UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

IMPLEMENTACIÓN DE ESTRATEGIA PARA CONTROL DE ESTIMULACIÓN EPIDURAL POR CIRCUITO CERRADO PARA TRATAMIENTO DE SÍNTOMAS PARKINSONIANOS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO

SERGIO IGNACIO EHIJO PAREDES

PROFESOR GUÍA:
MARCOS EDUARDO ORCHARD CONCHA

MIEMBROS DE LA COMISIÓN:
ROMULO FUENTES FLORES
ANDRÉS CABA RUTTE

Este trabajo ha sido parcialmente financiado por FONDECYT

SANTIAGO DE CHILE
2019

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: **SERGIO IGNACIO EHIJO PAREDES**
FECHA: 2019
PROF. GUÍA: MARCOS EDUARDO ORCHARD CONCHA

IMPLEMENTACIÓN DE ESTRATEGIA PARA CONTROL DE ESTIMULACIÓN EPIDURAL POR CIRCUITO CERRADO PARA TRATAMIENTO DE SÍNTOMAS PARKINSONIANOS

La enfermedad de Parkinson es un trastorno neurodegenerativo y objeto de interés en la comunidad científica, dado que existe una mayor incidencia luego de los 60 años en promedio y además la población mundial posee una alta esperanza de vida, implicando que existirá una mayor cantidad de casos en el futuro.

Los tratamientos actuales para este trastorno se componen de un generador de pulsos y electrodos, en donde se estimula de manera constante una zona del cuerpo (puede ser un área cerebral o parte de la médula) independiente del estado actual del paciente, lo cual conlleva en algunos efectos secundarios no deseados.

El presente trabajo de memoria se enfoca en esta problemática, ya que busca un lazo de control para estos tratamientos y corresponde a una de las primeras aproximaciones con aprendizaje de máquinas. En particular, se estudia un clasificador de movimiento con incertidumbre a partir de la actividad neural de un modelo animal de rata de 6-OHDA.

De esta manera, se realiza la extracción de movimiento a partir de un video y de señales cerebrales de un modelo animal de Parkinson a través de algoritmos de ventanas deslizantes, generando imágenes de potencia en ciertas bandas de frecuencia con una etiqueta respectiva de movimiento a partir del video. Estas imágenes sirven para entrenar a un clasificador de *Deep Learning Bayesiano*, el cual puede extraer incertidumbre en la clasificación.

Así, al utilizar Deep Learning Bayesiano con la forma de evaluación de MC Dropout se llega a obtener un *recall* de 80 % para la etiqueta de movimiento y la base de datos consistente en una ventana deslizante de medio segundo. Además, esta arquitectura es superior (para esta base de datos) en comparación a la de Deep Learning y evaluación estándar de dropout. Por otro lado, para estos resultados se tiene que con una mejor clasificación se obtiene una menor incertidumbre, lo cual es una de las ventajas al usar Deep Learning Bayesiano pues permite obtener una medida de confianza en la clasificación al realizar evaluaciones estocásticas.

Finalmente, cabe destacar que este trabajo puede usarse como base para obtener una estrategia de control para un circuito cerrado específico para cada paciente, el cual posee incertidumbre en predicciones implicando en la confianza que posee el sistema para cambiar un estado específico. Para generar una nueva estrategia más robusta con incertidumbre, se debería repetir este experimento agregando nuevos biomarcadores o indicadores fisiológicos, además de explorar otros algoritmos para extracción de movimiento para el etiquetado de la base de datos.

Para mi familia.

Agradecimientos

Agradezco a todas las personas que me ayudaron a lo largo de mis años de estudio en la FCFM, sobretodo en las semanas en que se necesitaba hablar de otras actividades y no de la carga académica. De manera especial me gustaría agradecer al Profesor Claudio Held, del Laboratorio de Bioingeniería, que supo apoyarme cuando tenía más dudas que respuestas sobre la elección de quedarme en Ingeniería o cambiar de carrera y seguir Medicina. Además, quiero agradecer también al *Laboratorio de Neuromodulación y Control Motor* por todo el apoyo que me han brindado en el último año de mi carrera y por promover el conocimiento de la neurociencia.

También agradezco al CEIE 2016 y al grupo de los Electrotutores, quienes apoyaron mis estudios y en conjunto intentamos hacer del DIE un lugar mejor para todos y todas.

Aprovecho de agradecer a mi profesor guía Marcos Orchard y a los miembros de mi comisión Rómulo Fuentes y Andrés Caba, quienes con su paciencia, empatía e interés me ayudaron cada vez que les solicitaba algún feedback, ya sea aclarando dudas, discutiendo sobre el estado actual del trabajo o bien generando nuevas ideas.

De manera especial, me doy el lujo de agradecer a Manuel y Lee Roy Sepúlveda, quienes con sus valores dentro y fuera de la cancha de basketball me han hecho aprender bastante, ya sea para convertirme en un mejor deportista como también para ser una mejor persona.

Por último agradezco a Freddy Flores, Mario Pérez, Pablo Cabargas, Ricardo Ramos, Camilo Carrasco, Germán García, Nicolás Tapia, Juan Pablo Pino, Gaspar Yévenes, Nicolás Espinoza, Esteban Leiva, Valentina Palma, Francisca Palaneck, Beatrice Trotter y a los *tíos* Rafa y Mari por todas las buenas vibras y palabras de ánimo en el desarrollo de este trabajo.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Motivación

La neurona es la unidad básica del sistema nervioso y su comportamiento no es simple, pues posee actividad eléctrica y habilidad para oscilar y resonar en distintos niveles de frecuencia, lo cual implica que existe información importante cuando se mide de manera correcta el tiempo de actividad de esta [1].

Para entender cómo ocurren distintos procesos a nivel cerebral es necesario hablar de *grupos de neuronas*, los cuales se definen como redes distribuidas locales de neuronas que se conectan dinámicamente de manera transitoria. Se cree que cuando surge un grupo neuronal, éste está detrás del funcionamiento de cada proceso cognitivo, y se plantea que las interacciones de los distintos grupos neuronales se logra mediante la sincronización de fase, la cual es un proceso que abarca tanto escalas temporales como espaciales del sistema nervioso. Existen distintos tipos de integración (conexiones entre los grupos neuronales), los cuales pueden ser locales o monosinápticas, o bien de gran escala o polisinápticas, las cuales pueden abarcar distintas regiones del cerebro [2].

En la escala de cientos de milisegundos ocurren varios procesos neuronales, y en uno de estos eventos una neurona puede disparar un par de impulsos nerviosos, pero estos impulsos no son lo suficientes para activar a una neurona de destino, a no ser que en el mismo tiempo otras neuronas de entrada produzcan un impulso en el mismo instante de tiempo; es decir, para activar a una neurona de destino, las neuronas de entrada deben sincronizarse en la producción de impulsos respectivos [3], indicando que es interesante analizar cómo se coordinan los distintos grupos neuronales para entablar interacciones a través de sinápsis[1] [2].

Para ver la actividad de estos grupos neuronales, se pueden medir sus señales de diferentes formas. La señal más local, llamada *actividad unitaria* o bien *single unit activity*, representa la actividad de espigas o *spikes* de una neurona (i.e. cuando esta dispara). Cuando en la señal medida las espigas no se pueden atribuir a una sola neurona, esta se procede a llamar *actividad*

[1]Regiones especializadas en las cuales se produce intercambio químico y/o eléctrico entre neuronas [4]

de multi neurona o *multiunit activity*. Hasta ahora se ha hablado de la señal que proviene de los grupos neuronales, es decir de sus outputs; según [1] los cambios lentos en el voltaje extracelular reflejan la suma de la actividad de las sinápsis cerca del electrodo de medición, es decir, la entrada o input neuronal. Cuando estas mediciones se realizan dentro del cerebro, se les otorga el nombre de **LFP**, por su sigla en inglés de *Local Field Potential* o potenciales de campo local. Cuando se realizan en el cuero cabelludo, se les llama electroencefalograma o EEG, y cuando se realizan en la corteza se les llama electrocorticograma o ECoG por sus siglas en inglés [5]. Las fuentes de los cambios lentos de voltaje en el medio extracelular son principalmente los potenciales sinápticos, seguidos de los potenciales de membrana que fluctúan en el tiempo, y también existe una contribución de las células gliales (como los oligodendrocitos y células de Schwann, que poseen una mayor población que las neuronas y tienen variadas funciones, entre las que destacan ayudar en el sistema inmune como también formar la vaina de mielina. Cuando se forma la mielina, tanto los oligondendrocitos como las células de Schwann ayudan en la conducción de la señal [4]). Dicho esto, es necesario enfatizar que las señales neurofisiológicas medidas pueden reflejar la salida o la entrada neuronal de un área del cerebro en específico, además de que la resolución espacial puede variar de poblaciones neuronales a medir tan sólo una neurona [6], como se puede ver en la Figura 1.1.

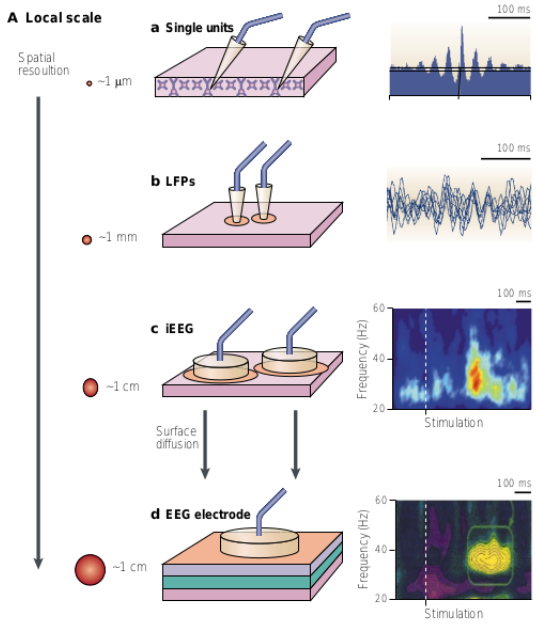


Figura 1.1: Escala local de medición. La resolución espacial va creciendo de **a** hacia **b**. La primera columna es un esquema de la medición, mientras que la segunda columna corresponde a una representación en el tiempo. En **a**, se tiene la medición de espigas para una sola neurona. En **b** se tiene la medición de *LFP* en donde la separación máxima entre electrodos fue de 7 mm. En **c** y **d** se tiene la medición de EEG, en donde el primero es intracranial. En ambos casos el gráfico correspondiente de la derecha es un espectrograma, el cual representa la potencia que se ve para ciertas frecuencias dentro del tiempo de una ventana o trial. Imagen de [2].

De manera histórica, las oscilaciones neuronales se han clasificado en distintas bandas de frecuencia, las cuales varían entre sus rangos dependiendo de los estudios en cuestión. Estas se conocen como las bandas delta, theta, alpha, beta y gamma, de menor a mayor frecuencia respectivamente. Por decir algunos ejemplos de procesos que tengan una correlación con ciertas bandas en particular, se tiene que la banda alpha tiene una correlación en momentos de vigilia y cuando los ojos se cierran, la banda delta se presenta en etapas de sueño profundo [7], mientras que existe un aumento de la banda beta cuando se sufre parkinsonismo [8].

Antes de hablar de la enfermedad de Parkinson (**BP**), será necesario realizar una descripción sobre los *núcleos de la base*[2] y ver cómo el circuito basal talamocortical se ve afectado por dicha enfermedad [9]. Los núcleos de la base se encargan de la regulación del movimiento y se componen de cuatro estructuras: el estriado (**STR**), el globo pálido (**GP**), la sustancia nigra (**SN**) y el núcleo subtalámico (**STN**) [4].

El estriado se compone por el núcleo caudado y el putamen, y es la estructura más importante de entrada del circuito, recibiendo variadas proyecciones del tálamo, tronco encefálico y corteza cerebral. El globo pálido se compone a su vez de dos partes, la externa *GPe* e interna *GPi*; la parte externa forma parte del circuito interno, mientras que la parte interna es la principal salida de los núcleos de la base. La sustancia nigra incluye a dos núcleos, siendo estos la *sustancia nigra pars compacta(SNpc)* y la *sustancia nigra pars reticulata(SNpr)*; la SNpc es un núcleo que contiene células dopaminérgicas (células que producen el neurotransmisor *dopamina*) que poseen varias proyecciones hacia el estriado además de proyectarse también a otros núcleos, mientras que la SNpr es la otra salida más grande de los núcleos de la base, de hecho, tanto la GPi como la SNpr se pueden ver como un solo canal separado por la cápsula interna. Finalmente, el núcleo subtalámico recibe proyecciones de parte del tálamo, tronco encefálico, la corteza cerebral y del globo pálido, y su salida se dirige hacia ambas partes del globo pálido y a la SPpr. La *vía hiperdirecta* se refiere a las entradas provenientes de la corteza y la vía subtalamopalidal (la salida del núcleo subtalámico hacia el globo pálido) [4]. El estriado se conecta con las dos salidas principales de los núcleos basales (GPi y SNpr) a través de la *vía directa* y la *vía indirecta*; la vía directa es la que va hacia el globo pálido interno, mientras que la vía indirecta pasa primero por el GPe, para luego tener conexiones tanto con el núcleo subtalámico como con el GPi [4], lo cual se puede ver en la Figura 1.2.

La enfermedad de Parkinson **PD** (*Parkinson's Disease*) es un trastorno o enfermedad neurodegenerativa que se ve causada por la muerte progresiva de neuronas primarias dopaminergicas(es decir, neuronas que liberan el neurotransmisor dopamina cuando generan un potencial de acción [11]) de la substancia nigra pars compacta. Como se puede ver en la Figura 1.2, cuando se genera esta muerte de neuronas dopaminérgicas, que por ahora se desconoce la causa [11], se produce un desbalance en la regulación hacia el putamen (estriado), lo cual lleva a desórdenes motores como rigidez, temblores y bradicinesia, que es la lentitud de algún movimiento efectuado [12] [13].

Se estima que la cantidad de enfermos en Chile llega a ser de cuarenta mil [14], y que los enfermos de *PD* van a ir aumentando, dado que la población mundial tiene una mayor esperanza de vida y generalmente hay una mayor incidencia después de los 60 años [15] [16].

[2]Cuando se habla de núcleo, se refiere a un *cluster* o grupo de neuronas dentro del Sistema Nervioso Central. Un núcleo en el Sistema Nervioso Periférico se conoce como ganglia [10].

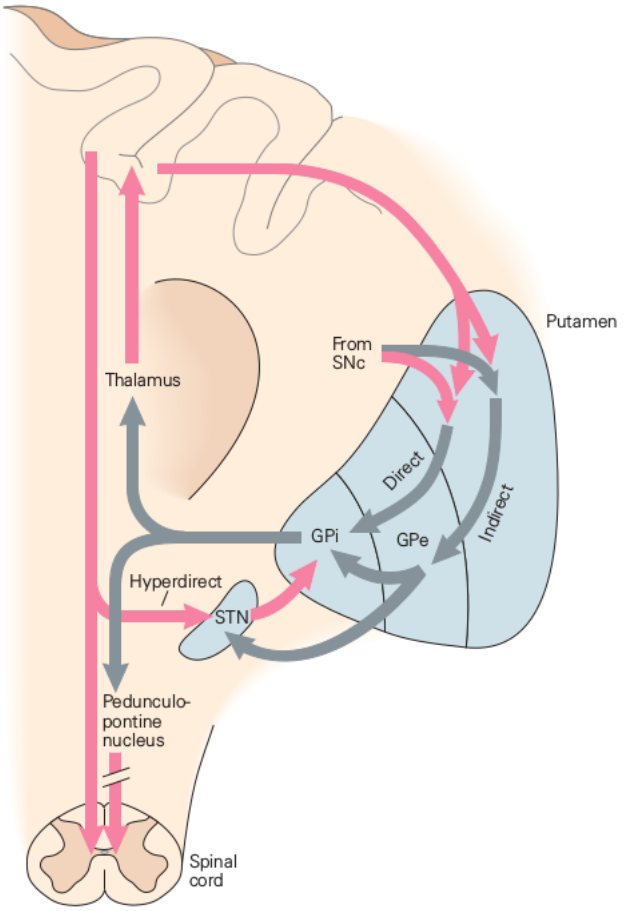


Figura 1.2: Esquema del **circuito ganglio talamocortical**, el cual se compone por el globus pallidus(GPi y GPe), el estriado representado por una de sus partes (el *putamen*), la sustancia nigra representada por la sustancia nigra pars compacta (SNc), y el núcleo subtalámico STN. Las proyecciones de parte de SNc hacia el estriado regulan la transmisión por las distintas vías. Las conexiones de inhibición son las grises, mientras que las excitatorias son las rosadas. Como se puede notar en la imagen, un estímulo proveniente de la corteza va hacia el putamen y al núcleo subtalámico, para luego seguir por alguna de las vías directa, indirecta o hiperdirecta. Imagen de [4].

Para poder estudiar esta enfermedad se han usado diferentes modelos animales para hacer un símil o mímica de la muerte de las neuronas dopaminérgicas a través de diferentes lesiones en el circuito ganglio talamocortical. De estos, destacan el modelo para primates **MTPT** (1-metil-4-fenil,6-tetrahidropiridina), el cual logra formar algunos síntomas parkinsonianos exceptuando los temblores de descanso en algunas especies [17] [18]; y también destaca el modelo de rata **6-OHDA**(6-hidroxidopamina) que se caracteriza por el agotamiento de dopamina en el lado cerebral en que se realizó la inyección de esta neurotoxina [19].

El tratamiento farmacológico preferido para tratar la enfermedad de Parkinson es a través de la administración de Levodopa (L-DOPA, L-3,4 dihidroxifenilalanina), el cual es un pre-

cursor químico de la dopamina (es decir, en este caso necesita de una enzima para sufrir una reacción de *descarboxilación* para transformarse en dopamina). Lleva más de 50 años siendo la droga más usada y útil para poder combatir los síntomas motores de la enfermedad, y además sirve como método para la confirmación de esta, dado que luego de 10 a 30 minutos pasados de la ingesta oral de esta pastilla el paciente puede mejorar su caminata. Lamentablemente este tratamiento no es óptimo pues se ha demostrado que luego de un tiempo prolongado ingestando L-DOPA, los pacientes pueden llegar a generar cambios de humor, ansiedad, discinesia[3]y fluctuaciones motoras, es decir, algunos de los síntomas motores de Parkinson pueden volver, además de *freezing* (episodios de congelamiento); estos surgen generalmente en períodos en que se acaban las dosis o bien en que la dosis sirve por un par de horas antes de que aparezcan [20] [21].

Para aliviar los síntomas motores que surgen debido a la ingesta prolongada de L-DOPA, se han creado diferentes tratamientos clínicos, de los cuales destacan **DBS** y **SCS**, *Deep Brain Stimulation* y *Spinal Cord Stimulation* respectivamente.

