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

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA BASADO EN "MACHINE LEARNING" PARA LA DETECCIÓN Y CARACTERIZACIÓN AUTOMÁTICA DE EVENTOS SÍSMICOS

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA.

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERA CIVIL ELÉCTRICA

CATALINA ANDREA MURÚA FREDES

PROFESOR GUÍA:
NÉSTOR BECERRA YOMA

MIEMBROS DE LA COMISIÓN:
FERNANDO HUENUPAN QUINAN
CÉSAR AZURDIA MEZA

Este trabajo ha sido parcialmente financiado por FONDEF ID20i-10212 y
Centro Nacional de Inteligencia Artificial CENIA FB210017, Financiamiento Basal ANID

SANTIAGO DE CHILE
2023

RESUMEN DE LA TESIS PARA OPTAR
AL GRADO DE MAGÍSTER EN CIENCIAS
DE LA INGENIERÍA, MENCIÓN ELÉCTRICA
Y MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERA CIVIL ELÉCTRICA
POR: CATALINA ANDREA MURÚA FREDES
FECHA: 2023
PROF. GUÍA: NÉSTOR BECERRA YOMA

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA BASADO EN "MACHINE LEARNING" PARA LA DETECCIÓN Y CARACTERIZACIÓN AUTOMÁTICA DE EVENTOS SÍSMICOS

El desarrollo de sistemas automáticos, confiables y robustos para detección de terremotos es una tarea desafiante y necesaria. El rendimiento de estos puede empeorar si disminuye la SNR de sismogramas y si se emplea en una región diferente a la base de entrenamiento. Consecuentemente, los sistemas entrenados con bases de datos locales deberían funcionar mejor, pero suelen ser limitados. Para abordar estas dificultades, se propone un sistema end-to-end basado en DNN-HMM, con *engineered features* y modelos de duración de estados y eventos sísmicos. Esta propuesta requiere 20 veces menos parámetros que métodos más avanzados, por ello necesita una base de datos de entrenamiento más pequeña. Además, los modelos de duración pueden aumentar significativamente la robustez al ruido. El sistema descrito proporciona un puntaje f1-score promedio 105 % mayor que los esquemas publicados con bases de datos de Iquique y del norte de Chile.También presenta una reducción del puntaje f1-score de un 16 %, cuando la SNR promedio se reduce en 17dB aproximadamente, siendo mínimo la mitad de la observada en otros sistemas avanzados con condiciones equivalentes. Respecto a la detección de pequeños sismos a cortas distancias epicentro-estación, el promedio de la precisión es al menos un 7 % superior a otros sistemas.

Agradecimientos

Agradezco a mis padres por haberme dado su apoyo incondicional en toda mi etapa de educación y por haberme enseñado que ante toda adversidad puedo hallar victoria con Dios. Sus consejos me ayudaron a sobrellevar todas mis frustraciones. Agradezco a mi mamá Jeannette por acompañarme en esas noches de vigilia cuando me faltaban muchos controles por pautear. Siempre recordaré las palabras de ánimo y los alfajores que me comprabas cuando no obtenía una buena nota. Agradezco a mi papá Marco por ir a buscarme a la Universidad, o a donde fuera necesario, cuando me quedaba hasta muy tarde estudiando. Siempre recordaré un lugar seguro junto a ti. Papá y mamá, ustedes fueron el motor que me impulsó a llegar a donde estoy hoy en día. Todo lo que tengo se lo debo a Dios y a ustedes.

También quiero agradecer profundamente a mis hermanas Jeannette, Michelle y Pamela y a mis abuelitos Santiago, Noemí y Berta. Gracias por acompañarme, quererme y cuidarme. No me imagino mi vida sin ustedes. Cada consejo que me dieron me ayudó mucho a crecer como persona. Gracias por el apoyo incondicional y por tenerme siempre en sus oraciones.

Quiero agradecer también a mis amigos de la universidad por dejarme tan bellos recuerdos en esta larga etapa. Gracias a mis amigos de plan común, Ariel, Camila, Camilo, Erick, Javier, Luccas, Max y Nacho, y a mis amigos de especialidad, Javier, Joaquín, Maximiliano, Iván, Rodrigo, Nicolás, Eduardo, Oscar, Franco, Carla y Pablo, por llenarme de alegrías y risas los últimos 7 años de mi vida. Gracias por su constante ayuda y por siempre motivarme a querer aprender más. Recuerdo con gran cariño todas esas tardes y horas de trabajo juntos, donde nunca faltó un opening. Con ustedes pude hacer de la Universidad mi segundo hogar.

Agradezco también al profesor Néstor Becerra por todos los aprendizajes que me brindó. Su constante guía y ayuda formó parte importante en el desarrollo de esta etapa final. Gracias a Jorge y Aaron del Laboratorio de Procesamiento y Transmisión de Voz, por todos los conocimientos que me brindaron.

Quiero agradecer de forma a especial a mi amado esposo Javier por todos los momentos vividos y por su infinita ayuda incondicional. De principio a fin, de amigos a esposos, has sido una persona imprescindible en esta etapa. Gracias por todas esas tardes de estudios, de salidas y de escapadas. Gracias por siempre escucharme y enseñarme. Hoy cierro una etapa académica, pero a la vez comienzo una etapa contigo que durará el resto mi vida.

Finalmente alabo y bendigo a mi Dios por esta oportunidad. Todo lo que soy y todo lo que seré se lo debo solo a Él.

Mi abuelita Berta siempre decía “Tienes que estudiar para llegar a ser una linda profesional”. Hoy doy gracias a Dios de que por fin se logró. Gracias a todos ustedes por hacer posible el éxito de esta etapa.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Nomenclatura

CE Cross Entropy

CNN Convolution Neural Network

CRED Cnn-Rnn Earthquake Detector

CSN Centro Sismológico Nacional

DFT Discrete Fourier transform

DL Deep Learning

DMC Data Management Center

DMEE Duration Models Event over Earthquake

DMEN Duration Models over Earthquake and Noise

DME Duration Models over Earthquakes

DMSEN Duration Models Events over Earthquake and Noise

DMSEN Duration Models States over Earthquake and Noise

DMSE Duration Models States over Earthquake

DNN-HMM Depp Neural Network - Hidden Markov Model

DNN Deep Neural Network

EDP Event Duration Penalty

EM Expectation–maximization

FAST Fingerprint and Similarity Thresholding

FFT Fast Fourier transform

FN False negative

FP False positive

GMM Gaussian Mixtures Model

GPU Graphics processing unit

HMM Hidden Markov Model

HSMM Hidden semi-Markov model

IRIS Include, Respect, I Self-Direct

LSTM Long short-term memory

LTA Long-time-average trigger

MFCC Mel-frequency cepstrum

MLP Multilayer perceptron

ML Machine Learning

MVN Mean-Variance Normalization

NDM No Duration Models

PDF Probability density function

RAM Random Access Memory

ReLU Rectified linear unit

RNN Recurrent neural network

SNR Signal to Noise Ratio

STA Short-time-average

STEAD STanford EArthquake Dataset

STFT Short-time Fourier transform

TP True positive

WFST Weighted Finite State Transducers

Capítulo 1

Introducción

En este capítulo se presenta el tema de investigación, planteando el problema junto con la solución propuesta. Además, se expone los objetivos del desarrollo del trabajo y se describe la estructura del presente documento.

En los últimos años, la comunidad sismológica ha perfeccionado la vigilancia sísmica ampliando el número de estaciones sísmicas y mejorando la resolución de las mediciones. Lo anterior ha generado un aumento del volumen de datos [1] impidiendo a los analistas realizar un seguimiento completo de estas señales. Consecuentemente, ha surgido la necesidad de desarrollar herramientas eficaces y prácticas que sean capaces de detectar automáticamente los eventos sísmicos. Los expertos pueden identificar los eventos sísmicos reconociendo correctamente el comienzo y el final del terremoto, incluso en presencia de ruido en las señales. Sin embargo, ante una gran cantidad de datos, esto supone un problema ya que exige una gran carga de recursos humanos y tiempo.

Generalmente, el monitoreo sísmico comprende de las siguientes etapas: detección sísmica y selección de fases, estimación de magnitudes, asociación de eventos y determinación del hipocentro. Es importante mencionar que la detección sísmica y el picado de fases están relacionadas, pero no son lo mismo. La detección se refiere a la identificación de las señales sísmicas entre una amplia variedad de señales no sísmicas y de ruido registradas por un sensor sísmico. El picado de fase es la medición en un sismograma de los tiempos de llegada de la onda P y la onda S. Dado que la detección utiliza una visión más global de las ondas sísmicas [2], es interesante introducir este problemas a técnicas *end-to-end*. Con un detector *end-to-end*, se podrían encontrar terremotos dentro de una señal continua sin elegir explícitamente las fases sísmicas. Este tipo de enfoque de extremo a extremo se puede aplicar a la estimación de magnitud ([3, 4]) y la determinación del epicentro ([5, 6]).

A nivel mundial, la mayoría de las zonas sismogénicas exhiben estrechas relaciones espaciales con los límites de las placas tectónicas [7]. Durante siglos, Chile ha experimentado una actividad sísmica extrema debido a la subducción de la placa oceánica sobre la placa sudamericana. Los registros de la sismicidad chilena van desde terremotos con magnitudes superiores a 1,2 Mw y aquellos cuyas magnitudes superan 9 Mw, como el terremoto de Valdivia de 1960 considerado como el más grande de la historia, 9,5 Mw [8] y [9]. Dado su tamaño y tasa de sismicidad, Chile se ha posicionado como una de las regiones más sismogénicas del mundo [8]. Considerando la variabilidad de los registros sísmicos chilenos, es importante generar un

modelo calibrado que considere estos efectos locales. En consecuencia, los modelos deben ser entrenados con una base de datos local. Según [10], la variabilidad de los datos está mejor representada cuando no se utilizan modelos exportados de otras regiones. Esta problemática es compartida con otras región sismogénica del mundo, pero plantea el problema de entrenar modelos robustos y precisos con datos de entrenamiento limitados.

Uno de los retos que destacan en el estudio de la sismicidad local es el problema del limitado registro de terremotos. Generalmente, el número de eventos disponibles tomados de los catálogos de terremotos es insuficiente para construir un detector preciso [11]. Restringe la posibilidad de reentrenar un modelo que requiere una amplia base de datos con datos locales. Además, según la relación Gutenberg-Richter [12], los terremotos más grandes son menos frecuentes, por lo que es casi imposible generar una gran base de datos con la misma frecuencia de diferentes magnitudes. Otra limitación importante es la baja relación señal-ruido (SNR) presente en los registros microsísmicos. Por debajo de una magnitud determinada, no se consigue la detección en un número adecuado de estaciones sísmicas, lo que da lugar a terremotos no catalogados y a datos de entrenamiento sesgados con respecto a los eventos de mayor magnitud [13]. Como la amplitud de los terremotos de pequeña magnitud expresa una sólida sensibilidad para la heterogeneidad geológica local [14], se requiere que su caracterización precisa se lleve a cabo con observaciones cercanas, es decir, con pequeñas distancias evento-estación. Dada la relativa escasez de terremotos registrados localmente, el entrenamiento de los métodos basados en ML para la búsqueda o detección de terremotos puede ser un reto debido a los limitados datos de entrenamiento [15].

Una posible solución para el problema de los datos de entrenamiento limitados sería el uso de características de ingeniería en lugar de datos brutos. Las características de ingeniería pueden reducir la dimensionalidad de entrada y representar mejor la característica del proceso objetivo [16]). Las características de ingeniería conducen a una descripción más compacta del proceso que se está modelando porque comprimen la señal original en bruto en menos parámetros, al mismo tiempo que preservan las características más importantes y reducen la redundancia [17]. Como resultado, las arquitecturas de ML requieren menos coeficientes para ser entrenadas. En el caso de las DNN, las primeras arquitecturas o capas empleadas para reducir la dimensionalidad de los datos de entrada no son necesarias.

En este trabajo se propone un sistema *end-to-end* para la detección automática de eventos sísmicos basado en el sistema DNN-HMM con modelos de duración de estados y eventos. Se destaca de este modelo la pequeña cantidad de parámetros entrenables, ya que se redujo en más de un orden de magnitud en comparación con los métodos basados en aprendizaje profundo publicados en otros lugares. Como consecuencia, se suaviza la restricción con respecto al tamaño de los datos entrenados. El sistema propuesto fue evaluado con la bases de datos *North Chile*. El enfoque de duración de estado y evento permite introducir restricciones para las duraciones mínima y máxima de los estados HMM y el evento completo. La incorporación de modelos de duración de eventos y estados en el procedimiento de decodificación de Viterbi aumenta la precisión de detección de manera complementaria con respecto a las características espectrales y disminuye la tasa de falsos positivos. Hasta donde sabemos, la aplicación de la tecnología DNN-HMM o el modelado de duración a la detección de eventos sísmicos no se han explorado de manera exhaustiva en la literatura.

1.1. Motivación

La detección de eventos sísmicos es importante para el estudio y avance de la sismología. El seguimiento de los mismos es una herramienta útil en el análisis posterior, ya que proporciona información sobre la dinámica, la forma y la composición del interior de la tierra, así como su interacción con la superficie [18].

Es importante detectar terremotos de todas las magnitudes, incluidos los que no son perceptibles, ya que un gran número de pequeños terremotos y su frecuencia los convierte en la clave para comprender la secuencia de eventos que vinculan los terremotos anteriores, principales y las réplicas [19].

La evaluación de la sismicidad local puede ser una herramienta valiosa para mitigar el riesgo de las actividades industriales, ya que proporcionaría información a los ingenieros sobre las fracturas y fallas preexistentes del lugar de explotación. Varios estudios ya aplican la monitorización microsísmica para caracterizar la fuente y mejorar la comprensión de los sistemas de fracturas naturales. Se puede ver en estudios relacionados con el análisis de estabilidad de taludes [20], la alerta de desastres dinámicos de gas [21], la predicción de colapsos en centrales eléctricas subterráneas, minas, túneles y otros proyectos [22], la irrupción de agua [23], entre otros. Con base en lo anterior, es fundamental contar con lineamientos y protocolos de toma de decisiones en la ingeniería civil que dependan de la actividad sísmica local, con los cuales se puedan gestionar o mitigar los peligros y riesgos y además garantizar la seguridad del personal y el avance de las obras [24].

Finalmente, la motivación de esta investigación radica en generar un sistema de detección automática de sismos robusto que pueda resolver las problemáticas mencionadas anteriormente.

1.2. Hipótesis de trabajo

Se propone como hipótesis para esta investigación los siguientes puntos:

- Mediante una técnica de aprendizaje automático es posible generar detecciones automáticas de sismos.
- La generalización de un sistema depende de la cantidad de parámetros definidos en el modelo.
- Es posible mejorar el rendimiento de un sistema de detección sísmica, incorporando en la decodificación modelos de duración.

1.3. Objetivos generales

El objetivo general consiste en detectar sismos, incluyendo los de baja magnitud que no son catalogados, de modo automático.

1.4. Objetivos específicos

Los objetivos específicos de este trabajo son los siguientes:

- Generar una base de datos, la cual incluya sismos de baja, mediana y elevada magnitud, para investigar y desarrollar los métodos propuestos y que pueda ser utilizada en futuras investigaciones.
- Evaluar el aporte que pueden hacer los modelos de duraciones de sismos en la detección automática de terremotos.
- Evaluar el aporte de Deep Learning en un sistema de detección automática de terremotos basado en HMM.
- Evaluar el efecto de la SNR en la exactitud de la detección automática de terremotos.

1.5. Estructura de la tesis

Esta tesis está compuesta por 4 capítulos aparte de la introducción. En el Capítulo 2 se presenta una visión general de los aspectos teóricos de este trabajo e incluye las principales investigaciones de sistemas automáticos de detección sísmica. En el Capítulo 3, por otro lado, se presenta la técnica propuesta para la detección automática de sismos, la cual consiste en un sistema DNN-HMM con modelos de duración de estados y eventos. También se describe la base de datos y los escenarios experimentales que se utilizaron para evaluar el desempeño de la técnica propuesta, los cuales consisten en experimentos con duraciones de estados y eventos, comparaciones con técnicas de la literatura basadas en modelos de aprendizaje profundo y evaluaciones de desempeño sobre la microsismicidad local. En el Capítulo 4 se presentan los resultados que se obtuvieron en los distintos escenarios experimentales. También presenta las discusiones sobre el desempeño de la técnica propuesta. Finalmente en el Capítulo 5 se describe las principales conclusiones extraídas y se presenta una perspectiva sobre el trabajo futuro que puede continuar desarrollándose como resultado de esta investigación.

Capítulo 2

Detección automática de eventos sísmicos

En este capítulo se presentan las bases teóricas que fundamentan este trabajo e incluye las principales investigaciones de sistemas automáticos de detección sísmica.

2.1. Actividad Sísmica

Chile es un país proclive a sufrir terremotos, pero gracias a la extensa red sismológica y geodésica desplegada por el Centro Sismológico Nacional (CSN) a lo largo de todo el territorio Chileno, los movimientos del suelo que se generan de forma natural y artificial son constantemente registrados. En este trabajo, se considera como área de estudio el Norte de Chile, abarcando aproximadamente 2000 [km] desde la Región de Arica y Parinacota hasta la Región Metropolitana. En función de lo anterior, en esta sección caracterizaremos el contexto sismotectónico de Chile y describiremos la composición de una señal sísmica.

2.1.1. Antecedentes sismológico de Chile

Durante siglos, Chile ha experimentado una actividad sísmica extrema debido a la subducción de la placa oceánica sobre la placa sudamericana, siendo este margen el lugar donde se generan o se han generado históricamente los terremotos que han causado más daños [25].

Chile se ha posicionado como una de las regiones más sismogénicas del mundo [8]. Desde los últimos cinco siglos, cada 10 años se produce en promedio un terremoto destructor de magnitud superior a 8 en alguna parte del territorio chileno [25]. Las magnitudes de los registros sísmicos van desde Mw$\sim 1,2$ hasta magnitudes que superan los 9 Mw, como por ejemplo el terremoto de Chile de 1960 de magnitud 9.5 Mw, el cual fue registrado como el más grande de la historia [8]. Los mega-terremotos en esta región pueden ocurrir en tierra y en alta mar, con un alto potencial de tsunami [9].

2.1.2. Descripción señales sísmicas

La Tierra es un planeta dinámico que día a día cambia a una velocidad casi imperceptible. Este dinamismo se manifiesta en la apertura y cierre de las cuencas oceánicas y en

el desplazamiento de las masas continentales, conocido como deriva continental [26]. Este reordenamiento constante, ha tenido un profundo efecto sobre la geología de la superficie de nuestro planeta ya que ha dado lugar a que los continentes se fragmenten, o que las colisiones formen grandes continentes con inmensas cadenas montañosas a lo largo de las intersecciones. Las rocas sometidas a estos permanentes esfuerzos, a veces acumulan más allá de su límite elástico, generando fracturas y súbitos desplazamientos. Las vibraciones causadas por lo anterior, ocasiona que en el interior y superficie de la tierra se propaguen ondas en todas las direcciones, originando lo que se conoce como sismo. El punto al interior de la tierra donde se inicia la ruptura se llama foco o hipocentro, y el punto sobre la superficie de la Tierra, directamente encima del hipocentro, se llama epicentro. Los sismos no son aleatorios, sino que son fuerzas naturales impulsadas por los procesos evolutivos del planeta en el que vivimos [27].

Las ondas sísmicas se pueden dividir en dos grandes grupos: Ondas Internas y Ondas Superficiales.

Ondas Internas:

Las ondas internas u ondas de cuerpo se propagan en el interior, siguiendo caminos curvos debido a la variada densidad y composición del interior de la Tierra. Las ondas internas se dividen en dos grupos: ondas primarias (P) y secundarias (S).

- **Onda Primarias (P)**: Son ondas longitudinales o compresionales, es decir que oscilan en la dirección de propagación de la onda. Esta onda puede viajar a través de cualquier tipo de material líquido o sólido y es la más veloz de todas las ondas sísmicas, por lo tanto, es la primera en ser sentida y registrada. El sonido de "retumbo" que se produce antes de un terremoto se asocia con el arribo de la onda P.
- **Onda Secundarias (S)**: Son ondas cuya propagación transversal a la dirección de propagación de la perturbación. Su velocidad es menor que la de las ondas primarias. Estas ondas no pueden viajar a través de elementos líquidos y son las ondas que generan las oscilaciones durante el movimiento sísmico.

Ondas Superficiales (L):

Las ondas superficiales se generan por la interacción de las ondas de cuerpo con la superficie de la Tierra y se propagan por esta última. A pesar de ser las más lentas de todas las ondas sísmicas, son las más destructivas. Las ondas superficiales son divididas en dos grupos: ondas Raleight y ondas Love.

- **Onda Raleight**: Son ondas superficiales que producen un movimiento elíptico retrógrado del suelo, y cuya amplitud disminuye exponencialmente con la profundidad. Su velocidad es menor que la velocidad de las ondas internas.
- **Onda Love**: Son ondas superficiales que se originan en la interfase de dos medios con propiedades mecánicas diferentes y producen un movimiento horizontal de corte en superficie. La velocidad de las ondas Love es menor que las de onda de cuerpo y ligeramente superior a la velocidad de las ondas Rayleigh.

Las fases sísmicas son imprescindibles para entender las características internas de la Tierra. Hoy en día gracias a los avances tecnológicos, los terremotos se pueden mapear, medir,

analizar y desmitificar. Para identificar las fases sísmicas se emplean los registros medidos por sismogramas, ya que el tiempo de llegada de cada onda sísmica tiene una forma representativa en el registro de la señal debido a las distintas velocidades que presentan.

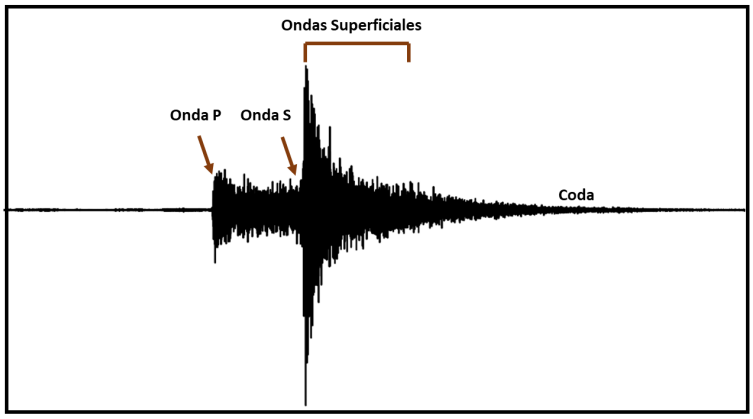


Figura 2.1: Sismograma con arribo de ondas

En la figura 2.1 se presenta un sismograma que contiene un sismo. En esta señal la onda sísmica que presenta el primer arribo es la onda P. Posteriormente le sigue la onda S y finalmente las ondas superficiales. Después de la llegada de ondas internas y superficiales, se observa en el sismograma una vibración residual cuya amplitud decae lentamente en el tiempo. Este segmento de la señal sísmica es conocido como coda, y se debe a la retrodispersión de numerosas heterogeneidades del terreno. La duración de la coda es proporcional al tamaño del sismo, aunque puede verse afectada por otros factores, como por ejemplo la naturaleza del suelo en el que se encuentra la estación [28].

2.2. Procesamiento de Señales

El procesamiento de señales es la manipulación matemática de la información de una señal y está está caracterizado por la representación en el dominio temporal, frecuencial u otro dominio discreto de señales. El estudio en el dominio de frecuencia, permite analizar el comportamiento cualitativo de un sistema.

