Entrega de Redes Generativas: Superresolución en Grafos

Federico Molina

***Resumen*—Se plantea un problema de super resolución en grafos usando la arquitectura AGRSNet. Se pruebas distintas modificaciones y se se obtienen resultados similares, logran mejorar levemente los resultados publicados en algunos casos.**

I. Introducción

La neuroimagen médica no invasiva ha facilitado la cartografía de la conectividad cerebral, lo que ha llevado al desarrollo de conectividades cerebrales morfológicas, estructurales y funcionales. La representación del cerebro como un grafo permite una comprensión integral de las actividades neuronales, dicha red cerebral denotada **conectoma** se define por dos componentes: los elementos neuronales del cerebro y sus conexiones neuronales [1]. Las GNNs (Graph Neural Networks), al aprovechar su capacidad para procesar datos no euclidianos, ofrecen un enfoque sofisticado para aprender estructuras de grafo profundas, mejorando significativamente el rendimiento en varias tareas de neurociencia de redes [2].

Los neurocientífcos modelan el cerebro como un grafo donde los nodos representan las regiones anatómicas del cerebro (regiones de interés, ROI) las cuales se conectan por aristas que referencian conectividades morfológicas, funcionales o estructurales de los nodos [3]. Es decir, se definen tres tipos de grafos cerebrales, los morfológicos, estructurales o funcionales, cada cual se obtiene a partir de técnicas diferentes. Estas representaciones dan lugar a la comprensión del grafo cerebral en tres dimensiones: resolución, dominio y tiempo [2].

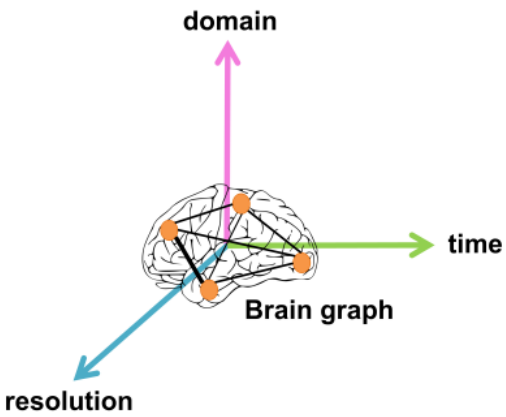


Figura 1: Representación del cerebro.

Federico Molina realizó esta entrega para la materia de Redes Generativas de la Facultad de Ingeniería de la Universidad de la República. correo: federicoandresmolina@gmail.com

En el caso temporal, se analizan los cambios del grafo cerebral a lo largo de un periodo de tiempo. Por ejemplo, como se modifica la matriz de adyacencia que representa la conectividad cerebral. Lo cual puede mostrar cambios en un cerebro sano a uno no sano, o ver como cambian las interacciones entre las distintas ROI. El dominio refiere a modalidad en que los datos fueron recolectados (fMRI, resonancia magnética (MRI) de difusión), puesto que las diferentes técnicas generan distintos tipos de conectomas (funcionales o estructurales, por ejemplo). Por tanto, es posible tener grafos cerebrales diferentes y complementarios según que modalidad se utilice, lo cual puede generar conjuntos de datos multimodales. Por último, el eje de resolución refiere a que un cerebro puede ser modelado por un conectoma a diferentes resoluciones, mayor resolución implica mayores ROI (más cantidad de nodos) y conexiones entre ellas.

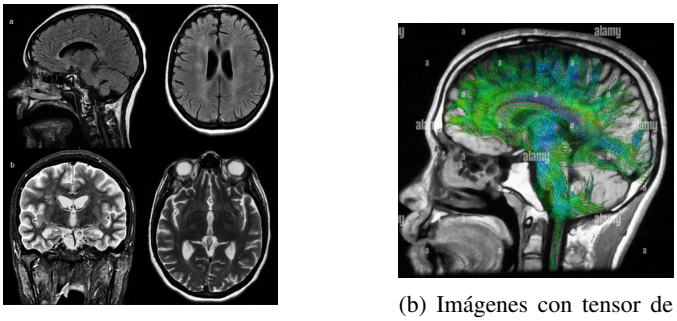
Este trabajo se enfoca completamente sobre el eje de resolución cerebral mediante datos funcionales del conectoma obtenidos mediante fMRI. Si bien el problema de resolución ha sido fuertemente estudiado para generar imágenes de alta resolución a partir de imágenes de baja resolución, el problema para el caso de grafos cerebrales no ha sido tan estudiado [4]. Enfocarse en la resolución en grafos para el caso cerebral (en vez de enfocarse solamente en imágenes) tiene que ver con que teóricamente, según los trabajos de Ramón y Cajal, las neuronas son células discretas que interactúan mediante conexiones sinápticas, lo cual clasifica naturalmente en una representación teórica de grafos [3].

La motivación principal del problema de resolución cerebral radica en el elevado costo de obtener datos de alta resolución, en la escasez de aparatos (especialmente en países de ingresos medios o bajos) para su obtención y el tiempo requerido para su procesamiento. De hecho, en el caso del conectoma el tiempo de cálculo por sujeto es muy alto y los pasos de preprocesamiento como el registro y la propagación de etiquetas son muy propensos a la variabilidad y sesgo [5].

La importancia radica en, por ejemplo, los avances importantes en el diagnóstico de trastornos cerebrales y exploración de la anatomía cerebral que se han podido realizar mediante neuroimágenes como MRI (resonancia magnética) o DTI (imágenes con tensor de difusión), ver Figura 2.

Por ejemplo, la resonancia magnética ayuda a mostrar variaciones detalladas en la estructura y función del cerebro [5]. En el caso del conectoma, su diversidad en resolución mejora la eficacia en el temprano diagnóstico de enfermedades [2].

La obtención del conectoma cerebral es un proceso complejo que integra imágenes de resonancia magnéticas y pasos como la extracción del cráneo hasta el espesor cortical, seg-



(a) Resonancia magnética.

(b) Imágenes con tensor de difusión.

