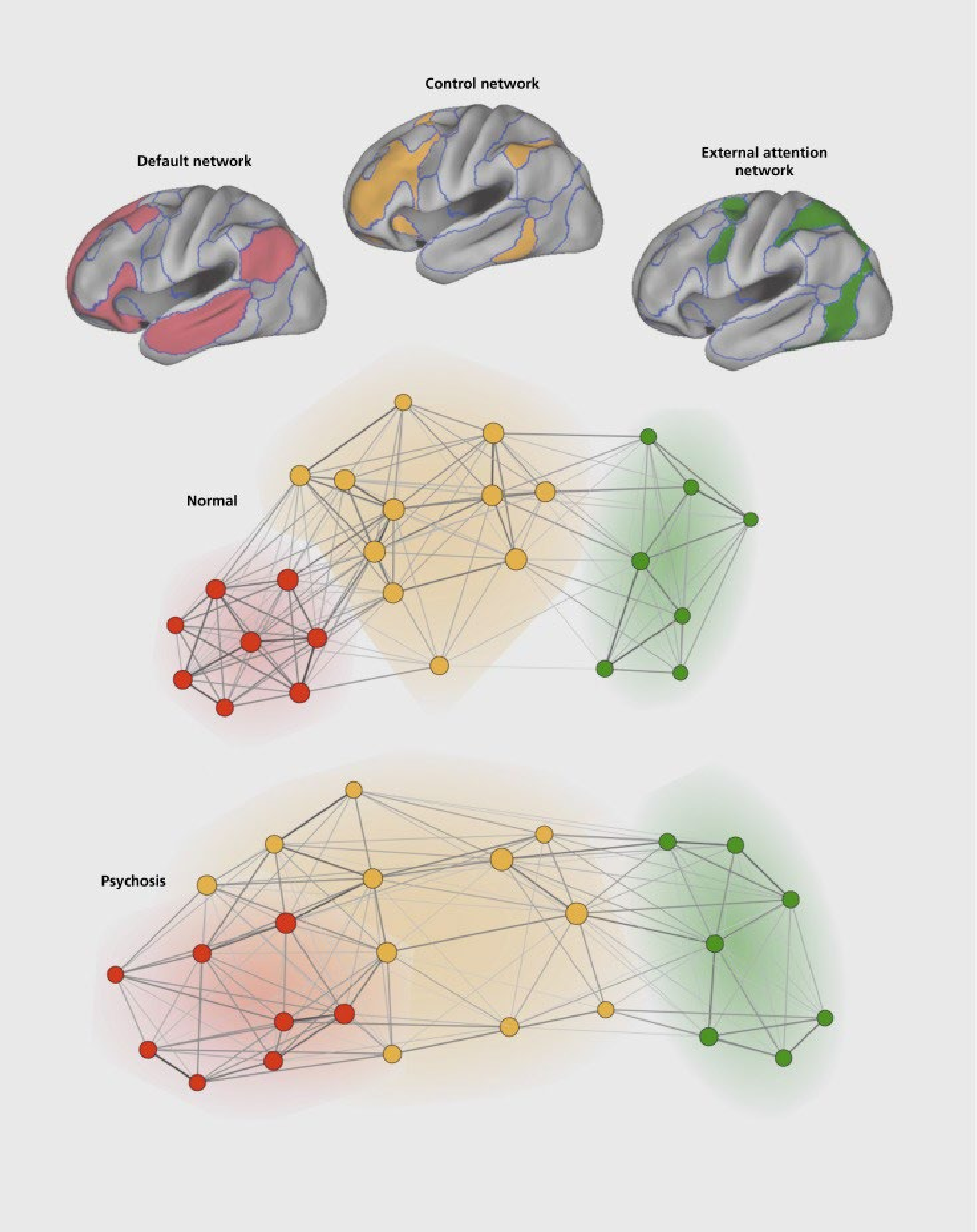


Figure 2 – The functional connectivity and the DMN of the normal network and the psychosis during the schizophrenia manifestation

As the figure 2 above demonstrates brain function and dysfunction can be examined by exploring how different brain networks interact. The top panel illustrates three networks that include the default network (red), a frontoparietal control network thought to be important to executive control (orange), and an external attention network hypothesized to guide attention and actions toward external sensory stimuli, often called the dorsal attention network (green). In the middle row, the circles and lines represent a graphical representation of the relationship of the brain regions involved in all three networks. Each circle is a brain region and the line thickness between the circles represents the functional correlation strength between the regions. Regions that are strongly functionally coupled are plotted near to one another. What emerges in normal control subjects is that regions within each network are tightly functionally coupled and distinct from the regions of the other networks. One hypothesis is that the frontoparietal control network regulates the interactions between the default network and external attention network. When the same analysis is applied to psychotic patients (n=100 including schizophrenia, schizoaffective disorder, and bipolar disorder with psychosis), the network interactions display an interesting difference: the frontoparietal control network shows a less modular structure and a less rigid boundary with the default network. (Buckner et al.,2013)

Besides from the dominance of the fMRI technique to track the functional neural connectivity Electroencephalography (EEG) and Magnetoencephalography (MEG) provide direct measurements of neural electrical activity with high temporal resolution, capturing rapid changes in functional connectivity during different cognitive states (Hillebrand & Barnes, 2002; Gross, 2019). These methods complement fMRI by offering insights into neural oscillations and synchronization across brain regions.

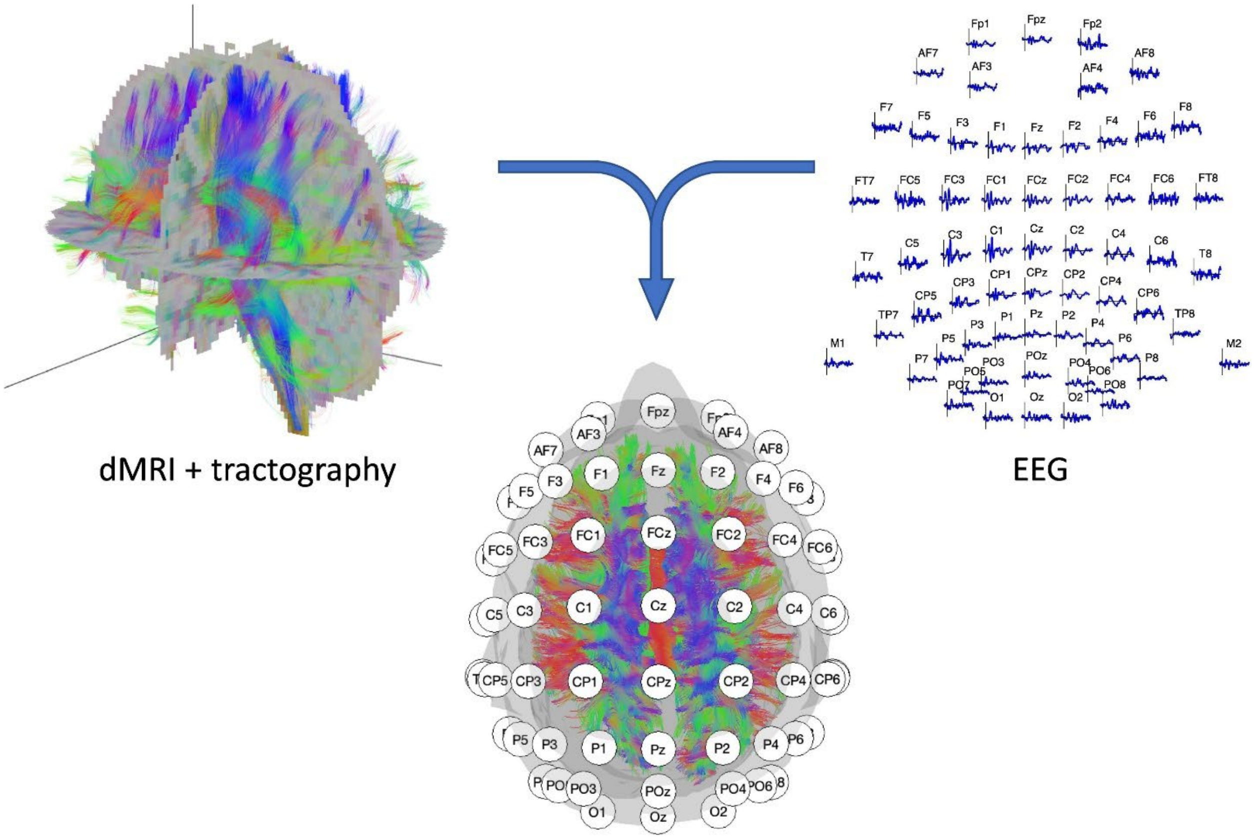


Figure 3 - Illustration of the two recording modalities. (Left top panel) dMRI recording with resulting fiber tracts. (Right top panel) 64-channel EEG signal with electrode layout conform the 10–20 systematic. (Middle lower panel) EEG/dMRI combination to explore the link between structural and functional connectivity. (Babaeeghazvini et al.,2021)

In order to understand and decipher the neural connectivity graph theory and network analysis are fundamental tools for quantifying and interpreting functional connectivity patterns (Bullmore & Sporns, 2009). These approaches treat brain regions as nodes and functional connections as edges, revealing the brain's complex organization in terms of network properties such as modularity and efficiency (Rubinov & Sporns, 2010).

Studies using fMRI and other imaging techniques have identified aberrant functional connectivity patterns in schizophrenia patients compared to healthy controls. These alterations involve disruptions in local and long-range connectivity, affecting networks such as the default mode network (DMN), front-parietal network, and salience network (van den Heuvel & Fornito, 2014; Whitfield-Gabrieli & Ford, 2012).

Default Mode Network (DMN): Reduced anti-correlation between the DMN and task-positive networks suggests a failure to segregate internal mental processes from external stimuli, contributing to cognitive deficits in schizophrenia (Whitfield-Gabrieli et al., 2009).

Fronto-Parietal Network: Dysfunctional connectivity within this network is associated with impairments in cognitive control and working memory, core features of schizophrenia (Repovs & Barch, 2012).Altered functional connectivity in schizophrenia correlates with symptom severity and cognitive impairments. Deficits in social cognition and auditory hallucinations are linked to disrupted connectivity within the auditory and language processing networks (Woodward et al., 2011; Shinn et al., 2013). Additionally, impaired connectivity between the prefrontal cortex and limbic system contributes to emotional dysregulation and negative symptoms (Anticevic et al., 2015). Longitudinal studies have highlighted developmental trajectories of functional connectivity abnormalities in schizophrenia. Early-stage schizophrenia shows subtle connectivity alterations that become more pronounced with disease progression (Alexander-Bloch et al., 2013). These findings underscore the dynamic nature of neural connectivity disruptions and their implications for early intervention strategies.

Advances in connectivity-based biomarkers hold promise for improving diagnostic accuracy and predicting treatment outcomes in schizophrenia (Crossley et al., 2017). Targeted interventions such as neuromodulation techniques (e.g., TMS) and cognitive remediation therapies aim to normalize aberrant connectivity patterns and alleviate symptoms (Lefaucheur et al., 2017; Wykes et al., 2011).

Data Acquisition Summary

The original study (Elena et al.,2012) where the data was adopted utilized resting-state scans from 405 healthy participants, with an average age of 21.0 years (ranging from 12 to 35 years) and a majority of 200 female participants. These scans were gathered from 34 different studies conducted by 18 principal investigators at the Mind Research Network. Prior to analysis, all participants provided informed consent in accordance with University of New Mexico guidelines, and their data were anonymized. This dataset represents a subset of a larger cohort used in a previous study (Allen et al., 2011), with stricter inclusion criteria applied to minimize the impact of motion and spatial normalization issues, and to enhance sample homogeneity by excluding participants over 35 years old.

Imaging was performed using a 3-T Siemens Trio scanner equipped with a 12-channel radio frequency coil. Functional images were acquired using a T2*-weighted gradient-echo EPI sequence with specific parameters: TE = 29 ms, TR = 2 s, flip angle = 75°, slice thickness = 3.5 mm, slice gap = 1.05 mm, field of view = 240 mm, matrix size = 64 × 64, and voxel size = 3.75 mm × 3.75 mm × 4.55 mm. Each resting-state scan lasted at least 5 minutes and 4 seconds (152 volumes), with any excess volumes discarded to ensure consistency across participants. During the scans, participants were instructed to keep their eyes open and focus on a centrally presented cross.