La Transformada de Fourier resulta útil para estos fines ya que genera una representación paramétrica con la menor cantidad de información redundante, incluyendo características temporales y espectrales [29]. También proporciona métodos para determinar la importancia de cada frecuencia simple o una banda de frecuencia en la construcción de una señal dada [30], permitiendo descartar o filtrar componentes de ruido. Este trabajo desarrolla un procesamiento frecuencial y temporal de señales, ya que se consideran aspectos en el dominio del tiempo de la señales y luego se analizan los datos en el dominio de la frecuencia.

2.2.1. Ventana de Hamming

Las ventanas son funciones matemáticas ampliamente utilizadas en aplicaciones de análisis y estimación de señales, diseño de filtros digitales y procesamiento de voz [31]. Esta técnica

establece que lo valores que estén fuera de un intervalo escogido sean igual cero. A pesar de que hay muchas funciones de ventana disponibles, todas tienen en común que están diseñadas para reducir los lóbulos laterales de la salida espectral de las rutinas de transformada rápida de Fourier (FFT) [32].

Una función de ventana ampliamente utilizada es la ventana de Hamming. Esta ventana se utiliza en comunicación y computación para suavizar los datos antes de aplicar el análisis de Fourier [33]. La ventana está optimizada para minimizar el lóbulo lateral máximo [34], por lo que su aplicación genera un espectro de apariencia más limpia y una menor interferencia de frecuencias lejanas en cualquier señal de interés.

La ventana de Hamming se define en la siguiente ecuación:

$$w(n) = \begin{cases} 0.54 - 0.46\cos\left(2\pi\frac{n}{M}\right), n = 0 < n < M \\ 0, \text{ caso contrario} \end{cases} \tag{2.1}$$

2.2.2. Transformada de Fourier de Tiempo Reducido (STFT)

La Transformada de Fourier de Tiempo Reducido (STFT) resuelve el problema del análisis de señales no estacionarias. Esta función divide las señal en segmentos de modo que se pueda asumir estacionariedad en cada intervalo. Luego por cada segmento de la señal, se aplica la transformada de Fourier. La STFT está dado por la siguiente ecuación:

$$\text{STFT}\{x[n]\} = X(m, w) = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-jwn} \tag{2.2}$$

donde $x[n]$ es la señal, $w[n]$ es una función ventana cuyo ancho representa el largo del segmento y m es el índice de tiempo discreto.

2.2.3. Relación Señal-Ruido (SNR)

El rendimiento de un sistema se puede juzgar en términos de la relación señal/ruido (SNR). A mayor valor de SNR, mejor es el resultado obtenido[35]. La relación señal-ruido (SNR) es una métrica de calidad y da cuenta de la relación entre la potencia de la señal requerida y la potencia del ruido de fondo. Si tenemos una señal $s(t) = x(t) + n(t)$, donde x(t) es la señal requerida sin ruido y n(t) es el ruido, la definición anterior está representada por la ecuación 2.3.

$$SNR = \frac{P_x}{P_N} \tag{2.3}$$

donde P_x es la potencial de la señal sin ruido y P_N es la potencia del ruido.

$$SNR_{dB} = 10\log_{10}\left(\frac{P_x}{P_N}\right) \tag{2.4}$$

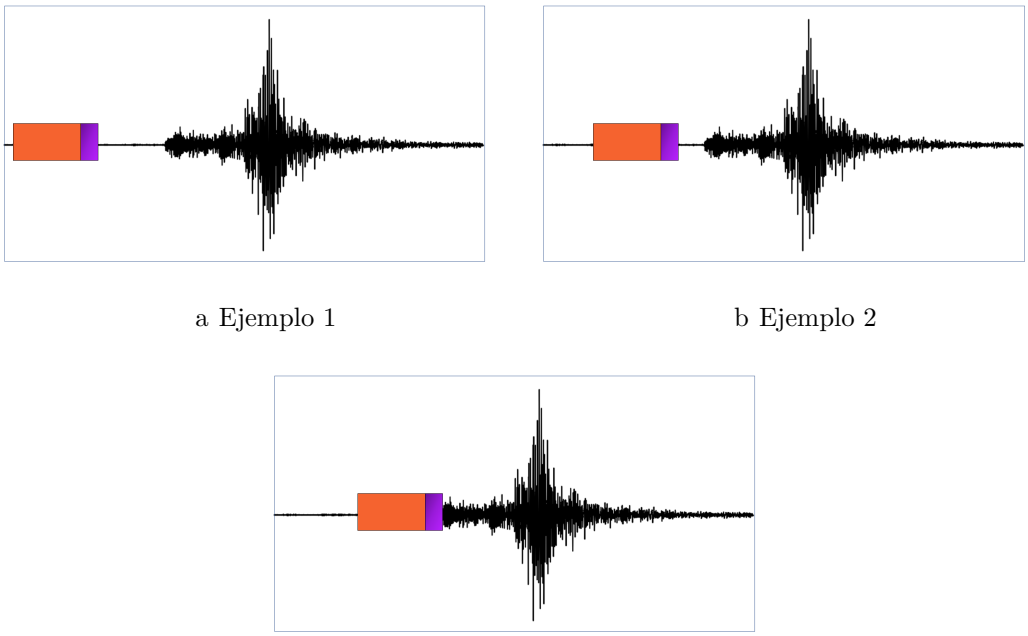
La ecuación 2.4 expresa el SNR en decibelios.

2.3. Métodos de detección automática sísmica

Dado el aumento en el volumen de datos sísmicos, se han propuesto diferentes métodos de detección automática para analizar estos registros. A continuación abordaremos algoritmos de análisis de señales con elementos estadísticos o reconocimiento de patrones como STA/LTA y FAST, además de diferentes algoritmos de aprendizaje automático (ML) utilizados para la detección generalizada de sismos.

2.3.1. Short-Time-Average/Long-Time-Average (STA/LTA)

El algoritmo *short-time-average/long-time-average* (STA/LTA) es un método de detección sísmica basado en la envolvente [36]. Esta técnica compara continuamente a lo largo del tiempo los promedios de amplitud de dos ventanas consecutivas de una señal sísmica [37], como se muestra en la figura 2.2.



a Ejemplo 1

b Ejemplo 2

c Ejemplo 3

Figura 2.2: Ejemplos del algoritmo STA/LTA. El segmento morado representa la ventana del promedio corto, mientras que el segmento naranjo es la ventana del promedio largo. La relación de estas es baja en los ejemplos 1 y 2, por lo que no se activará una detección. Por otro lado en el ejemplo 3 la relación es alta, lo que dará como resultado una detección activada.

El promedio de tiempo corto (STA) es sensible a los eventos sísmicos ya que representa el promedio instantáneo de una duración corta. Por otro lado, el promedio largo (LTA) informa sobre la amplitud temporal del ruido sísmico, ya que representa el promedio anterior de mayor duración. Si consideramos cada momento t para cada k-ésimo canal de datos x_t se tiene que

$$STA = \frac{1}{N_S} \sum_{n=1}^{N_S} y_{k,n} \tag{2.5}$$

$$LTA = \frac{1}{N_L} \sum_{n=-N_L}^{0} y_{k,n} \tag{2.6}$$

donde N_S son los puntos del promedio a corto plazo y N_S es el promedio a largo plazo. El parámetro y es la función característica que potencia los cambios de señal. Suele utilizarse funciones de energía, valor absoluto o función envolvente. La relación STA/LTA está dada por $R = \frac{\text{STA}}{\text{LTA}}$. Cuando la proporción R supera un umbral, los datos se comienzan a registrar ya que se declara la detección de un evento.

Este método encuentra dificultades para establecer parámetros, ya que se debe considerar una compensación entre variables sismológicas e instrumentales [37]. Además, STA/LTA es propenso a generar detecciones falsos positivos en presencia de un alto nivel de ruido [38].

2.3.2. Fingerprint and Similarity Thresholding (FAST)

La técnica *Fingerprint and Similarity Thresholding* (FAST) [39] es un método no supervisado para detección de terremotos de magnitudes variables, especialmente de magnitudes bajas. El origen de este método está basado en la aplicación de búsqueda de música *Wavepprint* [40], el cual combina métodos de procesamiento de datos a gran escala y técnicas de inteligencia artificial para hacer coincidir clips de audio similares. FAST fue diseñado para complementar métodos de detección existentes basados en STA/LTA y tiene como ventaja que no necesita información de eventos catalogados.

Este método extrae características de onda continuas y las convierte en una base de datos de huellas dactilares. Estas huellas dactilares representan de forma compacta a segmentos cortos de una señal y se organizan en una estructura de diccionario para una búsqueda eficiente de similitudes mediante hashing sensible a la localidad [15, 41].

FAST es un algoritmo no supervisado extendido a múltiples estaciones [42] y se ha mostrado prometedor para facilitar el procesamiento automatizado de grandes conjuntos de datos de forma de onda, dada su eficiencia computacional. Para aplicar FAST es necesario configurar una gran cantidad de parámetros cuyos valores dependen de la región y del tipo de instrumento de medición. Esta técnica es susceptible a detecciones falsas porque las fuentes de ruido locales persistentes pueden producir formas de onda repetitivas o similares [42]. Otra limitación de FAST es que si los terremotos son muy diferentes entre sí, es probable que sus huellas digitales también sean muy diferentes, lo que a su vez puede dificultar su detección. Además, este sistema no puede modelar la estructura temporal de un terremoto ya que se basa en hacer coincidir pequeños segmentos de una señal sísmica y no intenta representar la estructura global de un terremoto.

2.3.3. Machine Learning

Machine Learning (ML) trata con la evolución de algoritmos computacionales que simulan la capacidad de aprendizaje que tiene el humano sobre el entorno circundante [43]. En otras palabras, es el campo de estudio que busca dotar a las computadores con la capacidad de aprender sin ser programadas explícitamente. Los algoritmos de *machine learning* se basan en la idea de que a partir de grandes cantidades de datos los sistemas pueden aprender, identificar patrones y tomar decisiones sin la ayuda de un ser humano.

Hoy en día la rama más utilizada del aprendizaje automático es *Deep Learning* (DL), el cual utiliza capas de estructuras algorítmicas llamadas redes neuronales. Estas se inspiran directamente en el funcionamiento del cerebro humano, ya que la operación de los nodos de cálculo se comparan con el funcioamiento de las neuronas. Las técnicas de *deep learning* brindan las mejores soluciones a muchos problemas de clasificación, reconocimiento, detección y procesamiento del lenguaje natural.

A continuación se hará una descripción teórica de los algoritmos de *machine learning* que se emplean en la literatura para la detección automática de sismos.

a) Gaussian Mixtures Model- Hidden Markov-Model

La técnica *Gaussian Mixtures Model - Hidden MarkovModel* (GMM-HMM) es un modelo estadístico en el que cada estado oculto del HMM es representado por una distribución de mezcla gaussiana. El GMM-HMM está parametrizado por $\lambda = (\pi, A, B)$ donde π es un vector de probabilidades iniciales, A es una matriz de probabilidades de transición y B es una matriz de probabilidades de observación que depende de los parámetros de las distribuciones que modelan los datos. Particularmente en el caso de una distribución Gaussiana, los parámetros son (w, μ, Σ), donde w representa a los pesos Gaussianos, μ representa a los vectores de media y Σ representa a las matrices de covarianza de cada mezcla Gaussiana que modela cada estado. Dado que el modelo propuesto en este estudio se basa en DNN-HMM recomendamos revisar la sección 2.4.2 para ahondar en detalle en la definición y explicación de los HMM's. A continuación definiremos qué es una distribución de probabilidad Gaussiana y cuál es el algoritmo *Gaussian Mixtures Model*.

La distribución Gaussiana es una de las distribuciones de probabilidad más utilizada, ya que permite modelar el comportamiento estadístico de aplicaciones asociadas a la morfología [44], fisiología [45], sociología [46], entre otros. Para datos unidimensionales, la función de densidad de probabilidad de una distribución Gaussiana es

$$g(O \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(O-\mu)^2/2\sigma^2} \qquad O = (o_1, o_2 \dots o_t) \tag{2.7}$$

donde O es un vector de observaciones, μ es la media y σ^2 es la varianza. Para una función de distribución Gaussiana multidimensional (con un número d de dimensiones) la función de densidad de probabilidad está dada por

$$g(O) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} e^{\left[-\frac{1}{2}(O-\mu)^T\Sigma^{-1}(O-\mu)\right]} \tag{2.8}$$

donde μ es el promedio de cada dimensión y Σ es la matriz de covarianza que describe el grado de correlación entre las variables.

Gaussian Mixtures Model es una función de densidad de probabilidad (pdf) paramétrica que modela conjuntos de datos mediante sumas ponderadas de M funciones de distribución gaussianas [47]. La ecuación que modela la pdf es la siguiente:

$$\mathbb{P}(o_t) = \sum_{i=1}^{M} w_i g\left(o_t \mid \mu_i, \Sigma_i\right) = \sum_{i=1}^{M} w_i \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} e^{\left[-\frac{1}{2}(o_t-\mu_i)^T\Sigma_i^{-1}(o_t-\mu_i)\right]} \tag{2.9}$$

donde w_i es un peso gaussiano que satisface $\sum_{i=1}^{M} w_i = 1$. Se han desarrollado varios procedimientos para determinar los parámetros μ y Σ de un modelo de mezcla gaussiana a partir de un conjunto de datos. En la sección 2.4.2.7 se puede encontrar una revisión de como se calculan estos parámetros utilizando el algoritmo EM.

b) Redes Neuronales Convolucionales

La idea básica detrás de las redes neuronales convolucionales (CNN o ConvNets) es tener una solución que permita tener una red profunda con la menor cantidad de parámetros posible [48]. Las CNN's se han aplicado tanto a series temporales (una cuadrícula 1-D) como a imágenes (una cuadrícula 2-D)).

La popularidad del uso de la CNN se debe a los siguientes factores:

- Las características importantes se pueden aprender automáticamente, por lo que no es necesario un proceso previo para la extracción de información [49].
- Se pueden reutilizar redes preexistentes sobre tareas novedosas que utilizan otros conjuntos de datos. [50]
- Generan resultados de reconocimiento altamente precisos [51].
- Pueden adaptarse a diferentes tamaños de entrada [52].

Hay varios tipos de redes neuronales convolucionales, pero todas constan de una capa de entrada, capas ocultas y una capa de salida. Las capas ocultas se denominan asi porque porque sus entradas y salidas están enmascaradas por la función de activación y la convolución final. Normalmente la CNN se compone de tres importantes bloques: convolución, agrupación y capas totalmente conectadas. Una arquitectura típica consta de repeticiones de una pila de varias capas de convolución y una capa de agrupación, seguidas de una o más capas totalmente conectadas. En la figura 2.3 se puede observar la arquitectura de una red neuronal convolucional aplicado a la clasificación de animales.

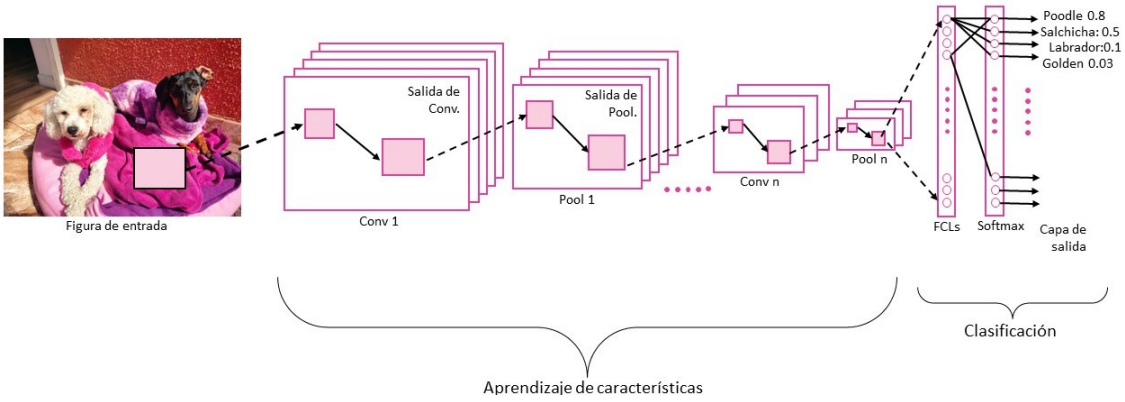


Figura 2.3: Ilustración de un ejemplo de arquitectura CNN en clasificación de razas de perros

Las capas convolucionales y las capas de agrupación realizan la extracción de caracteristicas, mientras que la capa totalmente conectada funciona como un clasificador. A continuación se describirá cada una de ellas en detalle:

- Las capas convoluciones consisten en una combinación de operaciones lineales y no lineales. La entrada a esta capa es convolucionada por un conjunto de filtros, en la cual cada uno de ellos activa ciertas características. Esta convolución reduce la cantidad de parámetros libres, lo que permite que la red sea más profunda. Las salida de esta operación lineal, se pasa a través de una función de activación no lineal. Generalmente se utiliza una capa de unidades lineales rectificadas (ReLu) que se define según la función: $f(x) = \max(0, x)$.

- Las capas de agrupación reducen las dimensiones espaciales de los mapas de características, ya que combina las salidas de grupos de neuronas aplicando una operación típica de reducción de muestreo. Las agrupaciones más comunes son el máximo y promedio. La agrupación máxima usa el valor máximo de cada grupo local de neuronas en el mapa de características, mientras que la agrupación promedio toma el valor promedio.

- Las capas totalmente conectadas tienen por entrada a un vector de números. El término "totalmente conectado" viene dado por las conexiones que hay entre cada una de las entradas con cada una de las salidas. Esta capa funciona como red neuronal perceptrón multicapa tradicional (MLP). La entrada de esta capa es la salida de la última capa de agrupación, mientras que la salida es un vector dimensional del mismo tamaño que del número de clases. A cada capa completamente conectada le sigue una función no lineal, como por ejemplo ReLU.

c) Long-Short Term Memory

Las redes *Long-Short Term Memory* (LSTM), propuesto por Hochreiter y Schmidhuber [53], son un tipo de redes neuronales recurrentes (RNN) que pueden modelar dependencias de largo plazo. A diferencia de otras arquitecturas de redes neuronales, LSTM no necesita que la secuencia de vectores de observación tenga la misma duración temporal, lo que a su vez es muy conveniente para capturar la dinámica de los eventos sísmicos. Estas redes neuronales fueron diseñadas para superar el desvanecimiento de gradiente presentado en las RNN's tradicionales. La solución a esta problema se basa en el uso de unidades de capa oculta conocidas

como celdas de memoria. La arquitectura de las celdas LSTM se compone de tres puertas o *gates*: puerta de olvido, puerta de entrada y puerta de salida.

Estas celdas actúan sobre las señales que reciben y, al igual que los nodos de la red neuronal, bloquean o transmiten información en función de su importancia, aprendiendo qué información almacenar en la memoria, cuánto tiempo almacenarla y cuándo leerla [54].

Para entender el funcionamiento de la LSTM, se estudiará el cálculo de un paso de tiempo representado gráficamente por la figura 2.4. Esta celda tiene una entrada (x_t), un estado oculto (h_t) y una salida adicional, conocida como estado de memoria (C_t). Este último elemento añade o elimina la información de la memoria de la red.

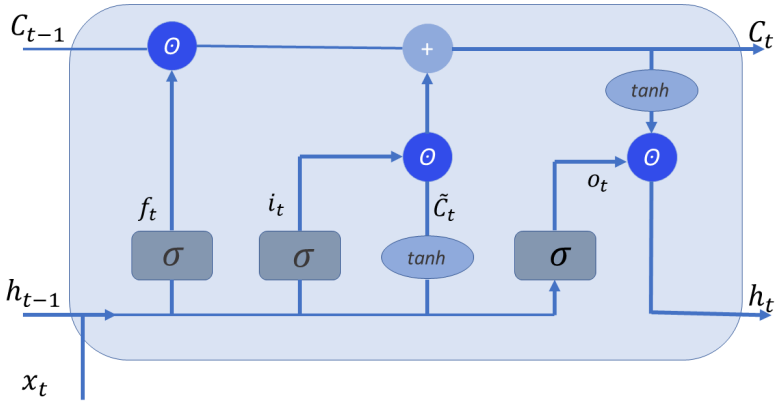


Figura 2.4: Celda de memoria para un paso de tiempo

La **puerta de olvido** decide que información se descarta, y por lo tanto, decide que información no pasa a la celda de estado.

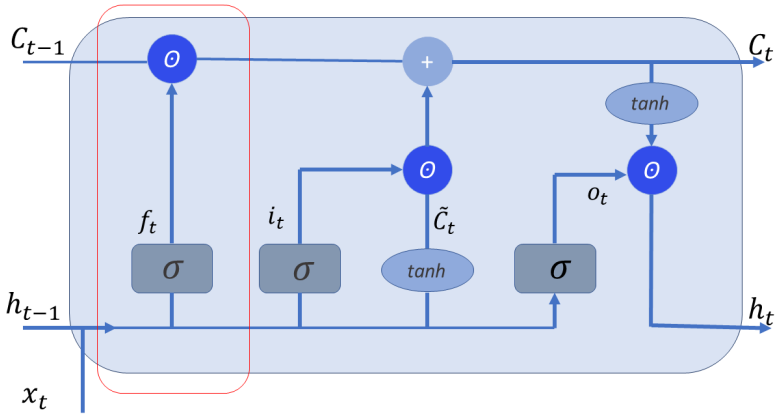


Figura 2.5: Celda de memoria para un paso de tiempo. El recuadro rojo encierra la celda de olvido.

Como se puede ver en la figura 2.5, en el flujo inicial de una celda de memoria se aplica una función de activación sigmoidea sobre la entrada actual x_t y sobre el estado oculto anterior

h_{t-1}. Si el valor de salida está más cerca de 0 se olvida, por otro lado si el valor de salida está más cerca de 1 se recuerda.

$$f_t = \sigma\left(W_f\left[h_{t-1}; x_t\right] + b_f\right) \tag{2.10}$$

La salida de la puerta de olvido está representado por la ecuación 2.10. Los coeficientes W_f y b_f se aprenden en el entrenamiento. Luego la ecuación 2.10 es multiplicado punto a punto con C_{t-1} para eliminar la información redundante de la celda de estado.

La **puerta de entrada** agrega nueva información a la celda, y selecciona la información más importante.

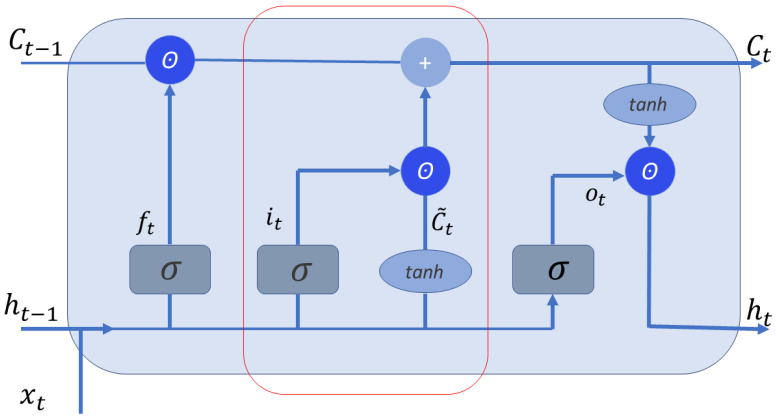


Figura 2.6: Celda de memoria para un paso de tiempo. El recuadro rojo encierra la celda de entrada.