Figura 2: Técnicas de Imágenes cerebrales.

mentación del tejido y el registro en un atlas cerebral. Para generar los conectomas a diferentes resoluciones se utiliza un atlas cerebral de imágenes para definir la división del cerebro en N (dependiendo de la resolución) regiones de interés. Entonces, un conectoma se compone de N nodos donde cada nodo identifica a una ROI del cerebro y el peso de las aristas que unen esas conexiones indican la intensidad de la conexión entre dos regiones, como se muestra en la Figura 3. Dichos grafos funcionales se logran mediante los fMRI, específicamente a partir de la señal dependiente del nivel de oxígeno en sangre (en inglés BOLD) que muestra los cambios en la oxigenación de la sangre a lo largo del tiempo vinculados a la actividad neuronal en una región particular del cerebro. Primero, la señal reportada se promedia dentro de cada ROI del cerebro. A continuación, se calcula una medida de correlación, entre pares de regiones, lo que resulta en la conectividad funcional que representa la comunicación entre pares de regiones cerebrales. En los gráficos cerebrales funcionales, los nodos no tienen características y las aristas generalmente son no dirigidas y ponderadas [2]. Esto último se debe a que los fMRI son técnicas no invasivas que no logran captar la dirección, pero teóricamente el conectoma debería ser un grafo dirigido [3].

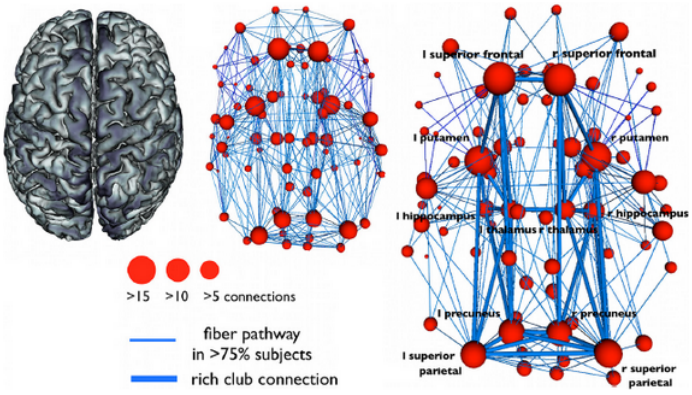


Figura 3: Conectoma cerebral.

Como el objetivo es entender como obtener un conectoma de alta resolución (HR) a partir de uno de baja resolución (LR), vamos a seguir la propuesta de una Adversarial Graph Super-Resolution Network (**AGSR-Net**) [4]. La clave de su funcionamiento se puede resumir en 4 pasos fundamentales:

1. Aprender características para cada región de interés (ROI) del cerebro en el conectoma LR (U-Autoencoder).
2. Diseñar de una operación de super-resolución que prediga un conectoma HR desde una matriz de conectividad y características obtenidas del paso anterior de un conectoma LR (Capa de alta resolución, GSR).
3. Aprender características de cada nodo en el grafo HR obtenido de la super-resolución en el paso anterior (Graph Convolution Network).
4. Utilizar las etapas anteriores como un generador y agregar un discriminador obteniendo una arquitectura GAN.

Para esto los autores adoptan una arquitectura de **graph U-net** [6], una arquitectura encoder-decoder basada en convoluciones, operaciones de pooling y unpooling adaptadas a grafos, lo cual lo diferencia de la arquitectura **U-net** [7]. Sin embargo las graph U-net se enfocan en predicción de enlaces o clasificación de nodos, en lugar de la super-resolución. En particular, es un enfoque centrado en nodos, donde un nodo n representa una muestra y se mapea a un espacio m-dimensional.

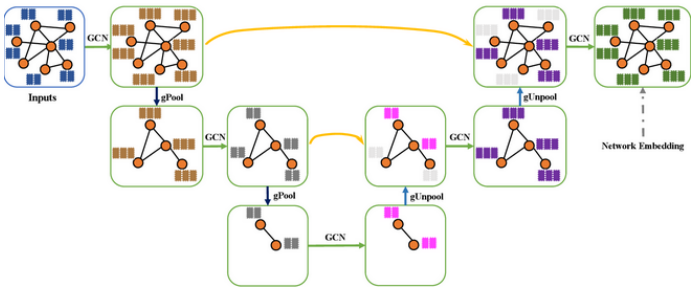


Figura 4: graph U-net.

Por otro lado, el **U-Autoencoder** utilizado es una arquitectura centrada en grafos. Donde una muestra está representada por un conectoma, un grafo totalmente conectado donde los nodos no tienen características y el peso de las aristas denotan la intensidad de las conexiones cerebrales entre dos nodos. La idea es aprender un mapeo entre el nodo n a un espacio m-dimensional que traduce la topología y relaciones entre los nodos en el conectoma como características de nodos.

Inicialmente se asignan vectores de características a cada ROI del cerebro y se aprenden características de los nodos promediando localmente las características de sus vecinos en función de sus pesos de conectividad.

Luego se agrega una capa GSR para generar un conectoma HR desde las características de los nodos del conectoma LR aprendido en el bloque del graph U-Autoencoder. Específicamente en el **bloque GSR** se propone una regla de propagación por capas basado en la teoría espectral de grafos [8].

Posteriormente, se agregan 2 capas adicionales de convolución en grafos para aprender mejor las características de cada ROI en el grafo HR. Finalmente se agrega una discriminador utilizando las etapas previas como un generador, obteniendo un modelo GAN, esta última etapa es relevante para que la distribución del cerebro HR real y predicho coincidan.

En nuestro caso, realizamos y planteamos diferentes modificaciones sobre el modelo. Por ejemplo, en la etapa final del generador en las capas finales de convolución ¿Es relevante la modificación en las capas de convolución? Si la respuesta es afirmativa, implica que una posible linea de trabajo futuro radica en plantear convoluciones que logren una mejora en la obtención de cerebros de HR. En las modificaciones realizadas no se obtienen mejoras en este sentido. Los autores plantean simetrizar las matrices de adyacencia y embedding de HR obtenidas mediante una operación que es invariante por simetría, sin embargo, esto es generalizable a utilizar cualquier función que simetrice una matriz. Probamos con reflejar y tomar el valor máximo obteniendo resultados similares. También se plantea una forma de inicializar la matriz HR de embedding, que relaciona los nodos que están en un camino de máximo dos con el objetivo de no obtener nodos aislados, nuevamente esto es generalizable a caminos de largo k que puedan incrementar aún más la conectividad entre nodos, aunque no se explora esta modificación. Otro camino es modificar el U-Autoencoder, en específico las operaciones de Unpooling. Una posible extensión es adaptar la propuesta de Unpooling Layer for Graph Generation [9]. En este trabajo se realiza una modificación en la forma de calculo del Graph Unpooling obteniendo resultados similares. Luego en el mismo U-Autoencoder se cambian las capas de GCN por capas GAT pero no se observan mejoras. Se plantea un cambio sobre el discriminador, agregando capas de *batch normalization* y Dropout, y los resultados son levemente inferiores.