DBS es un tratamiento quirúrgico que se usa para epilepsia, transtorno obsesivo-compulsivo y Parkinson, siendo el tratamiento preferido para *PD* ya que mejora la calidad de vida, además de mejorar los temblores y rigidez entre el 70 y 75 % de los pacientes [22]; se compone en general de un generador de pulsos implantable, un electrodo que se coloca en cierta área cerebral de destino, y un cable que une el generador y el electrodo [23]. Para Parkinson, específicamente, el electrodo se suele poner en el *STN* o bien en el *GPi*, y se caracteriza por estimulación constante de alta frecuencia (alrededor de los 130 Hz), en donde los parámetros del pulso como el ancho, amplitud y frecuencia quedan establecidos gracias a un médico [24].

Por otro lado, **SCS** se enfoca en atacar los síntomas axiales de la enfermedad, como la postura y la caminata, los cuales son síntomas en que ni DBS ni Levodopa poseen algún efecto positivo. Esta técnica es menos invasiva en comparación a DBS, pues no requiere de una operación en el cerebro; el generador de pulsos se instala en la región abdominal o en un glúteo, mientras que los electrodos se instalan entre las vértebras T1 y T2, y los cables que unen al electrodo con el generador van por el espacio epidural, es decir, en el espacio entre las vértebras y la médula espinal [25] [26].

1.2. Descripción del Problema

Hasta ahora, ambas técnicas que se han presentado en la sección 1.1 presentan la modalidad de circuito abierto u *Open Loop*, lo cual significa que no hay ningún lazo o señal de control para ver cuándo enviar los pulsos a través del generador de pulsos (*IPG, Implantable Pulse Generator*), por lo que éste siempre envía pulsos predeterminados con los respectivos parámetros constantes entre visitas al médico o técnico a cargo, las cuales pueden variar entre tres a doce meses [27].

DBS presenta algunos efectos secundarios como disminución en la capacidad de habla, síntomas de índole psiquiátrica y además de una aparición de complicaciones motoras adicio-

[3]Se refiere a pequeños *ticks* o movimientos involuntarios [21].

nales, todos los cuales se piensa que se producen debido a la estimulación constante de alta frecuencia a distintos circuitos locales del cerebro [28] [29]. Se postula que dichos efectos se podrían mitigar si los pulsos son enviados sólo cuando ocurre un evento dado, el cual se puede relacionar con biomarcadores. Por ende, surge la idea de generar un circuito adaptativo o cerrado, el cual primero mejore la cantidad de individuos tratados de manera satisfactoria evitando efectos secundarios por estimulación excesiva de otros circuitos cerebrales, y en segundo lugar que tenga un efecto positivo en la batería del generador de pulsos (*IPG*), dado que disminuiría la cantidad de operaciones de recarga [30].

En particular, para *SCS* no se han reportado efectos secundarios luego de la operación para iniciar el tratamiento, aunque los parámetros de estimulación difieren bastante entre distintos estudios (entre 5 a 300 Hz para la frecuencia de estimulación), implicando que con un circuito cerrado se podría disminuir la cantidad de operaciones para la recarga de batería, tener la posibilidad de incrementar la cantidad de pacientes operados que tengan resultados óptimos, es decir mejorar los síntomas axiales de la enfermedad, y además plantear parámetros que puedan estar en concordancia entre diferentes estudios [25].

En la Figura 1.3 se puede apreciar un diagrama de un circuito cerrado, en particular asociado al tratamiento de DBS, en donde se tiene un sensor que recolecta los datos, que son convertidos en indicadores o características. A partir de estos indicadores, se puede realizar una estimación del estado actual del sistema y generar una señal de error, generando una compensación en forma de estimulación [31].

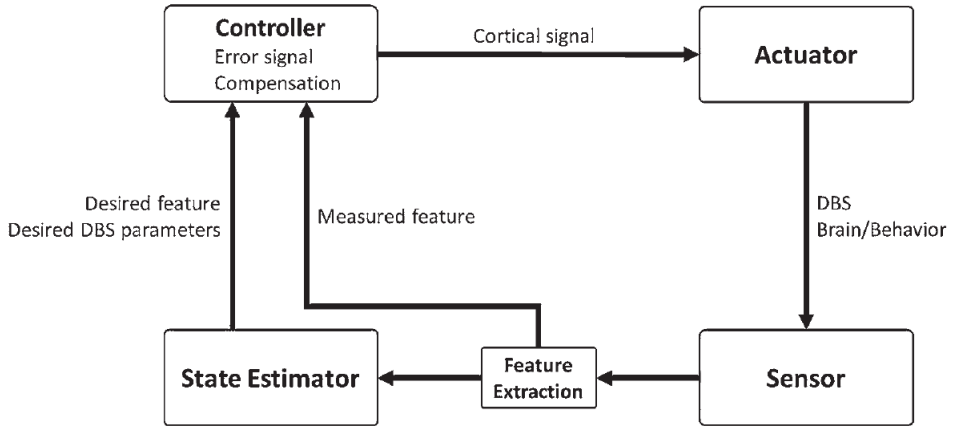


Figura 1.3: Diagrama de un circuito cerrado, compuesto por un sensor, un extractor de características, un estimador de estados, un controlador y un actuador o estimulador. Imagen de [31].

Por lo tanto, el problema a trabajar para esta Memoria de Título se enfoca en realizar la extracción de características y la estimación del estado actual del sistema. En específico, se caracteriza en generar un clasificador de algún biomarcador, para ejercer como detector de eventos para saber en un trabajo futuro cuándo generar un pulso o un episodio de estimulación de la zona epidural. En particular, se elige el biomarcador de *movimiento*, pues en de Da Silva et al (2018, [32]) muestran que la actividad de neuronas dopaminérgicas antes del inicio de movimiento modula el *vigor* o intensidad de movimiento futuro, implicando en la posibilidad

de explicar por qué pacientes que sufren la enfermedad de Parkinson seleccionan movimientos más débiles para empezar a moverse.

Ahora bien, la estimación del estado actual del sistema debiera poseer *incertidumbre*, es decir, un sistema de clasificación debiera entregar un resultado junto con la *confianza* que tiene el sistema en éste. Para ésto, se utilizarán dos técnicas (las cuales se explican en profundidad en la sección de Marco Teórico), las cuales se llaman Deep Learning (DL) y Deep Learning Bayesiano (DLB o BLD, por las siglas en inglés), las cuales son técnicas de clasificación y de aprendizaje supervisado especializadas en la clasificación de imágenes, en donde BLD posee un indicador de confianza sobre el estado que el modelo puede predecir.

1.3. Objetivos

1.3.1. Objetivo general

El objetivo general de esta memoria consiste en desarrollar un clasificador de movimiento con incertidumbre, en base a la actividad neural de un modelo animal de 6-OHDA de la enfermedad de Parkinson y a través de la comparación de algoritmos de inteligencia computacional.

1.3.2. Objetivos específicos

1. Generar una base de datos de imágenes de potencia a partir de la actividad neural de un modelo animal de rata de 6-OHDA bilateral de la enfermedad de Parkinson, en base a tratamiento sin estimulación.

2. Generar etiquetado de la base de datos a partir de la existencia de movimiento.

3. Diseñar estructuras de *Deep Learning* y *Deep Learning Bayesiano*.

4. Comparar ambas técnicas en base a su desempeño en la clasificación.

5. Comparar diferentes tiempos de ventana en el desempeño del clasificador.

1.4. Metodología de Trabajo

A continuación se detalla la metodología de trabajo que se desarrolla para encontrar la mejor estrategia para el circuito cerrado:

1. Elegir una base de datos de un modelo animal de rata 6-OHDA, que solo contenga el comportamiento de la rata sin haber sido estimulada externamente por un neuroestimulador.

2. A partir de la base de datos, realizar los marcados de la señal.

3. Luego de realizar los marcados, es necesario generar una transformación de las entradas para ser usados como imagen para la técnica de *Deep Learning Bayesiano*.

4. Se necesita caracterizar el diseño de la red a partir de tiempos característicos de la base de datos, utilizando el lenguaje de *Python*.

5. Se proceden a realizar entrenamiento y pruebas de clasificación de la red a partir del diseño hecho previamente, en conjunto de un análisis de la salida del clasificador.

6. Se comparan diferentes ventanas de tiempo para generar tanto las imágenes como el etiquetado, para evaluar al clasificador.

1.5. Aportes y Propuestas futuras

Luego de que este Trabajo de Título sea terminado con sus objetivos correspondientes cumplidos, se debe seguir una serie de pasos para poder probar este lazo de control; el más importante es lograr la integración de los distintos programas de software que deben formar parte de un circuito cerrado de estimulación, lo que actualmente radica en la generación del módulo particular de *PulsePal* en el software de *OpenEphys*. Luego de tener esto hecho, se debe integrar el clasificador realizado en este documento al Software base de Open Ephys, por ejemplo generando un *wrapper* de C++ y Python. Finalmente, con estos dos pasos hechos, se pueden empezar a generar pruebas de distintas versiones de circuitos cerrados en modelos animales de 6-OHDA, iterarando hasta encontrar un nuevo tratamiento; esto implica en potenciales trabajos de memoria de pregrado y de postgrado.

El presente trabajo de memoria es un aporte al área de neurociencias. Como ya se ha mencionado previamente, existe un gran interés por el desarrollo de nuevas técnicas de circuito cerrado para realizar nuevos tratamientos de la enfermedad de Parkinson, y el hecho de proponer algoritmos de aprendizaje de máquinas para ello abre la posibilidad de que otros equipos de ingeniería se interesen en trabajar en esta área, lo cual es óptimo pues se necesitan soluciones eficientes y pronto.

1.6. Estructura de la Memoria de Título

Este trabajo de título se compone de los siguientes capítulos:

(i) El *Capítulo 2* presenta el ***Marco Teórico y Estado del Arte***, en el cual se ve la principal teoría y los principales conceptos necesarios (desde el área de ingeniería) para entender el trabajo hecho, además de un breve resumen del estado actual de los circuitos cerrados en la enfermedad de Parkinson y de la técnica de *Deep Learning* usando como base de datos señales cerebrales.

(ii) El *Capítulo 3* exhibe la ***Metodología***, presentando los materiales usados y los pasos para realizar los experimentos.

(iii) El *Capítulo 4* muestra los ***Resultados*** obtenidos, junto a su correspondiente ***Análisis***.

(iv) El *Capítulo 5* está compuesto por las distintas ***Conclusiones*** del trabajo presentado. Es importante recalcar que además en este Capítulo se incorporan los principales desafíos y tareas a futuro con respecto a lo hecho en este documento.

Finalmente se presenta la ***Bibliografía*** utilizada.

Capítulo 2

Marco Teórico y Estado del Arte

En este capítulo se presentan los diferentes conceptos del área de ingeniería que serán necesarios para entender las técnicas a usar para obtener el mejor candidato de lazo cerrado, además de presentar el estado del arte en las técnicas de circuito cerrado existentes en neurociencia.

2.1. Marco Teórico

2.1.1. Redes neuronales

En Ingeniería, particularmente en el antiguo campo de automatización, siempre ha existido el interés por predecir o clasificar eventos, lo cual en el campo de *Machine Learning* se puede hacer a través del uso de datos y aprendizaje de una estrategia de predicción o red; en particular, una de estas estrategias se llama *red neuronal*, la cual es un tipo de aprendizaje supervisado[1]compuesto por un conjunto de capas o niveles que poseen unidades llamadas neuronas, que pueden ser del tipo *perceptrón* o del tipo *sigmoide*. Es conveniente empezar por explicar la unidad de perceptrón; esta se compone por diferentes entradas $\{x_j \mid j = 1, \ldots, n\}$ binarias (es decir, que pueden tomar dos valores, generalmente cero y uno) y pesos $\{w_j \mid j = 1, \ldots, n\}$ que se encargan de darle un cierto grado de importancia a la entrada. La salida de esta unidad se caracteriza por la comparación entre un valor constante llamado *treshold* y el producto punto entre el vector de entrada y el vector de pesos, lo cual se puede ver en Ec. 2.1, o su equivalente en Ec. 2.2 que demuestra una versión simplificada con el producto punto mencionado anteriormente.

$$\text{Valor de salida de un perceptrón} = \begin{cases} 0 & \text{si } \sum_j x_j * w_j \leq treshold \\ 1 & \text{si } \sum_j x_j * w_j > treshold \end{cases} \tag{2.1}$$

[1]Tipo de aprendizaje en que los datos poseen los respectivos *labels* o etiquetas. Se poseen típicamente tres conjuntos: uno de entrenamiento (en donde la red aprende), de validación (en donde se penaliza cómo aprende la red) y de prueba (en donde se evalúa el desempeño). Otro ejemplo de este tipo de red son los árboles de decisión.

$$\text{Valor de salida de un perceptrón} = \begin{cases} 0 & \text{si } x \cdot w + b \leq 0 \\ 1 & \text{si } x \cdot w + b > 0 \end{cases} \tag{2.2}$$

Por otro lado, la unidad *sigmoide* cuenta con la misma arquitectura que la del perceptrón, pero sus entradas pueden tomar cualquier valor entre cero y uno, y además posee una función de activación para su salida, la cual se llama *función sigmoide*. Esta posee como característica el entregar un output *entre* cero y uno, ser continua y derivable, y está definida por Ec. 2.3. Concretamente, el output de una unidad sigmoide viene dado por Ec. 2.4.

$$f(x) = \frac{1}{1 + \mathrm{e}xp(-x)} \tag{2.3}$$

$$\sigma(z) = \frac{1}{1 + \mathrm{e}xp(-z)} = \frac{1}{1 + \mathrm{e}xp(-\sum_j x_j * w_j - b)} \tag{2.4}$$

Las redes neuronales más frecuentemente usadas se conocen como *MLP (Multilayer perceptrons)* las cuales paradójicamente están compuestas por unidades[2]sigmoides; poseen capas características como la capa de *entrada*, *capas ocultas o hidden layers* y la capa de *salida*. Un ejemplo de este tipo de red se puede ver en la Figura 2.1.

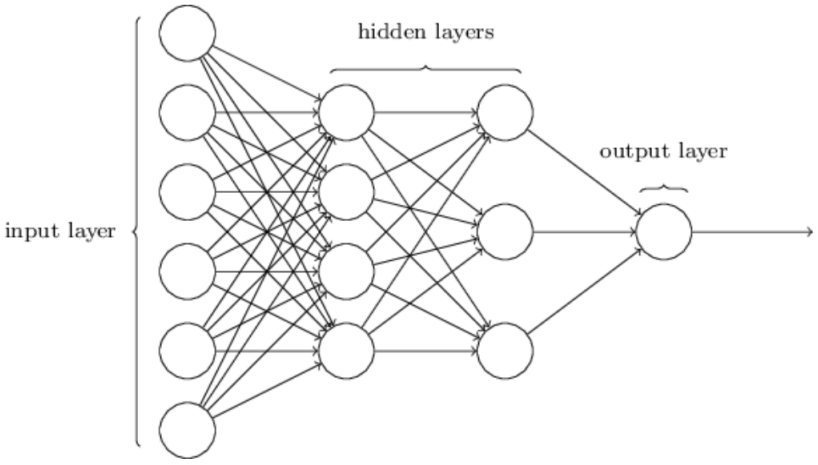


Figura 2.1: La capa de más a la izquierda corresponde a las entradas de la red (la que por ejemplo pueden ser números escritos a mano, pixeles de una imagen, texto, etc), luego las dos capas del centro corresponden a las *capas ocultas* debido a que simplemente no son ni entrada ni salida, y finalmente la capa de la derecha corresponde a la de salida de la red. En esta figura se tienen simplemente seis unidades sigmoides formando la capa de entrada, pero existen redes en que por ejemplo si se intenta codificar una imagen en escala de grises y esta es de 32x32 pixeles, se tendrían $32 * 32 = 1024$ unidades de entrada, cada una codificada entre cero y uno (por estar en escala de grises). Imagen de [33].

[2]En este trabajo de memoria se pretende usar la terminología de *unidad* para las neuronas que forman una red neuronal de ingeniería, y de neurona para la célula base del sistema nervioso.

En este tipo de redes *feedforward*, la salida de una unidad corresponde a la entrada para una unidad de la siguiente capa, lo cual se condice con el nombre debido a que la información fluye (*feed*) hacia adelante (*forward*) y no hay loops temporales (el tipo de redes con loops temporales se conoce como *Recurrent Neural Networks*. Por ende, dada una inicialización aleatoria de los pesos, la información fluye hacia adelante y surge la pregunta de cómo se realiza el aprendizaje de la red dados los datos de entrenamiento. Debido a que es aprendizaje supervisado, se desea que el resultado que la red entregue sea lo más cercano a la etiqueta inicial y para medir esto, se introduce el concepto de una función de costos. La razón de usar una función y no directamente aumentar directamente el número de predicciones correctas, es que la función puede depender de los pesos y los sesgos.

La primera función de costos a revisar se llama *Error Cuadrático Medio*, mejor conocido por sus siglas en inglés como *MSE*, el cual se define a través de la Ec. 2.5 en donde w representa los pesos, b representa el *bias* o treshold de Ec. 2.2, *y(x)* representa la etiqueta, n representa la cantidad de datos de entrenamiento, y_{output} representa la salida del sistema y finalmente $\|*\|$ representa la norma euclidiana de un vector (i.e., la raíz de la suma de cada componente al cuadrado).

$$C(w,b) = \frac{1}{2*n} \sum_{inputs} \|y(x) - y_{output}\|^2 \tag{2.5}$$

Uno de los problemas que presenta el MSE viene con la inicialización de pesos, generando lentitud para poder generar aprendizaje al minimizar su función de costos. Así, existe otra función que evita este problema y que se conoce como *Cross Entropy Function*, la cual se define por Ec. 2.6.

$$C(w,b) = -\frac{1}{n} \sum_{inputs} [y(x) ln(y_{output}) + (1 - y(x)) ln(1 - y_{output})] \tag{2.6}$$

Gradient Descent

Para minimizar la función de costos, se invita a pensar en una función de dos dimensiones (que tiene forma de valle). El método existente se llama *Gradient Descent*, el cual consiste en tomar pequeños pasos para que a partir de un punto se vaya retrocediendo al lugar más bajo del valle, el cual de manera óptima sería el mínimo global. Matemáticamente(y ya en n dimensiones), ir bajando por una función se traduce en tomar el gradiente de la función de costos, y la tasa con que se avanza o retrocede se conoce como *tasa de aprendizaje*. El gradiente de una función viene dado por Ec. 2.7, y el uso de Gradient Descent para cambiar los pesos y los sesgos se puede ver en Ec. 2.8, en donde η representa la tasa de aprendizaje.

$$\nabla C(X) = (\frac{\partial C}{\partial X_1}, \ldots, \frac{\partial C}{\partial X_n})^T \tag{2.7}$$

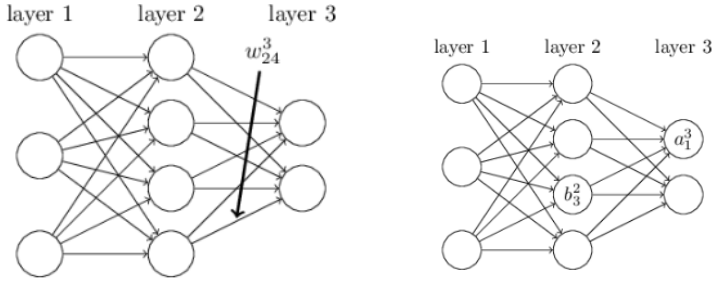
$$\begin{cases} w_{k+1} = w_k - \eta \frac{\partial C}{\partial w_k} \\ b_{k+1} = b_k - \eta \frac{\partial C}{\partial b_k} \end{cases} \tag{2.8}$$

Tanto en Ec. 2.5 como en Ec. 2.6 se puede ver que para calcular el costo total se debe obtener el costo por cada dato del sistema por separado, lo cual puede ser ineficiente para grandes volúmenes de datos. Esto se puede solucionar a través del método *Stochastic Gradient Descent*, el cual se basa en tomar una porción aleatoria de los datos y asumir que la suma de sus gradientes será similar a la suma de los gradientes de todos los datos, suponiendo que la porción tomada es lo suficientemente grande. Este conjunto de datos se conoce como *mini-batch*, y acelera el proceso de aprendizaje. Cuando se completa un mini-batch, se toma otra porción sin reposición de los mismos datos y se vuelve a iterar el proceso; cuando se cubren por completo los datos de entrada se dice que se completa una *época* de entrenamiento. Este proceso genera un cambio en la actualización de los pesos y sesgos (Ec. 2.9), en donde m es el tamaño del mini-batch.

$$\begin{cases} w_{k+1} = w_k - \frac{\eta}{m} \frac{\partial C}{\partial w_k} \\ b_{k+1} = b_k - \frac{\eta}{m} \frac{\partial C}{\partial b_k} \end{cases} \tag{2.9}$$

Backpropagation

Backpropagation nace como una manera eficiente para calcular $\dfrac{\partial C}{\partial w_k}$, dado que la función de costos de una unidad depende tanto de los pesos, sesgos y salidas de las unidades anteriores. Por ende, se empieza por ver una notación cómoda para entender cómo depende el output de una unidad de las unidades anteriores; para ello, se usará w^l_{jk} para representar el peso que va desde la unidad k-ésima desde la capa $l-1$ hacia la unidad j-ésima de la capa l, a^l_j y b^l_j para representar el valor de *activación* de la unidad y el bias o treshold de la unidad j-ésima de la capa l, respectivamente.



