

A todas aquellas personas que han dedicado parte de su tiempo a mi desarrollo
tanto personal como profesional.

Juan Carlos Martinez

Acrónimos

ADNI	Alzheimer's Disease Neuroimaging Initiative
AD	<i>Alzheimer Disease</i>
CMV	Voto por Mayoría Complejo
CNN	Red Neuronal Convolucional
CSF	Fluido Cerebro-Espinal
GM	Materia Gris
PCA	Análisis de Componentes Principales
PET	Tomografía por Emisión de Positrones
MCI	<i>Mild Cognitive Impairment</i>
MRI	Imagen Resonancia Magnética
NC	<i>Normal Control</i>
ROI	Régiones de Interés
SGD	Descenso en Gradiente Estocástico
SMV	Voto por Mayoría Simple
SVM	Máquina de Vectores de Soporte
VAE	Autoencoder Variacional
CVAE	Autoencoder Variacional Convolucional
WM	Materia Blanca

Índice

Acrónimos	III
Resumen	1
Summary	3
1 Introducción	5
1.1 Motivación	6
1.2 Objetivos	8
2 Contexto	9
2.1 La enfermedad del Alzheimer	9
2.1.1 Fases de la AD	10
2.1.2 Síntomas	12
2.1.3 Histopatología	13
2.2 Base de datos ADNI	15
2.2.1 Imágenes médicas	15
2.3 Estado del Arte	18
2.3.1 Modelos Generadores	18
2.3.2 Diagnóstico Asistido por Computador de AD	25
2.3.3 Métodos de Aprendizaje Profundo	26
2.4 Entorno de desarrollo	29
3 Fundamentos Teóricos	31
3.1 Autoencoder Variacional	31
3.1.1 Modelo de Variables Latentes	31
3.1.2 Modelo Probabilístico	32
3.1.3 Función Objetivo	33
3.1.4 Optimización de la función objetivo	34
3.1.5 El truco de Reparametrización	36
3.1.6 Interpretación de la función objetivo	37

3.1.7	Codificación y Decodificación	38
3.2	Redes Neuronales	40
3.2.1	Red Neuronal Densa	40
3.2.2	Red Neuronal Convolutacional	45
3.3	Herramientas Complementarias	50
3.3.1	Máquina de Vectores de Soporte	50
3.3.2	Métricas de Evaluación Estadística	51
3.3.3	Validación Cruzada	54
4	Trabajo Realizado	55
4.1	Estudio basado en régiones cerebrales	55
4.2	Tratamiento de Neuroimágenes	56
4.2.1	Fuente de Datos	56
4.2.2	Procesado Previo	57
4.2.3	Segmentación basada en régiones	59
4.3	Autoencoder Variacional	65
4.3.1	Modelo de grafos	65
4.3.2	Fase de Codificación	66
4.3.3	Fase de Reconstrucción	69
4.3.4	Evaluación del error	70
4.3.5	Entrenamiento	72
4.3.6	Almacenamiento del Grafo	75
4.3.7	Visualización del Grafo con <i>Tensorboard</i>	76
4.4	Autoencoder Variacional Convolutacional	78
4.4.1	Diseño del Modelo	79
4.4.2	Fase de Codificación	80
4.4.3	Truco de Reparametrización	82
4.4.4	Fase de Decodificación	83
4.4.5	Fase de Estimación del error	85
4.5	Modelo de Clasificación	86
4.5.1	Extracción de Características por Región	87
4.5.2	Evaluación por Región	89
4.5.3	Evaluación Conjunta	90
5	Pruebas y Resultados	95
5.1	Evolución de la reconstrucción de una region	95
5.1.1	Ejemplo con VAE	95
5.1.2	Ejemplo con CVAE	97
5.2	Reconstrucción completa del cerebro	99
5.2.1	Ejemplo con CVAE	100
5.2.2	Ejemplo con VAE	102

5.3	Visualización del código de la capa latente	104
5.4	Exploración del código de la capa latente	106
5.5	Clasificación	109
5.5.1	VAE con MRI	110
5.5.2	VAE con PET	111
5.5.3	CVAE con MRI	112
5.5.4	CVAE con PET	113
Conclusiones y líneas futuras		115
5.5.5	Líneas futuras	116
Índice alfabético		125

Índice de figuras

1.1	Esquema clásico de auto-encoder	7
2.1	Evolución del número de paciente en millones desde 2010 hasta 2050 [25]	9
2.2	Ovillos neurofílare y placas seniles de los pacientes con AD respecto a los normales. Figura obtenida de [32]	13
2.3	Ovillos neurofílare y placas seniles de los pacientes con AD respecto a los normales. Figura obtenida de [32]	13
2.4	(a) Cerebro con funciones cognitivas normales. (b) Cerebro con funciones cognitivas d	16
2.5	(Izquierda) Modelo de una única Gausiana, (Derecha) Modelo de mezcla de Gausianas	18
2.6	(Izquierda) Modelo de una única Gausiana, (Derecha) Modelo de mezcla de Gausianas	19
2.7	Módelo básico de RBM	20
2.8	<i>Stacked</i> RBMs	20
2.9	Autoencoder Lineal	21
2.10	Esquema de una Red Generativa Adversaria	23
2.11	Visualización del espacio de dimensionalidad reducida conseguido en el trabajo [45]. Las imágenes superpuestas corresponden a las imágenes reales, no a las generadas	24
2.12	Esquema clásico básico de un sistema de diagnóstico de AD asistido por computador	25
2.13	Diagrama del Autoencoder Convolutacional 3D empleado en el trabajo [54] para la extracción de características	27
2.14	Diagrama general del método desarrollado en el trabajo [55]	28

3.1	Modelo gráfico de variables latentes para el modelo generativo del VAE. Z es el espacio de variables lo más similar posible a un distri-	32
	bución normal ($N(0, I)$). El elemento θ es el conjunto de parámetros que aplicados de manera funcional sobre las variables latentes son capaces de generar el conjunto muestral X	
3.2	(Izquierda) Modelo de VAE sin Truco de Reparametrización. (Dere- cha) Modelo de VAE con Truco de Reparametrización	36
3.3	Esquematización simple de las funciones del Codificador y el Decodi- ficador en el VAE	38
3.4	Sistema global de proceso de una red neuronal	40
3.5	Principales funciones de activación.	41
3.6	(izquierda) Función de activación <i>Relu</i> . (Derecha) Función de activa- ción <i>leakyRelu</i>	42
3.7	Esquema de una red neuronal densa de una sola capa oculta	43
3.8	Representación esquemática del proceso de entrenamiento de una red neuronal	43
3.9	Red convolucional <i>LeNet5</i>	45
3.10	Ejemplo de aplicación del operador convolución sobre una imagen. Seleccionada una región de la imagen cuyas dimensiones son las mis- mas que las del kernel seleccionado, se aplica el producto pixel a pixel entre dicha región y los pesos propios del kernel. La suma de estos productos se almacena en la imagen de salida, respetando la ubicación espacial de la región evaluada.	45
3.11	(Izquierda) Modelo clasico de redes neuronales. (Derecha) Modelo de Red Convolutacional. Los datos son reagrupados en 3 dimensionados como se puede observar en una de las capas. Cada una de las capas tiene como entrada una imagen 3D y tiene como salida otra imagen 3D. La capa roja representa la capa de entrada por lo que la altura y la anchura son las dimensiones de la imagen y la profundidad son el número de canales	46
3.12	Representación de la conectividad local en una red neuronal	47
3.13	Desplazamiento del campo de recepción	47
3.14	Representación de la extracción de varias características con varios filtros	48
3.15	Representación del proceso de agrupamiento (<i>pooling</i>)	49
3.16	Representación de la separación binaria realizada por SVM	50
3.17	Representaciónd de la curva ROC	53
3.18	Proceso de Validación Cruzada	54
4.1	MRI image (a), MRI atlas (b), (c) PET image and (d) PET atlas (same slice is shown in MRI and PET images)	56

4.2	Muestra de neuroimagen MRI segmentada. (zquierda) MRI WM ima- ge. (Derecha) MRI GM image	57
4.3	Muestra de neuroimagen PET normalizada	58
4.4	Diagrama del proceso de vectorización de neuroimágenes 3D, con la posterior división en Vóxeles Útiles y Vóxeles de <i>background</i>	60
4.5	Elementos del Stack de Imágenes	60
4.6	Diagrama de Atlas AAL de Régiones	62
4.7	Proceso de segmentación de vectores por región	63
4.8	Proceso de segmentación de vectores por región	64
4.9	Ejemplo de Régiones de imágenes PET segmentadas.(Izquierda) Re- gión Nº 20. (Derecha) Región Nº 30. Capturas de imágenes 3D toma- das con el programa <i>MRIcrGL</i>	64
4.10	Esquema general del grafo elaborado en tensorflow	65
4.11	Esquema de Fase de Codificación	66
4.12	Ejemplo de Conexionado Básico con la función de activación empleada	67
4.13	Diagrama del Truco de Reparametrización	68
4.14	Diagrama de la fase de reconstrucción o decodificación del VAE . .	69
4.15	Funciones de Activación Empleadas	70
4.16	Diagrama de los elementos del sistema sobre los que se calcula cada tipo de error	71
4.17	Diagrama de boques funcionales del proceso de actualización de los parámetros del VAE	73
4.18	Diagrama del proceso iterativo de entrenamiento	75
4.19	Diagrama de Grafos representado por <i>Tensorboard</i>	77
4.20	Diagrama de la secuencia de bloques funcionales implementados en el CVAE	78
4.21	Captura de la representación gráfica de <i>Tensorboard</i> del modelo con- volucional implementado para el CVAE	79
4.22	Captura de la representación gráfica de <i>Tensorboard</i> del bloque fun- cional de codificación para el CVAE	81
4.23	Función de activación <i>lRelu</i>	82
4.24	Captura de la representación gráfica de <i>Tensorboard</i> de la implemen- tación del truco de reparametrización para el CVAE	83
4.25	Captura de la representación gráfica de <i>Tensorboard</i> del bloque fun- cional de decodificación para el CVAE	84
4.26	Captura de la representación de <i>Tensorflow</i> del bloque funcional de la estimación del error para el CVAE	85
4.27	Esquema básico de Autoencoder. Cabe notar como será el código generado en la capa latente lo que se empleará para la clasificación posterior	86

4.28	Diagrama básico del proceso de extracción de características aplicando el Autoencoder Variacional	87
4.29	Diagrama básico del proceso de extracción de características para las imágenes MRI	88
4.30	Proceso de evaluación por región basado en máquinas de vectores de soporte	90
4.31	Diagrama del proceso de clasificación basado en Régiones. El proceso señalado en rojo se corresponde con el proceso final encargado de la evaluación conjunta	91
4.32	Proceso de Evaluación Conjunta Alicando SVM	93
5.1	Evolución de la regeneración de la imagen 3D de la region nº3 del atlas AAL en función de la iteracción en la que se encuentre en el proceso de entrenamiento para un VAE.	96
5.2	Evolución de la regeneración de la imagen 3D de la region nº3 del atlas AAL en función de la iteracción en la que se encuentre en el proceso de entrenamiento para un CVAE.	98
5.3	Diagrama de la secuencia de pasos necesarios para llevar a cabo la síntesis completa del cerebro	99
5.4	Secciones sagital, horizontal y transversal de la neuroimagen reconstruida a la izquierda y de la neuroimagen original a la derecha. Se trata de una neuroimagen de un paciente AD y la reconstrucción ha sido realizada sobre un CVAE	100
5.5	Secciones sagital, horizontal y transversal de la neuroimagen reconstruida a la izquierda y de la neuroimagen original a la derecha. Se trata de una neuroimagen de un paciente AD y la reconstrucción ha sido realizada sobre un VAE	101
5.6	Secciones sagital, horizontal y transversal de la neuroimagen reconstruida a la izquierda y de la neuroimagen original a la derecha. Se trata de una neuroimagen de un paciente NOR y la reconstrucción ha sido realizada sobre un VAE.	102
5.7	Secciones sagital, horizontal y transversal de la neuroimagen reconstruida a la izquierda y de la neuroimagen original a la derecha. Se trata de una neuroimagen de un paciente AD y la reconstrucción ha sido realizada sobre un VAE.	103
5.8	Representación en tres dimensiones de la dispersión del código latente generado por un modeo VAE. Para la obtención de los valores de tres variables para el código de cada región se ha aplicado PCA sobre el código latente.	104

5.9	Diagrama del procedimiento de extrapolación sobre el código latente para cada una de las regiones. En el cuadro rojo se indica donde se realiza dicho proceso.	106
5.10	Secciones sagital, transversal y horizontal de dos neroimagenes originales, a la izquierda la de un sujeto NOR y la derecha un sujeto AD. La reconstrucción ha sido realizada sobre un VAE.	107
5.11	Secciones sagital, transversal y horizontal de dos neroimagenes reconstruidas con un modelo VAE, a la izquierda la de un sujeto NOR y la derecha un sujeto AD.	108
5.12	Resultados de clasificación sobre las imágenes MRI aplicando las tres técnicas de evaluacion conjunta de resultados. (Izquierda) Voto por Mayoría Simple. (Centro) Máquina de Vectores de Soporte. (Derecha) Voto por Mayoría Complejo.	110
5.13	Resultados de clasificación sobre las imágenes PET aplicando las tres técnicas de evaluacion conjunta de resultados. (Izquierda) Voto por Mayoría Simple. (Centro) Máquina de Vectores de Soporte. (Derecha) Voto por Mayoría Complejo.	111
5.14	Resultados de clasificación sobre las imágenes MRI empleando un CVAE y variando el tamaño del kernel empleado. En la figura se tiene una gráfica por cada una de las tres técnicas de evaluacion conjunta de resultados empleadas. (Izquierda) Voto por Mayoría Simple. (Centro) Máquina de Vectores de Soporte. (Derecha) Voto por Mayoría Complejo.	112
5.15	Resultados de clasificación sobre las imágenes MRI utiliando un CVAE y aplicando un barrido en el tamaño del kernel de Convolución. Cada imagen se corresponde con un método de evaluacion conjunta de resultados. (Izquierda) Voto por Mayoría Simple. (Centro) Máquina de Vectores de Soporte. (Derecha) Voto por Mayoría Complejo. . . .	113

Índice de Tablas

3.1	Tabla de Nomenclatura Estadística en Clasificación	52
4.1	Datos Demográficos Imágenes PET	57
4.2	Número de véxeles en imagen MRI por cada región	63
5.1	Tabla con la configuración de los distintos parámetros para la simulación de la figura 5.1	96
5.2	Tabla con la configuración de los distintos parámetros para la simulación de la figura 5.2	97

Resumen

Este proyecto estudió el uso de técnicas generativas aplicadas a la síntesis de neuroimágenes. La síntesis de neuroimágenes es un campo de un alto interés dado el elevado coste económico que tiene la adquisición de forma natural de este tipo de imágenes. Además, uno modelo eficaz de síntesis reduciría uno de los principales problemas que se encuentran los investigadores cuando se pretende generar modelos de diagnóstico de Alzheimer asistido por computador. Este problema, denominado maldición de la dimensionalidad, es debido a la pequeña cantidad de muestras de neuroimágenes en comparación a la alta dimensionalidad de dichas imágenes.

Además de evaluar la capacidad de síntesis, este proyecto analizó la capacidad de extracción de características del modelo generativo empleado, comprobando si la codificación realizada es capaz de extraer aquellos patrones que permiten distinguir entre sujetos sanos y sujetos con Alzheimer.

En este trabajo se ha empleado el modelo basado en aprendizaje automático conocido como autoencoder variacional. Este método ha sido introducido recientemente y presenta la novedad con respecto a modelos de autoencoder anteriores de generar un código latente variacional. Se trata de un código definido por una función normal de distribución, la cual dota de cierta libertad, dentro de los márgenes de la distribución, a la futura imagen regenerada a partir de dicho código.

Se han diseñado dos variantes basadas en el modelo del autoencoder variacional, una haciendo uso de redes neuronales densas y otra basada en redes neuronales convolucionales. Se ha evaluado tanto la capacidad de codificación así como la capacidad de regeneración de los dos modelos diseñados.

Los resultados obtenidos demuestran la capacidad del método para obtener las características principales de las neuronimágenes ya que se consiguen valores en la clasificación cercanos a los del estado actual del arte. No obstante, la síntesis de imágenes realizada resulta inefectiva ya que las imágenes generadas por un mismo modelo para distintas imágenes de origen generan imágenes muy similares en las que no se aprecian los detalles iniciales de las imágenes.

Summary

Este proyecto estudia el uso de técnica generativas aplicadas a la síntesis de neuroimágenes. La síntesis de neuroimágenes es un campo de un alto interés dado el elevado coste económico que tiene la adquisición de forma natural este tipo de imágenes. Además, uno modelo eficaz de síntesis reduciría uno de los principales problemas que se encuentran los investigadores a la hora de generar modelos CAD (Diagnóstico Asistido por Computador) conocido como maldición de la dimensionalidad.

A lo largo de este trabajo se analizará el modelo basado en aprendizaje automático conocido como autoencoder variacional. Este método de autoencoder ha sido introducido recientemente y presenta la novedad con respecto a modelos de autoencoder anteriores de generar un código latente variacional. Se trata de un código definido por una función normal de distribución, lo cual de dota de cierta libertad, dentro de los márgenes de la distribución, a la futura imagen regenerada a partir de dicho código.

Se diseñarán dos variantes basadas en el modelo del autoencoder variacional, una haciendo uso de redes neuronales densas y otra basada en redes neuronales convolucionales. Se evaluará tanto la capacidad de codificación así como la capacidad de regeneración de los dos modelos diseñados.

Capítulo 1

Introducción

La demencia engloba un amplio grupo de enfermedades mentales que provocan el deterioro progresivo de las facultades mentales de la persona que la padece tales como la memoria, el aprendizaje, el lenguaje o la orientación.

La enfermedad del Alzheimer (AD) es la forma de demencia más común en personas de la tercera edad. La AD es un desorden neurodegenerativo que afecta a la memoria en primer lugar, y progresivamente al resto de funciones cognitivas, provocando desajustes en el comportamiento de la persona que la padece. [1].

Actualmente esta enfermedad afecta a 30 millones de personas y se prevé que esta cifra alcance los 100 millones afectados en los próximos 50 años, por lo que además de ser un problema global de salud supone un reto socioeconómico para los países desarrollados y especialmente para aquellos países en vías de desarrollo.

La AD aún no tiene cura por ello es que la mayoría de las líneas de investigación relativas a esta enfermedad se centran en el diagnóstico temprano, con objeto de aplicar tratamientos que refuerzen el mantenimiento de la reserva cognitiva cerebral para la prevención de su avance. Esta reserva cognitiva es la resistencia de nuestro cerebro frente a esta enfermedad [2].

Las imágenes por resonancia magnética (MRI) son ampliamente empleadas como herramienta de soporte para el diagnóstico de problemas cerebrales, formando parte de la rutina habitual para el diagnóstico del Alzheimer. No obstante los cambios estructurales no pueden ser detectados hasta una etapa avanzada de la AD, por ello es que se han desarrollado técnicas de representación estructural más avanzadas como las imágenes volumétricas. Por otro lado, las imágenes funcionales del cerebro tales como la tomografía de emisión de positrones (PET) permiten identificar cambios más sutiles en el metabolismo del cerebro en una etapa más temprana de la enfermedad en comparación con las imágenes MRI [3].

Se conoce como Diagnóstico Ayudado por Computer (CAD, del inglés Computer Aided Diagnosis) al conjunto de técnicas que usan imágenes médicas cerebrales con objeto de detectar la AD en una etapa temprana de la enfermedad. Existen multitud

de aproximaciones, tanto empleando las clásicas imágenes MRI [4] o imágenes PET [5], o incluso ambos tipos de imágenes de forma combinada [6] [7] conocidas como modelos Multimodal.

Las técnicas CAD emplean diferentes procedimientos estadísticos capaces de extraer características relevantes de las imágenes y en última instancia determinar si la imagen pertenece a una persona que padece AD en función de dichas características relevantes [9] [10].

Uno de los principales problemas asociados al análisis estadístico de imágenes médicas es la maldición de la dimensionalidad (CoD, del inglés *Curse of dimensionality*). Este término fué ya expuesto en 1961 por Richard Bellman debido a los problemas encontrados en procesos de optimización [11]. La maldición de la dimensionalidad hace referencia a la aparente intratabilidad de sistematicamente obtener una función determinista sobre un espacio muestral de alta dimensionalidad, esto es, la inherente dificultad de integrar alta dimensionalidad en una única función [12]. Este problema ve acrecentada su repercusión debido a la escasez muestras, esto es lo que lo combierte en un ámbito de interés en el estudio de las imágenes cerebrales dado el número limitado de ejemplares.

Historicamente en el análisis estadístico de imágenes médicas cerebrales se han empleado técnicas de reducción de características capaces reducir la dimensionalidad del espacio muestral tales como Análisis de Componentes Principales (PCA), Análisis de Componentes Independientes (ICA) o *Sparse Filtering* [13].

En la actualidad un amplio colectivo científico involucrado en el diagnóstico temprano del AD ha desarrollado diferentes métodos de detección basados en Deep Learning [14] [15]. Se identifica por *Deep Learning* a una rama del aprendizaje automático que emplea redes neuronales de una o mas capas ocultas inspiradas con el propio cerebro humano. Esta técnica permite la modelación y abstracción de características complejas del espacio muestral sobre el que se aplica.

En la aproximación que se desarrollará en este trabajo se pretende aplicar técnicas de *Deep Learning* generativas tales como el Autoencoder Variacional [16]. Este método nos permite generar imágenes cerebrales no predefinidas, lo cual sería de utilidad a la hora de ampliar el espacio muestral tan limitado.

Motivacion

En este trabajo se empleará un auto-encoder, específicamente el autoencoder variacional, con objeto de generar sintéticamente imágenes médicas.

Se conoce como auto-encoder a una modelo estadístico que pretende de generar muestras de salida lo más parecidas posibles a las muestras de entrada dadas, esto es, con la menos distorsión posible, por lo que es necesario extraer las relaciones inherentes en el espacio muestral.

Aunque conceptualmente simples, han tomado un rol muy importante en el aprendizaje automático, afrotando el paradigma clásico de los sistemas de aprendizajes auto-organizables [17], capaces de adaptar su estructura en función de los datos de manera no supervisada.

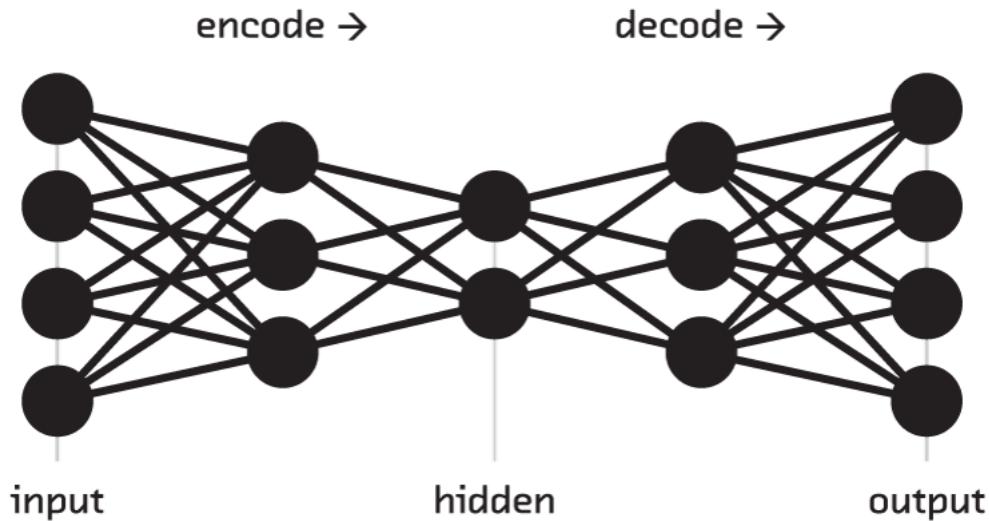


Figura 1.1: Esquema clásico de auto-encoder

Un auto-encoder está basado en el paradigma codificador-decodificador 1.1, donde el codificador se encarga de transformar la entrada en, típicamente, en una representación de baja dimensionalidad, mientras que el decodificador trata de usar esa salida de baja dimensionalidad para reconstruir la entrada original [18]. Esta capa intermedia recibe habitualmente el nombre de espacio latente.

La versión clásica de este modelo estadístico solo contenía una capa oculpa, la cual era una representación de baja dimensionalidad de los datos de entrada. En la última década, el auge de las redes neuronales profundas, ámbito más conocido por *Deep Learning*, ha provocado el uso de arquitecturas más complejas en los autoencoders, con varias capas ocultas tanto en el codificador como en el decodificador. En comparación con los modelos clásicos de auto-encoders se han mejorado ampliamente los resultados, aún cuando el número de parámetros a caracterizar es el mismo en ambos sistemas.

Un auto-encoder basado en aprendizaje profundo es capaz de extraer características de manera jerárquica gracias a sus distintas capas ocultas. Existen diferentes aproximaciones entre las que cabe destacar el auto-encoder de filtrado (*Denoising auto-encoder*) [19], el auto-encoder variacional (*Variational auto-encoder*) [16] o las

redes generativas adversarias (del inglés *Generative Adversarial Networks*).

Los auto-encoders variacionales constituyen una excelente herramienta para la extracción de las características principales o de patrones de un espacio muestral, pero también pueden ser considerados un modelo generativo[20], esto es, son capaces de generar datos sintéticos. Este modelo es capaz de asociar una distribución gaussiana a cada uno de los parámetros fundamentales extraídos por el propio sistema, esto es, es capaz de caracterizar estadísticamente a lo que anteriormente denominamos como espacio latente[16]. Esta capacidad es fundamental en cualquier sistema generativo.

Los modelos generativos tienen un amplio rango de aplicaciones como son la compresión, el filtrado, el aprendizaje no supervisado de características o la síntesis de datos.

Objetivos

Los principales propósitos del presente trabajo son los siguientes:

- Análisis e implementación en *Python* de un modelo de autoencoder variacional basado en redes neuronales densas.
- Análisis e implementación de un autoencoder variacional basado en redes neuronales convolucionales.
- Simular y evaluar la capacidad de síntesis de neuroimágenes de cada uno de los dos modelos generativos implementados. Debido a la alta dimensionalidad de las imágenes cerebrales empleadas, se tendrá que realizar la síntesis por cada región dado que el análisis completo de una imagen necesitaría unos recursos computacionales y de memoria que no están disponibles en los equipos en los que se realizan las pruebas.

La necesidad de analizar las regiones por separado conlleva que se necesiten un conjunto de funcionalidades que permitan manejar este tratamiento de las imágenes. Una de estas funcionalidades es la encargada de reconstruir el cerebro completo a partir de las imágenes 3D de regiones cerebrales en caso de utilizar el modelo convolucional.

- Eváluar la capacidad de los diseños implementados de distinguir entre los sujetos AD de los sujetos NOR, esto es la capacidad de clasificar de forma correcta las distintas neuroimágenes.

Capítulo 2

Contexto

La enfermedad del Alzheimer

En 1906 Alois Alzheimer, describió las lesiones cerebrales características del trastorno que recibió su nombre: placas seniles y ovillos neurofibrilares. La AD es ahora, 100 años después, la forma más común de demencia en el mundo, estando caracterizada por un espectro de características clínicas y fallos neuropatológicos [21].

Se han desarrollado numerosos estudios de investigación relativos a la naturaleza del AD, no obstante la motivación central ha sido señalar la AD como una categoría diferente de la demencia senil [22], así como determinar si la AD es la causa principal principal de la demencia en la tercera edad. Hay teorías que defienden que la AD es una consecuencia natural del envejecimiento, mientras que hay otras que definen lo contrario.

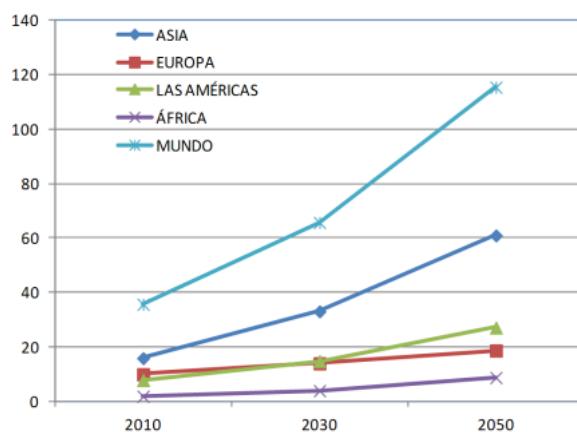


Figura 2.1: Evolución del número de paciente en millones desde 2010 hasta 2050 [25]

En la actualidad se postula desde la perspectiva de la mitocondria, orgánulo encargado de la respiración celular [23] y que toma un papel primordial en el desarrollo de la AD y del propio envejecimiento cerebral. La hipótesis en cascada de la mitocondria [24] postula que hay mecanismos comunes que conducen al envejecimiento cerebral y a la AD, así como que la producción de placas, ovillos neurofibrilares y la degeneración sináptica son consecuencias de la funcionalidad perturbada de la mitocondria.

La AD es uno de los desórdenes neurodegenerativos más severos y frecuentes en la población de la tercera edad teniendo severas repercusiones tanto para la salud como socioeconómicas. El impacto esperado de esta enfermedad se ve incrementado debido al aumento de la esperanza de vida, se estima que durante los próximos 20 años se duplicará el número de pacientes de dicha enfermedad principalmente en los países más desarrollados. La figura 2.1 muestra la evolución del número de pacientes de AD hasta el 2050, en función de los continentes.

Fases de la AD

Síntomas imperceptibles

La primera de las señales tiene que ver con el descenso de los niveles de la proteína beta amiloide en el líquido cefalorraquídeo (LCR). Este proceso se puede detectar hasta 25 años antes del inicio de la pérdida de la memoria mediante una resonancia magnética. Esta proteína es la causante de la formación de las placas seniles. Durante esta fase previa a la pérdida de la memoria se hacen perceptibles las alteraciones en las estructuras tanto en las estructuras cerebrales como en el hipocampo. Es por ello que los síntomas se producen varios años de que puedan ser percibidos por la propia persona o por sus familiares.

Predemencia

Esta fase es usualmente identificada como deterioro cognitivo o conductual leve. Los primeros síntomas perceptibles son a menudo confundidos con la propia vejez de la persona. Una evaluación neuropsicológica detallada es capaz de determinar evidencias de AD hasta 8 años de que se cumplan los criterios de diagnóstico[26].

La deficiencia más relevante es la pérdida de memoria, ya sea como la incapacidad de adquirir nueva información o la imposibilidad de recordar hechos recientes. No obstante pueden aparecer dificultades leves en funciones ejecutivas como la atención o el razonamiento, así como trastornos en la memoria semántica [27].

Demencia Inicial

El principal síntoma asociado a esta fase inicial es la pérdida de memoria puntual o incluso una pérdida de la memoria conocida a corto plazo, la cual supone dificultades para el paciente en la iteracción con familiares o amigos. Una pequeña porción de los pacientes sufre de dificultades con el lenguaje, con el reconocimiento de las percepciones o con la ejecución de movimientos. [29]

La capacidad de aprender nuevos conceptos ya sean abstractos o recuerdos reales, esto es, la memoria a corto plazo, es la que se ve más afectada durante esta fase frente a otras capacidades que se ven afectadas en menor medida como es la memoria a largo plazo, la memoria semántica o la memoria implícita, la cual hace referencia al conocimiento de como realizar acciones con el propio cuerpo. [28]

Demencia morerada

El síntoma diferencial de esta fase con respecto a la anterior son los cambios de conducta inesperada, incluso arranques violentos en personas que nunca han experimentado este comportamiento. Las manifestaciones neuropsiquiátricas más comunes son las distracciones, el desvarío y los episodios de confusión al final del día, así como la irritabilidad y la labilidad emocional, que incluyen llantos o risas inapropiadas. [30]

Los síntomas de fases anteriores anteriores se ven acrecentados provocando que el paciente sea incapaz de realizar tareas de cierta complejidad. Los problemas del lenguaje se hacen cada vez más evidentes, provocando parafasia. Las capacidades para leer y escribir también empeoran progresivamente. La memoria implíciticia y la memoria a largo que hasta entonces habían estado intactas también empiezan a verse afectadas. [29]

Demencia avanzada

Esta fase última de la enfermedad trae el deterioro de la masa muscular del paciente, perdiéndose la movilidad, la capacidad de autoalimentarse y en última instancia el encamamiento del paciente.

El lenguaje se vuelve totalmente desorganizado, incluso llegándose a perder completamente [29]. No obstante se conserva la capacidad de detectar y expresar señales emocionales.

La AD en sí no produce la muerte del paciente, si no que el fallo de otros sistemas que se ven afectados son los que la provocan. Los pacientes de Alzheimer pueden presentar dificultad para tragar y pueden inhalar los alimentos, lo cual puede originar neumonía por aspiración. La neumonía es la causa de la muerte en dos tercios de todas las muertes de pacientes de demencia, según la Sociedad de Alzheimer.

Síntomas

Los síntomas asociados en la AD varía en función de las características de cada individuo, por lo que es posible que se presente en diferente grado o incluso orden en función del paciente. Estos síntomas se agrupan en tres ámbitos.

Síntomas Cognitivos

Se ven afectadas la memoria a corto plazo en fases tempranas de la AD y seguidamente la memoria a largo plazo. La orientación espacial y temporal y la capacidad de ejecución también se ven afectadas. El síntoma principal es la incapacidad gradual de recordar a corto plazo debido a las lesiones que se producen en el hipocampo [31].

Síntomas Psicopatológicos

Se empiezan a presentar cambios conductuales como depresión, ansiedad, agresividad o trastorno del sueño. Estos cambios en el paciente están originados por los daños del lóbulo frontal.

Síntomas funcionales

La interrelación entre los síntomas cognitivos, psicológicos y conductuales provocan la incapacitación del paciente para realizar las tareas cotidianas habituales así como la limitación para emprender otras nuevas.

En esta lista se indican algunos síntomas cotidianos asociados al AD:

- Cambios de memoria que dificultan la vida cotidiana.
- Dificultad para planificarse y resolver problemas.
- Dificultad para resolver tareas en la casa, en el trabajo o en el tiempo libre.
- Desorientación de tiempo o lugar.
- Dificultad para comprender imágenes visuales y cómo los objetos se relacionan uno al otro en el ambiente.
- Colocación de objetos fuera de lugar.
- Disminución o falta de buen juicio.
- Perdida de iniciativa en el trabajo o en actividades sociales.
- Cambio en el humor o la personalidad.

Histopatología

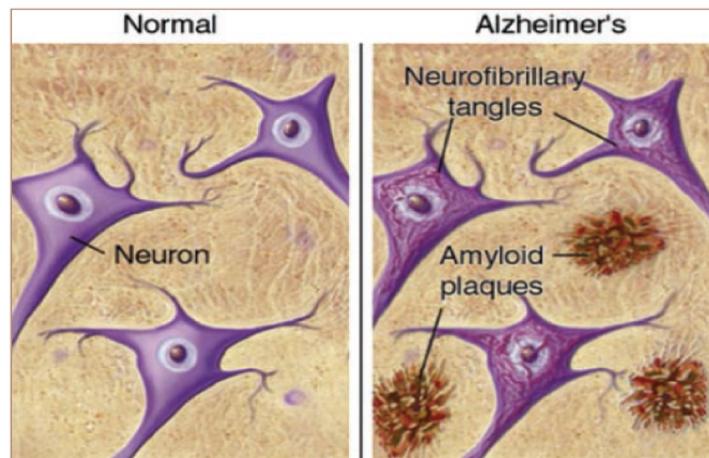


Figura 2.2: Ovillos neurofibrilares y placas seniles de los pacientes con AD respecto a los normales. Figura obtenida de [32]

Las lesiones neuropatológicas comienzan a desarrollarse años antes de la completa expresión de la demencia clínica. Actualmente se desconoce cual es el origen de este proceso de degeneración o porque los procesos normales asociados al envejecimiento se vuelven mucho más extremos en pacientes de esta enfermedad.

Desde una perspectiva patológica, los dos elementos característicos de la AD que son las placas neuríticas, la cual contiene la proteína beta amiloide ($A\beta$) y los ovillos neurofibrilares sirven como línea divisoria entre la AD y otras demencias, veáse la Fig. 2.4.