Como se observa en la figura 2.6, se toma la entrada actual x_t y el estado oculto anterior h_{t-1} para transformarlos y pasarlos por una función de activación sigmoidal.

$$\mathrm{i}_t = \sigma\left(W_{\mathrm{i}}\left[h_{t-1}; x_t\right] + b_{\mathrm{i}}\right) \tag{2.11}$$

Los coeficientes W_i y b_i también se aprenden en el entrenamiento. En este caso los valores que se quieren preservar en la red serán cercanos a 1. A continuación se crea un vector de valores candidatos, dado por la ecuación 2.12, para formar parte de la nueva memoria.

$$\widetilde{C}_t = \tanh\left(W_c\left[h_{t-1}; x_t\right] + b_c\right) \tag{2.12}$$

Nuevamente los parámetros W_c y b_c se aprenden en el entrenamiento. Para filtrar los valores de la compuerta de entrada, se hace una multiplicación punto a punto entre i_t y $\widetilde{C}_t$. Finalmente para actualizar la memoria, preservando el pasado e incorporando nueva información, se calcula C_t, el cual está dado por

$$C_t = C_{t-1} \odot f_t + \widetilde{C}_t \odot \mathrm{i}_t \tag{2.13}$$

La **puerta de salida** calcula el nuevo estado oculto de la celda. La representación gráfica

esta dada por la figura 2.7.

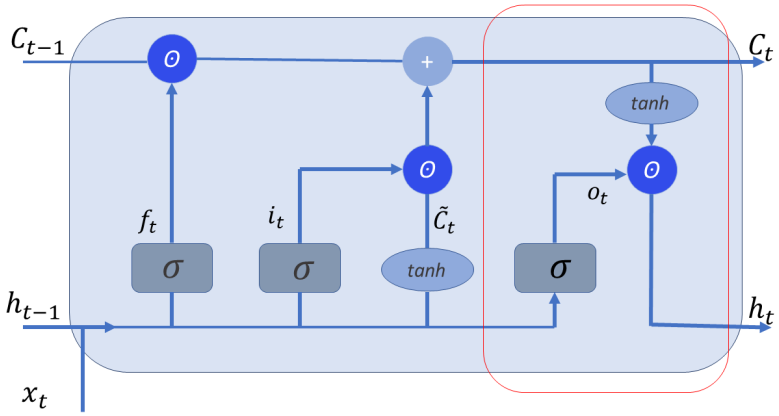


Figura 2.7: Celda de memoria para un paso de tiempo. El recuadro rojo encierra la celda de salida.

En primer lugar, ocupando la función tangente hiperbólica se escala el estado de memoria C_t para garantizar que esté en un rango de -1 a 1, . Luego se utiliza la compuerta de salida para determinar que porciones del estado de memoria pasará a formar parte del estado oculto. Las ponderaciones del estado de memoria están representadas por la ecuación 2.14.

$$o_t = \sigma\left(W_o\left[h_{t-1}; x_t\right] + b_o\right) \tag{2.14}$$

Nuevamente los parámetros W_o y b_o son aprendidos en el entrenamiento. Finalmente para obtener el estado actual, se utiliza la ecuación 2.15, en la cual se filtran los valores generados por el estado de memoria con los valores generados por la compuerta de salida.

$$h_t = \tanh\left(C_t\right) \odot o_t \tag{2.15}$$

d) Redes Transformer

Las redes *Transformers* se introducen con el articulo *Attention is all you need*, en donde se presenta una arquitectura codificador-decodificador basada en capas de atención [55]. Las principales ventajas de estos modelo son los cálculos globales y la perfecta memoria, lo que los hace más adecuados que los RNN en secuencias largas [56]. Los *transformers* ahora están reemplazando a los RNN en muchos problemas de procesamiento del lenguaje natural, procesamiento del habla y visión por computadora [57–62]. La figura 2.8 representa la arquitectura del modelo *transformer*.

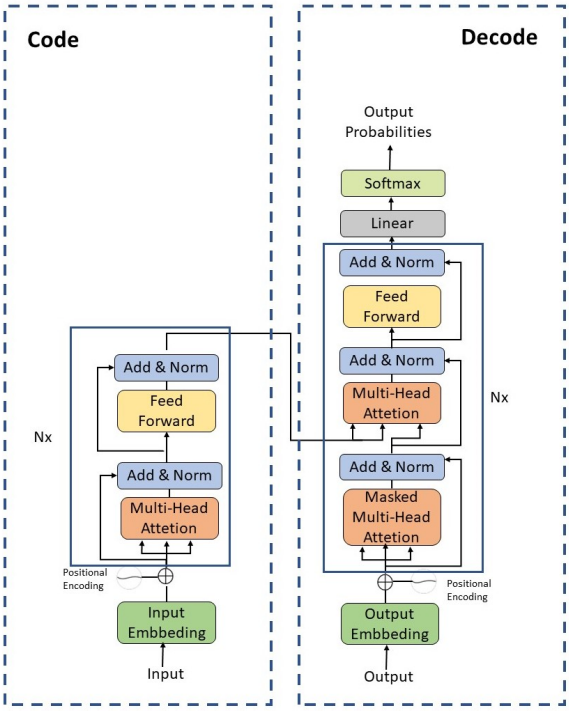


Figura 2.8: Arquitectura del modelo Transformer

El bloque de la izquierda de la figura 2.8 corresponde al **codificador**. Su tarea es mapear una secuencia de entrada a una secuencia de representaciones continuas para alimentar a un decodificador. Para entender su arquitectura, se revisará secuencialmente cada componente del bloque. En este caso vamos a considerar que la entrada es una secuencia de elementos.

La primera capa de este bloque es una capa de incrustación, la cual aplica una transformación en función del significado de cada elemento de la secuencia de entrada, y lo convierte en un vector. Posteriormente dado que cada elemento tiene significados diferentes en base al contexto, se utiliza codificadores posicionales para añadir información posicional.

El bloque de *Multi Head-Attention* genera múltiples vectores de atención por elemento de la secuencia y toma un promedio ponderado para calcular el vector de atención final. Lo anterior captura la relación contextual entre los elementos de la secuencia.

El siguiente paso es la Red Neuronal *Feed Forward*. Su función es transformar los vectores de atención para que la siguiente capa, ya sea codificador o decodificador, lo pueda recibir. Gracias a que los vectores de atención son independientes entre si, es posible aplicar paralelización en este bloque y así obtener simultáneamente conjunto de vectores codificados para cada elemento.

El bloque derecho de la figura 2.8 corresponde al **decodificador**. Este bloque recibe la salida del codificador junto con la salida del decodificador en el paso de tiempo anterior, para generar una secuencia de salida. Al igual que en el bloque anterior, se explicará secuencialmente cada componente.

La entrada del decodificador son las etiquetas de la secuencia que ingresa al codificador. Al igual que en el codificador, inicialmente se tiene una capa de incrustación y un codificador posicional.

La salida anterior pasa por un bloque llamado *Masked Multi Head-Attention*, donde se generan vectores de atención que relacionan contextualmente los elementos de la secuencia. Los vectores resultantes de esta capa y los vectores del bloque codificador son las entradas para otro bloque llamado *Multi Head-Attention*. Este último hace el mapeo de los elementos de entrada con las etiquetas y descubre la relación entre ellas. La salida de esta celda son vectores de atención para cada etiqueta y elemento de la secuencia.

Posteriormente cada vector de atención pasa por una unidad de avance, la cual transforma la salida para que las siguientes capas, ya sea decodificador o una capa lineal, puede recibirla. Finalmente la salida es convertida a una distribución de probabilidad por medio de una capa Softmax. Como resultado se tiene que la etiqueta más probable produce la probabilidad más alta.

Tanto en el codificador como en el decodificador hay capas de **autoatención**, las cuales se constituyen por unidades de *Scaled Dot-Product Attention*. Para cada unidad de atención, el modelo aprende tres matrices de pesos: clave (K), consulta (Q) y valor (V). Con estas matrices se puede calcular la función de atención dada por la ecuación 2.16.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.16}$$

donde d_k es la dimensión de el vector clave.

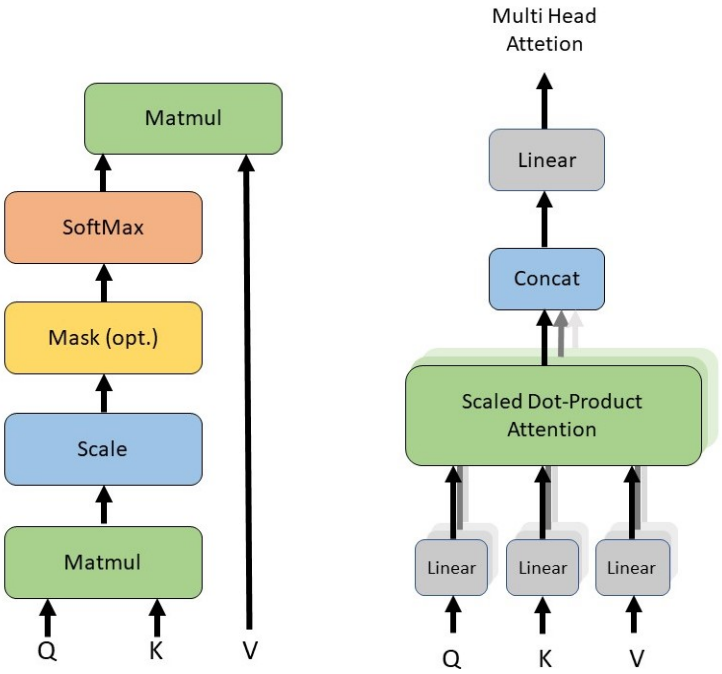
El mecanismo de *Multi Head-Attention* produce h representaciones de diferentes valores de Q, K, V y calcula una función de atención para cada representación. Las salidas se concatenan y se proyectan una vez más.

$$\text{Multihead } (Q, K, V) = (\text{ head }_1, \ldots, \text{ head }_h)\, W^O \tag{2.17}$$

$$\text{head }_i = \text{Attention } \left(QW_i^Q, KW_i^K, VW_i^V\right) \tag{2.18}$$

donde W_i^Q, W_i^K, W_i^V, W^O son matrices de pesos aprendidos.

La figura 2.9 muestra la arquitectura de *Scaled Dot-Product Attention* y *Multi Head-Attention*.



a Scaled Dot-Product Attention b Multi Head-Attention

Figura 2.9: Estructura interna del mecanismo de atención.

2.4. Deep Neural Network-Hidden Markov Model (DNN-HMM)

En esta sección se describe un sistema híbrido de un modelo oculto de Markov con una red neuronal profunda (DNN-HMM). El sistema DNN-HMM aprovecha la capacidad de aprendizaje de la DNN y la técnica de modelar secuencialmente del HMM, superando significativamente a los sistemas convencionales de modelo de mezcla gaussiana GMM-HMM [63]. A continuación describiremos la arquitectura y el procedimiento de entrenamiento del sistema híbrido DNN-HMM.

2.4.1. Deep Neural Network(DNN)

Las redes neuronales profundas (DNN) son perceptrones multicapa con muchas capas ocultas. La unidad computacional básica en una red neuronal es el perceptrón (ver figura 2.10).

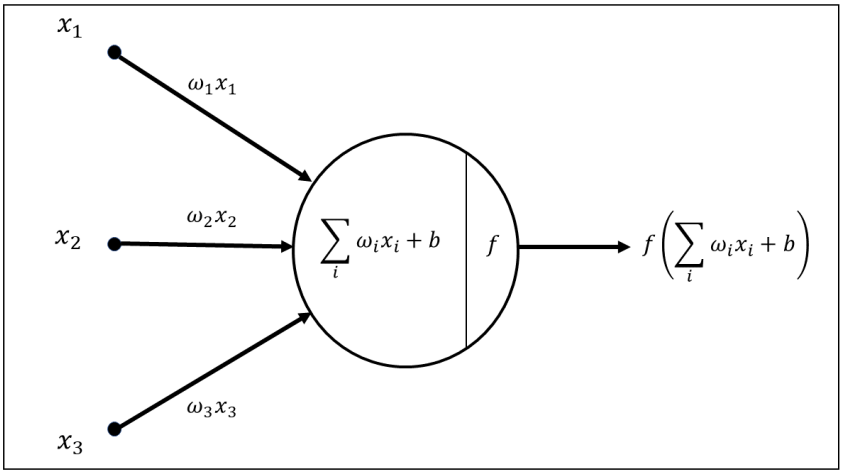


Figura 2.10: Modelo de Perceptrón

El perceptrón normalmente tiene múltiples entradas y una sola salida. Cada entrada de la neurona x_i está asociada un peso w_i. Luego se realiza suma ponderada de sus entradas y se agrega un parámetro de sesgo b a la suma ponderada. El resultado es un valor escalar dado por:

$$z = b + \sum_i w_i x_i \tag{2.19}$$

Luego la neurona aplica una transformación $f(z)$, llamada función de activación, para calcular su salida

$$v = f\left(b + \sum_i w_i x_i\right) \tag{2.20}$$

La función de activación es típicamente una función no lineal. La elección de la función de activación es un parámetro ajustable.

Las DNN's se han aplicado a problemas de reconocimiento de voz y detección de objetos [64]. A continuación, describimos la arquitectura de la DNN, las funciones de activación más populares, los criterios de entrenamiento e ilustramos el algoritmo de retropropagación.

a) The Deep Neural Network Architecture

Una red neuronal profunda (DNN) es un perceptrón multicapa (MLP) convencional con muchas capas ocultas (a menudo más de dos) [63]. A continuación, expanderemos la ecuación 2.20 de una neurona a muchas capas de neuronas. Denotaremos la capa de entrada como capa 0 y la capa de salida como capa L para un DNN de $L + 1$ capa. En las primeras L capas tenemos que

$$z^\ell = W^\ell v^{\ell-1} + b^\ell \tag{2.21}$$

$$v^\ell = f\left(z^\ell\right) = f\left(W^\ell v^{\ell-1} + b^\ell\right), \text{ for } 0 < \ell < L \tag{2.22}$$

donde $z^\ell \in R^{N_\ell \times 1}, v^\ell \in R^{N_\ell \times 1}, W^\ell \in R^{N_\ell \times N_{\ell-1}}, b^\ell \in R^{N_\ell \times 1}$, y $N_\ell \in R$ son, respectivamente, el vector de excitación, el vector de activación, el peso, el bias y el vector de salida. La ecuación 2.22 es la función de activación aplicada al vector de excitación.

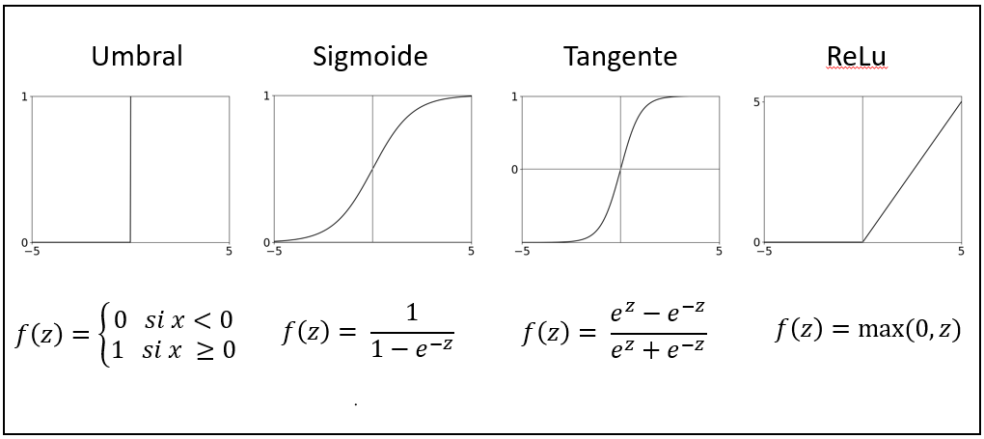


Figura 2.11: Funciones de activación

Como se puede ver en la figura 2.11, hay una variedad de opciones para la función de activación. La selección de esta debe ser en función de los requerimientos del modelo, pero generalmente la función más utilizada es la sigmoidea. Dado lo anterior, en las siguientes discusiones se asumirá que se utiliza la función de activación sigmoidea.

El vector de salida $v^L \in R^{N_L}$, tiene una dimensión de tamaño N_L. Para las tareas de clasificación multiclase, cada neurona de salida representa una clase $i \in 1, ..., C$, donde $C = N_L$. Finalmente se utiliza una función softmax para normalizar la excitación y representar la salida por una distribución de probabilidad multinomial válida.

$$v_i^L = \mathbb{P}_{\text{dnn}}(i \mid o) = \text{softmax}_i\left(z^L\right) = \frac{e^{z_i^L}}{\sum_{j=1}^{C} e^{z_j^L}}, \tag{2.23}$$

donde z_i^L es el i-ésimo elemento del vector de excitación z_L.

b)Algoritmo de entrenamiento

Los parámetros del modelo $\{W, b\}$ en una DNN son desconocidos y deben estimarse durante el entrenamiento mediante un criterio de optimización y un algoritmo de aprendizaje. Los criterios utilizados para calcular y optimizar los parámetros tratan siempre de minimizar una pérdida esperada [63]. Hay dos criterios de entrenamientos empíricos generalmente utilizado en el aprendizaje del modelo DNN. Para las tareas de regresión, el criterio es el error cuadrático medio (MSE) dado por

$$J_{MSE}(W, b; S) = \frac{1}{M} \sum_{m=1}^{M} J_{\text{MSE}}\left(W, b; o^m, y^m\right) \tag{2.24}$$

donde típicamente se utiliza

$$J_{\text{MSE}}(W, b; o, y) = \frac{1}{2} \left\| v^L - y \right\|^2 = \frac{1}{2} \left(v^L - y\right)^T \left(v^L - y\right) \tag{2.25}$$

Para las tareas de clasificación, el valor de (y) es una distribución de probabilidad y el criterio utilizado es la entropía cruzada (CE), dada por

$$J_{CE}(W, b; S) = \frac{1}{M} \sum_{m=1}^{M} J_{\text{CE}}\left(W, b; o^m, y^m\right) \tag{2.26}$$

donde típicamente se utiliza

$$J_{\text{CE}}(W, b; 0, y) = -\sum_{i=1}^{C} y_i \log v_i^L \tag{2.27}$$

$y_i = \mathbb{P}_{\text{emp}}(i \mid o)$ es la probabilidad calculada a partir del conjunto de entrenamiento de que la observación o pertenezca a la clase i, mientras que $v_i^L = \mathbb{P}_{\text{dnn}}(i \mid o)$ es la misma probabilidad anterior pero estimada a partir de la DNN.

Una vez definido el criterio de entrenamiento, se procede a actualizar los parámetros de la DNN mediante el algoritmo de *Backpropagation* [65]. De forma iterativa, los parámetros del modelo se pueden mejorar en función de la información del gradiente de primer orden como

$$W_{t+1}^{\ell} \leftarrow W_{\text{t}}^{\ell} - \varepsilon \Delta W_t^{\ell} \tag{2.28}$$

$$b_{t+1}^{\ell} \leftarrow b_{\text{t}}^{\ell} - \varepsilon \Delta b_t^{\ell} \tag{2.29}$$

$$\Delta W_t^{\ell} = \frac{1}{M_b} \sum_{m=1}^{M_b} \nabla_{W_t^{\ell}} J\left(W, b; o^m, y^m\right) \tag{2.30}$$

$$\Delta b_t^{\ell} = \frac{1}{M_b} \sum_{m=1}^{M_b} \nabla_{b_t^{\ell}} J\left(W, b; o^m, y^m\right) \tag{2.31}$$

donde W_t y b_t de las ecuaciones 2.28 y 2.29 respectivamente, son la matriz de pesos y el vector de sesgo en la capa después de la t-ésima actualización. El término ε es la tasa de aprendizaje. Luego las ecuaciones 2.30 y 2.31 corresponden al cálculo del gradiente de la matriz de peso promedio y al calculo del gradiente del vector de sesgo promedio en la iteración t. Se tiene que M_b es la cantidad de muestras y $\nabla_x J$ es el gradiente de J con respecto a x. Los gradientes deben ser calculados a partir de las funciones de perdidas vistas anteriormente. Por ejemplo si tomamos un problema de clasificación que utiliza el criterio de entrenamiento CE (ecuación 2.26) y una capa de salida softmax, el gradiente con respecto a la matriz de peso y bias de la capa de salida se puede derivar como:

$$\nabla_{W_t^L} J_{\mathrm{CE}}(W, b; o, y) = \left(v_t^L - y\right) \left(v_t^{L-1}\right)^{\mathrm{T}} \tag{2.32}$$

$$\nabla_{b_t^L} J_{\mathrm{CE}}(W, b; o, y) = \left(v_t^L - y\right) \tag{2.33}$$

Es importante recordar que las ecuaciones del algoritmo de *Backpropagation* se definen de acuerdo al objetivo de la tarea, ya que la función de perdida cambia dependiendo de si es una tarea de clasificación o una tarea de regresión.

2.4.2. Hidden Markov Model(HMM)

Los procesos que acontecen en el mundo real generalmente producen salidas observables que pueden ser caracterizadas por señales [66] que contienen información sobre su comportamiento o su naturaleza. Por lo anterior, poder caracterizar tales procesos en términos de modelos se convierte en un problema de gran interés, ya que definir un modelo proporciona información sobre la fuente de la señal sin tener acceso a ella, y permite realizar eficientes sistemas prácticos, como por ejemplo: sistemas de predicción, sistemas de reconocimiento, sistemas de identificación, entre otros.

Los modelos ocultos de Markov (HMM) es una técnica de modelización estadística diseñada para capturar la dinámica en serie, ya que permite predecir una secuencia de variables desconocidas (ocultas) a partir de un conjunto de variables observadas. La teoría básica de los HMM fue introducida por primera vez en una serie de artículos estadísticos en la década de los 70 por Leonard E. Baum y sus colegas, quienes le otorgaron el nombre de *funciones probabilísticas de cadenas de Markov* [67–69]. Posteriormente desde finales de la década de los 80 hasta la actualidad, se ha presentado una gran cantidad de aplicaciones de esta técnica en disciplinas asociadas al reconocimiento del habla [66, 70, 71], Biociencia [72], reconocimiento de gestos [73], el reconocimiento de escritura a mano [74], psicología [75], clustering de series temporales [76], bioinformática [77–79] economía y finanzas [80–82], entre otros.

La popularidad que se puede observar de los modelos Ocultos de Markov se debe a la riqueza que presentan en su estructura matemática, lo cual ha permitido que su aplicación pueda ser de uso masivo, garantizando adaptatibilidad, eficiencia computacional y facilidad para modelar procesos con variaciones dinámicas y temporales.

a) Procesos de Markov

Una cadena de Markov es un modelo estadístico introducido por el matemático ruso Andrei Andreyevich Markov [83], en el cual se modela la evolución de un proceso estocástico mediante una familia de variables aleatorias parametrizadas. En esencia, representa un proceso en tiempo discreto en el que una variable aleatoria o_t va cambiando en cada instante de tiempo. Su principal propiedad define que para cualquier secuencia temporal de eventos, la distribución de probabilidades de un evento actual o_t depende exclusivamente del evento anterior del sistema o_{t-1}.

Una cadena de Markov se puede definir como homogénea cuando las probabilidades de

transitar de un estado i a un estado j no depende del instante de tiempo, por lo tanto las probabilidades

$$\mathbb{P}\left(o_n = s_j \mid o_{n-1} = s_i\right) = \mathbb{P}\left(o_t = s_j \mid o_{t-1} = s_i\right), \quad \forall s, t \in \mathbb{N} \tag{2.34}$$

son las mismas en cada paso de tiempo.