También se plantean modificaciones sobre la función de pérdida, en específico sobre la pérdida de reconstrucción y se obtienen resultados levemente mejores que los publicados.

Por último, es posible agregar mayor cantidad de capas GCN tanto en el U-Autoencoder como luego de la capa GSRNet y agregar Dropout, sin embargo estos cambios no se pruebas. También es posible utilizar procesos de difusión [10]. Y finalmente, en el proceso se realiza un etapa de padding y unpadding para que los cerebros predichos tengan las mismas dimensiones que los reales, si bien este punto no se trabaja es un punto de discusión y mejora, la opción más fácil para eliminar esta etapa del entrenamiento es quedarse filtrar el cerebro HR predicho respecto a las dimensiones del real.

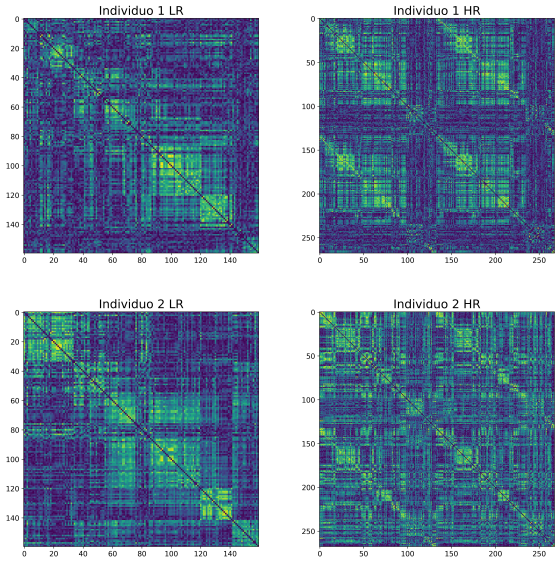


Figura 5: Muestra cerebros.

II. Datos

Los datos utilizados provienen de un estudio longitudinal (Southwest University Longitudinal Imaging Multimodal, SLIM) sobre datos neuroimagenológicos de test-retest de adultos jóvenes sanos en el suroeste de China. Se recolectaron datos multimodales de resonancia magnética (mMRI) y evaluaciones conductuales de 121 sujetos que completaron las tres sesiones, 211 sujetos con dos sesiones y 263 sujetos que participaron en una sola sesión, con un intervalo promedio de 817 días entre la primera y la tercera sesión. Los participantes tenían una edad promedio de 20 años (rango 17–27). El estudio destaca por su diseño longitudinal a largo plazo (2011-2015) con un rango de edad estrecho, proporcionando una valiosa base de datos para investigar la fiabilidad y reproducibilidad de las correlaciones cerebro-conductales. El estudio incluye imágenes estructurales, de resonancia magnética funcional en estado de reposo (rs-fMRI), y de imagen por tensor de difusión (DTI), junto con amplias evaluaciones conductuales, enfocándose principalmente en la creatividad y los factores de riesgo de trastornos afectivos. En nuestro caso solo se utilizan los datos obtenidos en la sesión 1, de 277 participantes. Para cada participante se produjeron dos redes cerebrales funcionales separadas con resoluciones de 160 × 160 (LR) y 268 × 268 (HR) utilizando dos enfoques de parcelación cerebral global [11]. En la Figura 5 se puede ver una muestra de los datos utilizados, son dos individuos que se observan tanto en alta (HR) como en baja (LR) resolución.

III. AGSR-Net

A continuación se describe la arquitectura AGSR-Net [4] y se discuten algunas modificaciones implementadas.

Un conectoma puede ser representado como un grafo $C = \{V, E, X\}$, donde V son los nodos y E son las aristas. Los nodos son definidos para cada ROI del cerebro.

La matriz de adyacencia $A \in \mathbb{R}^{N\times N}$ (N es el número de nodos) denota la intensidad de conectividad entre la ROI i y j en el valor A_{ij}, usando una métrica específica (ej: actividad neuronal, similaridad en morfología cerebral).

$X \in \mathbb{R}^{N\times F}$ denota la matriz de características de cada nodo, donde N es el número de nodos y F el número de características por nodo.

Cada sujeto de entrenamiento s en el dataset es representado por 2 matrices de conectividad en los dominios de LR y HR, denotados como $C_l = \{V_l, E_l, X_l\}$ y $C_h = \{V_h, E_h, X_h\}$ respectivamente.

Dado un grafo C_l el objetivo es aprender un mapa $f : (A_l, X_l) \to (A_h, X_h)$ obteniendo un grafo C_h. La arquitectura AGSR-Net puede verse en la Figura 6.

La arquitectura tiene un generador (tres primeras etapas) y discriminador:

- Un **Graph U-Autoencoder** que toma la matriz de adyacencia A_l y la matriz de características de los nodos X_l del cerebro LR donde $X_l = I_d$ y aprende el embedding de la matriz de características de los nodos Z_l. $(A_l, X_l) \to Z_l$.
- Una capa **Graph Super-Resolución** (GSR) que toma la matriz de adyacencia A_l y el embdedding Z_l obtenido en el paso anterior del cerebro LR y lo mapea a una matriz de adyacencia $\tilde{A}_h$ y una matriz de embedding de características $\tilde{X}_h$ del cerebro en HR. $(A_l, Z_l) \to (\tilde{A}_h, \tilde{X}_h)$.
- 2 capas **GCN** (graph convolutional network) para propagar las relaciones topológicas entre los nuevos nodos $\tilde{A}_h$ y la matriz de embedding de características aprendida $\tilde{X}_h$ en HR. $(\tilde{A}_h, \tilde{X}_h) \to Z_h$.
- Un discriminador que clasifica si los datos del generador son reales o no.

III-A. Bloque Graph U-Autoencoder

El **U-Autoencoder** incluye inicialmente capa GCN para aprender características de los nodos en LR. El input es $(A_l, X_l = I_d)$ y la salida $Z_0 \in \mathbb{R}^{N\times NK}$ es el embeding (inicial) de la matriz de características con NK número de características por nodo, siendo K el factor de resolución a incrementar para predecir un grafo HR de uno LR.

La regla de propagación de la capa inicial GCN es:

$$Z_0 = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}X_l W_l)$$

donde $\hat{D}$ es la matriz diagonal de grados, $\hat{A} = A + I$ matriz de adyacencia unido con loops y σ función de activación. W_l es una matriz de pesos entrenable.