En la AD los ovillos neurofibrilares tienden a ser mas numerosos en las estruc-

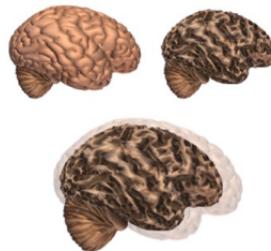


Figura 2.3: Ovillos neurofibrilares y placas seniles de los pacientes con AD respecto a los normales. Figura obtenida de [32]

turas del lóbulo temporal, incluyendo el hipocampo. Dentro del hipocampo los ovillos nuerofibrilares tienden a ocupar gran parte del espacio dejado por las neuronas piramidales muertas.

Debido al depósito y acumulación de placas seniles y ovillos neurofibrilares se genera estrés resultante de la inflamación y oxidación que se añaden a la cadena patológica de las consecuencias.

A medida que aumenta la enfermedad se ve reducido el número de células nerviosas y de conexiones entre ellas provocando un deterioro notable del cerebro como se aprecia en la imagen.

Base de datos ADNI

Las imágenes empleadas en este proyecto pertenecen a la iniciativa de neuro-imagen de la enfermedad de Alzheimer (ADNI, del inglés *Alzheimer Disease Neuroimaging Initiative*). Esta iniciativa fue fundada en 2004 [33] por un conjunto de instituciones de la salud norteaméricas en colaboración con diferentes compañías farmaceúticas. Una de las instituciones fundadoras más reconocidas es Instituto Nacional de la Salud (NIH, del inglés *National Institute of Health* creado en 1887, siendo actualmente referente en el ámbito de la salud en Estados Unidos.

Esta iniciativa reúne a las principales instituciones médicas tanto en Estados Unidos como en Canadá, siendo la organización que lidera las investigaciones dirigidas al entendimiento de los biomarcadores del cerebro asociados con el funcionamiento cognitivo del mismo. El investigador principal de esta iniciativa es Michael Weiner, profesor de la Universidad de California.

Hasta la fecha se han registrado hasta 1500 personas de entre 50 y 90 años en conjunto de los diferentes protocolos de la iniciativa. Se trata de una base de datos longitudinal de sujetos de la tercera edad, o cercanos a ella, que padecen AD o MCI.

Los principales objetivos de la iniciativa ADNI son los siguientes [33]:

- El desarrollo de métodos óptimos para la estandarización de la adquisición de biomarcadores, especialmente neuroimágenes, tanto MRI como PET , de una manera longitudinal sobre individuos que padecen AD o MCI.
- Uso de estos métodos optimizados de adquisición de imágenes longitudinales, tanto estructurales como metabólicas sobre un amplio conjunto de individuos sanos, de sujetos MCI o AD, acompañando dichas imágenes de una validación clínica del estado real de esos pacientes.
- Estudio de aquellos biomarcadores, medidas cognitivas o imágenes neurológicas que generan el mayor poder de diagnóstico sobre pacientes MCI y AD.
- Creación de un repositorio de datos, tanto imágenes como informes clínicos, con información longitudinal de cambios cerebrales, de metabolismo, de funcionamiento o de biomarcadores en los individuos estudiados.

Imágenes médicas

Las imágenes neurológicas constituyen una herramienta esencial para el estudio y el diagnóstico de los trastornos psiquiátricos de desarrollo neurológico. Estas imágenes permiten el estudio longitudinal de aquellos pacientes que sufren un deterioro cognitivo o funcional, además de permitir realizar una comparación con respecto a lo que se conoce como un desarrollo neurológico normal.

En el trabajo aquí realizado nos hemos centrado en las imágenes de resonancia magnética y las imágenes tomográficas por emisión de positrones.

Imágenes MRI

Las imágenes de resonancia magnética (MRI, del inglés *Magnetic Resonance Image*) son imágenes estructurales obtenidas en base a la aplicación de campos magnéticos sobre un cuerpo.

La técnica MRI esta basada en la resonancia magnética nuclear (NMR, del inglés *Nuclear Magnetic Resonance*). Ciertos núcleos atómicos son capaces de emitir energía a una determinada frecuencia al entrar en contacto con un campo magnético externo [36]. Generalmente, son átomos de hidrógeno los utilizados para la extracción de estas imágenes dado que este tipo de átomos existen de manera natural en las personas. Por esta razon, los escáneres evalúan la localización de la señal en el espacio, generando una imagen en función de la intensidad de la señal generada. Es posible variar el tipo de señal generada cambiando el tipo de campo magnético empleado.

Esta técnica fué inventada por Paul C. Lauterbur en Septiembre de 1971 [37], siendo aplicada por primera vez en el estudio del cerebro por Ian Robert Young y Hugh Clow en 1986 [38]. Desde entonces es considerada una técnica esencial tanto para el estudio del cuerpo humano como parea la investigación biomédica, convirtiéndose en una herramienta esencial de diagnóstico.

Una de las principales limitaciones de las imágenes MRI es el ruido. El bajo nivel SNR es provocado pr diversos factores, como es la alta emperatura de ruido generada por el escáner o el movimiento del cuerpo explorado. Es por ello que el procesado de filtrado de ruido es uno de los principales ámbitos de investigación en torno as las imágenes MRI [39]. La tecnología de los escáneres empleados ha evolucionado en aspectos como la resolución espacial o la disminución del tiempo de adquisición muestral.

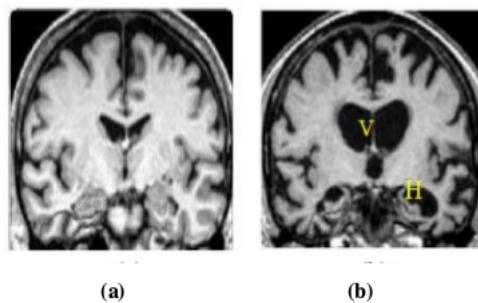


Figura 2.4: (a) Cerebro con funciones cognitivas normales. (b) Cerebro con funciones cognitivas d

Imágenes PET

La tomografía por emisión de positrones (PET, del inglés *Positron emission tomography*) es una técnica de extracción de imágenes funcionales que permite observar los procesos metabólicos del cuerpo.

Esta modalidad de imágenes neurológicas está basada en la detección de la radioactividad emitida por un pequeño vial inyectado en el paciente. Este vial está compuesto por radionúclidos, isótopos radioactivos emisores de positrones. Los más utilizados en las exploraciones PET son Carbono-11, Nitrógeno-13, Oxígeno-15, Fluor-18, Cobre-62, Galio-68, Rubidio-82.

Estos isótopos son elegidos principalmente por su corto periodo de vida [13]. Los radionúclidos son incorporados en algún compuesto para expandirlo por el organismo, en el caso del Alzheimer lo más común es la glucosa, a estos compuestos se los conoce como radiofármacos. En la actualidad el radiofármaco más utilizado es el fluorodesoxiglucosa (FDG) donde el flúor de la molécula se convierte en F18. Este radiofármaco es el más utilizado debido a sus características metabólicas ya que algunos de sus compuestos están presentes en el cuerpo humano y también por su rápida expulsión del organismo sin provocar ningún efecto secundario. FDG es incorporado principalmente en las células con elevadas tasas de glucosa, como por ejemplo el cerebro, donde la fosforilación de la misma impide que sea liberada al metabolismo.

En el caso de Alzheimer, debido a su alta tasa de glucosa en las células cerebrales, la imagen PET muestra una disminución de glucosa en sus fases iniciales lo que nos permite identificar rápidamente la enfermedad. También se podrá conocer la efectividad de los tratamientos, en cuyo caso se observará un aumento del metabolismo cerebral en relación con la situación inicial.

Estado del Arte

El presente proyecto tiene como principal objetivo la búsqueda de un modelo estadístico que nos permita generar neuroimágenes del cerebro, es por ello conveniente exponer algunos métodos clásicos de modelado generativo así como otros más novedosos en la sección que sigue a continuación.

No obstante, dado qué no existen estudios de la aplicación de la técnica usada con un fin generativo sobre neuroimágenes, se expondrá algunos de los trabajos relativos a la detección y el diagnóstico temprano de AD mediante computador con objeto de mostrar el alto índice de acierto en el diagnóstico de las técnicas actuales.

Modelos Generadores

En estadística, se define por modelo generador aquel sistema o modelo capaz de generar muestras (u observables) pertenecientes a una determinada clase o tipo respetando la función de distribución conjunta, esto es, una función de distribución multivariada asociada a dicho tipo.

Modelos clásicos

Un modelo generador viene definido por un conjunto de distribuciones de probabilidad las cuales son capaces de aproximar de manera adecuada un conjunto de datos.

Uno de los métodos tradicionalmente más empleados es el **Modelo de Mezcla de Gausianas**. Se trata de un modelo probabilístico que asume que todos los observables de un conjunto de datos son generados por un conjunto finito de gausianas. La obtención del modelo se basa en estimadores de máxima verosimilitud.

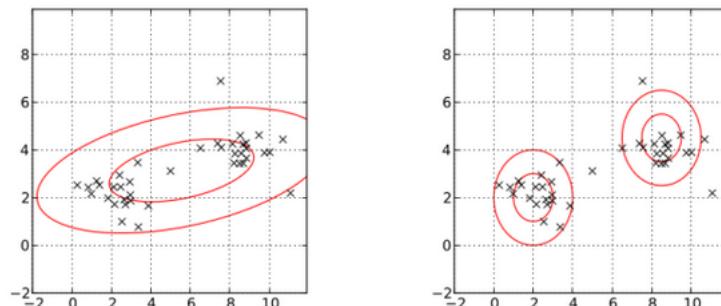


Figura 2.5: (Izquierda) Modelo de una única Gausiana, (Derecha) Modelo de mezcla de Gausianas

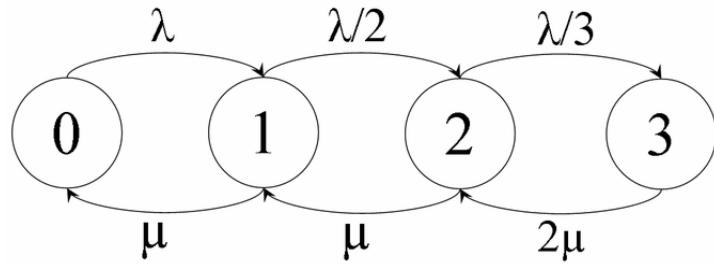


Figura 2.6: (Izquierda) Modelo de una única Gausiana, (Derecha) Modelo de mezcla de Gausianas

EL GMM es una de las técnicas más usadas para el modelado de datos del mundo de real. Intuitivamente podemos pensar en este método como la mezcla de varias gausianas mono-modales.

Un **Modelo Oculto de Márkov** (HMM, del inglés *Hidden Markov Model*) es un modelo generador que asume que el proceso o sistema a modelar es un proceso de Markov. El objetivo es determinar los parámetros desconocidos de dicha cadena a partir de los parámetros.

Un HMM genera de manera explícita la distribución de probabilidad de los estados del proceso evaluado debido a la probabilidad condicional de transición entre estados, es por ello que es considerado un modelo generativo. Este tipo de modelos han sido ampliamente usado ámbitos como el reconocimiento del habla o en teoría de colas.

La Máquina de Boltzmann

Una Máquina de Boltzman (BM del inglés *Boltzmann Machine*) es un tipo de red neuronal estocástica. Esta técnica puede emplearse para la obtención o aprendizaje de la distribución de probabilidad del conjunto de muestras en cuestión.

Dado que este proceso es costoso, se suele imponer un conjunto de restricciones en la topología de la red neuronal lo cual se conoce como máquina restrictiva de Boltzman (RBM, del inglés *Restrictive Boltzmann Machine*). [40]

Una RBM es un modelo generativo parametrizado que representa una probabilidad de distribución. Dado un conjunto de observaciones, la elaboración de una RBM implica el ajuste de los parámetros con el objetivo de que la distribución de probabilidad asociada a la BM sea lo más similar posible a la distribución real de los datos.

Tras un proceso de aprendizaje exitoso, uno RBM es capaz de extraer la distribución de probabilidad latente u oculta en un conjutno de datos. Esto puede ser empleado como referencia a la comparar con nuevas muestras, lo que se conoce como proceso de clustering clasificatorio, o puede permitirnos extraer muestras del

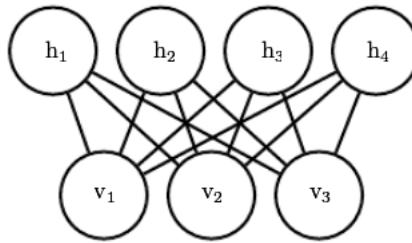


Figura 2.7: M odelo b asido de RBM

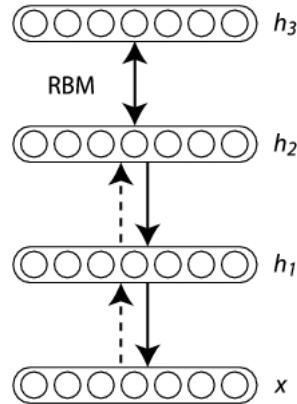


Figura 2.8: *Stacked RBMs*

conjunto muestreando a partir de la distribuci n aprendida.

En la figura 2.7 se puede observar un modelo cl sico de RBM. Las caracte sticas principales de este modelo son:

- Esta constituido por dos partes bien diferenciadas; una capa de unidades ocultas y otra de unidades visibles. A menudo la unidades visibles son referenciadas como estados visibles.
- No hay conexiones entre unidades de una misma capa.
- Las unidades ocultas estan probabilisticamente condicionadas a las unidades visibles, aunque son independientes entre s .
- El uso de RBM concatenadas permite la extracci n de caracte sticas mas complejas, ver figura 2.8. En este modelo, el cual es com unmente denominado *stacked RBMs*, las unidades ocultas(h) se convierten en los datos de entradas de las siguientes capas.

Las RBM son consideradas redes neuronales aplicables para el aprendizaje no supervisado, capaces de caracterizar un espacio muestral. Otra modelo similar es el Autoencoder, técnica empleada en este trabajo.

Autoencoder

Se conoce como Autoencoder al tipo de red neuronal que tiene como objetivo la caracterización de un conjunto de datos con objeto de imitar el dato de entrada a la salida de la red. Esta proceso de imitación está basado en la extracción de características, las cuales son obtenidas en lo que se denomina como capa latente. Esto permite que los autoencoders sean un método de reducción de dimensionalidad altamente empleado.

En el siguiente capítulo se profundizará en las expresiones matemáticas por lo que durante este únicamente se dará una visión general del modelo.

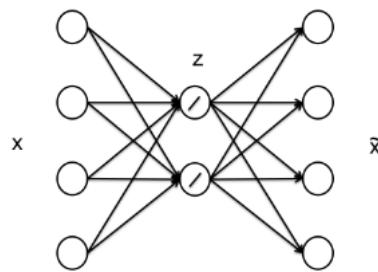


Figura 2.9: Autoencoder Lineal

En la figura 2.10 se puede observar un modelo de Autoencoder básico denominado Autoencoder Lineal. Las características principales son:

- Codificación: $\mathbf{Z} = \mathbf{F}(\mathbf{X})$. $\mathbf{F}(\mathbf{X})$ es la función de codificación basada en los productos de los datos de entrada y unos pesos, obtenidos durante el entrenamiento, aplicándose una función de activación, típicamente la función sigmoidal.
- Decodificación. $\hat{\mathbf{X}} = \mathbf{G}(\mathbf{Z})$. $\mathbf{G}(\mathbf{Z})$ realiza el proceso inverso, tomando como entrada el vector de valores del espacio latente \mathbf{Z} .

Si el espacio latente \mathbf{Z} tiene una dimensionalidad menor que el espacio \mathbf{X} , entonces la función \mathbf{F} tiene capacidad de compresión sobre las muestras de entrada x . Es por ello que un vector latente z , obtenido a partir de un vector de entrada x , puede considerarse una representación de dimensionalidad reducida de x .

Es interesante mencionar que en este caso tanto $\mathbf{F}(\mathbf{X})$ como $(\mathbf{G}(\mathbf{Z}))$ son funciones deterministas, a diferencia de las empleadas en una RBM, donde son funciones probabilísticas.

Las principales aplicaciones del Autoencoder son la extracción de características de las muestras a partir de la capa latente \mathbf{Z} y la posibilidad de generación de nuevas muestras con unas características similares a las utilizadas durante el proceso de caracterización del modelo. Esta capacidad de generación se basa en modificar los códigos latentes z de las muestras x .

No obstante, una de las principales limitaciones asociadas a este modelo lineal de autoencoder es la posibilidad de que la función codificadora y decodificadora únicamente aprendan una función identidad de la muestra de entrada, lo cual imposibilita tanto el proceso de reducción de dimensionalidad sobre muestras no observadas durante el proceso de entrenamiento como el proceso de generación de nuevas muestras.

El Autoencoder de filtrado (denominado *denoising Autoencoder* en inglés) tiene como objetivo evitar dicho problema [41]. Para ello a todas las muestras de entrada x se les aplica un ruido no determinista con objeto de forzar que la capa latente tenga que aprender características robustas que extraigan pasiciones identidad de las muestras.

Un Autoencoder de filtrado realiza tres acciones principales:

- Mezclado de ruido. La muestra de entrada x es mezclada con un ruido aleatorio.
- Codificación. El código latente generado z debe preservar la información primaria de la muestra previa al mezclado del ruido.
- Decodificación. La muestra de salida \hat{x} ha de ser lo más parecida posible a la muestra de entrada x .

En el trabajo realizado por Pascal Vincent y Hugo Laroché en 2008 [42], el ruido es introducido en las muestras mediante un proceso estocástico asignando cero a algunos de los valores de las muestras. En este caso el Autoencoder de filtrado está intentando caracterizar o "predecir" los valores eliminados a partir de los valores presentes de las muestras.

Gracias al reciente de las redes profundas, se han propuesto modelos de autoencoders cada vez más complejos con varias capas intermedias, lo cual permite extraer características más complejas. El **Autoencoder Variacional** (*VAE* del inglés *Variational Autoencoder*) se ha convertido en uno de los métodos mas populares para el aprendizaje no supervisado de distribuciones complicadas.

El VAE, a diferencia de los modelos de autoencoders clásicos, busca la caracterización del espacio muestral de entrada no mediante una función determinista sino con una función de probabilidad, realizando una serie de restricciones sobre la función de distribución de los valores del espacio latente. Los conceptos matemáticos en profundidad relativos a este algoritmo serán explicado en el próximo capítulo debido a que es la principal herramienta en este trabajo.

Red Generativa Adversaria

Otro modelo novedoso, impulsado por el desarrollo de las redes neuronales profundas, es la Red Generativa Adversaria[43]. Este modelo tiene una alta capacidad de caracterización, siendo capaz de generar imágenes realmente realistas. [44].

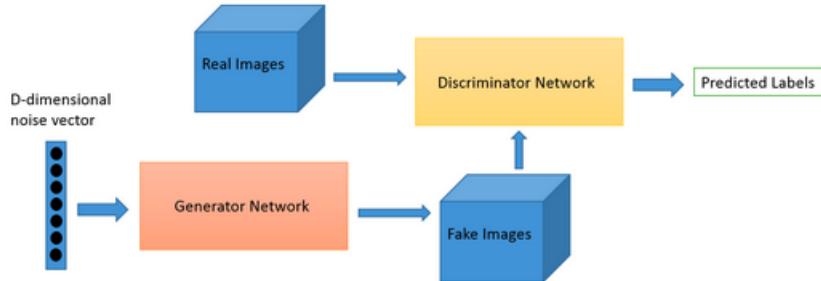


Figura 2.10: Esquema de una Red Generativa Adversaria

En este tipo de modelos tenemos dos redes bien diferenciadas la red generadora y la red discriminadora. Normalmente, la red generadora es la encargada de generar las muestras a partir del espacio latente, mientras que la red discriminadora ha de comparar las muestras generadas con las originales del espacio muestral con objeto de determinar si verdaderamente se pueden considerar muestras artificiales de dicho espacio.

El proceso de entrenamiento de la red generativa tiene como objetivo generar muestras que sean lo más parecidas a las originales, y por lo tanto, que no puedan ser detectadas por el discriminador. Mientras que el objetivo del entrenamiento de la red discriminativa es justo el contrario, esto es, ser lo más estricta posible.

Es por ello que el entrenamiento de este sistema es realmente complejo, considerándose más complicado de conseguir unos valores de entrenamiento óptimos que con respecto al VAE.

Trabajos Previos

En la búsqueda de estudios similares al aquí desarrollado nos encontramos con el trabajo desarrollado por Eunbyung Park de la universidad de Carolina del Norte, Estados Unidos. Este trabajo parte con el objetivo de la extracción de características útiles de imágenes MRI tanto para el diagnóstico del AD como para su uso generativo [45].

En dicho trabajo se usa un autoencoder variacional convolucionial 2d, lo cual difiere de los empleados en nuestro trabajo que, como se explicará más adelante, son convolucionales 3D o totalmente densos.

Se expone como dicho modelo es capaz de caracterizar componentes estructurales de las imágenes, permitiendo diferenciar la información generada en un espacio 2d, aplicando reducción de dimensionalidad mediante T-SNE. No obstante, se menciona como las imágenes regeneradas son algo borrosa, algo característico del VAE.

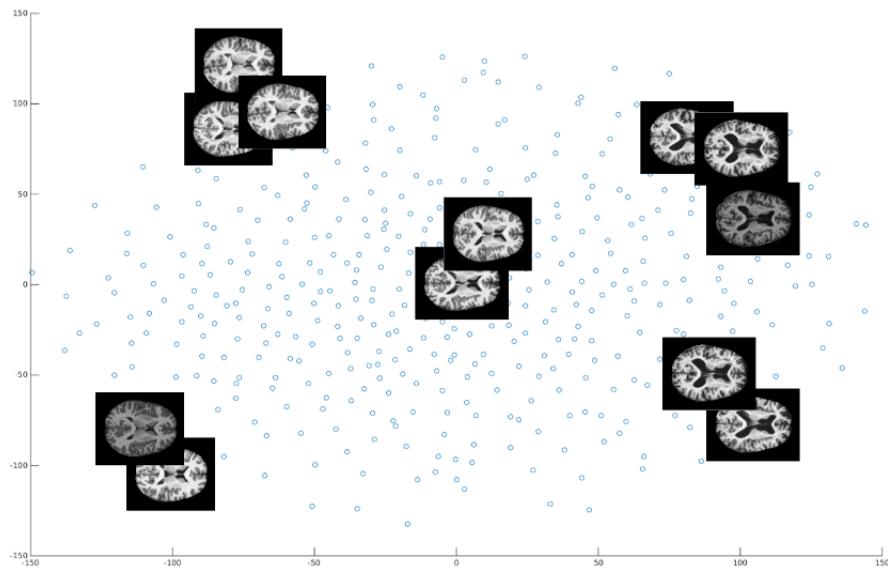


Figura 2.11: Visualización del espacio de dimensionalidad reducida conseguido en el trabajo [45]. Las imágenes superpuestas corresponden a las imágenes reales, no a las generadas

En la figura 2.11 se aprecia como es posible la diferenciación por características principales de las imágenes MRI en un espacio de dos dimensiones.

Diagnóstico Asistido por Computador de AD

Aunque el objetivo final del trabajo es generar imágenes útiles para el diagnóstico y estudio del AD debido a los escasos trabajos que apliquen el VAE a este fin, se cree conveniente la exposición de las siguientes técnicas de diagnóstico, ya que es necesario para comprender la utilidad y el estado actual de las diferentes herramientas del ámbito tratado.

Durante la última década se han desarrollado todo tipo de aproximaciones para el diagnóstico de AD asistido por computador. Este gran desarrollo es debido, en parte, a asociaciones de gran calado como ADNI.

En este tipo de técnicas CAD, normalmente nos encontramos con dos fases bien diferenciadas, una es la extracción de características mientras que la siguiente es la clasificación realizada sobre dichas características.

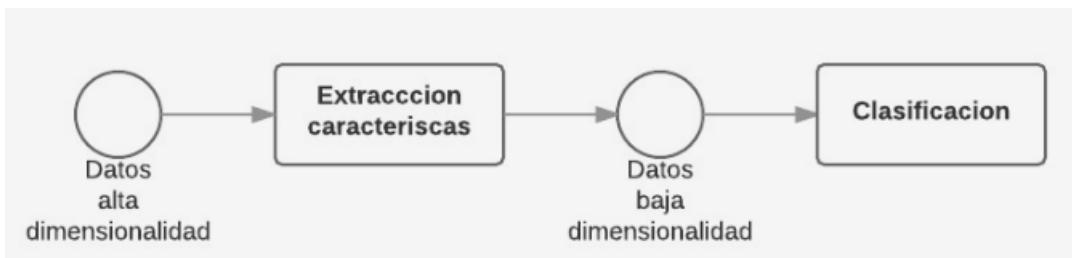


Figura 2.12: Esquema clásico básico de un sistema de diagnóstico de AD asistido por computador

Métodos Clásicos

Algunas técnicas clásicas de extracción de características son las siguientes [46]:

- Análisis de componentes independientes (ICA, del inglés *Independent Component Analysis*). Esta técnica de transformación de parámetros permite capturar información de un orden alto de dimensionalidad y transformarlo a un orden menor, utilizando componentes vectoriales estadísticamente diferentes. [47]
- Análisis de Componentes Principales (PCA, del inglés *Principal Component Analysis*). Esta técnica es usada cuando el objetivo es reducir el número de características y convertirlas a un espacio de alta varianza y menor dimensionalidad [48].
- *Wavelets*. Se trata de un conjunto de funciones matemáticas encargadas de descomponer los datos en función de la frecuencia. La Transformada de Fourier

solo genera información frecuencial relativa al contenido, es por ello, que la Transformada *Wavelet* es una mejor herramienta para el estudio de las imágenes. [49]

- Matriz de co-ocurrencias de niveles de gris (**GLCM** del inglés *Gray Level Co-Occurrence Matrix*). Esta técnica está basada en la extracción de características estadísticas de la imagen. La matriz caracteriza la distribución de los niveles de gris de una imagen o de una región.

Algunos de los métodos que se expondrán a continuación son utilizados para la clasificación de los vectores de datos generados por los métodos de extracción de características.

- Clasificador KNN (del inglés *K-Nearest neighbor*). Este método de clasificación es uno de los más usados históricamente [50]. Este clasificador se basa en la evaluación de la distancia de una muestra en cuestión x_i con respecto al conjunto de clases o instancias posibles M , previamente predefinidas durante el entrenamiento. Se considerará que la muestra x_i pertenecerá a la clase m con respecto a la cual la distancia sea la mínima.
- Clasificador de Bayes (*Naïve bayes Classifier*). Se trata de un clasificador basado en el teorema de Bayes. En este clasificador se considera que todas las características contribuyen de manera independiente a la probabilidad de pertenecer a una clase u otra, sin tener en cuenta la presencia del resto de variables.
- Maquina de Vectores de Soporte (SVM del inglés *Support Vector Machine*). Este método de clasificación es uno de los mejores algoritmos de aprendizaje supervisado, siendo diseñado originariamente para la clasificación binaria.

Este será el método usado en este trabajo para la evaluación de las características extraídas por el sistema usado, en nuestro caso es un Autoencoder Variacional, siendo las características los valores de la capa latente. Algunos detalles de este algoritmo serán expuestos en siguientes capítulos.

Métodos de Aprendizaje Profundo

Aunque las técnicas anteriormente expuestas constituyen los métodos clásicos relativos a las herramientas CAD, son los métodos basados en aprendizaje profundo (*deep learning*) los que han conseguido los mejores resultados de clasificación [51].

En estas técnicas de aprendizaje se han conseguido más de un 95 % de precisión en el diagnóstico del Alzheimer [52][53].

Estas técnicas hacen uso de redes neuronales capaces de extraer características complejas de las imágenes. En algunos casos, se emplean un primer tipo de red,

como por ejemplo de un Autoencoder, como método de extracción de características y seguidamente un método de clasificación ya sea clásico o basado también en aprendizaje profundo.

Generalmente se usa un tipo de red neuronal denominadas redes neuronales convolucionales (CNN del inglés *Convolutional Neural Networks*), ideales para la extracción de información de las imágenes. Este tipo de redes están basadas en el comportamiento de la vista humana, ya que son capaces de extraer información espacial de los datos, lo cual es ideal para las imágenes.

En el trabajo realizado por Ehsan Hossini-Asl y Robert Keynton se consiguen resultados de exactitud de 97.6 % en la diferenciación de pacientes AD de los NC, mientras que consigue hasta un 90 % en la diferenciación de los MCI de los AD [54]. Este trabajo emplea un modelo convolucional 3D, capaz de aprender características genericas empleando para ello un Autoencoder, el cual es entrenado para capturar las variaciones estructurales en las imágenes MRI.

El planteamiento realizado en este trabajo es diferente al nuestro dado que se emplea un tipo de autoencoder en el que se capturan variables locales debido al uso de conexión mediante nodos locales en lugar de conexión global que es nuestro caso. A priori, el empleo de conexiones no globales puede dificultar la extracción de características, pero permite reducir de forma notable el tiempo necesario para el entrenamiento del sistema. Por otro lado, en nuestro trabajo se evalúan imágenes seccionadas por regiones cerebrales, en lugar de emplear imágenes completas.

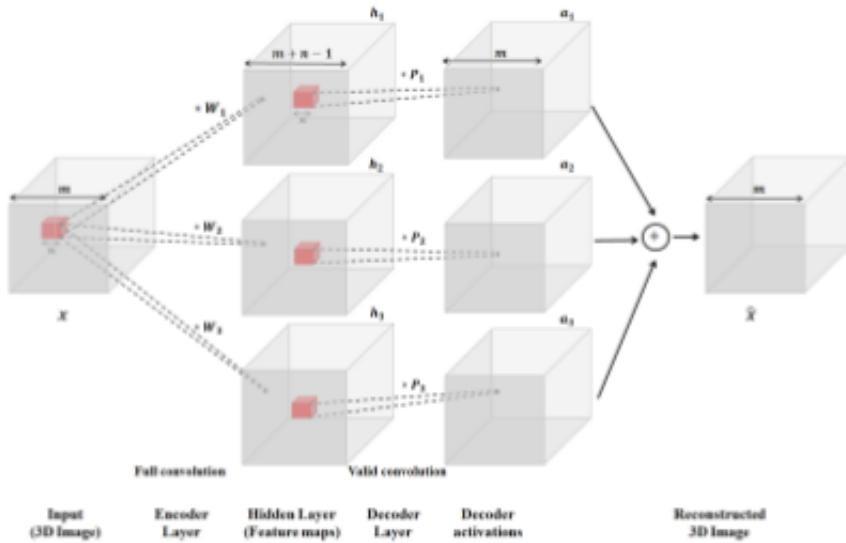


Figura 2.13: Diagrama del Autoencoder Convolutacional 3D empleado en el trabajo [54] para la extracción de características

Como método de clasificación se empleó una red neuronal densa, esto es, totalmente conectada entre las unidades de capas adyacentes, la cual recibía los datos generados por el Autoencoder.

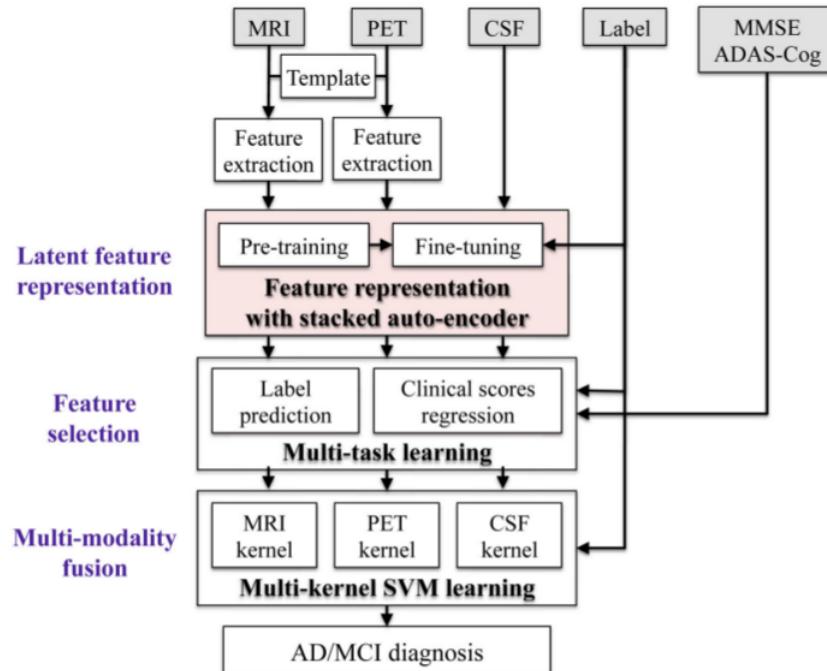


Figura 2.14: Diagrama general del método desarrollado en el trabajo [55]

Uno de los primeros estudios enfocados a la aplicación de aprendizaje profundo para el diagnóstico de AD fue el desarrollado por Heung-Il y Dinggang Shen en 2013 [55]. Es interesante notar como un Autoencoder es usado como método de extracción de características. El modelo propuesto en este estudio es el de la imagen 2.14.

La principal innovación de este método es la capacidad de extracción de correlaciones no lineales entre los datos gracias al uso de un autoencoder, lo cual teóricamente permite la extracción de mejores características globales.

Se trata de un modelo de diagnóstico multimodal ya que emplea diferentes tipos de neuroimágenes como son MRI, PET y CSF. Se emplea un Autoencoder en cascada, que permite la extracción de características, pero a diferencia del modelo Variacional, usado en este trabajo, este modelo de Autoencoder emplea una función determinista en lugar de probabilística. Como herramienta de clasificación se emplea un SVM multikernel, aplicado cada kernel a las diferentes fuentes de datos.

Los resultados indicados son de un 95 % de exactitud en la diferenciación AD sobre NC.

Entorno de desarrollo

Los diferentes elemtos software generados en este proyecto han sido desarrollados sobre Python. Este lenguaje de programación cuenta con una comunidad científica en auge. Los principales motivos por lo que se ha seleccionado son:

- Se trata de un lenguaje de código libre, lo cual evita cualquier tipo de coste asociado a la licencia de lenguaje.
- Cuenta con reconocidas librerías de métodos numéricos y estadísticos que agilizan el desarrollo de los modelos. Algunas de estas librerías son *SciPy*, *Numpy* o *Sklearn*. *Numpy* es una librería que trabaja con vectores, lo cual resulta ideal para cieníficos provenientes del entorno Matlab.
- Debido a su amplia comunidad hay una gran cantidad de código en repositorios públicos los cuales son útiles como referencia.

Para el desarrollo de los algoritmos basados en aprendizaje profundo hemos hecho uso de *Tensorflow* [56]. Esta biblioteca desarrollado por *Google* constituye una interfaz para el desarrollo de algoritmos de aprendizaje que facilita la implementación de dichos algoritmos.

El código de esta librería fué hecho público por *Google* el 9 de Noviembre de 2015. *Tensorflow* fué originalmente desarrollado por el equipo *Google Brain* desde 2011, denominándose *DisBelief*.

Basado en la unión de grafos de las diferentes unidades del algoritmo implementado, esta librería esta principalmente orientada al desarrollo de redes neuronales, proveyendo métodos para generar redes neuronales densas y convolucionales.

Otro punto que fomenta el uso *Tensorflow* es la posibilidad de ejecutar los algoritmos sobre tarjetas gráficas (GPU, del inglés, *Graphical Processing Unit*) en lugar de sobre el procesador central (CPU, del inglés *Central Processing Unite*).

Capítulo 3

Fundamentos Teóricos

El trabajo realizado en este proyecto es englobado dentro de la temática denominada visión por computador, dado que los métodos empleados se basan en la detección de patrones sobre las imágenes dadas, en nuestro caso neuroimágenes.

En el ámbito de la visión por computador cada imagen en sí misma es una muestra de miles dimensiones, cada uno de los pixeles. Un modelo generativo trata de capturar la relación entre las múltiples dimensiones de los datos. En nuestro caso el modelo empleado para capturar dichas relaciones es el Autoencoder Variacional.

Es por ello que este capítulo se centrará en la exposición de este método en primer lugar. Dado el VAE esta fundamentado en el aprendizaje profundo, se dedicará la siguiente sección a las redes neuronales, haciendo especial hincapie a aquellas empleadas en este trabajo. Finalmente se expondrán brevemente los métodos estadísticos usados de manera auxiliar a lo largo de este proyecto.

Autoencoder Variacional

Este apartado está dedicado a la exposición del Autoencoder Variacional desde una perspectiva meramente teórica con objeto de mostrar los fundamentos y, en última instancia, la capacidad de convergencia del método, basada en una función objetivo sobre la cual se puede aplicar descenso en gradiente estocástico.