(a) Peso en función de la capa de llegada y unidades involucradas

(b) Valor de bias y de activación de cada unidad para cierta capa dada

Figura 2.2: Ejemplo de peso, bias y valor de activación en función de la capa correspondiente y salidas o entradas a la red. Imágenes de [33].

Se entiende el término de *activación* como el output de una unidad sigmoide, y esta viene dada en función de las activaciones de las unidades de la capa anterior según Ec. 2.10. En dicha ecuación se presenta una manera de eliminar la sumatoria de la notación a través del uso del producto punto, y también se presenta el término z que es simplemente es el resultado de una unidad antes de pasar por la función de activación.

$$a_j^l = \sigma\Big(\sum_k w_{jk}^l a_j^{l-1} + b_j^l\Big) \Leftrightarrow a^l = \sigma(w^l a^{l-1} + b^l) \Leftrightarrow a^l = \sigma(z^l) \tag{2.10}$$

A continuación se define el producto Hadamard (Def. 2.1), el cual es útil para ahorrar en notación y poder definir las ecuaciones del algoritmo de Backpropagation (**BP**).

Definición 2.1 (Producto de Hadamard) *Sean dos vectores de la misma dimensión. Se tiene que el producto Hadamard viene dado por*

$$u \odot v = (u \odot v)_j = u_j \times v_j$$

Como ejemplo, se puede ver que $\begin{bmatrix}1\\2\end{bmatrix} \odot \begin{bmatrix}3\\4\end{bmatrix} = \begin{bmatrix}1*3\\2*4\end{bmatrix} = \begin{bmatrix}3\\8\end{bmatrix}$, lo cual es fácilmente generalizable para n dimensiones.

Con esta última definición, es posible empezar a caracterizar el algoritmo de BP. Para ello, se puede comenzar definiendo el error δ de una unidad de cierta capa l con respecto a la función de costos, como se puede ver en la Ec. 2.11. También se puede ver esta ecuación en forma matricial en Ec. 2.12, considerando que cada componente de la fórmula es un vector que contiene a las unidades por capa.

$$\delta_j^L = \frac{\partial C}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L) \tag{2.11}$$

$$\delta^L = \nabla_a C \odot \sigma'(z_j^L) \tag{2.12}$$

Suponiendo que se tiene el error de la capa *L+1*, se puede obtener el error de la capa L a través de Ec. 2.13, el cual contiene la traspuesta de los pesos de la capa *L+1*, que intuitivamente es una manera de ponderar el error hacia atrás.

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z_j^l) \tag{2.13}$$

Con esto se puede obtener una ecuación para la variación de la función de costos con respecto a los sesgos y pesos (para poder obtener el aprendizaje de Ec. 2.9, los cuales se ven en Ec. 2.14 y Ec. 2.15 respectivamente.

$$\frac{\partial C}{\partial b_j^L} = \delta_j^l \tag{2.14}$$

$$\frac{\partial C}{\partial w_{jk}^L} = a_k^{L+1}\delta_j^L \tag{2.15}$$

Así, se puede generar una síntesis del algoritmo de BP a continuación:

Algorithm 1 Backpropagation

1: Input: Inicializar los pesos y las entradas, generando a^1 (activación de la primera capa)
2: Alimentar la red hacia adelante o hacer *feedforward*. Por cada capa $l = 2, 3, ..., L$, hacer el cálculo de $z^l = w^l a^{l-1} + b^l$ y de $a^l = \sigma(z^l)$.
3: Obtener el error de salida de la red, es decir calcular el vector $\delta^L = \nabla_a C \odot \sigma'(z_j^L)$
4: Aprender del error, es decir, hacer backpropagation para cada $l = L-1, L-2, ..., 2$, calcular $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z_j^l)$
5: Generar el output y actualizar los pesos, en donde los gradientes para reemplazar en Ec. 2.9 vienen dados por $\frac{\partial C}{\partial b_j^L} = \delta_j^l$ y $\frac{\partial C}{\partial w_{jk}^L} = a_k^{L+1} \delta_j^L$.

2.1.2. Métodos de regularización

Hasta ahora el método de BP que se ha introducido es el más básico, y en varios casos puede producir una mala generalización produciendo *overfitting*, es decir que se entrena muy bien con los datos de entrenamiento pero al momento de hacer pruebas con los *datos de test* la red no posee buen desempeño. Para evitar esto, existen diferentes técnicas que permiten mejorar el aprendizaje y generalización de la red; en particular, en este capítulo se verán por encima los términos de *regularización L1*, *regularización L2* y *Dropout*.

Comenzando por *regularización L2*, esta técnica consiste en agregar un término extra a la función de costos, como se puede ver en la Ec. 2.16 aplicado para la función de entropía cruzada, generando un cambio en la actualización de los pesos, mientras que los sesgos no sufren efecto con esta técnica como se puede ver en Ec. 2.17. Generalmente se dice que el término de regularización λ sirve para manejar y controlar los pesos hacia valores más pequeños, y de cierta manera pesos más pequeños poseen una complejidad menor y una manera más fácil de explicar los datos.

$$C(w,b) = -\frac{1}{n} \sum_{inputs,j} [y_j(x) ln(a_j^L) + (1 - y_j(x)) ln(1 - a_j^L)] + \frac{\lambda}{n} \sum_w w^2 \tag{2.16}$$

$$\begin{cases} w_{k+1} = w_k(1 - \frac{\eta\lambda}{n}) - \frac{\eta}{m} \frac{\partial C}{\partial w_k} \\ b_{k+1} = b_k - \frac{\eta}{m} \frac{\partial C}{\partial b_k} \end{cases} \tag{2.17}$$

Sigue la *regularización L1*, que difiere de L2 en solo tomar la suma de los valores absolutos de los pesos como se puede ver en Ec. 2.18, y en general esto afecta más para los pesos pequeños causando que esta técnica haga que la red se concentre en pesos de importancia, mientras que los pesos de poca importancia se vayan hacia cero. La actualización de pesos se ve en la Ec. 2.19.

$$C(w,b) = -\frac{1}{n} \sum_{inputs,j} [y_j(x) ln(a_j^L) + (1 - y_j(x)) ln(1 - a_j^L)] + \frac{\lambda}{n} \sum_w |w| \tag{2.18}$$

.

$$w_{k+1} = w_k(1 - \frac{\eta\lambda}{n}) - \frac{\eta}{m} \frac{\partial C}{\partial w_k} \tag{2.19}$$

Dropout

Dropout es una forma extraordinaria de regularización porque no ataca la función de costos sino que hace que la red en sí cambie. El proceso de *forward-propagation y backpropagation* se ven afectados de cierta manera, dado que cada unidad de cada capa se mantiene activa con una probabilidad p específica para cada una. Específicamente, se generan vectores con una distribución de *Bernoulli* en que cada componente toma el valor de cero con una probabilidad $0 \leq p_{\text{i}} \leq 1$, y cada vector de unidades de una cierta capa es multiplicado a través del producto de *Hadamard* (multiplicación por componente) generando el vector de unidades de esa capa que participarán del entrenamiento para un ejemplo específico.

Para hacer el entrenamiento, se hace el forward-propagation con la red luego de haber hecho el dropout, y luego se hace backpropagation con el resultado. Luego de hacer esto para varios ejemplos, se actualizan los pesos y sesgos. Se repite el proceso para otras unidades activadas aleatoriamente, y también se procede a hacer el FP y BP, actualizando finalmente el peso y bias de cada una.

Repitiendo este proceso una y otra vez, cada unidad va a aprender de manera más robusta las características de los inputs porque no sabe cuáles están conectadas a su alrededor, por lo tanto aquí hay un tipo de optimización; cuando finalmente se conectan todas las unidades, el peso al principio tomaba en cuenta una cantidad de unidades activadas dada por la probabilidad p, por ende de manera tradicional se compensa dividiendo por la mitad o por un múltiplo de p (esta es la manera tradicional de evaluación de dropout).

A modo de intuición, con dropout se están promediando los efectos de una gran cantidad de redes diferentes, y estas redes van a producir sobreajuste en distintas maneras, por lo que se espera que con este mecanismo se pueda reducir el *overfitting*, según [34]. En la Figura 2.3 se puede ver cómo es afectada una red neuronal producto del dropout.

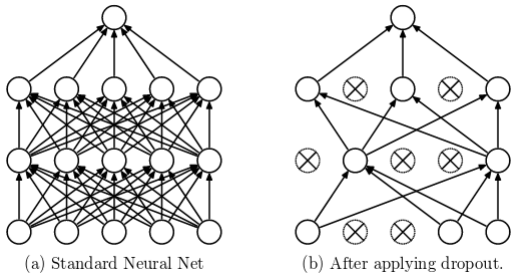


Figura 2.3: En **(a)** se tiene una red neuronal normal, mientras que en **(b)** se tiene la misma red luego de que dropout haya sido aplicado para un ejemplo [34]. Imagen de [34].

2.1.3. Deep Learning

En el caso particular de clasificación de imágenes, resulta un poco extraño usar redes neuronales *fully connected* (es decir, las redes de la sección 2.1.1) dado que éstas dan la misma importancia a pixeles que pueden estar bastante separados, perdiendo la dimensión e importancia espacial de éstos en la imagen. Para ello, se introduce el concepto de las *redes convolucionales* o *CNNs, Convolutional Neural Networks*, las cuales sí tienen la ventaja de dar una cierta importancia a la estructura espacial de la imagen, y entre otras características, generalmente están compuestas por varias capas ocultas, lo cual se ha probado que son arquitecturas bastante buenas para clasificar imágenes [35]. Las *CNNs* poseen tres componentes básicas que se analizan a continuación [33] :

1. La primera componente básica corresponde a los *campos receptivos locales*. Para ello, es importante pensar en la entrada de la red como un cuadrado de $m \cdot m$ unidades, en vez de las unidades en vertical que se presentaban para las redes neuronales tradicionales (ver Figura 2.1). Dicho esto, los *campos receptivos locales* son zonas de la entrada que son recorridas por un filtro de $n \cdot n$ unidades, el cual va recorriendo la matriz de entrada píxel por píxel[3]y formando cada campo receptivo local; por cada campo, existe un peso y un treshold que se traducirá en formar una unidad en la primera capa oculta, es decir, cada unidad de una capa oculta aprende información de un campo receptivo local específico de la capa de entrada. Esto se puede ver en la Figura 2.4.

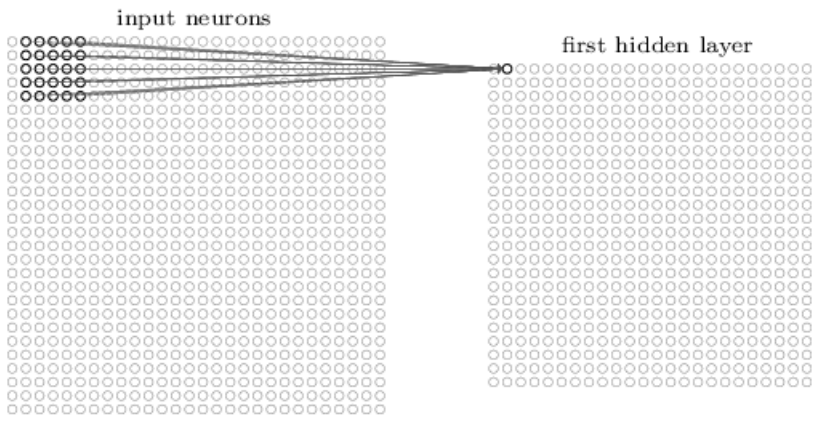


Figura 2.4: En la parte izquierda de la imagen se puede notar la matriz de entrada de la red, además del filtro que forma cada campo receptivo local. Por cada campo, se forma una unidad de la primera capa oculta, la cual está representada en la parte derecha de la imagen. Luego, con un *stride* determinado, se va moviendo el filtro alrededor de la imagen para ir construyendo la primera capa oculta. Imagen de [33].

.

[3]Para las CNN, se puede pensar que cada unidad de la capa de entrada corresponde a un píxel de la imagen de entrada. Existen arquitecturas en que el movimiento del filtro se hace cada dos píxeles, lo cual se dice que la CNN posee un *Stride* de dos.

2. La segunda componente importante corresponde a los pesos y tresholds compartidos. Como se puede ver en la Figura 2.4, cada unidad de la capa oculta posee $n \cdot n$ pesos y un treshold asociado a ella. Lo interesante de la arquitectura de *CNN*, es que para toda la capa oculta se usan los mismos pesos y el mismo treshold. Específicamente, para la unidad *j,k* se tiene que su salida está dada por la Ec. 2.20, en donde σ representa a alguna función de activación que se decida usar, $w_{l,s}$ representa los pesos y $a_{j,k}$ representa la activación de una unidad de la capa de entrada en esa posición.

$$A^1_{j,k} = \sigma(b + \sum_{l=0}^{n}\sum_{s=0}^{n} w_{l,s} a_{j+l,k+s}) \tag{2.20}$$

Esto indica que todas las unidades de la misma capa oculta buscan un cierto *feature*, es decir, una cierta característica específica en distintos lugares de la imagen de entrada. Por esta razón, es que en la literatura se le da el nombre de *feature map* a esta capa. Además, a los pesos y tresholds se les llama *pesos compartidos* y *tresholds compartidos*, respectivemente. Estos pesos, en conjunto con el treshold, definen un *filtro o núcleo (kernel)*. Pero buscar solo una característica en una imagen no es suficiente para hacer reconocimiento ni clasificación, y es por eso que las arquitecturas en general se componen de uno o más *feature maps*, como se puede ver en la Figura 2.5. Cada *feature map* compone una capa convolucional, y esta capa recibe este nombre debido a que la operación hecha en Ec. 2.20 es la convolución matemática.

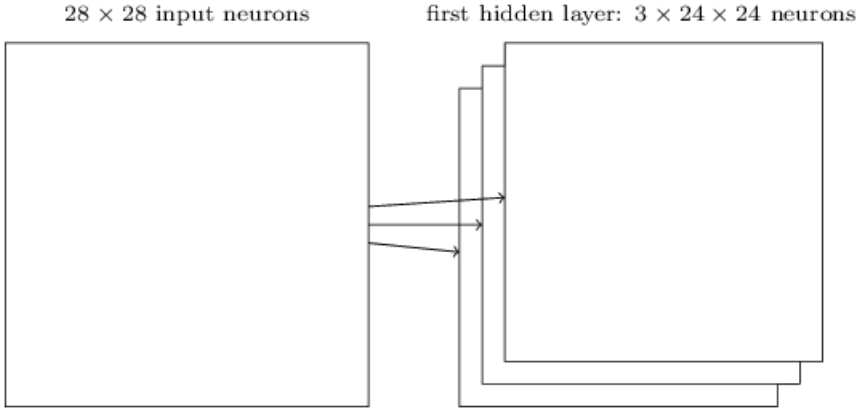


Figura 2.5: En la parte izquierda de la imagen se puede notar la matriz de entrada de la red, que en este ejemplo en particular es de 28 por 28 pixeles. El filtro que se usa es de 5 por 5 con un stride de uno, generando que cada *feature map* sea de 24 por 24 unidades, y con ello se forman tres mapas de características para este ejemplo. Imagen de [33].

.

3. La tercera y última componente de las CNN corresponde a las *capas de pooling*. Luego de cada capa convolucional, se tiene una capa de pooling, la cual tiene la misión de hacer más simple la información de salida de cada capa convolucional. Específicamente, lo que hacen este tipo de capas es que toman la activación de unidades de la capa

convolucional, evalúan una zona de esta capa y a través de operaciones matemáticas se condensa la información. Los tipos de pooling más conocidos se llaman *Max-Pooling* y *Mean-Pooling*, en donde se toma por ejemplo un filtro de $2 \cdot 2$ unidades, se empieza a pasar por la capa de convolución y se elige el máximo valor y el promedio, respectivamente, generando una unidad en la capa de pooling, luego el filtro va moviéndose hasta recorrer toda la capa convolucional. El valor de cada unidad de max-pooling corresponde simplemente al máximo de los valores de activación en el lugar que pasó el filtro, ya que la idea detrás de este tipo de pooling corresponde a preguntarse si se encontró cierta característica, dándole más importancia a si fue encontrado en cierto lugar con respecto a otras características; por otro lado el pooling de mean o de promedio toma en cuenta todas las características de los diferentes *pixeles*.

Finalmente, se puede ver un ejemplo de una red de Deep Learning en la Figura 2.6, en donde se usa la misma idea de una imagen de 28 por 28 pixeles. Es importante notar que la última capa corresponde a una del tipo *fully-connected*, dado que todas las unidades de las capas de pooling se conectan a las de salida.