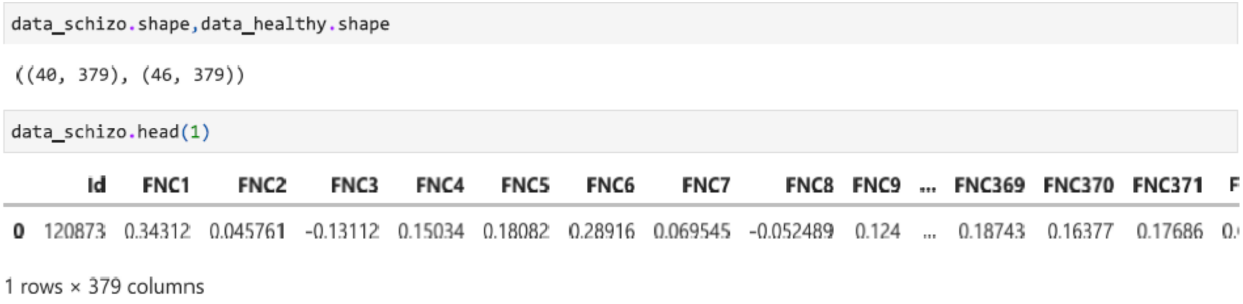
Data preprocessing was conducted using an automated pipeline centered around SPM 5 software. This included removing the first 4 image volumes to account for T1 equilibration effects, realigning images to correct for motion using INRIalign, correcting for slice timing using the middle slice as a reference, normalizing images into Montreal Neurological Institute space, reslicing to 3 mm × 3 mm × 3 mm voxels, and applying Gaussian smoothing (FWHM = 5 mm). Voxel time series were z-scored to standardize variance across space, minimizing potential biases in subsequent data analysis steps. The choice of variance normalization aimed to prioritize temporal modulation over amplitude differences, aligning with the study's research focus.

Following preprocessing, data underwent group-level spatial Independent Component Analysis (ICA) using the GIFT toolbox. A high model order (C = 100 components) was selected to achieve a detailed functional parcellation of cortical and subcortical components, aligned with established anatomical and functional segmentations. Subject-specific data reduction via Principal Component Analysis (PCA) retained 120 principal components, while group data reduction retained C = 100 principal components using the EM algorithm to manage computational demands.

To identify intrinsic connectivity networks (ICNs) from the ICA results, component time courses underwent further postprocessing. This involved removing residual noise sources such as low-frequency scanner drifts, motion-related variance not captured in distinct components due to spatial nonstationarity, and other nonspecific noise artifacts. Postprocessing techniques included detrending to remove linear, quadratic, and cubic trends, regression of motion parameters and their derivatives, outlier detection and removal, and low pass filtering with a cutoff frequency of 0.15 Hz.

Data Preprocessing Before training

The collected data was then classified and separated based on the diagnosis of schizophrenia. Two sets of data with 379 features (including the ID of the study participants) were prepared.



```
data_schizo.shape,data_healthy.shape
```

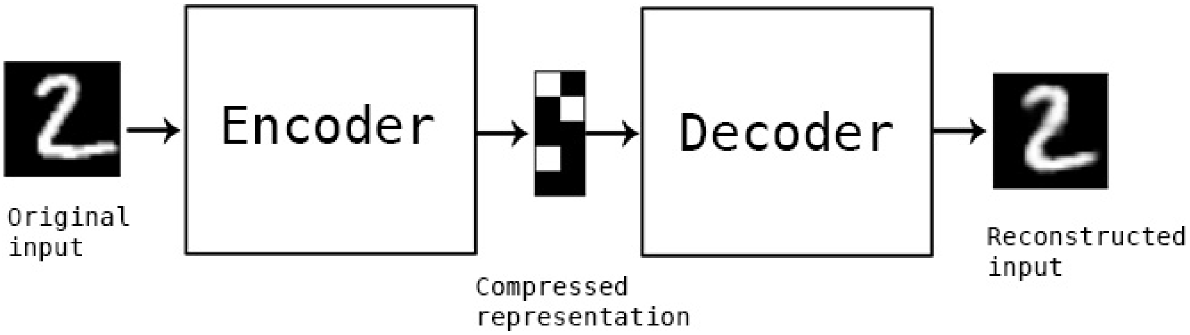
```
((40, 379), (46, 379))
```

```
data_schizo.head(1)
```

	Id	FNC1	FNC2	FNC3	FNC4	FNC5	FNC6	FNC7	FNC8	FNC9	...	FNC369	FNC370	FNC371	F
0	120873	0.34312	0.045761	-0.13112	0.15034	0.18082	0.28916	0.069545	-0.052489	0.124	...	0.18743	0.16377	0.17686	0.

1 rows × 379 columns

Autoencoder Philosophy



An autoencoder is a type of artificial neural network used for unsupervised learning of efficient data representations. It aims to learn a compressed, lower-dimensional representation of the input data, typically for the purpose of dimensionality reduction, denoising, or feature learning.

Key Components of an Autoencoder:

1.Encoder: The encoder transforms the input data into a compressed representation, often referred to as the latent space or encoding. It consists of one or more layers of neurons that progressively reduce the input into a lower-dimensional representation.

2.Decoder: The decoder reconstructs the input data from the compressed representation produced by the encoder. It mirrors the encoder architecture but in reverse, expanding the latent space representation back into the original dimensionality.

3. Objective: The primary objective of training an autoencoder is to minimize the reconstruction error between the input data and the output data reconstructed by the decoder. Commonly used loss functions include mean squared error (MSE) or binary cross-entropy, depending on the nature of the input data.

Types of Autoencoders:

-Basic Autoencoder: The standard autoencoder described above, which learns a general-purpose representation of the input data.

- Sparse Autoencoder: Introduces sparsity constraints to the hidden layers, encouraging the model to learn sparse representations, where most of the neurons are inactive.

- Variational Autoencoder (VAE): Extends the basic autoencoder to learn a probabilistic latent space. VAEs are trained to approximate the true posterior distribution of the latent variables, enabling generation of new data points like those in the training data.

- Denoising Autoencoder: Trains the model to reconstruct the original input from a corrupted or noisy version of the input. It encourages the autoencoder to learn robust features that are resilient to noise.

- Contractive Autoencoder: Adds a regularization term to the loss function that penalizes the derivative of the encoder's output with respect to its input. This encourages the model to learn stable representations that are insensitive to small variations in the input data.

Applications of Autoencoders:

- Dimensionality Reduction: Learning a compact representation of high-dimensional data while preserving important features.

- Image and Signal Compression: Efficiently compressing images or signals while retaining important information.

- Feature Learning: Discovering meaningful features or representations from raw data, which can then be used as inputs for other machine learning models.

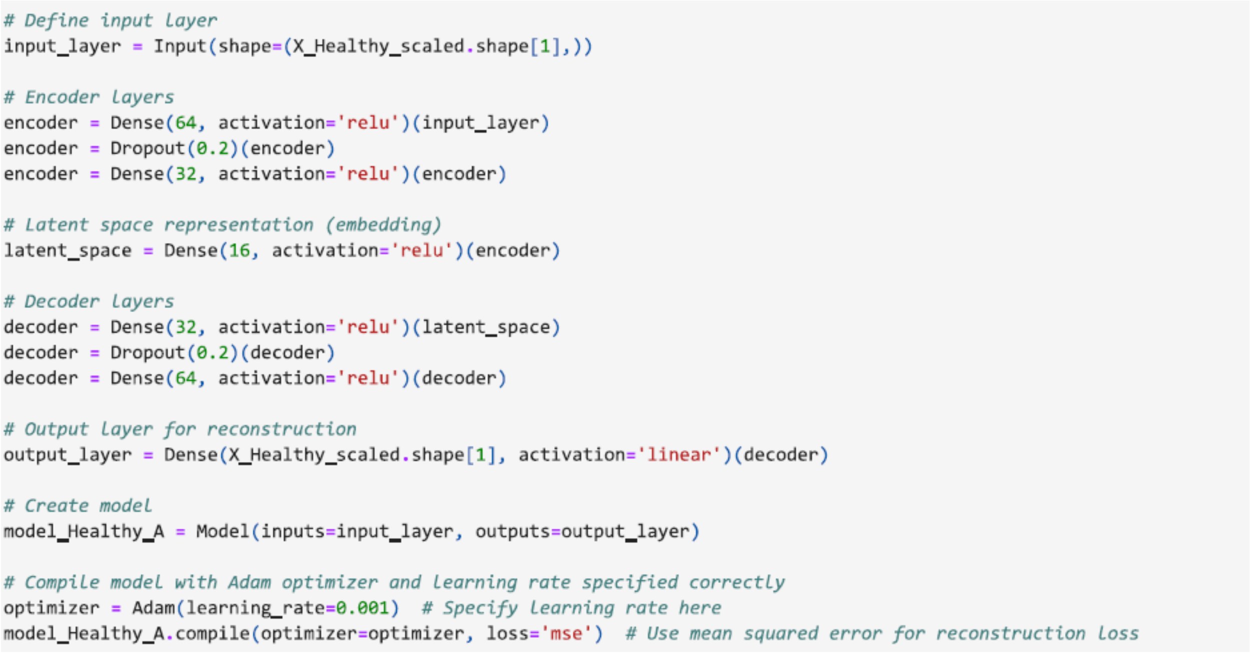
- Anomaly Detection: Autoencoders can reconstruct normal data well, making them effective for detecting anomalies or outliers that do not conform to learned patterns.

- Data Generation: Variational autoencoders can generate new data samples like those in the training data by sampling from the learned latent space.

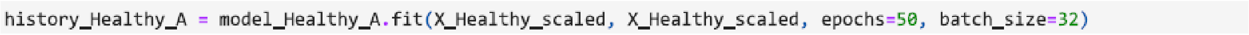
In Practice

Healthy Population Analysis

The initial model started as a deep autoencoder in which it used 3 layers as encoder and 3 symmetric layers as decoder along with the latent space and representation through one layer. Although the input layer dimension was the number of the FNC nodes as 378, the bottleneck of the latent space was induced through the compressed dimension of 16. Regularization to avoid overfitting was done through the “drop out” strategy.

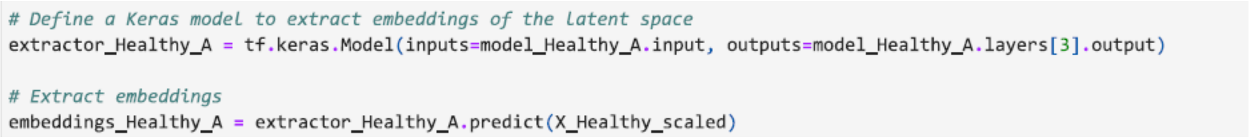


```
# Define input layer
input_layer = Input(shape=(X_Healthy_scaled.shape[1],))

# Encoder layers
encoder = Dense(64, activation='relu')(input_layer)
encoder = Dropout(0.2)(encoder)
encoder = Dense(32, activation='relu')(encoder)

# Latent space representation (embedding)
latent_space = Dense(16, activation='relu')(encoder)

# Decoder layers
decoder = Dense(32, activation='relu')(latent_space)
decoder = Dropout(0.2)(decoder)
decoder = Dense(64, activation='relu')(decoder)

# Output layer for reconstruction
output_layer = Dense(X_Healthy_scaled.shape[1], activation='linear')(decoder)

# Create model
model_Healthy_A = Model(inputs=input_layer, outputs=output_layer)

# Compile model with Adam optimizer and learning rate specified correctly
optimizer = Adam(learning_rate=0.001)  # Specify learning rate here
model_Healthy_A.compile(optimizer=optimizer, loss='mse')  # Use mean squared error for reconstruction loss
```



```
history_Healthy_A = model_Healthy_A.fit(X_Healthy_scaled, X_Healthy_scaled, epochs=50, batch_size=32)
```

The latent space of the model assumes to contain the most meaningful and relevant features of the functional connectivity. This latent representation of the input neural functional connectivity got extracted through embeddings.



```
# Define a Keras model to extract embeddings of the latent space
extractor_Healthy_A = tf.keras.Model(inputs=model_Healthy_A.input, outputs=model_Healthy_A.layers[3].output)

# Extract embeddings
embeddings_Healthy_A = extractor_Healthy_A.predict(X_Healthy_scaled)
```

Where in the extractor, the layer 3 represents the position of the hidden latent layer. The hierarchical unsupervised clustering on embeddings of the latent representation of the original dataset revealed 46 clusters.