Modelo de Variables Latentes

A lo largo del entrenamiento o la caracterización de un modelo generativo, la parte más complicada es la extracción de las dependencias entre las múltiples dimensiones. Son estas relaciones multidimensionales las que permiten generar muestras artificiales pertenecientes a clases distintas. Se denomina variable latente, a las

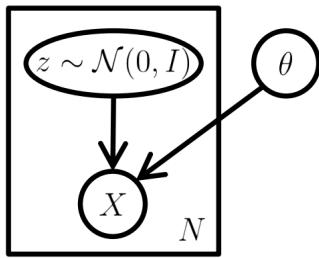


Figura 3.1: Modelo gráfico de variables latentes para el modelo generativo del VAE. Z es el espacio de variables lo más similar posible a un distribución normal ($N(0, I)$). El elemento θ es el conjunto de parámetros que aplicados de manera funcional sobre las variables latentes son capaces de generar el conjunto muestral X

unidades del modelo generativo capaces de discernir entre las distintas clases, esto es, capacitan al modelo para generar elementos diferenciados.

Un modelo generativo es representativo de un espacio muestral (X) si para cada una de las muestras de dicho espacio (x) hay al menos alguna configuración de las variables latentes (z) que genera un variable (\hat{x}) muy similar a la original. Formalmente, dada una función $f(z, \theta)$ parametrizada por un vector θ en un espacio Θ tal que:

$$f : Z \times \Theta \rightarrow X \quad (3.1)$$

Modelo Probabilístico

El objetivo es maximizar la probabilidad de cada x de el espacio muestral de acuerdo con:

$$P(X) = \int P(X|z; \theta) \quad (3.2)$$

En la ecuación 3.2, $f(z; \theta)$ es reemplazada por la distribución $P(X|z; \theta)$, la cual nos permite hacer explícita la dependencia de X sobre z , debido a la probabilidad condicionada. La idea de detrás de dicha expresión es principio de máxima verosimilitud (ML, del inglés *Maximum Likelihood*), el cual indica que si el modelo es capaz de generar muestras del espacio X , entonces será probable que le modelo generativo construya muestras similares.

En el VAE, la función de probabilidad $P(X|z; \theta)$ es las siguiente:

$$P(X|z; \theta) = N(X|f(z; \theta), \sigma^2 * I) \quad (3.3)$$

El uso de una distribución gausiana nos permite emplear descenso en gradiente durante la optimización, con objeto de caracterizar el modelo. Esta caracterización permite incrementar $P(X)$, entendida como la probabilidad global de generar algún tipo de muestra de dicho espacio. Esto no sería posible si esta función de probabilidad fuera una delta de Dirac. Es importante notar que es fundamental disponer de una función $P(X|z)$ que sea computable y continua sobre θ .

Teóricamente, para la mayoría de los valores z , $P(X|z)$ será aproximadamente cero, y por lo tanto su contribución para la estimación de $P(X)$ será prácticamente nula.

Función Objetivo

La principal idea en la que se fundamenta el VAE es en muestrear los valores de z a partir de X , esto es, necesitamos una nueva función $Q(z|X)$ que nos permita generar el conjunto de valores del espacio Z a partir de X . Esto nos reduce el espacio de Z ya que, teóricamente, este se verá limitado en $Q(z|X)$. En última instancia, esto nos permitirá estimar $E[P(X|z)]$, siendo ésta el valor esperado de la distribución de probabilidad de los valores de X generados.

La relación entre $E(P(X|z))$ y $P(X)$ es uno de los fundamentos de los métodos variacionales Bayesianos. Comencemos con la definición de la divergencia de Kullback-Leibler (KL o D) entre una distribución $P(z|X)$ y $Q(z)$:

$$D[Q(z)||P(z|X)] = E[\log(Q(x)) - \log(P(z|X))] \quad (3.4)$$

La expresión anterior, ecuación 3.4, es una medida no simétrica de la similitud o diferencia entre las dos funciones de probabilidad $P(X|z)$ y $Q(z)$. Dicha expresión mide diferencia (o el extra de información) entre un código $P(x)$ y uno $Q(z)$. Aplicando la regla de Bayes sobre la expresión anterior conseguimos dejarlo en función de $P(X)$ y $P(X|z)$:

$$D[Q(z)||P(z|X)] = E_z[\log(Q(x)) - \log(P(X|z)) - \log(P(z)) + \log(p(X))] \quad (3.5)$$

Ordenando la expresión anterior, y teniendo en cuenta que $\log(p(X))$ no depende de z por lo que puede salir del valor esperado:

$$\log(p(X)) - D[Q(z)||P(z|X)] = E_z[\log(P(X|z))] - D[Q(z)||P(z)]. \quad (3.6)$$

Llegados a este punto es importante notar que el espacio X es fijo y por lo tanto también lo es su función de probabilidad $P(X)$. No obstante $Q(z)$ puede ser cualquier distribución, siempre que nos permita generar Z a partir de X .

Dado que en nuestro caso estamos interesados en inferir $P(X)$, es necesario generar una función Q dependiente sobre X que permita que la divergencia $D[Q(z)||P(z|X)]$ sea pequeña, esto es, haya la menor perdida de información entre ambas distribuciones.

$$\log(p(X)) - D[Q(z|X)||P(z|X)] = E_z[\log(P(X|z))] - D[Q(z|X)||P(z)]. \quad (3.7)$$

La expresión anterior, ecuación 3.7, es la principal del VAE, por lo que es necesario examinarla detenidamente. Analizando cada término por separado:

- La expresión de la izquierda representa la cantidad que se pretende maximizar: $\log(P(x))$, mas un término de error representado por $D[Q(z)||P(z|X)]$ que es la capacidad de generar z a partir de X . Este término de error será disminuido si Q es de alta capacidad.

Se trata de maximizar $\log(P(X))$ mientras simultáneamente $D[Q(z|X)||P(z)]$ se minimiza. El término de probabilidad $P(z|X)$ no es computable analíticamente, describe la distribución de valores de z que son capaces de generar X .

- La expresión de la derecha es lo que se pretende optimizar mediante el descenso en gradiente, dada una correcta selección de $Q(z)$.

Este segundo término fuerza la similitud entre $(Q(z|X))$ y $P(X|z)$. Asumiendo que el término $Q(z|X)$ es de alta capacidad, tendremos que el término de divergencia KL será cercano a cero. En última instancia, conseguiremos manejar de forma analítica $P(z|X)$ gracias a su similitud con $Q(z|X)$.

Optimización de la función objetivo

Con objeto de poder realizar el descenso en gradiente sobre la expresión de la derecha de la ecuación 3.7, necesitamos definir de manera más exacta la forma de $Q(z|X)$. La elección habitual es la siguiente:

$$Q(z|X) = N(z|\mu(X; \vartheta), \Sigma(X; \vartheta)) \quad (3.8)$$

donde μ y Σ son funciones deterministas con una serie de parámetros ϑ (en las siguientes expresiones se omitirá ϑ). Normalmente tanto μ como Σ son implementados mediante redes neuronales y Σ está limitada a un función diagonal, que permite facilitar los cálculos.

El segundo término de la expresión 3.7, $D[Q(z|X)||P(z)]$, al ser una divergencia KL entre dos funciones de gausianas multivaradas queda definida por:

$$D(N(\mu_0(X), \Sigma_0(X))||N(\mu_1(X), \Sigma_1(X))) = \frac{1}{2} \left(\text{tr}(\Sigma_1^{-1} \Sigma_0) + (\mu_1 - \mu_0)^T \Sigma_1^{-1} (\mu_1 - \mu_0) - k + \log\left(\frac{\det\Sigma_1}{\det\Sigma_0}\right) \right)$$

donde k es la dimensionalidad de la distribución, la expresión queda de la siguiente manera:

$$D[N(\mu(X), \Sigma(X))||N(0, I)] = \frac{1}{2} \left(\text{tr}(\Sigma(X)) + (\mu(X))^T (\mu(X) - k - \log(\det(\Sigma(X)))) \right) \quad (3.9)$$

El primer término de la expresión 3.7, $E_z[\log(P(X|z))]$, es algo más complicado de determinar, aunque a priori se podría estimar usando un número suficientes de z y aplicando al función f asociada a $P(X|z)$, aunque esto sería tremadamente costoso computacionalmente.

En su lugar, se aplica un procedimiento denominado Descenso en Gradiente Estocástico (SGD, del inglés *Stochastic Gradient Descent*), que se basa en tomar únicamente un valor de z aplicarlo sobre $P(X|z)$, por lo que se obtendría una aproximación de $E_z[\log(P(X|z))]$. Durante este proceso, estamos tomando como referencia cada una de las muestras X de un conjunto de datos D a la hora de estimar el error. Teniendo en cuenta esto, la ecuación completa que se pretende optimizar es:

$$\begin{aligned} E_X[\log(P(X)) - D[Q(z|X)||P(z|X)]] &= \\ E_X[E_z[\log(P(X|z))] - D[Q(z|X)||P(z)]] \end{aligned} \quad (3.10)$$

Tomando el gradiente de la expresión anterior, reducimos la expresión a los valores internos de las esperanzas. Además, podemos tomar un único valor de X y un único valor de z de la distribución $Q(z|X)$, lo que no nos permite hacer computable el gradiente de la siguiente forma:

$$\log(P(X|z)) - D[Q(z|X)||P(z)]. \quad (3.11)$$

No obstante hay un problema significativo en la ecuación 3.1.4 ya que $E_z[\log(P(X|z))]$ depende de los parámetros de P y también de los valores de Q . Esto es problemático a la hora de realizar el descenso en gradiente, quedando resuelto con lo que se conoce como "Truco de Reparametrización".

El truco de Reparametrización

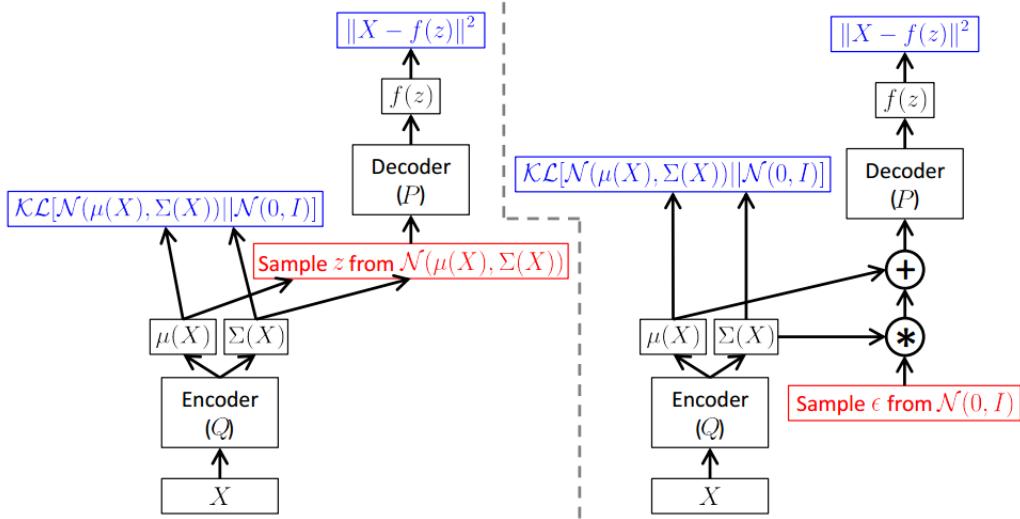


Figura 3.2: (Izquierdo) Modelo de VAE sin Truco de Reparametrización. (Derecha) Modelo de VAE con Truco de Reparametrización

Para garantizar el correcto funcionamiento del VAE es necesario que la función codificadora (f) asociada a $Q(z|X)$ genere un conjunto Z capaz de ser decodificado por la función generadora (g) asociada a $P(X|z)$.

Analizando el problema desde otra perspectiva, tomando como referencia el diagrama de izquierda de la figura 3.2. El paso hacia delante¹ funciona de manera de correcta y es de esperar (si los parámetros están correctamente entrenados) que la salida produzca un resultado acertado de manera general.

No obstante, es necesario realizar el paso hacia atrás² teniendo que determinar el gradiente sobre la función $Q(z|X)$ encargada de generar z , pero este modelo de

¹En el ámbito de las redes neuronales se denomina paso hacia delante (del inglés *forward pass*) al proceso inicial de evaluar la salida generada a partir de una determinada entrada. En nuestro caso la entrada es X y la salida $f(z)$, siendo la evaluación realizada $\|X - f(z)\|$

²En el ámbito de las redes neuronales, el paso hacia atrás (del inglés *backpropagation*) hace referencia al proceso de evaluar el gradiente en cada uno de los elementos del sistema, tomando como referencia que el error se?a el determinado del paso hacia delante

generación esta basado en el mapeo sobre una distribución gausiana, lo cual es una función no continua.

La solución a este problema se denomina truco de reparametrización (del inglés *reparameterization trick*) el cual se basa en trasladar el mapeo sobre la distribución gausiana a una capa de entrada.

Dados μ_X y Σ_X , media y covarianza respectivamente de $Q(z|X)$, podemos mapear $N(\mu_X, \sigma(X))$ tomando un valor de la función Normal ($\epsilon \sim N(0, I)$) y aplicando la siguiente expresión:

$$z = \mu(X) + \Sigma(X)^{1/2} * \epsilon. \quad (3.12)$$

Por lo tanto la función final, la cual queda representada en el diagrama de la derecha de la figura 3.2, sobre la que se aplica el gradiente es la siguiente:

$$E_{X \sim Z} [E_{\epsilon \sim N(0, I)} [\log(P(X|z = \mu(X) + \Sigma^{1/2} * \epsilon))] - D[Q(z|X)||P(z)]] . \quad (3.13)$$

Cabe notar que ninguna de las esperanzas son con respecto a las distribuciones características del sistema (ni $P(X|z)$ ni $Q(z|X)$) lo que nos permite realizar el gradiente sin ningún problema sobre los elementos contenidos dentro de los valores esperados, ya que el gradiente es la derivada sobre los parámetros funcionales de estas distribuciones.

Por lo tanto dado un valor de X y ϵ la función 3.1.5 será continua y determinista sobre los parámetros de P y Q , lo cual nos permite realizar el paso hacia atrás de manera eficaz.

Interpretación de la función objetivo

La función de pérdidas de un VAE se identifica con el logaritmo negativo de la probabilidad de verosimilitud con un regularizador. Se puede descomponer la función de pérdidas en términos de la pérdida asociada a cada una de las muestras l_i . Por lo tanto dicha función tendrá la siguiente forma $\sum_{i=1}^N l_i$ para las N muestras totales. La función de pérdidas l_i para una muestra x_i será:

$$l_i(\theta, \phi) = -E_z[\log_{p_\phi}(x_i|z)] + KL(q_\theta(z|x_i)||p(z)) \quad (3.14)$$

El primer término es la pérdida de reconstrucción, más precisamente se trata del logaritmo negativo del valor de verosimilitud de la muestra i . Este valor esperado está tomado con respecto a la distribución generada por el codificador (es decir el espacio latente generado) para la muestra i dada. Este término implica que el decodificador tratará de reconstruir de la manera más precisa posible la muestra en cuestión. Si la muestra generada por el codificador es muy distinta de la original conllevará una alta penalización en la pérdidas.

El segundo término es el regularizador. Se trata de la divergencia de Kullback-Leibler entre la distribución del codificador $q_\theta(z|x)$ y $p(z)$. Esta divergencia mide cuánta información se pierde cuando se usa el espacio q para representar p .

Otro aspecto importante del VAE es que el espacio $p(z)$ queda definido una distribución Normal de media cero varianza unidad o $p(z) = \text{Normal}(0, 1)$. Si la salida de codificador difiere de esa distribución estadística, el codificador recibirá una penalización.

Codificación y Decodificación

La eficacia y tratabilidad del método reside en la asunción de que $Q(z|X)$, la función codificadora, puede ser modelada como una gaussiana con una media determinada $\mu(X)$ y varianza $\Sigma(X)$, por otro lado es necesario que $P(X)$ converja de manera eficaz a la distribución real de los datos del espacio D . Estas condiciones solo son superadas si y solo si $D[Q(z|X)||P(z|X)]$ es cercana a cero.

Es por ello necesario una función Q de alta capacidad, lo cual puede llevarnos a modelos complejos. Los modelos basados en funciones usados en los VAE son las redes neuronales

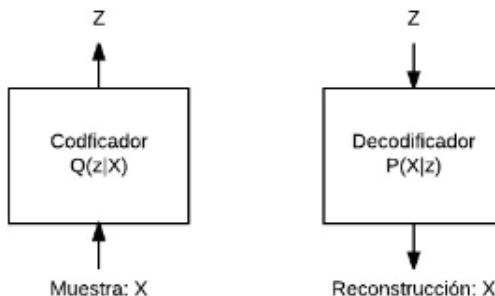


Figura 3.3: Esquematización simple de las funciones del Codificador y el Decodificador en el VAE

El codificador es una red neuronal. Su entrada es el dato X y su salida es la

representación latente z . Representa la distribución de probabilidad $Q(z|X)$, y esta determinada por el conjunto de parámetros y pesos de la red neuronal asociada. Denominaremos a la función encargada de la codificación $q_\theta(z|x)$

El codificador se identifica a menudo con el proceso de reducción de la dimensionalidad de x a z . Cabe notar que el codificador tiene asociadas dos funciones, una encargada de obtener la media $q_\mu(X)$ y otro la varianza $q_\Sigma(X)$ del espacio latente. Para la obtención final de z se ha de aplicar el truco de reparametrización, ver sección 3.1.5, con respecto a los valores Σ y μ obtenidos anteriormente.

El decodificador es otra red neuronal. Su entrada es la variable del espacio latente z y su salida es la reconstrucción del dato inicial X . Denominaremos a la función encargada de la decodificación $p_\phi(x|z)$, donde ϕ son el conjunto de parámetros y pesos que definen la red neuronal.

El hecho de que ambas funciones estén basadas en redes neuronales hace el aprendizaje profundo sea una parte primordial del VAE. Típicamente los formatos de redes neuronales aplicados en este sistema son dos; redes neuronales densas (DNN) o redes neuronales convolucionales (CNN).

Redes Neuronales

Las Redes Neuronales permiten generar funciones complejas no lineales gracias a su capacidad inherente de aprendizaje con el proceso denominado propagación hacia atrás, que permiten ajustar los pesos de las distintas unidades o neuronas del sistema.

Dada la complejidad del ámbito del aprendizaje profundo, en las siguientes secciones se pretenden exponer las ideas fundamentales para comprender el comportamiento de las funciones de codificación y decodificación del VAE, sin entrar en explicaciones excesivamente teóricas sobre los fundamentos de las redes neuronales.

Es por ello que en primer lugar se expondrá el modelo de redes neuronales densas, aprovechando para exponer de manera somera algunos conceptos de redes neuronales, como son el concepto de funciones de activación o el proceso de propagación hacia atrás.

Seguidamente se expondrá el otro modelo de aprendizaje profundo utilizado en este trabajo que son las redes neuronales convolucionales, explicando por qué son ideales para la captura de patrones sobre imágenes.

Red Neuronal Densa

Este modelo constituye el paradigma básico de redes neuronales. Fundamentado en el estándar de neuronal artificial según los principios descritos Rumelhart y McClelland en 1986 [57]. Siguiendo dichos principios, la i -ésima neurona artificial consiste en:

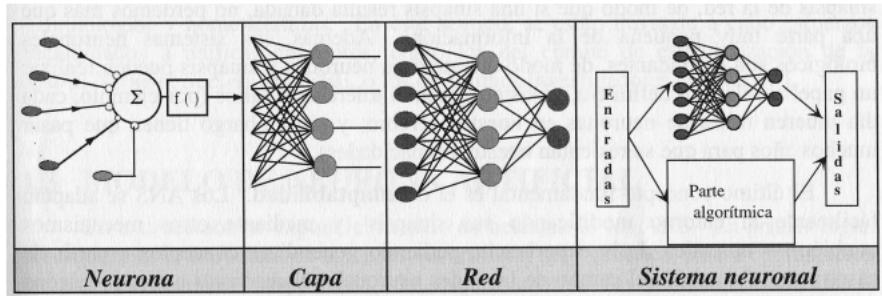


Figura 3.4: Sistema global de proceso de una red neuronal

- Un conjunto de entradas x_j con un conjunto de pesos sinápticos asociados w_{ij} , con $j = 1, 2 \dots n$

- Una regla de propagación h_i a definida partir del conjunto de entradas y de los pesos sinápticos. Normalmente la regla de propagación utilizada el producto lineal entre los pesos sinápticas y las entradas. Esto es:

$$h_i(x_1, \dots, x_m, w_{i1}, \dots, w_{in}) = \sum_{j=1}^n w_{ij} * x_j \quad (3.15)$$

- Una función de activación, la cual representa simultáneamente la salida de la neurona y su estado de activación. Denotando por y_i dicha función de activación:

$$y_i = f_i(h_i) = f_i(\sum_{j=0}^n w_{ij}x_j) \quad (3.16)$$

Función de Activación

La elección de la función de activación constituye una parte determinante en el diseño de redes neuronales, dado que afectará en gran medida al la capacidad de decisiónnd de la red y la rapidez con que la red sea capaz de converger durante el entrenamiento [58].

En general el principal requerimiento sobre estas funciones es que sean capaces de respetar el proceso del propagación hacia atrás, no provocando que el gradiente se haga cero lo cual repercutiría negativamente en el proceso del descenso en gradiente. Este es uno de los problemas asociadas a la clasica función sigmoide, dado que para

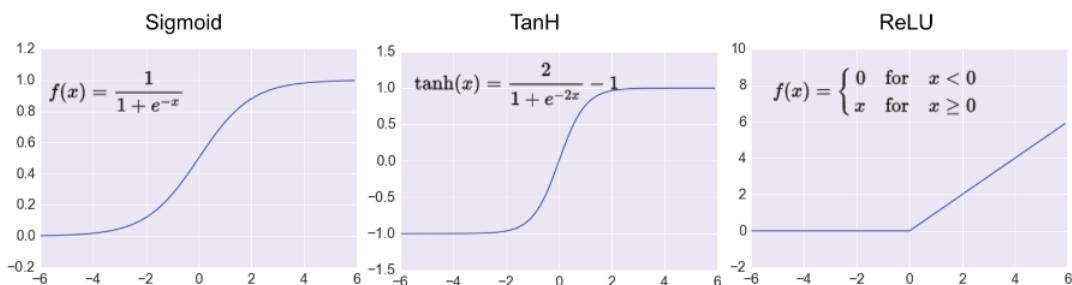


Figura 3.5: Principales funciones de activación.

valores de x ampliamente negativos o positivos, provoca que el gradiente sea cero³, interrumpiendo el descenso en gradiente para la neurona en cuestión y, por tanto, la optimización de sus pesos sinápticos.

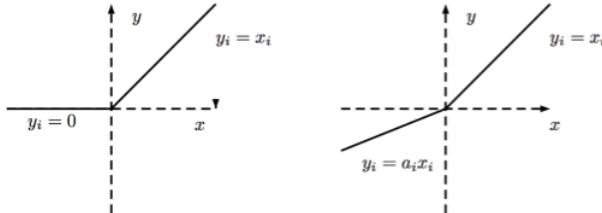


Figura 3.6: (izquierda) Función de activación *Relu*. (Derecha) Función de activación *leakyRelu*.

Actualmente la función de activación más utilizada es la unidad lineal de rectificación [59] (ReLU del inglés *Rectifier Linear Unit*), representada en la figura 3.5. No obstante, otro tipo de función de activación basada en la anteriormente expuesta denominada unidad lineal de rectificación con pérdidas (leakyRelu) ha ganado peso en el ámbito. La única diferencia entre ambas funciones es la capacidad de la *leakyRelu* de no hacer nulo el gradiente para valores negativos, ver figura 3.6 para apreciar esta diferencia. En este proyecto han sido utilizadas tanto la función Sigmoid como la función *leakyRelu*

Topología de Conexiónado

Otro concepto determinante en el comportamiento de las redes neuronales es la topología empleada, esto es, el patrón de conexiónado de una red neuronal. En una red neuronal artificial los nodos se conectan entre sí, siendo este conjunto de conexiones internas junto con los pesos sinápticos lo que determina el comportamiento de la red y, en última instancia, la función asociada a la red.

Las unidades neuronales suelen agruparse en lo que se denominan capas. La unión de dos o más capas constituyen una red neuronal. Se distinguen tres tipos de capas: de entrada, de salida y ocultas. Una capa de entrada está compuesta por las neuronas que reciben las señales. Una capa de salida está constituida por el conjunto de neuronas que proporcionan la respuesta de la red. Las capas ocultas no tienen conexiónado con el exterior. A más capas Socultas más capacidad de aprendizaje tendrá el sistema, aunque el tiempo necesario para su optimización aumentará considerablemente.

³Este efecto es comúnmente denominado como saturación

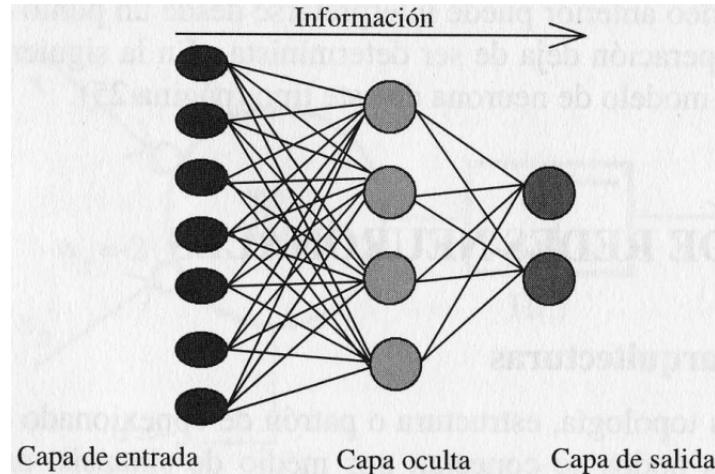


Figura 3.7: Esquema de una red neuronal densa de una sola capa oculta

Propagación Hacia Atrás

Se denomina propagación hacia atrás al proceso empleado para el entrenamiento de las redes neuronales. Este entrenamiento tiene como objetivo el ajuste de los pesos sinápticos de la red. Se considera un buen ajuste de pesos aquel que minimiza el error a la salida de una red [60]. De manera breve los principales pasos de este proceso de entrenamiento son:

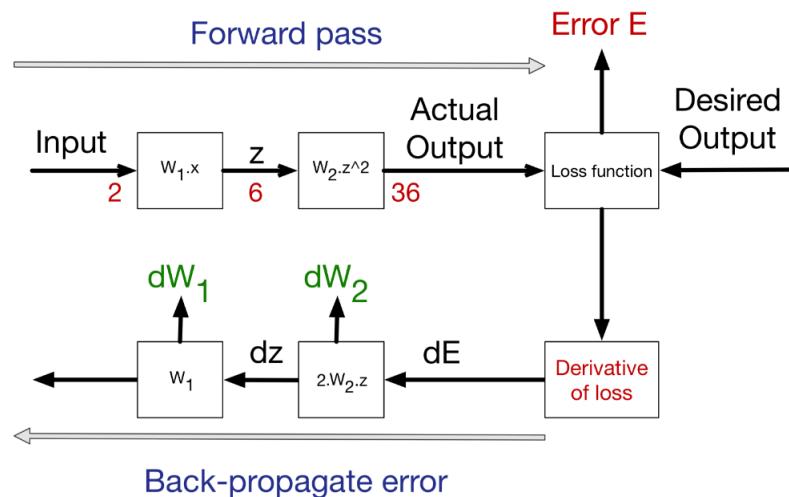


Figura 3.8: Representación esquemática del proceso de entrenamiento de una red neuronal

- Inicialización. Se asigna un valor por defecto a los distintos pesos. Se considera un paso determinante, puesto que una mala inicialización puede implicar la saturación de los gradientes en los nodos.

Los siguientes pasos constituyen un proceso iterativo, durante el cual se irá minimizando progresivamente el error asociado a la salida de la red.

- Paso hacia delante (*Fast Forward*). Se comprueba el comportamiento de la red, se calcula la salida de la red para un conjunto de muestras de entrada.
- Estimación del error de salida. Dada una salida, se evalúa la diferencia con respecto a la salida esperada según las muestras de entrada.
- Se realiza la propagación hacia atrás. Dado el error a la salida se realizan el conjunto de derivadas necesarias recorriendo desde la salida hacia la entrada la red, identificando el comportamiento del gradiente del error con respecto a los diferentes pesos de la red.
- Se modifican los pesos en función del gradiente previamente calculado.

Red Neuronal Convolucional

Las redes convolucionales (CNN, del inglés *Convolutional Neural Networks*) son una categoría de redes neuronales que se consideran un método altamente eficaz en áreas como el reconocimiento de imágenes [62][63].

Este modelo fué introducido en 1989 [61] por Yann leCunn, la red de este trabajo fue denominada *LeNet5*. Dicha red se puede observar en la imagen 3.9

Las redes convoluciones suelen ser aplicadas a las imágenes. Cada imagen puede ser representada por una matriz de números sí se trata de una imagen en escala de grises, o por tres matrices sí es una imagen a color. Es esta propiedad de las imágenes donde cada dimensión, es decir cada pixel, queda definida espacialmente con respecto al resto de dimensiones, lo que convierte a las imágenes en las muestras ideales para este tipo de red.

Se asume que los conjuntos de pixeles vecinos formarán unas características más significativas que sí tomaramos grupos sin tener en cuenta su disposición espacial

a capa anterior lo que se emplea es el operador de convolución.

Este tipo de redes derivan su nombre del operador de red convolución cuyo objetivo es extraer características de las imágenes preservando la relación espacial

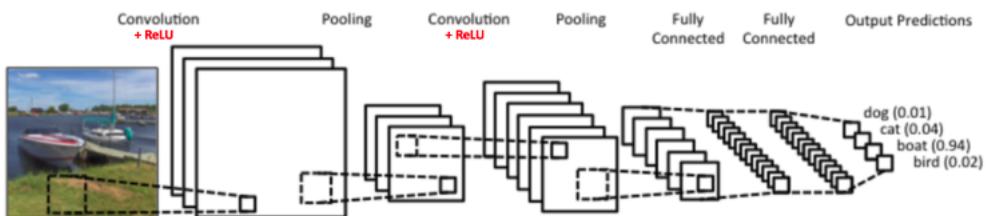


Figura 3.9: Red convolucional *LeNet5*

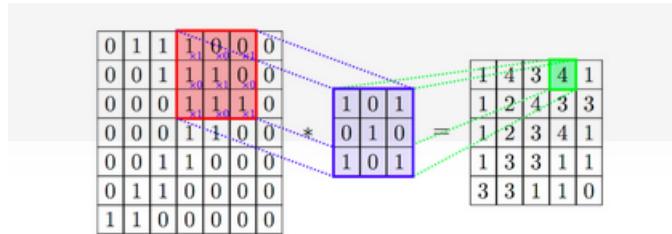


Figura 3.10: Ejemplo de aplicación del operador convolución sobre una imagen. Seleccionada una región de la imagen cuyas dimensiones son las mismas que las del kernel seleccionado, se aplica el producto pixel a pixel entre dicha región y los pesos propios del kernel. La suma de estos productos se almacena en la imagen de salida, respetando la ubicación espacial de la región evaluada.

entre pixeles.

Dada una imagen bidimensional I y una matriz K de dimensiones $h \times w$ (denominada kernel de convolución) la cual es capaz de extraer algún tipo de característica relevante. La operación de convolución se puede representar como:

Formalmente, se puede expresar como:

$$(I * K) : xy = \sum_{i=1}^h \sum_{j=1}^w K_{ij} I_{x+i-1, u+j-1} \quad (3.17)$$

A diferencia de las redes neuronales convencionales en las redes convolucionales los datos a la entrada y entre el conexiónado de capas se agrupan en 3 dimensiones: ancho, alto y profundidad. En este caso nos referimos a "profundidad" por capa no a la profundidad de la red, lo cual se refiere al número de capas de la red en cuestión. Por ejemplo, dada una imagen de entrada de tres canales (los tres canales de color) de 32x32 pixeles, la agrupación de los datos en la capa de entrada será 32x32x3. Ver figura 3.11

Otra diferencia con respecto a las redes neuronales clásicas es que las unidades de una capa solo están conectadas a un espacio reducido de unidades de la capa inmediatamente anterior.

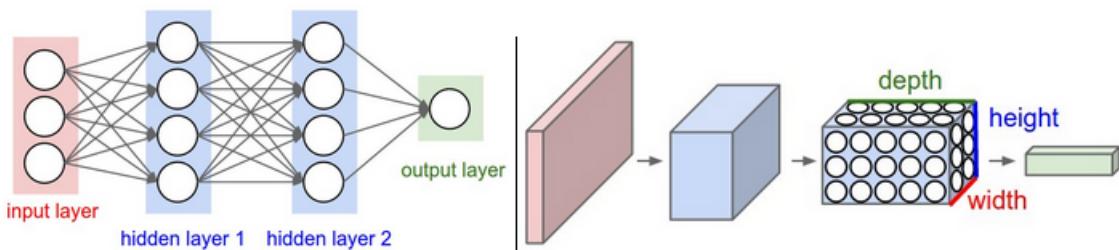


Figura 3.11: (Izquierda) Modelo clásico de redes neuronales. (Derecha) Modelo de Red Convolucional. Los datos son reagrupados en 3 dimensionados como se puede observar en una de las capas. Cada una de las capas tiene como entrada una imagen 3D y tiene como salida otra imagen 3D. La capa roja representa la capa de entrada por lo que la altura y la anchura son las dimensiones de la imagen y la profundidad son el número de canales

Las redes neuronales convolucionales se fundamentan en tres principios básicos que son los campos receptivos locales, los pesos compartidos y el empleo de agrupaciones o *pooling*

Filtros Locales

En una red neuronal densa, esto es, una red totalmente conectada como la de imagen 3.11 las entradas se interpretan como un conjunto "vertical" de unidades. Sin embargo, en un red convolucional es preferible organizar las unidades de entrada en forma bidimensional.

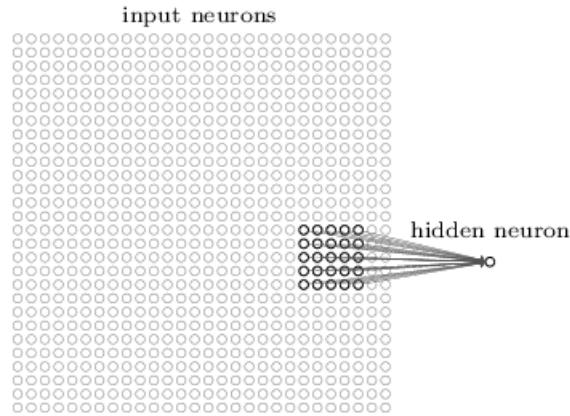


Figura 3.12: Representación de la conectividad local en una red neuronal

Las capas consecutivas estarán conectadas entre sí, pero cada unidad de una capa oculta estará conectada solo a un conjunto de unidades de la capa inmediatamente anterior.

Se denomina filtro local a la ventana que se aplica a las diferentes regiones seleccionables de la imagen, cada una de estas regiones seleccionables estan conectadas a una única unidad de la siguiente capa oculta. A este término a menudo nos referimos como kernel. Este filtro se desplazará por toda la imagen, realizando el proceso de

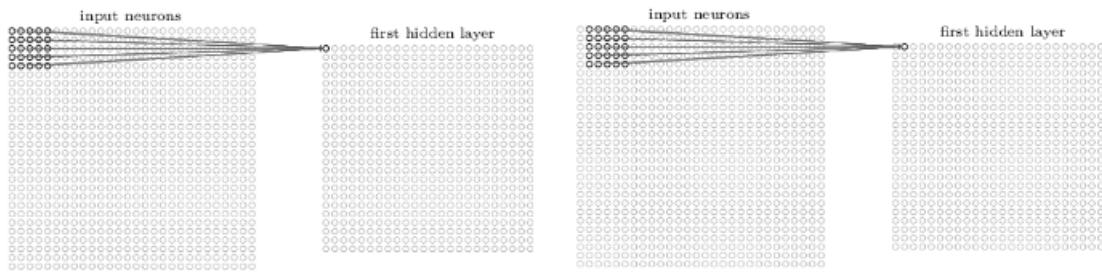


Figura 3.13: Desplazamiento del campo de recepción

convolución por toda ella, ver imagen 3.13. Es esto lo que permite extraer características de manera local por toda la imagen

Normalmente el desplazamiento se hace pixel a pixel aunque es posible aumentar el número de pixeles por desplazamiento. Este hiperparámetro se denomina generalmente *stride*. En este trabajo se ha utilizado un valor de dos. Otro concepto a tener en cuenta es que por lo general hay varios tipos de filtros para la extracción de características en las diferentes capas.