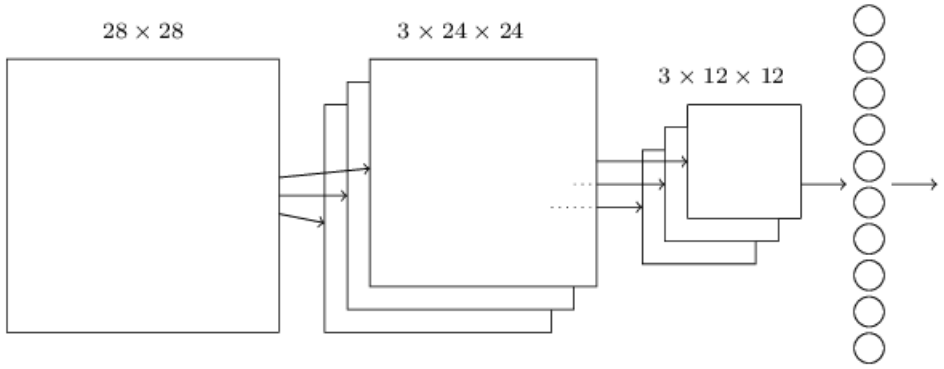


Figura 2.6: En la parte izquierda de la imagen se puede notar la matriz de entrada de la red, que en este ejemplo en particular es de 28 por 28 pixeles. El filtro que se usa es de 5 por 5 con un stride de uno, generando que cada *feature map* sea de 24 por 24 unidades y con ello formando tres mapas de características para este ejemplo. Luego, cada capa de convolución pasa por una capa de pooling con un filtro de 2 por 2, generando una condensación y eliminación de varios parámetros. Finalmente, cada capa de pooling se conecta con la capa de salida de manera *fully-connected* (se dibuja una flecha simplemente por simplicidad), es decir, todas las unidades de las capas de pooling se conectan con las de salida. Este ejemplo en particular es de clasificación de imágenes y posee 10 clases como salida. Imagen de [33].

.

Es necesario mencionar que en la literatura se usa una notación específica para caracterizar los elementos de una CNN; se usan los conceptos de alto (*height* H), ancho (*width* W) y profundidad (*depth* D) caracterizando un volumen para cada capa. Por ejemplo, si la entrada

de una CNN fuera una imagen RGB y esta tuviera un alto y ancho de H_1, W_1 respectivamente, su tensor vendría dado por $H_1 \times W_1 \times 3$. En particular, la profundidad del filtro de convolución a usar debe ser el mismo que el de la entrada, pensando en que cada filtro se relacionará con cada *color* de la imagen de entrada.

2.1.4. Incertidumbre en Deep Learning

En esta sección se presentan definiciones sobre *modelos probabilísticos*, se hace el símil entre *Inferencia Bayesiana* y la técnica de *Deep Learning*, y se presentan varios resultados demostrados por Yarin Gal [36] en su disertación doctoral, los cuales sirven para caracterizar la herramienta de **Deep Learning Bayesiano**, la cual es la técnica elegida para ser analizada como clasificador en este trabajo.

Softmax y tipos de incertidumbre

Dado un conjunto de datos $\mathbf{X} = \{x_j \mid j = 1, \ldots, N\}$ de entrenamiento y su correspondiente conjunto de *etiquetas* o salida $\mathbf{Y} = \{y_j \mid j = 1, \ldots, N\}$, en la clásica forma de regresión se busca una función $y = f_w(x)$ con ciertos parámetros $\mathbf{w}$ que probablemente hayan generado la salida. En general, para problemas de **clasificación**, se puede hablar de la posibilidad entregada por el cálculo de *softmax* para un conjunto de datos de prueba, de acuerdo a la Ec. 2.21, en donde $y = \mathrm{d}$ representa una etiqueta.

$$p(y = \mathrm{d}|x, w) = \frac{\mathrm{e}xp(f^w_{\mathrm{d}}(x)}{\sum_{\mathrm{d}^*} \mathrm{e}xp(f^w_{\mathrm{d}^*}(x))} \tag{2.21}$$

Ahora bien, la salida de softmax da un valor de posibilidad para cada clase, lo cual es diferente a incertidumbre. Esta posibilidad recae en que el punto evaluado sea de una cierta clase con respecto a las demás y por ende, no explica en la confianza del modelo en que el resultado de ese punto corresponda a esa clase.

Un ejemplo de un tipo de incertidumbre es que se entrena una red con imágenes de diferentes animales, para simplificar, de perros y gatos. El detalle es que en el entrenamiento solo se entregaron imágenes de una sola raza de perros. Lo interesante radica en saber qué predice la red al entregar como prueba una imagen de otra raza de perro. Evidentemente esta red tradicional entregará un resultado, mientras que un modelo con incertidumbre entregaría un resultado con un bajo grado en confianza.

Existen varios tipos de incertidumbre, pero las más importantes son *aleatoria* y *epistémico*. La incertidumbre aleatoria radica en medir la confianza sobre la información que el modelo no puede explicar, y es importante cuando existen datos que poseen mayor información ruidosa que otros dentro de la misma base de datos. Por otro lado, la incertidumbre epistémica, o de conocimiento, radica en simplemente medir lo que el modelo no sabe, asociado simplemente a pocos datos de entrenamiento, y es importante porque identifica diferentes situaciones en que el modelo no fue entrenado debido a datos que no estaban en el entrenamiento (como el ejemplo dado anteriormente sobre la predicción de una raza de perro diferente).

Modelaje Bayesiano

A partir del Teorema de Bayes, dado por Teo. 2.2, se puede buscar la distribución $p(w|X,Y)$ *a posteriori* (después de las mediciones, de aquí se saca el nombre de *estimación Bayesiana* dado que se basa en la regla de Bayes) de los parámetros de esta función, en donde $p(w)$ es la distribución a priori (creencia de cómo se distribuyen los pesos antes de tomar en cuenta mediciones), $p(Y|X,w)$ es la verosimilitud o *likelihood* (modelo probabilístico a partir del cual los datos de entrada generaron la salida dados los parámetros), y $p(Y|X)$ es la *evidencia* del modelo definida por la Ec. 2.22. La evidencia del modelo es netamente un promedio de todos los posibles modelos dados por el espacio de parámetros w, y comienza a ser difícil de calcular por métodos numéricos de integración a medida que los modelos se vuelven más complejos [37].

Teorema 2.2 (Teorema de Bayes) *Para el caso particular de regresión Bayesiana, se tiene que*

$$p(w|X,Y) = \frac{p(Y|X,w) * p(w)}{p(Y|X)}$$

$$p(Y|X) = \int p(Y|X,w)p(w)\mathrm{d}w \tag{2.22}$$

Antes de continuar, es necesario dar una breve descripción de una red neuronal artificial bayesiana (**BNN** Bayesian Neural Network), la cual es simplemente una red en donde se *sitúa* una distribución de probabilidad a los pesos, induciendo a la vez una distribución en un conjunto de funciones, ya que ésta distribución en los pesos determina qué funciones se van a generar (Si el lector quiere profundizar, revisar los *Procesos Gaussianos* en [38]).

Inferencia variacional

Dados los problemas de cálculo para la evidencia y por ende de la distribución *a posteriori* real, se necesitan de métodos numéricos que puedan aproximarla. Uno de estos métodos se llama *inferencia variacional*, en donde se realiza optimización y operaciones de derivadas en vez de cálculos de integrales, además de realizar optimizaciones sobre distribuciones en vez de puntos en particular, para el caso de Deep Learning. Para explicar esto, es necesario introducir el concepto de la divergencia de Kullback-Leibler (KL) [39], que en simples palabras es una medida de *similitud* entre dos distribuciones. Para el caso de distribuciones continuas, la definición 2.3 es equivalente cambiando la suma por una integral sobre el espacio muestral.

Definición 2.3 (Divergencia de Kullback-Leibler) *Sean p y q dos distribuciones de probabilidad discretas sobre un espacio muestral* Ω*. Se define la divergencia de Kullbacl-Leibler como*

$$KL(p||q) = \sum_{x\in\Omega} p(x) log \frac{p(x)}{q(x)}$$

con la convención de que $0 \cdot ln\frac{0}{0} = 0,\ 0 \cdot ln\frac{0}{n} = 0$ *y* $0 \cdot ln\frac{n}{0} = \infty$.

Sea entonces $q_\theta(w)$ una distribución, parametrizada por θ, fácil de evaluar que sirva para aproximar lo más posible a la probabilidad *a posteriori*. Para evaluar la similitud se busca minimizar la divergencia de KL, con respecto a θ, entre la aproximación *variacional* y la probabilidad real, la cual viene dada por la Ec. 2.23, siendo $q^*_\theta(w)$ el óptimo de esta minimización.

$$KL(q_\theta(w)||p(w|X,Y)) = \int q_\theta(w) log \frac{q_\theta(w)}{p(w|X,Y)} \mathrm{d}w \tag{2.23}$$

Para hacer un símil entre la minimización de KL y la cota de la evidencia (ELBO, *evidence lower bound*) es necesario recordar la desigualdad de Gibbs (Ec. 2.4), el Teorema de Bayes general (Teorema 2.2) y propiedades básicas de logaritmos. Con estos pasos se puede llegar a la función objetivo de ELBO, dada por Ec. 2.24, lo cual es importante dado que realizar la minimización de KL es equivalente a realizar la maximización de ELBO con respecto a $q_\theta(w)$.

Teorema 2.4 (Desigualdad de Gibbs) *Sean p y q dos distribuciones de probabilidad discretas sobre un espacio muestral* Ω*. La divergencia de Kullbacl-Leibler tiene la siguiente propiedad:*

$$KL(p||q) \geq 0$$

siendo igual a 0 si y sólo sí $p(x) = q(x)$ *para todo* $x \in \Omega$*.*

$$\mathcal{L}_{VI}(\theta) := \int q_\theta(w) \cdot log\ p(Y|X,w) \mathrm{d}w - KL(q_\theta(w)||p(w)) \leq p(Y|X) = \log \text{ de la evidencia} \tag{2.24}$$

La Ec. 2.24, conocida como *inferencia variacional*, será la función objetivo central de la técnica de *Deep Learning Bayesiano* a analizar, en donde maximizar el primer término (esperanza del logaritmo de la verosimilitud) hace que la distribución variacional $q_\theta(w)$ explique bien los datos, mientras que minimizar el segundo término (que en literatura se nota como *KL a priori*) penaliza distribuciones variacionales muy complejas y hace que $q_\theta(w)$ se acerque lo más posible a la probabilidad a priori de los parámetros w.

Dado un punto de prueba x^* y su etiqueta y^*, se puede obtener la distribución de *predicción* dada por Ec. 2.25.

$$p(y^*|x^*, X, Y) = \int p(y^*|x^*, w) q^*_\theta(w) \mathrm{d}w \tag{2.25}$$

Notar que si se cambia el signo de la Ec. 2.24, se convierte en un problema de minimización, donde se puede dividir el problema por la suma de la salida de cada entrada ($f_w(x_\mathrm{i})$, como se puede notar en la función objetivo Ec. 2.26:

$$\mathcal{L}_{VI}(\theta) := -\sum_{\mathrm{i}=1}^{N} \int q_\theta(w) \cdot log\ p(y_\mathrm{i}|f_w(x_\mathrm{i})) \mathrm{d}w + KL(q_\theta(w)||p(w)) \tag{2.26}$$

Se puede ver que este objetivo toma en cuenta todos los puntos para realizar cálculos (problema similar al visto en *Gradient Descent*) cuya solución también se puede dar a través

del uso de *mini-batches*, lo cual induce la función objetivo (Ec. 2.27) del nuevo estimador *insesgado* de la Ec. 2.26, dado por la optimización estocástica en donde el conjunto S es un conjunto en donde se toman M muestras aleatorias del conjunto de entrada :

$$\widehat{\mathcal{L}}_{VI}(\theta) := -\frac{N}{M}\sum_{\mathrm{i}\in S}\int q_\theta(w)\cdot log\ p(y_\mathrm{i}|f_w(x_\mathrm{i}))\mathrm{d}w + KL(q_\theta(w)||p(w)) \tag{2.27}$$

El óptimo (generalmente mínimo local) de esta minimización estocástica es la vez un óptimo de la minimización dada en Ec. 2.26 [40], y por ende es necesario centrarse en cómo calcular las integrales conocidas como las esperanzas de las log-verosimilitudes, las cuales no son fáciles de calcular numéricamente y por ende se necesitan de ciertos mecanismos de aproximación. En particular, se procede a describir la técnica de *Monte Carlo* en conjunto con un estimador *estocástico* útil para el cálculo de las integrales mencionadas para este problema de minimización.

Estimadores de Monte Carlo

Los métodos de ***Monte Carlo*** son técnicas útiles para solucionar problemas analíticamente complejos o de probabilidad desconocida, basado en representaciones de muestras dada una cierta distribución de probabilidad. La idea central de estos métodos radica en representar la distribución a aproximar como un estimador compuesto por muestras aleatorias, y a medida que se va tomando una mayor cantidad de estas, converger a los *momentos* que importan, como la esperanza y varianza (basado en la *Ley de los Grandes Números*).

Del problema de optimización Ec. 2.27, surgen en el cálculo las derivadas de las integrales con respecto a los parámetros θ de la distribución aproximadora elegida $q_\theta(w)$. Luego, se presenta el problema general de estimación de la *derivada de la integral* dada por Ec. 2.28:

$$I(\theta) = \frac{\partial}{\partial\theta}\int f(x)p_\theta(x)\mathrm{d}x \tag{2.28}$$

Existen tres métodos de estimación Monte Carlo para inferencia variacional, los cuales son la *función de puntaje*, *derivada de camino* (o *pathwise derivative*) y el estimador de *función característica*. En este texto sólo se hablará un poco más del segundo estimador, mientras que una explicación más en detalle de los demás estimadores se puede encontrar en [36].

El método de derivada de camino, conocido también como *stochastic backpropagation*, consiste en asumir que $p_\theta(x)$ se puede reparametrizar como $p(\varepsilon)$ a través de una transformación determinística de dos variables $x = g(\theta, \varepsilon)$. Un ejemplo es que si $p_\theta(x)$ es una Gaussiana con parámetros (μ, σ^2), sigue que $g(\theta, \varepsilon) = \mu + \sigma\cdot\varepsilon$. Con esto se tiene su estimador dado por la Ec. 2.29 el cual es insesgado (la esperanza es $I(\theta)$).

$$\widehat{I}(\theta) = f'(g(\theta,\varepsilon))\frac{\partial}{\partial\theta}g(\theta,\varepsilon) \tag{2.29}$$

Para usar esta nueva herramienta en el problema de minimización de *ELBO*, se debe reparametrizar cada fila de la matriz de pesos de cierta capa como $w_{l,\mathrm{i}} = g(\theta_{l,\mathrm{i}}, \varepsilon_{l,\mathrm{i}})$ para una

cierta capa l, lo que en general se escribe como $w = g(\theta, \varepsilon)$ para una cierta distribución $p(\varepsilon)$. Así, reparametrizando cada integral de Ec. 2.27 con respecto a $p(\varepsilon)$ se llega al objetivo de minimización dado por la Ec. 2.30, y al reemplazar el estimador estocástico de las esperanzas de las log verosimilitudes (Ec. 2.29) en este, se llega al nuevo *estimador de Monte Carlo* insesgado (su esperanza con respecto a ε es $\mathcal{L}_{VI}(\theta)$) descrito en Ec. 2.31, el cual cumple que converge al mismo óptimo de $\mathcal{L}_{VI}(\theta)$ [41].

$$\widehat{\mathcal{L}}_{VI}(\theta) := -\frac{N}{M}\sum_{\mathrm{i}\in S}\int p(\varepsilon)\cdot log\ p(y_{\mathrm{i}}|f_{g(\theta,\varepsilon)}(x_{\mathrm{i}}))\mathrm{d}\varepsilon + KL(q_\theta(w)||p(w)) \tag{2.30}$$

$$\widehat{\mathcal{L}}_{MC}(\theta) := -\frac{N}{M}\sum_{\mathrm{i}\in S} log\ p(y_{\mathrm{i}}|f_{g(\theta,\varepsilon)}(x_{\mathrm{i}})) + KL(q_\theta(w)||p(w)) \tag{2.31}$$

Se puede aproximar la distribución de probabilidad *predictiva* (Ec. 2.25) siguiendo un enfoque de *integración Monte Carlo* lo cual viene dado por Ec. 2.32, con $\widehat{w}_t \sim q_\theta(w)$.

$$\begin{aligned}\widetilde{q}_\theta(y^*|x^*) := \frac{1}{T}\sum_{t=1}^{T} p(y^*|x^*,\widehat{w}_t) &\underset{T\to\infty}{\longrightarrow} \int p(y^*|x^*,\widehat{w})q_\theta(w)\mathrm{d}w \\ &\approx \int p(y^*|x^*,w)p(w|X,Y)\mathrm{d}w \\ &= p(y^*|x^*,X,Y)\end{aligned} \tag{2.32}$$

A continuación se realiza la presentación de un resultado importante sobre la relación entre inferencia variacional y el uso de *Dropout*.