Luego, para aprender el embedding de las caracteríticas de los nodos dado un conectoma C_l, se utiliza un Graph U-Autoencoder compuesto por un **Graph U-Encoder** y un **Graph U-Decoder**.

$$Zl = GraphU\,Autoencoder(A_l, Z_0)$$

III-A1. Graph U-Encoder: El input del encoder es una matriz de adyacencia $A_l \in \mathbb{R}^{N\times N}$ del grafo $C_l = \{V_l, E_l, X_l\}$ donde $X_l \in \mathbb{R}^{N\times F}$ es la matriz de características de los nodos. En ausencia de características de los nodos originales, asignamos una matriz identidad $I_N \in \mathbb{R}^{N\times N}$ a la matriz X_l.

Construimos el **U-encoder** concatenando múltiples módulos de encoding, los cuales contienen un **graph pooling** seguido de una **GCN**. La capa de graph pooling forma un nuevo grafo del cerebro más pequeño seleccionando adaptivamente los nodos y la capa GCN agrega (promedio local) las características de los nodos vecinos basado en sus pesos de conectividad.

La capa **graph pooling** adaptativa selecciona un conjunto de nodos desde un grafo más chico basado en una proyección escalar sobre un vector entrenable u. Esto significa decrementar el número de ROIs seleccionando los nodos más significativos. **La regla de propagación** se define como:

- Encontrar la proyección escalar de X_l con el vector u obteniendo un vector $v \in \mathbb{R}$, donde v_i es la proyección escalar de cada nodo sobre el vector u.
$$v = \frac{X_l^{(l)} u^{(l)}}{\|u^{(l)}\|}$$
- Tomar los k valores más altos en v y guardar índices. $indices = rank(v, k)$
- Extraer de la matriz de características, las filas que se encuentran en los índices elegidos. Obteniendo la matriz de adyacencia respectiva.
$$\tilde{X_l}^{(l)} = X_l^{(l)}(indices, :)$$
$$A_l^{(l+1)} = A_l^{(l)}(indices, indices)$$
Esto reduce el tamaño del grafo de N a k, $A_l^{(l+1)} \in \mathbb{R}^{k\times k}$.
- Aplicamos una **sigmoide** al vector proyectado v en los índices seleccionados.
$$\tilde{v} = sigmoid(v(indices)) \qquad \tilde{v} \in \mathbb{R}^k$$
- Multiplicar $\tilde{v}$ por 1_F^T (vector con F elementos igual a uno), luego multiplicar elemento a elemento con la matriz de características extraída obteniendo una matriz de características con los nodos seleccionados $X_l^{(l+1)} \in \mathbb{R}^{k\times F}$.
$$X_l^{(l+1)} = \tilde{X}_l^{(l)} \odot (\tilde{v} 1_F^T)$$

Adicionalmente se plantea una modificación a la propuesta de Graph Pooling donde se modifica la etapa inicial y se remueve la utilización de la sigmoide. La modificación del paso inicial se basa en la SVD y en la utilización del concepto de residuo [12] dado que no es posible utilizar los vectores singulares derechos (ni izquierdos) debido a que los valores singulares de la matriz de embedding son degenerados. La idea es seleccionar los nodos (filas de X) que se alinean más cerca de los k más importantes componentes principales. Solo se detalla esta modificación, los pasos restantes son iguales:

- Obtener la SVD de $X^{(l)} + [A_{LR}, A_{LR}]$ (concatenación de matriz A_{LR}).
- Proyectar X sobre el top k de componentes principales

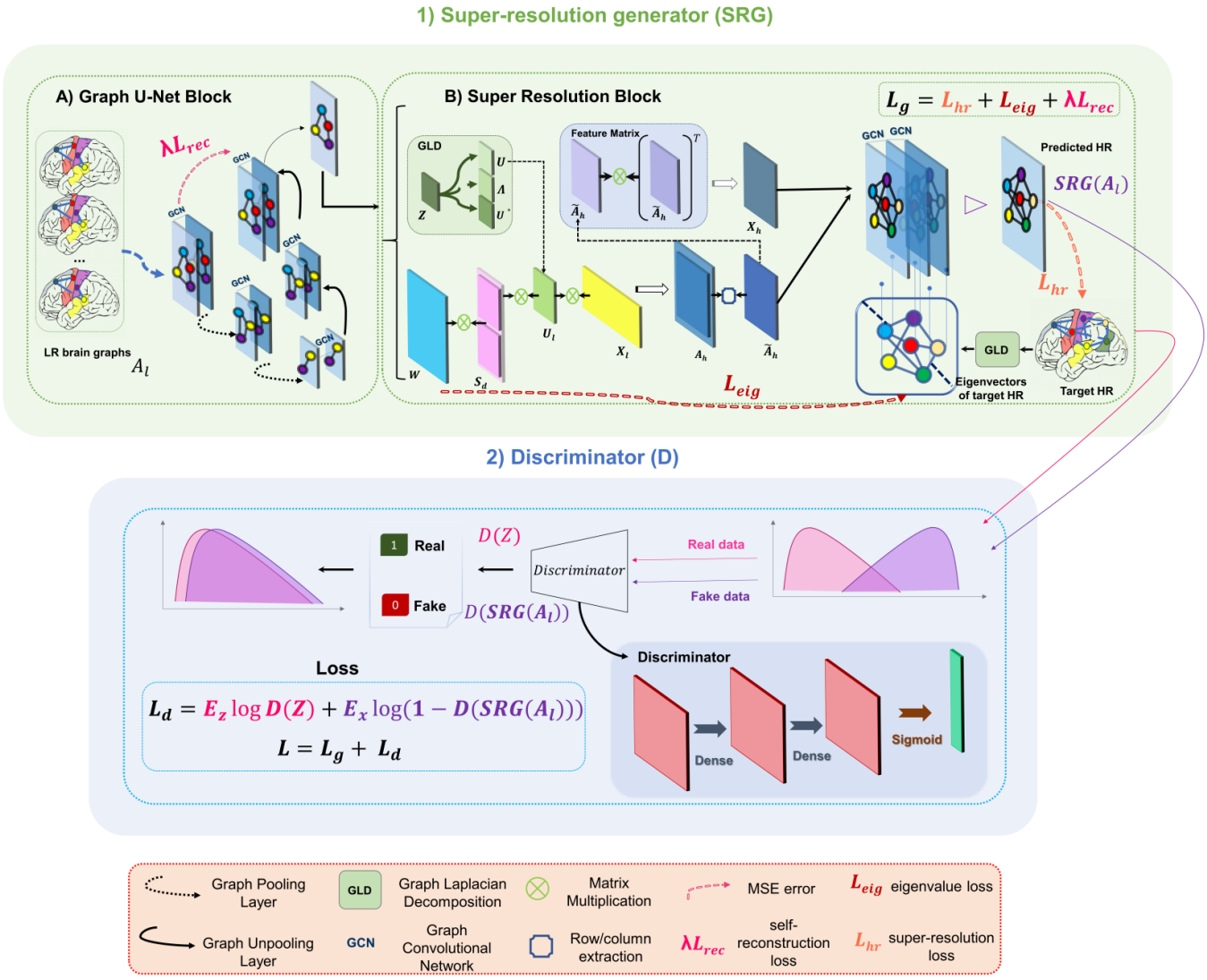


Figura 6: AGSR-Net.