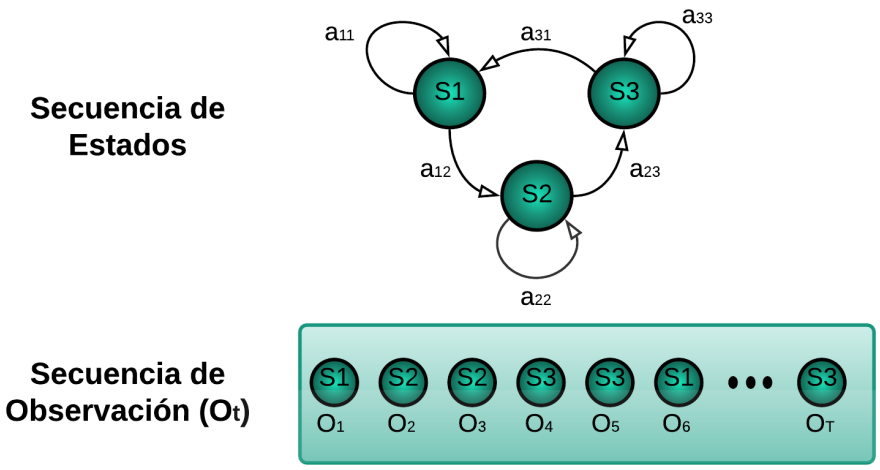


Figura 2.12: Diagrama de estados de un modelo de Markov

El diagrama de la figura 2.12 es un modelo de Markov con estados finitos que representa un proceso aleatorio en el tiempo con intervalos discretos. Según este grafo, si en un instante dado $(t-1)$ ocurre un evento, en el siguiente instante (t) se efectuará una transición de estados según una probabilidad a_{ij}. Entonces, si consideramos una cadena homogénea con m posibles estados (en este ejemplo se tiene $m = 3$), cuyos instantes de tiempos están denotados por $t = 1, 2, 3...T$, se tiene que la notación de que la cadena esté en un estado cualquiera en cada etapa t es o_t. A partir de lo anterior, es posible presentar esta secuencia de estados generada aleatoriamente mediante una *secuencia de observación* denotada por $O = \{o_1, o_2, o_3, o_4, o_5, ..., o_T\}$. Por ejemplo, según la secuencia de observación de la figura 2.12 se tiene que los estados en los primeros 3 instantes de tiempos son $o_1 = s_1$, $o_2 = s_2$ y $o_3 = s_2$.

La sucesión de estados en una secuencia de observación, se van realizando de acuerdo a un conjunto de probabilidades condicionales relacionadas a los estados previos del instante de tiempo que se está analizando. Esta propiedad determina lo que se llama *probabilidad de transición* y se denota como:

$$a_{ij} = \mathbb{P}\left(o_t = s_j \mid o_{t-1} = s_i\right), \qquad 1 \leq i, j \leq m \tag{2.35}$$

donde a_{ij} se define como la probabilidad de que la cadena que pase al estado s_j desde el estado s_i, en cualquier etapa.. La expresión anterior se utiliza para cadenas de Markov de primer orden.

Suponiendo que el sistema tiene un número finito N de estados, decimos que la cadena de Markov es finita, lo cual permite que las probabilidades de transición entre estados, a_{ij}, se

combinen formando una matriz de transición A de tamaño $N \times N$

$$A = [a_{ij}] = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{NN} \end{bmatrix} \tag{2.36}$$

Para cada fila de s_i, la serie de valores de a_{ij} es una distribución de probabilidad, ya que en cualquier instante de tiempo puede ocurrir alguno de los sucesos s_1, s_2,...,s_N, los cuales son mutuamente excluyentes. Las propiedades que satisfacen las matrices de transición son las siguientes:

1. Todas sus entradas son positivas, $a_{ij} \geq 0$, $\forall i, j$.
2. Como cada fila de la matriz es una distribución de probabilidad, la suma de sus coeficientes en cada fila equivale a 1, es decir:

$$\sum_{j=1}^{N} a_{ij} = 1, \quad \forall i = 1, \ldots, N \tag{2.37}$$

En general la cadena de Markov determina en que estado s_i se efectúa el punto de partida considerando una distribución de probabilidad del estado inicial sobre el espacio de estados. La distribución inicial de la cadena se representa en forma de vector,

$$\mathbb{P}^{(0)} = \left(a_1^{(0)}, \ldots, a_i^{(0)}, \ldots, a_N^{(0)}\right) \tag{2.38}$$

donde $\mathbb{P}_i^{(0)} = \mathbb{P}\left(o_0 = s_i\right)$. En el vector anterior cada componente indica la probabilidad de que la cadena inicie en el estado s_i. Esta distribución inicial también cumple que $a_i^{(0)} \geq 0$ y $\sum_{i=1}^{N} a_i^{(0)} = 1$.

b) Formalización de un HMM

Un modelo oculto de Markov (HMM) es un tipo de modelo estadístico que proviene de la cadena de Markov. Como vimos anteriormente en una cadena de Markov cada estado es visible por el observador y corresponde a un evento observable, mientras que en una cadena oculta de Markov los estados son ocultos y la observación es una función probabilística del estado. En base a lo anterior, un HMM puede ser entendido como un proceso doblemente estocástico, siendo el primero un proceso subyacente que consiste en un conjunto finito de estados ocultos secuenciales que no son directamente visibles, y sólo pueden ser observados a través de otro conjunto de procesos estocásticos que producen la secuencia de observación, los cuales son físicamente observables y tienen una función de distribución de probabilidad dependiendo del estado en el que se encuentren en un determinado instante de tiempo.

Según [66, 70, 71] un HMM discreto se define formalmente por 5 elementos fundamentales:

I. **Número de estados (N)**

Se define $\mathrm{S} = \{s_1, s_2, ..., s_N\}$ como el conjunto finitos de estados ocultos del HMM, donde N es la cantidad de estados en el modelo. Para muchas aplicaciones prácticas suele haber algún significado físico asociado a los estados o a los conjuntos de estados del modelo determinado por el usuario. Denotaremos a $\mathrm{Q} = \{q_1, q_2, ..., q_T\}$ como una secuencia de estados finitas. cuyo largo es igual a la cantidad de observaciones. El estado actual en el instante de tiempo t se denotará como q_t

II. **Número de observaciones distintas por estado (M)**

Se define $\mathrm{V} = \{v_1, v_2, ..., v_M\}$ como el conjunto finito de posibles símbolos observables, donde M es el número de símbolos que podrían tomar las observaciones que se presentan en el proceso a modelar.

Por consiguiente, en la secuencia de observaciones que se genera del proceso, definido como $\mathrm{O} = \{o_1, o_2, ..., o_T\}$ (donde T es la cantidad total de observaciones), cada observación o_t está relacionado a un elemento del conjunto V por medio de la cuantización vectorial.

III. **Matriz de distribución de probabilidad inicial de los estados (π)**

Se define $\pi = \{\pi_1, \pi_2, ..., \pi_N\}$ como la función de distribución de probabilidad del estado inicial, donde

$$\pi_i = \mathbb{P}(q_1 = i)$$

lo cual constituye la probabilidad de que el sistema comience en el estado *si*.

IV. **Matriz de distribución de probabilidades entre estados (A)**

Se define $\mathrm{A} = \{a_{ij}\}$ como la matriz distribución de probabilidades de transición de estados de dimensión $N \times N$, en la cual cada uno de sus elementos cumple que

$$a_{ij} = \mathbb{P}\left[q_{t+1} = j \mid q_t = i\right] \qquad 1 \leq i, j \leq N$$

V. **Matriz de distribución de probabilidad de las observaciones (B)**

Se define $B = \{b_i\,(k)\}$ como la matriz distribución de probabilidad de estar en cualquier estado en un instante de tiempo determinado. Considerando que esta matriz tiene una dimensión de $N \times M$, se tiene que el símbolo de observación en el estado i está dado por

$$b_i(k) = \mathbb{P}\left[o_t = v_k \mid q_t = i\right], \quad \begin{array}{l} 1 \leq i \leq N \\ 1 \leq k \leq M \end{array}$$

donde v_k denota el k-ésimo símbolo , y o_t el actual vector de observaciones. Estas probabilidades cumplen que

$$\begin{aligned} & b_i(k) \geq 0 \\ & \sum_{k=1}^{M} b_i(k) = 1 \end{aligned}$$

La PDF más común y exitosa utilizada en el procesamiento del habla para caracterizar la distribución de probabilidad de observación continua en el HMM es una distribución gaussiana de mezcla multivariada para valores vectoriales de observación:

$$b_i\left(o_t\right) = \sum_{i=1}^{M} w_i \frac{1}{(2\pi)^{d/2}|\Sigma_{i,m}|^{1/2}} e^{\left[-\frac{1}{2}(o_t-\mu_{i,m})^T \Sigma^{-1}(o_t-\mu_{i,m})\right]} \tag{2.39}$$

Entonces, en la práctica un HMM requiere la especificación de los parámetros estructurales del modelo N y M, la especificación de los símbolos de observación y la especificación de las medidas de probabilidad A, B y π. Por conveniencia, la notación del grupo de parámetros de medida probabilística será:

$$\lambda = (A, B, \pi)$$

en el cual en un modelo tradicional de HMM, λ es constante en el tiempo. Siguiendo esta formalización se puede ver un ejemplo de HMM en la figura 2.13.

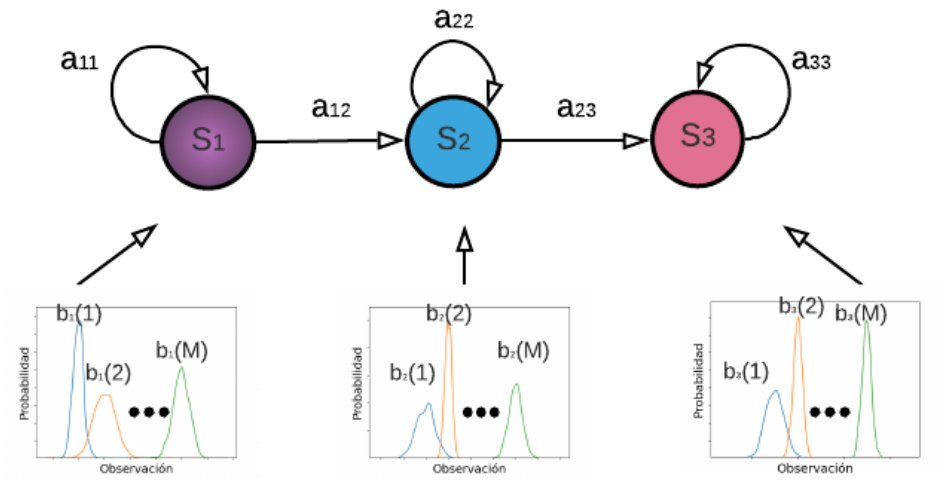


Figura 2.13: Representación gráfica de un HMM de N = 3 estados con, con M mezclas gaussianas por estado. Este es un caso particular de HMM denominado izquierda-derecha (left - right)

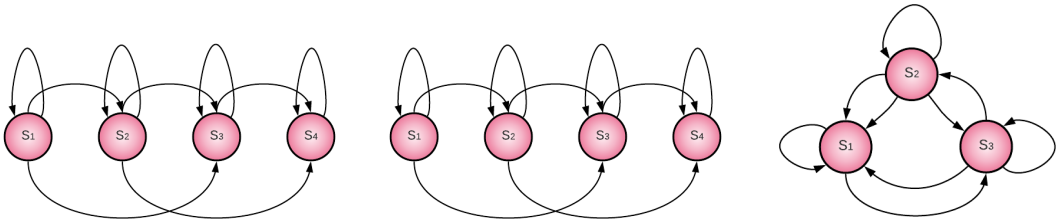
Finalmente para generar una secuencia de observaciones O, se detallará a continuación mediante los siguientes pasos el ciclo del funcionamiento de un HMM.

Algoritmo 2.1: Generación de una secuencia de observación

1 De acuerdo con la distrinución incial π, se debe eligir un estado inicial $q_1 = i$, con $1 \leq i \leq N$.
2 Hacer $t = 1$.
3 Se debe seleccionar $o_t = v_k$ de acuerdo a la distribución de probabilidad del símbolo $b_i(k)$ en el estado i.
4 Transitar a un nuevo estado $q_{t+1} = j$ de acuerdo a la distribución de probabilidad de las transiciones de estado (a_{ij}).
5 Asignar $t = t + 1$: Si $(t < T)$ devolverse al paso 3; en caso contrario entregar $O = \{o_1, o_2, ..., o_T\}$ y terminar.

c) Topologías de los HMM

La topología que presentan los modelos ocultos de Markov depende de la cantidad de estados que lo integra y de las transiciones que se dan entre dichos estados. Para una elección correcta de esta estructura probabilística, es necesario conocer directamente la naturaleza de la aplicación que tiene el problema. Es importante mencionar que al elegir una estructura, debe existir un equilibrio entre la simplicidad y la complejidad, ya que un modelo demasiado simple tendría complicaciones para generar el conjunto de datos con alta probabilidad, mientras que un modelo demasiado complejo tendría problemas para generalizar bien la información de los datos. La figura 2.14 presenta las topologías más utilizadas en HMM.



a Left-to-right model b Parallelleft-to-right model c Ergodic model

Figura 2.14: 3 diferentes topologías de modelos ocultos de Markov

En la topología de izquierda a derecha, como se muestra en la figura 2.14.a y 2.14.b, las transiciones de estado ocurren hacia adelante o hacia sí mismo a medida que aumenta el tiempo, prohibiendo aquellas transiciones que retroceden a estados pasados. Esto indica que la matriz de transición A es triangular superior:

$$a_{ij} \geq 0 \quad \forall i > j$$

Por otro lado en la topología ergódica, como se muestra en la figura 2.14.c, todos sus estados ocultos son no nulos, no periódicos y recurrentes. En otras palabras, las conexiones de los estados del HMM están completamente habilitados para que se puedan realizar transiciones a un estado desde cualquier otro estado. Esto indica que las probabilidades de la matriz de

transición A son:

$$a_{ij} > 0 \quad \forall i, j$$

d) Los tres problemas fundamentales de los HMM

Para poder aplicar HMM en la vida real, es necesario tener en cuenta que existen tres problemas fundamentales [66, 70, 71], los cuales son:

- **Problema I:** Dada una secuencia de observación O $= \{o_1, o_2, ..., o_T\}$ y un modelo $\lambda = (A, B, \pi)$, surge la pregunta de cómo calcular eficientemente la probabilidad de la secuencia de observación para un modelo fijo, es decir, la probabilidad $\mathbb{P}(O \mid \lambda)$.
- **Problema II:** Dada una secuencia de observación O $= \{o_1, o_2, ..., o_T\}$ y un modelo $\lambda = (A, B, \pi)$, surge la pregunta de cómo encontrar la sucesión de estados ocultos q $= \{q_1, q_2, ..., q_T\}$ y un modelo $\lambda = (A, B, \pi)$ más factible que puede haber generado dicha secuencias de observaciones.
- **Problema III:** Dada una secuencia de observación O $= \{o_1, o_2, ..., o_T\}$, surge la pregunta de cómo estimar los parámetros del modelo $\lambda = (A, B, \pi)$ para maximizar $\mathbb{P}(O \mid \lambda)$.

El primero constituye un problema de evaluación, y la resolución a este problema provee la probabilidad de que la secuencia de observación O $= \{o_1, o_2, ..., o_T\}$ sea generada por el modelo analizado. El segundo problema constituye un problema de decodificación, ya que tiene por tarea descubrir la parte oculta del modelo. Por último, el tercer problema constituye un problema de entrenamiento, ya que su tarea es adecuar en forma óptima los parámetros de un modelo para cada aplicación específica en función de las secuencias de entrenamiento correspondientes.

A continuación se abordarán la resolución eficiente para cada los problemas mencionados, tratando en detalle los algoritmos necesarios.

e) Solución al Problema I

Este problema se conoce como un problema de evaluación, ya que para solucionarlo se requiere calcular la probabilidad de una secuencia de observaciones O $= \{o_1, o_2, ..., o_T\}$, dado un modelo λ. Una manera más directa de calcular estas probabilidades, es enumerando toda posible secuencia de estados de largo T, donde la Probabilidad de O, dada la secuencia q y el modelo $\lambda = (A, B, \pi)$ es:

$$\mathbb{P}(O \mid Q, \lambda) = b_{q_1}(\mathbf{o}_1) \cdot b_{q_2}(\mathbf{o}_2) \cdot \ldots \cdot b_{q_T}(\mathbf{o}_T) \tag{2.40}$$

donde se asume independencia entre observaciones. Por otro lado la probabilidad de la secuencia de estado Q se puede encontrar como

$$\mathbb{P}(Q \mid \lambda) = \pi_{q_1} \cdot a_{q_1 q_2} \cdot a_{q_2 q_3} \ldots a_{q_{T-1} q_T} \tag{2.41}$$

La probabilidad conjunta de O y Q, es decir, la probabilidad de que O y Q ocurran simultáneamente, es el producto de los dos términos anteriores, es decir:

$$\mathbb{P}(O, Q \mid \lambda) = \mathbb{P}(O \mid Q, \lambda) \cdot \mathbb{P}(Q \mid \lambda) \tag{2.42}$$

Finalmente para obtener $\mathbb{P}(O \mid \lambda)$ se debe sumar los cálculos de la ecuación 2.42 para cada secuencia posible de estados de tamaño T del modelo:

$$\mathbb{P}(O \mid \lambda) = \sum_{\forall Q} \mathbb{P}(O \mid Q, \lambda) \cdot \mathbb{P}(Q \mid \lambda) \tag{2.43}$$

$$\tag{2.44}$$

o equivalentemente

$$\mathbb{P}(O \mid \lambda) = \sum_{\forall Q} \prod_{t=1}^{T} a_{q_{t-1} q_t} \cdot b_{q_t}(o_t)$$

Notar que si N es el número de estados del modelo, se necesita $(2T-1)N^T$ multiplicaciones y $N^T - 1$ sumas [71]. Dado el crecimiento exponencial de los cálculos en función de la longitud de secuencia T, resolver computacionalmente estos cálculos es un proceso inviable. A causa de esto, se requiere un procedimiento más eficiente para la resolución de este problema de evaluación. Para resolver lo anterior, se utiliza el *Algoritmo Forward* y el *Algoritmo Backward*. Para más detalles de estos algoritmos, se recomienda revisar los Anexos.

f) Solución al Problema II

A este problema se le conoce un problema de decodificación, ya que su solución corresponde a encontrar la secuencia *óptima* de estados para una secuencia de observación O dado un modelo λ. A diferencia del Problema I, no se puede encontrar una solución exacta que defina una única secuencia de estados, pues existen diferentes criterios de optimalidad que encuentran la secuencia más probable de acuerdo a las secuencias de observaciones dadas [66]. La dificultad recae en definir cual es la secuencia de estados óptima.

Una posible solución radica en descubrir la mejor secuencia de estados considerando íntegramente todos los instantes de tiempo, es decir, la secuencia de estados $Q = q_1, q_2, ..., q_N$ que maximiza $\mathbb{P}(Q \mid O, \lambda)$. Afortunadamente, para dicho fin existe un método basado en la programación dinámica conocido como Algoritmo de Viterbi, el cual es una técnica formal ampliamente utilizado para encontrar la mejor trayectoria de las secuencias de estados. Para más detalles de este algoritmo, se recomienda revisar los Anexos.

g) Solución al Problema III

Este problema se asocia a un problema de entrenamiento, pues se busca ajustar de forma óptima los parámetros del modelo λ con el fin de maximizar la probabilidad de que las observaciones vengan de dicho modelo, en otras palabras, se busca ajustar los parámetros (A, B, π) para maximizar las probabilidades $\mathbb{P}(O \mid \lambda)$.

Este problema es el más difícil de los tres, ya que no existe un método analítico para encontrar el máximo global de los parámetros del modelo. Sin embargo, existe un algoritmo de iteración conocido como Baum-Welch el cual maximiza localmente las probabilidades $\mathbb{P}(O \mid \lambda)$. La derivación de las ecuaciones de reestimación utilizadas en el método Baum-Welch se encuentran en Anexos. También es posible encontrar en los Anexos la estimación de los parámetros de máxima verosimilitud y, en particular, la aplicación del algoritmo EM para resolver el problema de aprendizaje del HMM.

h) Dificultades de implementación de los HMM

A continuación se mencionarán de manera general soluciones analíticas y empíricas que resuelven las algunas limitaciones presentes en los algoritmos que solucionan los tres problemas fundamentales de los HMM.

I. **Escalamiento** [66]: Al implementar un HMM es posible que con largas secuencias de observaciones se produzcan problemas computacionales, ya que a menudo los cálculos de las variables de los algoritmos *Forward*, *Backward* y Viterbi dan como resultado probabilidades extremadamente pequeñas, produciendo lo que se conoce como subdesbordamiento de punto flotante. Esta inestabilidad numérica se resuelve aplicando logaritmo sobre el algoritmo de Viterbi y un escalamiento sobre las variables Forward y Backward.

II. **Estimación de los parámetros iniciales [66]**: Para facilitar la convergencia en el algoritmo de Baum-Welch, es de gran importancia asignar desde un principio valores adecuados para los parámetros del modelo λ. A pesar de que no existe una respuesta exacta para esto, la experiencia empírica permite dar un par de alineamientos [66, 71]. Se ha demostrado que π y A se pueden inicializar de manera aleatoria o uniforme, produciendo reestimaciones útiles de estos parámetros en casi todos los casos. En el caso de los parámetros B, las buenas estimaciones iniciales pueden producirse mediante diferentes métodos: segmentación manual de las secuencias de observación en estados con un promedio de las ocurrencias de las mismas en dichos estados, segmentación de máxima verosimilitud de observaciones con promedio, segmentación con agrupamiento utilizando el algoritmo *K-means*, entre otros.

III. **Datos de entrenamiento insuficientes**[84]: Un problema que afecta al entrenamiento de los parámetros de un HMM es cuando no se dispone de datos suficientes, ya que se corre el riesgo de no obtener estimaciones óptimas en los parámetros. Algunas soluciones para este problema son los siguientes: Incrementar el tamaño de las secuencias de observaciones de entrenamiento, reducir el tamaño del modelo, interpolar un conjunto de estimaciones de parámetros con otro conjunto de estimaciones de parámetros de un modelo para el que existe una cantidad adecuada de datos de entrenamiento, añadir restricciones adicionales a los parámetros del modelo, entre otros.

IV. **Elección del modelo**[66]: Este problema está enfocado a la elección de la topología del modelo, a la elección tamaño del modelo (número de estados) y a la elección de los símbolos de observación.De acuerdo a [66, 71] no existe una forma simple y teóricamente correcta de hacer tales elecciones, si no más bien las elecciones dependerán del tipo de señal que se está modelando.

2.4.3. Procedimiento de entrenamiento para DNN-HMM

En una arquitectura DNN-HMM, la dinámica de la señal se modela con HMM y las probabilidades de observación se estiman a través de una DNN. El primer paso en el entrenamiento de una DNN-HMM es entrenar un sistema GMM-HMM, generando una alineación forzada a nivel de estado. Lo anterior se lleva a cabo utilizando el algoritmo de Viterbi forzado sobre el conjunto de entrenamiento. Posteriormente, para el entrenamiento de la DNN, se utiliza como entrada las alineaciones generadas por el GMM-HMM. Con el modelo GMM-HMM también podemos generar un HMM, que contiene las mismas probabilidades de transición de estado que se usan en el sistema DNN. Es importante entrenar un buen sistema GMM-HMM como modelo inicial, ya que la calidad de la etiqueta generada por el entrenamiento del GMM-HMM DNN puede afectar el rendimiento del sistema DNN. Finalmente cada neurona de salida de la DNN es entrenada para estimar la probabilidad a posteriori del estado de los HMM de densidad continua dadas las observaciones.