Dropout y la condicion de KL

Se puede recordar la Ec. 2.10 que representa la salida de una unidad de cierta capa l de una red neuronal artifical dadas las capas anteriores. En el caso específico de la técnica de *Dropout*, esta se puede reescribir como se indica en la Ec. 2.33 (en donde la segunda parte representa la notación por capa o *vectorial*), en la que se tiene que cada vector $\widehat{\varepsilon^L}$ posee una distribución de Bernoulli, $\odot$ representa el producto de *Hadamard* (Def. 2.1) y para la capa de entrada se tiene la misma metodología, es decir, $\widehat{x} = x \odot \widehat{\varepsilon^1}$. Recordando la definición de esta técnica, cada unidad de cierta capa estará activa para cada ejemplo con una probabilidad $1-p$.

$$\begin{aligned}\widehat{a_j^l} = a_j^l \cdot \widehat{\varepsilon_j^l} = \sigma\Big(\sum_k w_{jk}^l a_j^{l-1} + b_j^l\Big)\cdot\widehat{\varepsilon_j^l} \\ \widehat{a^L} = a^L \odot \widehat{\varepsilon^L}\end{aligned} \tag{2.33}$$

Generalmente la función de optimización (función de costos más métodos de regulación como *L2*) para una red en la que se usa la técnica de *Dropout* viene dada por la Ec. 2.34 para el caso específico de la función de costos de entropía cruzada, la cual en el caso general posee una notación del tipo $\sum_{\mathrm{i}\in S} E_{\widehat{w},b}(x_{\mathrm{i}}, y_{\mathrm{i}})$, en donde E representa una función de costos cualquiera, $\widehat{w}$ representa que hay ciertas unidades de cada capa que no están presentes para cierto input

y por ende estos pesos pueden no estar activos[4], y el conjunto S representa el uso de *mini batches* de tamaño M.

$$\widehat{\mathcal{L}}(w,b) := -\frac{1}{M}\sum_{inputs,j}[y_j(x)ln(\widehat{a_j^l}) + (1-y_j(x))ln(1-\widehat{a_j^l})] + \lambda_1\sum_w w^2 + \lambda_2\sum_b b^2 \quad (2.34)$$

La función de costos general $E_{\widehat{w},b}(x_{\mathrm{i}}, y_{\mathrm{i}})$ puede ser representada como la log verosimilitud negativa más una constante [42], como se puede ver en la Ec. 2.35.

$$E_{\widehat{w},b}(x,y) = const + -\frac{1}{\tau}logp(y|f_{\widehat{w},b}(x)) \quad (2.35)$$

de donde $p(y|f_{\widehat{w},b}(x)) = \mathcal{N}(y; f_{\widehat{w},b}(x), \tau^{-1}\mathbf{I})$ con τ^{-1} el ruido de observación. Sigue que se puede reparametrizar cada peso como $w_{\mathrm{i}} = g(\theta, \widehat{\varepsilon}_{\mathrm{i}})$, implicando en una nueva función objetivo, cuyas derivadas con respecto a los parámetros viene dada por la Ec. 2.36.

$$\frac{\partial}{\partial\theta}\widehat{\mathcal{L}}_{Dropout}(\theta) = -\frac{1}{M}\sum_{\mathrm{i}\in S}\frac{\partial}{\partial\theta}logp(y_{\mathrm{i}}|f_{g(\theta,\widehat{\varepsilon}_{\mathrm{i}})}(x)) + \frac{\partial}{\partial\theta}(\lambda_1\sum_w w^2 + \lambda_2\sum_b b^2) \quad (2.36)$$

A partir de la Ec. 2.36, y eligiendo la distribución a priori sobre los pesos $p(w)$ como distribuciones gaussianas independientes sobre cada peso, se llega a la *condición KL* dada por la primer componente de la Ec. 2.37, la que implica en la identidad de las funciones objetivos entre *Dropout* y *estimación Monte Carlo*.

$$\frac{\partial}{\partial\theta}KL(q_\theta(w)||p(w)) = \frac{\partial}{\partial\theta}N\tau(\lambda_1\sum_w w^2 + \lambda_2\sum_b b^2) \Rightarrow \frac{\partial}{\partial\theta}\widehat{\mathcal{L}}_{Dropout}(\theta) = \frac{1}{N\tau}\frac{\partial}{\partial\theta}\widehat{\mathcal{L}}_{MC}(\theta) \quad (2.37)$$

Esta condición es sumamente importante pues indica que realizar el entrenamiento de una red que usa *Dropout*, con su enfoque probabilístico, es equivalente a realizar el entrenamiento de una red neuronal artificial Bayesiana. Más específicamente, la distribución a priori puede escribirse como $p(w) = \prod_{\mathrm{i}=1}^{L}\mathcal{N}(0, \frac{1}{l_{\mathrm{i}}^2}\mathbf{I})$, es decir una distribución Gaussiana para cada matriz de pesos entre capas L y L-1, en donde l_{i}^2 representa el largo a priori, el cual viene dado por la siguiente propiedad Ec. 2.38, la cual indica que para valores pequeños de escala de largo l_{i} y pequeños de varianza τ^{-1} genera valores pequeños de los términos de regulación λ implicando en que pueden generalizar mal en *testing*, mientras que para valores más grandes de escala de largo l_{i} y de varianza implica en un alto valor en el término de regulación, penalizando más los pesos. Además, se demuestra que la unidad de medida de los pesos es idéntica a esta escala de largo, implicando que tener una escala de largo l_{i}^2 implica a multiplicar las entradas por l_{i} con una distribución de probabilidad a priori $\mathcal{N}(0, \mathcal{I})$ sobre los pesos.

$$\lambda_{\mathrm{i}} = \frac{l_{\mathrm{i}}^2(1-p_{\mathrm{i}})}{2N\tau} \quad (2.38)$$

[4]Técnicamente, en la literatura, se presentan los pesos de manera matricial, y a partir de una transformación del espacio de unidades hacia el espacio de parámetros, estas matrices se convierten en una *observación* o *realización* de una variable aleatoria, pues el vector de probabilidades se transforma en una matriz diagonal que permite generar esta transformación. Para más detalle, ver [36], sección 3.3.2 (página 41).

MC Dropout

A continuación se introduce el estimador *MC Dropout*, con el cual es posible obtener incertidumbre a partir de un modelo de redes neuronales artificiales bayesianas, y como se demostró anteriormente, en redes neuronales artificiales entrenadas con la técnica de *Dropout*.

Para ello, se empieza por ver la esperanza y varianza (primeros dos momentos) de este estimador. Recordando que la distribución predictiva aproximada es

$$q_\theta^*(y^*|x^*) = \int p(y^*|f_w(x*))q_\theta^*(w)\mathrm{d}w$$

en donde $q_\theta^*(w)$ es el óptimo de la función objetivo Ec. 2.31, se tienen las siguientes dos proposiciones demostradas en [36], tomados T términos y siguiendo la *integración Monte Carlo*.

Proposición 2.5 *Dada* $p(y^*|f_w(x^*)) = \mathcal{N}(y^*; f_w(x^*), \tau^{-1}\mathbf{I})$ *con algún* $\tau > 0$*, se puede estimar la esperanza* $\mathbf{E}_{q_\theta^*(y^*|x^*)}[y^*]$ *con el estimador insesgado*

$$\widetilde{\mathbf{E}}[y^*] := \frac{1}{T}\sum_{t=1}^{T} f_{\widehat{w_t}}(x^*) \underset{T\to\infty}{\longrightarrow} \mathbf{E}_{q_\theta^*(y^*|x^*)}[y^*]$$

con $\widehat{w}_t \sim q_\theta^*(w)$*.*

Proposición 2.6 *Dada* $p(y^*|f_w(x^*)) = \mathcal{N}(y^*; f_w(x^*), \tau^{-1}\mathbf{I})$ *con algún* $\tau > 0$*, se puede estimar la esperanza* $\mathbf{E}_{q_\theta^*(y^*|x^*)}[(y^*)^T(y^*)]$ *con el estimador insesgado*

$$\widetilde{\mathbf{E}}[(y^*)^T(y^*)] := \tau^{-1}\mathbf{I} + \frac{1}{T}\sum_{t=1}^{T} f_{\widehat{w_t}}(x^*)^T f_{\widehat{w_t}}(x^*) \underset{T\to\infty}{\longrightarrow} \mathbf{E}_{q_\theta^*(y^*|x^*)}[(y^*)^T(y^*)]$$

con $\widehat{w}_t \sim q_\theta^*(w)$ *e* y^**,* $f_{\widehat{w_t}}(x^*)$ *vectores fila.*

A partir de estas dos proposiciones, se pueden obtener la esperanza y la varianza predictiva del modelo a partir de estimadores insesgados, caracterizados en la Ec. 2.39.

$$\begin{aligned} \widetilde{\mathbf{E}}[y^*] :&= \frac{1}{T}\sum_{t=1}^{T} f_{\widehat{w_t}}(x^*) \\ \widetilde{Var}[(y^*)] :&= \tau^{-1}\mathbf{I} + \frac{1}{T}\sum_{t=1}^{T} f_{\widehat{w_t}}(x^*)^T f_{\widehat{w_t}}(x^*) - \widetilde{\mathbf{E}}[(y^*)]^T\widetilde{\mathbf{E}}[(y^*)] \end{aligned} \tag{2.39}$$

CNN Bayesiana

La técnica propuesta funciona de la siguiente manera: entrenar la red con *Dropout* con un enfoque probabilístico, luego realizar T evaluaciones estocásticas (es decir, realizar T veces *forward propagation*, explicado anteriormente) para luego obtener incertidumbre en

la clasificación. ¿Cómo se puede extender esto para la técnica de *Deep Learning* dadas las imágenes de señales de diferentes zonas del cerebro, que son las que se desean usar?

Para responder esta pregunta, es necesario ver cómo se ha aplicado hasta ahora dropout en Deep Learning. Generalmente en la literatura se habla de que se aplica al final del modelo, antes de la capa *fully connected*. En este caso, *Dropout* será usado luego de cada capa convolucional, es decir antes de la capa de *pooling*, lo que se logra a través de una manipulación de cada kernel de cada capa convolucional, formando matrices a las cuales se les da una distribución a priori o inicial, y luego estas se multiplican por una matriz diagonal que en cada componente posea una probabilidad sacada de una distribución de Bernoulli. Es decir, a través de la técnica de *MC Dropout*, se podrá generar un aproximado de la distribución predictiva, a través de Ec. 2.32 y Ec. 2.39.

2.1.5. Métricas de evaluación de un clasificador

Para aprendizaje supervisado en el área de aprendizaje de máquinas existen diferentes métricas de evaluación para un clasificador dado, las cuales se basan en una *matriz de confusión*, que muestra las clases evaluadas tanto de manera correcta e incorrecta. En particular, en la Tabla 2.1 se muestra una matriz de confusión para el caso binario de clasificación de 1 o 0 (puede verse también como de 'Sí' y 'No'), en donde se presentan las cuatro salidas posibles dada la clasificación y el etiquetado original:

- TP: (o Verdaderos Positivos), corresponde al caso en que la clase del punto original es 1 y la predicción es también un 1.
- TN: (o Verdaderos Negativos), corresponde al caso en que la clase del punto original es 0 y la predicción es también un 0.
- FP: (o Falsos Positivos), corresponde al caso en que la clase del punto original es 0 y la predicción es de 1.
- FN: (o Falsos Negativos), corresponde al caso en que la clase del punto original es 1 y la predicción es de 0.

Tabla 2.1: Matriz de Confusión para un caso binario general.

Predicción\Etiqueta	Etiqueta = Sí	Etiqueta = No
Predicción = Sí	***TP***	***FP***
Predicción =No	***FN***	***TN***

La medida más usada se conoce como *exactitud* y corresponde al porcentaje de clasificaciones correctas, dado por la Ec. 2.40. Es una medida a usar cuando las clases están balanceadas en una base de datos.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.40}$$

Otras medidas que se usan cuando las clases no están balanceadas, es decir cuando existen más datos de una clase que otra en la base de datos, corresponden a la *sensibilidad*, *especificidad*, *precisión* y *F1 Score*. La sensibilidad o la *tasa de verdaderos positivos* o *Recall*, dada por Ec. 2.41, corresponde al número de elementos correctamente clasificados como positivos dado el total de positivos en el conjunto de etiquetas, y ayuda para saber cuántos elementos de esta clase fueron identificados correctamente.

$$Recall = \frac{TP}{TP + FN} \tag{2.41}$$

La especificidad, dada por Ec. 2.42, es el equivalente de la sensibilidad para la clase negativa y también se conoce como la *tasa de verdaderos negativos*. Por otro lado, la precisión, descrita por Ec. 2.43, corresponde al número de elementos correctamente clasificados como positivos dado el total de elementos *clasificados como positivos*, la cual ayuda para saber la proporción de elementos que fueron clasificados como positivos y que realmente lo eran.

$$Especificidad = \frac{TN}{TN + FP} \tag{2.42}$$

$$Precisión = \frac{TP}{TP + FP} \tag{2.43}$$

Cuando se desea comparar diferentes modelos probados sobre la misma base de datos, en donde ésta se encuentra desbalanceada, generalmente se usan las métricas de sensibilidad y de precisión. El detalle de esto es que no es directo, puede ser que un modelo tenga mayor precisión pero menor sensibilidad que el otro, por ende, ¿Cómo realizar esta comparación?. Para ello se tiene la métrica de *F1-Score*, la cual es la media armónica del recall y de la precisión, y es simplemente una medida estadística de exactitud del modelo cuando se tiene el detalle de que las clases se encuentran balanceadas. En particular, en la Ec. 2.44, se caracteriza matemáticamente al *F-Score*, el cual se encuentra equilibrado cuando $\beta = 1$ (caso que corresponde a F1-Score) mientras que para $\beta > 1$ favorece a la precisión, y a la sensibilidad evidentemente cuando $\beta < 1$ [43].

$$F1\text{-}Score = \frac{(\beta^2 + 1) * precision * recall}{(\beta^2 * precision) + recall} \tag{2.44}$$

2.1.6. Transformadas Tiempo-Frecuencia

Actualmente los métodos de análisis espectral se utilizan con frecuencia en estudios electrofisiológicos en modelos animales y seres humanos. Si bien existen diferentes métodos, en esta sección se habla de dos y de cuándo se recomienda usarlos. En particular estos dos métodos implican en el uso de la Transformada de Fourier, es decir, descomponen la señal en funciones sinusoidales en que, a medida que cambie la frecuencia, cambian la amplitud y la fase. La diferencia entre los métodos radica en el uso de distintas funciones.

El análisis tradicional de Fourier radica en realizar una convolución [5]entre la señal temporal y una sinusoide, realizando un análisis a través de pequeñas ventanas para mejorar la especificidad temporal. En particular, en computación se realiza la FFT (*Fast Fourier Transform* o *Transformada rápida de Fourier*), que es una implementación de la DFT (*Transformada discreta de Fourier*), que es un algoritmo de bajo costo computacional. El problema de este análisis tradicional radica en que está hecho solo para funciones estacionarias, es decir que su promedio no cambia significativamente en el tiempo, además de tener una resolución tiempo-frecuencia baja y utilidad limitada a una banda de frecuencia específica a la ventana elegida [44].

El primer método que sirve como alternativa es el de análisis de ***wavelets***, que viene dado por una función elegida y conocida como la *wavelet madre*, en donde la señal temporal a analizar se convoluciona con *wavelets hijas* las cuales son versiones desplazadas en tiempo y proporcionales en amplitud con respecto a la wavelet madre. Una diferencia importante con respecto al análisis de Fourier corresponde a que el tamaño de la ventana a elegir depende de la frecuencia, implicando en una mayor precision temporal para altas frecuencias. Además, la transformada de wavelet tiene una mejor resolución tiempo-frecuencia con respecto al análisis de Fourier, haciendo que sean útiles para el análisis de señales no estacionarias [45].

La wavelet madre de tipo Morlet es la que se usa de manera común en el análisis de EEG [46] dada su forma de sinusoide multiplicada por una gaussiana, implicando en que puede estrechase en los extremos, generando la capacidad de tener mayor éxito cuando se quieren extraer valores de la señal temporal. En particular, y para evitar discontinuidades en el espectro, se usa la transformada de Morlet compleja [47], la cual puede notarse en la Figura 2.7.

[5]La operación de convolución mide el traslape de dos funciones, integrando sobre todos los posibles desplazamientos al mover una función por sobre la otra.

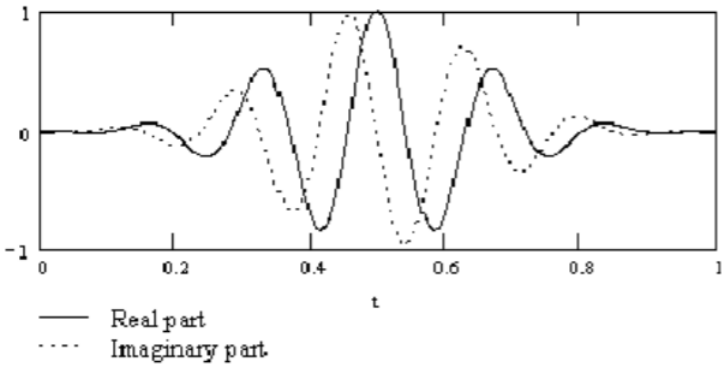


Figura 2.7: Esta figura muestra una señal wavelet de tipo *Morlet compleja*, en donde la señal con línea punteada representa a la parte imaginaria mientras que la parte real está dada por la línea contínua. Imagen de [47].

Por otro lado, el segundo método que sirve como análisis espectral corresponde al de ***multitapers***, en el que se realiza un promedio de una cierta cantidad independiente de estimaciones espectrales de Fourier, generando que este método sea útil para procesos con variaciones rápidas. El hecho de que las estimaciones espectrales sean independientes radica en que antes de calcular cada una de estas, se multiplica la señal temporal por un *taper* o ventana, y se impone que los diferentes tapers sean ortogonales. La elección de la cantidad de tapers implica un tradeoff entre la varianza y el sesgo del estimador de multitaper, y si existe un mayor sesgo, existe mayor pérdida de datos entre frecuencias en el espectrograma [48].

Una diferencia entre wavelets y el análisis de multitapers es que, en el método de wavelets, la señal es convolucionada con la función base (wavelet hija correspondiente a cierta frecuencia) mientras que en el caso de multitapers la señal temporal es separada por ventanas con cada taper y luego se le realiza una transformada de Fourier, es decir, el análisis de multitaper radica en aplicar FFT varias veces con diferentes funciones base (tapers).

Otra diferencia entre estas dos técnicas radica en que la especificidad de tiempo y de frecuencia crece para el caso de wavelets, lo cual no pasa para multitapers. De hecho, para señales con amplitudes más bajas la transformada continua de wavelet es mejor, mientras que para una mejor especificidad en la frecuencia se suelen utilizar más los multitapers [49].

2.2. Estado del Arte

Dadas las técnicas de tratamiento de Parkinson ya expuestas en el capítulo de Introducción, se procede ahora a realizar un pequeño resumen de las técnicas actualmente usadas para enfrentar el problema de circuito cerrado.

Meidahl et al (2017, [30]) realizaron una revisión sobre lazos de control para circuitos cerrados que respondan a *amplitud* y *fase* de una señal de entrada (es decir, circuitos *adaptativos* que responden en base a un marcador, en este caso amplitud y/o fase respectivamente) para tratamientos de DBS, lo cual se puede observar en la Figura 2.8. En esta revisión, se analizan distintos estudios hechos a la fecha con respecto a circuitos cerrados adaptativos a fase y a amplitud, además de analizar las limitaciones de estos nuevos tratamientos, como el hecho de usar funciones de lazo de control tan simples que generan que el sistema pueda estar obviando ciertos parámetros o simplemente no considerarlos, recomendando que para investigaciones futuras se pueda usar una técnica de ingeniería llamada control PID (proporcional, integral y derivativo) [50], el cual ha tenido bastante éxito y está bien establecido. Además, llaman la atención e invitan a tener cuidado al considerar propuestas más demandantes en computación debido al alto consumo de recursos y batería.

En particular, un ejemplo de investigación sobre DBS con control de fase recae en un paper publicado por Rosin et al. (2011, [51]) en el cual demuestran que esta técnica funciona mejor frente a la de circuito abierto, en donde la alarma es un potencial de acción de neurona en el Globo Pálido Interno o en la corteza motora. Por otro lado, Pais Viera et al. (2016, [52]) fueron los primeros en probar el tratamiento de circuito cerrado con el tratamiento de estimulación epidural (llamado DCS en el paper, *Dorsal Column Stimulation*) para el tratamiento de epilepsia, demostrando disminuir el número de episodios de ataques (*seizures*) además de su duración, en conjunto con cambiar el patrón de la señal de LFP con el tiempo.