- Medir la *alineación* como la magnitud de las proyecciones. Se usa la norma euclidiana de la proyección de cada fila para determinar la *influencia*. Esto es generalizable a otra medida de influencia.
- Rankear los nodos (filas) según la magnitud de la proyección, donde mayor magnitud implica mayor influencia.

Por último, también se pueden modificar las capas GCN por cualquier tipo de capa, acá se probaron las GAT o GATV2.

III-A2. Graph U-Decoder: El **U-Decoder** se compone de una concatenación de bloques de decoding, los cuales contienen una capa de **graph unpooling** seguido de una capa **GCN**. Cada módulo actúa como la operación inversa al módulo de coding, el pooling restaura la estructura original del grafo y la GCN agrega información de la vecindad a cada nodo.

La capa **graph unpooling** retrae la operación de graph pooling anterior, realojando los nodos en sus posiciones originales según los índices guardados.

$$X^{(l+1)} = relocate(0_{N\times F}, X_l^{(l)}, indices)$$

Con $0_{N\times F}$ matriz de características reconstruida del nuevo grafo (inicialmente la matriz está vacía). $X_l^{(l)} \in \mathbb{R}^{k\times F}$ matriz de características reducida y la operación de realocate asigna los vectores fila de $X_l^{(l)}$ en $0_{N\times F}$ acorde a los índices.

III-A3. Optimización: Para que el embedding del conectoma LR preserve la estructura topológica A_l y la información X_l, se pide que el embedding de la matriz de características Z_l coincida con la matriz Z_0.

En la función de pérdida se incorpora una regularización de autoreconstrucción que minimiza el error cuadrático medio (MSE) entre la representación Z_0 y la salida Z_l. Notar que esta pérdida cuadrática promedio, no es otra cosa que la norma de Frobenius normalizada por la cantidad de nodos.

Sin embargo, esto lo que busca es que el resultado del graph unpooling sea similar al resultado de aplicar una GCN sobre (A, X) con $X = I$ al inicio. Esto no se traduce necesariamente en que dicho embedding represente correctamente (en una dimensión menor) a A. Se agrega una modificación sobre el planteo original, se calcula la descomposición en valores singulares de A y de Z_l, y se calcula la diferencia entre un

top k de los valores singulares de forma que la *energia* que representan sea similar, quedando la pérdida de reconstrucción de la siguiente forma:

$$\mathcal{L}_{rec} = \lambda \frac{1}{N} \sum_{i=1}^{N} \|Z_{0i} - Z_{li}\|_2^2 + (S_k^A - S_k^Z)$$

Otra es idea similar, es usar la norma operador. Calculando la diferencia entre la norma operador de A y Z_l o bien la norma de su diferencia.

III-B. Bloque de Super Resolución

Para la super resolución se utiliza una técnica basada en un upsampling que mantiene las características del grafo en el dominio frecuencial.

Sea $L_0 \in \mathbf{R}^{N \times N}$ y $L_1 \in \mathbf{R}^{NK \times NK}$ los laplacianos de los grafos del grafo original LR y el HR (upsampled) respectivamente (K es el factor de incremento del grafo).

Sean las descomposiciones en valores propios:

$$L_0 = U_0 \Lambda U_0^* \qquad L_1 = U_1 \Lambda U_1^*$$

$U_0 \in \mathbb{R}^{N \times N}$ y $U_1 \in \mathbb{R}^{NK \times NK}$.

En forma matricial el grafo upsampled puede definirse como: $x_u = U_1 S_d U_0^* x$. Donde $S_d = [I_{N \times N} I_{N \times N}]^T$, x es la señal de input del grafo y x_u es la señal upsampled.

Generalizando a una matriz de señal $X_l \in \mathbb{R}^{N \times F}$ con F canales de entrada como:

$$\tilde{A}_h = U_1 S_d U_0^* X_l$$

Para generar un grafo de resolución $\{NK\}^2$ los inputs F de X_l deben ser NK.

III-B1. Super-resolución de la estructura del grafo: Para predecir $\tilde{A}_h$, se toma la matriz de características del grafo LR aprendida en el autoencoder y la descomposición en valores propios del Laplaciano del grafo LR.

Para forzar que la descomposición de valores propios del Laplaciano del grafo HR coincida con con el grafo objetivo HR, se introduce una matriz de parámetros entrenables $W \in \mathbb{R}^{NK \times NK}$. El entrenamiento busca la minimización del error entre la matriz de pesos W y el vector propio U_1 de la matriz de adyacencia objetivo A_h. La **regla de propagación** para la capa es:

$$\tilde{A}_h = W S_d U_0^* Z_l$$

III-B2. Super-resolución de las características del grafo: Luego de expandir el tamaño del grafo LR, la representación de los nuevos nodos no tiene significado y se quiere asignar un vector de características a cada nuevo nodo.

Al agregar nuevos nodos y aristas, es probable que algunos nodos y aristas queden aislados causando pérdida de información en las capas subsecuentes. Para evitarlo, se inicializa la matriz de características objetivo $\tilde{X}_h$ como:

$$\tilde{X}_h^{(l)} = \tilde{A}_h^{(l)} (\tilde{A}_h^{(l)})^T$$

Esta operación relaciona nodos de a lo sumo 2 pasos de distancia e incrementa la conectividad entre ellos, esto bien puede generalizarse a a relacionar nodos a lo sumo a k caminos de distancia. A cada nodo le es asignado un vector de características que cumple dicha propiedad. Se simetriza la matriz de adyacencia y de características dado que las adyacencias reales son simétricas.

$$\tilde{A}_h = \frac{\tilde{A}_h + \tilde{A}_h^T}{2} \;\; , \;\; \tilde{X}_h = \frac{\tilde{X}_h + \tilde{X}_h^T}{2}$$

Otras formas de simetrizar una matriz son posibles, por ejemplo, se puede reflejar la triangular superior o inferior sobre la diagonal. Tomar el mínimo o el máximo entre A y A^T, tomar $\sqrt{AA^T}$.