Pesos Compartidos

Cada uno de los filtros de recepción serán aplicados a toda la imagen con el mismo peso para todas las diferentes regiones. Esto significa que el patrón de selección de características será el mismo, por lo que las neuronas de la siguiente capa detectarán el mismo tipo de característica.

El punto anterior se fundamenta en que generalmente un patrón de una parte de la imagen es probable que se repita en otra parte de la imagen dada la propia naturaleza de las imágenes.

Con objeto de no limitar cada capa a la extracción de un tipo de característica se aplican numeros filtros en cada una de las capas de convolución. Gracias a esto se consiguen extraer distintos tipos de patrones.

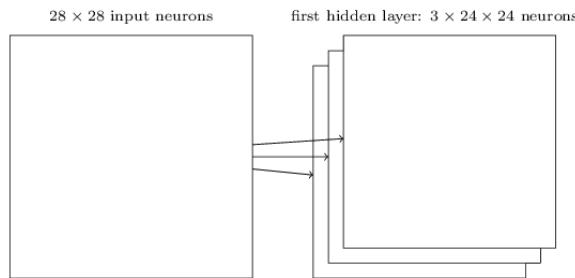


Figura 3.14: Representación de la extracción de varias características con varios filtros

Una de las ventajas del uso de pesos compartidos es que permite reducir el número de parámetros de la red.

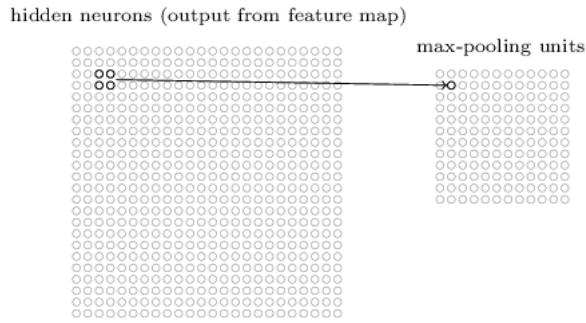


Figura 3.15: Representación del proceso de agrupamiento (*pooling*)

Agrupamiento

Otro tipo de capa característica de las redes convolucionales son las capas de agrupamiento o *pooling*. Esta capa tiene como objetivo reducir el número de datos generado, realizando una estimación del valor más importante de una determinada región. Esto permite reducir progresivamente el tamaño de la imagen. Este proceso de agrupamiento se aplica individualmente a cada una de las imágenes generadas por cada filtro.

No obstante, esta funcionalidad no ha sido utilizada en el modelo generado en este trabajo dado que actualmente la librería empleada (*TensorFlow*) no tiene implementada esta operación para imágenes 3D.

Herramientas Complementarias

Con objeto de evaluar la validez y la capacidad de diferenciación entre imágenes de los datos generados en la capa latente de los modelos de VAE empleados, se han utilizado un conjunto de procedimientos de clasificación que serán expuestos en la sección 4.5. Los fundamentos teóricos de los métodos empleados serán explicados a continuación de manera resumida.

Máquina de Vectores de Soporte

El método denominado Máquina de Vectores de Soporte (SVM, del inglés *Support Vector Machine*) ha sido ampliamente usado para la clasificación y regresión [64] [65], diseñado para la separación de un conjunto binario de datos, previamente etiquetados, mediante un hiperplano, ver figura 3.16.

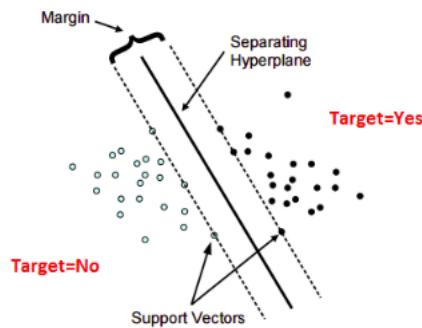


Figura 3.16: Representación de la separación binaria realizada por SVM

Específicamente, se emplea un método de optimización que pretende establecer el hiperplano que logre la mayor separación posible entre clases, usando una función de decisión de la forma $R^n \rightarrow \{\pm 1\}$, correspondiéndose con un espacio $n - dimensional$ de vectores de entrenamiento y etiquetas de clase y_i :

$$(f_1, y_1), (f_2, y_2), \dots, (f_s, y_s) \in R^n \times \{\pm 1\} \quad (3.18)$$

de tal manera que g es una función lineal capaz de clasificar nuevas muestras (f, y) . Dicha función es la encargada de definir el hiperplano de decisión y tiene la siguiente forma:

$$g(f) = w^T f + v_0 \quad (3.19)$$

donde w es un vector de pesos mientras que v_0 es el sesgo (elemento encargado de establecer el umbral entre clases). Por lo tanto, la división de clases a partir de la función f será:

$$y_i = \begin{cases} +1 & \text{si } w^T f + v_0 \geq +1 \\ -1 & \text{si } w^T f + v_0 \leq -1 \end{cases} \quad (3.20)$$

donde el vector de pesos w ha de ser ortogonal al hiperplano de decisión. El proceso de optimización es el encargado de encontrar los parámetros desconocidos de w and v_0 de la expresión 3.19, dichos parámetros son los que definen el hiperplano óptimo que separa las dos clases en cuestión.

Adicionalmente, es posible estimar la importancia relativa de cada una de las n características evaluadas. De hecho, sea N_s el número de vectores de soporte empleados durante la fase de entrenamiento, el siguiente vector (W) puede ser calculado como:

$$W = \sum_{j=1}^{N_s} y_j, \lambda_j, f_j, \quad (3.21)$$

donde y_j son las etiquetas de cada una de las muestras, λ_j son los parámetros Lagrangianos correspondientes, los cuales también son optimizados durante la fase de entrenamiento y f_j son las muestras de entrenamiento. La coordenada i del vector W , (W_i con $1 \leq i \leq n$) hace referencia a la relevancia de la dimensión i del vector de características f_j [66]. Más precisamente cuanto mayor sea el valor absoluto de W_i mayor será la relevancia de la característica i . Por contraste $|W_i| = 0$ indica que la característica i no aporta ningún tipo de información en la clasificación.

Métricas de Evaluación Estadística

Hay diferentes métricas de evaluación referidas a las características del diagnóstico. Algunas se utilizan para evaluar la capacidad discriminativa de la prueba y otras para estimar su propiedad de predicción, siendo esta última muy sensible a las características de las poblaciones evaluadas.

Con objeto de valorar la calidad del test de diagnóstico es necesario saber cuán buena y confiable es una prueba. Esta cuantificación es llevada por el conjunto de medidas que se explicarán a continuación, las cuales están basadas en los índices expuestos en la tabla 3.3.2.

		Condición Real	
		Positivo	Negativo
Predicción	Positivo	Verdadero Positivo (VP)	Falso Positivo (FP)
	Negativo	Falso Negativo (FN)	Verdadero Negativo (VN)

Tabla 3.1: Tabla de Nomenclatura Estadística en Clasificación

Medida F

$$Valor - F = \frac{*VP}{2 * VP + FP + FN} \quad (3.22)$$

Recall. Sensibilidad

La precisión mide la capacidad de una prueba diagnóstica de identificar los sujetos enfermos con respecto al total de sujetos enfermos de la población.

$$Precision = \frac{VP}{VP + FN} \quad (3.23)$$

Especificidad. Precisión

La especificidad hace referencia a la capacidad de la prueba a identificar a los sujetos enfermos y excluir a aquellos que no lo están.

$$Precision = \frac{VP}{VP + FP} \quad (3.24)$$

Precision. Accuracy

Indica la capacidad la del método de clasificación de identificar tanto pacientes sanos como pacientes enfermos con respecto al total de la población.

$$F1 = \frac{VP + VN}{NºSamples} \quad (3.25)$$

Curva ROC

La curva ROC (del inglés *Receiver Operating Characteristic*) es una técnica gráfica que nos permite evaluar la precisión del modelo estadístico para clasificar dos clases, AD y NOR. La curva se obtiene calculando la sensibilidad (proporción de resultados positivos verdaderos) y la especificidad del modelo en cada punto de corte posible, y trazando la sensibilidad frente a 1-especificidad (proporción de resultados falsos positivos).

Cada punto en el espacio ROC muestra el equilibrio entre la sensibilidad y la especificidad, es decir, que el aumento de sensibilidad va acompañado de una disminución en la especificidad.

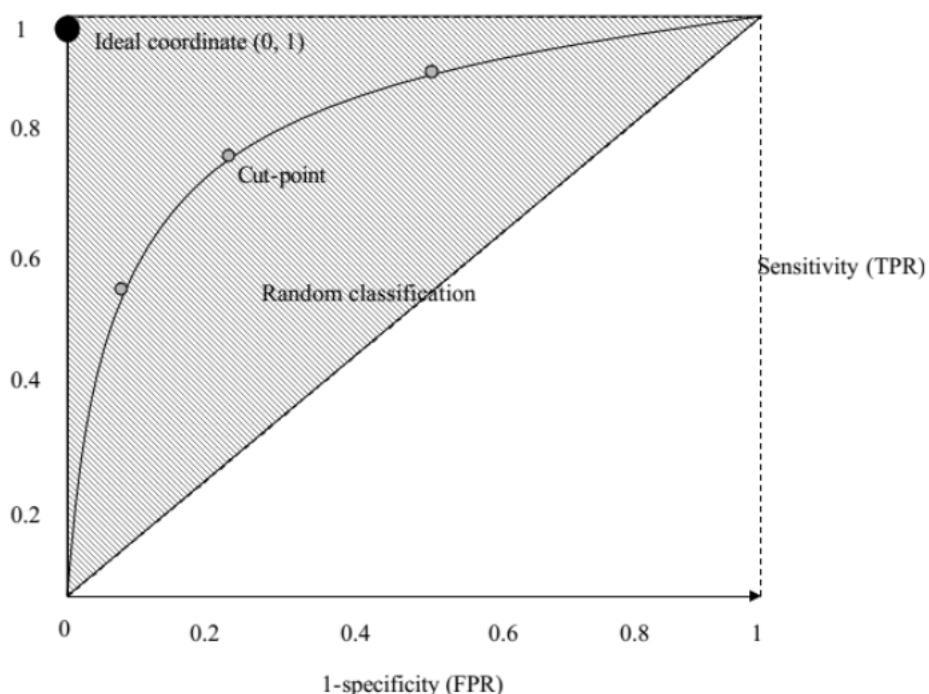


Figura 3.17: Representación de la curva ROC

Cada punto de corte de una prueba de diagnóstico define un único punto en el espacio ROC, los diferentes puntos posibles definen la curva ROC. Esto es análogo a lo dicho para un solo punto, por tanto cuanto más se acerquen los puntos de la curva ROC a la coordenada ideal, más exacta será la prueba y viceversa, cuanto más se aleje peores resultados obtendremos.

Área bajo la curva

El área bajo la curva ROC se denomina AUC (del inglés *area under the curve*) y se interpreta como el promedio de precisiones positivas y negativas. Este índice es especialmente útil en los estudio comparativo de pruebas de diagnóstico. Siendo deseable comparar toda la curva ROC en lugar de en un punto particular.

Validación Cruzada

La validación es una técnica utilizada para evaluar los resultados de un análisis estadístico y garantizar que son independientes de la partición entre datos de entrenamiento y de prueba. En cualquier análisis estadístico es necesario aplicar un conjunto de datos diferentes durante el entrenamiento y durante la fase de prueba, con objeto de evitar que los resultados y por tanto el sistema diseñado estén sesgados al conjunto de datos con los que se ha entrenado [67].

En este trabajo, para la construcción de los conjuntos de prueba y entrenamiento se usará el método conocido como *k-folds*. En este método los datos se dividen en *k* subconjuntos iguales, realizando *k* iteraciones. En cada iteración se toma uno de los subconjuntos como datos de prueba y el resto (*k*-1) como datos de entrenamiento. El proceso de validación cruzada es repetido durante *k* iteraciones, con cada uno de los subconjuntos como datos de prueba, ver Fig. 3.18

Las métricas de evaluación finales se calculan como la media aritmética de las métricas cada iteración para obtener un único resultado, evitando que los resultados estén sesgados al espacio muestral.

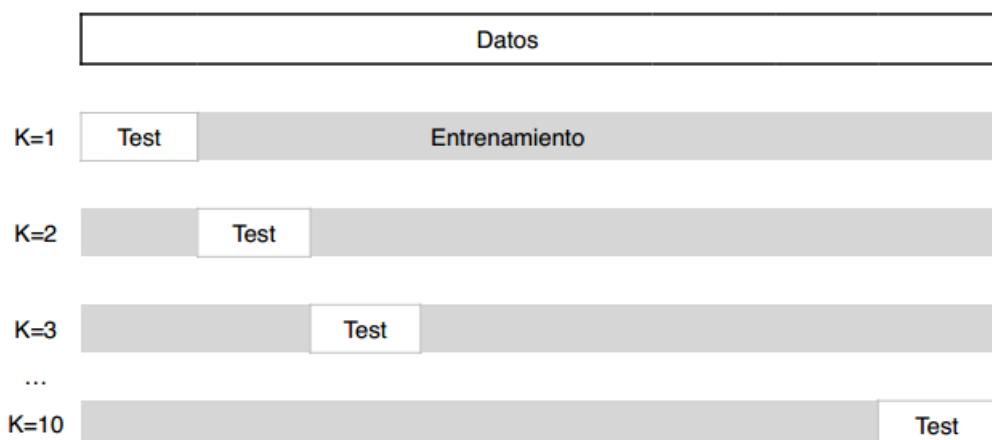


Figura 3.18: Proceso de Validación Cruzada

Capítulo 4

Trabajo Realizado

Estudio basado en régiones cerebrales

Uno aspecto clave de este trabajo consiste en la división del cerebro en diferentes áreas con objeto de ser caracterizadas de manera aislada, y en última instancia, poder generar cada una de las áreas o régiones por separado. Para llevar a cabo esta separación se ha usado en atlas AAL (del inglés *Automated Anatomical Labeling*) [68] que define un total de 116 regiones las cuales se corresponden con las diferentes áreas anatómicas. Este atlas permite obtener los voxels asociados a cada región de manera normalizada.

Aunque esta aproximación tiene la ventaja de permitirnos caracterizar las regiones por separado, el principal motivo por el que hemos el alto coste computacional que lleva asociado el uso de redes de aprendizaje profundo cuando son aplicadas en datos de alta dimensionalidad, como es nuestro caso. Otro problema derivado el amplio tiempo necesario para la caracterización de los parámetros de la red.

El uso de una aproximación basada en regiones nos permite reducir de forma considerable el número de voxels a caracterizar por cada red neuronal y por lo tanto reducir los tiempos de caracterización y los costes computacionales.

Dentro de las 116 regiones en las que se han dividido las neuroimágenes, normalmente aquellas a las que se les atribuye que aportan información sobre la detección del AD se las denomina Régiones de Interés (ROI, del inglés *Regions of Interest*).

Se ha almacenado el atlas AAL junto a cada uno de las distintas modalidades de imágenes empleadas que son las MRI y las PET. Esto nos permite extraer las regiones indicadas mediante el Atlas de las distintas imágenes, lo cual nos posibilita hacer el procesado de regiones de manera independiente. En la imagen se 4.2 muestra un ejemplo de imagen PET y MRI con su atlas correspondiente al lado.

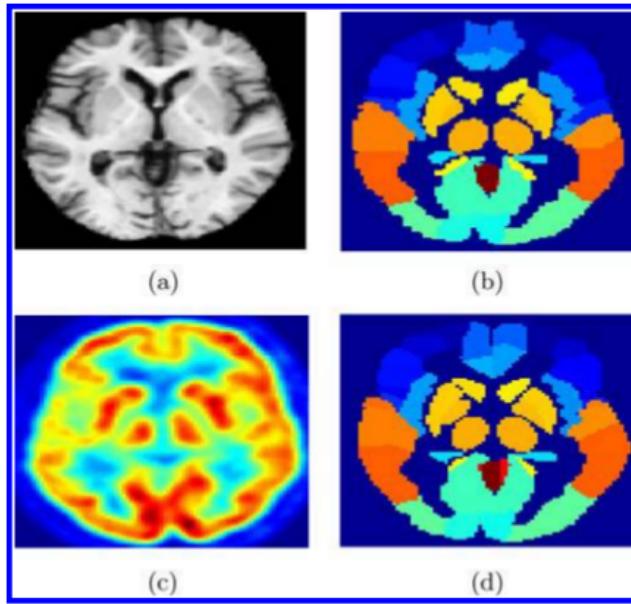


Figura 4.1: MRI image (a), MRI atlas (b), (c) PET image and (d) PET atlas (same slice is shown in MRI and PET images)

Tratamiento de Neuroimágenes

En todo proceso de caracterización de muestras o de aprendizaje estadístico un aspecto esencial es aplicar un tratamiento y procesado efectivo de las muestras previo al algoritmo principal, ya que si las muestras no son correctas o se producen irregularidades en su tratamiento previo se estará avocando a unos resultados incorrectos (por muy bien elaborado que esté el algoritmo principal).

En primer lugar se explicarán las características demográficas de las neuroimágenes empleadas. Posteriormente se expondrá de manera resumida el procesado de las imágenes realizado por el grupo de investigación de And?es Ortiz.

Fuente de Datos

Las neuroimágenes empleadas en este trabajo pertenecen a la iniciativa ADNI. Se han empleado tanto imágenes MRI como imágenes PET. Se disponen de 229 imágenes MRI de sujetos NC y 188 de sujetos de AD. Por otro lado, en el caso de las imágenes PET se disponen de 70 imágenes de sujetos AD y 68 de sujetos NC. La distribución demográfica de los sujetos se puede observar en las tablas 4.1 y ??

Diagnosis	Number	Age	Gender (M/F)	MMSE
Control	68	75.81 ± 4.93	43/25	29.06 ± 1.08
AD	70	73.06 ±	46/24	22.84 ± 2.61

Tabla 4.1: Datos Demográficos Imágenes PET

Procesado Previo

Las imágenes PET y MRI de la base de datos ADNI han sido espacialmente normalizadas de acuerdo con el modelo de morfometría basada en vóxeles T1 [69] (*VBM-T1* del inglés *Voxel-Based Morphology*), con objeto de garantizar que cada vóxel corresponde con la misma posición anatómica en cada una de las neuroimágenes. Posteriormente las imágenes MRI fueron redimensionadas a $121 \times 145 \times 121$ vóxeles con un tamaño de vóxel de 1.5 mm (Sagital) \times 1.5 mm (Coronal) \times 1.5 mm (axial). Por otro lado, las imágenes PET fueron redimensionadas a $79 \times 95 \times 68$ con un tamaño de vóxel de 3 mm (Sagital) \times 3 mm (Coronal) \times 2 mm (axial).

Las imágenes MRI son tratadas de manera diferente que las PET ya que son segmentadas en tejido de materia gris (GM del inglés *Grey Matter*) y tejido de materia blanca (WM del inglés *White Matter*) aplicando la herramienta SPM [70][71] de normalización espacial. Este proceso es capaz de generar información sobre la distribución del tejido de GM, de WM o de fluido Cerebro-Espinal (CSF del inglés *Cerebrospinal Fluid*) en las neuroimágenes, quedando caracterizado por una probabilidad de pertenencia para cada uno de los tejidos de rango [0, 1].

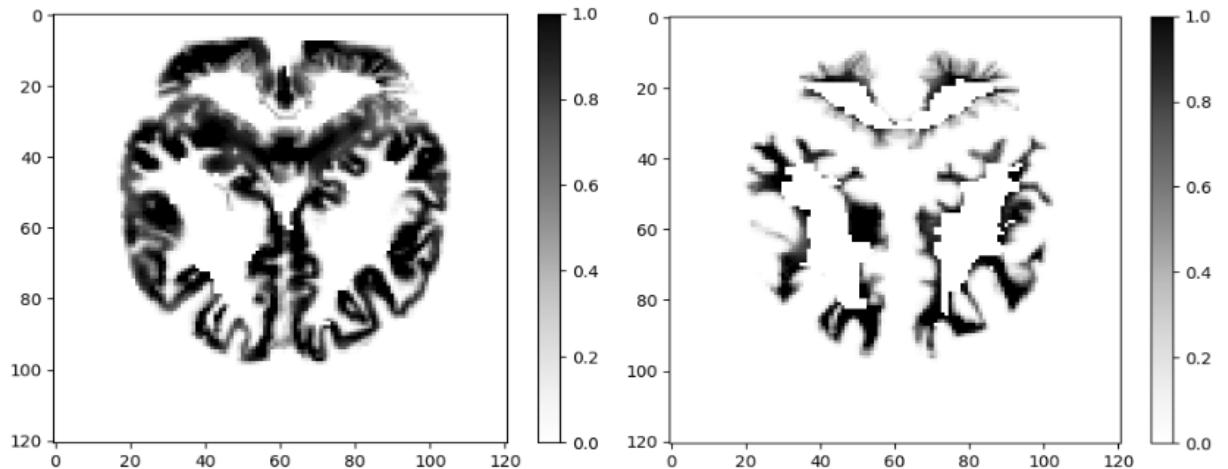


Figura 4.2: Muestra de neuroimagen MRI segmentada. (zquierda) MRI WM image. (Derecha) MRI GM image

Por otro lado, las imágenes PET son normalizadas con respecto al nivel de in-

tensidad. Este nivel máximo de intensidad se toma a partir del nivel medio del 1% de los véxeles con mayor activación del cerebro [72], dado que esta región cerebral es considerada con activación constante. Este proceso de normalización permite la homogeneización de los niveles entre los véxeles permitiéndole la posterior comparación entre véxeles.

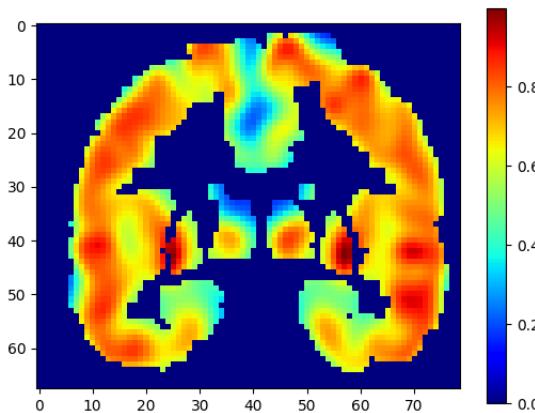


Figura 4.3: Muestra de neuroimagen PET normalizada

Preselección de Véxeles

La preselección de véxeles se ha aplicado a cada modalidad de imagen con objeto de eliminar los véxeles poco significativos. Esto nos permite reducir el alto coste computacional asociado a la alta dimensionalidad de las imágenes. Esta preselección de características ha sido realizada mediante el *t-test de Welch* separadamente sobre cada tipo de imagen.

El *t-test Welch* permite evaluar la diferencia entre la media de dos espacios muestrales, en nuestro caso NC y AD, cuando las varianzas no son iguales y puede ser calculado usando la siguiente expresión:

$$I^t = \frac{I_{NC}^\mu - I_{AD}^\mu}{\sqrt{\frac{I_{NC}^\sigma}{N_{NC}} + \frac{I_{AD}^\sigma}{N_{AD}}}} \quad (4.1)$$

donde I_{NC}^μ y I_{AD}^μ son las medias de las imágenes de los sujetos NC y AD respectivamente, mientras que I_{NC}^σ y I_{AD}^σ son las varianzas de las imágenes y N_{NC}, N_{AD} son el número de muestras NC y AD. Las imágenes de medias I_{NC}^μ y I_{AD}^μ se calculan como:

$$I_{NC}^\mu = \frac{1}{N_{NC}} \sum_{j=1}^{N_{NC}} I_j, \quad I_{AD}^\mu = \frac{1}{N_{AD}} \sum_{j=1}^{N_{AD}} I_j, \quad (4.2)$$

mientras que las varianzas de las imágenes I_{NC}^σ y I_{AD}^σ son calculadas mediante:

$$I_{NC}^\sigma = \frac{1}{N_{NC}} \sum_{j=1}^{N_{NC}} (I_j - I_{NC}^\mu)^2, \quad I_{AD}^\sigma = \frac{1}{N_{AD}} \sum_{j=1}^{N_{AD}} (I_j - I_{AD}^\mu)^2 \quad (4.3)$$

En la ecuación 4.1 el término I^t corresponde al valor del test *Welch* para cada uno de los vóxeles de la imagen, lo cual es una medida significativa de la diferencia de medias. De manera intuitiva un alto valor de este elemento indica que hay una diferencia significativa entre las muestras de un espacio y otro, y, por lo tanto, el vóxel en cuestión es significativo.

De manera teórica, altos valores del test (*t-valor*) se corresponden con valores bajos de probabilidad (*p-valor*), donde se referencia por *p-valor* la probabilidad de observar un valor *t-valor*. Queda definida la hipótesis nula en la igualdad entre las medias de imágenes. Por lo tanto, valores pequeños de *p* indicarán el rechazo de la hipótesis nula en cuestión.

En nuestro caso, se ha fijado el umbral de decisión sobre la hipótesis nula en $p - \text{valor} < 0,05$, esto es, un valor de significancia del 5 %.

Segmentación basada en régiones

Tal y como se ha comentado al principio de este capítulo, un aspecto básico de el trabajo realizado es la división o segmentación de las neuroimágenes en regiones para su estudio posterior, lo cual conlleva un procesado asociado.

Dado que se han empleado dos modelos de aprendizaje bien diferenciados será necesario llevar a cabo una segmentación de regiones diferente para cada tipo. Uno de los modelos de aprendizaje está basada en el estudio de las imágenes 3D de las regiones mientras que el otro está basado en caracterizar un vector de vóxeles pertenecientes a la región estudiada.

Componentes iniciales

Es necesario analizar el conjunto de datos iniciales a partir de los cuales se obtendrán las imágenes segmentadas, ya sea en formato vectorial 1D o en formato de imagen 3D.

Vector de véxeles de imagen. Por cada neuroimagen de cada paciente, ya sea una imagen PET o MRI se tendrá un vector de véxeles asociado. Este vector contiene los valores de intensidad de los véxeles dispuesto en forma vectorial en lugar de en una imagen 3D.

Cada uno de estos vectores de véxeles por paciente contendrán únicamente los véxeles útiles de las imágenes, esto es, aquellos véxeles que se consideran que no forman parte del entorn que envuelve al cerebro. A menudo nos referiremos a este término como *background*.

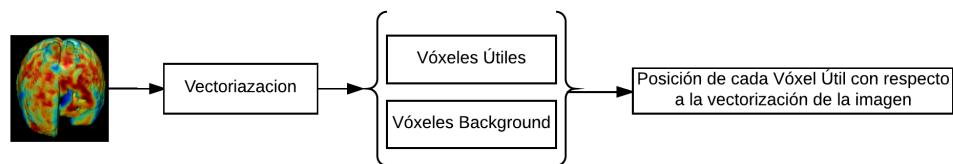


Figura 4.4: Diagrama del proceso de vectorización de neuroimágenes 3D, con la posterior división en Vóxeles Útiles y Vóxeles de *background*

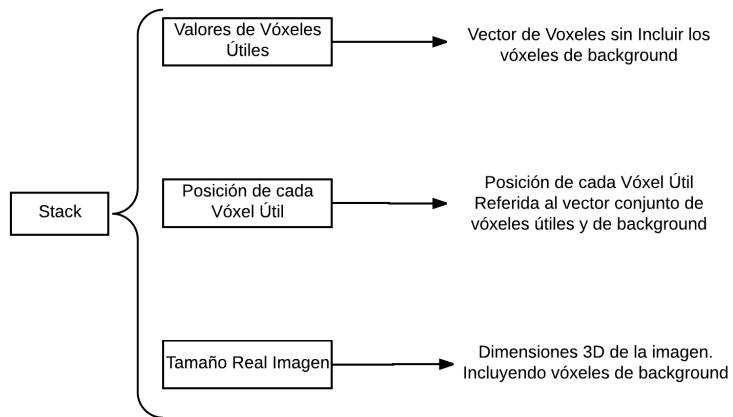


Figura 4.5: Elementos del Stack de Imágenes

Es importante notar que en los datos empleados se tiene una matriz de datos por cada tipo de imagen, ya sea PET o MRI de materia blanca o materia gris. Cada fila de esta matriz se referirá a los véxeles útiles vectorizados de cada paciente. Es por ello crucial tener una referencia real de la posición de cada véxel útil sobre la imagen vectorizada conjunta de véxeles útiles y de *background*.

Otro aspecto necesario es poder contar con el tamaño original de la imagen 3D para poder redimensionar los datos de vector a imagen en 3D si así se desea. Este conjunto de elementos necesarios para el tratamiento de las imágenes están agrupados en lo que se ha denominado *stack*. Ver fig. 4.5

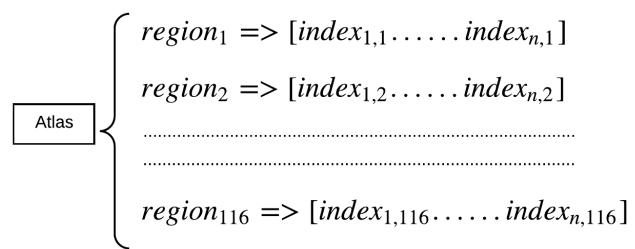


Figura 4.6: Diagrama de Atlas AAL de Régiones

Atlas AAL. El Atlas contiene un total de 116 listas distintas, cada una de ellas asociadas a una de las regiones. Cada lista contiene un conjunto de índices referidos a la posición de los vóxeles en el vector de vóxeles útiles de la región evaluada. Es decir, son índices referidos al vector de vóxeles útiles, no al vector conjunto de vóxeles útiles y de vóxles de background. Ver Fig. 4.6

Segmentación en vectores 1D

Este tratamiento tiene como objetivo generar un vector de vóxeles para cada una de las 116 regiones del atlas *AAL*. El procedimiento queda representado en la imagen 4.7. El principal aspecto a comentar es que cada región tendrá un número de vóxeles asociado diferente, encargándose el atlas AAL de seleccionar cuales son los vóxeles pertenecientes a cada región tras la previa selección de los voxels significativos.

Cabe notar como para cada región tendremos un número de voxels distinto. En la tabla 4.2 se observa la amplia diferencia entre regiones en las imágenes MRI.

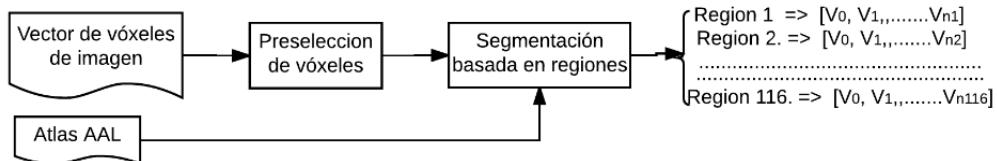


Figura 4.7: Proceso de segmentación de vectores por región

Región	NºVóxeles
1	8272
20	5535
40	2708
60	5191
80	567
100	4260
116	560

Tabla 4.2: Número de vóxeles en imagen MRI por cada región

Segmentación en imágenes 3D

En el caso de la obtención de las regiones cerebrales en imágenes 3D se ha realizado un procesado basado en la posición de los vóxeles de cada una de las regiones.

Para ello, se ha reconstruido el atlas, extrayendo de aquí los límites en cada una de las dimensiones de la posición de la región evaluada.

Posteriormente, se ha usado este conocimiento de la posición exacta de los vóxeles en 3D para llevar a cabo la extracción de la región en cuestión. Este proceso ha sido esquematizado en la imagen 4.8. En la imagen 4.9 se pueden observar dos regiones extraídas y representadas en 3D.

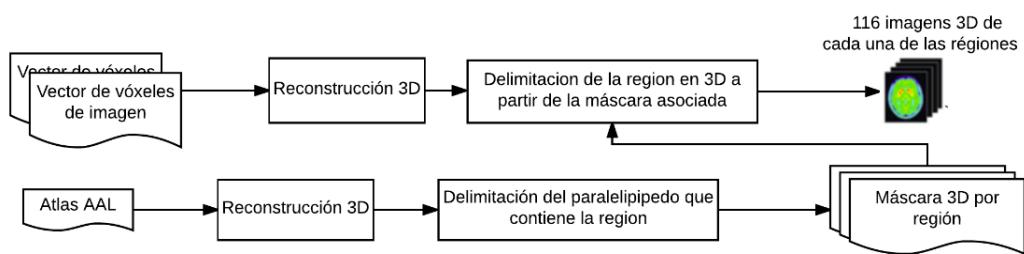


Figura 4.8: Proceso de segmentación de vectores por región

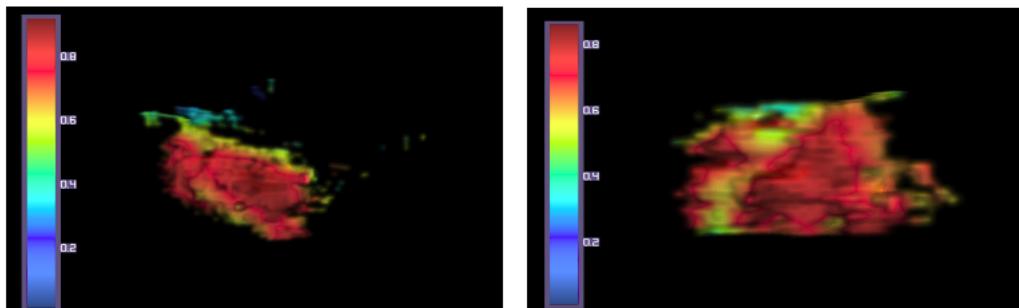


Figura 4.9: Ejemplo de Régiones de imágenes PET segmentadas.(Izquierda) Región N° 20. (Derecha) Región N° 30. Capturas de imágenes 3D tomadas con el programa *MRIcrGL*

Autoencoder Variacional

El principal método de caracterización y aprendizaje en el que se ha basado este trabajo ha sido en el VAE, es por ello que se pretenden exponer los detalles principales de la implementación realizada.

A lo largo de esta sección se explicarán conceptos del VAE que hace referencia al Autoencoder Variacional que emplea redes neuronales de interconexiones densas únicamente, el cual no se debe confundir con el CVAE que emplea redes neuronales convolucionales.

Modelo de grafos

Uno de los aspectos primordiales para la implementación de una red neuronal sobre *Tensorflow* es el concepto de grafo. El modelo computacional para *Tensorflow* es un grafo dirigido, donde los nodos (típicamente representados por círculos o cajas) son funciones de cálculo mientras que las uniones entre nodos (típicamente flechas) son números o matrices. Este modelo es especialmente útil para la implementación de redes neuronales.

Cuando en este trabajo nos referimos a grafo hacemos referencia al diseño de red realizado, concepto que incluye el conjunto de bloques computacionales empleados y a las conexiones entre ellos.

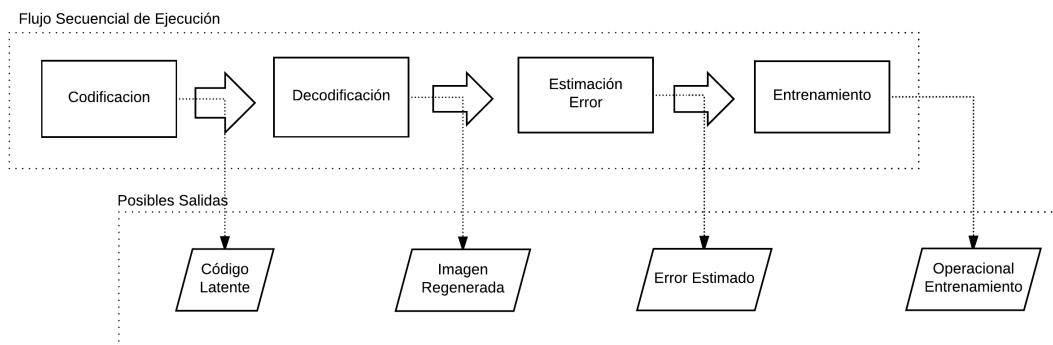


Figura 4.10: Esquema general del grafo elaborado en tensorflow

En la figura 4.10 se puede observar cual es el flujo secuencial de ejecución de los distintos bloques elaborados. Por defecto en *Tensorflow* no se ejecuta todo el grafo en cualquier tipo de ejecución, el tipo de ejecución vendrá determinado según cual sea el tipo de datos solicitado.