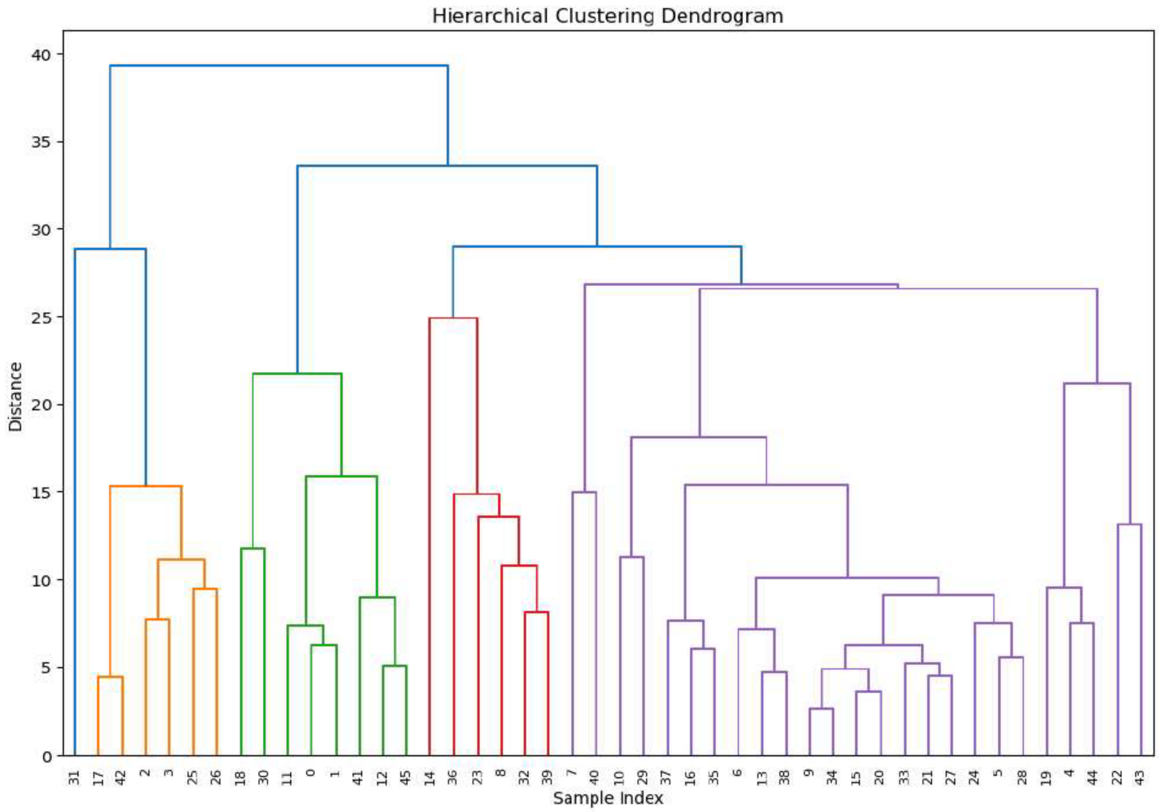


Figure 4 – The dendrogram and hierarchical clustering of the latent representation of the original functional neural connectivity. Total number of 46 clusters were identified.

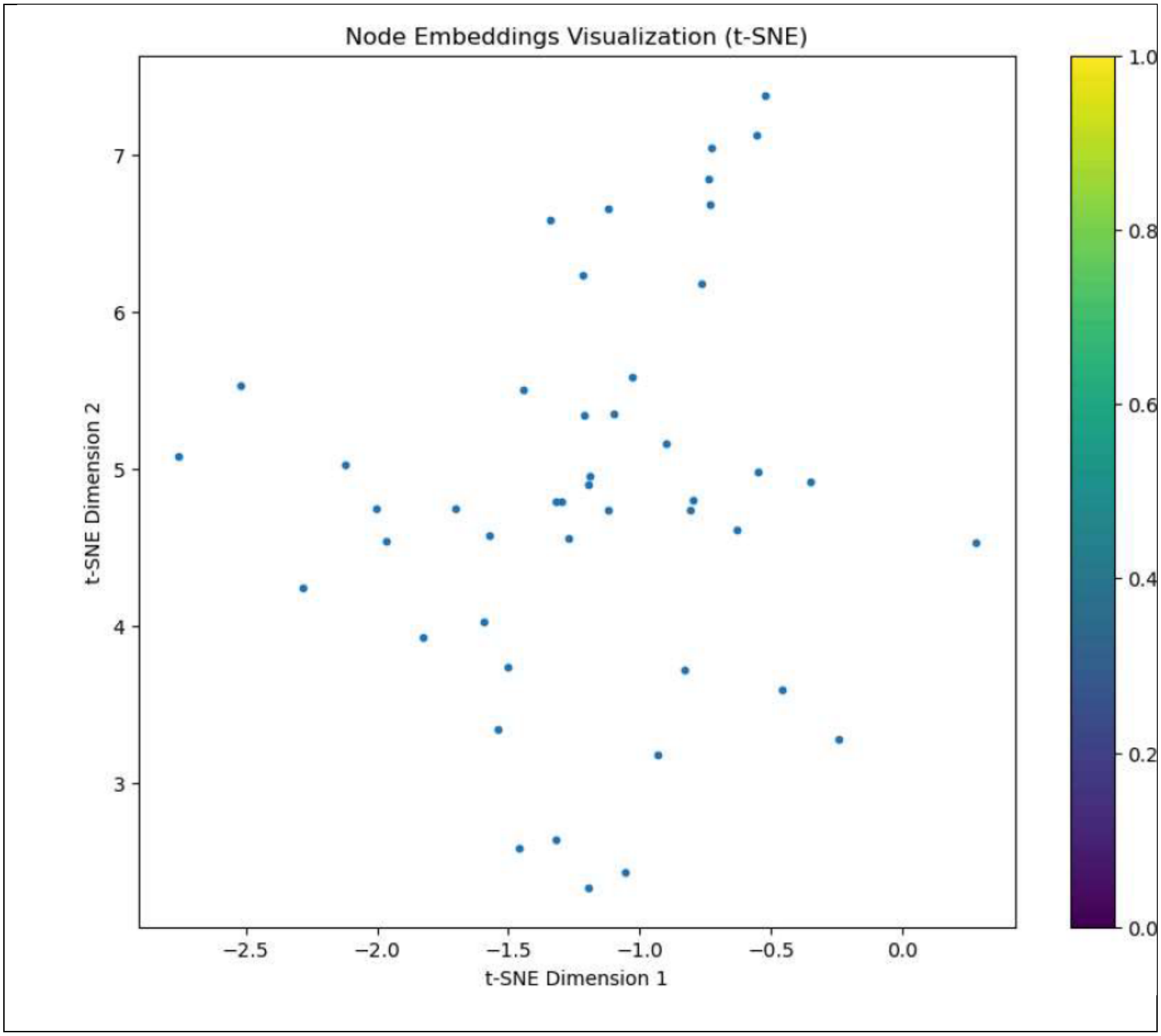
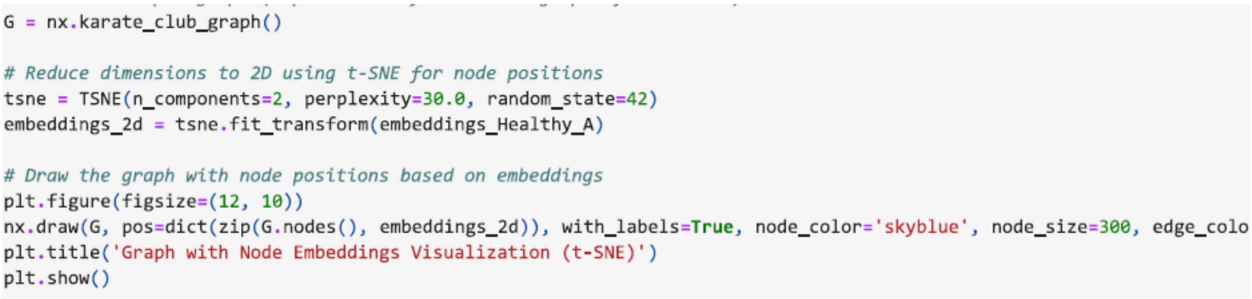


Figure 5 – The two-dimensional representation of the latent space by t-Distributed Stochastic Neighbour Embedding (t-SNE)

Applying the fundamental concepts of the graph theory, if each individual functional neural connectivity represents a node in the graph representation, the wedges inevitably will represent the connectivity and the way that each node are co-connected or coactivated during the resting state. This connectivity through wedges of a graph revealed the pattern as below:



```
G = nx.karate_club_graph()

# Reduce dimensions to 2D using t-SNE for node positions
tsne = TSNE(n_components=2, perplexity=30.0, random_state=42)
embeddings_2d = tsne.fit_transform(embeddings_Healthy_A)

# Draw the graph with node positions based on embeddings
plt.figure(figsize=(12, 10))
nx.draw(G, pos=dict(zip(G.nodes(), embeddings_2d)), with_labels=True, node_color='skyblue', node_size=300, edge_colo
plt.title('Graph with Node Embeddings Visualization (t-SNE)')
plt.show()
```