III-B3. Optimización: Para entrenar los filtros de super-resolución, minimizamos el error cuadrático medio entre los pesos y los vectores propios U_1 de l matriz A_h (MSE).

$$\mathcal{L}_{eig} = \frac{1}{N} \sum_{i=1}^{N} \|W_i - U_{1i}\|_2^2$$

III-C. Graph embedding layers

Posterior a la GSR, para aprender un embedding más representativo de las ROI del grafo HR, se añaden 2 capas GCN.

$$Z_h^0 = GCN(\tilde{A}_h, \tilde{X}_h) \;\; , \;\; Z_h = GCN(\tilde{A}_h, \tilde{Z}_h^0)$$

Para cada nodo, estas capas agregan los vectores de características de sus vecinos, para traducir los pesos de la conectividad a características de nodos del grafo HR.

La salida de este paso es la predicción final de toda la GSR-Net, el conectoma HR a partir del conectoma LR de entrada.

Sin embargo las predicciones Z_h tienen tamaño $NK \times NK$ y el grafo HR puede no satisfacer dichos tamaños. Para solucionarlo se puede agrega padding en la matriz de adyacencia del HR en el entrenamiento y remover el padding extra en el paso de la evaluación de la Loss y en la predicción final.

III-D. Optimización

El proceso de entrenamiento es guiado por la minimización del MSE entre el grafo HR predicho por la red y el orinial.

La función de Loss total está compuesta por la loss de recontrucción, de descomposición de vectores propios y de super-resolución:

$$\mathcal{L}_{hr} + \mathcal{L}_{eig} + \lambda \mathcal{L}_{rec} + (S_k^A - S_k^Z)$$

$$= \frac{1}{N} \left(\sum_{i=1}^{N} \|Z_{hi} - A_{hi}\|_2^2 + \sum_{i=1}^{N} \|W_i - U_{1i}\|_2^2 + \lambda \sum_{i=1}^{N} \|Z_{0i} - Z_{li}\|_2^2 \right) + (S_k^A - S_k^Z)$$

III-E. Regularización adversaria de AGSR-Net

El enfoque se centra en introducir regularización al grafo HR generado a partir de un grafo LR forzando al grafo predicho a seguir la distribución del grafo HR real. Con la introducción de nuevos nodos y conectividad de aristas en la predicción de un grafo HR a partir de un grafo LR, es necesario que el grafo HR predicho preserve la distribución de los pesos de conectividad originales.

La capa adversaria está motivado por la red generativa adversaria (GAN) [13]. GAN comprende dos módulos que interactuan: el generador y el discriminador. La idea clave es hacer que los datos generados por el generador coincidan con la distribución de datos previos. Para lograr esto, el discriminador distingue si una muestra de entrada proviene de la distribución previa verdadera del HR o del generador. Simultáneamente, el generador se entrena para engañar al discriminador generando muestras que son indistinguibles de los datos que provienen de la distribución previa. El modelo del discriminador se construye sobre un perceptrón multicapa estándar: comprende dos capas densas y una capa sigmoide que devuelve un solo valor que clasifica la entrada como real o falsa. $D(A_h)$ es la estimación del discriminador de la probabilidad de que una instancia de un conectoma HR objetivo A_h es real y $D(SRG(A_l, X_l))$ es la estimación del discriminador de la probabilidad de que una instancia de un conectoma HR generado por SRG es real. El costo se puede calcular de la siguiente manera [13]:

$$\mathcal{L}_d = -\frac{1}{2}\mathbb{E}_{p_{\text{real}}}[\log D(A_h)] - \frac{1}{2}\mathbb{E}_{p_{\text{fake}}}[\log(1 - D(SRG(A_l, X_l)))]$$

El costo total es:

$$\mathcal{L} = \mathcal{L}_g + \mathcal{L}_d$$

Se puede formular la regularización del conectoma HR generado como un problema min-max, con la minimización del costo de cross-entropy para entrenar al discriminador de la siguiente manera:

$$\min_{SRG} \max_{D} \mathbb{E}_{p_{\text{real}}}[\log D(A_h)] + \mathbb{E}_{p_{\text{fake}}}[\log(1 - D(SRG(A_l, X_l)))]$$

IV. Resultados

En la Tabla I se ve la comparación respecto al MSE en el conjunto de test, los datos totales son $N = 277$, se usa un 30 % en test. Se fijaron los mismos valores de parámetros utilizados en el artículo de AGSRNet de forma que la comparación sea justa [5]. Se elige el Optimizador Adam con una tasa de aprendizaje de 0.0001 y el número de neuronas tanto en el Autoencoder como en las capas de convolución se establece en NK. El parámetro λ de la regularización es de 0.1.

Se probaron inicialmente distintas modificaciones una a una con el objetivo de ver cuales pueden lograr una mejora por si solas y también ver en que casos se empeora de forma *relevante*. Esto último puede ser importante en la medida que muestra que partes de la red pueden ser más sensibles implicando que cambios en esos componentes si bien ante las modificaciones planteadas empeoraron, en el futuro podrían mejorar.

Se cambio la forma de simetrizar la matriz ($\max(a_{ij}, a_{ji})$) y los resultados fueron prácticamente iguales.

Se cambiaron las capas finales de GCN, probando con una modificación de la GCN original (se agregan parametros), se eligieron también capas GAT y GATV2 y tampoco se obtuvieron mejores resultados.

En la capa U-net se modificaron todas las GCN (inicial, intermedias y final) por capas GAT sin obtener mejores resultados. En la capa U-net se modifico la capa de Graph Unpooling y los resultados obtenidos fueron prácticamente los mismos pero levente peores. Esto es interesante puesto que se puede modificar la forma de calcular la *influencia* y analizar si se generan mejoras.

La estructura del discriminador también se modifico y resultados fueron levemente inferiores.

En los únicos casos (cambios por separado) que se obtuvieron mejoras fue al modificar la función de pérdida del generador mediante la inclusión del componente asociado a los valores singulares. En dicho caso los resultados mejoraron levemente, el aprendizaje de la red tuvo un comportamiento similar y el error de test fue mejor como se puede ver en la Tabla I.