Para la ejecución completa del sistema es necesario recorrerlo de la fase de codificación hasta la de entrenamiento. Esto se consigue alimentándolo con los datos de entrada y solicitando el operacional de entrenamiento ¹.

El análisis del grafo elaborado se dividirá en tres secciones, en la primera se explicará la fase de codificación, en la segunda la fase de reconstrucción y en la tercera se expondrán los bloques computaciones que evalúan el error. La cuarta parte del grafo están constituidos por los bloques funcionales encargados de la actualización de los parámetros del enrenamiento, lo que constituye la fase de entrenamiento.

Fase de Codificación

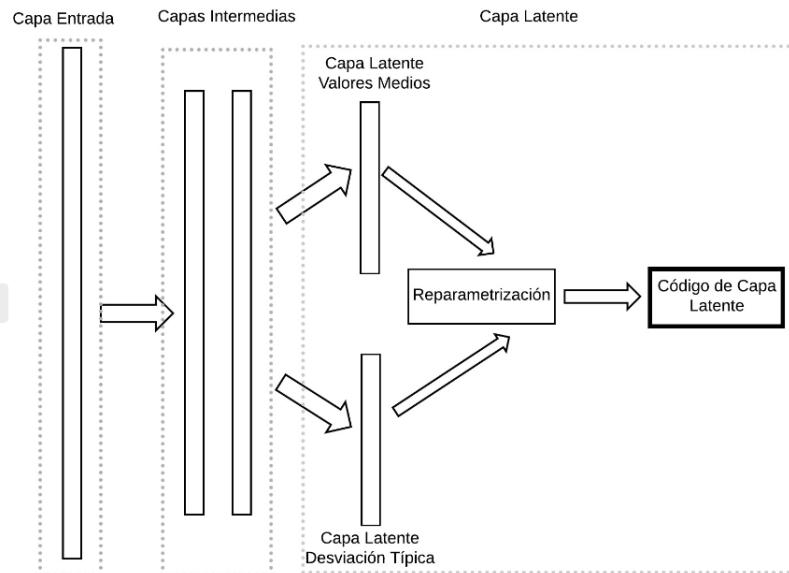


Figura 4.11: Esquema de Fase de Codificación

La fase de codificación del VAE se identifica como conjunto de bloques encargados de servir de entrada para las imágenes hasta que se genera el código en la capa latente. A grandes rasgos el modelo implementado Fig. 4.11.

En primer lugar nos encontramos con la capa de entrada. Esta capa tendrá tantas neuronas como véxoles se vayan a estudiar de la región a evaluar. Cabe recordar que en este trabajo se estudia cada región cerebral por separado por lo cual cada región tendrá asociado un Autoencoder diferente. Esta capa simplemente sirve de entrada,

¹Una ejecución se identifica con el comando *run* de la librería. Sobre este comando se aplicarán un conjunto de datos de entrada y se requerirán otros, siendo los datos requeridos los que determinen hasta qué punto se va a ejecutar el grafo

a priori no ha de aplicarse ningún tratamiento adicional sobre los datos porque estos ya han debido de ser normalizados en el preprocesamiento previo.

Posteriormente nos encontramos con lo que hemos denominado capas intermedias, las cuales se pueden observar en la Fig. 4.11. Estas capas de neuronas son las encargadas de ir reduciendo progresivamente el número de neuronas y por lo tanto el número de características de las regiones evaluadas.

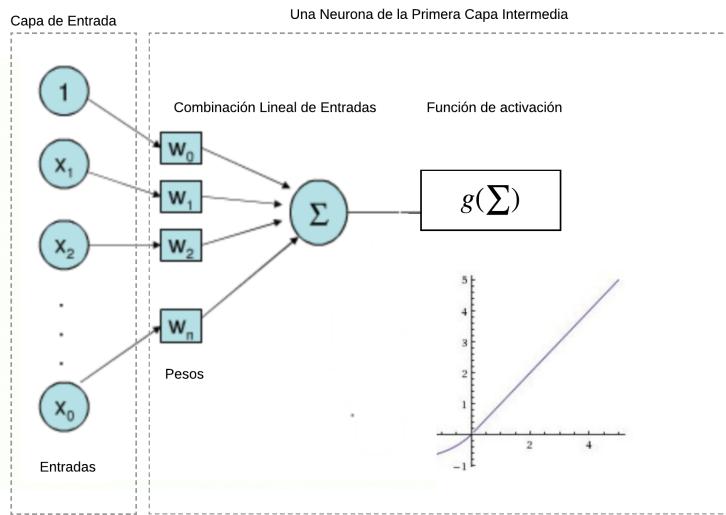


Figura 4.12: Ejemplo de Conexión Básico con la función de activación empleada

Cada una de las neuronas de la primera capa intermedia estarán conectadas a todas las neuronas de la primera capa, esto es el modelo básico de perceptrón que se aplica a todas las conexiones en las redes neuronales densas.

A modo de ejemplo se ha añadido una figura clásica, ver Fig. 4.12, se corresponde con el modelo de perceptrón que simboliza la conexión de una neurona de una capa con todas las neuronas de la capa inmediatamente anterior.

Tras la combinación lineal de las neuronas de entrada, se ha de aplicar una función de activación, en nuestro caso se ha empleado la función denominada ELU (del inglés *Exponential Linear Unit*) que se caracteriza por emplear una función lineal para valores de entrada mayores que 0 y una función exponencial para valores menores que 0. Emplear esta puerta nos garantiza que el gradiente asociado a los distintos parámetros no se haga nulo durante el proceso de propagación atrás, problema que si tendríamos si empleáramos otras puertas como la sigmoide o la función de tangente hiperbólica.

El sistema implementado está preparado para generar tantas capas conectadas entre sí como elementos tenga el vector que define el número de vértices por capas.

En dicho vector se ha de incluir la capa de entrada, que tendrá tantas neuronas con véxeles se evaluan, las capas intermedias y la capa latente.

Por normal general, durante las simulaciones de este trabajo se han empleado entre dos o tres capas intermedias dado que al añadir más capas se aumentan los tiempos de entrenamiento necesarios y además estamos limitados por la capacidad del servidor. Por otro lado, no se ha observado en las simulaciones realizadas que el aumento en el número de capas intermedias aumente de forma considerable los resultados, se entiende que con una o dos capas el Autoencoder es capaz de extraer la información primordial de las regiones segmentadas.

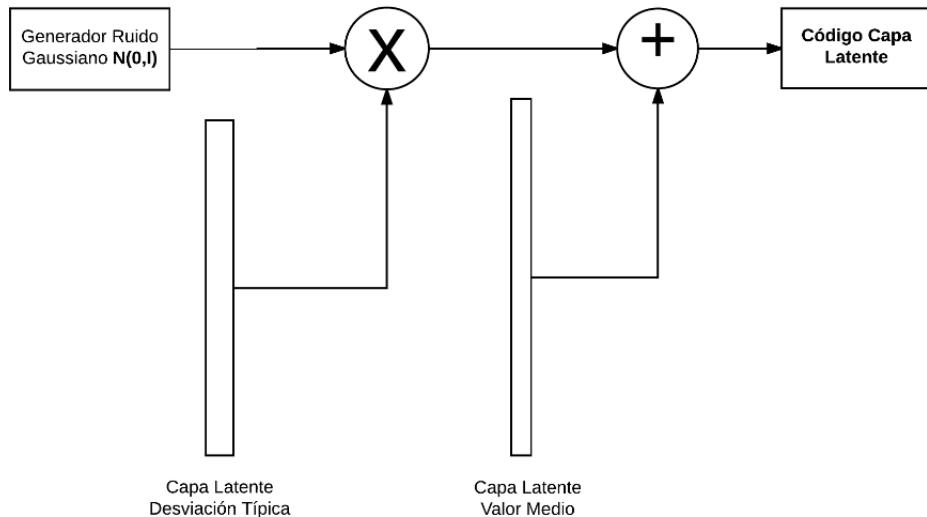


Figura 4.13: Diagrama del Truco de Reparametrización

La última de las capas intermedias estará conectada a un total de dos capas distintas de forma simultánea, esto es, si la última capa intermedia tiene N_i neuronas cada una de estas neuronas estarán conectadas tanto a una capa que denominaremos L_m como a otra L_d . Estas dos últimas capas conforman lo que se denomina capa latente.

El VAE al ser un modelo variacional no genera un valor determinado en la capa latente, en su lugar genera una función distribución que vendrá definida por las capas L_m y L_d . Por ejemplo, la característica con posición i tendrá una distribución gaussiana de media L_{mi} y de desviación típica L_{di} .

A partir de esa distribución y aplicando el truco de reparametrización, el cual se ha representado en la Fig. 4.14, se generará el código latente final L_C . Básicamente, este método consiste en generar un ruido gaussiano ($N(0, I)$), generando un vector

de tantos elementos (N_L) como sea el tamaño de la capa latente. Cada elemento de dicho vector i se multiplicara por su valor asociado de desviación típica L_d y, seguidamente, se le sumará su valor medio L_m .

Cabe notar que el tamaño de la capa latente N_L indica el nivel de codificación que se alcanza, cuanto menor sea mayor será nivel de codificación o compresión se consigue.

Fase de Reconstrucción

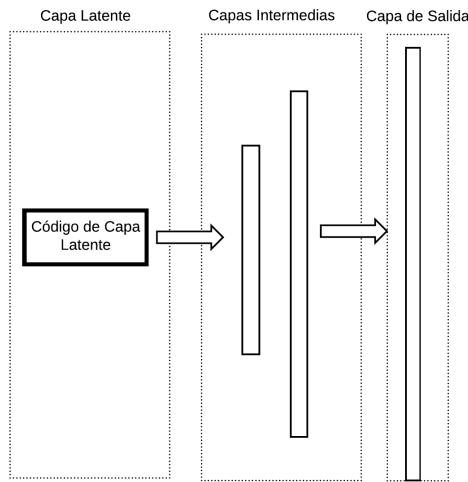


Figura 4.14: Diagrama de la fase de reconstrucción o decodificación del VAE

La fase de reconstrucción contempla el conjunto de bloques funcionales desde que se ha generado el código latente hasta que se consigue regenerar de manera aproximada el conjunto de datos orginales.

Uno de los aspectos esenciales es la simetría en el tamaño de las capas del Autoencoder. Por lo tanto, si en la fase de codificación se tiene, por ejemplo, una capa de entrada de 1500 elementos, dos capas intermedias consecutivas de 1000 y 500 elementos y una latente de 100 elementos, en la capa de reconstrucción se tendrán unas capas intermedias de 500 y 1000 elementos consecutivos, y una capa de salida de 1500 elementos.

En las capas intermedias se seguirá aplicando el mismo tipo de función de activación que en la fase de codificación. Sin embargo, en la capa de salida se espera que cada uno de los elementos de salida tenga un rango delimitado entre $[0, 1]$ (debido a la normalización previa de los datos de entrada), por lo que es necesario aplicar una

función de activación que restrinja la salida a este rango. En nuestro caso se ha usado la función Sigmoide, aunque es posible aplicar cualquier otro tipo de función que sea capaz de restringir la salida a ese rango. En la imagen fig. 4.15 se puede observar la direccionalidad en el rango de salida de las dos funciones comentadas anteriormente

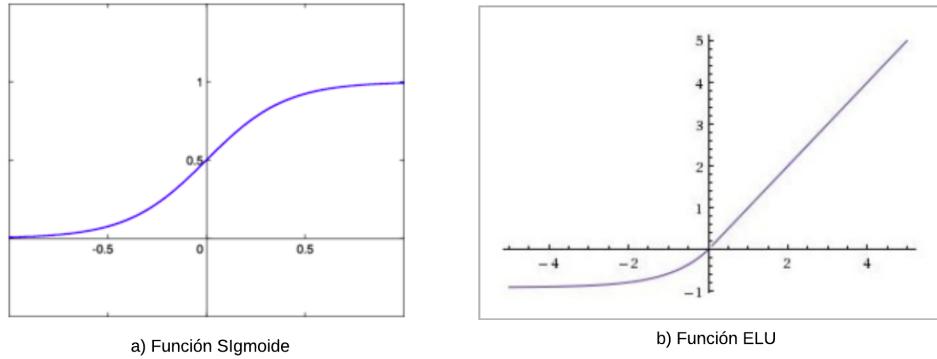


Figura 4.15: Funciones de Activación Empleadas

Evaluación del error

Uno de los elementos principales en todo modelo de aprendizaje profundo es la definición del error. Estableciendo como se calcula el error es posible establecer una función objetivo que nos permita ajustar los parámetros de la red neuronal mediante el procedimiento de propagación hacia atrás.

En el caso del VAE, tal y como se expuso en la sección 3.1.6, el error está compuesto por la diferencia entre los valores de entrada y los valores de salida, y por la divergencia de *Kullback Leibler* entre la función de distribución de los datos de la capa latente y la función de distribución deseada que es la distribución Normal $N(0, I)$.

Otro factor de error que se ha tenido en cuenta es el de regularización de pesos. Uno de los problemas asociados a redes neuronales de varias capas y muchas neuronas por capa ² es la posibilidad de sobre-entrenar el sistema, esto es, provocar que los pesos paramétricos de las conexiones neuronales se ajusten demasiado a las

²Es una definición bastante pobre del concepto, dado que el problema se dará cuando el sistema esté sobredimensionado con respecto al espacio muestral a tratar, por lo que el concepto es realtivo a la dimensionalidad de las muestras tratadas. No obstante lo importante es constatar que el factor de regularización persigue que el sistema sea capaz de generalizar más allá de las muestras de entrenamiento

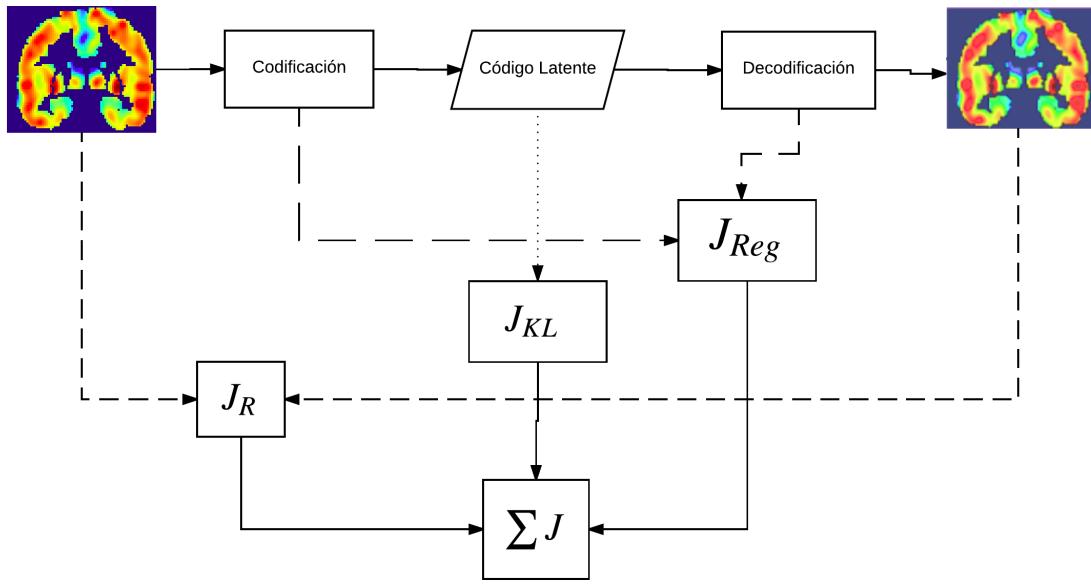


Figura 4.16: Diagrama de los elementos del sistema sobre los que se calcula cada tipo de error

muestras de entrenamiento y, en última instancia, provoque que el sistema no sea capaz de generalizar los resultados a las muestras de test y finalmente a las muestras reales del sistema.

Error de Reconstrucción J_D . Este error se calcula a partir de los valores de entrada y los valores de salida del VAE. Dado que el objetivo del VAE es reconstruir los valores de entrada originales, a mayor divergencia entre entrada y salida, mayor será el error. Para su cálculo se ha empleado la función entropía cruzada, que maximiza el error de manera exponencial.

$$J_D = - \sum_j (x_j * \ln(y_j) + (1 - x_j) * \ln(1 - y_j)) \quad (4.4)$$

Error de Similitud a la Normal en la Capa Latente J_{KL} . Este error viene definido por la divergencia de *Kullback Leiber*. La siguiente expresión representa dicho error de manera general para dos distribuciones p y q .

$$KL(p, q) = \ln\left(\frac{\sigma_q^2}{\sigma_p^2}\right) + \frac{\sigma_p^2 + (\mu_p - \mu_q)^2}{2\sigma_q^2} - \frac{1}{2} \quad (4.5)$$

En nuestro caso la función de distribución p es la función deseada que será la normal $N(0, 1)$, por lo que $\sigma = 1$ y $\mu = 0$. Por lo que la ecuación 4.7 se reduce

$$KL(N, q) = \ln(\sigma_q) + \frac{\mu_q^2}{2\sigma_q^2} - \frac{1}{2} \quad (4.6)$$

Esa será la expresión para cada uno de los elementos de la capa latente, por lo que la expresión final será el sumatorio sobre el factor de divergencia de cada elemento.

$$J_{KL} = \sum_j \left(KL(N, L_j) \right) \quad (4.7)$$

Error de Regularización. Este error vendrá determinado por un factor de configuración que indicará cuanto se penalizará a los pesos paramétricos.

$$J_{Reg} = \sum_\theta \left(\lambda * W_\theta \right) \quad (4.8)$$

donde θ representa el índice de cada uno de los pesos W que conectan las distintas neuronas de la red, mientras que λ es el parámetro que regula la penalización. A un mayor valor de λ , mayor sera la penalización y mayor será la dificultad de que los pesos W alcancen valores muy altos y que por lo tanto este sobre-ajustados.

Entrenamiento

A la hora de abordar como entrenar los parámetros del VAE debemos distinguir entre dos procesos o bloques de operación totalmente diferentes, uno es el diseño del propio grafo mediante *Tensorflow* que se encargue de realizar la actualización de los parámetros (ver Fig. 4.17), mientras que el otro proceso es externo a *Tensorflow* y tiene como objetivo realizar un proceso iterativo que vaya reduciendo progresivamente el error asociado al VAE.

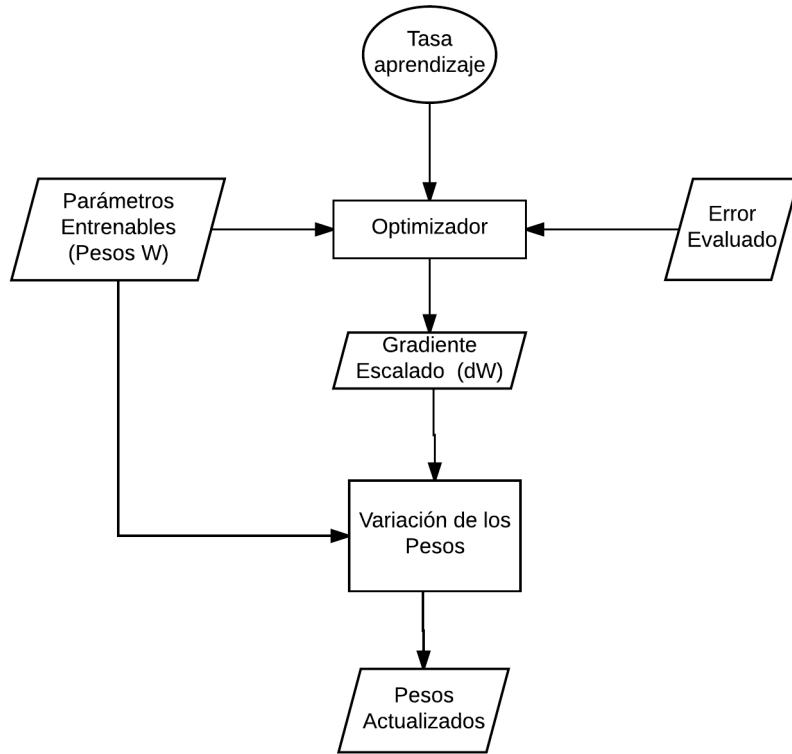


Figura 4.17: Diagrama de boques funcionales del proceso de actualización de los parámetros del VAE

Diseño del Grafo de Entrenamiento

La estructura del grafo de entrenamiento es la que se puede observar en la figura 4.17. La librería *Tensorflow* basa el proceso de actualización de cada uno de los pesos a entrenar (W_θ) en emplear un bloque denominado optimizador.

Este bloque ha de recibir los siguiente elementos:

- **Error evaluado previamente.** Se trata del error general que quedó definido en la sección 4.3.4. Gracias a este error cuantificable se realizará la estimación del gradiente asociado a cada parámetro entrenable mediante el proceso de propagación hacia atrás. Todo ese proceso es interno a *Tensorflow*.
- **Parámetros entrenables W_θ .**
- **Tasa de aprendizaje:** Este elemento es un parámetro de configuración que determinará la rapidez con la que se realizará el proceso de descenso en gradiente. A mayor tasa de aprendizaje mayor variación habrá en cada iteración,

por lo que el descenso será más rápido pero se correrá el riesgo de que el proceso empiece a diverger [75].

En este trabajo se ha empleado el método de actualización denominado *Adam* [76] el cual es un método ampliamente conocido, caracterizado por garantizar un proceso de actualización de parámetros más eficaz en comparación con el proceso de descenso en gradiente escocástico (SGD, del inglés *Stochastic Gradient Descend*).

En este artículo [77] se hace una comparativa entre los distintos métodos de descenso en gradientes. Tras la exposición y la comparación de todos los métodos el autor, finalmente, recomienda el uso del método *Adam*.

Proceso iterativo

Para llevar a cabo el ajuste de los parámetros es necesario llamar de manera iterativa al grafo de *Tensorflow* encargado de llevar a cabo el entrenamiento. Tal y como se ha explicado en la sección anterior, hay un conjunto de bloques funcionales conectados a las salidas del VAE que se encargan de llevar a cabo la actualización de los parámetros. Dichos bloques se nutren del error obtenido a la salida en función de las entradas.

En el diagrama de la Fig. 4.10 está representado el proceso secucencial de ejecución de *Tensorflow*. Es importante notar como para realizar el entrenamiento es necesario llevar a cabo el proceso de codificación, decodificación y la posterior evaluación del error.

Cada sesión de ejecución en *Tensorflow* tiene como origen el dato de entrada con el cuál se alimenta el grafo, y como fin el dato de salida que se le solicita al grafo. En el caso de querer realizar el entrenamiento de los parámetros los datos de entrada será la información que se pretende codificar, en nuestro caso serán los véxeles a evaluar, y como dato de salida se tendrá el operacional de entrenamiento propio de *Tensorflow*. Al solicitar este operacional, se está obligando a la sesión a operar sobre los bloques funcionales del entrenamiento, que son los que se encargan de actualizar los pesos del grafo construido.

En el diagrama de la Fig. 4.18 se puede observar cuál es el flujo del proceso iterativo de entrenamiento. Uno de los aspectos más importantes es la selección de las muestras durante cada iteración, a las muestras seleccionadas se las denomina *batch* de entrenamiento.

Otros aspectos interesantes de la implementación realizada es la posibilidad de almacenar el estado del grafo cada cierto número de iteraciones, así como de mostrar por pantalla el error evaluado en cada iteración. Esto permite observar como va evolucionando el proceso de descenso en gradiente conforme avanzan las iteraciones.

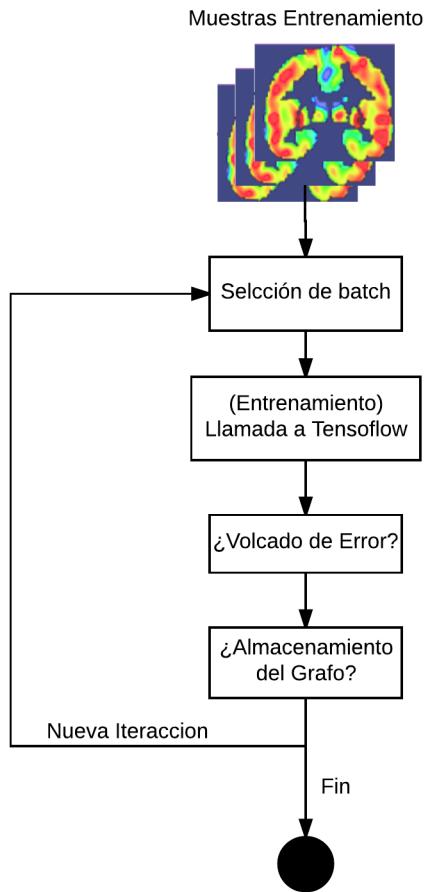


Figura 4.18: Diagrama del proceso iterativo de entrenamiento

Almacenamiento del Grafo

Los procedimientos de entrenamiento en este tipo de sistemas de aprendizaje son altamente costosos tanto en términos de tiempo como de computación, por lo que el almacenamiento de los sistemas ya entrenados se antoja deseable.

La librería empleada (*Tensorflow*) cuenta con funcionalidades que nos permiten almacenar tanto la arquitectura de grafos como los valores de cada uno de los pesos de la red en cuestión.

En la implementación realizada se permite manejar esta funcionalidad de forma cómoda permitiendo almacenar de forma periódica cada cierto número de iteraciones el grafo generado con sus pesos asociados. En fase de ejecución es posible indicar el directorio donde se encuentran almacenados los grafos con lo que es posible volver a

utilizar un modelo previamente caracterizado.

El manejo interno de las distintas variables que conforman el grafo y que permiten la carga de las variables almacenadas es ciertamente complejo, ya que todos aquellos nodos de *Tensorflow* que están conectados al ".exterior" han de ser almacenados en un tipo de manejador de variables que se guarda asociado al grafo y que permite a la librería, durante la carga del modelo, referenciar cada uno de los nodos.

Visualización del Grafo con *Tensorboard*

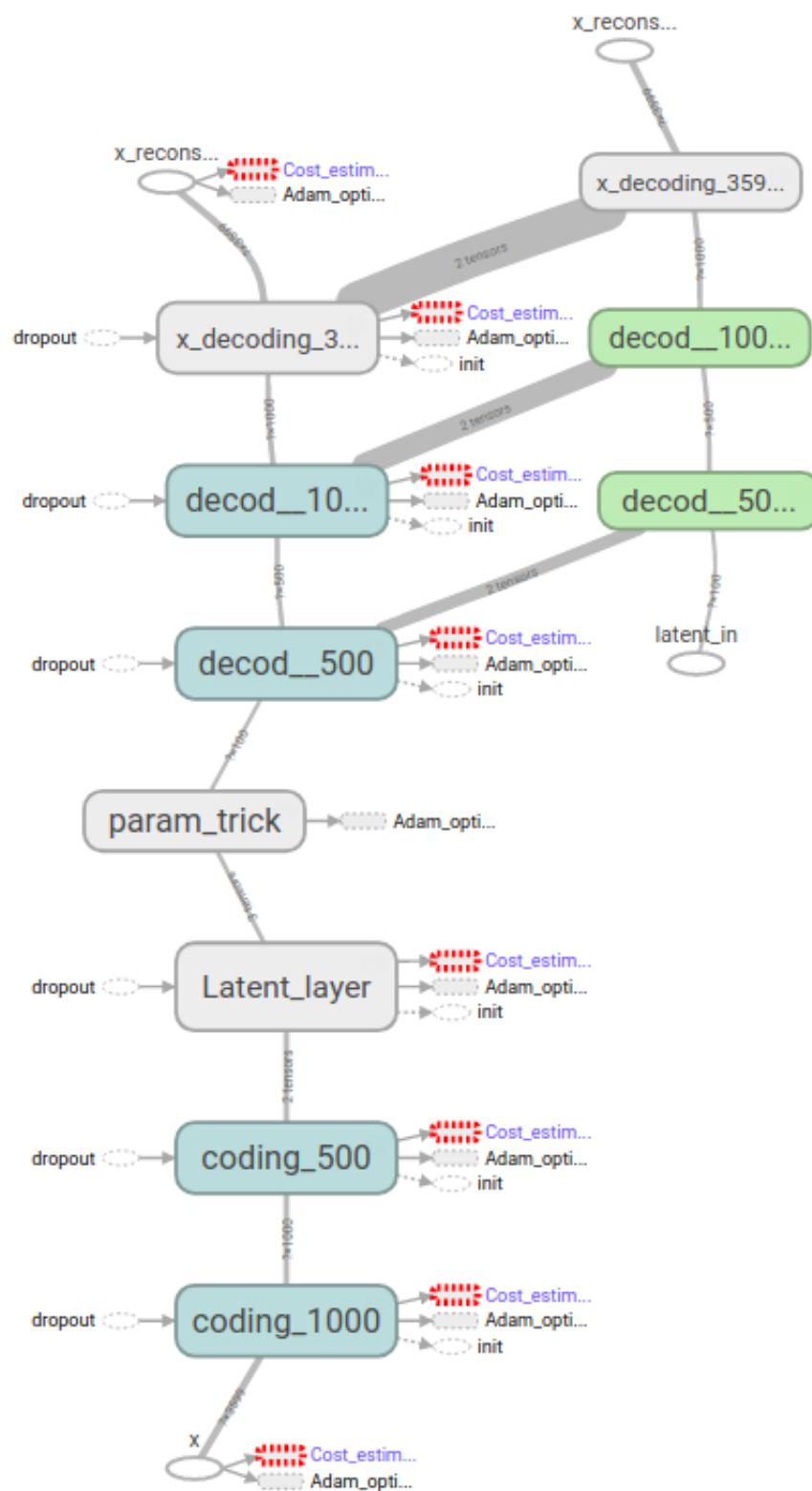
Tensorflow dispone de una herramienta denominada *Tensorboard*, que nos facilita la comprensión, la depuración y la optimización de los sistemas diseñados con esta librería. *Tensorflow* nos permite visualizar el diseño de grafos realizados, pudiéndose observar las conexiones entre los distintos componentes

En un uso más avanzado, esta herramienta permite observar la evolución de los valores en cada uno de los nodos del sistema.

La figura 4.19 se corresponde con la visualización realizada por *Tensorboard* del diseño elaborado. Es interesante notar como se van sucediendo las capas de codificación, en este caso hay únicamente 3 capas, la primera es la capa inicial de entrada donde son introducidos los valores que servirán para alimentar el resto de la red, la siguiente capa es de 1000 neuronas, le sigue otra de 500 y la capa latente final de 100 elementos. Posteriormente se tiene el bloque funcional del truco de reparametrización, el cuál se ve regulado por los valores obtenidos de la capa latente.

Finalmente se tiene la fase de decodificación, el cuál tiene un orden inverso en el tamaño de las capas, esto es, primero la capa de 100, que son los propios valores generados por el truco de reparametrización, luego una capa de 500 neuronas, seguida de una de 1000, y finalmente la capa de salida que tendrá tantos elementos como dimensiones tenga el espacio muestral caracterizado.

La rama que se observa en la zona superior derecha de la imagen se corresponde con la una parte del grafo elaborado con el fin de únicamente realizar la decodificación de elementos de la capa latente, por lo que los datos de entrada no son generados por el bloque funcional del truco de reparameterización sino que son introducidos desde una fuente externa, desde el bloque denominado *latent_in*.

Figura 4.19: Diagrama de Grafos representado por *Tensorboard*

Autoencoder Variacional Convolucional

El segundo modelo propuesto es el Autoencoder Variacional basada en redes neuronales convolucionales. Este tipo de redes neuronales, tal y como se expuso en la sección 3.2.2, son altamente efectivas en problemas de tratamiento de imágenes dada la capacidad que tienen de guardar la relación posicional entre los píxeles o véxoles gracias a la operación de convolución. Sin embargo, son un ámbito del aprendizaje profundo que es más complejo de tratar y que además tiene un coste de computación mayor en comparación con las redes neuronales densas.

En nuestro caso debido a que se está tratando un tipo de imágenes en 3D tenemos la problemática de que la librería empleada (*Tensorflow*) no cuenta con soporte para la realizar la operación de *depooling* o de desagrupamiento en la arquitectura de red. La operación de agrupamiento permite ir reduciendo progresivamente el tamaño de la imagen ya que se realiza una selección de los píxeles más significativos (en algunos casos esta operación es la media de un conjunto de píxeles y en otros se toma el valor del píxel cuyo valor es máximo, ver sección 3.2.2).

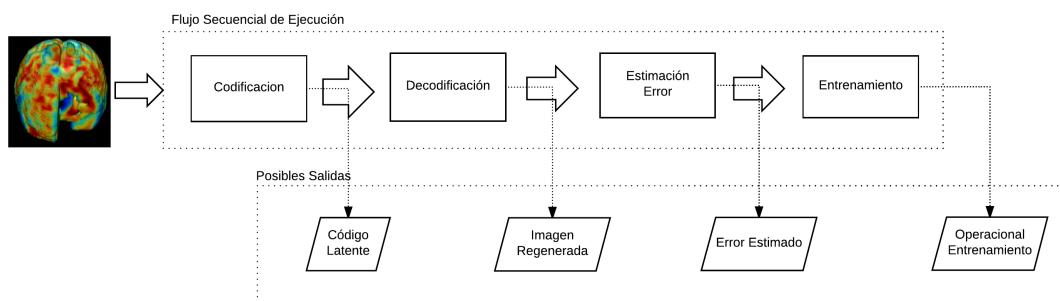


Figura 4.20: Diagrama de la secuencia de bloques funcionales implementados en el CVAE

Por lo tanto la operación de desagrupamiento realiza el proceso inverso el cual es normalmente complicado de realizar, dada la complejidad de determinar un conjunto de véxoles a partir de uno solo.

No obstante, dado que el objetivo del agrupamiento es simplemente ir reduciendo las dimensiones de la imagen a tratar, en la implementación realizada se deja esta funcionalidad para el parámetro denominado desplazamiento o *stride*. Este parámetro es el encargado de regular el desplazamiento entre imágenes durante el proceso de convolución y fué expuesto en la sección 3.2.2. En nuestro caso se ha usado un parámetro de dos por lo que en cada proceso de convolución se reducía a la mitad cada una de las dimensiones de la imagen tratada.

Diseño del Modelo

El diseño funcional del sistema es similar al del VAE dado que cada uno de los bloques realizará una función similar. Será dentro de cada uno de los bloques funcionales que se observan en la figura 4.20 donde se realizarán las modificaciones con respecto al modelo del VAE. Para explorar cada uno de los bloques funcionales implementados se usarán las gráficas proporcionadas por *Tensorboard*.

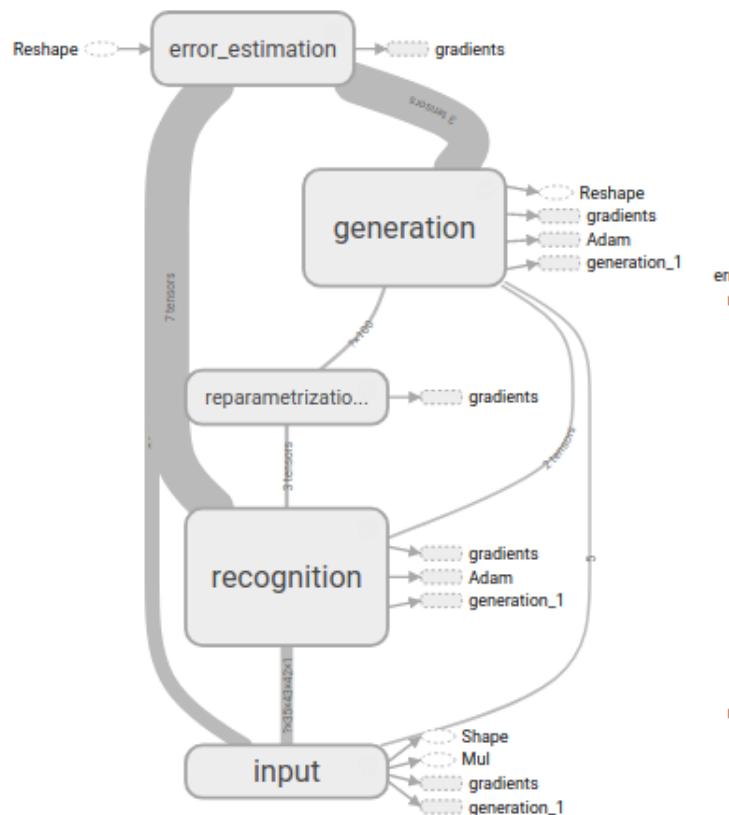


Figura 4.21: Captura de la representación gráfica de *Tensorboard* del modelo convolucional implementado para el CVAE

En la figura 4.21 se tiene una captura de la representación gráfica del modelo de grafos generada automáticamente por **Tensorflow**. Cada uno de los bloques tiene un nombre en inglés dado que todo el código del proyecto ha sido escrito en este idioma. La correspondencia entre estos bloques y los bloques funcionales de la figura 4.20 es la siguiente:

- El bloque denominado ***input***. Se corresponde con el bloque encargado de recibir los datos de entrada. Tiene como objetivo garantizar que los datos de entrada tienen un dimensionado correcto. En la implementación realizada el nodo de entrada recibe los datos vectorizados, siendo internamente el propio grafo el encargado de realizar la transformación a tres dimensiones.
- El bloque denominado ***recognition***. Se corresponde con bloque de codificación. El término angosajon se traduce por reconocimiento, que hace referencia a la capacidad de la capa de codificación de reconocer o identificar el tipo de imagen y en función de esto generar un código de capa latente u otro.
- El bloque denominado ***reparameterization_trick***. Se corresponde con bloque del truco de reparameterización del error
- El bloque denominado ***generation***. Se corresponde con bloque de decodificación. A este bloque también nos podemos referir como bloque de generación si entendemos que a partir de la capa se está regenerando la imagen
- El bloque denominado ***error_estimation***. Se corresponde con bloque funcional de estimación del error.