Nurse et al (2016, [53]) realizan un análisis de clasificación de LFP usando una técnica clásica de Deep Learning, para luego combinarlo en un nuevo chip llamado *TrueNorth* que posee características ideales para ser usado para análisis real de datos de actividad cerebral. Para realizar el etiquetado de su base de datos, se tuvo que realizar la medición de EEG en conjunto con EMG (electromiografía) por aproximadamente treinta minutos. Para generar las clases, definen movimiento a partir de operaciones matemáticas luego de manejar los datos del EMG a través de *sliding windows* o ventanas móviles, y cuando se pasaba un cierto *umbral* o *threshold*, se asignaba movimiento para dicha ventana. Se formaron imágenes como entradas para la red, de acuerdo a cierta cantidad de puntos y a la cantidad de canales, obteniendo un 81 % de exactitud en la clasificación en el conjunto de validación, pero esto tiene un detalle. Al ver las Figuras de `loss` o pérdidas a partir de salida de la función de costos en función de las épocas, se puede notar que la pérdida en el conjunto de validación es bastante mayor a la pérdida en el entrenamiento, lo cual es una clara muestra de sobre-entrenamiento u *overfitting* generando que la red en cuestión tiene un pobre rendimiento al generalizar para datos que no estuvieron presentes durante el entrenamiento ni la validación, lo cual genera que el porcentaje de exactitud no sea tan creíble debido a que la red está sobre-entrenada.

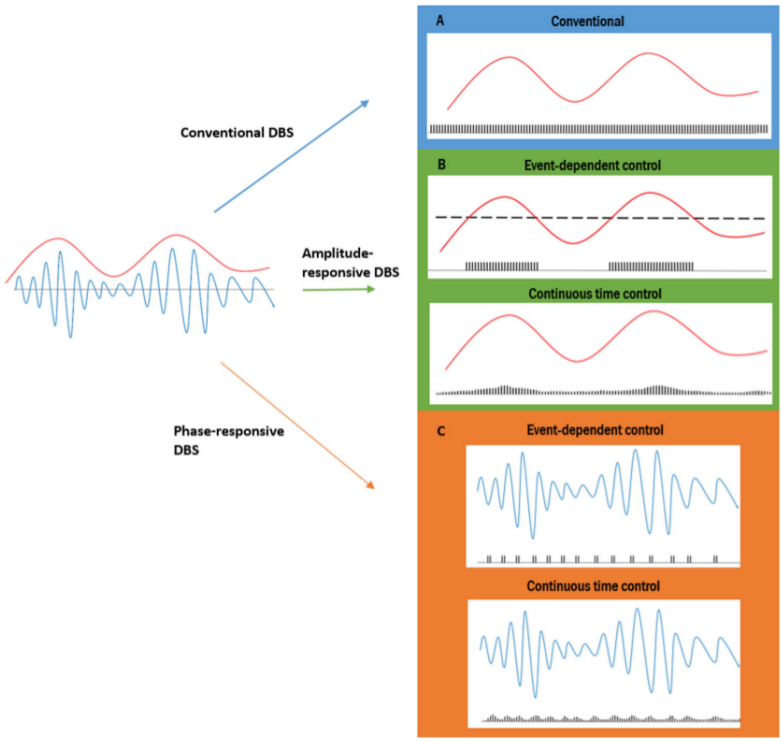


Figura 2.8: El cuadro A muestra DBS funcionando con circuito abierto. El cuadro B corresponde a DBS con alarma dada la amplitud de la señal, en donde el primero corresponde a enviar pulsos dado un evento (pasar un treshold por ejemplo) mientras que el segundo corresponde a una estimulación proportional a la amplitud. El cuadro corresponde a DBS con alarma dada la fase, teniendo estimulación dados eventos o siendo proporcionales a la fase. Imagen de [30].

Da Silva et al (2018, [32]) realizan experimentos con ratones sin *recompensa* o premios ni estímulos externos en un campo abierto, y además investigan la actividad de neuronas dopaminérgicas de la sustancia nigra pars compacta en relación a la iniciación del movimiento, entre otros experimentos. A partir de estas pruebas, muestran que la actividad de neuronas dopaminérgicas antes del inicio de movimiento modula el *vigor* o intensidad de movimiento futuro, implicando en la posibilidad de explicar por qué pacientes que sufren la enfermedad de Parkinson seleccionan movimientos más débiles para empezar a moverse. Esto motiva a realizar investigaciones futuras enfocadas en un circuito cerrado basado en la entrega de estimulaciones transientes en el circuito basal tálamo-cortical usando un lazo de control relacionado con la iniciación de movimiento, enfocándose en áreas cerebrales relacionadas a la planificación de movimiento.

Capítulo 3

Metodología

En este capítulo se presentan los diferentes mecanismos para generar una base de datos y etiquetas a partir del procesamiento de señales electrofisiológicas y video de un modelo de rata de 6-OHDA parkinsoniano, para luego presentar la arquitectura diseñada para las técnicas de *Deep Learning* y *Deep Learning Bayesiano*, con el fin de generar etiquetas de existencia de movimiento a partir del video e imágenes a partir de las señales cerebrales.

3.1. Generación de base de datos

Las señales cerebrales con las que se trabaja en este Trabajo de Título son adquiridas a partir de una rata Sprague Dawley, a la cual se le somete una lesión en ambos hemisferios con la neurotoxina de 6-OHDA (explicada anteriormente en la sección 1.1) para poder generar el modelo animal parkinsoniano. Para obtener datos cerebrales del animal, es necesario el uso de electrodos implantados. El arreglo a usar consta de cincuenta y dos electrodos, en particular 18 electrodos en la CFA (*Caudal Forelimb Area*, parte de la corteza motora primaria), 14 en el estriado o STR, 12 en el área parafascicular o Pf y de 8 electrodos en el area VPL (*tálamo ventral posterolateral*).

El *setup* de los experimentos de *open field* se puede notar en la Fig. 3.1. El hardware para recopilar los datos de las señales electrofisiológicas es el módulo de registro de OpenEphys [54], al cual se le conecta la salida de los electrodos y además se conecta a un computador. A través de su software se pueden grabar por canal estas señales en el computador, las cuales son amplificadas por un factor de $\frac{1}{0{,}195}$, filtradas de forma analógica entre 0.1-15.000Hz y con una resolución de 30 mil muestras por segundo (30kS/s).

Por otro lado, mientras se recopilan los datos cerebrales, también se puede grabar el comportamiento del animal a través de una cámara de vídeo FireWire Flea2, que en estos experimentos usa una resolución de 15 cuadros por segundo (fps). Además de grabar el comportamiento de la rata, se graba una luz LED que se enciende cada un segundo de acuerdo a un protocolo programado en un Arduino UNO, el cual sirve para sincronizar el registro del vídeo y de las señales electrofisiológicas.

Se realiza una sola grabación del animal sin usar estimulación eléctrica externa, es decir, solo se ve su comportamiento de movimiento y de señales cerebrales en un *open field*. Este registro de vídeo posee una duración de alrededor de cuarenta y tres minutos, lo cual es importante debido a la necesidad de producir una gran cantidad de datos para el uso apropiado de los algoritmos de clasificación.

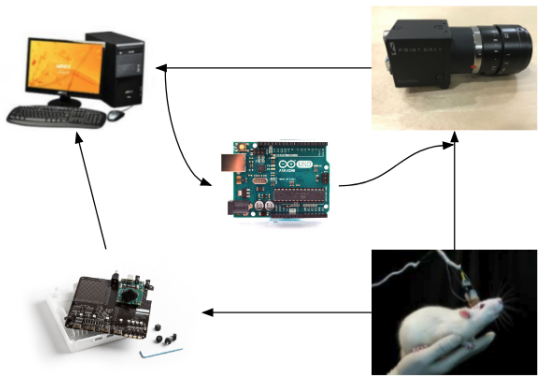


Figura 3.1: Setup general para la obtención de datos. Consta de una cámara de vídeo FireWire Flea2, la cual graba el comportamiento en la caja del animal; un módulo de registro de datos OpenEphys, que recibe la señal electrofisiológica; electrodos, una rata Sprague Dawley y un Arduino Uno, para la sincronización temporal entre el vídeo y la señal del animal. Imagen creada por el autor.

3.1.1. Sincronización entre etiquetas e imágenes

Como ya se ha mencionado a lo largo de esta sección, se desean obtener etiquetas de movimiento a partir de una grabación de vídeo, e imagénes a partir de una señal electrofisiológica. Ahora bien, ¿cómo se pueden sincronizar el vídeo y la señal temporal?¿cómo asegurarse de que la etiqueta extraída de movimiento corresponde efectivamente a la imagen extraída de la señal en cuestión?

Para responder estas inquietudes, basta con insertar un arduino en la grabación de movimiento, como puede notarse en la Fig. 3.1. Este sirve netamente para solucionar el problema en cuestión, dado que el arduino se programa de tal manera de que prenda la luz de un LED cada un segundo, implicando que el tiempo cero para la generación de etiquetas y de imágenes corresponde al instante en que el LED se prende por primera vez.

En particular, encontrar este instante de tiempo para el vídeo es trivial dado que la luz puede notarse en este, mientras que para la señal electrofisiológica entra en acción el programa *OpenEphys*, el cual mide tanto los canales auxiliares (arduino, estimulador eléctrico, entre otros) y las señales medidas a partir de los electrodos.

Así, la señal de arduino puede notarse en la Fig. 3.2, en donde en el cuadro rojo se muestra el pulso perteneciente a la primera emisión de luz por parte de LED, generando que ese tiempo específico sea el tiempo de origen para la generación de imágenes a partir de las señales cerebrales, mientras que en el cuadro negro son todas las señales que el arduino emite al compilar el protocolo, el cual el programa de OpenEphys recibe de todas maneras. Esto es importante pues se tiene que comprobar el ancho y período de estos pulsos con los siguientes para encontrar el pulso que corresponde al primer haz de luz. Luego, con ambos tiempos iniciales (de vídeo y de señal) se pueden realizar los cálculos de etiquetado y de generación de imágenes sincronizados en el tiempo.

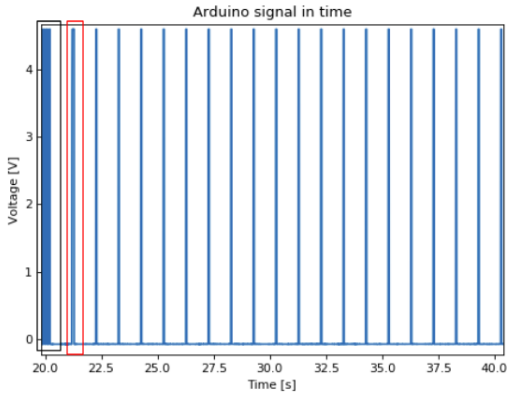


Figura 3.2: Señal de salida de Arduino Uno para sincronizar el vídeo y la señal electrofisiológica. En la caja negra se pueden notar todos los pulsos que emite el arduino al compilar el protocolo, mientras que encerrado en la caja roja puede notarse el pulso que corresponde al primer encendido del LED. Imagen creada por el autor.

3.1.2. Trayectoria del animal

Dado que la cámara posee una resolución de 15 frames por segundo, se puede calcular el centro de gravedad de cada frame generando 15 puntos por segundo. Es decir, a partir de cierto umbral (siempre hablando de valor de pixel) se genera un algoritmo que simplemente indica que todo lo que es mayor a ese valor es el animal, mientras que lo que esté por debajo es la caja de campo abierto. Luego, se genera una *referenciación* de cada valor del frame con este umbral, es decir se toma el valor absoluto de la resta entre estos, y a partir de este valor se calcula el centro de gravedad de cada imagen del video. De esta manera uno se asegura de estar tomando el centro de masa del animal.

Para el animal inyectado con la neurotoxina, se puede apreciar al lado izquierdo en la Fig. 3.3 el área de la imagen a utilizar, mientras que a la derecha se muestran los valores de la imagen que están por sobre un cierto umbral, de valor de intensidad de pixel de 200.

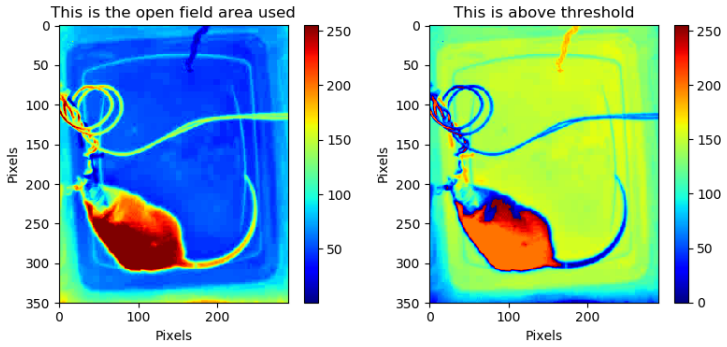


Figura 3.3: Para el animal *parkinsoniano*, a la izquierda se puede notar el área de la imagen a utilizar, mientras que a la derecha se puede apreciar que por sobre el umbral de 200 está casi todo el cuerpo del animal. Imagen creada por el autor.

Por otro lado, en la Fig. 3.4 se puede notar la trayectoria del centro de masa del animal a lo largo del video, la cual fue filtrada para eliminar la mayor cantidad de *outliers* que representan valores imposibles de movimiento, que significan en velocidades que el animal no puede alcanzar [55].

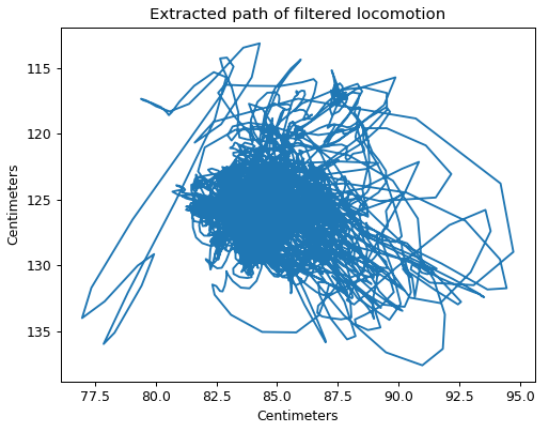


Figura 3.4: Locomoción del animal a lo largo del vídeo, luego de tomar el centro de masa por cada frame existente en este. Imagen creada por el autor.

3.1.3. Velocidad y umbrales de movimiento

A partir de la trayectoria, para cada eje se puede tomar la diferencia entre dos frames (es decir, calcular la derivada con respecto a un frame) para obtener la velocidad con respecto a ese eje. Como se forma un vector, se toma la norma de este para extraer la velocidad del animal a lo largo del vídeo.

Luego, por inspección visual (bastan cinco minutos para poder notar ciertos cambios) se va mirando esta curva y además el vídeo, para ir notando cuándo el animal se desplaza y cuándo es que posee *movimientos locales*, es decir que se mantiene en la misma posición en el campo pero aún así realiza cierto movimiento. De esta inspección visual, se impone un umbral para desplazamiento y movimientos locales, respectivamente; es decir, si se sobrepasa el umbral de movimiento local significa que el animal dejó de estar quieto para mover alguna extremidad o la cabeza, mientras que cuando se pasa el umbral de desplazamiento es porque el animal cambió su posición con respecto y se desplazó.

Un extracto de la curva de velocidad (en azul) y de estos umbrales puede apreciarse en la Fig. 3.5, en donde el valor para movimientos locales está representado en rojo y el umbral de desplazamiento está representado en negro.

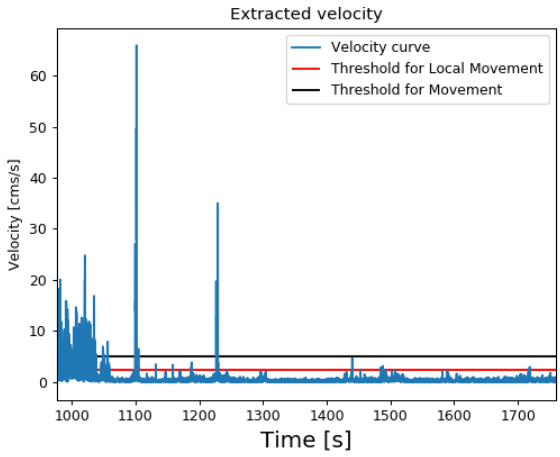


Figura 3.5: Locomoción del animal a lo largo del vídeo, luego de tomar el centro de masa por cada frame existente en este. Imagen creada por el autor.

3.1.4. Etiquetado de la base de datos

Para generar las etiquetas se usa el método de *sliding window*. Este método se basa en simplemente utilizar una ventana móvil de un tiempo específico y con un traslape dado; en particular en la Fig. 3.6 se puede apreciar una ventana de un segundo con un traslape de 333 ms. Es importante además notar que se generan nuevas bases de datos con ventanas de medio segundo y dos segundos de duración, con un traslape de 167 y 666 ms respectivamente. Los cálculos a realizar dentro de cada ventana son simples; si tres puntos seguidos dentro de

esta son mayores al umbral de movimiento local entonces se asigna el tag de movimiento a toda la ventana. La razón de esto radica en capturar los cambios en frecuencia al producirse algún movimiento, y se toman puntos seguidos para desechar aquellos puntos *outliers* o de ruido que son mayores al umbral pero no poseen continuidad en el tiempo en la señal real, es decir no existen puntos (frames) vecinos que también sean mayores.

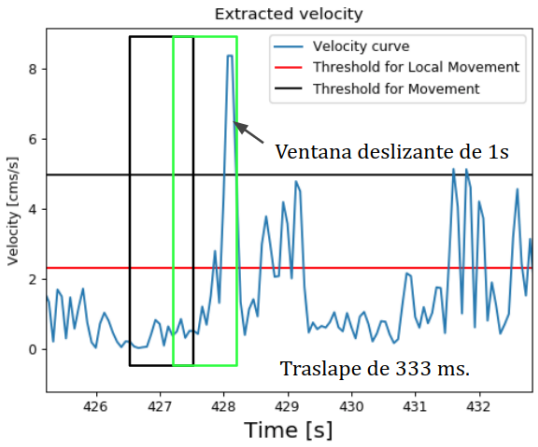


Figura 3.6: Uso de una ventana móvil de un segundo con un traslape de 333 ms para el etiquetado de la base de datos. Para ejemplicar este algoritmo, se presentan dos ventanas traslapadas con la intención de mostrar de que si existe un movimiento al final de una ventana y al comienzo de otra, esto se puede notar a través del traslape, lo se puede ver entre la ventana verde y la siguiente. Imagen creada por el autor.

3.1.5. Datos electrofisiológicos e imágenes

Como ya se mencionó al inicio de esta sección, las señales electrofisiológicas son almacenadas con una resolución de 30 mil datos(muestras) por segundo. Evidentemente, esto no es óptimo para realizar cálculos computacionales, por lo que se debe generar un *downsampling* para estas señales. Para evitar efectos de aliasing en la nueva señal, antes se debe realizar un filtrado pasa bajo a una frecuencia de corte menor que la mitad de la nueva de frecuencia de muestreo (para cumplir el teorema de Nyquist), lo cual se conoce como *decimación en frecuencia* [56]. En este caso, se desea realizar un downsampling a 1 kSample/sec, lo cual implica que la frecuencia de corte del filtro pasa bajo debe ser menor a 500 Hz; en este caso se elige de 450 Hz.

Una vez hecha la decimación en frecuencia, se procede a eliminar canales ruidosos y a realizar una referenciación ya que la medida de cada electrodo solo da información dada la diferencia de potencial entre dos puntos del cráneo. Por ende se necesita de una medida que sirva como referencia para poder generar esta diferencia y que las señales de cada canal sean estricamente locales (válidas).