A diferencia de un sistema GMM-HMM que usa un GMM diferente para modelar cada uno de los estados, un sistema DNN-HMM entrena una sola DNN para estimar la probabilidad a posteriori $\mathbb{P}\left(q_t = s \mid o_t\right)$ de todos los estados $s \in [1, S]$. Además para explotar la información del contexto, la entrada a la DNN normalmente no es un solo frame, sino que es un conjunto de frames que están en torno a un frame central.

Dado que en el proceso de decodificación el algoritmo de Viterbi requiere el likelihood $\mathbb{P}\left(o_t \mid q_t\right)$ en lugar de la probabilidad a posteriori, se necesita convertir $\mathbb{P}\left(q_t = s \mid o_t\right)$ a la forma

$$\mathbb{P}\left(o_t \mid q_t = s\right) = \mathbb{P}\left(q_t = s \mid x_t\right) \mathbb{P}\left(o_t\right) / \mathbb{P}(s) \tag{2.45}$$

donde $\mathbb{P}(s) = \frac{T_s}{T}$ es la probabilidad estimada del conjunto de entrenamiento de cada estado, T_s es el número de frames etiquetados como estado s y T es el número total de frames. $\mathbb{P}\left(o_t\right)$ es independiente de la secuencia de palabras y, por lo tanto, puede ignorarse en el proceso de decodificación. Esto conduce a que se pueda utilizar un likelihood escalado de la forma $\mathbb{P}\left(o_t \mid q_t\right) = \mathbb{P}\left(q_t = s \mid o_t\right) / \mathbb{P}(s)$ [85].

2.5. Ejemplos de sistemas de detección sísmica con ML

Desde principios de la década del 2000, muchos de los trabajos de investigación que han utilizado HMM en el monitoreo sísmico, han tenido como objetivo la detección automática de avalanchas de nieve [86], clasificación de microsismos en minas subterráneas [87], detección de deslizamientos de tierra [88], detección y clasificación eventos sísmicos de origen vulcanológico [29, 89–93], detección y clasificación de eventos sísmicos de origen sismológico [94, 95] [96–98], entre otros.

Varios de los trabajos sismológicos, han basado sus aportes únicamente en la parametrización utilizada sin considerar en los modelos la duración de los eventos sísmicos. Por ejemplo, en [99] se propone el uso de transformadas wavelet para la extracción de características de un sistema de clasificación basado en GMM-HMM, el cual se aplica sobre eventos sísmicos de origen volcánico. En [87] presentan un sistema GMM-HMM para la clasificación automática de

microsismos en minas subterráneas utilizando Mel-Frequency Cepstral Coficients (MFCC), que a su vez había sido aplicado con éxito al reconocimiento de voz y clasificación de eventos volcánicos [100]. En [97], se emplearon características obtenidas a partir de técnicas basadas en análisis de polarización, análisis de trazas complejas y estimación de sonogramas [101].

A pesar de que los HMM's suelen aplicarse en un esquema *plug-and-play* y no consideran información sobre la duración de los eventos analizados, existen algunas excepciones. En [100], el número de estados del HMM es equivalente a la duración mínima del evento observado. Sin embargo, no se considera la duración máxima de estos eventos. En [96], se propuso un sistema de detección basado en GMM-HMM donde los estados se agrupan según la similitud de las gaussianas. En [98], se emplea un modelo semi-oculto de Markov (HSMM) con transductores de estado finito ponderados (WFST) para construir un decodificador que aplica criterios de longitud mínima y máxima para los eventos sísmicos. Los valores de duración mínimo y máximo se eligen de acuerdo a ciertos deciles de la distribución de duración del estado. Una desventaja de este modelo es que la decodificación WFST presenta costos computacionales muy altos y que los criterios de longitud no representan la duración natural de los segmentos del terremoto. En [92], se presenta un sistema de detección automática de eventos volcánicos basado en GMM-HMM con modelos de estado y duración de eventos. La incorporación de restricciones de estado y duración de eventos redujo la tasa de falsos positivos hasta en un 31 %, con una precisión de verdaderos positivos igual al 94 %.

Se han propuesto varios métodos basados en el aprendizaje profundo para resolver problemas sismológicos. Los sitemas basados en CNN se han utilizado con bastante éxito para abordar el problema de la selección de fases [102–107]. En cuanto a la detección de terremotos, se han propuesto diferentes métodos. En [41], se presentó ConvNetQuake, una red convolucional profunda para la detección y localización de terremotos a partir de una sola forma de onda. Al estudiar la sismicidad inducida en Oklahoma, EEUU, el sistema pudo detectar 17 veces más terremotos que los catalogados previamente por el Servicio Geológico de Oklahoma. Una limitación de esta metodología es el requerimiento de una cantidad considerable de señales de entrenamiento para obtener un buen desempeño en la detección y localización de terremotos. En [108], se adaptó un marco CNN para proponer ConvNetQuake_INGV, un método para la detección de terremotos locales y la clasificación epicentral utilizando formas de onda de una sola estación. Los resultados obtenidos de este estudio son la caracterización de sismos en un amplio rango de distancias y magnitudes, logrando una precisión del 87 %. Esta técnica presenta un error de predicción importante para las estaciones individuales, probablemente causado por la inestabilidad que dan las CNN respecto a ligeras variaciones en los datos de entrada [109, 110]. En [111], se presenta ArrayConvNet, una arquitectura basada en CNN que emplea datos continuos de una red sísmica para detectar y localizar eventos. Esta metodología se probó con eventos en Hawái y logró una precisión de detección del $99,4\,\%$ y una baja diferencia de kilometraje en la predicción de ubicación con las referencias del catálogo del Servicio Geológico de EEUU. Esta técnica requiere una gran cantidad de datos de entrenamiento y una red de estaciones sísmicas. En [112], se propuso SCALODEEP, un método de aprendizaje profundo para la detección generalizada de terremotos. El sistema SCALODEEP presenta 24,629,053 parámetros y emplea una representación de frecuencia de tiempo de los datos de entrada y saltos en conexiones de red profundas para mejorar el rendimiento de generalización. Esta técnica está restringida por la resolución limitada de la gráfica de tiempo-frecuencia. En [113] se propuso una arquitectura de *end-to-end* que consta

de tres subredes para el procesamiento de múltiples estaciones de formas de onda sísmicas. La primera subred extrae características de formas de onda sin procesar; el segundo realiza una fase de selección de ondas P y S; y la tercera subred realiza la detección de eventos basada en la asociación de características de varias estaciones en una red sísmica. Este enfoque presenta problemas en el estudio de la microsismicidad ya que la detección de un pequeño terremoto a menudo no se observa en múltiples estaciones.

En [38] presentan a un detector de terremotos llamado CRED. Este método utiliza una combinación de capas convolucionales y unidades de memoria bidireccionales de largo a corto plazo (LSTM). La arquitectura de la red utiliza capas convolucionales, recurrentes y totalmente conectada, en una estructura residual.

Las entradas utilizada para la red corresponden a los espectrogramas de los sismogramas de las componentes Z, E y N. Luego, las capas convolucionales de la CNN procesan estos espectrogramas para aprender y extraer automáticamente sus características. La salida que se obtiene de estas capas entran a las capas LSTM bidireccionales, las cuales ayudan en el aprendizaje y modelación del patrón secuencial de datos. Finalmente la salida de las capas LSTM alimentan a las capas finales totalmente conectadas, las cuales asignan las clases de salida deseadas. Con esta arquitectura, CRED alcanza a tener 256K parámetros entrenables. El modelo aprendido tiene baja sensibilidad al nivel de ruido de fondo y logra una precisión superior al 96 %.

En [2] se propuso el sistema EarthQuake Transformer (EQTransformer), una red de transformadores basada en la atención para detectar y seleccionar eventos de fase. Esta técnica presenta dos niveles de atención: global (detección de eventos) y local (selección de fase). La arquitectura de la red consiste en un codificador profundo y tres decodificadores separados.

Tanto el codificador como los decodificadores están compuestos por convoluciones 1D, LSTM bidireccionales y unidireccionales, capas hacia adelante, red en red, conexiones residuales, capas de alimentación hacia adelante, transformador y capas de autoatención. En este sistema, el codificador genera información contextual sobre las dependencias temporales de las señales sísmicas. La sección final de este codificador son unidades de atención global, las cuales dirigen la atención de la red a los segmentos de sismos en la señal. Por otro lado, los decodificadores utilizan esta información contextual para mapear características y asignar probabilidades en cada instante de tiempo a la existencia de un sismo y a las fases P y S. En el inicio de los decodificadores asociados a las fases, existe una unidad de atención local que intenta dirigir la atención hacia las características locales de las fases sísmicas que hay dentro de la señal. Esta red emplea 56 capas con 372K parámetros entrenables. Al aplicar el sistema a 5 semanas de datos continuos registrados durante los terremotos de Tottori en Japón en el año 2000, fue posible detectar el doble de terremotos que estaban registrados en el catálogo Nacional. También se obtuvo que el modelo es capaz de seleccionar las fases P y S con una precisión cercana a las selecciones manuales realizadas por analistas humanos.

EQTransformer y CRED son métodos computacionalmente exigentes, por lo que están limitados por la capacidad de la GPU. Como se discutió aquí, la mayoría de los métodos basados en el aprendizaje profundo para la detección de terremotos son redes complejas que usan muchos parámetros para ser entrenados, lo que significa que estos modelos requieren

extensas bases de datos de entrenamiento. Los parámetros entrenables empleados por los clasificadores basados en aprendizaje profundo mencionados anteriormente son al menos del orden de magnitud de 10^5, lo que a su vez requiere grandes conjuntos de datos para evitar el sobreajuste [41].

Capítulo 3

Sistema DNN-HMM con modelos de duración de estados y eventos

En este capítulo se presenta la técnica utilizada para desarrollar el sistema de detección automática de sismos. También se describe la base de datos y los escenarios experimentales que se utilizaron para evaluar el desempeño de la técnica propuesta.

3.1. Técnica propuesta

En este trabajo se propone un sistema *end-to-end* basado en DNN-HMM con modelos de duración de estados y eventos para la detección automática de señales sísmicas. Los parámetros de los modelos de duración se obtienen con los datos de entrenamiento. Cada señal registrada en una estación sísmica se representa como una secuencia de dos eventos: sismo y ruido. Los eventos sísmicos y el ruido se modelaron con una topología HMM de tres estados de izquierda a derecha sin saltos de transición entre estados no consecutivos.

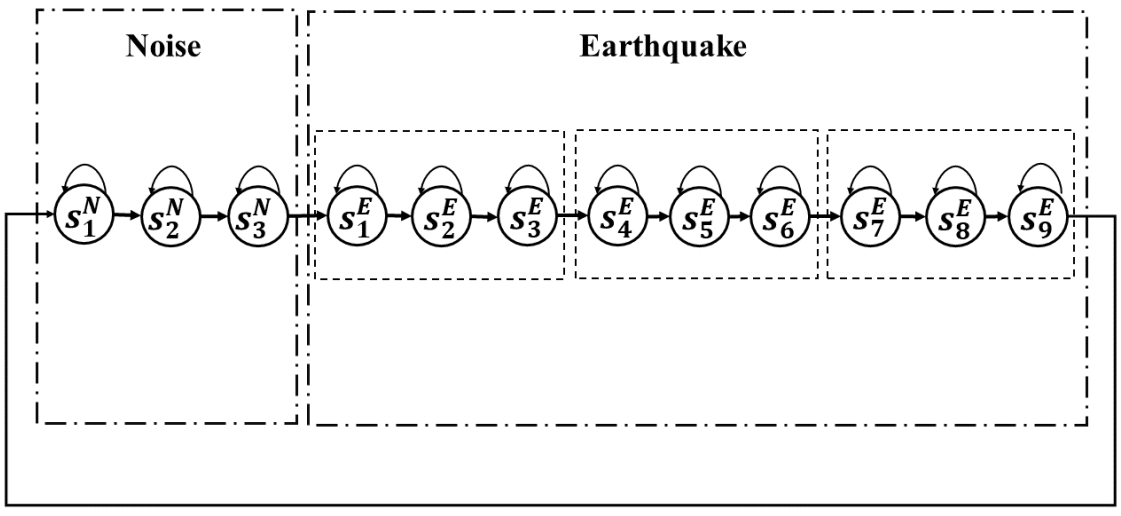


Figura 3.1: Topología del sistema HMM

La Figura 3.1 muestra la topología utilizada en el modelo propuesto. Como se puede observar, los sismos se modelaron con tres HMM's de tres estados (es decir, un total de nueve estados), los cuales representan la llegada de la onda P, las ondas S y superficiales y la coda. Por otro lado, el ruido de fondo se modeló con un HMM de tres estados. En cuanto al HMM del terremoto, los tres primeros estados representan la onda P, los estados cuatro a seis corresponden a las ondas S y superficiales, y los estados siete a nueve modelan la coda. De acuerdo con la red propuesta, un evento sísmico siempre va precedido y seguido únicamente por intervalos de ruido.

En el modelo híbrido DNN-HMM propuesto, se entrena una red neuronal profunda (DNN) para estimar la probabilidad de observación dado un frame y su contexto (ver Figura 3.2.

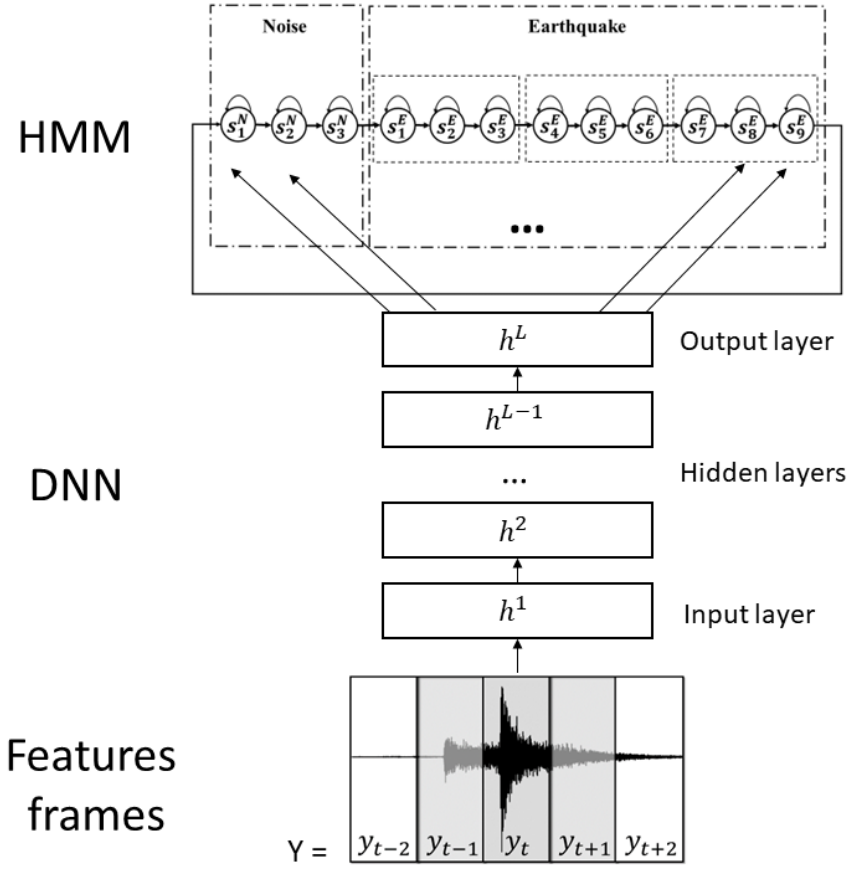


Figura 3.2: Arquitectura del sistema DNN-HMM

En la Figura 3.3 se ilustra de la arquitectura los procesos de entrenamiento y decodificación. En el proceso de entrenamiento, se extraen las características de las señales para entrenar los HMM's, obteniendo como resultado que cada señal se convierte en una secuencia de vectores de características o frames. Además, los parámetros de duración de estados y eventos se estiman empleando el algoritmo de Viterbi forzado en los datos de entrenamiento una vez entrenado los HMM's. En el proceso de decodificación, se extraen las características de cada señal de prueba para generar una secuencia de frames o vectores de características. Luego, se obtiene la alineación óptima entre la secuencia de frames de una señal y los estados del HMM mediante el algoritmo de Viterbi. En nuestro sistema, el algoritmo de Viterbi in-

corpora las restricciones de duración de estado y evento. Obsérvese que la alineación óptima proporcionada por el algoritmo de Viterbi apunta a proporcionar la detección de terremotos a partir de la posición en el tiempo de las ondas P. En consecuencia, el procedimiento de decodificación es tanto un procedimiento de detección como de clasificación, ya que proporciona la secuencia más probable de eventos sísmicos e intervalos de ruido. Los componentes del sistema se describen a continuación.

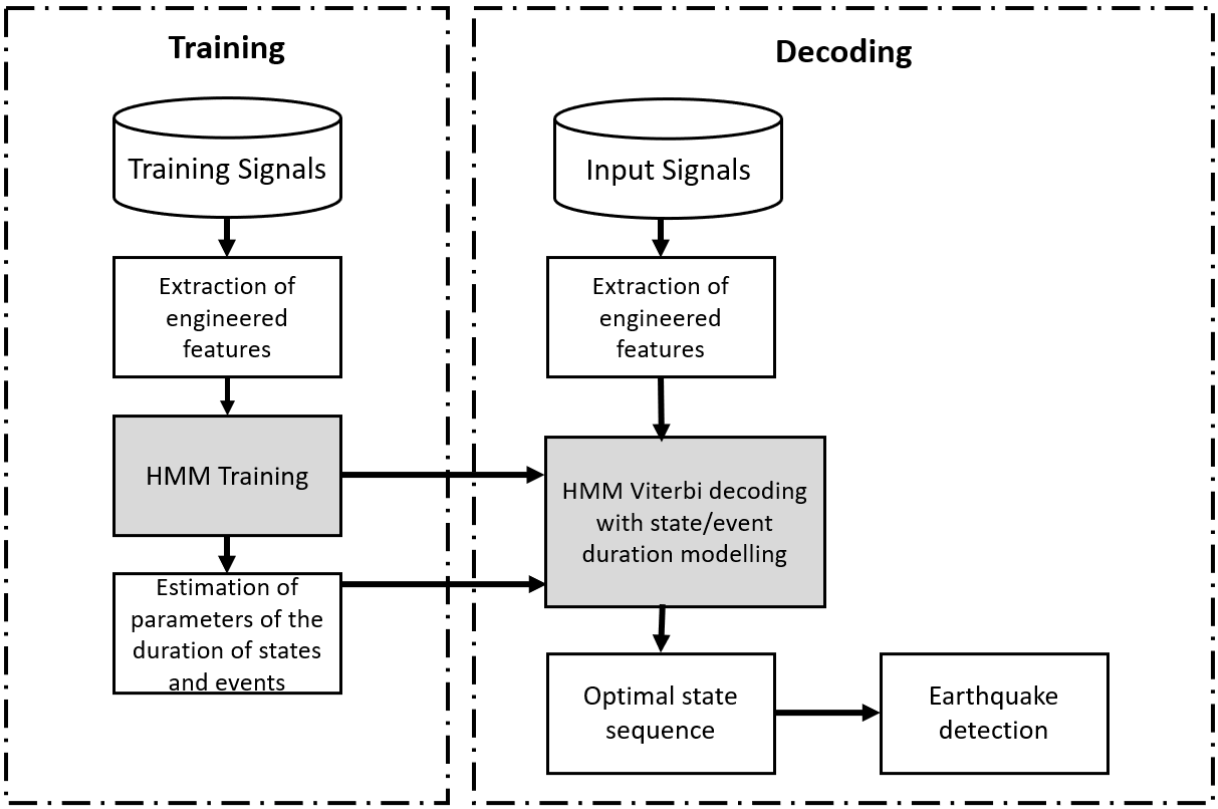


Figura 3.3: Arquitectura global del sistema propuesto

3.1.1. Extracción de características

La extracción de características utilizada se basó en el análisis de espectro y energía a corto plazo. En primer lugar, los datos en cuenta se convierten a velocidad, eliminando la respuesta instrumental. A continuación, se describe el proceso optimizado de extracción de características. El ruido de baja frecuencia se elimina de cada señal aplicando un filtro paso alto con una frecuencia de corte en 1 Hz. Tras hacer que la frecuencia de muestreo sea igual a 40 muestras por segundo, cada señal se divide en ventanas de 2 segundos con un 50 % de traslape. Se aplica una ventana de Hamming sobre cada ventana y luego se calcula la FFT de 64 muestras para obtener el espectro logarítmico en los primeros 33 intervalos de frecuencia. Este proceso se lleva a cabo para las componentes E, N y Z de cada sismograma, lo que da como resultado $33 \cdot 3 = 99$ características estáticas. Luego se incorpora una característica de energía logarítmicas por canal, es decir una energía por cada componente E,N y Z, las cuales se normalizan por el máximo de cada sismograma. Esta caracteristica se calcula a partir

de las señales enventanadas en el tiempo. Posteriormente las características estáticas son normalizadas según la media y la varianza (MVN) de cada señal. Finalmente, las funciones delta Δ y delta-delta $\Delta\Delta$ se concatenan con los parámetros estáticos. Los coeficientes Δ y $\Delta\Delta$ intentan representar la dinámica de los eventos sísmicos y se define como:

$$\Delta(i) = \frac{\sum_{i=-M}^{M} ix(i)}{\sum_{i=-M}^{M} i^2} \tag{3.1}$$

$$\Delta\Delta(i) = \frac{\sum_{i=-M}^{M} i\Delta(i)}{\sum_{i=-M}^{M} i^2} \tag{3.2}$$

donde $x(i)$ denota al vector de características estático del frame i; y M es el número de frames hacia delante o hacia atrás. En este trabajo, M es igual a dos. Por lo tanto, el número total de características estáticas por cuadro es igual a $34 \cdot 3 = 102$, que combinado con los coeficientes da un vector final de 306 características. La Figura 3.4 resume los pasos de extracción de características.