En las siguientes secciones se explicarán algunos de los detalles de cada uno de los bloques antes mencionados. El bloque de entrada no contará con su propia sección dado que su contenido es simplemente un proceso de ajuste dimensionalidad de un vector a una imagen 3D.

Fase de Codificación

El bloque funcional de esta fase tiene como objetivo transformar la neuroimagen en 3D en los parámetros que regulan las funciones de distribución de las variables del espacio latente. Estos parámetros serán una media y una desviación típica para cada variable latente.

En la figura 4.22 se tiene una captura de la secuencia de los elementos que conforman el bloque. La función de cada elemento es la siguiente:

- El elemento ***first_layer*** es el encargado de realizar la primera función de convolución. Recibe como entrada la imagen en 3D de el bloque de entradas. Además se puede apreciar como tiene salidas conectadas a los bloques encargados del proceso de entrenamiento.

Estas salidas denominadas de entrenamiento son necesarias dado que los pesos de la función de convolución se ajustan durante el entrenamiento y el valor actual de estos pesos es necesario para determinar cual será el tipo de ajuste.

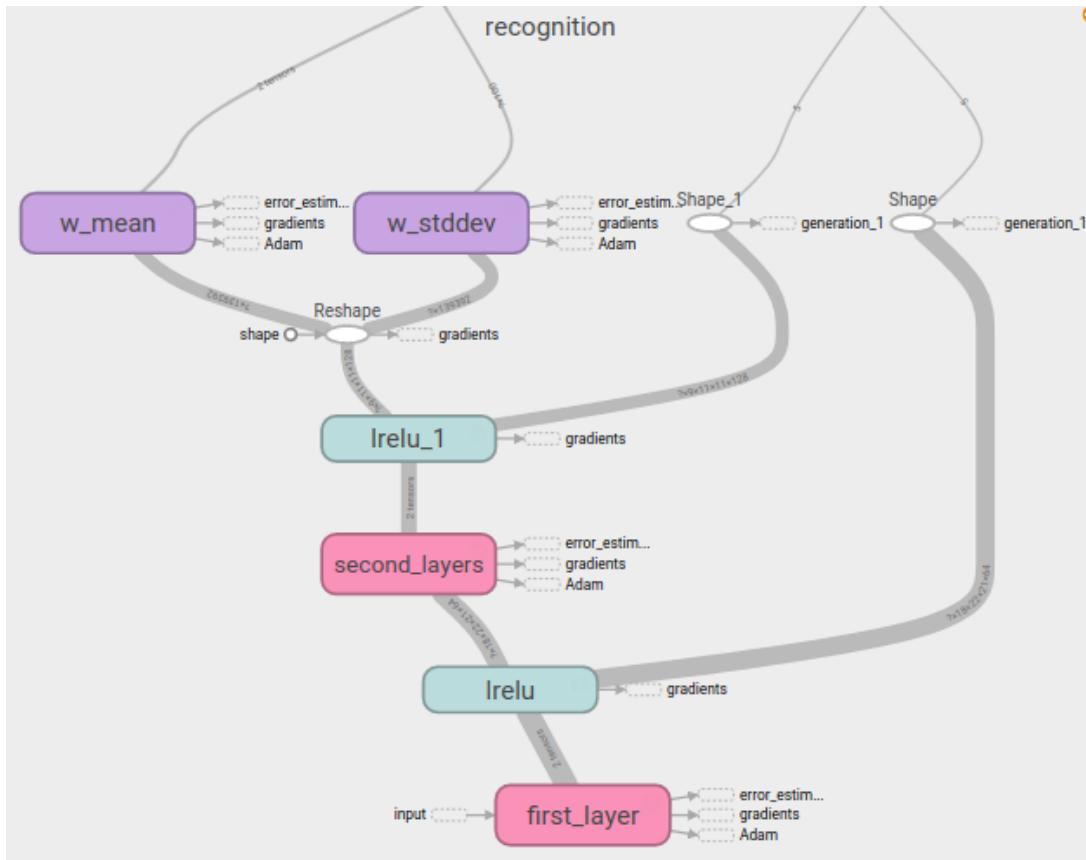
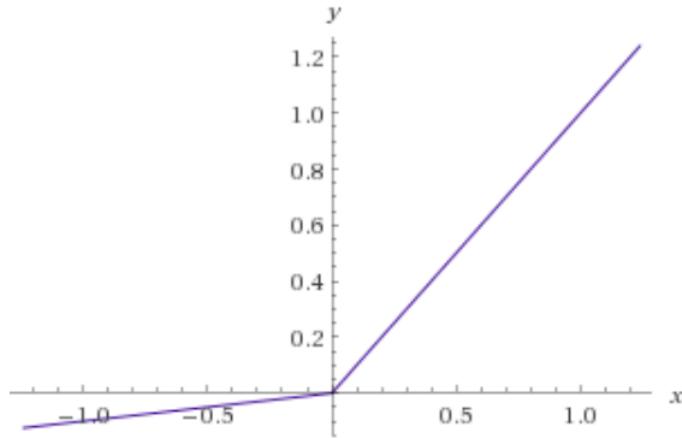


Figura 4.22: Captura de la representación gráfica de *Tensorboard* del bloque funcional de codificación para el CVAE

- El elemento *lrelu* se encarga de aplicar una función de activación sobre las variables a la salida de la función de convolución. El término *lReLU* (del inglés *Leaky Rectified Linear Unit*) se refiere a la función de activación de rectificación lineal con pérdidas para valores negativos. Esta función permite que el gradiente no se desvanezca para valores negativos durante el proceso de descenso en gradiente, lo cual permite que el proceso sea más rápido. En la figura 4.23 se tiene la representación de dicha función de activación.

Este elemento también estará conectado al bloque funcional de entrenamiento, pero en este caso solo afectará el bloque del cálculo de gradientes, dado que esta función de activación no tiene ningún parámetro ajustable.

- El siguiente elemento es otro bloque de convolución denominado *second_layer*, cuya operación es similar al bloque expuesto anteriormente. A la salida de este elemento tenemos de nuevo un bloque de función de activación.

Figura 4.23: Función de activación *lRelu*

- El siguiente bloque, denominado *Reshape* se encarga de realizar la transformación del conjunto de variables dispuestas en un espacio tridimensional a un espacio con una sola dimensión, que es como se presentan las variables latentes.
- Por último tenemos los bloques de salida que en el gráfico vienen denominados por *w_mean* y *w_stddev*. Estos bloques se encargan de realizar sobre el espacio previamente redimensionado la operación básica de red neuronal densa, esta operación fué expuesta en la sección 4.3.2.

Como salida se tendrán dos vectores de variables, uno representa la media de la función de distribución normal de las variables latentes (salida de *W_mean*), mientras que el otro representa la desviación típica (salida de *W_stddev*)

Truco de Reparametrización

Este bloque funcional es el encargado de generar los valores finales de la capa latente a partir de las funciones normales previamente caracterizadas durante la fase de codificación. El gráfico de este bloque funcional se puede observar en la figura 4.24. Es importante notar que lo que este bloque funcional implementa es la expresión de la ecuación 3.12. Los elementos que nos encontramos en este bloque son los siguientes:

- El elemento de entrada es *shape* que sirve para indicar el número de variables de la capa latente.
- El elemento *random_normal* se encarga de generar ruido aleatorio gaussiano.

- El elemento *mul* multiplica el valor de error por la media por el valor de media de la función de distribución de cada variable latente.
- El elemento *add* suma al valor obtenido anteriormente la desviación típica de cada variable latente.

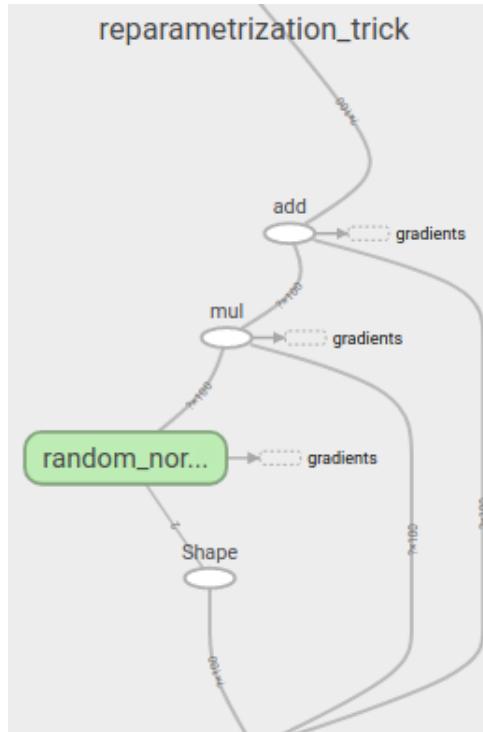


Figura 4.24: Captura de la representación gráfica de *Tensorboard* de la implementación del truco de reparametrización para el CVAE

Fase de Decodificación

La fase de decodificación contiene el conjunto de bloques funcionales encargados de llevar a cabo la regeneración de los datos partir del código latente. En la figura 4.25 se tiene una captura de la representación de dichos bloques realizada por la herramienta *Tensorflow*. Dichos bloques son los siguientes:

- Elemento *z_matriz*. Contiene el vector de código latente a decodificar. Se ha denominado matriz dado que se procesan varias muestras simultáneamente.
- Sobre el elemento anterior se aplica el bloque *reshape* para pasar de una matriz de datos a un espacio de n muestras siendo cada muestra una imagen.

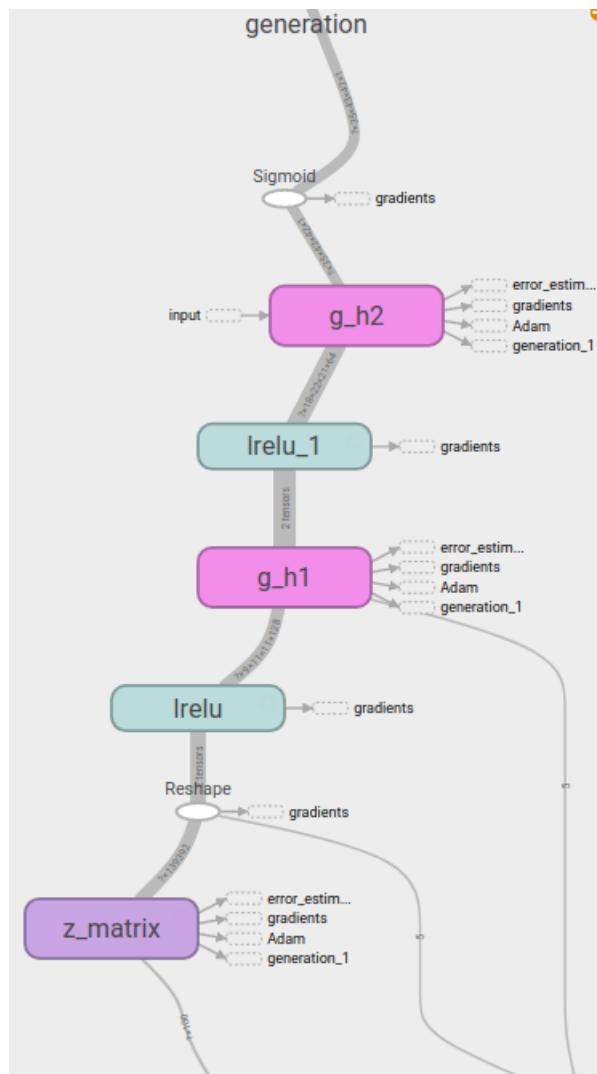


Figura 4.25: Captura de la representación gráfica de *Tensorboard* del bloque funcional de decodificación para el CVAE

- El resto de bloques son los componentes en cascada puestos en orden inverso con respecto al proceso codificación.
- Finalmente, el bloque con función sigmoide que permite limitar el rango de los valores de salida entre 0 y 1.

Fase de Estimación del error

Esta fase se encarga de calcular el error asociado a la reconstrucción de un conjunto de muestras. El resultado determinará el comportamiento del proceso de descenso en gradiente. En la figura 4.26 se puede ver como hay tres bloques funcionales bien diferenciados cada uno de los cuales encargados los tres tipos de errores definidos en los modelos de VAE implementados.

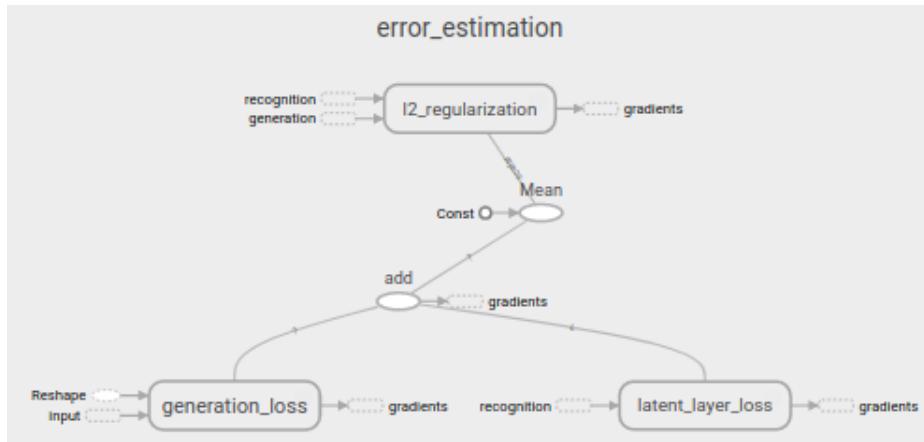


Figura 4.26: Captura de la representación de *Tensorflow* del bloque funcional de la estimación del error para el CVAE

Modelo de Clasificación

El diseño y aplicación de un modelo de clasificación es de especial interés dado que la capacidad de clasificación está intimamente relacionada con el método de extracción características empleado, que en nuestro caso será el autoencoder variacional.

Tal y como ya se ha explicado previamente, el autoencoder es capaz generar un conjunto de características representativas en lo que se denomina espacio latente, siendo estas variables obtenidas sobre las que posteriormente se les aplicará el proceso de clasificación. Estas variables latentes serán generadas por región, por lo que al final tendremos tantos vectores de variables latentes como número de regiones se estén evaluando.

Es importante notar que el procesamiento de las regiones por separado implica la aplicación de un procedimiento de clasificación en el cuál debemos de unir la información de evaluación de cada región. En la figura 4.27 se puede apreciar lo mencionado anteriormente.

A lo largo de esta sección se detallará el método explicando el procedimiento realizado y mencionando algunas de las problemáticas encontradas como son la extracción de características por región o el mezclado de información para los dos tipo de imágenes MRI.

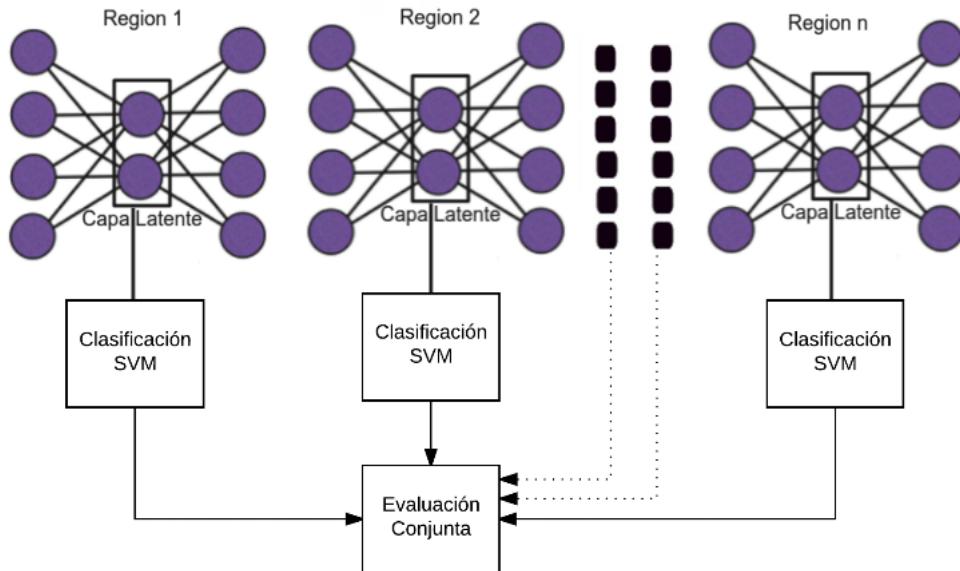


Figura 4.27: Esquema básico de Autoencoder. Cabe notar como será el código generado en la capa latente lo que se empleará para la clasificación posterior

Extracción de Características por Región

El proceso de extracción de características aplica el Autoencoder Variacional. Una esquematización de dicho proceso se puede observar en la imagen 4.28. Es de esperar que el aumento del tamaño de la capa latente, esto es, que haya más neuronas en dicha capa conlleve una mejora en los resultados de la clasificación dado que se ha comprimido menos la información de las imágenes.

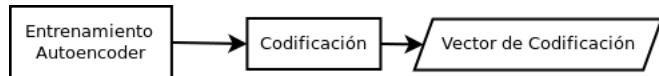


Figura 4.28: Diagrama básico del proceso de extracción de características aplicando el Autoencoder Variacional

Se pueden diferenciar dos fases, la de entrenamiento y la de extracción de características, mientras que el tercer bloque de la figura 4.28 hace referencia al vector de variables esperadas a la salida.

- **Entrenamiento del Autoencoder.** El entrenamiento tiene como objetivo caracterizar el Autoencoder en función del tipo de muestra, en nuestro caso en función de la tipología de neuroimagen de cada una de las regiones evaluadas. Es este entrenamiento el proceso mas costoso, tanto computacionalmente como en términos de coste en el desarrollo de este modelo de clasificación.

Este proceso permite la extracción efectiva de características ya que es el que ajusta el conjunto de parámetros encargados de generar el conjunto de variables latentes en función del tipo de imagen de entrada.

- **Codificación.** Dado el conjunto de véxoles de cada neuroimagen asociado a un tipo de región, este proceso se encargará de generar el conjunto de variables latentes.

En esta fase es posible aplicar tanto el VAE de capas densas como el CVAE dado que lo importante es garantizar que la salida será un vector de variables sin importar el procedimiento de extracción empleado. No obstante, el tipo de VAE empleado modificará el procesado previo de las imágenes dado que si se emplea el VAE de redes densa se deberán vectorizar las imágenes mientras que si es el CVAE se deberá delimitar en 3D las distintas regiones.

Un aspecto diferencial en el tratamiento de las imágenes PET y MRI es que para las MRI tenemos dos modalidades de imágenes, las de materia blanca y las de materia gris, es por ello que para las imágenes MRI necesitaremos dos *Autoencoders* lo cual duplica el coste computacional. En la figura 4.29 queda representado este concepto

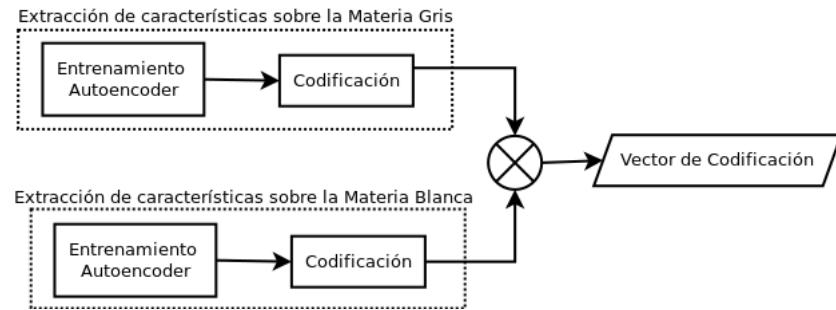


Figura 4.29: Diagrama básico del proceso de extracción de características para las imágenes MRI

Por lo tanto, para el caso de las imágenes MRI se tendrá a la salida un vector de características que será la concatenación de las variables latentes obtenidas a partir de las imágenes de materia blanca y materia gris.

Evaluación por Región

Una vez que se ha generado de cada región un vector de características se procederá a la construcción de un clasificador basado en dicho vector. Dichas características se corresponden con la versión codificada de una región siendo de un espacio $n - dimensional$ en función del tamaño de la capa latente del VAE empleado.

Es importante notar que para cada región se tendrá un vector de características distinto, dado que el VAE aplicado sobre cada región es diferente, y por lo tanto los clasificadores serán totalmente independientes.

En este fase se han de diferenciar dos procesos, los cuáles han sido esquematizados en la figura 4.30

- **Entrenamiento:** Dados los vectores de codificación de las muestras de entrenamiento X_E y de sus etiquetas asociadas Y_E se aplicará el proceso de optimización de parámetros en el SVM con objeto de entrenar el clasificador.

Test: Dado un clasificador SVM con los parámetros previamente entrenados, se aplicarán tanto los vectores de regiones codificadas de entrenamiento X_E como las de test X_T .

A la salida del SVM se obtendrá la distancia ($d_i \in R$) del vector de caraterísticas ($x_i \in R^n$) al hiperplano de separación SVM. El umbral de decisión en la clasificación SVM es 0, por lo tanto las muestras con una distancia asociada mayor que 0 serán de un tipo y del otro tipo en el caso de que sean menor que 0. En nuesro caso se corresponderán con sujetos AD y NOR respectivamente.

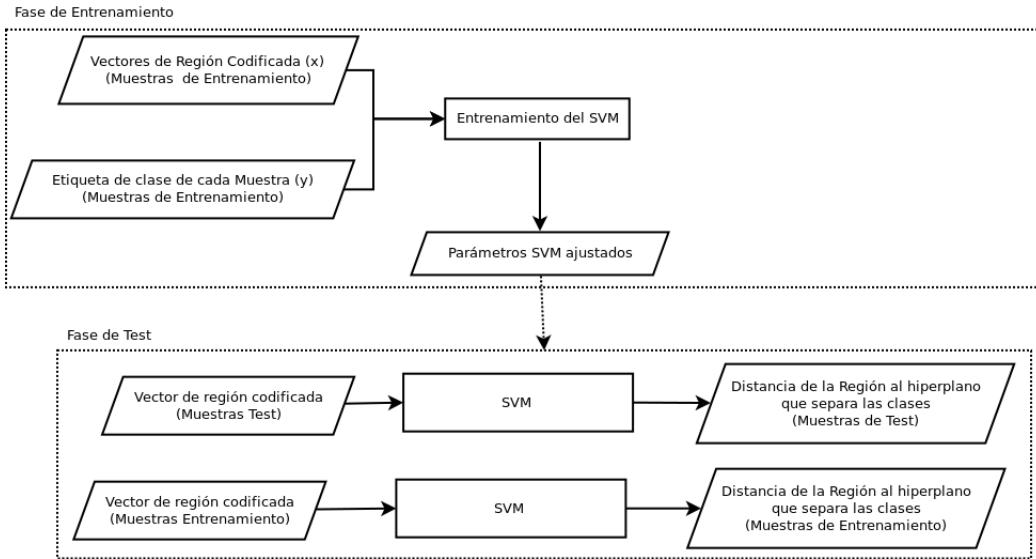


Figura 4.30: Proceso de evaluación por región basado en máquinas de vectores de soporte

Evaluación Conjunta

La fase final del proceso de clasificación tiene como objetivo determinar el tipo de paciente (NOR o AD) en función de la información generada previamente por cada una de las regiones evaluadas. En el esquema de la figura 4.31 se puede observar como encaja este proceso final dentro del proceso general de clasificación.

Para cada Sujeto evaluado ($D_j \in R^R$) se dispondrá de un vector de parámetros, donde cada uno de los parámetros será relativo a cada una de las regiones y por lo tanto el tamaño del vector D_j se corresponderá con el número de regiones evaluadas R . Estos parámetros se corresponden a la distancia de cada región al hiperplano de separación del SVM empleado previamente, se puede expresar tal que:

$$D_j = (d_{j0}, d_{j1} \dots d_{ji} \dots d_{jR}) \quad (4.9)$$

Para llevar a cabo esta evaluación se han utilizado tres métodos diferentes e independientes entre sí, esto es, como entrada toman el vector de Distancias D_j para cada una de las muestras y generan una puntuación de clasificación, pero en ningún momento hay iteracción entre los métodos propuestos, simplemente son diferentes maneras de evaluar el resultado final.

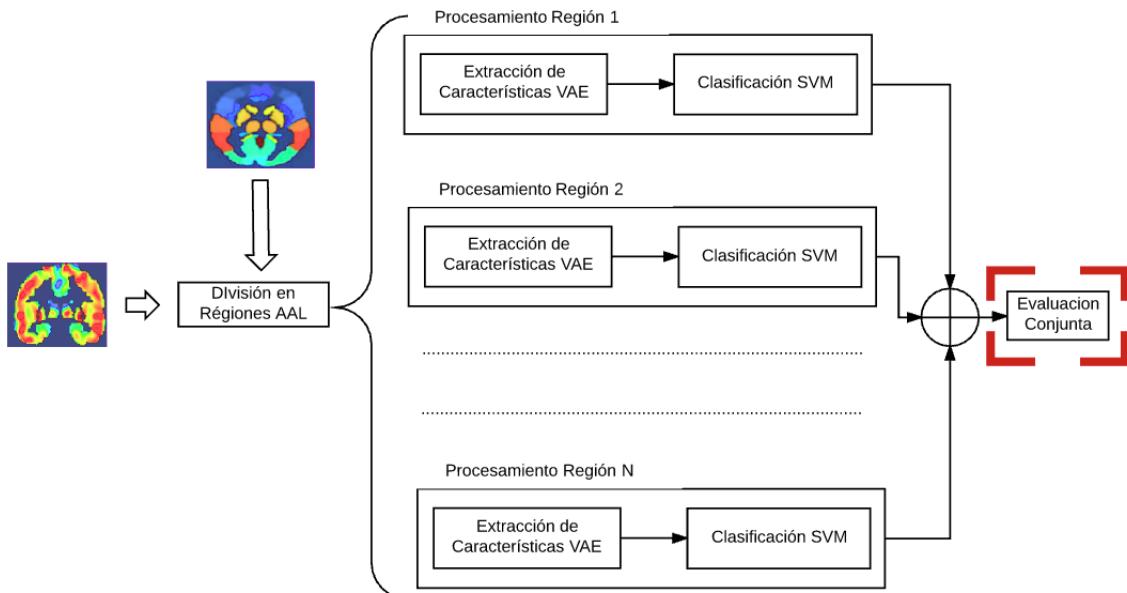


Figura 4.31: Diagrama del proceso de clasificación basado en Régiones. El proceso señalado en rojo se corresponde con el proceso final encargado de la evaluación conjunta

Voto por Mayoría Simple

En este método en función de las distancias d_{ji} del vector D_j se asigna la pertenencia a un grupo u a otro si solo tuvieramos en cuenta la región i referenciada por la distancia d_{ji} en cuestión.

$$s_{ij} = \begin{cases} 1 & \text{si } D_{ij} > 0 \\ 0 & \text{si } D_{ij} < 0 \end{cases} \quad (4.10)$$

Por lo tanto a la salida de este proceso se tendrá un vector de unos y ceros de tamaño R al que denominaremos S_i , y a cada elemento de dicho vector s_{ji} . Sobre este vector se aplicará un sumatorio y se dividirá por el número de regiones evaluadas R , tomando el umbral (th) para determinar en función del resultado de la operación anterior la pertenencia del sujeto evaluado a una clase u a otra.

$$\bar{S}_i = \frac{\sum_{j=1}^{R} s_{ji}}{R} \rightarrow \begin{cases} \bar{S}_i > th \rightarrow AD \\ \bar{S}_i < th \rightarrow NOR \end{cases} \quad (4.11)$$

El valor del umbral th por defecto será 0.5, aunque se ha posibilitado determinarlo a partir de las muestras de entrenamiento, extrayendose a partir del punto óptimo de corte en la curva ROC para dichas muestras de entrenamiento. No obstante, los resultados han sido mejores sin utilizar esta funcionalidad dado que en cierto modo se está sobre-ajustando el sistema al usar las muestras de entrenamiento para ajustar dicho parámetro.

El principal problema asociado a este método es la problemática de aproximar distancias a valores de voto (realizado en la expresión 4.13) que de por sí implica la perdida de información, dado que se toma que una distancia de -10 sea igual a una de -1 lo cual no es lo más recomendable.

Voto por Mayoría Complejo

En este otro método se ha tomado directamente el vector de distancias D_j y se ha aplicado un sumatorio sobre cada uno de los elementos de dicho vector. Gracias a esto se evita la aproximación para cada una de las distancias por región.

$$\bar{D}_i = \sum_j^{0 < j < R} d_{ji} \quad (4.12)$$

Para determinar a que clase pertenece cada sujeto se aplicará un umbral que por defecto será 0, aunque tal y como se ha hecho en el otro modelo se ha habilitado la posibilidad de determinarlo a partir del punto óptimo de clasificación para las muestras de entrenamiento.

$$\begin{cases} \bar{D}_i > th \rightarrow AD \\ \bar{D}_i < th \rightarrow NOR \end{cases} \quad (4.13)$$

No obstante, el hecho de sumar todas las distancias tampoco garantiza que la aproximación sea la más óptima, debido a que cada una de estas distancias está referida a un hiperplano diferente, esto es, cada distancia ha sido obtenida a partir de la proyección de un conjunto de valores de entrada (que es el vector de características de cada region) sobre la función de decisión del SVM en cuestión, donde cada SVM es distinto por región.

Por lo tanto no es lo más preciso, de hecho, el proceso de sumar las distancias implica la suma de elementos calculados a partir de espacios vectoriales no iguales y por lo tanto no es correcto.

Sin embargo, más que obtener un valor de suma preciso, lo que este método pretende es dar un valor cuantitativo a la pertenencia a un grupo u a otro por región, en lugar de realizar la bruta aproximación de unos o ceros del voto por mayoría simple.

Máquina de Vectores de Soporte

Por último se ha querido realizar un método que evaluará cada elemento del vector de muestras D_j de forma ponderada, determinando posteriormente el grupo al que pertenece en función de dicha suma ponderada.

En nuestro caso se ha empleado un SVM, dado que lo que este método realiza es similar aplicando las ventajas los vectores de soportes y además es el método que garantiza mejores resultados para tales propósitos. Otro factor que ha provocado que nos decantemos por el SVM es la facilidad de uso gracias a la librería *scikit-learn* de *Python*.

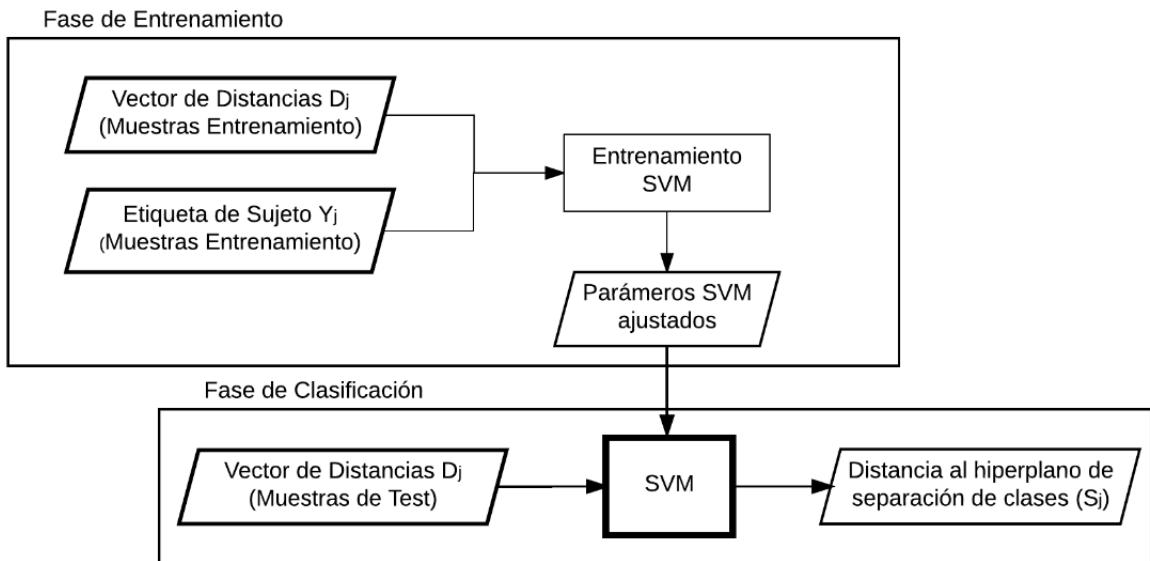


Figura 4.32: Proceso de Evaluación Conjunta Alicando SVM

Para este proceso contaremos con dos fases bien diferenciadas. Una de **entrenamiento** empleada para ajustar los parámetros del SVM donde únicamente se utilizarán las muestras de entrenamiento.

La otra fase será de la **clasificación** donde para cada vector de distancias D_j se generará un valor indicativo de la distancia al hiperplano generado por el SVM que separa las dos clases evaluadas. A dicho valor se le denominará S_j .

Dados los valores S_j se considerará que la muestra j pertenece a una clase u a otra en función de la siguiente lógica:

$$\begin{cases} S_j > th \rightarrow AD \\ S_j < th \rightarrow NOR \end{cases} \quad (4.14)$$

De nuevo tal y como se ha mencionado en los método, el valor por defecto del umbral th será 0, aunque será posible fijarlo a partir de las muestras de entrenamiento.

Capítulo 5

Pruebas y Resultados

Evolución de la reconstrucción de una region

Una manera interesante de evaluar que los sistemas diseñados funcionan correctamente es comprobar que las neuroimágenes 3D generadas se asemejan a las originales. Durante el proceso de caracterización del sistema, esto es, el proceso de entrenamiento de los distintos pesos de la red es posible comprobar como las imágenes generadas se asemejan cada vez más a la original, según se va desarrollando el procedimiento y van aumentado el número de iteracciones de entrenamiento realizadas.

Ejemplo con VAE

En esta simulación, se ha empleado el diseño del autoencoder variacional con la configuración expuesta en la tabla 5.1. Una captura de la imagen en 3D para distintas iteracciones se puede observar en la figura 5.1.

Se aprecia claramente como para las iteracciones iniciales (100 repeticiones) apenas se aprecia la estructura del elemento final y conforme se va desarrollando el proceso de entrenamiento esta estructura se va aproximando cada vez más al aspecto de la región original.