Para eliminar canales ruidosos, además de realizar inspección visual, se puede generar la distribución de la derivada de cada canal, y aquellos que no presenten una forma más gaussiana, como aquellos que presenten más de una moda, son canales que presentan artefactos [57]. Por otro lado, para realizar la referenciación, esta se realiza por área, es decir se realiza un *z-score* luego de calcular la media y desviación estándar de todos los canales que estén presentes en un área, teniendo la restricción de no combinar hemisferios.

Luego de realizar la elimiación de canales ruidosos y la referenciación, se procede a utilizar un último filtro pasa bandas entre 3 y 100 Hz, enfocando las frecuencias a utilizar en las bandas beta y gamma. Para resumir estos pasos de pre-procesamiento, se puede ver el esquema de la Fig. 3.7.

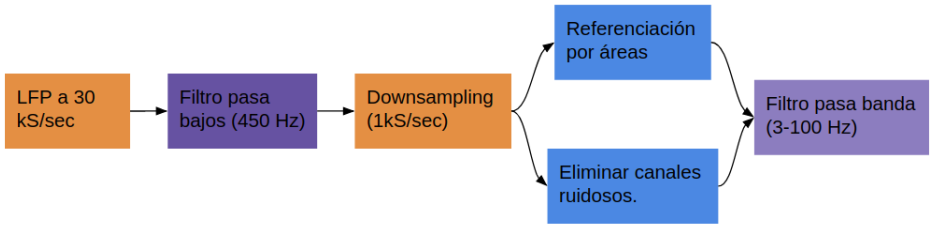
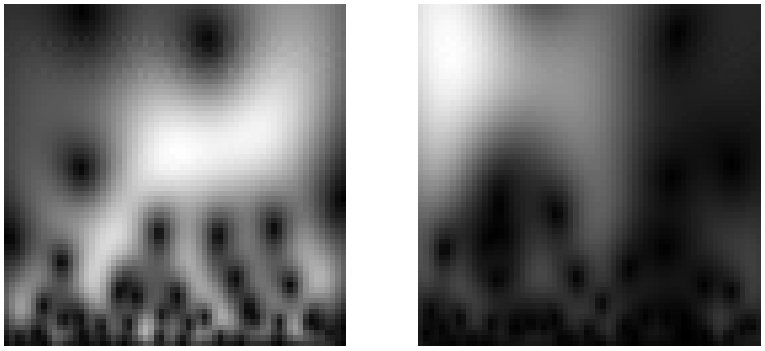


Figura 3.7: Metodología propuesta para el preprocesamiento de las señales fisiológicas. Imagen creada por el autor.

Para generar las imágenes por canal, también se usa el algoritmo de sliding window, pero esta vez por cada ventana se calcula la transformada continua de Wavelet y se guarda la imagen correspondiente. Es decir, por cada canal se realiza el algoritmo, y para imágenes que estén sincronizadas en tiempo se les asigna la misma etiqueta. Cada imagen es de 64x64 pixeles, y en la generación de estas se ocupa una wavelet de tipo Morlet compleja. En la Fig. 3.8 se puede ver un ejemplo de cada clase, es decir, de movimiento y no movimiento respectivamente.



(a) Ejemplo de Movimiento (b) Ejemplo de No Movimiento

Figura 3.8: Ejemplo de imágenes correspondientes a cada clase de movimiento, en particular en (a) se puede ver una imagen con etiqueta de movimiento, mientras que en (b) puede verse una de no movimiento. Imagen creada por el autor.

Por otro lado, en la Tabla 3.1 se puede ver la cantidad de datos disponibles en cada base de datos a partir del tamaño específico de ventana y, además, cada base de datos es dividida en entrenamiento (en donde la red aprende), validación (en donde se ve cómo la red va generalizando durante el entrenamiento) y prueba o test, en porcentajes de 60, 20 y 20 % respectivamente.

Tabla 3.1: Tamaño de las bases de datos

Tamaño de Ventana	**Imágenes por canal**	**Total**
0.5 s	7484	329.296
1 s	3736	164.384
2 s	1870	82.280

3.2. Diseño de arquitectura del clasificador

En esta sección se presenta la arquitectura general de Deep Learning a utilizar para las diferentes bases de datos. Es necesario notar y recalcar que para comparar tanto la técnica de Deep Learning con dropout *estándar* y la técnica de Deep Learning Bayesiano, la arquitectura de la red es exactamente la misma, lo que cambia en el diseño son los lugares en dónde se aplica la técnica de Dropout además de cómo se evalúan cada una de estas arquitecturas. Así, en la Fig. 3.9 se puede notar la arquitectura general de la red *deep* diseñada para este trabajo, compuesta por dos capas convolucionales (capas de tipo Conv en la figura) y dos capas de mean-pooling, además de dos capas fully-connected, en donde la última es una capa con dos neuronas dado que las etiquetas han sufrido el proceso de *one hot encoding*, que consta de transformar las etiquetas categóricas o de formato de palabra (como movimiento o no movimiento) en números que son más eficientes de entender para el algoritmo.

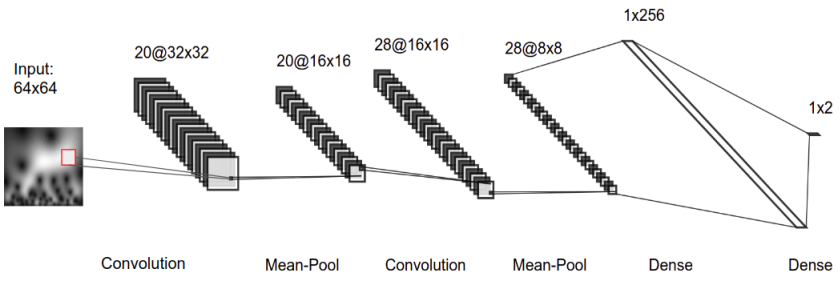


Figura 3.9: Arquitectura diseñada para este trabajo. Se tiene que cada imagen en las diferentes bases de datos es de 64x64 pixeles y es la entrada para cada red. Luego se tienen dos capas convolucionales (Conv1 y Conv2), en donde cada una es seguida por una capa de mean-pooling. Finalmente se tienen dos capas FC, en donde la última presenta dos salidas debido a las dos clases: movimiento y no movimiento. Imagen creada por el autor.

3.3. Pruebas a realizar

En esta sección se detallan las diferentes pruebas a realizar para comparar ambas técnicas de Deep Learning en las bases de datos generadas a partir del modelo animal de 6-OHDA parkinsoniano. Para ello, se definen las diferentes implementaciones de la siguiente manera:

(i) Al usar dropout ya sea luego de las capas convolucionales como también entre las capas fully-connected, se nombra como *DropAll*.

(ii) Al usar dropout de manera tradicional, es decir solo entre las capas fully connected, se nombra como *DropGeneral*.

Por otro lado, al momento de realizar pruebas, se tienen dos maneras diferentes de evaluar estas implementaciones; la primera, a la cual se denomina como *DropTestEstándar* es la manera usual que se utiliza en la literatura, es decir, se realiza una cierta ponderación a todos los pesos en base a la probabilidad de dropout. La segunda, que ya se conoce como *MC Dropout* o *Dropout de Monte Carlo*, consta de evaluar estocásticamente T veces cada ejemplo de test luego del entrenamiento. Es importante, de nuevo, notar las diferencias: al momento de realizar DropTestEstándar, todas las unidades están presentes y cada una es ponderada por un factor en base a la probabilidad de dropout, mientras que en MC Dropout cada unidad o filtro está con una cierta probabilidad en cada evaluación de las T existentes.

De esto, se realizan las siguientes pruebas:

1. Se compara la arquitectura con *DropAll* y las diferentes maneras de evaluar las redes, *DropTestEstándar* y *MC Dropout* para las tres diferentes bases de datos existentes.

2. Para la base de datos de 0.5 segundos, realizar una comparación entre *DropAll* y *DropGeneral*, y las diferentes maneras de evaluación.

3. Analizar cómo varía la incertidumbre epistémica en las diferentes bases de datos, luego de realizado el entrenamiento con la arquitectura de *DropAll* y hecha la evaluación con *MC Dropout*.

4. Comparar el error en los datos de prueba al variar la cantidad de T evaluaciones estocásticas para la base de datos de 0.5 segundos, para la arquitectura *DropAll* y el modo de prueba de *MC Dropout*.

Capítulo 4

Análisis y Resultados

En este capítulo se presentan los diferentes resultados, con un análisis respectivo, a partir de la comparación de dos técnicas diferentes de clasificación de movimiento (*Deep Learning* y *Deep Learning Bayesiano*) de diferentes bases de datos extraídas de un experimento con un modelo animal de 6-OHDA parkinsoniano. Se recuerda que se tienen tres bases de datos diferentes, cada una caracterizada por el tamaño de la ventana deslizante utilizada (0.5, 1 y 2 segundos respectivamente).

El primer experimento realizado consta de comparar la arquitectura de *DropAll* con las dos formas de evaluación, *DropTestEstándar* y *MC Dropout*, con $T = 50$ evaluaciones, para las tres diferentes bases de datos existentes. Cada arquitectura fue configurada de manera de tener una probabilidad de dropout de 0.5, una tasa de aprendizaje de 0.0006 con un optimizador de tipo *Adam*, una regularización de 0.026 dada por un largo de escala fijo de 45. De hecho, para largos de escala mayores, la red propuesta se sobreentrenaba y por ende tenía comportamientos no deseables, mientras que para largos de escala menores la red tampoco podía generalizar bien. Por otro lado, es importante notar que un largo de escala grande (como es 45) es útil para priorizar los datos que poseen poca frecuencia, como es el caso de la clase de *movimiento*.

En particular, se pueden notar en la Tabla 4.1 diferentes valores de *accuracy* para las bases de datos de uno y dos segundos, con las respectivas evaluaciones, mientras que en la Tabla 4.2 se presentan los errores de RMSE o *root mean square error* en *test*, recordando que RMSE es calcular la raíz cuadrada del error cuadrático medio, explicado por la Ec. 2.5.

Se debe tener en cuenta que a menor error y mayor exactitud se posee un clasificador mejor con respecto a otro, por ende para ambas bases de datos la evaluación con MC Dropout para la arquitectura de DropAll es superior en términos de ejecución a la forma de evaluación estándar de dropout.

Tabla 4.1: Exactitud para bases de datos de ventanas de uno y dos segundos, con la arquitectura de *DropAll* y ambas formas de evaluación.

Base de Datos\ Accuracy	DropTestEstándar	MC Dropout
1 s	57.666 %	**60.495 %**
2 s	58.817 %	**62.238 %**

Tabla 4.2: RMSE para bases de datos de ventanas de uno y dos segundos, con la arquitectura de *DropAll* y ambas formas de evaluación.

Base de Datos\ RMSE	DropTestEstándar	MC Dropout
1 s	0.650	**0.628**
2 s	0.641	**0.614**

Ahora bien, es importante notar que cada base de datos se encuentra desbalanceada (la proporción de movimiento es casi de uno es a tres con respecto a no movimiento), por lo que tener a la exactitud como métrica no es adecuado, dado que le da una mayor importancia a los datos que tengan una mayor prevalencia en la base de datos: de hecho, al explorar la matriz de confusión se puede notar que los ejemplos de prueba de "no movimiento" son clasificados correctamente en su mayoría mientras que los de "movimiento" no, lo cual representa un problema. Por ende, para la base de datos de medio segundo se realiza el siguiente procedimiento para poder realizar pruebas: primero, se efectúa un *undersampling*, es decir para la base de datos de entrenamiento (que es de 60 % de la base de datos total) se ignoran diferentes imágenes de la clase de "no movimiento" de manera aleatoria hasta igualar la cantidad existente de imágenes de la clase "movimiento": y como segundo y paso final se usa el *recall* y *score f1* para evaluar el desempeño de cada clasificador.

Dicho lo anterior, en la Fig. 4.1 se puede observar el error de evaluación y de entrenamiento en función de las épocas (se recuerda que una época es cuando el clasificador recorre los datos de entrenamiento una vez), de la cual se puede notar que al estar ambas curvas acercándose a medida que avanzan las épocas, se tiene el caso de que el clasificador aprende de manera adecuada (si se tuviera el caso en que el error de validación fuera mucho mayor al de entrenamiento entonces la red estaría *sobreentrenada*, es decir que generaliza de una manera ineficiente), mientras que en la Tabla 4.3 se pueden apreciar los resultados de las métricas de recall y de score de tipo f1 al comparar ambos clasificadores en la base de datos de medio segundo, además del error de tipo rmse en los datos de prueba. Se notan mejoras en el recall para MC Dropout con respecto a la forma estándar de evaluación de dropout, lo cual implica a la vez en un Score F1 mayor como también un error rmse menor.

Tabla 4.3: Recall, Score F1 y RMSE para base de datos de 0.5 segundos, con la arquitectura de *DropAll* y ambas formas de evaluación.

BD de 0.5 s	DropTestEstándar	MC Dropout
Recall	76 %	**80 %**
F1-Score	60 %	**62 %**
RMSE	0.7079	**0.705**

Un detalle importante es que en ambas formas de evaluación la precisión no pudo superar el 52 %, lo cual es una de las tareas a futuro a mejorar de este clasificador. Esto se puede atribuir a la presencia de ruido en las transformaciones de filtrado durante el preprocesamiento de la señal, o en el proceso de generación de cada imagen luego de la transformada de wavelet, o bien debido a la manera de generar las etiquetas de movimiento a partir de vídeo: dado que el animal pierde movimiento con el tiempo debido a la neurotoxina de 6-OHDA, en el video de conducta se insertan diferentes elementos para que la rata pueda interactuar y moverse, pero en el proceso de insertar elementos o *juguetes* se puede notar la mano del científico a cargo, lo cual podría causar cambios en el centro de masas de cada frame, generando que este algoritmo de etiquetado no sea tan robusto. Para ello se recomienda explorar otros mecanismos para extraer movimiento del animal, como por ejemplo la acelerometría, de manera de comparar esta manera de extracción de datos de etiquetado con el trabajo presentado en este documento.

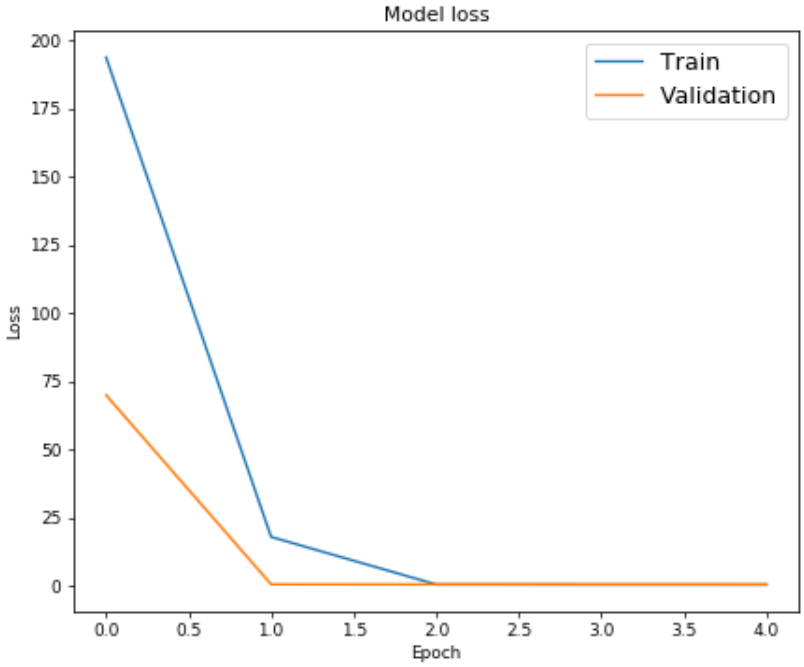


Figura 4.1: Error de entrenamiento y validación en función de las épocas, frenado debido a Early Stopping de validación. Imagen creada por el autor.

El segundo experimento consiste en comparar las arquitecturas de *DropAll* y *DropGeneral* con las diferentes maneras de evaluación en los datos de prueba. Para ello, se recuerda que la arquitectura de *DropGeneral* consiste en solo aplicar dropout entre las capas fully-connected, por ende las capas de dropout que estaban entre las capas convolucionales se deben ignorar para este caso. Además, en la Tabla 4.3 ya se tienen los resultados de ambas maneras de evaluación para la arquitectura de *DropAll*, por que solo basta generar los resultados para la arquitectura faltante, los cuales se pueden notar en la Tabla 4.4.

En la Tabla 4.4 se puede notar que, para la arquitectura de *DropGeneral*, tanto el Recall como el score de f1 se mantienen iguales en porcentaje, mientras que el error RMSE es *menor* al evaluar esta red de manera estándar frente a la evaluación con T=50 de Monte Carlo, lo cual al menos se condice con la literatura para bases de datos como MNIST o CIFAR-10 [36].

Tabla 4.4: Recall, Score F1 y RMSE para base de datos de 0.5 segundos, con la arquitectura de *DropGeneral* y ambas formas de evaluación.

BD de 0.5 s \ DropGeneral	DropTestEstándar	MC Dropout
Recall	57 %	57 %
F1-Score	53 %	53 %
RMSE	**0.7066**	0.7079

El tercer experimento consta de analizar cómo varía la incertidumbre epistémica en las diferentes bases de datos luego de realizado el entrenamiento con la arquitectura de *DropAll* y una vez hecha la evaluación con *MC Dropout*. En particular, de la Tabla 4.5 se pueden notar estos resultados, de donde es necesario mencionar que cada base de datos fue evaluada con T=50 evaluaciones, y en donde se puede ver que el comportamiento de la red, con respecto a incertidumbre, se condice con las mejoras en clasificación al variar el tamaño de ventana. Es decir, de la Tabla 4.1 se tiene la exactitud para la base de datos de dos segundos es mejor que la de un segundo, y además se cumple que la varianza para la red de dos segundos es menor que con respecto a la de un segundo, lo cual también quiere decir que para esta base de datos se tiene una mayor *confianza* al realizar predicciones. Por otro lado se tiene que la red de medio segundo, que posee un recall de 80 %, presenta una varianza menor que las otras bases de datos, lo cual también quiere decir que la confianza en las clasificaciones para esta base de datos es mayor que en las demás.

Tabla 4.5: Media de la varianza al generar T=50 evaluaciones estocásticas en cada base de datos, generando la incertidumbre epistémica.

Base de Datos	Incertidumbre Epistémica
0.5 s	**3.335**
1 s	3.395
2 s	3.351

Finalmente, se tiene el cuarto y final experimento, el cual consiste en comparar el error RMSE en los datos de prueba, usando la arquitectura de *DropAll* con el método de evaluación de *MC Dropout* al variar el parámetro de T evaluaciones estocásticas. En particular, se usan valores de T de 5, 15, 30, 50 y 100, en donde cada experimento fue realizado cinco veces. De la Fig. 4.2, se puede notar que para evaluaciones de $T > 30$, se tiene una diferencia significativa (más de una desviación estándar) con respecto a la evaluación de la arquitectura de *DropGeneral* con forma de evaluación estándar. Esto es importante, porque existe un *tradeoff* entre la eficiencia temporal durante la evaluación de los datos de prueba y la cantidad de evaluaciones estocásticas, que es lo que se busca con el proyecto general en donde este trabajo se encuentra inmerso.