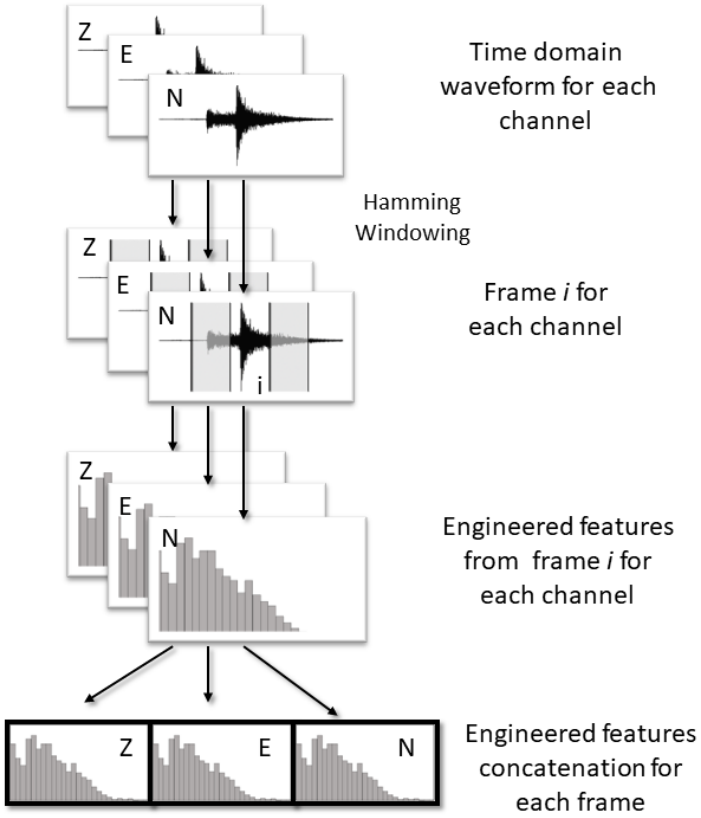


Figura 3.4: Ingeniería de extracción de características

3.1.2. Procedimiento de Entrenamiento

El proceso de entrenamiento es iterativo y se basa en el algoritmo de gradiente, lo que a su vez lo hace muy dependiente del punto inicial. Para contrarrestar esta limitación, se adoptó una solución inicial supervisada basada en la segmentación manual de las señales de entrenamiento en lugar de una condición inicial aleatoria. Esta estrategia hace que la convergencia sea más rápida y proporciona modelos más precisos.

a) Modelo inicial y entrenamiento

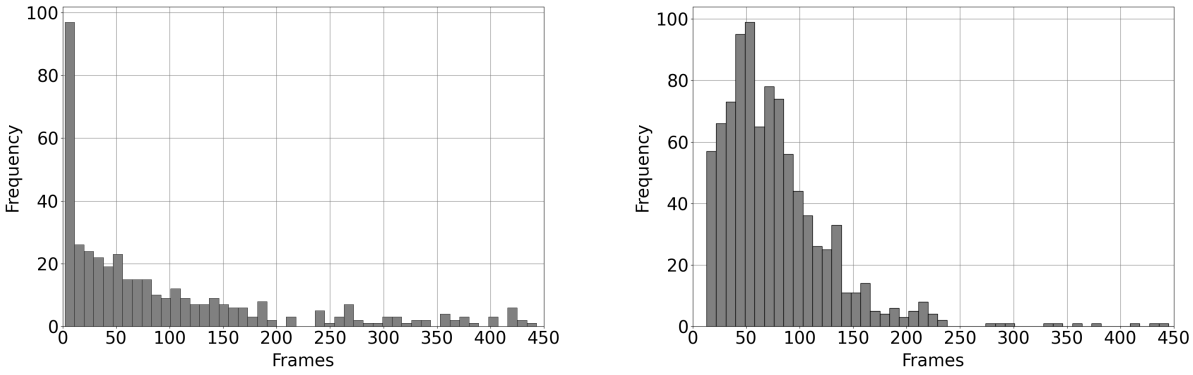
El primer sistema GMM-HMM fue entrenado con Kaldi [114], un kit de herramientas de código abierto ampliamente utilizado en el campo del reconocimiento de voz para entrenar modelos de HMM's junto a diferentes arquitecturas de redes neuronales. Esta plataforma es flexible y extensible, lo que la hace altamente personalizable. El software para entrenar el componente DNN se escribió en Python según[115]. Como se mencionó anteriormente, se propone una solución inicial basada en la segmentación manual de señales. Los vectores de características sísmicas se asignaron a los estados del modelo HMM de terremotos según el tipo de onda: primero, la onda P se divide por igual en tres segmentos y se asigna secuencialmente a los tres primeros estados (es decir, de uno a tres); segundo, la combinación de la onda S y las ondas superficiales se divide uniformemente y se asigna secuencialmente a los estados cuatro a seis; y finalmente, la onda de coda se asigna igualmente a los últimos tres estados (es decir, de siete a nueve). En consecuencia, las frames de segmento de ruido se dividieron uniformemente y se asignaron secuencialmente a los tres estados HMM de ruido. Tras la asignación inicial de frames a estados de todas las señales de entrenamiento, se determinan las distribuciones gaussianas de cada estado para estimar las probabilidades de observación. Inicialmente, estas se componen de una única distribución gaussiana. No obstante, a medida que se itera el entrenamiento, la cantidad de gaussianas por estado puede aumentar según la cantidad de frames asignados a cada estado. Cabe destacar que la reasignación de frames dentro de un GMM conduce a una reestimación de los parámetros gaussianos.En cuanto a las probabilidades de transición iniciales, $a^N_{i,j} = 0,5$ excepto $a^N_{33} = 0.75$, donde $a^N_{i,j}$ es la probabilidad de transición del estado i al estado j en el modelo de ruido. . En el modelo de terremotos, las probabilidades iniciales de transición al mismo estado y al siguiente corresponden a $a^E_{i,i} = 0.75$ y $a^E_{i,i+1} = 0.25$, donde $a^E_{i,j}$ es la probabilidad de transición del estado i al estado j.

El entrenamiento de DNN se realiza con los alineamientos generados por el sistema GMM-HMM como entrada. Esto significa que la precisión del sistema DNN-HMM resultante depende de la calidad del GMM-HMM entrenado previamente. Finalmente el sistema DNN proporciona la probabilidad de observación de cada estado en función del contexto [116]. Para más información de la salida de la DNN, se recomienda revisar la sección 2.4.3.

b) Estimación de parámetros de duración de estados y eventos

Después de entrenar el sistema DNN-HMM, se utiliza el algoritmo de Viterbi forzado para asignar cada cuadro a un estado dada una señal sísmica y la correspondiente secuencia de transcripción ruido-terremoto. A partir de esta alineación forzada, se calculan la media, desviación estándar, duraciones mínimas y máximas de los estado y eventos. La media y desviación estándar de los eventos sísmicos y ruido son, respectivamente, de 78.7 $\pm$ 59.8

frames y de 108 ± 146.2 frames. La Figura 3.5 muestra los histogramas de la duración en frames de los eventos sísmicos y los intervalos de ruido. La Figura 3.6 muestra los histogramas de las duraciones en frames de los estados de terremoto y de ruido. En ambas figuras fue empleada la base de datos *North Chile*. Cabe destacar que la mayoría de las señales de los sismogramas comienzan seis segundos antes de los primeros eventos sísmicos. Esto explica la primera barra en los histogramas de la duración del ruido. La duración considerada en los histogramas corresponde al 99 % del total de datos.

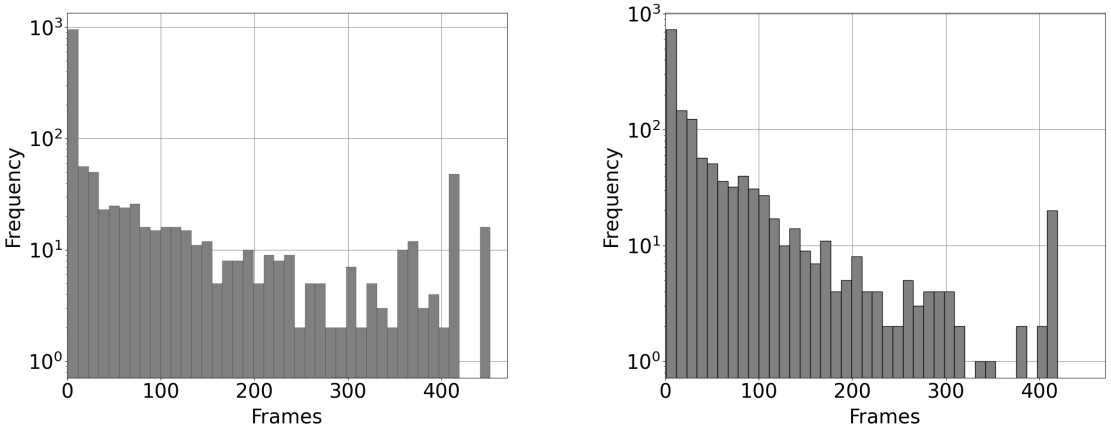


a Histogramas de ruido

b Histograma de sismos

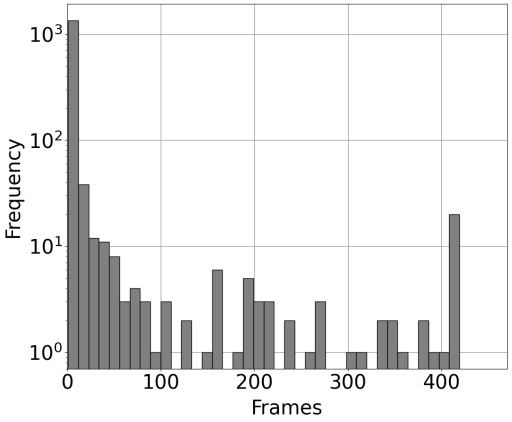
Figura 3.5: Histogramas de las duraciones en frames de los eventos sísmicos y ruidos de fondo de la base de datos North Chile

La Figura 3.6 muestra los histogramas de las duraciones en frames de los estados de los sismos y de los segmentos de ruidos de la base de datos *North Chile*.

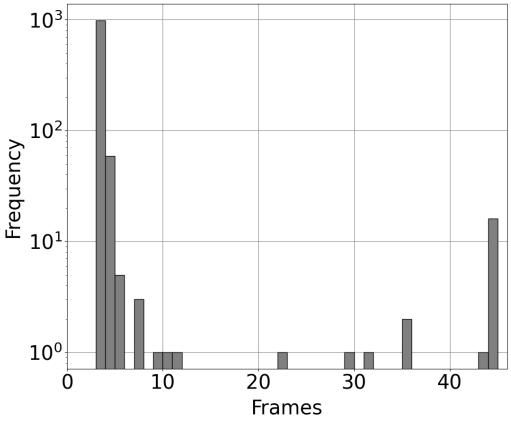


(a) Estado 1 de ruido

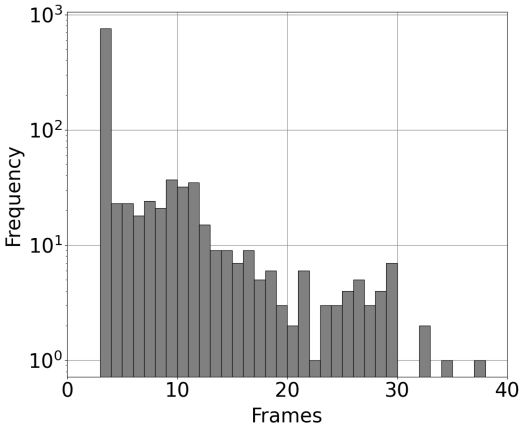
(b) Estado 2 de ruido



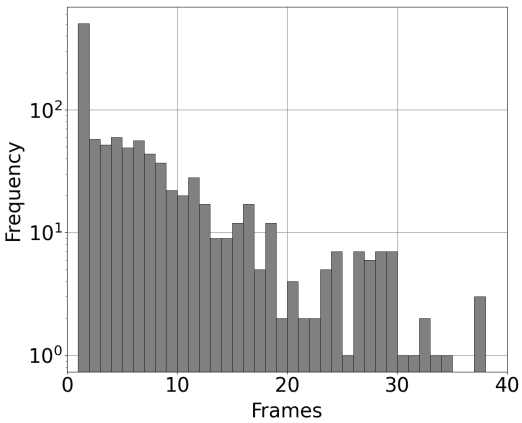
(c) Estado 3 de ruido



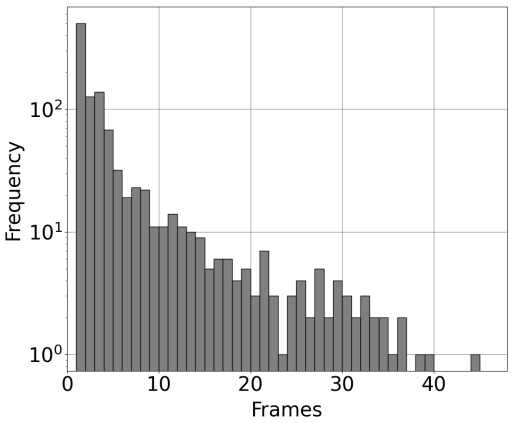
(d) Estado 1 de sismo



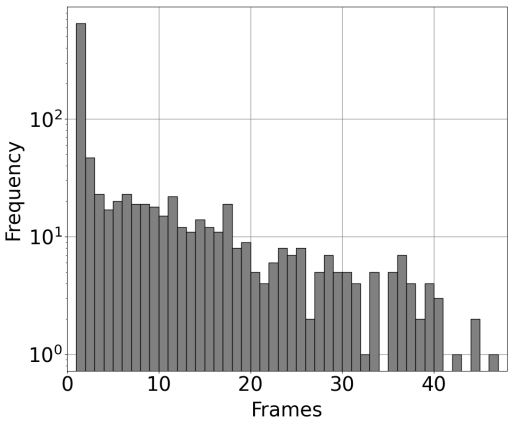
(e) Estado 2 de sismo



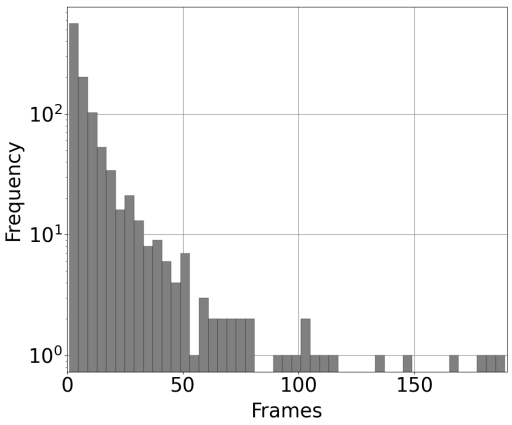
(f) Estado 3 de sismo



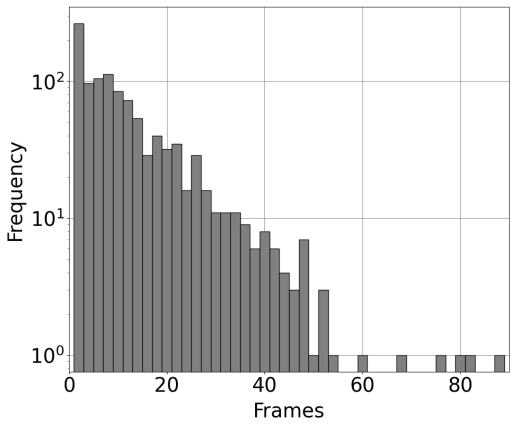
(g) Estado 4 de sismo



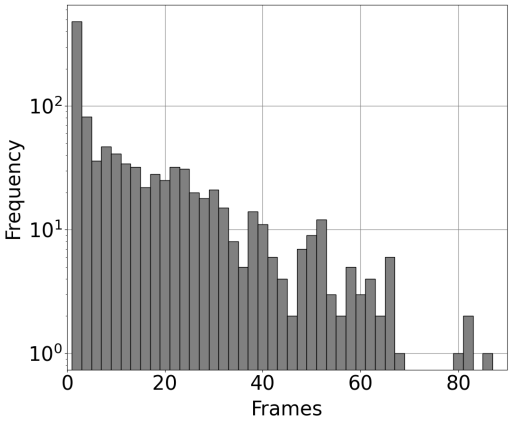
(h) Estado 5 de sismo



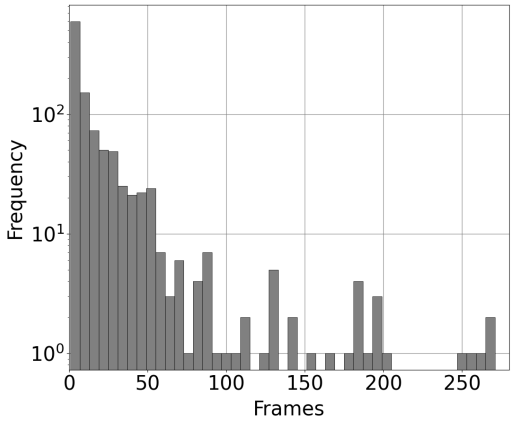
(i) Estado 6 de sismo



(j) Estado 7 de sismo



(k) Estado 8 de ruido



(l) Estado 9 de ruido

Figura 3.6: Histogramas de las duraciones en frames de los estados de sismo y ruido de la base de datos *North Chile*

3.1.3. Procedimiento de Decodificación

La etapa de decodificación consta inicialmente de calcular vectores de características para cada señal sísmica de prueba, tal como se ilustra en la Figura 3.4. Posteriormente, se estima el alineamiento óptimo mediante la aplicación del algoritmo de Viterbi [117] con modelos de duración.

a) Algoritmo de Viterbi

La decodificación de Viterbi tiene como objetivo estimar la secuencia de estados más probable que da una secuencia de observación o vectores de características. Este trabajo propone la combinación de la decodificación de Viterbi con modelos de duración de estados y eventos. El algoritmo ordinario de Viterbi decide a qué estado pertenece un frame en cada paso y elimina las alineaciones menos probables hasta que solo queda un superviviente [115].

Esta decisión se basa en la maximización de la probabilidad acumulada de una secuencia de estados y la correspondiente secuencia de vectores de observación. Cuando se evalúa la probabilidad de que un frame se asigne a un estado determinado, se tiene en cuenta el número de vectores de observación anteriores asignados al mismo estado.

b) Restricciones de duración de estado y restricciones de duración de eventos

Para incorporar modelos de duración de estado en la detección de eventos sísmicos tectónicos, se modificaron las probabilidades de transición como en ([118] y [92]). En nuestro modelo, los estados de terremoto y ruido tienen una duración máxima y mínima (Figura 3.6). En consecuencia, las probabilidades de transición dependen ahora de la duración del estado, es decir, $a_{i,i}(\tau_i)$ y $a_{i,i+1}(\tau_i)$, donde τ_i es la duración actual del estado i

$$a_{i,i}(\tau_i) = \begin{cases} 1, & \text{if } \tau_i < t_{\min\_\, i} \\ 0, & \text{if } \tau_i \geq t_{\max\_\, i} \\ a_{i,i}, & \text{otherwise} \end{cases} \tag{3.3}$$

$$a_{i,i+1}(\tau_i) = \begin{cases} 1, & \text{if } \tau_i < t_{\min\_\, i} \\ 0, & \text{if } \tau_i \geq t_{\max\_ i} \\ a_{i,i}, & \text{otherwise} \end{cases} \tag{3.4}$$

En la ecuación 3.4, si i es el último estado del modelo de terremoto, i+1 corresponde al primer estado del HMM de ruido, y viceversa. . Definimos $t_{\min_i} = t_{\min_i} \cdot \min_i(\tau_i)$ y $t_{\max_i} = tol_{\max_i} \cdot \max_i(\tau_i)$ como las duraciones de estado mínimas y máximas, respectivamente, que se permiten en la decodificación de Viterbi para cada estado i donde: $tol_{\min_i}$ y $tol_{\max_i}$ son tolerancias constantes; y, $\min_i(\tau_i)$ y $\max_i(\tau_i)$ son las duraciones mínimas y máximas estimadas, respectivamente.

También se incorpora una penalización de duración a todo el evento de sismo y ruido, es decir, penalización de duración del evento (EDP), con una función de densidad de probabilidad gamma truncada. La siguiente expresión se aplica a los eventos de sismo y ruido por separado:

$$EDP = \begin{cases} -\infty, & \text{if } \tau_e < t_{\min\_\, e} \\ -\infty, & \text{if } \tau_e \geq t_{\max\_\, e} \\ \log\left(K e^{-\alpha\tau_e} \tau_e^{\rho-1}\right), & \text{otherwise} \end{cases} \tag{3.5}$$

donde τ_e es la duración en frames de los eventos sísmicos o del evento de ruido. De la misma manera que la restricción de duración de estado, definimos $t_{\min_e} = tol_{\min_e} \cdot \min_e(\tau_e)$ y $t_{\max_e} = tol_{\max_e} \cdot \max_e(\tau_e)$, los cuales corresponden a los umbrales mínimo y máximo de duración de los eventos. Las constantes $tol_{\min_e}$ y $tol_{\min_e}$ son tolerancias para la duración mínima y máxima de cada evento- y los valores $\min_e$ y $\max_e$ son las duraciones mínima y máxima de los eventos que se estimaron a partir de los datos de entrenamiento.

Los parámetros α y ρ son valores positivos y K es un término de normalización. Los parámetros α y ρ son estimados por,

$$\alpha = \frac{E(\tau_e)}{\mathrm{Var}(\tau_e)} \tag{3.6}$$

$$\rho = \frac{E^2(\tau_e)}{\mathrm{Var}(\tau_e)} \tag{3.7}$$

donde la duración media $E(\tau_e)$, la varianza $\mathrm{Var}(\tau_e)$ y las duraciones $\min_e$ y $\max_e$ son parámetros calculados a partir de los eventos etiquetados en las señales de entrenamiento. La aplicación EDP se realiza en el dominio de probabilidad logarítmica como un término aditivo.

3.2. Descripción experimental y del sistema

En esta sección se detalla los experimentos realizados para evaluar el rendimiento del sistema propuesto.

3.2.1. Bases de Datos

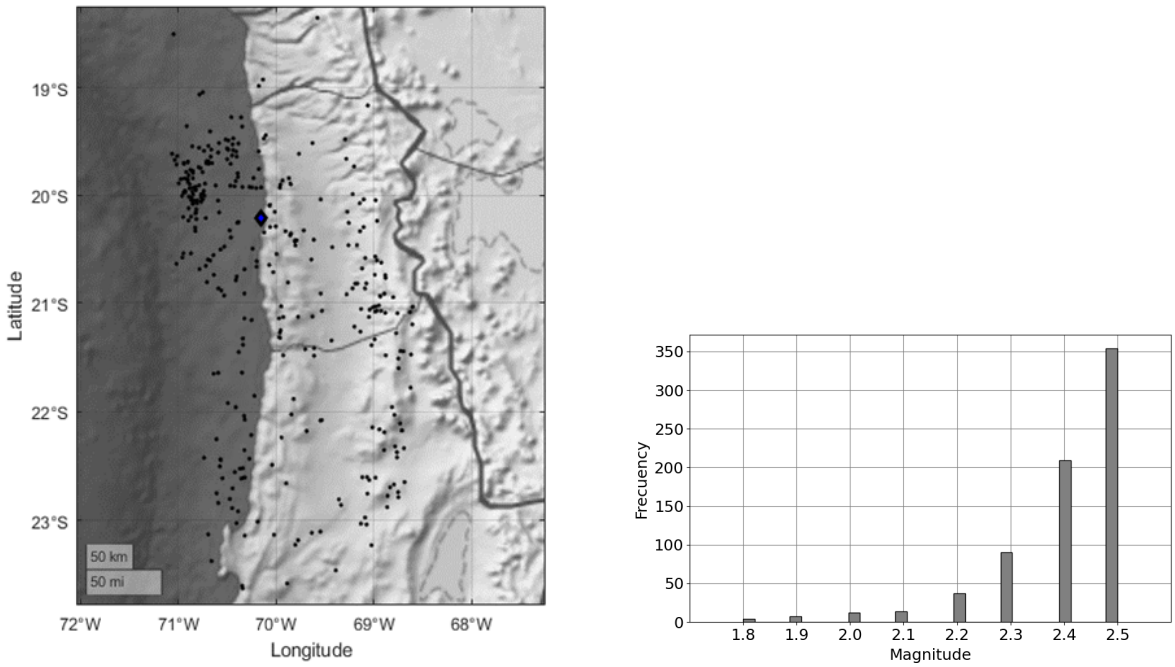
El sistema propuesto fue evaluado con tres conjuntos de datos: *North Chile*, Iquique y *Nearby Location*. Todos los sismos que componen estos conjuntos de datos fueron observados en estaciones sismológicas ubicadas en el norte de Chile. También cada sismo fue segmentado por un analista en base a la onda P, onda S, ondas superficiales y coda. Dado que estas selecciones se etiquetaron manualmente, la precisión del tiempo está sujeta al criterio del analista

La base de datos *North Chile* está compuesta por sismogramas que fueron descargados del Centro de Gestión de Datos (DMC) del consorcio de universidades Instituciones de Investigación Incorporadas para Sismología (IRIS). Este conjunto de datos está compuesto por 3,237 sismos identificados en 1,253 sismogramas entre los años 2014 y 2020. La duración de estos sismogramas varía aproximadamente entre 1 minuto y 2 horas.Los sismos fueron observados en 57 estaciones ubicadas entre las latitudes 18.23°S y 33.86°S, cuyas frecuencias de muestreo son 40 o 100 Hz. El conjunto de datos se dividió en subconjuntos de entrenamiento, validación y prueba con 67 %, 8 % y 25 % de sismogramas, respectivamente.