Tipo de Imagen	PET
Muestra	1
Región	3
Número de Capas	3
Neuronas por Capa	[3896, 1000, 500]
Tasa de aprendizaje	0.000005
Stride	2

Función de Activación	elu
Tamaño del batch de muestras	64
Dropout	0.90
Regularización Lambda L2	0.0001

Tabla 5.1: Tabla con la configuración de los distintos parámetros para la simulación de la figura 5.1

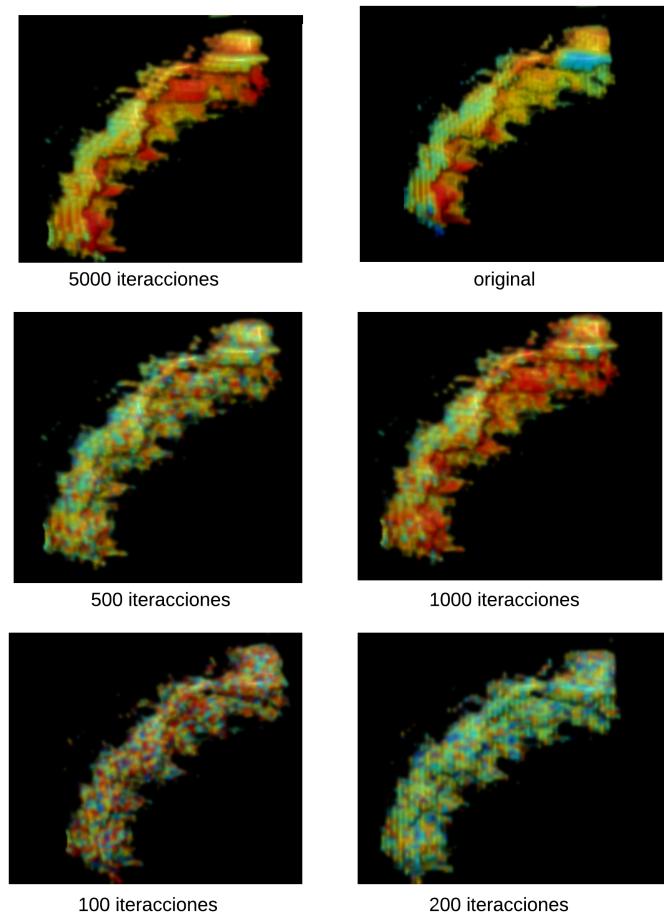


Figura 5.1: Evolución de la regeneración de la imagen 3D de la region n°3 del atlas AAL en función de la iteracción en la que se encuentre en el proceso de entrenamiento para un VAE.

Ejemplo con CVAE

Tipo de Imagen	PET
Muestra	1
Región	3
Número de Capas	3
Tamaño de Imagen	[14, 43, 42]
Tasa de aprendizaje	0.0001
Tamaño del Kernel	5
Tamaño Capa Latente	50
Stride	2
Función de Activación	lrelu
Tamaño del batch de muestras	50
Tasa de decaimiento	0
Regularización Lambda L2	0.0001

Tabla 5.2: Tabla con la configuración de los distintos parámetros para la simulación de la figura 5.2

En este caso se ha empleado el diseño del autoencoder variacional convolucional con la configuración expuesta en la tabla 5.1. Una captura de la imagen en 3D para distintos momentos del proceso de entrenamiento puede observarse en la figura 5.1.

Observando dicha imagen se puede apreciar como según se va desarrollando el proceso de entrenamiento la imagen resultante se asemeja cada vez más a la imagen original. No obstante si comparamos la imagen de la figura 5.1, del VAE con respecto a la imagen 5.1 se puede apreciar como el proceso de entrenamiento es más lento en el caso del autoencoder convolucional, ya que se necesitan más iteracciones para conseguir una imagen que se asemeje a la original.

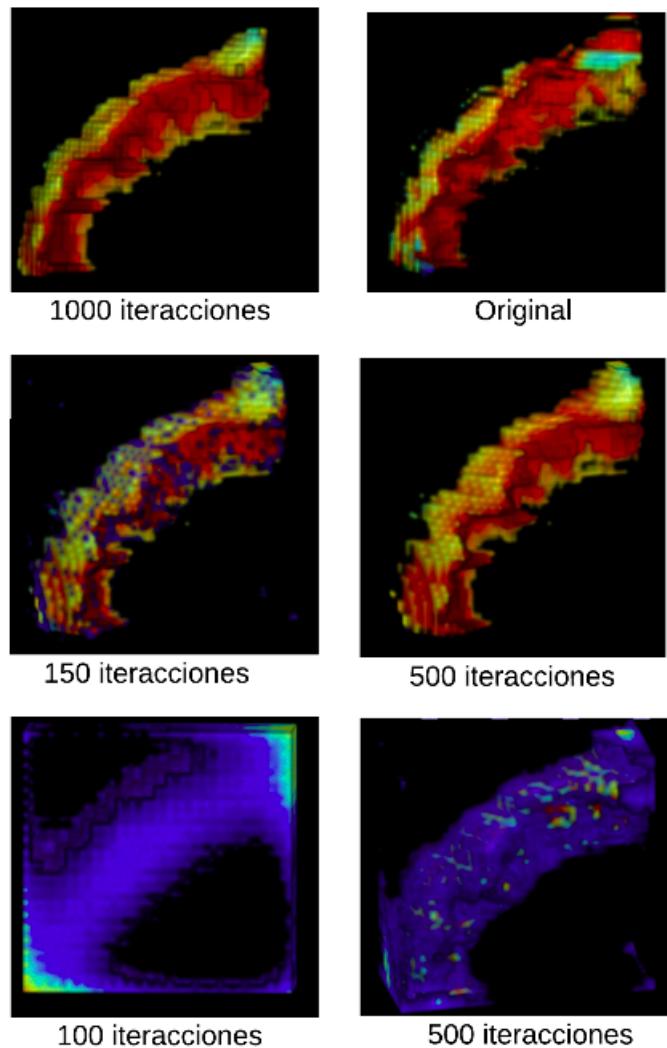


Figura 5.2: Evolución de la regeneración de la imagen 3D de la region n°3 del atlas AAL en función de la iteracción en la que se encuentre en el proceso de entrenamiento para un CVAE.

Reconstrucción completa del cerebro

En este simulación se prentede reconstruir un cerebro completo a partir de cada una de las muestras. Para ello, es necesario realizar un conjunto de procedimientos cuyo flujo de realización ha sido esquematizado en la figura 5.3.

Dado que el sistema diseñado es capaz de caracterizar las regiones, no el cerebro completo ya que sería tremadamente costoso computacionalmente, se deberá realizar un procedimiento basado en la síntesis de las regiones cerebrales, para su posterior reconstrucción o reubicación de cada una de estas regiones en el cerebro.

Es necesario disponer de un VAE o CVAE, dependiendo del diseño empleado, para cada una de las regiones previamente entrenado. En el esquema de la figura 5.3 se ha incluido un bloque con este paso aunque por lo general el entrenamiento del VAE solo ha de realizarse una vez ya que es posible almacenar dicho modelo ya caracterizado.

Dadao un modelo de VAE caracterizado para cada región se procederá con la codificación de cada región cerebral de la muestra en cuestión. Esto generará un valor de media y otro de desviación típica para el código latente asociado a cada región. Se seleccionará el valor de la media como entrada al bloque decodificador del VAE, lo cual producirá a la salida una reconstrucción de la región.

Finalmente cuando se tenga una reconstrucción de cada una de las regiones, se realizará un procedimiento para encajar los vóxles de cada region en la imagen del cerebro final.

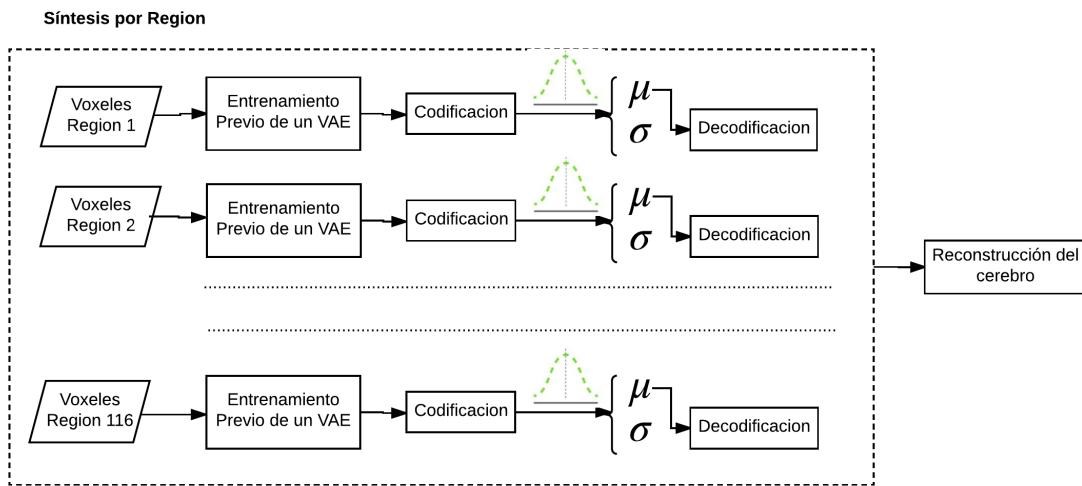


Figura 5.3: Diagrama de la secuencia de pasos necesarios para llevar a cabo la síntesis completa del cerebro

Ejemplo con CVAE

En este caso se va a realizar la reconstrucción completa del cerebro sobre el autoencoder variacional convolucional. Se mostrará la reconstrucción de una neuroimagen de un sujeto AD y de un sujeto NOR.

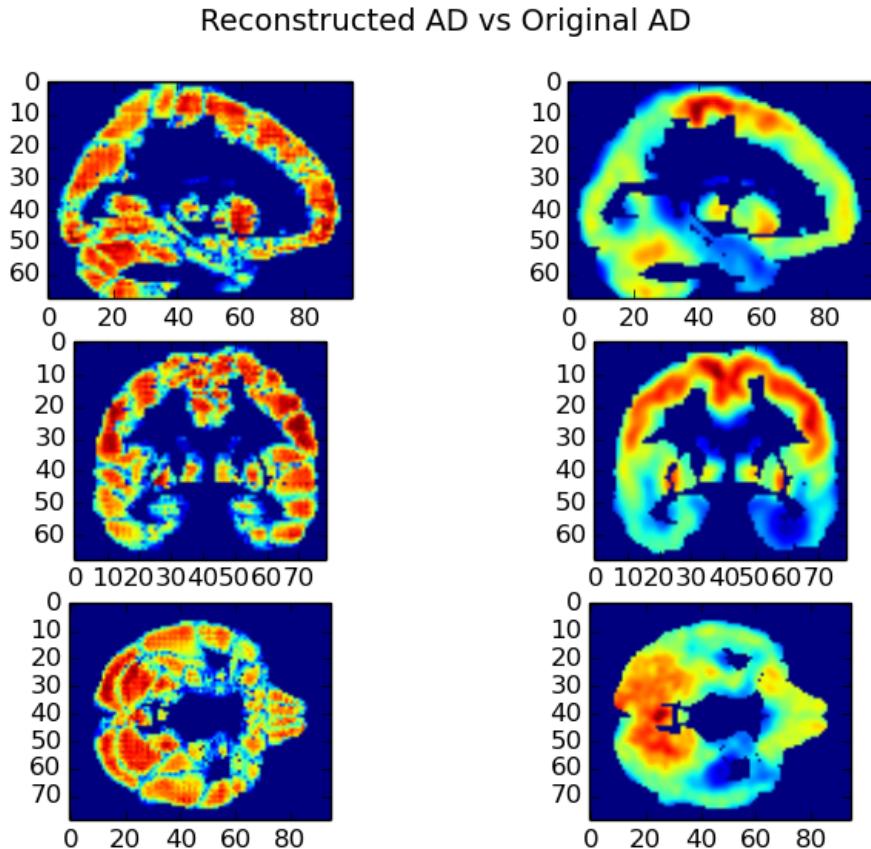


Figura 5.4: Secciones sagital, horizontal y transversal de la neuroimagen reconstruida a la izquierda y de la neuroimagen original a la derecha. Se trata de una neuroimagen de un paciente AD y la reconstrucción ha sido realizada sobre un CVAE

En la figura 5.4 se tiene el resultado de la reconstrucción para un paciente mientras que en la figura 5.5 se tiene la reconstrucción para uno NOR.

Se tiene un resultado nefasto ya que la reconstrucción en ambas imágenes no se asemejan a las imágenes originales. Además la imagen reconstruida es similar en ambos casos, lo cual indica que el proceso de decodificación de alguna manera provoca que el código latente genere una misma imagen final.

En la sección 5.4 y en la sección final de los resultados de clasificación se puede verificar que basándonos en el análisis del código latente generado a partir de las muestras ambos tipos de sujetos (AD y NOR), lo cual hace indicar que el proceso de codificación es capaz de generar un código distinto para cada tipo de sujeto.

Por lo tanto el hecho de que la imagen reconstruida en esta simulación sea muy similar para ambos sujetos indica que el problema esta en la fase de decodificación.

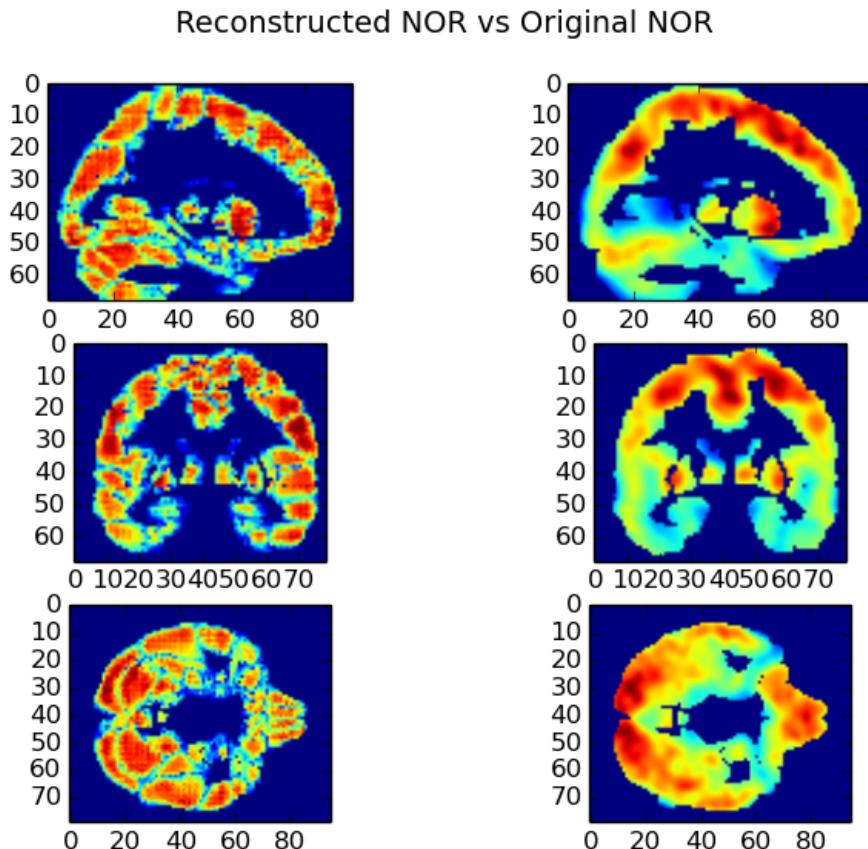


Figura 5.5: Secciones sagital, horizontal y transversal de la neuroimagen reconstruida a la izquierda y de la neuroimagen original a la derecha. Se trata de una neuroimagen de un paciente AD y la reconstrucción ha sido realizada sobre un CVAE

Ejemplo con VAE

En esta segunda prueba se ha empleado un modelo de VAE y se ha realizado la simulación tanto en la neuroimagen de un sujeto AD como en uno NOR.

En la figura 5.6 se tiene el resultado de la reconstrucción para un paciente mientras que en la figura 5.7 se tiene la reconstrucción para uno NOR.

Se tiene un resultado similar que para el CVAE dado que a pesar de que las imágenes originales de los sujetos AD y NOR son claramente diferentes, se tiene que la imagen reconstruida para ambos caso es muy similar.

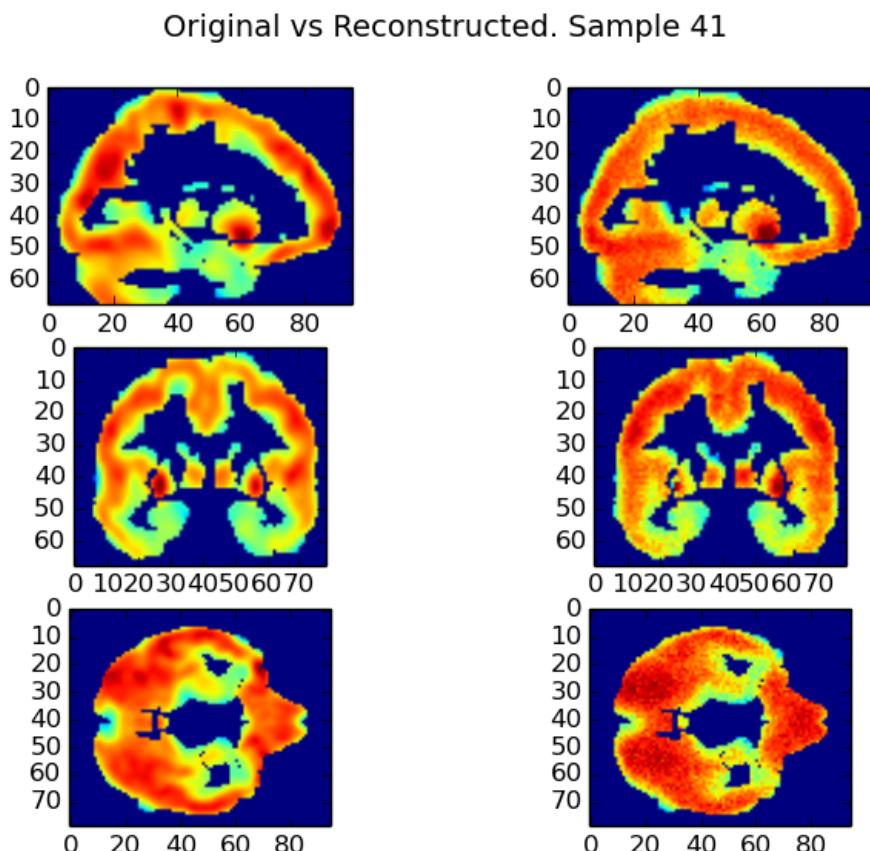


Figura 5.6: Secciones sagital, horizontal y transversal de la neuroimagen reconstruida a la izquierda y de la neuroimagen original a la derecha. Se trata de una neuroimagen de un paciente NOR y la reconstrucción ha sido realizada sobre un VAE.

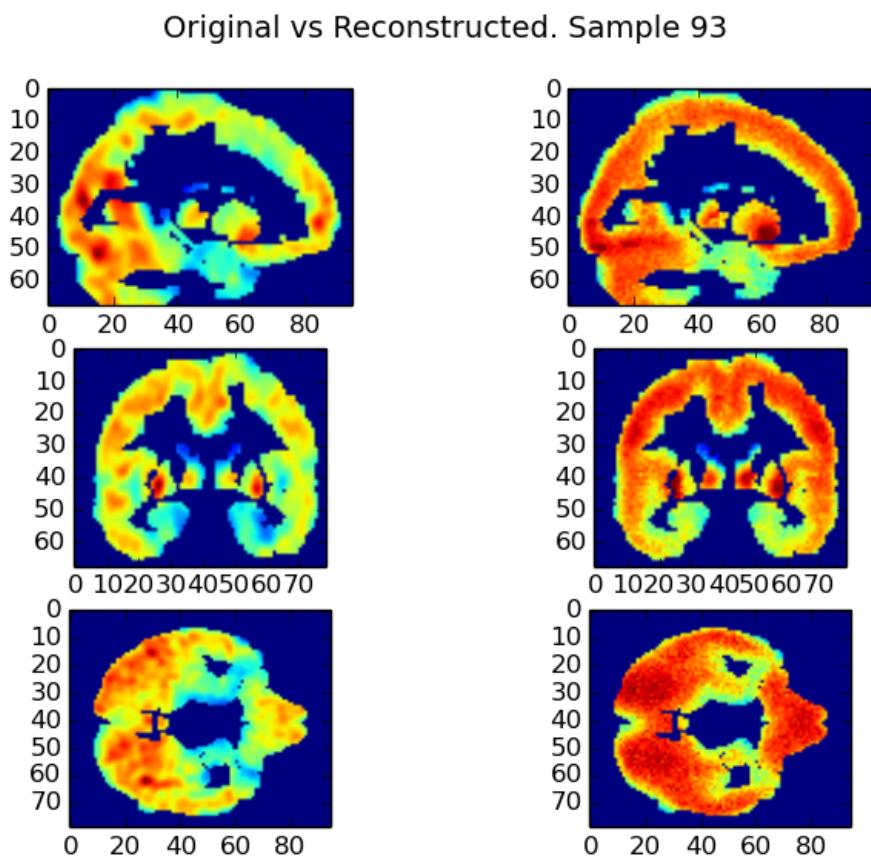


Figura 5.7: Secciones sagital, horizontal y transversal de la neuroimagen reconstruida a la izquierda y de la neuroimagen original a la derecha. Se trata de una neuroimagen de un paciente AD y la reconstrucción ha sido realizada sobre un VAE.

Visualización del código de la capa latente

En esta sección se pretende comprobar como en el código latente generado para cada tipo de region es posible diferenciar entre neuroimágenes de sujetos NOR y AD. Se ha aplicado un modelo de VAE con una capa latente de 100 neuronas.

Con este objetivo, se ha aplicado el proceso de codificación sobre el espacio muestral de las imágenes PET, obteniéndose un vector de código latente para cada región. Cada vector tendrá una dimensionalidad igual al tamaño de la capa latente.

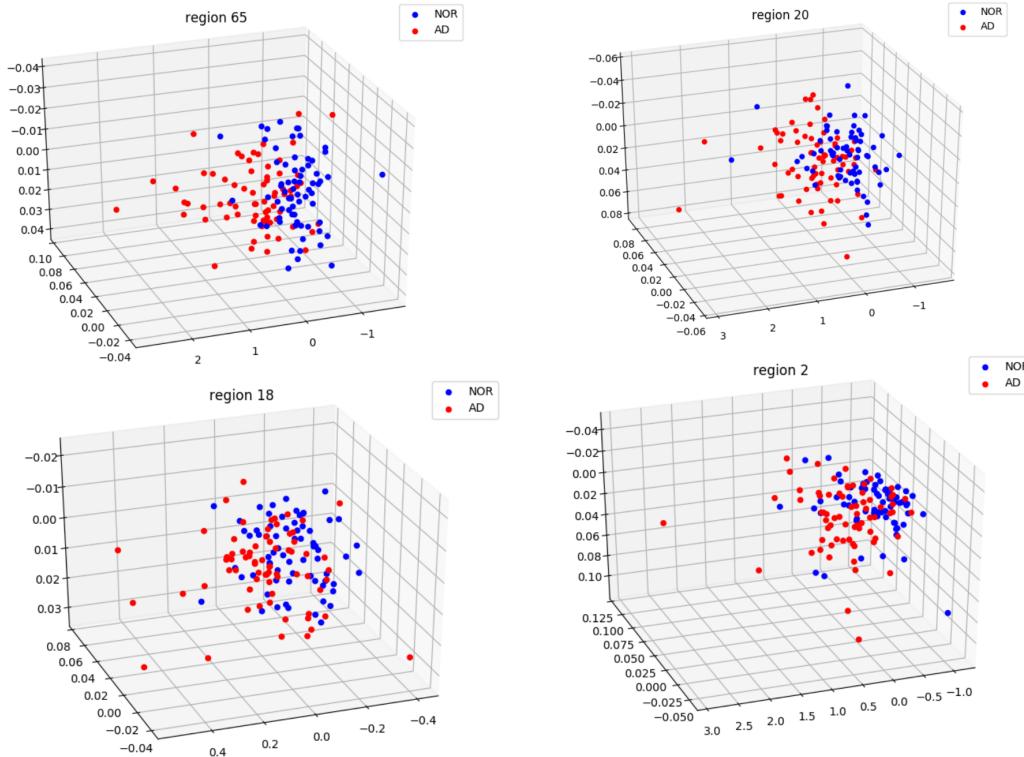


Figura 5.8: Representación en tres dimensiones de la dispersión del código latente generado por un modeo VAE. Para la obtención de los valores de tres variables para el código de cada región se ha aplicado PCA sobre el código latente.

Sobre el espacio codificado se ha aplicado un algoritmo denominado PCA (Análisis de Componentes Principales) que nos permite reducir la dimensionalidad. Este algoritmo se basa en aplicar una transformación del espacio vectorial en el que están expresados los valores de las muestras. Técnicamente, el PCA busca la proyección

según la cual los datos queden mejor representados en términos de mínimos cuadrados.

En la figura 5.8 se tiene el resultado de esta simulación donde se puede observar a simple vista como el código de los sujetos NOR es separable del de los sujetos AD. Esto nos indica que el modelo de VAE empleado es capaz de diferencia entre los dos tipos de imagen y, por lo tanto, es de esperar que la capacidad de clasificación del sistema sea alta.

Es importante notar, que al aplicar PCA para reducir un código de dimensionalidad 100 a 3 se pierde parte de la información. Es probable que esta simplificación provoque que sea más difícil separar visualmente el grupo de pacientes NOR de los AD en la imagen mostrada, dado que en cierta medida esta pérdida de información implica la homogenización de los datos.

Exploración del código de la capa latente

En la simulación expuesta en esta sección se pretende explorar y comprobar como la variación del código latente generado a partir de una imagen tiene su influencia en la imagen final reconstruida.

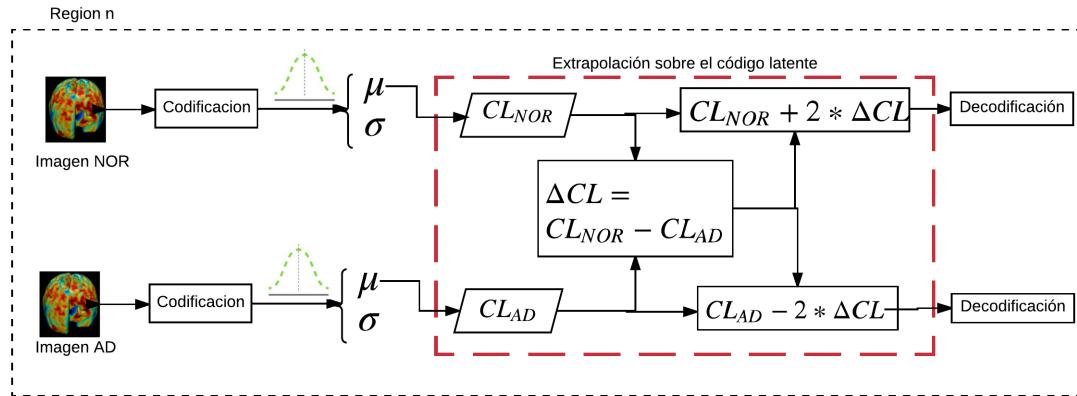


Figura 5.9: Diagrama del procedimiento de extrapolación sobre el código latente para cada una de las regiones. En el cuadro rojo se indica donde se realiza dicho proceso.

La idea general del proceso llevado a cabo es generar, para cada región, el código latente asociado a dos muestras representativas tanto del tipo de imágenes de sujetos AD como a los sujetos NOR y desplazar este código latente en la dirección contraria a la posición del código de la muestra del otro tipo. Esta dirección viene determinada por la resta entre el código latente generado para la imagen del sujeto NOR menos el código de la imagen del sujeto AD. Esto nos da una referencia vectorial de la distancia entre los dos códigos latentes y será esta distancia la que se usará para aumentar aún más la distancia entre ambas muestras.

En la figura 5.10 se tiene una figura donde se pueden observar las neur imágenes cerebrales empleadas. Es fácil comprobar como el paciente AD tiene una actividad cerebral mucho menor en comparación al paciente NOR.

En la figura de la imagen 5.9 se tiene representado un diagrama sobre el proceso que se ha realizado en esta simulación. Es importante notar que el proceso representado se deberá de aplicar para cada una de las 116 regiones del cerebro. Obviamente una vez que se haya realizado la reconstrucción de cada regón a partir del código latente modificado, será necesario realizar la ubicación de los vértices de todas las regiones para conseguir reconstruir totalmente el cerebro.

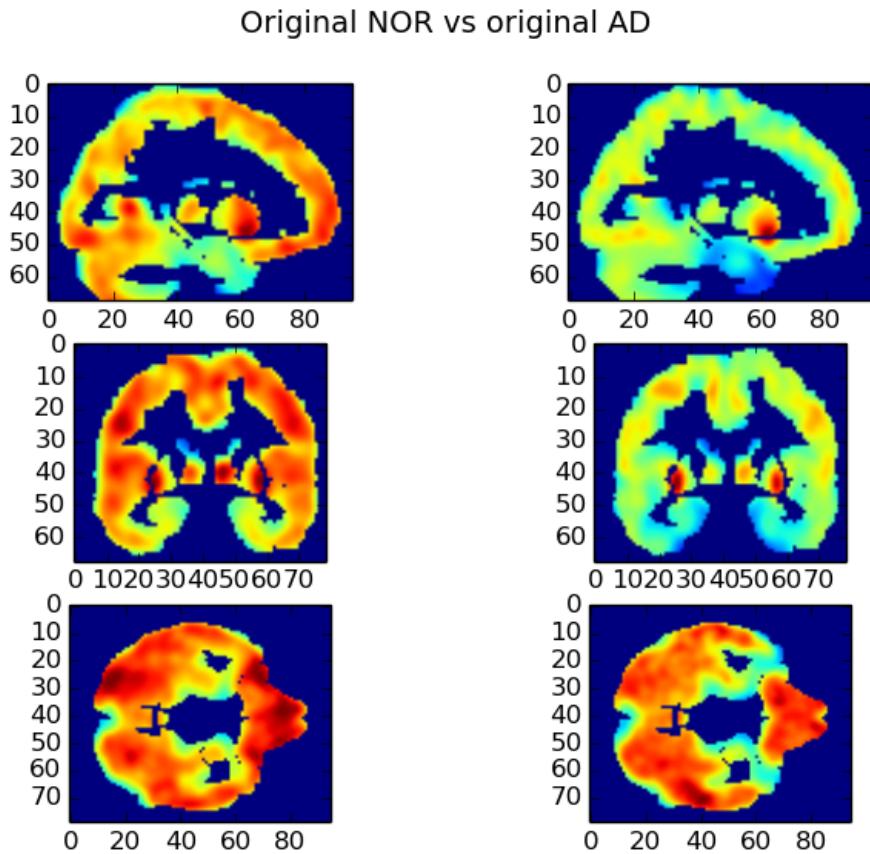


Figura 5.10: Secciones sagital, transversal y horizontal de dos neroimagenes originales, a la izquierda la de un sujeto NOR y la derecha un sujeto AD. La reconstrucción ha sido realizada sobre un VAE.

En la figura 5.11 se tiene el resultado de esta simulación. A continuación se realizará un comentario para cada una de las secciones representadas

- Sección sagital. Se aprecia en la parte izquierda-alta no tiene activación en el sujeto NOR mientras que en el AD está muy activada. Lo contrario ocurre con en la parte alta situada a la derecha donde la activación se da en el sujeto AD. Si comparamos con la imagen original, ver fig 5.10, se observa como la parte derecha-alta se activa más en el sujeto NOR, lo cual puede ser el origen del efecto antes comentado.
- Sección transversal. En la imagen original hay mayor activación de manera global en el sujeto NOR que en el AD. Sin embargo, no se aprecian grandes diferencias en las imágenes reconstruidas.

- Sección Horizontal. Ocurre algo similar a lo ocurrido con la sección sagital. La zona alta-izquierda en la imagen del sujeto NOR tiene mucha activación, mientras que en el sujeto AD apenas tiene activación. No obstante en este caso no se aprecia una gran diferencia de activación en la zona alta-derecha para el sujeto NOR con respecto al AD.

Por lo tanto, la dirección del cambio del código latente debido a la extrapolación ha provocado estos cambios, o dicho de otro modo, las variables latentes modificadas son las encargadas de regular la activación las zonas de las neuroimágenes en las cuales se observan las mayores diferencias entre los sujetos AD y los NOR.

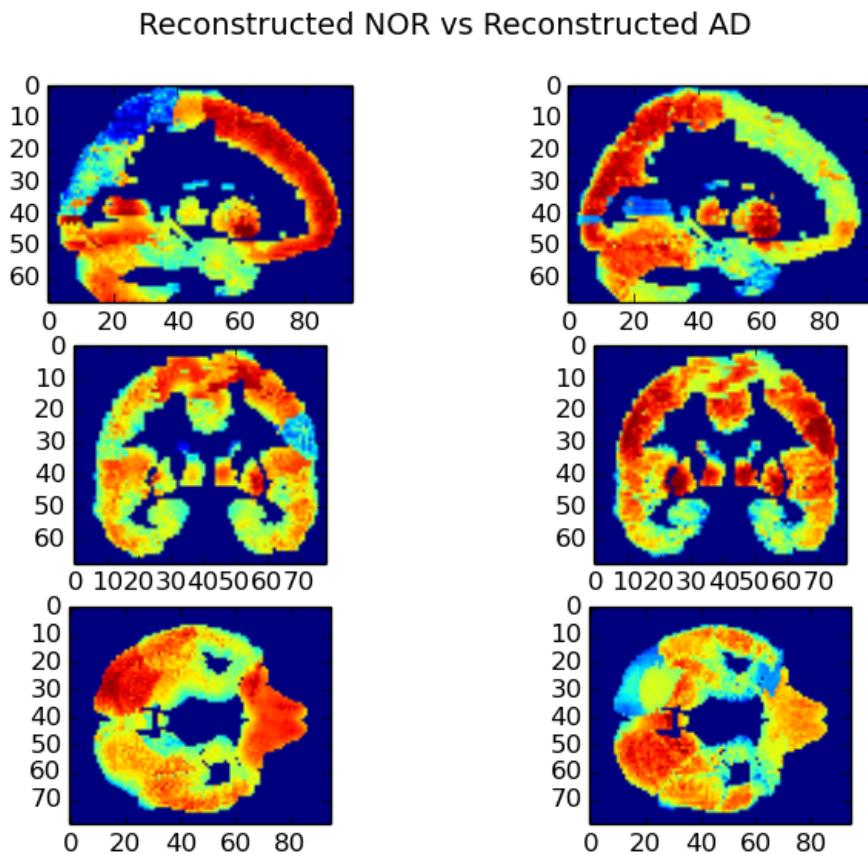


Figura 5.11: Secciones sagital, transversal y horizontal de dos neuroimágenes reconstruidas con un modelo VAE, a la izquierda la de un sujeto NOR y la derecha un sujeto AD.

Clasificación

El conjunto de pruebas de esta sección tiene como objetivo evaluar la capacidad del código latente generado por el VAE para discernir entre los tipos de sujetos AD y NOR. Para ello, se realizará un proceso de clasificación que utilizará como datos de entrada el código latente generado por el autoencoder tal y como quedó expuesto en la sección 4.5.

Se han realizado un total de cuatro simulaciones diferentes empleando cada uno de los dos modelos de autoencoder diseñados tanto VAE como CVAE con los dos espacios muestrales disponibles, estos son las imágenes MRI y las imágenes PET.

Las métricas empleadas para evaluar la efectividad y calidad de la clasificación son las expuestas en la sección 3.3.2. De manera general, tanto el indicador F1 como el valor de la precisión serán las métricas en las que nos centraremos para comentar los resultados dado que son las que mejor representan la capacidad de clasificación del sistema.

Un aspecto importante del proceso de clasificación es la evaluación conjunta que ha de realizarse para aunar los datos generados por la evaluación de cada región, esta última se realiza sobre el código latente generado para cada región cerebral. Cabe recordar que cada VAE o CVAE caracterizará una región por lo que se necesitarán un total 116 *autonencoders* para llevar a cabo el proceso de clasificación, cada uno de los cuáles se deberá de ajustar conforme a la región que caracterice. Se han empleado tres métodos diferentes de evaluación conjunta los cuales están expuestos dentro de la sección de trabajo realizado, ver 4.5.3. Los métodos empleados son el SVM (Máquina de Vectores de Soporte), el CMV (Voto por Mayoría Compleja) y el SMV (Voto por Mayoría Simple).

Con objeto de garantizar la validez de los resultados de clasificación se han empleado técnicas de validación cruzada como es el *K-fold*, dividiéndose el espacio muestral en 10 grupos y realizándose la prueba de clasificación 10 veces tomando uno de los grupo como espacio de test. Esta técnica quedó expuesta dentro de los fundamentos teóricos en la sección 3.3.3.

VAE con MRI

En esta prueba se ha utilizado el modelo VAE y se ha realizado un barrido de valores sobre el número de neuronas de la capa latente. Es de esperar que conforme aumentemos la dimensionalidad de la capa latente, el sistema sea capaz de codificar más información en el código latente y por lo tanto se tengan mejores resultados.