En la Tabla I se ve la comparación respecto al MSE en el conjunto de test. La convolución (GCN) por defecto es una variación de la TAGCONV genérica implementada en Pytorch Geometric [14], se implementa la GATV2 [15], GAT [15] y otra variación más flexible y con activación de la TAGCONV.

Nombre del Archivo	Error Cuadrático Medio
Modelo defecto	0.0198
Modelo defecto + simetría 2	0.0210
Modelo defecto + GCN modificada	0.0250
Modelo defecto + GAT	—
Modelo defecto + GATV2	0.042
Modelo defecto + Unet GAT	0.0228
Modelo defecto + Pérdida SVD	0.0191
Modelo defecto + GPooling	0.0199
Modelo defecto + Discriminador	0.0205

Cuadro I: Comparación de los valores promedio del Error Cuadrático Medio de test para diferentes modificaciones.

En la Figura 7 se ve la predicción de un cerebro en alta resolución a partir de grafo de baja resolución.

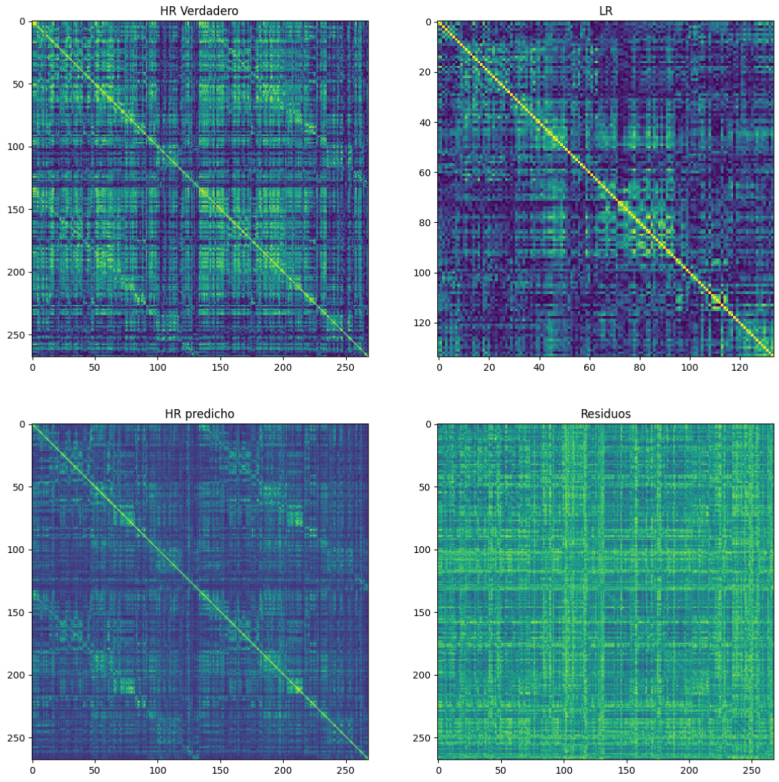


Figura 7: En la imagen se observa la predicción de un grafo de alta resolución a partir de uno de baja resolución. También se observan los residuos (errores) entre el grafo de alta resolución verdadero y el predicho

En la Figura 8 se muestra el aprendizaje de la red para algunas modificaciones. La forma de las curvas de las funciones de pérdida acumuladas no muestran el mismo comportamiento, en algunos casos la pérdida se estabiliza antes y si bien no lo mostramos en casi todos los casos en torno a las 100 épocas la pérdida vuelva a crecer y el modelo sobre ajusta empeorando el error de test. Por ejemplo, el error en test del modelo por defecto pasa de 0.01984 a 0.02096 al pasar de 100 a 200 épocas. La primera iteración no se agrega puesto que el error es muy grande y la escala no deja ver la evolución de las curvas de error. En todos los casos, el error disminuye de forma monótona hasta cerca de la época 50 y luego sigue disminuyendo aunque se observan leves aumentos.

V. Discusión

Sería interesante probar esta estructura de red (o sus mejoras) en datos de personas de otro rango etario, de forma de verificar cuanta validez externa tienen los resultados. Recordemos, que tanto los datos de entrenamiento como test corresponden a personas de entre 17 y 27 años de edad. Por otro lado, en la arquitectura de GraphUnet, es posible modificar la capa de Unpooling, una posible extensión es adaptar la propuesta de Unpooling Layer for Graph Generation [9] donde se propone una capa de Unpooling para la generación de grafos, si bien dicha capa requiere como input un grafo con características (features), la GrapUnet resuelve eso utilizando como input una matriz de identidad. Además, los autores muestran que el resultado del Unpooling genera un grafo conectado.

La modificación generada de la capa de Unpooling obtuvo casi los mismos resultados a los publicados, pero las magnitudes de las diferencias son pequeñas. Sería interesante buscar otra forma de hacer Graph Unpooling.

Se pueden realizar pruebas con distintas funciones de activación, nosotros usamos ReLU, LeakyRelu y sin activación. No observamos diferencias relevantes pero podría analizarse más en detalle. Dependiendo el tipo de datos, alguna podría ser mejor que otra. También se pueden hacer más pruebas con regularización mediante Dropout en distintas capas. Por ejemplo, para el mismo modelo, tomar una grilla y ver que efecto tiene.

Otra posible modificación es sobre la función de pérdida, los autores plantean la suma de tres pérdidas distintas pero entendemos se puede discutir más respecto a su formulación. Aquí se planteo una modificación que mejora los resultados obtenidos al tomar en consideración cuanto *estiran* el espacio los mapas utilizados.

La estructura adversaria (GAN) al final del modelo GSR-Net, mejora los resultados de la arquitectura GSRNet [5]. Si bien se realizaron modificaciones al discriminador utilizado por defecto, el mismo no obtuvo mejores resultados. Pero se podría profundizar más al respecto.

Otra linea de trabajo, busca realizar solucionar el mismo problema pero utilizando modelos de difusión [10]. Sería interesante hacer una comparación con dichos métodos.

Finalmente, podría trasladarse todo el código a Pytorch geometric de forma que el mismo se ejecute más rápido, sea más fácil hacer modificaciones sobre las capas de convolución puesto que Pytorch Geometric tiene una implementación muy basta al respecto.