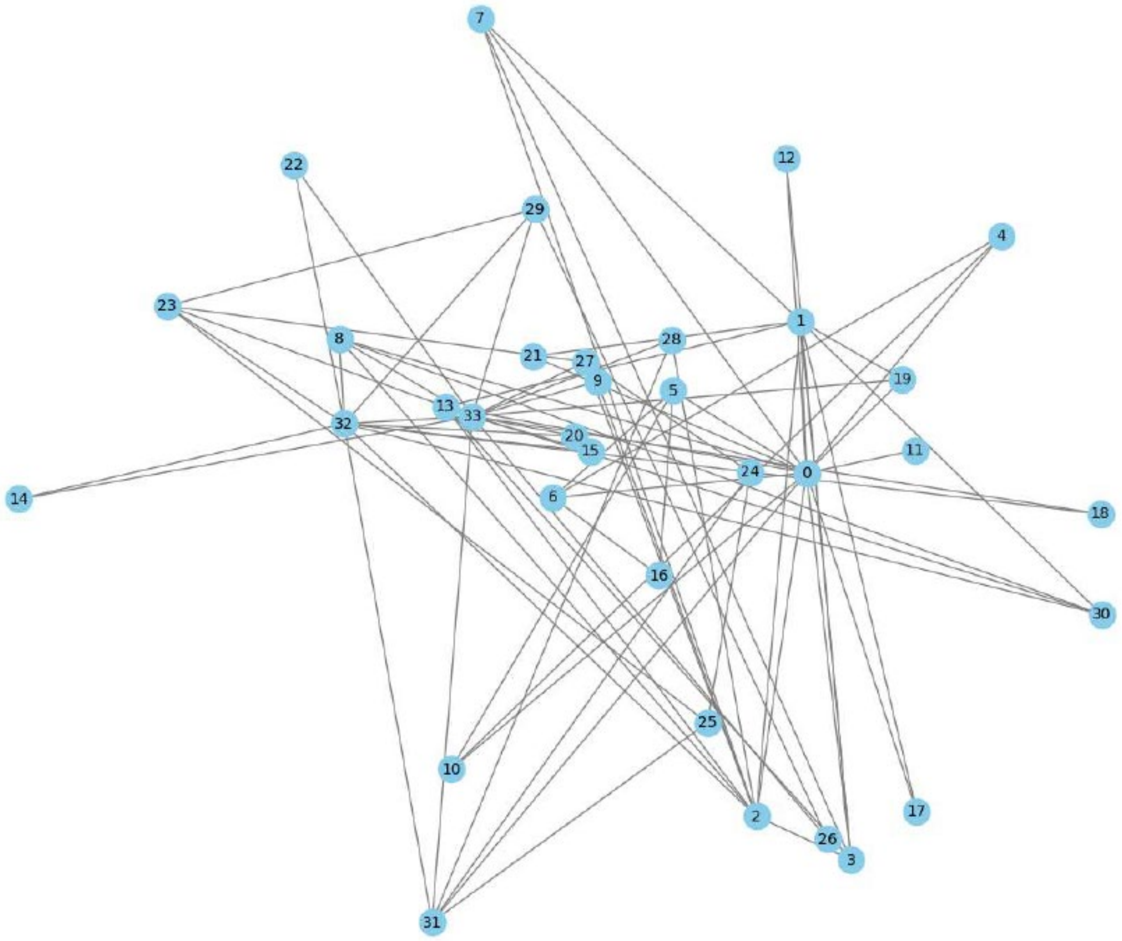
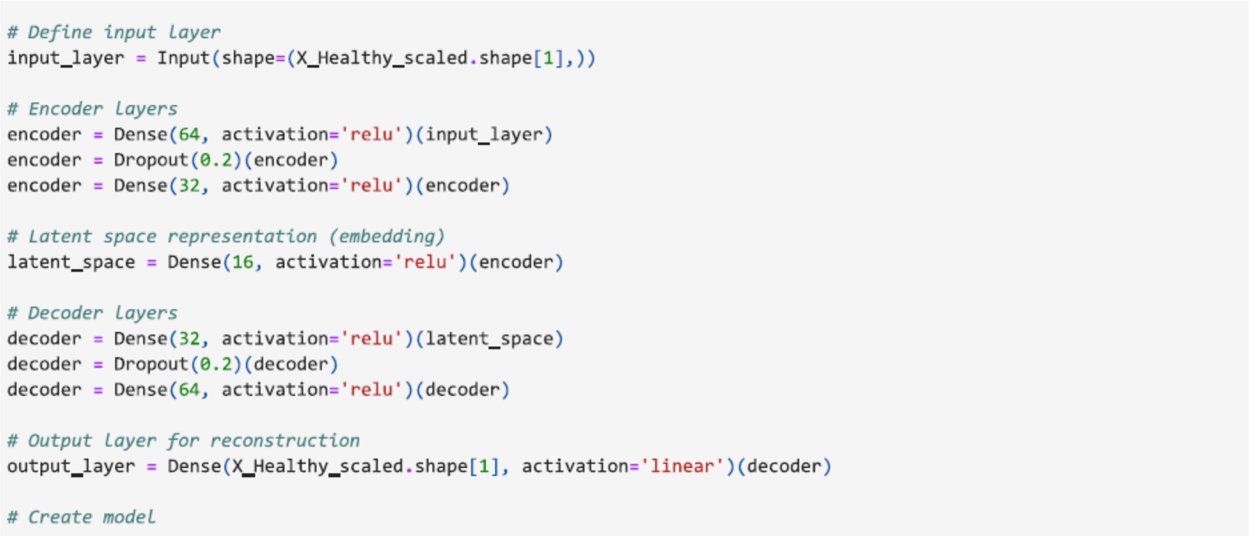
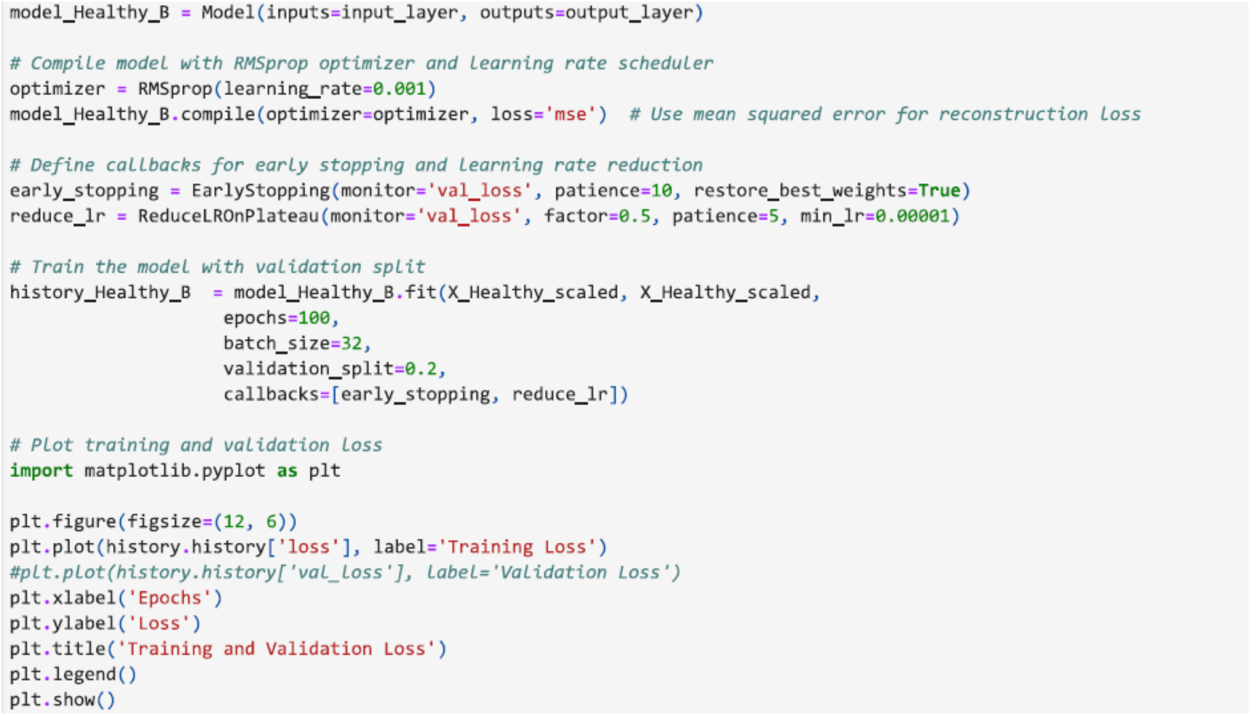


Figure 7 – The connectivity pattern between the clusters of the embeddings represented the latent space

However, quite surprisingly, this pattern of connectivity showed sensitivity to the training model parameters adopted. In the second model trained on the functional connectivity of the healthy cohort, the same architecture of the deep autoencoder was applied however, the optimizer and its effective learning rate for convergence was changed.



```
# Define input layer
input_layer = Input(shape=(X_Healthy_scaled.shape[1],))

# Encoder layers
encoder = Dense(64, activation='relu')(input_layer)
encoder = Dropout(0.2)(encoder)
encoder = Dense(32, activation='relu')(encoder)

# Latent space representation (embedding)
latent_space = Dense(16, activation='relu')(encoder)

# Decoder layers
decoder = Dense(32, activation='relu')(latent_space)
decoder = Dropout(0.2)(decoder)
decoder = Dense(64, activation='relu')(decoder)

# Output layer for reconstruction
output_layer = Dense(X_Healthy_scaled.shape[1], activation='linear')(decoder)

# Create model
```



```
model_Healthy_B = Model(inputs=input_layer, outputs=output_layer)

# Compile model with RMSprop optimizer and learning rate scheduler
optimizer = RMSprop(learning_rate=0.001)
model_Healthy_B.compile(optimizer=optimizer, loss='mse')  # Use mean squared error for reconstruction loss

# Define callbacks for early stopping and learning rate reduction
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5, min_lr=0.00001)

# Train the model with validation split
history_Healthy_B  = model_Healthy_B.fit(X_Healthy_scaled, X_Healthy_scaled,
                    epochs=100,
                    batch_size=32,
                    validation_split=0.2,
                    callbacks=[early_stopping, reduce_lr])

# Plot training and validation loss
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
plt.plot(history.history['loss'], label='Training Loss')
#plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.legend()
plt.show()
```

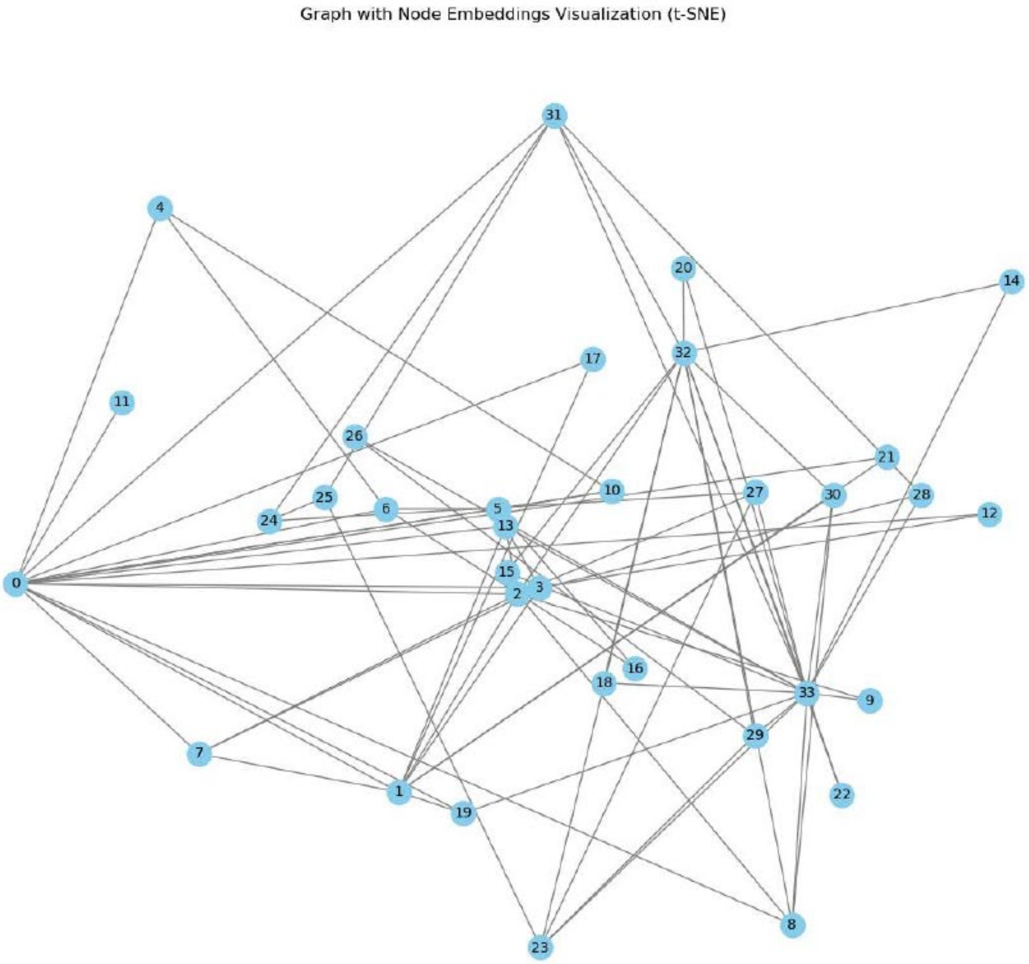
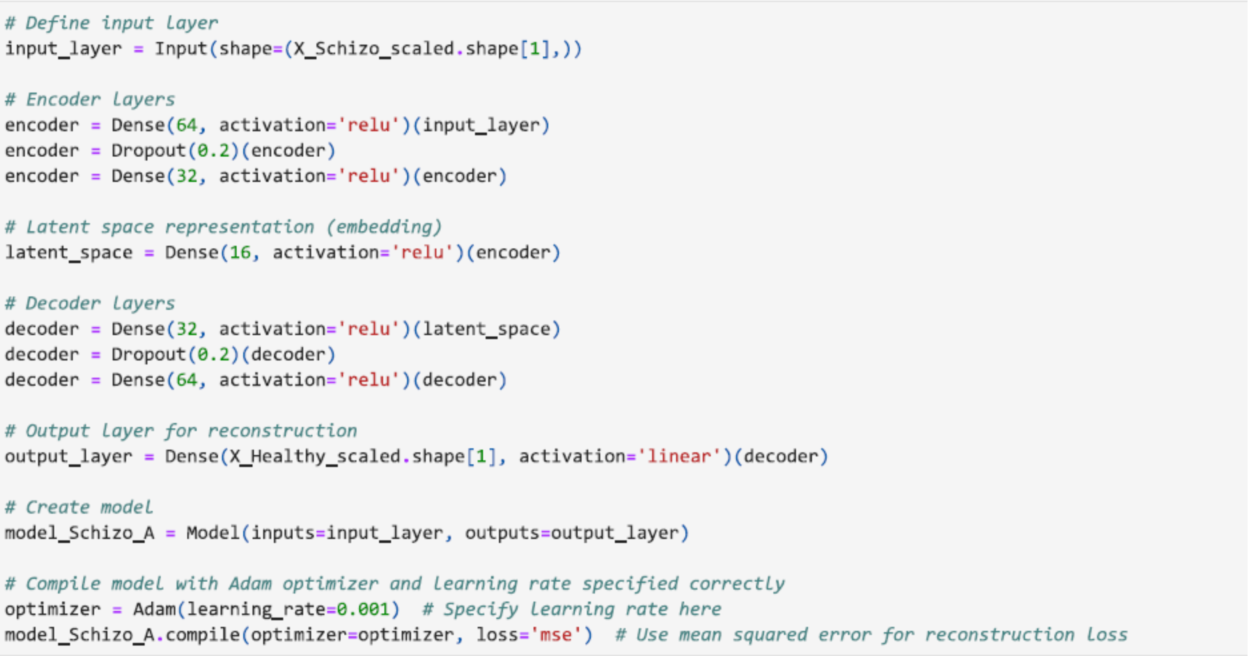


Figure 7 – The connectivity pattern of the nodes of functional neural connectivity with the different optimiser RMSprop used in the autoencoder training on the latent representation of the input data. This is markedly different than the first model where the optimizer adopted was ADAM.