El conjunto de datos de Iquique es un catálogo de 13.400 sismogramas registrados de marzo a mayo del año 2014. Estos sismos corresponden a la secuencia de réplicas del terremoto de M=8,1 de Iquique, uyo epicentro fue en la localidad de Pisagua, en el norte de Chile en el año 2014. Los sismogramas se registraron con una red compuesta por 65 estaciones de banda ancha distribuidas a lo largo del norte Chile y sur del Perú. odas las estaciones emplearon una frecuencia de muestreo de 100 Hz. Los sismogramas se dividieron en subconjuntos de entrenamiento, validación y prueba correspondientes al 60 %, 10 % y 30 %, respectivamente, de toda la base de datos según [102].

Finalmente, la base de datos *Nearby Location* está conformado por sismos 725 terremotos menores que M2.5 con una distancia de epicentro/estación inferior a 100 [km]. La Figura 3.7 muestra las localizaciones y el rango de magnitudes de los terremotos en este conjunto de

datos. Estos sismos fueron catalogados por el Centro Sismológico Nacional (CSN), y la fecha de su origen fluctúa entre enero de 2014 y diciembre de 2020. Cada sismograma contiene un solo sismo y se emplearon 28 estaciones sísmicas distribuidas en el norte de Chile. Sus frecuencias de muestreo eran 40Hz o 100Hz. Esta base de datos se usó solo para pruebas.



a Ubicaciones de los epicentros de los sismos.

b Frecuencia de magnitudes

Figura 3.7: Información descriptiva de Local Database.

3.2.2. Métrica de rendimiento

Para evaluar y comparar el rendimiento del sistema de detección propuesto, se utilizan las siguientes métricas: tasa de falsos positivos, tasa de falsos negativos, precision, recall y f1-score. Si el sistema detecta como sismo un segmento de la señal que no está catalogada en la referencia, se considera como falso positivo, mientras que si existen algún sismo de la referencia que no es detectado por el sistema, se considera como una detección falso negativo. En este problema, la métrica de recall informa del número de terremotos que el modelo puede identificar y la métrica precision indica la calidad de esas detecciones. Por último, f1-score es una media armónica de precision y recall. Sus correspondientes definiciones son:

$$\text{recall} = \frac{TP}{TP + FN} \tag{3.8}$$

$$\text{precision} = \frac{TP}{TP + FP} \tag{3.9}$$

$$F1 - \text{ score} = \frac{2 \cdot \text{ precision } \cdot \text{ recall}}{\text{precision } + \text{ recall}} \tag{3.10}$$

donde TP es verdadero positivo, FP es falso positivo y FN es falso negativo.

El criterio utilizado para definir la detección de un sismo, tiene relación con la diferencia absoluta entre la detección y cualquier referencia. Si esta diferencia es inferior a un umbral (20 [s]), se considera que el sistema detectó el sismo. En el caso de que una referencia esté ligada a más de una detección, se selecciona aquella con la que tenga la menor diferencia absoluta.

3.2.3. Optimización de funciones, hiperparámetros, arquitectura y entrenamiento

Para construir un sistema robusto, se examinaron diferentes configuraciones del modelo hasta optimizar las características, los hiperparámetros y la arquitectura. Los parámetros que se evaluaron en la extracción de características fueron los siguientes: FFT de 32 a 256 bines; tamaño de la ventana de 1 a 6 segundos; solapamiento de la ventana de análisis del 15 % al 50 %; representación de la energía del frame con y sin logaritmo; normalización de las características con MVN por señal, MVN global (extracción de la media y la varianza a partir de toda la base de datos), Min-Max por señal y Min-Max global (extracción del mínimo y el máximo a partir de toda la base de datos).

En cuanto a la configuración de la red, los hiperparámetros ajustados fueron los siguientes: tamaño de la ventana de contexto, número de capas, número de neuronas y tasa de aprendizaje. Para el tamaño del contexto, se probaron de cero a cinco ventanas para atrás y adelante. Se evalúa la cantidad de uno a cinco capas ocultas y se prueba de 16 a 256 neuronas. Por último, se realizaron pruebas con valores entre 0.00001 y 0.1 para la tasa de aprendizaje. Las métricas utilizadas para evaluar el rendimiento de estos experimentos fueron precision, recall y f1-score.

3.2.4. Experimentos con restricciones de duración

Usando las porciones de entrenamiento y prueba de la base de datos North Chile, se evalúan siete versiones del sistema de detección propuesto basado en el algoritmo de Viterbi, los cuales difieren en el uso de restricciones de duración de estado/evento. Estos sistemas se describen a continuación:

- Implementación del algoritmo de Viterbi sin restricciones de duración de estados y eventos. Este sistema se denomina Viterbi_NDM (No Duration Models)
- Implementación del algoritmo de Viterbi con restricción de duración de estado según las ecuaciones 3.3, 3.4, aplicado solo al modelo de sismo. Este sistema se denomina Viterbi_DMSE (Duration Models States over Earthquake)
- Implementación del algoritmo de Viterbi con restricción de duración de evento según la ecuación 3.5, aplicado solo al modelo de sismo. Este sistema se denomina Viterbi_DMEE (Duration Models Event over Earthquake)
- Implementación del algoritmo de Viterbi con restricción de duración de estado y eventos según las ecuaciones 3.3, 3.4 y 3.5, aplicado solo al modelo de sismo. Este sistema se denomina Viterbi_DME (Duration Models over Earthquakes)

- Implementación del algoritmo de Viterbi con restricciones de duración de estado según las ecuaciones 3.3, 3.4 y 3.5, aplicados a los modelos de sismo y ruido. Este sistema se denomina como Viterbi_DMSEN (Duration Models States over Earthquake and Noise).

- Implementación del algoritmo de Viterbi con restricciones de duración de eventos según la ecuación 3.5, aplicados a los modelos de sismo y ruido. Este sistema se denomina como Viterbi_DMSEN (Duration Models Events over Earthquake and Noise).

- Implementación del algoritmo de Viterbi con restricciones de duración de estado y eventos según las ecuaciones 3.3, 3.4 y 3.5, aplicados a los modelos de sismo y ruido. Este sistema se denomina como Viterbi_DMEN (Duration Models over Earthquake and Noise).

Las tolerancias para los límites inferior y superior en (3.3) y (3.4) y (3.5) variaron entre cero y uno, y entre uno y tres, respectivamente.

3.2.5. Rendimiento entre dominios

Evaluamos y comparamos el rendimiento del sistema de detección propuesto con métodos de aprendizaje profundo que también realizan una etapa de detección. Los sistemas a comparar corresponden a EQTransformer [2] y CRED [38]. Estos sistemas se ejecutaron utilizando el conjunto de herramientas SeisBench ([119]. El modelo EQTransformer que se implementa en SeisBench está preentrenado con *STanford EArthquake Dataset* (STEAD). Esta base de datos contiene 1,2 millones de formas de onda etiquetadas, 450,000 terremotos y 19,000 horas de datos procedentes de 2613 estaciones sismológicas. Dada por la arquitectura del modelo, EQTransformer tiene definido 376,935 parámetros. Por otro lado, el modelo CRED está preentrenado con una base de datos compuesta por 550,000 sismogramas de 3 componentes y 30 segundos cada uno, registrados por 889 estaciones de banda ancha de periodo corto del norte de California. Este último modelo contiene un total de 293,596 parámetros.

En este estudio EQTransformer fue entrenado con la base de datos de Iquique. pero no así CRED. Dado que la base de datos de Iquique no posee suficientes ejemplos de formas de onda de ruido y que los intervalos de ruido antes de la llegada de la onda P son demasiado cortos, esta conjunto de datos no cumple con los requisitos para entrenar al modelo de CRED. La Tabla 1 muestra un resumen de los modelos evaluados y su número de parámetros entrenables, así como la información de las bases de datos. Para comparar el desempeño de los sistemas indicados en la Tabla 3.1, estos modelos son probados en las bases de datos del *North Chile*, Iquique y *Nearby Location* Utilizamos precision, recall y f1-score como métricas de evaluación.

Tabla 3.1: Resumen de los modelos de detección, base de datos empleada y número de parámetros entrenables.

Base de datos de entrenamiento	Duración en horas	Número de sismogramas	Cantidad de sismos	Modelo de entrenamiento	Número de parámetros entrenables
North California	4.583	550.000	250.000	CRED	293.596
STEAD	20.000	1.200.000	450.000	EQTransformer	376.935
North Chile	407	1.253	3.237	Viterbi_DME	15.514
Iquique	223	13.400	13.400	EQTransformer Viterbi_DME	376.935 15.514

3.2.6. Relación magnitud y distancia

Dada la importancia de examinar la microsismicidad local, se estudia y compara el rendimiento de los diferentes modelos sobre *Nearby Location*. El análisis de los datos microsísmicos no es fácil, ya que el ruido de fondo suele dificultar la detección de eventos de baja magnitud, lo que hace que los datos del estudio no siempre sean fiables [4]. En base a lo anterior, es fundamental diferenciar correctamente lo que es un terremoto y lo que es ruido, por lo que en este experimento se utilizará una curva de precisión para analizar el rendimiento de los modelos. El cálculo de la precisión depende del número de falsos positivos, y como estos no tienen una distancia epicentral, se distribuye por igual el total de falsos positivos entre los intervalos de distancia.

Capítulo 4

Resultados y Discusión

En este capítulo se presentan los resultados que se obtuvieron en los distintos escenarios experimentales, a modo de dar respuesta a los objetivos planteados en el capítulo 1. También presenta las discusiones sobre el desempeño de la técnica propuesta.

4.1. Optimización de funciones, hiperparámetros, arquitectura y entrenamiento

Tras una búsqueda de la parametrización del modelo que presenta el mejor desempeño, se obtuvo que la configuración óptima de las características está dado por los siguientes parámetros: FFT de 64 bines; ventana de dos segundos con un solapamiento del 50 %; energía con logaritmo; normalización MVN.

En cuanto a la optimización de hiperparámetros, se eligió la configuración que empleaba menos parámetros entre las que presentaban mejores resultados. Resumiendo, los hiperparámetros seleccionados fueron: ventana de contexto compuesta por tres frames, es decir, un frame a la izquierda y otro a la derecha con respecto al frame central; dos capas ocultas; 16 neuronas por capa; y tasa de aprendizaje igual a 0,0001. Además, el tamaño de lote y el número de épocas seleccionado es igual a 256 y 100 épocas, respectivamente. Se emplearon ReLU y Softmax como funciones ocultas y de activación de salida. Como resultado se obtuvo un modelo con solo 15.514 parámetros.

4.2. Experimentos con restricciones de duración

En la Tabla 4.1 se muestran los resultados de los 3 sistemas descritos en la sección 3.2.4. Los valores de tolerancia obtenidos de la sensibilización de parámetros son: $tol_{min_i} = 0.7$ y $tol_{min_e} = tol_{max_i} = tol_{max_e} = 1$.

Como se ilustra en la Tabla 4.1, Viterbi_NDM proporcionó un 11 % y un 16 % de falsos negativos más bajos que los den Viterbi_DME y Viterbi_DMEN, respectivamente. Esto podría ser causa de que los modelos de duración de estado y de eventos puede descartar terremotos potenciales que no cumplan con las restricciones de duración según las ecuaciones (3.3), (3.4) y (3.5), cuyas cotas están basados en las estadísticas obtenidas de los datos de entrenamiento. Por el contrario, la tasa de falsos positivos obtenida con Viterbi_NDM es

un 53 % superior a la de Viterbi_DME. Este resultado sugiere que los modelos de duración de estados ciertamente pueden filtrar eventos hipotéticos que no cumplen con las ecuaciones (3.3), (3.4) y (3.5). Observe que Viterbi_NDM proporciona una tasa de falsos positivos de un 18 % superior al de Viterbi_DMEN. Esto debe ser resultado de que la duración de los intervalos de ruido entre sismos consecutivos es poco discriminante, y su modelo es menos efectivo. Además, el ruido sísmico es un proceso aleatorio y estocástico, muy difícil de modelar.

De acuerdo con la figura 3.5.a, el histograma de las duraciones de los estados sísmicos sugiere una distribución de probabilidad unimodal que cae bruscamente desde una duración igual a 100. Sin embargo, las duraciones de los intervalos de ruido entre terremotos consecutivos están altamente concentradas en unos pocos frames y conducen a un histograma bastante plano, aproximadamente desde 10 o 20 frames. Observe que el comienzo y el final de los sismogramas corresponden a intervalos de ruido de fondo que se acortaron cuando se segmentaron las señales. Como se puede ver en la figura 3.5.b las duraciones máximas de estado en el modelo de terremoto están entre 50 y 90 frames. Por el contrario, la duración máxima del estado en el modelo de ruido puede ser de hasta 400 frames. Si consideramos la tasa de falsos negativos, el sistema Viterbi_NDM es la mejor opción y proporciona la puntuación de recuerdo más alta. Sin embargo, el esquema Viterbi_DME es el que tiene la mayor precisión y puntuaciones f1-score.

También se presentan en la Tabla 4.1 resultados de modelos que implementaron únicamente restricciones de estados y modelos que utilizaron solo restricciones de eventos. Como puede ver, Viterbi_DMSE es 11 % menor en la tasa de falsos negativos y 31 % mayor en la tasa de falsos positivos con respecto a Viterbi_DME, mientras que Viterbi_DMEE es 5 % mayor en la tasa de falsos negativos e igual en la tasa de falsos positivos que Viterbi_DME. De la misma manera, se observa que Viterbi_DMSEN es 21 % menor en la tasa de falsos negativos y 24 % mayor en la tasa de falsos positivos con respecto a Viterbi_DMEN, mientras que Viterbi_DMEEN es 5 % mayor en la tasa de falsos negativos y 35 % menor en la tasa de falsos positivos que Viterbi_DMEN. A pesar de que el f1-score no varía significativamente entre los distintos modelos, los resultados muestran que la estimación de las duraciones causan variabilidad en la calidad de las detecciones. A partir de los resultados se observa que al restringir solo las duraciones de estados, la tasa de falsos positivos es mayor que los modelos que solo restringen eventos. Esto sugiere que hubo un sobreentrenamiento de las duraciones máximas de los estado, lo cual causa que al restringir mucho los estados de ruido o de sismo, se generen muchas transiciones entre eventos, y consecuentemente, aumenten las detecciones. Esto se se visualiza con los modelos Viterbi_DMSE y Viterbi_DMSEN, los cuales solo restringen a los estados. A pesar de lo anterior, los mejores resultados se obtienen al utilizar los modelos de duración de estados y los modelos de duración de evento, por lo tanto, la integración de ambas restricciones hace que se aprovechen los aportes de cada una.

Por último, los resultados de la Tabla 4.1 destacan la ventaja conseguida gracias a la incorporación de los modelos de estado y duración del evento en los terremotos, por lo que se decide utilizar Viterbi_DME en nuestro modelo final. Dada la variabilidad de los datos sísmicos de *North Chile*, para el resto de los experimentos se emplearon los parámetros de duración estimados con este conjunto de datos.

Tabla 4.1: Comparación del rendimiento en la decodificación de Viterbi con Viterbi_NDM, Viterbi_DME con restricciones de estados (Viterbi_DMSE), Viterbi_DME con restricciones de eventos (Viterbi_DMEE), Viterbi_DME con restricciones de estados y eventos, Viterbi_DMEN con restricciones de estados (Viterbi_DMSEN), Viterbi_DMEN con restricciones de eventos (Viterbi_DMEEN), Viterbi_DMEN con restricciones de estados y eventos (Véase la sección 3.2.4)

Model	FN(%)	FP(%)	Precision	Recall	F1-Score
Viterbi_NDM	0.16	0.2	0.81	0.84	0.82
Viterbi_DMSE	0.16	0.17	0.84	0.84	0.84
Viterbi_DMEE	0.19	0.13	0.86	0.82	0.84
Viterbi_DME	0.18	0.13	0.87	0.82	0.84
Viterbi_DMSEN	0.15	0.21	0.80	0.84	0.82
Viterbi_DMEEN	0.20	0.11	0.87	0.8	0.83
Viterbi_DMEN	0.19	0.17	0.83	0.82	0.82

4.3. Rendimiento entre dominios

Las figuras 4.1, 4.2 muestran los resultados con los sistemas discutidos en la sección en la sección 3.2.5. En la figura 4.1, los modelos fueron probados con la base de datos *North Chile*. Como se puede observar, el f1-score más alto se obtuvo con el sistema Viterbi_DME entrenado con la base de datos *North Chile*, cuyo valor fue de 0,84. Este puntaje es 20 %, 342 % y 223 % superior a los obtenidos con EQTransformer entrenado con STEAD, EQ-Transformer entrenado con el conjunto de datos de Iquique y CRED entrenado con datos de California, respectivamente. Este resultado puede ser bastante sorprendente si tenemos en cuenta que las bases de datos de STEAD y California son mucho más grandes que las de Iquique y *North Chile*. Cabe destacar que STEAD contiene eventos de las localidades de Iquique y del Norte de Chile. Este resultado también corrobora la importancia de utilizar datos locales de entrenamiento y minimizar la cantidad de parámetros a entrenar. Observe que el f1-score proporcionado por Viterbi_DME entrenado con Iquique es 326 % más alto que el de EQTransformer entrenado con los mismos datos. Este resultado podría deberse a que Viterbi_DME requiere casi 24 veces menos parámetros que EQTransformer. Además, cabe señalar que la baja puntuación f1-score de CRED entrenado con los datos de California se debe principalmente a la baja puntuación de precision, que a su vez es consecuencia de una tasa de FP más alta, como se puede ver en la Fig.4.1.

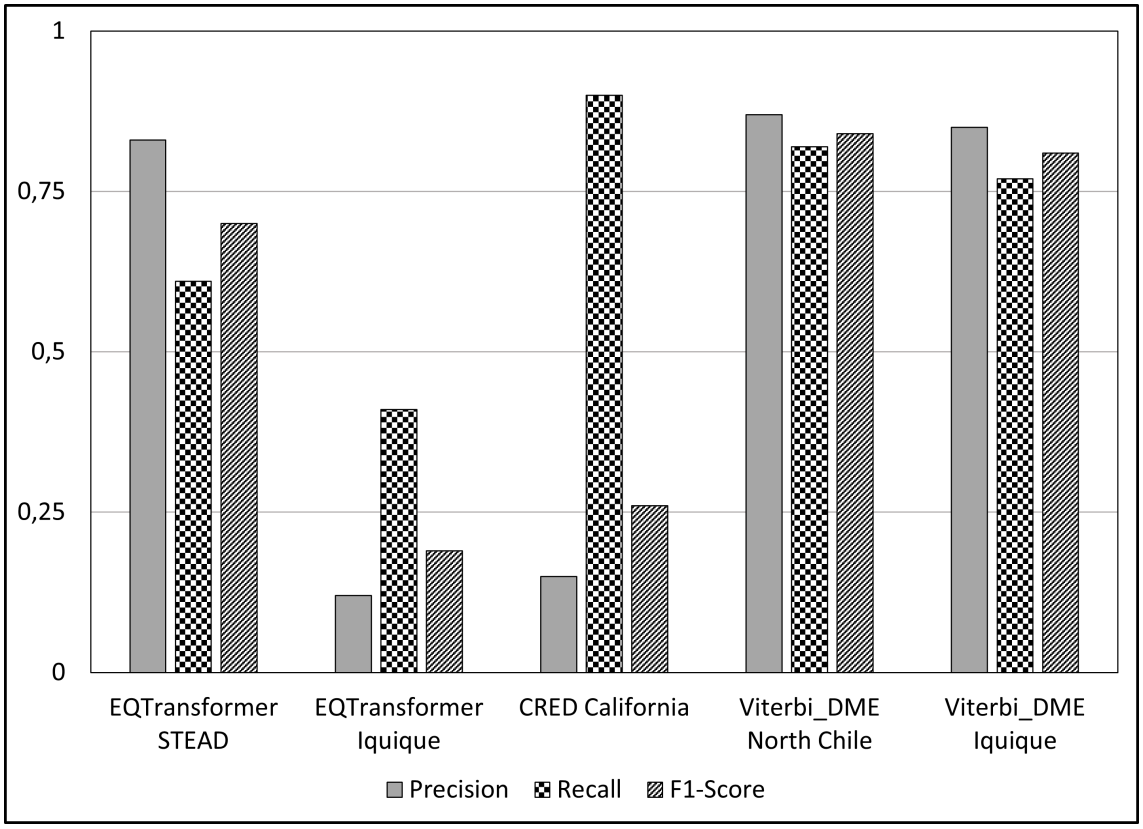


Figura 4.1: Testeo sobre la base de datos del North Chile

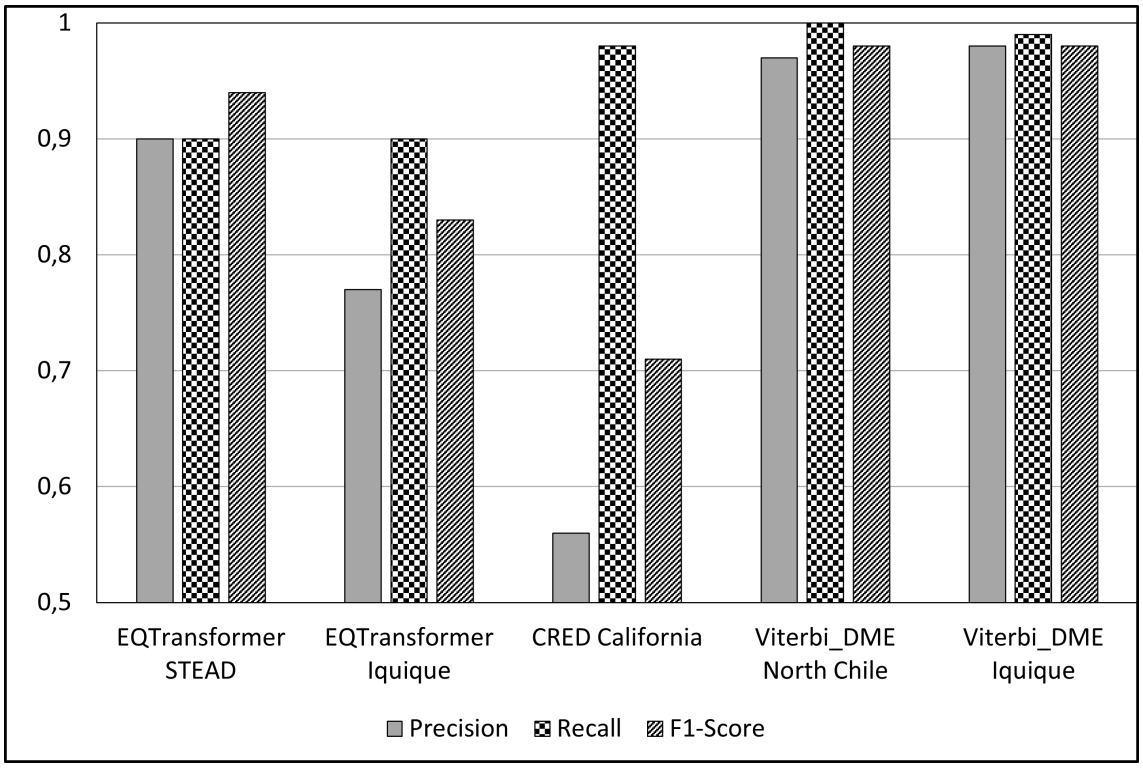


Figura 4.2: Testeo sobre la base de datos de Iquique

Los resultados de la Fig.4.2 básicamente confirman los de la Fig.4.1. En la Fig. 4.2, el esquema Viterbi_DME entrenado con datos de Iquique condujo a un puntaje f1-score que es 4 %, 18 % y 38 % más alto que los obtenidos con EQTransformer entrenado con STEAD, EQTransformer entrenado con datos de Iquique y CRED entrenado con datos de California, respectivamente. Nuevamente, el bajo puntaje de precision generado por CRED entrenado con datos de California, conduce a un bajo puntaje de f1-score. En general se observa a CRED como un modelo que es altamente sensible al ruido, ya que a pesar de tener el menor recall, la tasa de falsos positivos que detecta es bastante alta. Observe que los resultados de Viterbi_DME entrenado con sismogramas del norte de Chile o Iquique da el mismo puntaje f1-score (0.98). Esto debe ser resultado del hecho de que, como se mencionó anteriormente, Iquique se encuentra en el norte de Chile y los sismogramas en la base de datos correspondiente deben compartir características similares a las del conjunto de datos *North Chile*. Cabe destacar que el número de eventos sísmicos en la base de datos *North Chile* es menor que el número de sismogramas que pertenecen a la base de datos de Iquique (ver 3.1). Esto es sin duda una consecuencia del bajo número de parámetros que necesitan ser entrenados en la arquitectura Viterbi_DME.