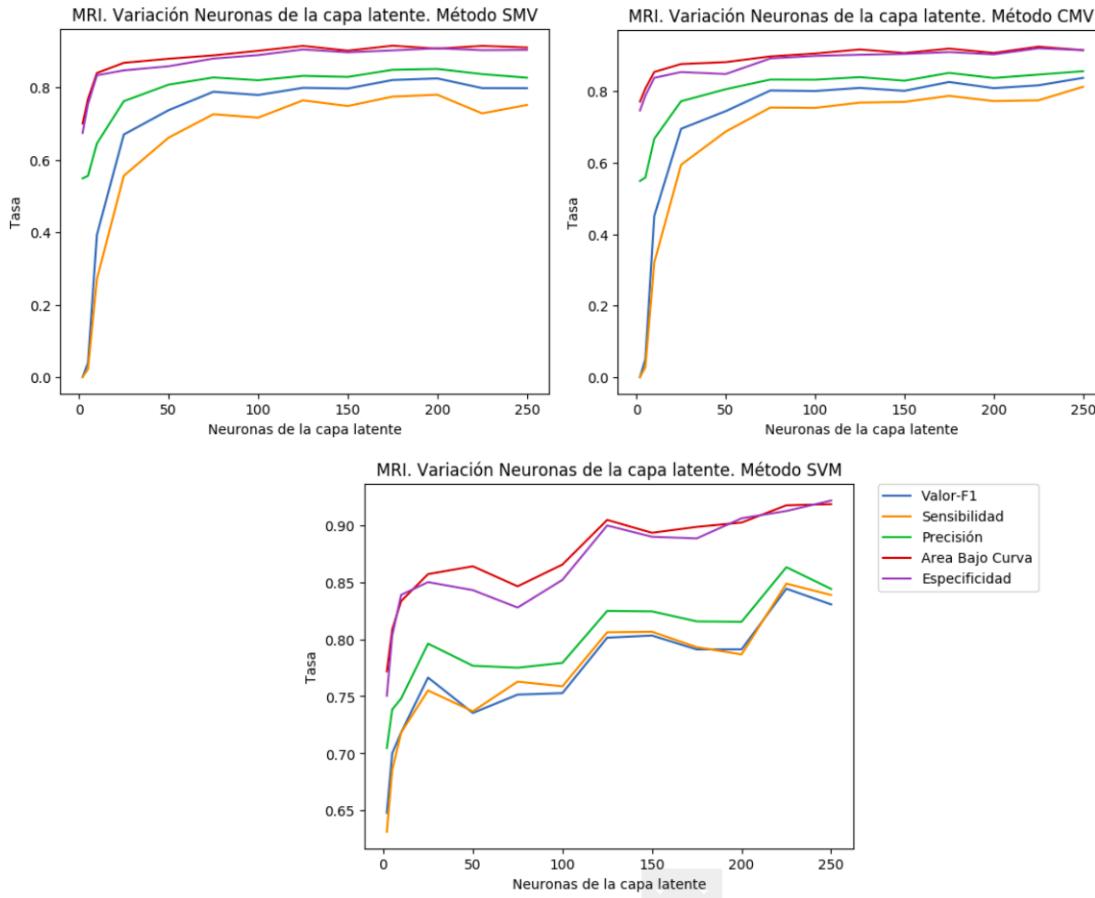


Figura 5.12: Resultados de clasificación sobre las imágenes MRI aplicando las tres técnicas de evaluación conjunta de resultados. (Izquierda) Voto por Mayoría Simple. (Centro) Máquina de Vectores de Soporte. (Derecha) Voto por Mayoría Compleja.

En la figura 5.13 se comprueba como al aumentar el tamaño de la capa latente se consiguen mejores resultados, especialmente para el caso del clasificador SVM.

Si comparamos los tres tipos de clasificación conjunta observamos que los denominados SMV y CMV se comportan de manera muy similar y no consiguen superar valores del 80 % para ninguna de las métricas empleadas. Sin embargo, en el método

SVM se observa como evolucionan progresivamente los valores de las métricas conforme aumentamos el valor de la capa latente hasta llegar a valores de precisión del 85 % y de puntuación cercanos al 84 % .

VAE con PET

En esta prueba se emplea de nuevo un modelo de VAE de redes neuronales densas pero en este caso aplicado sobre las imágenes PET. Se realizar un barrido sobre la dimensionalidad del código latente, o lo que es lo mismo, sobre el número de neuronas de la capa latente.

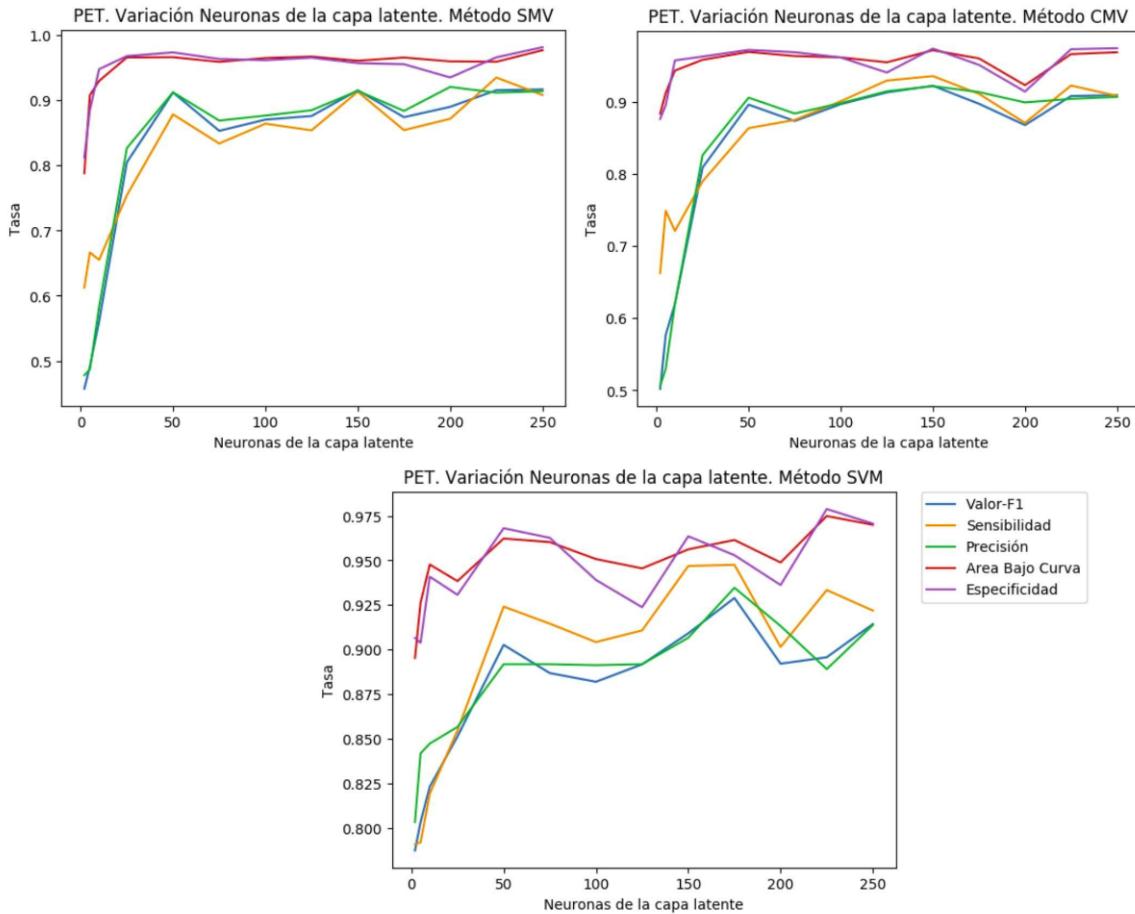


Figura 5.13: Resultados de clasificación sobre las imágenes PET aplicando las tres técnicas de evaluación conjunta de resultados. (Izquierda) Voto por Mayoría Simple. (Centro) Máquina de Vectores de Soporte. (Derecha) Voto por Mayoría Complejo.

En la figura 5.13 se puede comprobar como nuevo los mejores valores de clasi-

ficación se consiguen para el método SVM. Como mejores resultados se consiguen para una capa latente de 150 neuronas una puntuación F1 del 90,90 % y de precisión 90,60 %.

CVAE con MRI

En esta prueba se ha empleado un CVAE y se ha realizado un barrido sobre el tamaño del kernel del procedimiento de convolución del autoencoder. Se ha utilizado como espacio muestral las imágenes MRI.

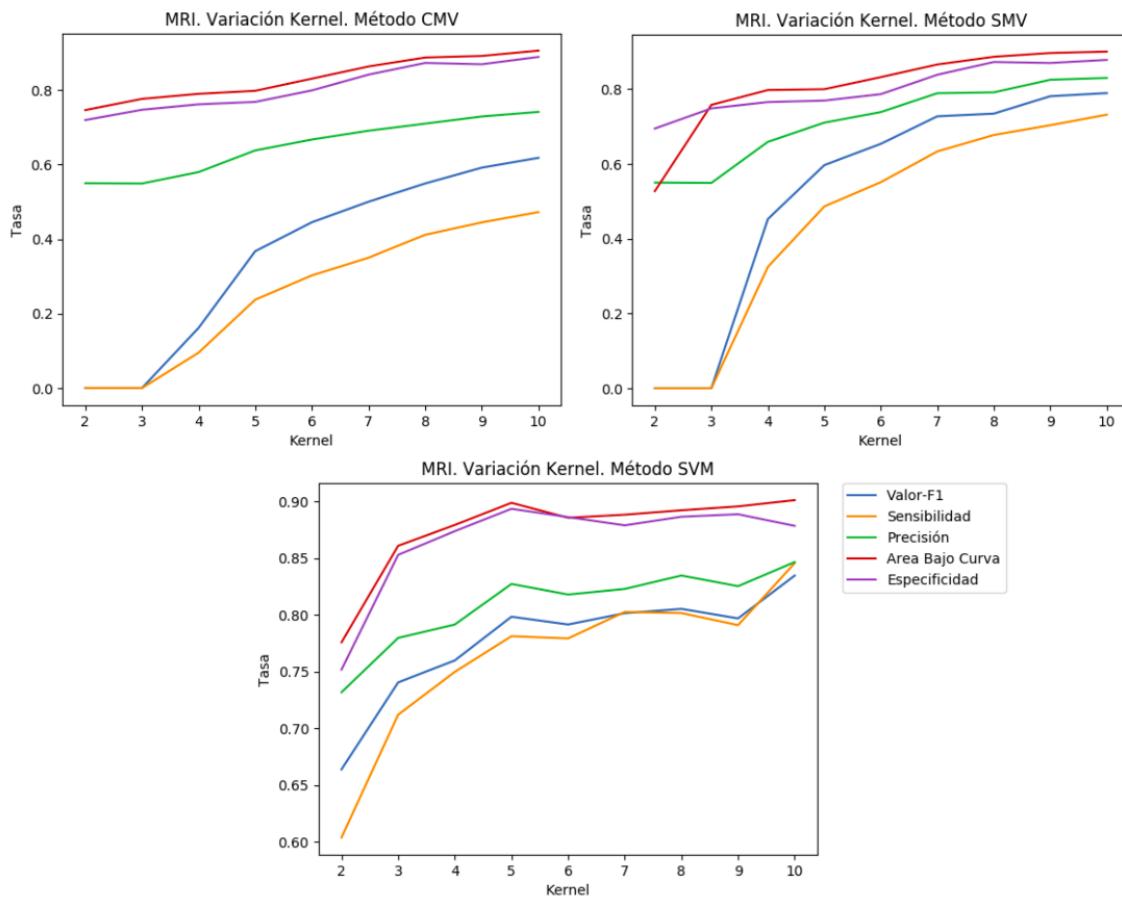


Figura 5.14: Resultados de clasificación sobre las imágenes MRI empleando un CVAE y variando el tamaño del kernel empleado. En la figura se tiene una gráfica por cada una de las tres técnicas de evaluación conjunta de resultados empleadas. (Izquierda) Voto por Mayoría Simple. (Centro) Máquina de Vectores de Soporte. (Derecha) Voto por Mayoría Complejo.

CVAE con PET

En esta prueba se ha empleado un CVAE y se ha realizado un barrido sobre el tamaño del kernel del procedimiento de convolución del autoencoder. Se ha utilizado como espacio muestral las imágenes PET.

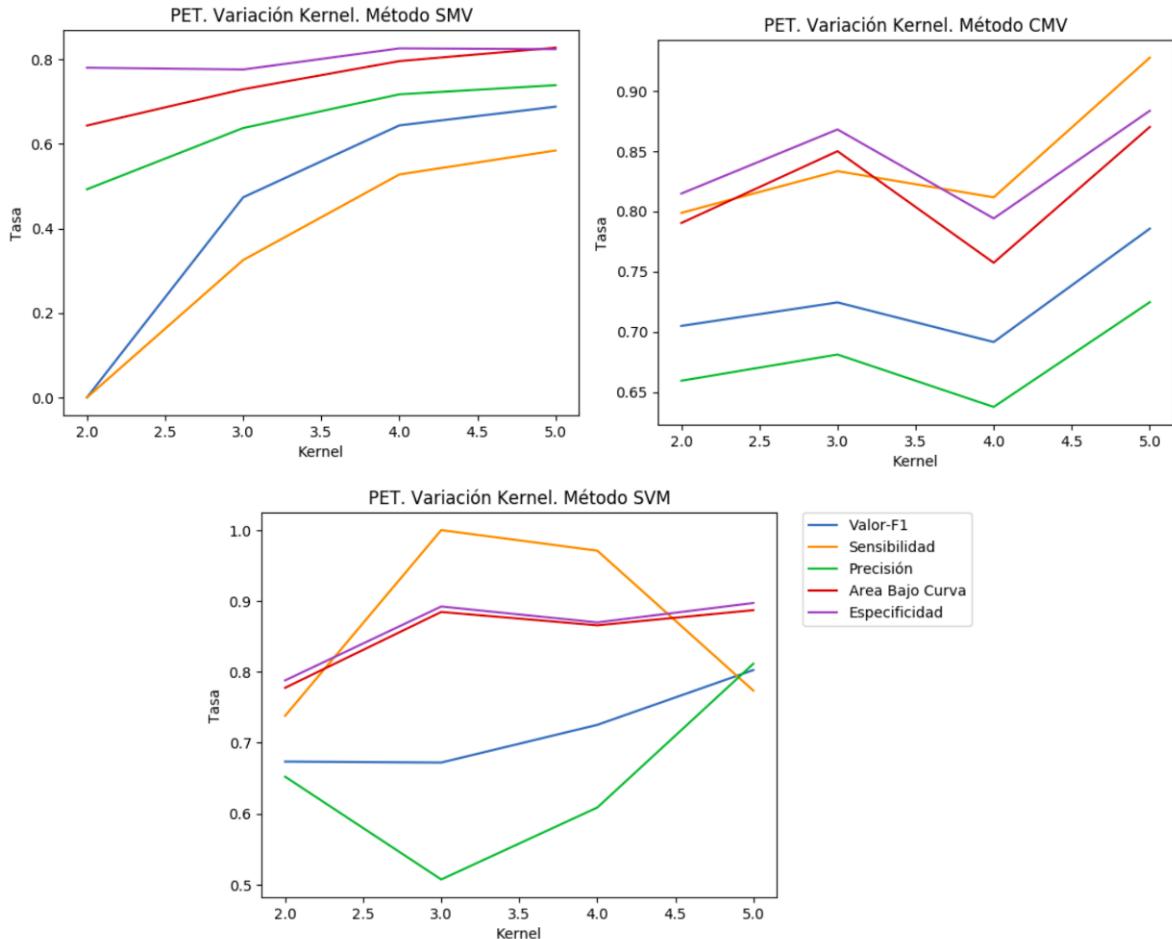


Figura 5.15: Resultados de clasificación sobre las imágenes MRI utilizando un CVAE y aplicando un barrido en el tamaño del kernel de Convolución. Cada imagen se corresponde con un método de evaluación conjunta de resultados. (Izquierda) Voto por Mayoría Simple. (Centro) Máquina de Vectores de Soporte. (Derecha) Voto por Mayoría Complejo.

En este caso se han obtenido valores en las métricas de evaluación menores en comparación con lo obtenido en la simulaciones de clasificación con el VAE. A priori, era de esperar que el modelo convolucional funcionaría mejor ya que este modelo de redes neuronales es capaz de analizar y caracterizar las imágenes. No obstante, esta

simulación no ha podido realizarse de forma completa debido a problemas con el servidor en el cuál realizamos las simulaciones. Algunas de las problemáticas que han podido alterar el resultado son:

- Se ha empleado un K-fold de tres grupos en lugar de diez, lo cual reduce el espacio muestral de entrenamiento y, por lo tanto, reduce la capacidad de generalización y de conseguir un ajuste óptimo de los pesos del sistema.
- El modelo de CVAE utilizado, caracterizado por los parámetros de configuración, no es el más óptimo posible dado que posteriormente durante la realización de diversas pruebas se ha comprobado que otros modelos con diferentes ajustes te parámetros funcionaban mejor. No se ha podido realizar la simulación con estos modelos más óptimos debido a problemas con el servidor.

Conclusiones y líneas futuras

En este trabajo se ha analizado la técnica generativa denominada Autoencoder Variacional realizándose dos implementaciones Software de dicha técnica utilizando el lenguaje *Python*. Esta modelo de aprendizaje no supervisado está basado en aprendizaje profundo, es por ello que se ha empleado la librería *Tensorflow* que da soporte a la implementación de redes neuronales para su ejecución tanto en CPU como en GPU. Algunos de los aspectos a destacar, así como algunas de las contribuciones realizadas durante el desarrollo del trabajo son las siguientes:

- Se ha implementado un modelo de VAE basado en redes neuronales densas. El modelo es fácilmente configurable tanto en variables de configuración como es la tasa de aprendizaje o en parámetros estructurales como son el número de capas del modelo o el número de neuronas para cada capa.
- Se ha implementado un modelo de VAE basado en redes neuronales convolucionales. En este caso las variables de configuración son modificables fácilmente, y se han habilitado una serie de modelos que habilitan la posibilidad de seleccionar entre distintas estructuras para el sistema.

Esta capacidad para variar tanto los parámetros del modelo como ciertos aspectos estructurales permite evaluar diferentes configuraciones con objeto de obtener la óptima.

- Se ha evaluado la capacidad de los modelos implementados de discernir entre pacientes AD y NOR empleando tanto muestras PET como MRI.

Para las imágenes MRI se ha obtenido una precisión máxima de clasificación de hasta el 84 % para el VAE. En otros estudios, empleándose este mismo espacio muestral se han conseguido resultados superiores al 90 %, por lo que son bastante superiores a los conseguidos en este trabajo. No obstante el objetivo no era conseguir un clasificador óptimo que consiguiera competir con los modelos actuales sino demostrar la capacidad de discernir entre las neuroimágenes de un sujeto NOR y las de uno AD.

Para las imágenes PET se ha obtenido una precisión máxima en torno al 90 %. De nuevo estos resultados no compiten con los modelos del estado del arte actual que ronda el 95 % con este mismo espacio muestral.

- El objetivo principal del trabajo era conseguir realizar la síntesis de neuro-imágenes 3D a partir de la caracterización del VAE. No obstante, las síntesis realizadas utilizando el código generado por las propias imágenes del espacio muestral han generado unas imágenes que apenas se asemejaban a las originales.

Tanto para el modelo VAE de redes densas como para el CVAE se ha tenido un resultado nefasto para la síntesis aunque hay diversos aspectos en la implementación de ambos modelos que son diferentes. Esto genera la duda de si el problema está en la implementación del modelo o en el método VAE en sí que no se ajuste al objetivo de este trabajo.

En las simulaciones de visualización del código latente se ha comprobado como es posible separar para algunas regiones el código generado para imágenes de sujetos AD y para sujetos NOR. Esto nos indica que el autoencoder es capaz de generar un código diferente para los dos tipos de sujetos. Sin embargo durante la fase de reconstrucción se tiene que los códigos generan de nuevo el mismo tipo de imagen. Una imagen final que puede representar un valor medio de imagen que permite reducir el error de regeneración durante el proceso de entrenamiento.

En resumen, es realmente desconcertante que códigos latentes diferenciables entre NOR y AD generen finalmente un mismo tipo de imagen, o al menos muy similar.

- Otro aspecto importante es que el uso de redes convolucionales 3D como modelo extracción de características para el VAE no es una metodología que se haya usado en muchos trabajos relativos a esta temática, al menos durante la fase inicial de este proyecto no se encontró ningún código de implementación sobre *Tensorflow* de este tipo de red. Además, *Tensorflow* no dispone de la función encargada de realizar el proceso de agrupamiento (*pooling*) para imágenes 3D, lo cual lleva la imposibilidad de probar esta funcionalidad de reducción de características en el sistema.

Líneas futuras

En este trabajo la implementación se ha apoyado en *Tensorflow* para la realización de los diseños. Existen diversas librerías sobre *Python* que facilitan el mismo conjunto de funcionalidades que *Tensorflow*, tales librerías son *Keras* o

Theano, es por ello que una posible línea de trabajo sería realizar la implementación sobre alguna de estas librería y evaluar si los resultados de la síntesis de imágenes son similares. Esto nos permitiría descartar si los problemas de este trabajo son debidos a las implementaciones aquí realizadas.

Además de VAE, existen otros tipos de técnicas generativas que se basan en el uso de aprendizaje profundo. Las redes generativas adversarias, que fueron brevemente comentadas en la sección 2.3.1, constituyen un método caracterizado por generar una síntesis de imágenes más realista que el VAE pero que es ciertamente más complejo de entrenar y de llegar a una configuración óptima de los parámetros.

Bibliografía

- [1] Henley, David B., Sundell, Karen L., Sethuraman, Gopalan, Siemers, Eric R., 2011, *Safety profile of Alzheimer's disease populations in Alzheimer's Disease Neuroimaging Initiative and other 18-month studies*, 407-416
- [2] Rodríguez Álvarez M., Sánchez J. L., 2004, *Reserva cognitiva y demencia*, 2004, vol. 20, nº 2 (diciembre), 175-186
- [3] Petrella J.R, Coleman R.E., Doraiswamy P.M., Neuroimaging and Early Diagnosis of Alzheimer Disease: A Look to the Future. Radiology 2003; 226:315?336.
- [4] Rémi Cuingnet et al, *Automatic classification of patients with Alzheimer's disease from structural MRI: A comparison of ten methods using the ADNI database*, Neuroimage, vol. 56, no. 2, pp. 766-781, 2011.
- [5] W. Cai, D. Feng, R. Fulton, *Content-based retrieval of dynamic PET functional images* IEEE Trans. Inf. Technol. Biomed., vol. 4, no. 2, pp. 152-158, 2000.
- [6] Ortiz, A.; Fajardo, D.; Górriz, J.M.; Ramírez, J.; Martínez-Murcia, F.J., *Multimodal image data fusion for Alzheimer's Disease diagnosis by sparse representation*, International Conference on Innovation in Medicine and Healthcare (InMed), 2014
- [7] Daoqiang Zhang, Yaping Wang, Luping Zhou, Hong Yuan, Dinggang Shen, *Multimodal Classification of Alzheimer's Disease and Mild Cognitive Impairment* Neuroimage. 2011 April 1; 55(3): 856-867
- [8] D. Salas-Gonzalez, J. M. Gorriz, J. Ramírez, M. Lopez, I Alvarez, *Compute-aided diagnosis of Alzheimer's disease using support vector machines and classification trees* Phys. Med. Biol. 55 (2010) 2807-2817
- [9] Ruaa Adeeb Abdulmunem Al-falluji *MRI based Techniques for Detection of Alzheimer: A Survey* International Journal of Computer Applications (0975 - 8887) Volume 159 - No 5, February 2017

- [10] S.Mareeswari1, Dr.G.Wiselin, *A survey Early Detection of Alzheimer's Disease using different techniques* International Journal on Computational Science and Applications (IJCSA) Vol.5, No.1,February 2015
- [11] Bellman RE, 1961, *Adaptive control processes: a guided tour*, Princeton University Press.
- [12] David L. Donoho Department of Statistics *High-Dimensional Data Analysis: The Curses and Blessings of Dimensionality* August 8, 2000
- [13] Benson Mwangi, Tian Siva Tian, Jair C. Soares *A review of feature reduction techniques in neuroimaging* Neuroinformatics. 2014 April ; 12(2): 229-244. doi:10.1007/s12021-013-9204-3.
- [14] Siqi Liu, Sidong Liu, *EARLY DIAGNOSIS OF ALZHEIMER'S DISEASE WITH DEEP LEARNING*
- [15] Saman S., Ghassem T., *Classification of Alzheimer's Disease Structural MRI Data by Deep Learning Convolutional Neural Networks* 22 Jul 2016
- [16] Diederik P. Kingma, Max Welling, *Auto-Encoding Variational Bayes* 1 May 2014
- [17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning Internal Representations by Error Propagation 9 October 1986
- [18] : I. Guyon, G. Dror, V. Lemaire, G. Taylor and D. Silver Autoencoders, Unsupervised Learning, and Deep Architectures 2012
- [19] P. Vincent H. Larochelle I. Lajoie Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion 2010
- [20] Danilo J. Rezende, Shakir Mohamed, Daan Wierstra Stochastic Backpropagation and Approximate Inference in Deep Generative Models
- [21] Bennett, D. A., Evans, D. A., 1922. Alzheimer's Disease. Disease-a-Month 38(1), 7-64.
- [22] Nakako, S., Kato, T., Nakamura., 1996. Acetylcholinesterase activity in cerebrospinal fluid of patients with alzheimer's disease and senile dementia. Journal of the Neurological Sciences 75(2).
- [23] Rodriguez-Violante M., Cervantes A., Vargas S., 2010. Papel de la función mitocondrial en las enfermedades neurodegenerativas. Arch Neurocienc (Mex) Vol. 15, N°1: 39-46

- [24] Swerdlow, R., 2011 Brain aging, alzheimer's disease, and mitochondria. *Biochim Biophys Acta* 1812(12), 1630-1639
- [25] Nations U., 2008 Department of economic and social affairs, world population prospects.
- [26] Arnáiz E., Almkvist O., 2003,. Neuropsychological features of mild cognitive impairment and preclinical Alzheimer's disease.
- [27] Palmer K., Berger A. K., Monastero R., Winblad B., Bäckman L., Fratiglioni L. 2007. Predictors of progression from mild cognitive impairment to Alzheimer disease. *Neurology* 68 (19): 1596-1602. PMID 17485646. doi:10.1212/01.wnl.0000260968.92345.3f.
- [28] Carlesimo GA, Oscar-Berman M (junio de 1992). «Memory deficits in Alzheimer's patients: a comprehensive review». *Neuropsychol Rev* 3 (2): 119-69. PMID 1300219.
- [29] Frank EM (septiembre de 1994). «Effect of Alzheimer's disease on communication function». *J S C Med Assoc* 90 (9): 417-23. PMID 7967534.
- [30] Volicer L, Harper DG, Manning BC, Goldstein R, Satlin A., Mayo de 2001. Sundowning and circadian rhythms in Alzheimer's disease». *Am J Psychiatry* 158 (5): 704-711. PMID 11329390.
- [31] Mu Y., Gage FH, 2011 Dec, *Mol Neurodegener.* 2011 Dec 22;6:85. doi: 10.1186/1750-1326-6-8 Adult hippocampal neurogenesis and its role in Alzheimer's disease.
- [32] Feldman H. H., *Atlas of Alzheimer's Disease*. Informa Healthcare
- [33] Susanne G. Mueller, Michael W. Weiner, *Neuroimaging Clin N Am.* 2005 November ; 15(4): 869?xii. The Alzheimer's Disease Neuroimaging Initiative
- [34] Gorji, H. T.; Haddadnia, J. (2015-10-01). ".^A novel method for early diagnosis of Alzheimer's disease based on pseudo Zernike moment from structural MRI". *Neuroscience.* 305: 361?371. ISSN 1873-7544. PMID 26265552. doi:10.1016/j.neuroscience.2015.08.013.
- [35] Zhu, Xiaofeng; Suk, Heung-Il; Shen, Dinggang (2014-10-15). ".^A novel matrix-similarity based loss function for joint regression and classification in AD diagnosis". *NeuroImage.* 100: 91?105. ISSN 1095-9572. PMC 4138265?Freely accessible. PMID 24911377. doi:10.1016/j.neuroimage.2014.05.078.

- [36] Wright G., Magnetic Resonance Imaging EEE Signal Process. Mag. 14 (1997)56?66.
- [37] Lauterbur P. C., 1973. "Image Formation by Induced Local Interactions: Examples of Employing Nuclear Magnetic Resonance". Nature. 242 (5394):
- [38] "Britain's brains produce first NMR scans". New Scientist: 588. 1978.
- [39] Snehal More, V.V.Hanchate, .^A Survey on Magnetic Resonance Image Denoising MethodsInternational Research Journal of Engineering and Technology (IRJET), Volume: 03 Issue: 05 | May-2016
- [40] Asja Fischer, Christian Igel, *An Introduction to Restricted Boltzmann Machines*
- [41] Vincent H., Larochelle H., Lajoie. I., *Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion*
- [42] Vincent, Pascal Larochelle, Hugo Bengio, Yoshua *Extracting and Composing Robust Features with Denoising Autoencoders*, (ICML'08), pages 1096 - 1103, 2008
- [43] Goodfellow, Ian J.; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron; Bengio, Yoshua (2014), "Generative Adversarial Networks". arXiv:1406.2661?
- [44] Salimans T., Goodfellow I., Zaremba., Cheung, Vicki; Radford, Alec; Chen, Xi (2016). "Improved Techniques for Training GANs". arXiv:1606.03498?
- [45] Eunbyung Park University of North Carolina at Chapel Hill
- [46] *MRI based Techniques for Detection of Alzheimer: A Survey* Ruaa Adeeb Abdalmunem Al-falluji University of Babylon, International Journal of Computer Applications (0975 ? 8887), Volume 159 ? No 5, February 2017
- [47] Yang, Wenlu, et al. "Independent component analysis- based classification of Alzheimer's disease MRI data."Journal of Alzheimer's disease 24.4 (2011): 775-783.
- [48] 18F-FDG PET imaging analysis for computer aided Alzheimer's diagnosis? I.A. Illán, J.M. Górriz,J. Ramírez, D. Salas-Gonzalez, M.M. López, F. Segovia, R. Chaves, M. Gómez-Rio c, C.G. Puntonet, Information Sciences 181 903?916, 2011.
- [49] Herrera, Luis Javier, et al. Classification of MRI Images for Alzheimer's Disease Detection."Social Computing (SocialCom), 2013 International Conference on. IEEE, 2013.

- [50] Zhang, H., Berg, A.C., Maire, M., Svm-knn, J.M.: Discriminative nearest neighbor classification for visual category recognition. In: CVPR ?06, pp. 2126?2136. IEEE Computer Society, Los Alamitos, CA, USA (2006)
- [51] Weiner, Michael (2017). Recent publications from the Alzheimer's disease neuroimaging initiative: reviewing progress toward improved AD clinical trials.". *Alzheimer's and dementia*.
- [52] Suk, Heung-Il; Lee, Seong-Whan; Shen, Dinggang; Alzheimer's Disease Neuroimaging Initiative (2016-06-01). "Deep sparse multi-task learning for feature selection in Alzheimer's disease diagnosis".
- [53] Zu, Chen; Jie, Biao; Liu, Mingxia; Chen, Songcan; Shen, Dinggang; Zhang, Daoqiang; Alzheimer's Disease Neuroimaging Initiative (2016-12-01). "Label-aligned multi-task feature learning for multimodal classification of Alzheimer's disease and mild cognitive impairment". *Brain Imaging and Behavior*.
- [54] Hosseini-Asl E., Keynton R., El-Baz A., *Alzheimer's disease diagnostics by adaptation of 3d convolutional network* (Julio 2016), arXiv:1607
- [55] Suk H-I., Shen D., *Deep Learning-Based Feature Representation for AD/MCI Classification*, Medical image computing and computer-assisted intervention?(MICCAI), International Conference on Medical Image Computing and Computer-Assisted Intervention. 2013;16(0 2):583-590.
- [56] Abadi M, Agarwal A., Barham P., B. Eugene,. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*, November 9, 2015
- [57] D.E. Rumelhart, J.L. MacClelland (eds.) (1986). Parallel Distributed Processing. Vol 1. Foundations, MIT Press
- [58] Glorot, X., Bengio, Y., (2010). *Understanding the difficulty of training deep feedforward neural networks* Journal of Machine Learning Research - Proceedings Track. 9. 249-256.
- [59] LeCun, Yann; Bengio, Yoshua; Hinton, Geoffrey (2015). "Deep learning". *Nature*. 521 (7553): 436?444. Bibcode:2015Natur.521..436L. PMID 26017442. doi:10.1038/nature14539.
- [60] Xavier Glorot and Yoshua Bengio, *Understanding the difficulty of training deep feedforward neural networks*, (2010) In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS?10). Society for Artificial Intelligence and Statistics

- [61] LeCun, Y., Boser, B., Denker, J. S., Backpropagation applied to handwritten zip code recognition. *Neural Comput.* , 1(4):541?551, 1989
- [62] Visualizing and Understanding Convolutional Networks Ciresan, D. C., Meier, J., and Schmidhuber, J. Multi- column deep neural networks for image classification. In CVPR , 2012.
- [63] Krizhevsky, A., Sutskever, I., and Hinton, G.E. Im- agenet classification with deep convolutional neural networks. In NIPS , 2012.
- [64] Y. Zhang and W. Zhou, Multifractal analysis and relevance vector machine-based automatic seizure detection in intracranial, *Int. J. Neural Syst.* 25(6) (2015) 1?14.
- [65] E. Castillo, D. Peteiro-Barral, B. Guijarro Berdinhas and O. Fontenla-Romero, Distributed one-class support vector machine, *Int. J. Neural Syst.* 25(7) (2015) 1?17
- [66] A. Hidalgo-Mun? noz, J. Górriz, J. Ramírez and P. Padilla, Regions of interest computed by svm wrapped method for Alzheimers disease examination from segmented mri, *Front. Aging Neurosci.* 6(20) (2014)
- [67] Hatie T., Tibshirani R., Friedman J., The elements of statistical learning, data mining, inferencen, and prediction. Springer, 2008
- [68] G. Flandin, F. Kherif, X. Pennec, D. Riviere, N. Ayache and J.-B. Poline, fMRI data analysis, Proceedings of the IEEE Int. Symposium on Biomedical Imaging (7?11 July 2002, Prague, Czech Republic), pp. 907?910.
- [69] Ashburner J., Friston KJ. *Voxel-based morphometry. The methods* Neuroimage. 2000 Jun;11(6 Pt 1):805-21.
- [70] J. Ashburner and T. Group, SPM8 Manuel, Vol. 12 (Functional Imaging Laboratory, Institute of Neurology, UK, 2011).
- [71] Structural Brain Mapping Group, Department of Psychiatry, Available at <http://dbm.neuro.unijena.de/vbm8/VBM8-Manual.pdf> [Accessed on 10 March 2014]
- [72] N. S. V. Villemagne, S. Berlangieri, S. Lee, M. Cherk, S. Gong, U. Ackermann, T. Saunder, H. TochonDanguy, G. Jones, C. Smith, G. OKeefe, C. Masters and C. Rowe, *Visual assessment versus quantitative assessment of 11c-pib pet and 18f-fdg pet for detection of Alzheimer? disease*, *J. Nucl. Med.* 48(4) (2004) 34?41.

- [73] Wen Zhu, Nancy Zeng, Ning Wang. *Sensitivity, Specificity, Accuracy, Associated Confidence Interval and ROC Analysis with Practical SAS Implementations* 1K&L consulting services, Inc, Fort Washington, PA Octagon Research Solutions, Wayne, PA, 2010.
- [74] Hwee Bee Wong, Gek Hsiang Lim. *Measures of Diagnostic Accuracy: Sensitivity, Specificity, PPV and NPV* Health Services Research and Evaluation Division, Ministry of Health, Singapore
- [75] Mandic, Danilo P. and Chambers, Jonathon A. *Towards the Optimal Learning Rate for Backpropagation* 2000, 01 Feb, Neural Processing Letters
- [76] Diederik P. Kingma, Jimmy Lei Ba. *Adam: A method for stochastic Optimization* arXiv:1412.6980v9 [cs.LG] 30 Jan 2017
- [77] Sebastian Ruder *An overview of gradient descent optimization algorithms* arXiv:1609.04747v2 [cs.LG] 15 Jun 2017