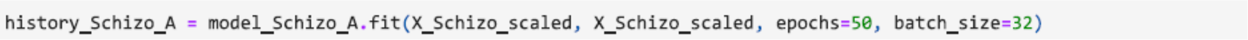
Schizophrenic Cohorts

In order to be consistent with the healthy cohort analysis, the same deep autoencoder architecture was applied to the preprocessed data obtained from (…) for the schizophrenic cohorts. The datasets before feeding into the neural network was normalized.

The deep autoencoder model with its hyperparameters was trained as below for 50 epoch and the loss was visualized.

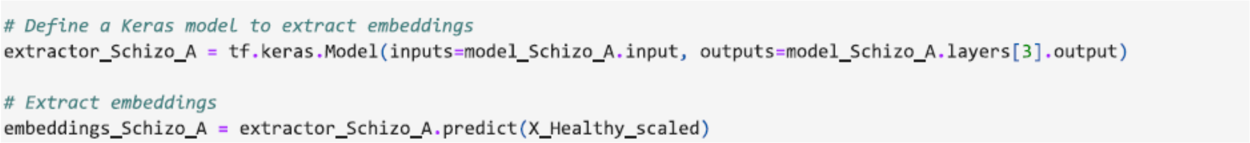


```
# Define input layer
input_layer = Input(shape=(X_Schizo_scaled.shape[1],))

# Encoder layers
encoder = Dense(64, activation='relu')(input_layer)
encoder = Dropout(0.2)(encoder)
encoder = Dense(32, activation='relu')(encoder)

# Latent space representation (embedding)
latent_space = Dense(16, activation='relu')(encoder)

# Decoder layers
decoder = Dense(32, activation='relu')(latent_space)
decoder = Dropout(0.2)(decoder)
decoder = Dense(64, activation='relu')(decoder)

# Output layer for reconstruction
output_layer = Dense(X_Healthy_scaled.shape[1], activation='linear')(decoder)

# Create model
model_Schizo_A = Model(inputs=input_layer, outputs=output_layer)

# Compile model with Adam optimizer and learning rate specified correctly
optimizer = Adam(learning_rate=0.001)  # Specify learning rate here
model_Schizo_A.compile(optimizer=optimizer, loss='mse')  # Use mean squared error for reconstruction loss
```



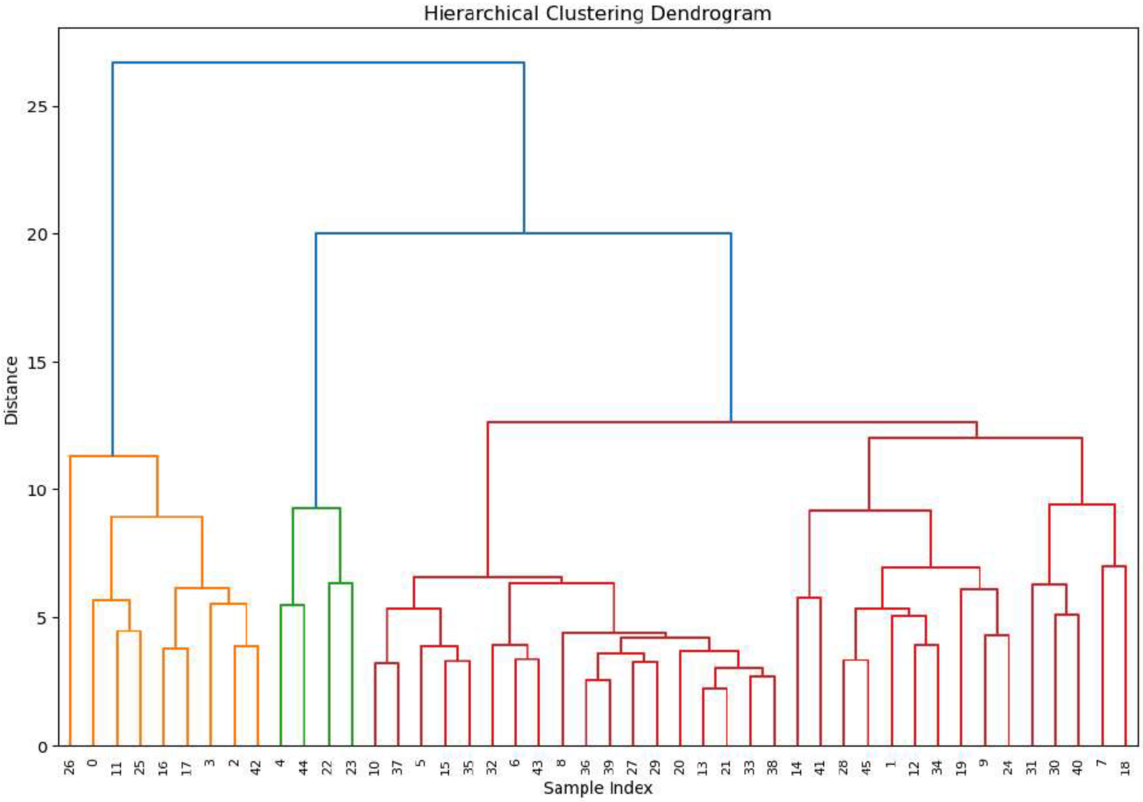
```
history_Schizo_A = model_Schizo_A.fit(X_Schizo_scaled, X_Schizo_scaled, epochs=50, batch_size=32)
```

And the information encapsulated in the latent space was extracted as previously done for the healthy cohort.



```
# Define a Keras model to extract embeddings
extractor_Schizo_A = tf.keras.Model(inputs=model_Schizo_A.input, outputs=model_Schizo_A.layers[3].output)

# Extract embeddings
embeddings_Schizo_A = extractor_Schizo_A.predict(X_Healthy_scaled)
```

The number of the clusters form the inputs of the functional neural connectivity was consistent as the healthy cohorts, 46 but the pattern of connectivity showed variation as will be discussed. The unsupervised hierarchical clustering shows the details as below:



```
Number of clusters: 46
```

Figure 9 – The dendrogram of the FNC of the patients diagnosed with schizophrenia

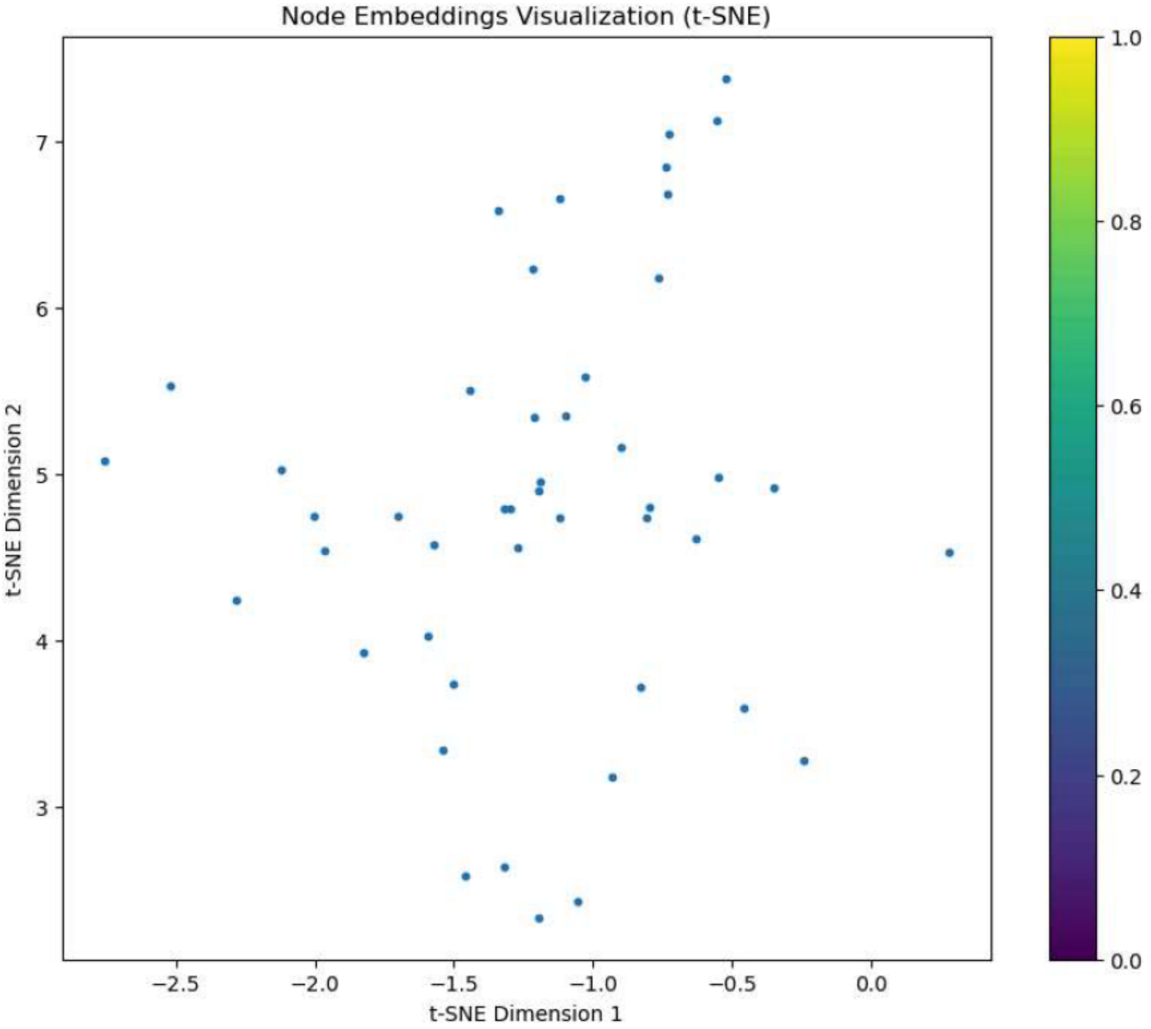


Figure 10 – The visualization of the clustering from the FNC by the t-SNE method

The graph representation of the pattern of the connectivity between the clusters of the functional neural connectivity crunched into the 2D plane by the t-distributed stochastic neighbor embedding reveals the pattern of the static connectivity during the resting state as:

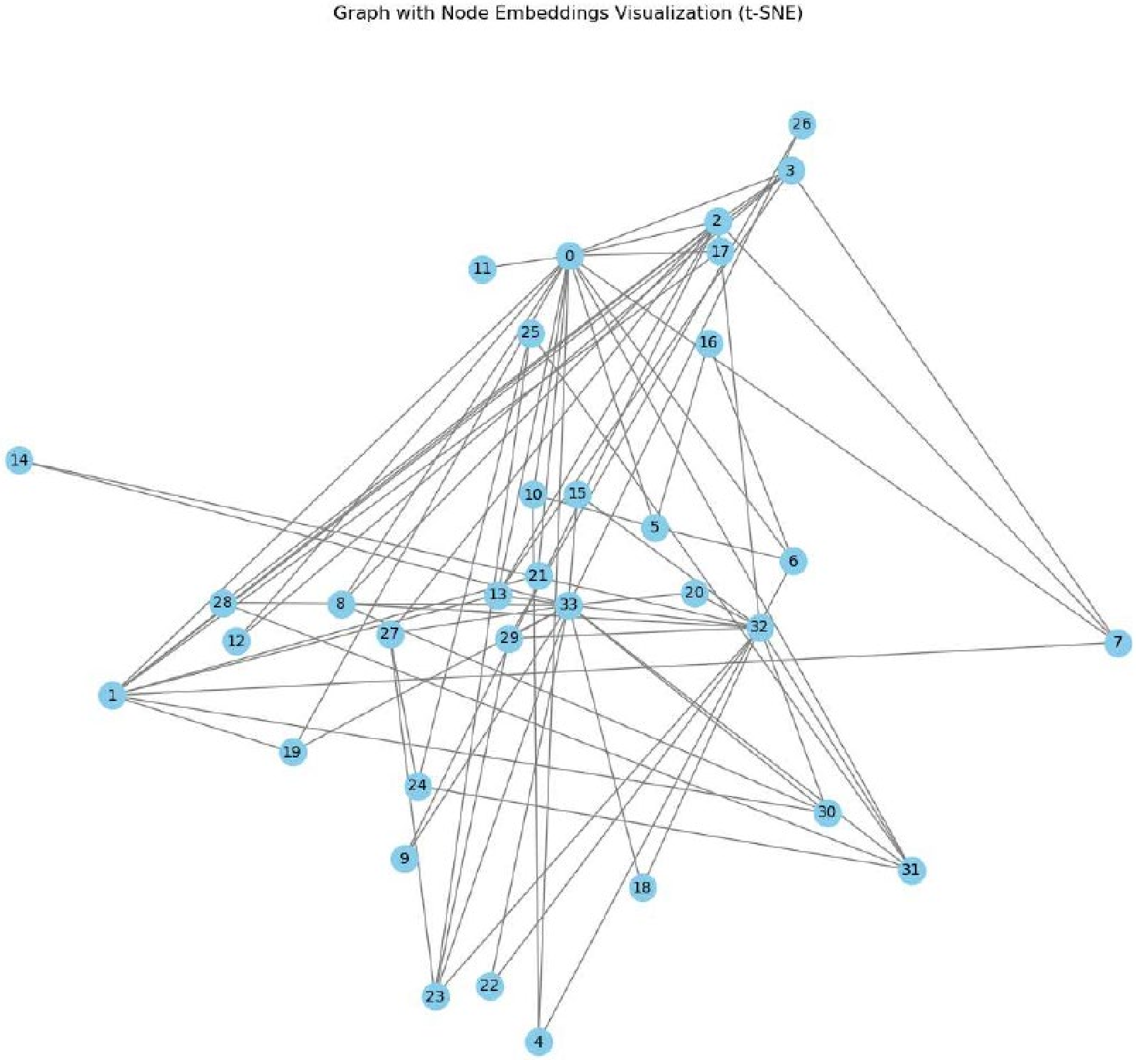
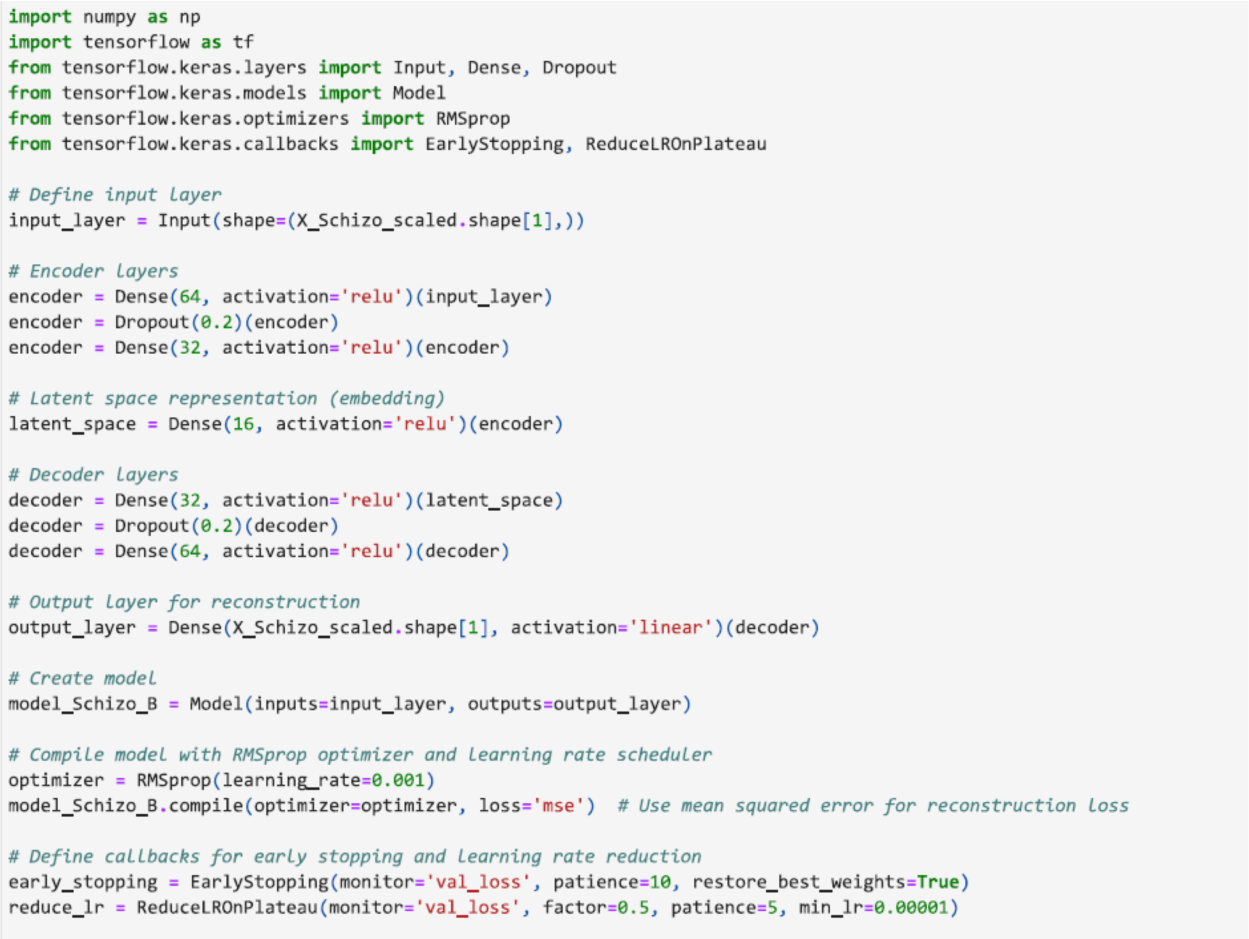
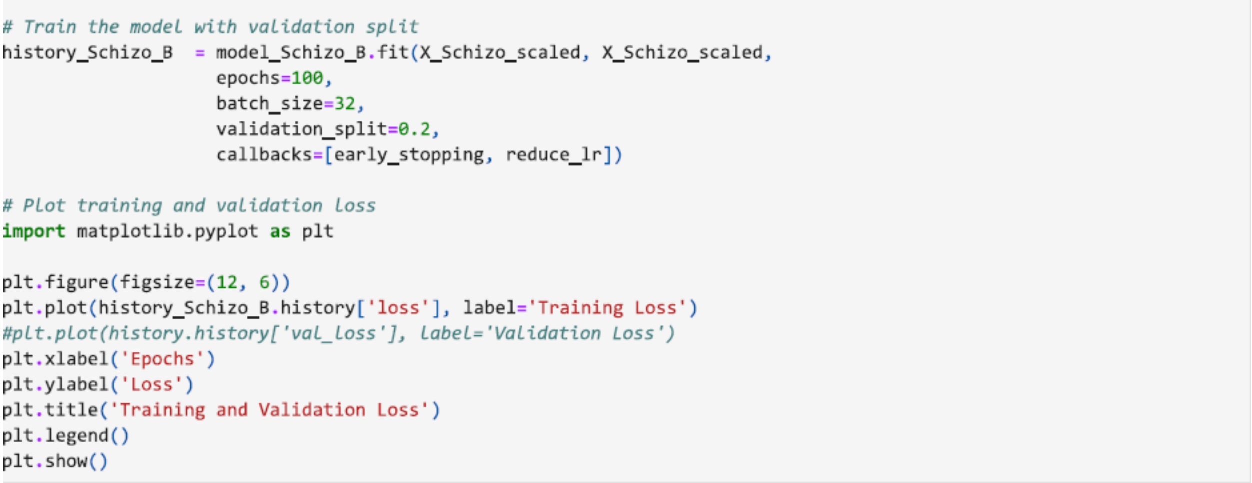


Figure 11 – t-SNE visualization of the graph for the connectivity pattern of the FNC of the patients with schizophrenia collected from fMRI scan during resting state.

However, as it was observed with the healthy cohort, this pattern of connectivity was sensitive to the model hyperparameters, such as the optimizers and learning rate. To preserve the consistency with the healthy cohort analysis, the data from the schizophrenic cohort was run by the second deep autoencoder with different hyperparameter.



```python
import numpy as np
import tensorflow as tf
from tensorflow.keras.layers import Input, Dense, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau

# Define input layer
input_layer = Input(shape=(X_Schizo_scaled.shape[1],))

# Encoder layers
encoder = Dense(64, activation='relu')(input_layer)
encoder = Dropout(0.2)(encoder)
encoder = Dense(32, activation='relu')(encoder)

# Latent space representation (embedding)
latent_space = Dense(16, activation='relu')(encoder)

# Decoder layers
decoder = Dense(32, activation='relu')(latent_space)
decoder = Dropout(0.2)(decoder)
decoder = Dense(64, activation='relu')(decoder)

# Output layer for reconstruction
output_layer = Dense(X_Schizo_scaled.shape[1], activation='linear')(decoder)

# Create model
model_Schizo_B = Model(inputs=input_layer, outputs=output_layer)

# Compile model with RMSprop optimizer and learning rate scheduler
optimizer = RMSprop(learning_rate=0.001)
model_Schizo_B.compile(optimizer=optimizer, loss='mse')  # Use mean squared error for reconstruction loss

# Define callbacks for early stopping and learning rate reduction
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5, min_lr=0.00001)
```



```python
# Train the model with validation split
history_Schizo_B  = model_Schizo_B.fit(X_Schizo_scaled, X_Schizo_scaled,
                    epochs=100,
                    batch_size=32,
                    validation_split=0.2,
                    callbacks=[early_stopping, reduce_lr])

# Plot training and validation loss
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
plt.plot(history_Schizo_B.history['loss'], label='Training Loss')
#plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.legend()
plt.show()
```

Although the number of the clusters were the same as 46, the pattern of connectivity was different