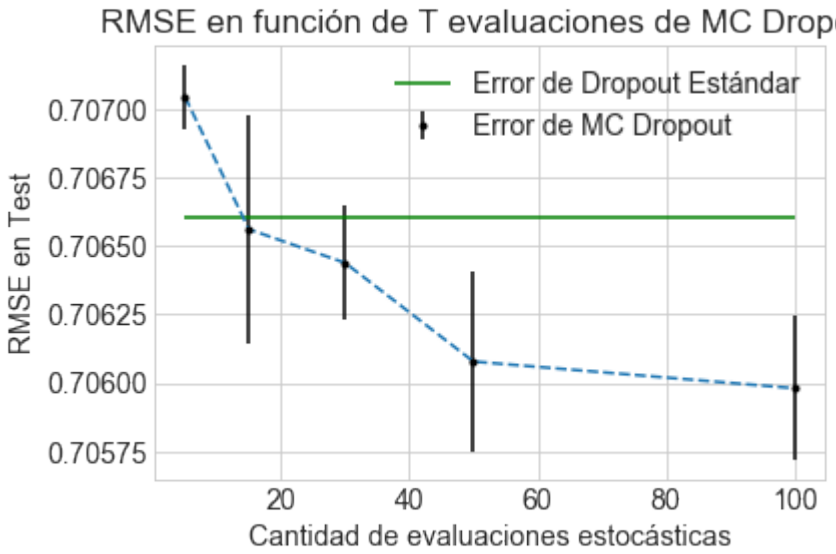


Figura 4.2: RMSE en función de T evaluaciones de MC Dropout, para T de 5, 15, 30, 50 y 100 evaluaciones. Cada experimento fue realizado cinco veces, con lo cual se obtiene que para T>30 existen diferencias significativas con respecto a Dropout Estándar. Imagen creada por el autor.

Hecho esto, y considerando tanto la incertidumbre como la media extraida utilizando MC Dropout y además el número de evaluaciones estocásticas, se recomienda elegir el clasificador de Deep Learning con la arquitectura de *DropAll* y la forma de evaluación de *MC Dropout*, con un algoritmo deslizante de 0.5 segundos. Si el sistema de circuito cerrado no posee las caracteristicas de un poder computacional suficiente para generar imágenes de manera automática con un algoritmo de ventana deslizante de medio segundo, entonces se recomienda elegir un algoritmo de ventana deslizante de dos segundos, en donde en todos los casos se elija un parámetro T mayor a 30 evaluaciones, y en donde además los entrenamientos (no en tiempo real sino para generar parámetros de las arquitecturas de las redes) ocupen una base de datos equilibrada, ya sea a través de undersampling o a través de *data augmentation*, que es una forma de *oversampling*, consistente en aumentar la cantidad de ejemplos de las clases con menor frecuencia en cada base de datos.

Capítulo 5

Conclusiones y Trabajo Futuro

El trabajo de memoria presentado en este documento consiste en un estudio que utiliza diferentes algoritmos de aprendizaje de máquinas para la clasificación de movimiento en base a la actividad neural de un modelo animal de 6-OHDA parkinsoniano, con el fin de formar parte como bloque de un futuro circuito cerrado para un nuevo tratamiento de estimulación epidural, utilizando señales de campos potenciales locales y velocidades extraídas a partir de un video. Para lograr esta clasificación, se generan diferentes bases de datos de imágenes de potencia a través de algoritmos de ventanas deslizantes, además de la extracción de etiquetas a partir de cada frame de un vídeo. Las bases de datos se caracterizan por el tamaño de cada ventana deslizante, tanto para generar imágenes como para extraer etiquetas, y estas son de 0.5, 1 y 2 segundos respectivamente.

Los algoritmos de aprendizaje de máquinas que se comparan son *Deep Learning* y *Deep Learning Bayesiano*, y se logran resultados que se condicen con la literatura actual para las diferentes formas de evaluación de cada arquitectura. De hecho, uno de los resultados más importantes radica en que al utilizar Deep Learning Bayesiano con la forma de evaluación de MC Dropout se llega a obtener un *recall* de 80 % para la base de datos de medio segundo, lo cual es interesante para trabajos futuros que puedan pulir las bases de datos generadas para obtener mejores resultados. Además, se obtiene que a medida que el clasificador obtiene un mejor desempeño entonces existe una menor incertidumbre en la clasificación, lo cual puede servir como parámetros de entrada para un controlador que se conecte en el siguiente bloque del sistema de circuito cerrado.

Es importante notar que este trabajo considera una mejora importante con respecto al estado del arte actual, debido a que por ahora no se han utilizado en demasía algoritmos de aprendizaje de máquinas. De hecho, se espera que quede claro cuáles son las métricas adecuadas a usar para este tipo de bases de datos en que la presencia de eventos importantes a clasificar no es tan frecuente, debido a que el único paper que presentaba una clasificación de movimiento con Deep Learning usaba la métrica de *accuracy*, la cual es inadecuada para clases que se encuentran desbalanceadas.

Por otro lado, es importante tomar en consideración que se deben realizar más experimentos con diferentes algoritmos de aprendizaje de máquinas, con el fin de buscar uno que

pueda minimizar el costo energético de generar ejemplos que formen parte de la base de datos respectiva. De hecho, al poner en marcha este clasificador de Deep Learning Bayesiano con T=50 evaluaciones estocásticas, se necesitaría de un hardware especial, lo cual puede venir en contra de la idea de minimización energética inicial que tiene como propósito crear el circuito cerrado. Para ello, es importante el experimento realizado en el que se varía la cantidad de evaluaciones estocásticas de MC Dropout, pues ya cuando se obtienen diferencias significativas con respecto a otras formas de evaluación (para T >30), entonces se tiene la decisión de buscar un parámetro T que además minimice el costo energético de los cálculos computacionales.

Si se deciden realizar más experimentos con las bases de datos creadas, se recomienda explorar diferentes formas para extraer movimiento (al mismo tiempo en que se esté midiendo la actividad cerebral), como por ejemplo la acelerometría, o bien, seguir con el algoritmo actual (extracción de centro de masas de cada frame) pero teniendo un mayor cuidado al momento de insertar objetos dentro de la caja en donde se encuentre el animal, para evitar problemas con el algoritmo.

Finalmente, se cumple el objetivo general y los objetivos específicos de este trabajo, recomendando utilizar el clasificador de *Deep Learning Bayesiano* con T=50 evaluaciones estocásticas y el algoritmo de ventanas deslizantes de 0.5 segundos.

Bibliografía

[1] György Buzsáki and Andreas Draguhn. *Neuronal Oscillations in Cortical Networks. Science*, 304(5679):1926–1929, 2004.

[2] Francisco Varela, Jean-Philippe Lachaux, Eugenio Rodriguez, and Jacques Martinerie. *The brainweb: Phase synchronization and large-scale integration*, Apr 2001.

[3] M. Abeles. *Corticonics: Neuronal Circuits of the Cerebral Cortex.* Cambridge University Press, Cambridge, England, 1st edition, 1991.

[4] E.R. Kandel, T.M. Jessell, J.H. Schwartz, S.A. Siegelbaum, and A.J. Hudspeth. *Principles of Neural Science, Fifth Edition.* Principles of Neural Science. McGraw-Hill Education, 2013.

[5] Edward Stein and Izhar Bar-Gad. *Beta oscillations in the cortico-basal ganglia loop during parkinsonism. Experimental Neurology*, 245:52 – 59, 2013. Special Issue: Neuronal oscillations in movement disorders.

[6] A. Moran and I. Bar-Gad. *Revealing neuronal functional organization through the relation between multi-scale oscillatory extracellular signals. Journal of Neuroscience Methods*, 186(1):116 – 129, 2010.

[7] Jose L Cantero, Mercedes Atienza, and Rosa M Salas. *Human alpha oscillations in wakefulness, drowsiness period, and REM sleep: different electroencephalographic phenomena within the alpha band. Neurophysiologie Clinique/Clinical Neurophysiology*, 32(1):54 – 71, 2002.

[8] Gatev Plamen, Darbin Olivier, and Wichmann Thomas. *Oscillations in the basal ganglia under normal conditions and in movement disorders. Movement Disorders*, 21(10):1566–1577.

[9] D. Joel and I. Weiner. *The organization of the basal ganglia-thalamocortical circuits: Open interconnected rather than closed segregated. Neuroscience*, 63(2):363 – 379, 1994.

[10] D.E. Haines and M.D. Ard. *Fundamental Neuroscience for Basic and Clinical Applications.* Haines, Fundamental Neuroscience for Basic and Clinical Appl. Churchill Livingstone, 2006.

[11] Shankar J. Chinta and Julie K. Andersen. *"Dopaminergic neurons". The International*

Journal of Biochemistry & Cell Biology, 37(5):942 – 946, 2005. Cancer and Aging at the Crossroads.

[12] Stephanie Martin, Inaki Iturrate, Ricardo Chavarriaga, Robert Leeb, Aleksander Sobolewski, Andrew M. Li, Julien Zaldivar, Iulia Peciu-Florianu, Etienne Pralong, Mayte Castro-Jimenez, David Benninger, Francois Vingerhoets, Robert T. Knight, Jocelyne Bloch, and Jose del R. Millan. *Differential contributions of subthalamic beta rhythms and neural noise to Parkinson motor symptoms. bioRxiv*, 2018.

[13] A. Berardelli, J. C. Rothwell, P. D. Thompson, and M. Hallett. *Pathophysiology of bradykinesia in Parkinson's disease. Brain*, 124(11):2131–2146, 2001.

[14] Pedro Chaná C, Magdalena Jiménez C, Violeta Díaz T, and Carlos Juri. *Mortalidad por enfermedad de Parkinson en Chile. Revista médica de Chile*, 141:327 – 331, 03 2013.

[15] Jane A. Driver, Giancarlo Logroscino, J. Michael Gaziano, and Tobias Kurth. *Incidence and remaining lifetime risk of Parkinson disease in advanced age. Neurology*, 72(5):432–438, 2009.

[16] E. R. Dorsey, R. Constantinescu, J. P. Thompson, K. M. Biglan, R. G. Holloway, K. Kieburtz, F. J. Marshall, B. M. Ravina, G. Schifitto, A. Siderowf, and C. M. Tanner. *Projected number of people with Parkinson disease in the most populous nations, 2005 through 2030. Neurology*, 68(5):384–386, 2007.

[17] R S Burns, C C Chiueh, S P Markey, M H Ebert, D M Jacobowitz, and I J Kopin. *A primate model of parkinsonism: selective destruction of dopaminergic neurons in the pars compacta of the substantia nigra by N-methyl-4-phenyl-1,2,3,6-tetrahydropyridine. Proceedings of the National Academy of Sciences*, 80(14):4546–4550, 1983.

[18] Susan H. Fox and Jonathan M. Brotchie. *Chapter 7 - The MPTP-lesioned non-human primate models of Parkinson's disease. Past, present, and future.* In Anders Björklund and M. Angela Cenci, editors, *Recent Advances in Parkinson'S Disease*, volume 184 of *Progress in Brain Research*, pages 133 – 157. Elsevier, 2010.

[19] Urban Ungerstedt. *6-hydroxy-dopamine induced degeneration of central monoamine neurons. European Journal of Pharmacology*, 5(1):107 – 110, 1968.

[20] M.D. LeWitt, Peter A. *Levodopa for the Treatment of Parkinson's Disease. The New England journal of medicine*, 359(23):2468–76, Dec 04 2008. Copyright - Copyright © 2008 Massachusetts Medical Society. All rights reserved; Última actualización - 2017-10-31; CODEN - NEJMAG.

[21] Jankovic Joseph. Motor fluctuations and dyskinesias in parkinson's disease: Clinical manifestations. *Movement Disorders*, 20(11):S11–S16.

[22] Alim Louis Benabid, Stephan Chabardes, John Mitrofanis, and Pierre Pollak. *Deep brain stimulation of the subthalamic nucleus for the treatment of Parkinson's disease. The Lancet Neurology*, 8(1):67 – 81, 2009.

[23] A. Amon and F. Alesch. *Systems for deep brain stimulation: review of technical features. J Neural Transm Vienna*, 124(9):1083–1091, Sep 2017.

[24] Bronstein JM, Tagliati M, Alterman RL, and et al. *Deep brain stimulation for parkinson disease: An expert consensus and review of key issues. Archives of Neurology*, 68(2):165, 2011.

[25] Emerson Magno de Andrade, Maria Gabriela Ghilardi, Rubens Gisbert Cury, Egberto Reis Barbosa, Romulo Fuentes, Manoel Jacobsen Teixeira, and Erich Talamoni Fonoff. *Spinal cord stimulation for Parkinson's disease: a systematic review. Neurosurgical Review*, 39(1):27–35, Jan 2016.

[26] Romulo Fuentes, Per Petersson, William B. Siesser, Marc G. Caron, and Miguel A. L. Nicolelis. *Spinal Cord Stimulation Restores Locomotion in Animal Models of Parkinson's Disease. Science*, 323(5921):1578–1582, 2009.

[27] Deuschl Günther, Herzog Jan, Kleiner-Fisman Galit, Kubu Cynthia, Lozano Andres M., Lyons Kelly E., Rodriguez-Oroz Maria C., Tamma Filippo, Tröster Alexander I., Vitek Jerrold L., Volkmann Jens, and Voon Valerie. *Deep brain stimulation: Postoperative issues. Movement Disorders*, 21(14):S219–S237.

[28] Anna Castrioto, Eugénie Lhommée, Elena Moro, and Paul Krack. *Mood and behavioural effects of subthalamic stimulation in Parkinson's disease. The Lancet Neurology*, 13(3):287 – 305, 2014.

[29] Chiung Chu Chen, Christof Brücke, Florian Kempf, Andreas Kupsch, Chin Song Lu, Shih Tseng Lee, Stephen Tisch, Patricia Limousin, Marwan Hariz, and Peter Brown. *Deep brain stimulation of the subthalamic nucleus: A two-edged sword. Current Biology*, 16(22):R952 – R953, 2006.

[30] Meidahl Anders Christian, Tinkhauser Gerd, Herz Damian Marc, Cagnan Hayriye, Debarros Jean, and Brown Peter. *Adaptive Deep Brain Stimulation for Movement Disorders: The Long Road to Clinical Therapy. Movement Disorders*, 32(6):810–819.

[31] Chao-Hung Kuo, Gabrielle A. White-Dzuro, and Andrew Ko. Approaches to closed-loop deep brain stimulation for movement disorders. *Neurosurgical Focus*, 45:E2, 08 2018.

[32] Joaquim Alves da Silva, Fatuel Tecuapetla, Vitor Paixão, and Rui M Costa. *Dopamine neuron activity before action initiation gates and invigorates future movements. Nature*, 554(7691):244, 2018.

[33] Michael A. Nielsen. *Neural Networks and Deep Learning*, 2018.

[34] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. *Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[35] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. *Gradient-based learning applied to document recognition. Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[36] Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

[37] James V. Candy. *Bayesian Signal Processing: Classical, Modern and Particle Filtering Methods*. Wiley-Interscience, New York, NY, USA, 2009.

[38] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[39] S. Kullback and R. A. Leibler. *On Information and Sufficiency. Ann. Math. Statist.*, 22(1):79–86, 03 1951.

[40] Herbert Robbins and Sutton Monro. *A Stochastic Approximation Method. Ann. Math. Statist.*, 22(3):400–407, 09 1951.

[41] Donald B. Rubin. *The Bayesian Bootstrap. Ann. Statist.*, 9(1):130–134, 01 1981.

[42] Naftali Tishby, Esther Levin, and Sara A. Solla. *Consistent inference of probabilities in layered networks: Predictions and generalization*. In Anon, editor, *IJCNN Int Jt Conf Neural Network*, pages 403–409. Publ by IEEE, 12 1989.

[43] Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. *Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation*. In *Australasian joint conference on artificial intelligence*, pages 1015–1021. Springer, 2006.

[44] Yang Zhan, David Halliday, Ping Jiang, Xuguang Liu, and Jianfeng Feng. *Detecting time-dependent coherence between non-stationary electrophysiological signals—a combined statistical and time–frequency approach. Journal of neuroscience methods*, 156(1-2):322–332, 2006.

[45] J Sinkkonen, H Tiitinen, and R Näätänen. *Gabor filters: an informative way for analysing event-related brain activity. Journal of neuroscience methods*, 56(1):99–104, 1995.

[46] Steven J Schiff, Akram Aldroubi, Michael Unser, and Susumu Sato. *Fast wavelet transformation of EEG. Electroencephalography and clinical neurophysiology*, 91(6):442–455, 1994.

[47] Michel Lemistre and Daniel Balageas. *Structural health monitoring system based on diffracted Lamb wave analysis by multiresolution processing. Smart Materials and Structures*, 10(3):504, 2001.

[48] Behtash Babadi and Emery N Brown. *A review of multitaper spectral analysis. IEEE Trans. Biomed. Engineering*, 61(5):1555–1564, 2014.

[49] Marieke K van Vugt, Per B Sederberg, and Michael J Kahana. *Comparison of spectral analysis methods for characterizing brain oscillations. Journal of neuroscience methods*, 162(1-2):49–63, 2007.

[50] K. J. Åström and T. Hägglund, editors. *PID Control: Theory, Design, and Tuning*. Instrument Society of America Press, Research Triangle Park, NC, second edition, 1995.

[51] Boris Rosin, Maya Slovik, Rea Mitelman, Michal Rivlin, Suzanne Haber, Zvi Israel, Eilon Vaadia, and Hagai Bergman. *Closed-Loop Deep Brain Stimulation Is Superior in Ameliorating Parkinsonism.* 72:370–84, 10 2011.

[52] Miguel Pais-Vieira, Amol Yadav, Derek Moreira, David Guggenmos, Amilcar Santos, Mikhail Lebedev, and Miguel A L Nicolelis. *A Closed Loop Brain-machine Interface for Epilepsy Control Using Dorsal Column Electrical Stimulation.* 6, 10 2016.

[53] Ewan Nurse, Benjamin S. Mashford, Antonio Jimeno Yepes, Isabell Kiral-Kornek, Stefan Harrer, and Dean R. Freestone. *Decoding EEG and LFP Signals Using Deep Learning: Heading TrueNorth.* pages 259–266, 2016.

[54] Joshua H Siegle, Aarón Cuevas López, Yogi A Patel, Kirill Abramov, Shay Ohayon, and Jakob Voigts. *Open Ephys: an open-source, plugin-based platform for multichannel electrophysiology. Journal of neural engineering*, 14(4):045003, 2017.

[55] Kenneth A Clarke and Andrew J Parker. *A quantitative study of normal locomotion in the rat. Physiology & behavior*, 38(3):345–351, 1986.

[56] Parishwad P Vaidyanathan. *Multirate digital filters, filter banks, polyphase networks, and applications: a tutorial. Proceedings of the IEEE*, 78(1):56–93, 1990.

[57] Shanbao Tong and Nitish Vyomesh Thakor. *Quantitative EEG analysis methods and clinical applications.* Artech House, 2009.