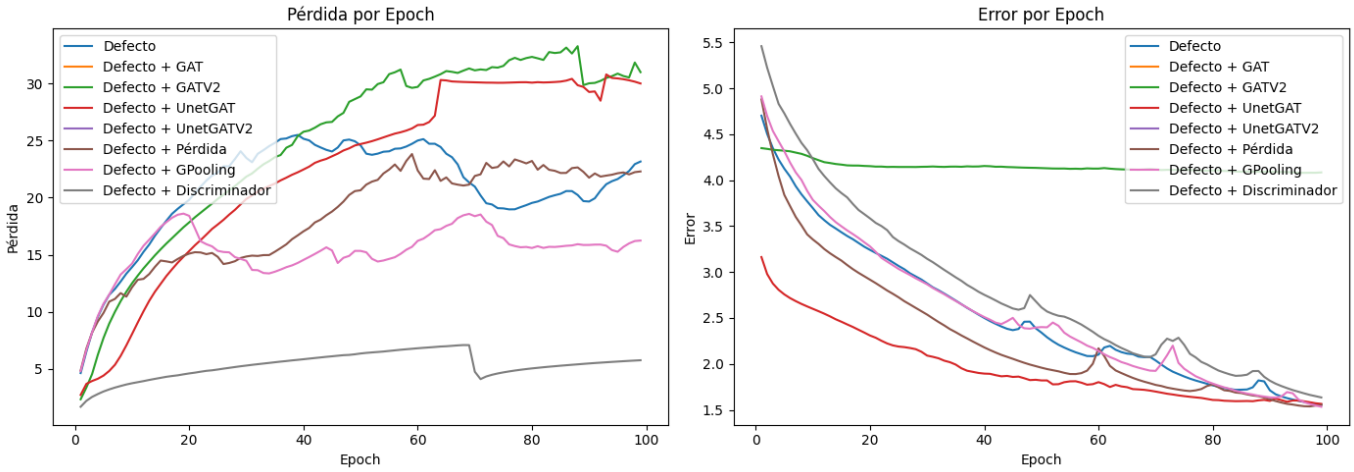


Figura 8: Evolución de la función de pérdida y error del modelo para cada época

APÉNDICE A
EJECUCIÓN

El proceso de entrenamiento del modelo comienza con un **pre-procesamiento** de los datos de los conectomas LR (160×160) y HR (268×268) de cada sujeto, obteniendo así un conjunto de matrices de adyacencia LR de entrada y un conjunto de matrices de adyacencia HR de salida en el conjunto de **Train** y **Test**.

Creamos el modelo **GSRNet** el cual contiene 4 capas:

- **GraphUnet:**
  1. Implementa la capa inicial GCN para obtener Z_0.
  2. Aplica el **U-Encoder** como varios bloques de GCN y pooling donde cada bloque reduce el número de nodos en función de un parámetro **ks** que es una lista de reales entre 0 y 1 que es cuanto se reduce la dimensión en cada coding.
     En nuestro caso usamos $ks = [0{,}9, 0{,}7, 0{,}6, 0{,}5]$
  3. Aplica el **U-Decoder** como la misma cantidad de bloques de GCN y unpooling.
- **GSRLayer:** Obtiene la descomposición en valores propios de la matriz de adyacencia y luego aplica la regla de propagación para obtener $\tilde{A}_h$ y luego obtiene $\tilde{X}_h$
- 2 **GraphConvolution:** Implementa una convolución en grafos, la cual contiene un dropout para regularizar en entrenamiento y la capa de activación final es una Relu.
- Elección del discriminador y la función de pérdida respectiva.

Luego se entrena el modelo dejando un 30 % de datos para test y **Adam** como optimizador.

Tenemos varios parámetros configurables para el entrenamiento:

Luego de entrenar, verificamos como funciona el modelo, testeando con el conjunto de test. Para cada matriz de adyacencia en LR predecimos su matriz HR y calculamos su MSE.

Parámetro	Descripción
epochs	Número de épocas del entrenamiento
lr	Learning rate de Adam
splits	Número de folds de la validación cruzada
lmbda	hyper parámetro de la Loss de self-reconstruction
hidden_dim	número de hidden GCN layer neurons
mean_gaussian	media distribución normal
std_gaussian	desvío distribución normal
dropput_rate	probabilidad de dropout
padding	dimensión del padding

Cuadro II: Tabla de algunos parámetros en el entrenamiento

REFERENCIAS

[1] O. Sporns, G. Tononi, and R. Kötter, "The human connectome: A structural description of the human brain," *PLoS Computational Biology*, vol. 1, no. 4, pp. 0245–0251, 2005.

[2] A. Bessadok, M. A. Mahjoub, and I. Rekik, "Graph Neural Networks in Network Neuroscience," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5833–5848, 2023.

[3] A. Fornito, A. Zalesky, and E. T. Bullmore, *Fundamentals of Brain Network Analysis*. Academic Press, 2016.

[4] M. Isallari and I. Rekik, "Brain Graph Super-Resolution Using Adversarial Graph Neural Network with Application to Functional," *Medical Image Analysis*, 2021.

[5] ——, "Graph Super-Resolution Network for predicting high-resolution connectomes from low-resolution connectomes," in *International Workshop on PRedictive Intelligence In MEdicine*, 2020, pp. 139–149.

[6] H. Gao and S. Ji, "Graph U-Nets," *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[7] W. Weng and X. Zhu, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *IEEE Access*, vol. 9, pp. 16 591–16 603, 2021.

[8] Y. Tanaka, "Spectral domain sampling of graph signals," *IEEE Transactions on Signal Processing*, vol. 66, no. 14, pp. 3752–3767, 2018.

[9] Y. Guo, D. Zou, and G. Lerman, "An Unpooling Layer for Graph Generation," *Proceedings of Machine Learning Research*, vol. 206, pp. 3179–3209, 2023.

[10] N. Rajadhyaksha and I. Rekik, "Diffusion-based graph super-resolution with application to connectomics," in *Predictive Intelligence in Medicine*, I. Rekik, E. Adeli, S. H. Park, C. Cintas, and G. Zamzmi, Eds. Cham: Springer Nature Switzerland, 2023, pp. 96–107.

[11] W. Liu, D. Wei, Q. Chen, W. Yang, and J. Meng, "Data Descriptor : Longitudinal test-retest neuroimaging data from healthy young adults in southwest China," *Nature*, pp. 1–9, 2017.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[13] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[14] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, pp. 1–14, 2017.
[15] P. Veličković, A. Casanova, P. Liò, G. Cucurull, A. Romero, and Y. Bengio, "Graph attention networks," *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, pp. 1–12, 2018.