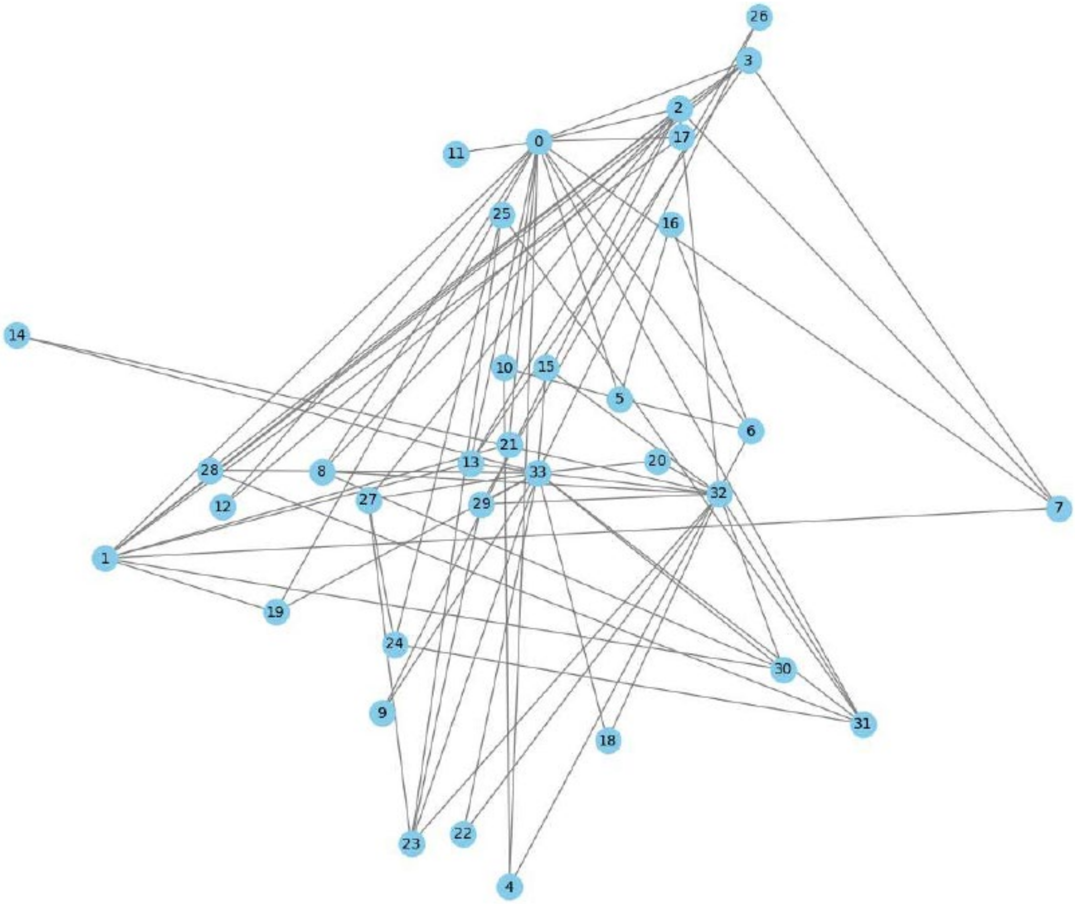


Figure 11 – The connectivity pattern during the resting state of the clusters of the FNC for the schizophrenic cohort where the hyperparameters were changed.

As it was observed the deep autoencoder model and the extracted pattern of the latent space revealed different connectivity pattern between the clusters of the functional neural network. This connectivity pattern also was subject to variation due to changes of the training model hyperparameters. As we changed the optimizers from ADAM to RMSprop with different learning rate that model converged faster, the pattern of the connectivity changed. This can heuristically imply that the pattern observed from the more robust training model is the one that will be more reliable for downstream analysis. In the next step, the probabilistic version of the deep autoencoder, variational autoencoder is used in order to provide a probabilistic inference on the nature of the functional neural connectivity.

Variational Autoencoder Philosophy & Application

variational autoencoders extend traditional autoencoders by introducing probabilistic modeling of the latent space. They are powerful tools for learning complex data distributions and generating new data samples, making them widely used in modern machine learning for tasks involving generative modeling and unsupervised learning.

Key Concepts of Variational Autoencoders:

1.Probabilistic Latent Space: Unlike traditional autoencoders, which learn deterministic mappings from input to latent space, VAEs learn a probabilistic mapping. This means that instead of learning a single point in the latent space for each input, VAEs learn a probability distribution over the latent variables.

2. Encoder (Recognition Model): The encoder in a VAE maps the input data to a distribution in the latent space. Specifically, it outputs parameters (mean and variance) of a multivariate Gaussian distribution that represents the latent variables.

3. Reparameterization Trick: To enable backpropagation and efficient gradient-based optimization, VAEs use a reparameterization trick during training. Instead of sampling directly from the inferred Gaussian distribution, the model samples from a standard Gaussian distribution and then scales and shifts the samples using the inferred mean and variance from the encoder.

4. Decoder (Generative Model): The decoder in a VAE takes samples from the latent space (which are drawn from the distribution output by the encoder) and reconstructs the original input data. This allows the model to generate new samples from the learned data distribution.

5. Objective Function: VAEs are trained to maximize the evidence lower bound (ELBO), which consists of two parts:

- Reconstruction Loss: Measures how well the reconstructed output matches the original input. It is typically a measure like mean squared error (MSE) for continuous data or binary cross-entropy for binary data.

- KL Divergence: Penalizes the divergence between the learned latent distribution and a prior distribution (usually a standard Gaussian). This encourages the latent space to be well-structured and smooth.

Advantages of Variational Autoencoders:

- Generative Modeling: VAEs can generate new data points by sampling from the latent space, allowing them to create new data that resembles the training data distribution.

- Continuous Latent Space: The latent space learned by VAEs tends to have a continuous and smooth structure, making them suitable for tasks like interpolation and exploration of latent representations.

- Regularization: The KL divergence term in the loss function acts as a regularizer, preventing overfitting and encouraging the model to learn meaningful latent representations.

Applications of Variational Autoencoders:

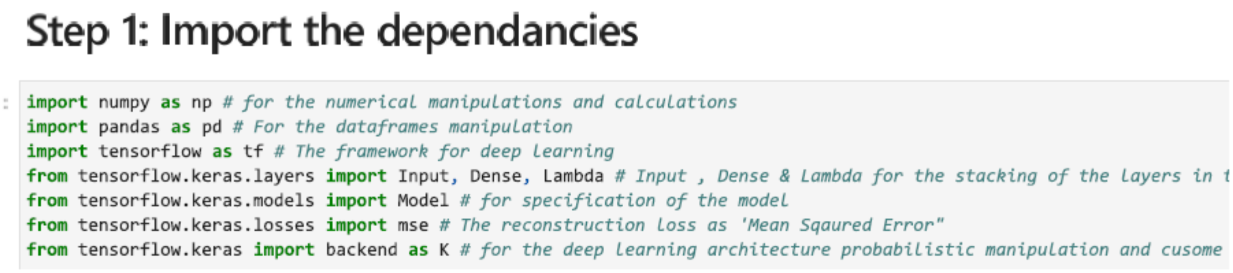
- Image Generation: VAEs can generate realistic images from learned latent representations, useful in tasks like image inpainting and data augmentation.

- Anomaly Detection: By reconstructing data and measuring reconstruction error, VAEs can identify anomalies or outliers that do not fit well into the learned data distribution.

- Representation Learning: VAEs learn compact and meaningful representations of data, which can be used as inputs for downstream tasks such as classification or clustering

Variational Autoencoders in action

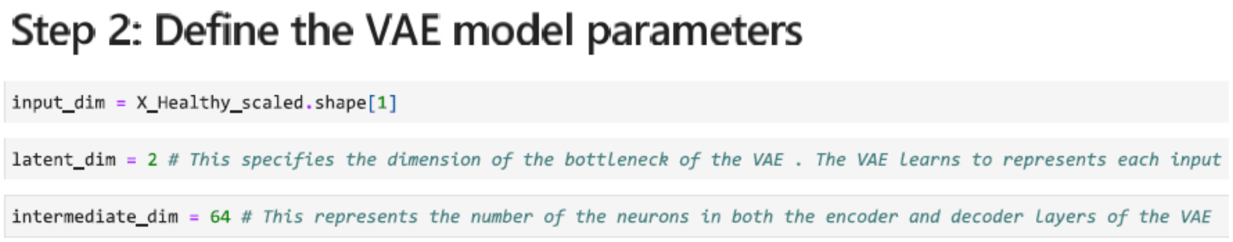
To analyse the network connectivity from the preprocessed data from the resting state fMRI with the probabilistic version of the autoencoders, the following libraries were installed:



Step 1: Import the dependancies

```
import numpy as np # for the numerical manipulations and calculations
import pandas as pd # For the dataframes manipulation
import tensorflow as tf # The framework for deep learning
from tensorflow.keras.layers import Input, Dense, Lambda # Input , Dense & Lambda for the stacking of the layers in t
from tensorflow.keras.models import Model # for specification of the model
from tensorflow.keras.losses import mse # The reconstruction loss as 'Mean Sqaured Error"
from tensorflow.keras import backend as K # for the deep learning architecture probabilistic manipulation and cusome
```

However, the hyperparameters of the model , for the dimension of the latent representation of the input as the functional connectivity was set as:



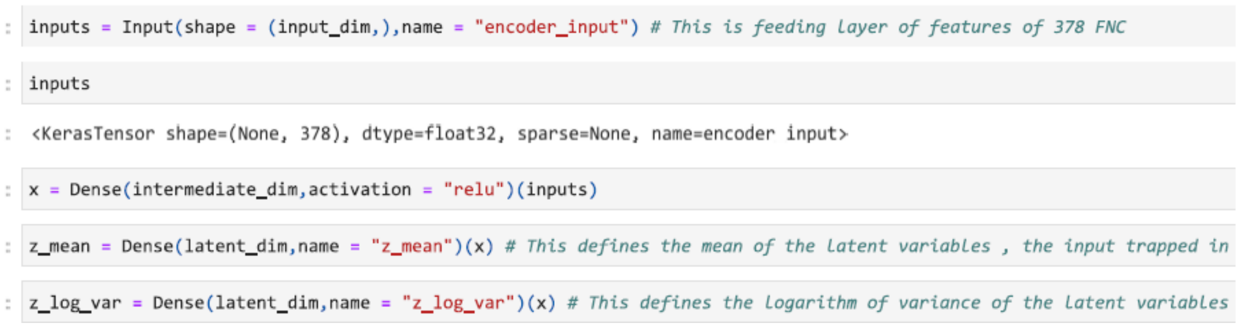
Step 2: Define the VAE model parameters

```
input_dim = X_Healthy_scaled.shape[1]
```

```
latent_dim = 2 # This specifies the dimension of the bottleneck of the VAE . The VAE learns to represents each input
```

```
intermediate_dim = 64 # This represents the number of the neurons in both the encoder and decoder layers of the VAE
```

As the latent representation of the input is represented by a probability distribution of multivariate normal distribution the following parameters were set accordingly to express the mean and dispersion of the distribution.



```
inputs = Input(shape = (input_dim,),name = "encoder_input") # This is feeding layer of features of 378 FNC
```

```
inputs
```

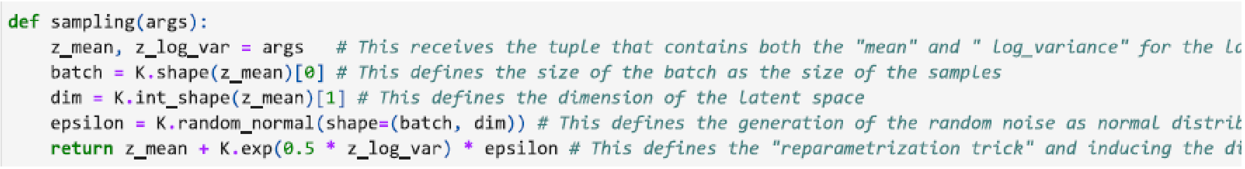
```
<KerasTensor shape=(None, 378), dtype=float32, sparse=None, name=encoder input>
```

```
x = Dense(intermediate_dim,activation = "relu")(inputs)
```

```
z_mean = Dense(latent_dim,name = "z_mean")(x) # This defines the mean of the latent variables , the input trapped in
```

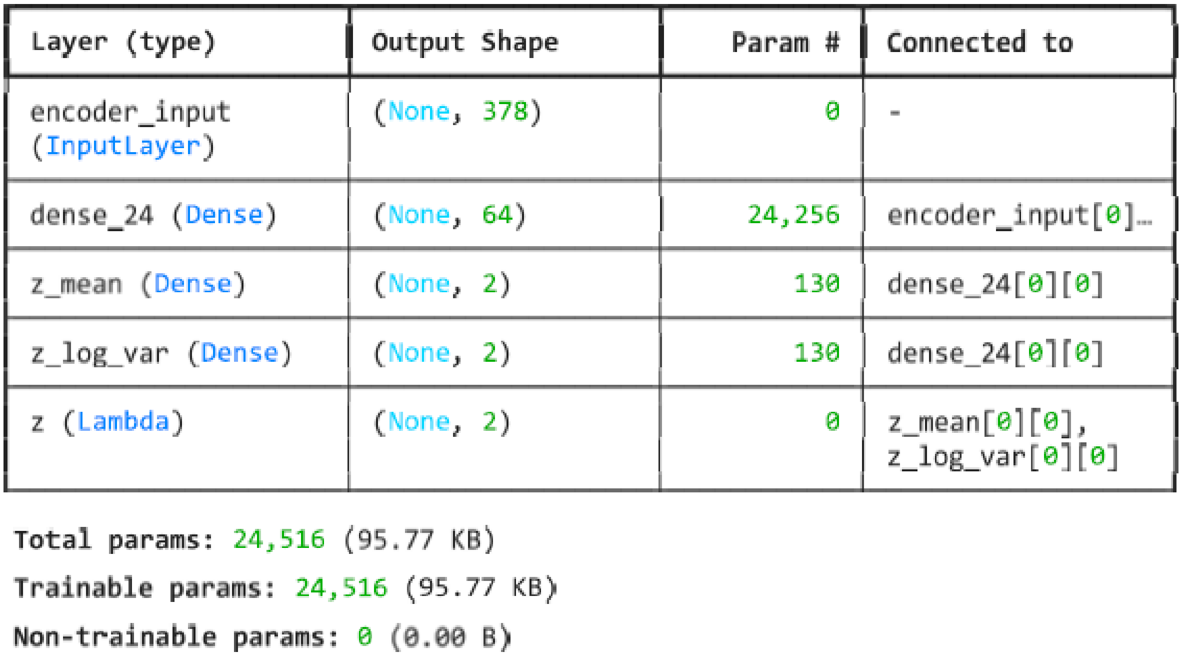
```
z_log_var = Dense(latent_dim,name = "z_log_var")(x) # This defines the logarithm of variance of the latent variables
```

However, the stochastic sampling of the latent space for the posterior distribution was



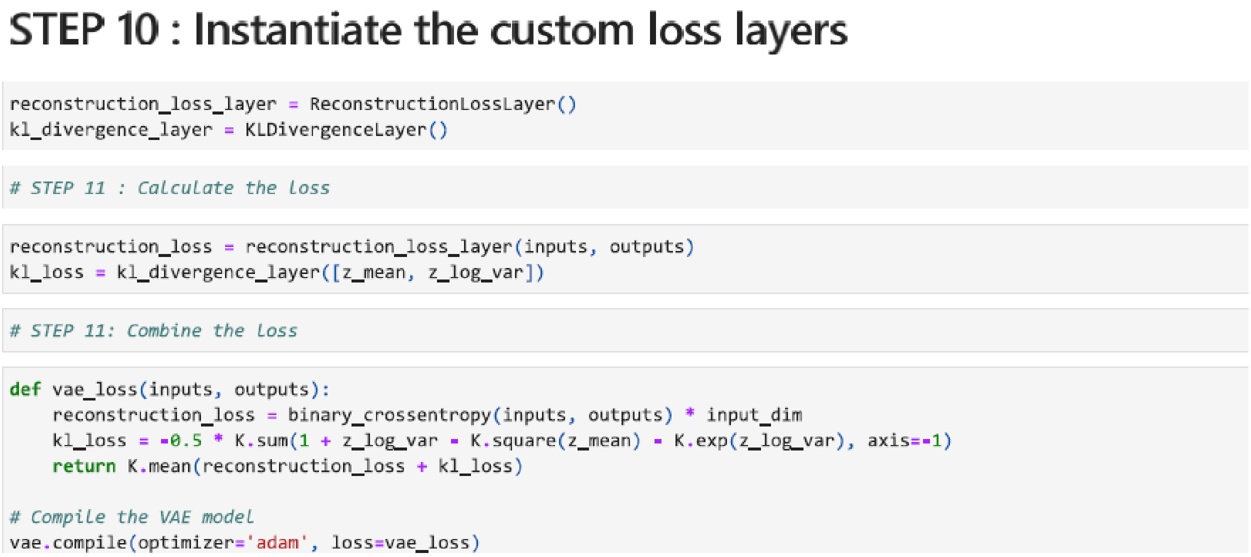
```
def sampling(args):
    z_mean, z_log_var = args   # This receives the tuple that contains both the "mean" and " log_variance" for the la
    batch = K.shape(z_mean)[0] # This defines the size of the batch as the size of the samples
    dim = K.int_shape(z_mean)[1] # This defines the dimension of the latent space
    epsilon = K.random_normal(shape=(batch, dim)) # This defines the generation of the random noise as normal distrib
    return z_mean + K.exp(0.5 * z_log_var) * epsilon # This defines the "reparametrization trick" and inducing the di
```

The encoder representation through the representation of the input features produced the following metrics.



```
┏━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━┓
┃ Layer (type)        ┃ Output Shape       ┃     Param # ┃ Connected to       ┃
┡━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━┩
│ encoder_input       │ (None, 378)        │           0 │ -                  │
│ (InputLayer)        │                    │             │                    │
├─────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ dense_24 (Dense)    │ (None, 64)         │      24,256 │ encoder_input[0]…  │
├─────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ z_mean (Dense)      │ (None, 2)          │         130 │ dense_24[0][0]     │
├─────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ z_log_var (Dense)   │ (None, 2)          │         130 │ dense_24[0][0]     │
├─────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ z (Lambda)          │ (None, 2)          │           0 │ z_mean[0][0],      │
│                     │                    │             │ z_log_var[0][0]    │
└─────────────────────┴────────────────────┴─────────────┴────────────────────┘
 Total params: 24,516 (95.77 KB)
 Trainable params: 24,516 (95.77 KB)
 Non-trainable params: 0 (0.00 B)
```

One of the amin differences of the variational autoencoders are the specificity of the loss function that is designed as a factor for the minimization to optimally train the model. The loss function is composed of tow part , the fist one is the one that conventionally measures in the Euclidean space the difference between the target and the input and the other one is more conventional with respect to the Bayesian methods where the difference between the distribution is more highlighted through the divergence as Kulback -Liebler metric where it measures the divergence between pairwise distribution.



STEP 10 : Instantiate the custom loss layers

```
reconstruction_loss_layer = ReconstructionLossLayer()
kl_divergence_layer = KLDivergenceLayer()
```

```
# STEP 11 : Calculate the loss
```

```
reconstruction_loss = reconstruction_loss_layer(inputs, outputs)
kl_loss = kl_divergence_layer([z_mean, z_log_var])
```

```
# STEP 11: Combine the Loss
```

```
def vae_loss(inputs, outputs):
    reconstruction_loss = binary_crossentropy(inputs, outputs) * input_dim
    kl_loss = -0.5 * K.sum(1 + z_log_var - K.square(z_mean) - K.exp(z_log_var), axis=-1)
    return K.mean(reconstruction_loss + kl_loss)

# Compile the VAE model
vae.compile(optimizer='adam', loss=vae_loss)
```

As the code above demonstrates the model with that custom loss function was trained. The variational autoencoder as well produced the same connectivity network as its non probabilistic counterpart, deep autoencoder.

Conclusion

In this brief review, the functional neural connectivity from the resting state between healthy and schizophrenic patients were analysed with both deep autoencoder and variational autoencoder. It was demonstrated through the graph analysis that the pattern of the connectivity between these two subjects were drastically different. Although of higher importance, this pattern was shown to be sensitive to the hyperparameters of the model when extracted the embedding from the latent representation of the resting state functional neural connectivity. Further analysis is required to obliterate this sensitivity.

References

Alexander-Bloch, A. F., Reiss, P. T., Rapoport, J., & McAdams, H. (2013). Abnormal cortical growth in schizophrenia targets normative modules of synchronized development. *Biological Psychiatry, 74*(6), 438-446. https://doi.org/10.1016/j.biopsych.2013.03.031

Anticevic, A., Cole, M. W., Murray, J. D., Corlett, P. R., Wang, X. J., & Krystal, J. H. (2015). The role of default network deactivation in cognition and disease. *Trends in Cognitive Sciences, 16*(12), 584-592. https://doi.org/10.1016/j.tics.2012.10.008

Babaeeghazvini, P., Rueda-Delgado, L. M., Gooijers, J., Swinnen, S. P., & Daffertshofer, A.(2021, September 10). Brain structural and functional connectivity: A review of combined works of Diffusion Magnetic Resonance Imaging and electro-encephalography. Frontiers. https://www.frontiersin.org/journals/human-neuroscience/articles/10.3389/fnhum.2021.721206

Bassett, D. S., & Sporns, O. (2017). Network neuroscience. *Nature Neuroscience, 20*(3), 353-364. https://doi.org/10.1038/nn.4502

Biswal, B., Yetkin, F. Z., Haughton, V. M., & Hyde, J. S. (1995). Functional connectivity in the motor cortex of resting human brain using echo-planar MRI. *Magnetic Resonance in Medicine, 34*(4), 537-541. https://doi.org/10.1002/mrm.1910340409

Bullmore, E., & Sporns, O. (2009). Complex brain networks: Graph theoretical analysis of structural and functional systems. *Nature Reviews Neuroscience, 10*(3), 186-198. https://doi.org/10.1038/nrn2575

Buckner RL. The brain's default network: origins and implications for the study of psychosis. Dialogues Clin Neurosci. 2013 Sep;15(3):351-8. doi: 10.31887/DCNS.2013.15.3/rbuckner. PMID: 24174906; PMCID: PMC3811106.

/full

Buckner, R. L., Andrews-Hanna, J. R., & Schacter, D. L. (2008). The brain's default network: Anatomy, function, and relevance to disease. Annals of the New York Academy of Sciences, 1124(1), 1-38. https://doi.org/10.1196/annals.1440.01

Crossley, N. A., Mechelli, A., Vértes, P. E., Winton-Brown, T. T., Patel, A. X., Ginestet, C. E., ... & Bullmore, E. T. (2014). Cognitive relevance of the community structure of the human brain functional coactivation network. *Proceedings of the National Academy of Sciences, 110*(28), 11583-11588. https://doi.org/10.1073/pnas.1220826110

Allen EA, Damaraju E, Plis SM, Erhardt EB, Eichele T, Calhoun VD. Tracking whole-brain connectivity dynamics in the resting state. Cereb Cortex. 2014 Mar;24(3):663-76. doi: 10.1093/cercor/bhs352. Epub 2012 Nov 11. PMID: 23146964; PMCID: PMC3920766.

Farah, M. J. (2017). Ethics of neuroscience and neurotechnology. *Annual Review of Psychology, 68*, 363-386. https://doi.org/10.1146/annurev-psych-010416-044202

Fornito, A., Zalesky, A., & Breakspear, M. (2012). The connectomics of brain disorders. *Nature Reviews Neuroscience, 16*(3), 159-172. https://doi.org/10.1038/nrn4081

Friston, K. J. (2011). Functional and effective connectivity: A review. *Brain Connectivity, 1*(1), 13-36. https://doi.org/10.1089/brain.2011.0008

Hillebrand, A., & Barnes, G. R. (2002). A quantitative assessment of the sensitivity of whole-head MEG to activity in the adult human cortex. *NeuroImage, 16*(3), 638-650. https://doi.org/10.1006/nimg.2002.1131

Pettersson-Yeo, W., Allen, P., Benetti, S., McGuire, P., & Mechelli, A. (2011). Dysconnectivity in schizophrenia: Where are we now? *Neuroscience and Biobehavioral Reviews, 35*(5), 1110-1124. https://doi.org/10.1016/j.neubiorev.2010.11.004

Raichle, M. E. (2015). The brain's default mode network. *Annual Review of Neuroscience, 38*, 433-447. https://doi.org/10.1146/annurev-neuro-071013-014030

Rubinov, M., & Sporns, O. (2010). Complex network measures of brain connectivity: Uses and interpretations. *NeuroImage, 52*(3), 1059-1069. https://doi.org/10.1016/j.neuroimage.2009.10.003

van den Heuvel, M. P., & Fornito, A. (2014). Brain networks in schizophrenia. *Neuropsychology Review, 24*(1), 32-48. https://doi.org/10.1007/s11065-014-9248-7 Whitfield-Gabrieli, S., & Ford, J

Whitfield-Gabrieli, S., & Ford, J. M. (2012). Default mode network activity and connectivity in psychopathology. Annual Review of Clinical Psychology, 8, 49-76. https://doi.org/10.1146/annurev-clinpsy-032511-143049