Cabe destacar que los resultados de la Fig.4.2 son mejores que los de la Fig.4.1. Esto se debe a que la base de datos de Iquique (31dB) tiene una SNR promedio mayor que la del Norte de Chile (14dB). Además, cuando se reduce la SNR promedio de la Fig.4.2 a la 4.1 la degradación del rendimiento no es uniforme en todos los sistemas que se están evaluando. Cuando se comparan los resultados obtenidos con el conjunto de datos de Iquique, los experimentos con la base de datos *North Chile* arrojan una reducción promedio en el puntaje f1-score igual al 16 %, 47 % y 42 % con respecto al sistema con modelos de duración (entrenado con datos de Iquique o del Norte de Chile), EQTransformer (entrenado con el conjunto de datos de Iquique o Norte de Chile) y CRED (entrenado con el conjunto de datos de California). Esto sugiere que los modelos de duración del estado empleado aquí aumenta la solidez del sistema de detección, particularmente cuando no está entrenado con grandes bases de datos. Es importante mencionar que los sismogramas con alto ruido generalmente conducen a más errores al evaluarlos, por lo que es necesario contar con un método robusto e insensible a la variación de ruido.

Estos experimentos reflejan que Viterbi_DME es un modelo que responde bien frente a bases de datos con bajo SNR, obteniendo en todos los experimentos uno de los mejores resultados. Vale la pena resaltar que la robustez al ruido proporcionada por el modelado de estado y duración es muy interesante considerando que el sistema Viterbi_DME fue entrenado con bases de datos, es decir, Iquique y Norte de Chile, que son 5 % o menos que los conjuntos de datos STEAD o California.

4.4. Relación magnitud y distancia

En esta sección se muestran los resultados del experimento descrito en la sección 3.2.6. La figura 4.3 muestra las curvas de precisión de las distancias epicentro-estación obtenidas con la base de datos *Nearby Location*.En este experimento, se calcula el promedio de precision de los diferentes intervalos de distancias utilizando los modelos Viterbi_DME (entrenado con el conjunto de datos *North Chile*), EQTransformer (entrenado con datos STEAD) y CRED (entrenado con el conjunto de datos de California) obteniendo como resultado 0.98, 0.92 y

0.53, respectivamente. Como fue discutido anteriormente, CRED proporciona una alta tasa de FP, lo que explica que el puntaje de precisión sea más bajo que Viterbi_DME y EQTransformer. Dada la similitud entre las curvas de los modelos Viterbi_DME y EQTransformer, se utilizó la prueba de McNemar para analizar los errores cometidos por ambos modelos. La prueba de McNemar se utiliza principalmente para comparar el rendimiento de dos clasificadores y utiliza una Tabla de contingencia de 2x2. Las celdas de la Tabla incluyen el número de detecciones clasificadas correcta e incorrectamente. La hipótesis nula establece que no existen diferencias significativas entre los modelos y se rechaza cuando el valor del estadístico de prueba obtenido es $\chi^2 > 3.84$ o, alternativamente, p $< 0{,}05$, que χ^2 corresponde a la distribución Chi-cuadrado y p es la significancia estadística [120]. Entonces, de acuerdo a los resultados obtenidos al comparar las detecciones de Viterbi_DME y EQTransformer, se rechaza la hipótesis nula (p<0,001), por lo que afirmamos que la diferencia en la puntuación de precision entre Viterbi_DME y EQTransformer es estadísticamente significativa.

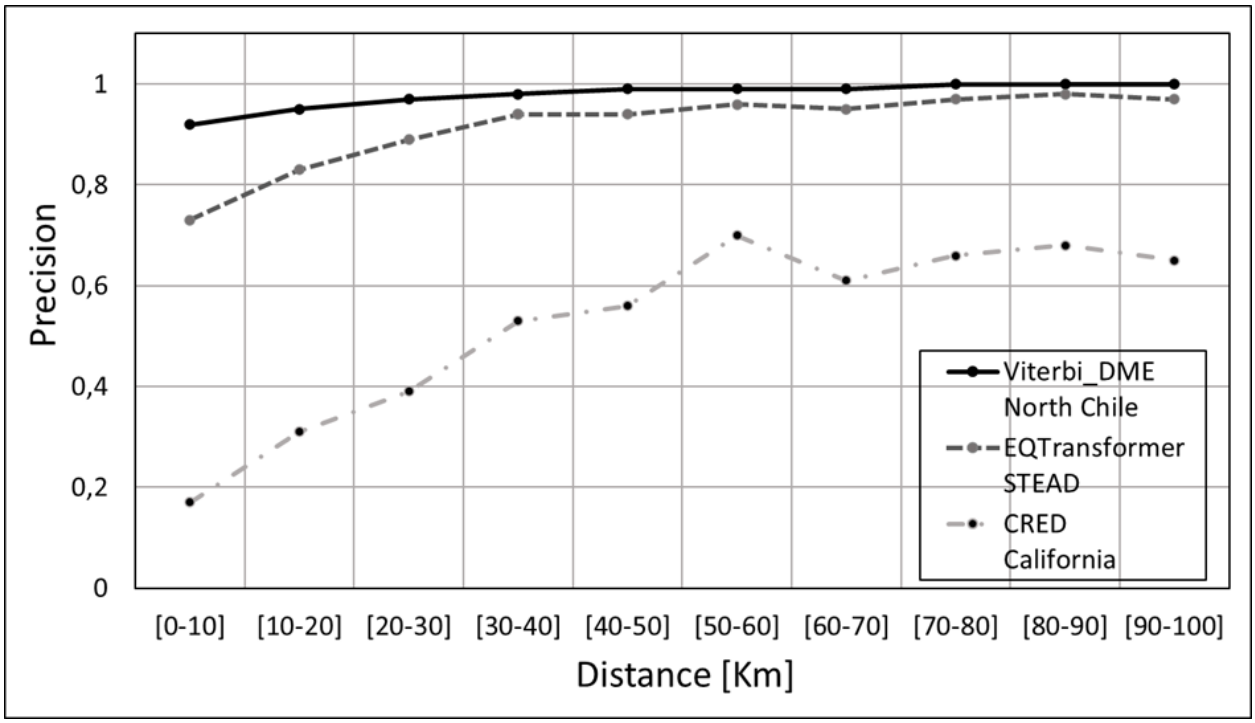


Figura 4.3: Prueba de los modelos Viterbi_DME North Chile, EQTransformer STEAD y CRED California en la base de datos local. La curva representa el valor de precisión de cada modelo sobre distancias epicentrales que llegan hasta 100 [km].

La Tabla 4.2 muestra una comparación del rendimiento de los modelos Viterbi_DME. EQTransformer y CRED utilizando las métricas de recall, precision y f1-score. Se observa que Viterbi_DME tiene los mejores resultados, ya que posee los valores más altos en todas las métricas. Esto refleja un satisfactorio rendimiento de nuestro modelo en la detección de microsismicidad local. En general, Viterbi_DME puede distinguir correctamente los terremotos de baja magnitud, aunque el ruido de fondo dificulta la detección de estos eventos.

Tabla 4.2: Comparación de los modelos Viterbi_DME entrenado con North Chile, EQTransformer entrenado con STEAD y CRED entrenado California del rendimiento promedio de los distintos intervalos de distancia que presentan los sismos de *Nearby Location*.

Model	recall	precision	f1-score
Viterbii_DME	1	0.98	0.99
EQTransformer	0.97	0.92	0.94
CRED	0.98	0.53	0.69

4.5. Demanda computacional

Los entrenamiento de los modelos EQTransformer y CRED fueron realizados usando una máquina con una GPU Nvidia A100 con 40 GB de memoria GPU. En [119] no es especificado el tiempo que tomó el entrenamiento. Por otro lado, el sistema propuesto fue entrenado utilizando un computador con 3.6GHz 8-core desktop(Intel Core i7-77000) con 64GB RAM. El tiempo que dura el proceso de entrenamiento con *North Chile* es de 50 minutos aproximandamente. Este valor considera la ejecución de Viterbi forzado, *engineered features* y entrenamiento de la red

Utilizando la útlima máquina descrita, el modelo propuesto, EQTransofmrer y CRED son testeados para calcular el tiempo de ejecución. El Viterbi_DME requirió un tiempo promedio de 693[s] para cada señal, mientras que 2.08[s] para el conjunto de datos de test de *North Chile*. Respecto a EQTransformer, requirió un tiempo promedio de 0.20[s] para cada señal, mientras que 63[s] para el conjunto de datos de test de *North Chile*. Por útlimo, CRED requirió un tiempo promedio de 0.13[s] para cada señal, mientras que 40[s] para el conjunto de datos de test de *North Chile*.

De los resultados anteriores, es posible visualizar que el Viterbi_DME es el modelo más lento para testear toda la base de datos de test de *North Chile* en un 1000 % y 1575 %, que los métodos EQTransformer y CRED, respedrivamente. Esto se atribuye a que el método propuesto incluye un postprocesamiento tras la salida de la red que no tienen los otros modelos, la cual corresponde al Algoritmo de Viterbi. Además, en este tiempo se considera la extracción de características de los sismogramas, lo cual también aporta en gran medida al tiempo final de ejecución. Finalmente se observa que el modelo con mayor rapidez en ejecución es el modelo de CRED.

4.6. Comparación de modelos en ejemplos de señales sísmicas

Para obtener más información sobre los modelos, presentamos varios ejemplos para los cuales comparamos las predicciones de los diferentes modelos. Todos los ejemplos son de los conjuntos de prueba de *North Chile*.

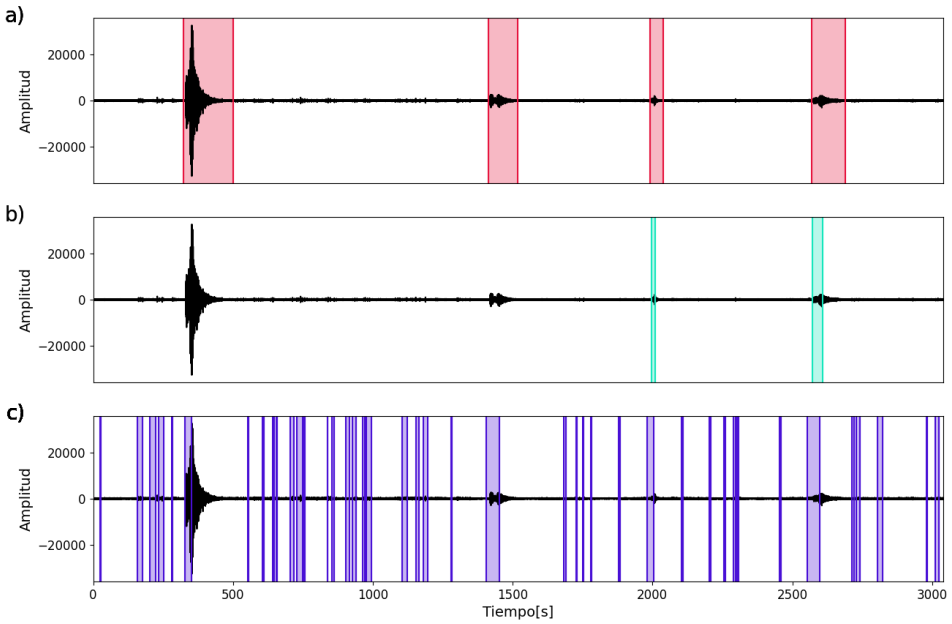


Figura 4.4: Mediciones realizadas por la estación sismológica CO06 a) Detecciones generadas por Viterbi_DME, b) Detecciones generadas por EQTransformer, c) Detecciones generadas por CRED

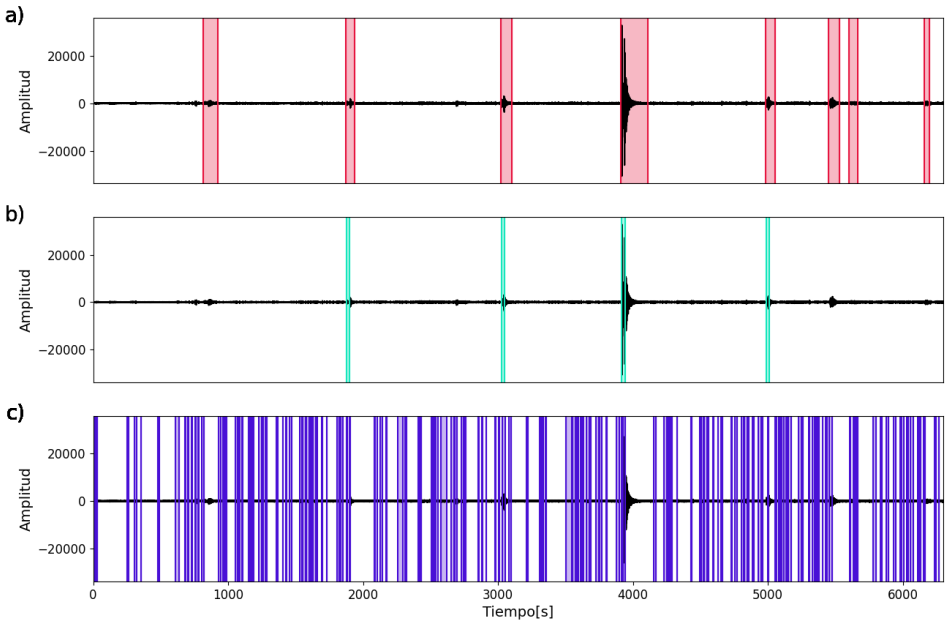


Figura 4.5: Mediciones realizadas por la estación sismológica VA03 a) Detecciones generadas por Viterbi_DME, b) Detecciones generadas por EQTransformer, c) Detecciones generadas por CRED

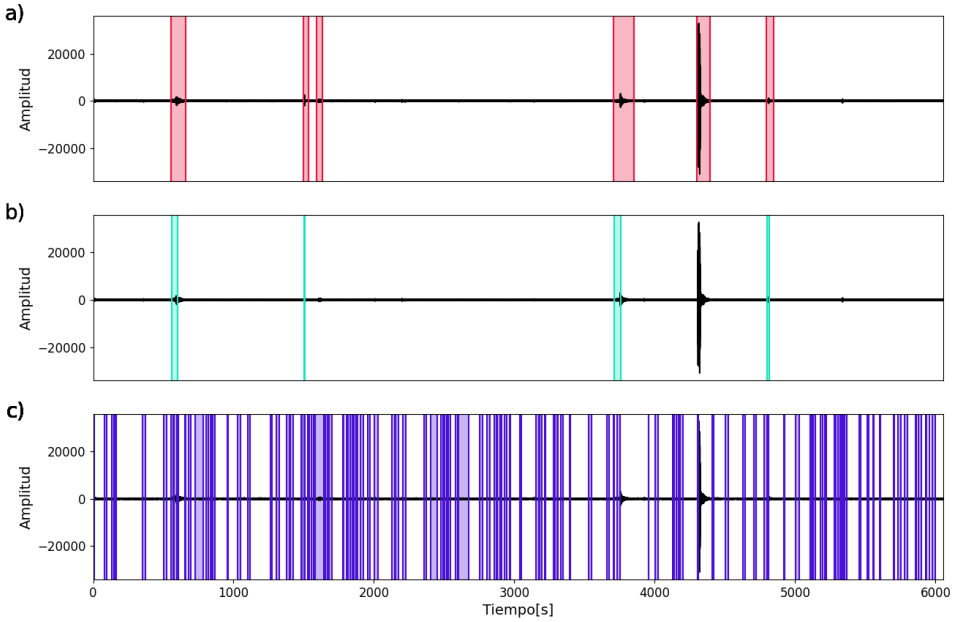


Figura 4.6: Mediciones realizadas por la estación sismológica GO03 a) Detecciones generadas por Viterbi_DME, b) Detecciones generadas por EQTransformer, c) Detecciones generadas por CRED

Las figuras 4.4, 4.5 y 4.6 corresponden a sismogramas medidos entre los tiempos 16:57:50 25/09/2020 y 18:42:40 25/09/2020 desde las estaciones sismológicas CO06, VA03 y GO03, respectivamente. Se observa que la cantidad de sismos visualizados en cada figura varía a pesar que las mediciones corresponden al mismo intervalo de tiempo. Esto se debe a que la localización de las estaciones sismológicas es distinta, por lo tanto una estación puede registrar un sismo que no alcanza a ser medido en otra.

En la figura 4.4, se observa a simple vista que Viterbi_DME es capaz de detectar correctamente todos los sismos. Por el contrario, EQTransformer posee dos falsos negativos y CRED a pesar de que detecta a todos los sismos, genera una gran cantidad de falsos positivos de corta duración. En este ejemplo se observa que las detecciones del modelo propuesto no solo generan una buena detección de la onda P, sino que también define correctamente los intervalos de duración de cada detección.

En la figura 4.5, se observan resultados bastantes similares a la figura anterior. Se visualiza que Viterbi_DME nuevamente es capaz de detectar con buena calidad todos los sismos de la señal. Por el contrario, EQTransformer otra vez no detecta a todos los sismos y el modelo de CRED vuelve a genera una gran cantidad de falsos positivos de corta duración. Es importante notar que nuestro sistema es capaz de detectar tanto a los sismos pequeños como a los grandes.

En la figura 4.6, se observa la presencia de unos diminutos sismos que no fueron detectados por el modelo propuesto ni por EQTransformer. No así el modelo de CRED, el cual es capaz de detectarlos pero generando a la vez una gran cantidad de falsos positivos.

Los resultados observados en estas figuras se contrastan con los resultados que se pueden observar en la figura 4.1, ya que los mejores resultados se tienen con Viterbi_DME, seguido por EQTransformer y finalmente CRED.

Capítulo 5

Conclusiones y Trabajo Futuro

En este capítulo se presentan las principales conclusiones respecto al trabajo realizado y se comenta sobre el posible trabajo futuro a desarrollar en esta línea de investigación.

Dado el aumento del volumen de datos sísmicos, es necesario desarrollar herramientas eficientes, confiables y robustas que puedan detectar terremotos de forma automática. Dos condiciones hacen que la tarea sea particularmente difícil. En primer lugar, el rendimiento de tales sistemas se degrada cuando disminuye la SNR de las señales del sismograma. En segundo lugar, los sistemas de detección de terremotos pueden funcionar peor o al menos por debajo de la precisión óptima si se prueban en una región diferente de aquella en la que corresponde la base de datos de entrenamiento. En otras palabras, los sistemas entrenados con bases de datos locales deberían funcionar mejor. Particularmente en Chile, la alta tasa de sismicidad, la falta de adquisición de registros sísmicos de baja magnitud en la distancia local y la baja SNR de los sismogramas que contienen microsismos impiden que se pueda tener un registro sísmico completo, generando que muchos sismos no sean catalogados y que los datos de entrenamiento estén sesgados solo hacia eventos de grandes magnitudes. Para contrarrestar estas limitaciones, este trabajo propone un sistema *end-to-end* basado en DNN-HMM con *engineered features* y modelos de duración de estados y eventos sísmicos. Esta propuesta requiere unas 20 veces menos parámetros que los métodos más avanzados. Cuanto menor sea el número de parámetros entrenables, menor será la base de datos de entrenamiento necesaria. Además, los modelos de estado y duración de eventos puede aumentar significativamente la robustez del ruido del sistema de detección, particularmente con datos de entrenamiento limitados.

Utilizando la base de datos *North Chile*, fueron evaluados en el sistema propuesto siete variaciones de los modelos de duración, los cuales difieren en el uso de las restricciones de duración de estados y eventos sobre sismo y/o sobre ruido. Los resultados discutidos en este trabajo muestran que el mejor rendimiento se obtiene con la aplicación de los modelos de duración de estados y eventos solo sobre sismos, consiguiendo reducir la tasa de falsos positivos en un $33,3\,\%$ con respecto a una decodificación sin restricciones. También el rendimiento de detección del sistema propuesto fue comparado cuantitativamente con dos métodos avanzados: CRED y EQTransformer. Los resultados discutidos aquí muestran que el sistema propuesto proporciona un puntaje f1-score promedio $105\,\%$ más alto que EQTransformer y CRED probados con las bases de datos de Iquique y *North Chile*. Además, el sistema presentado proporciona una reducción en la puntuación f1-score igual al $16\,\%$ cuando la SNR

media se reduce en aproximadamente 17 dB. Esta reducción en la puntuación f1-score es al menos la mitad de la observada con EQTransformer o CRED en las mismas condiciones de prueba. En cuanto a la detección de pequeños sismos a corta distancia epicentro-estación, la precisión promediada proporcionada por el sistema DNN-HMM con modelado de duración es al menos un 7 % superior a EQTransformer, el sistema con el puntaje de precisión más cercano.

A pesar de que el sistema propuesto fue entrenado con pocos datos, los resultados muestran que se puede adaptar de buena forma a los datos regionales, corroborando que el uso de pocos parámetros ocasiona una adecuada capacidad de generalización con un entrenamiento que presenta datos limitados. También, utilizando *engineered features* se pudo preservar las características esenciales de la señal y evitar el sobreajuste del modelo. Por otro lado, la integración de modelos de duración de estados y eventos en el procedimiento de decodificación de Viterbi logró aumentar la precisión de forma complementaria en el problema de detección y disminuir la tasa de falsos positivos. Finalmente, se concluye que se cumplieron los objetivos propuestos ya que se logró desarrollar e implementar un método robusto y confiable para la detección automática de sismos utilizando técnicas de machine learning.

5.1. Trabajo Futuro

El desarrollo de esta Tesis y sus conclusiones dan origen a una serie de futuras investigaciones que pueden mejorar los resultados obtenidos y que pueden aportar en gran manera a la comunidad científica. En primer lugar, se propone investigar modificaciones al diseño del sistema que puedan mejorar su desempeño, como por ejemplo:

- Reemplazar la DNN por otras arquitecturas de redes neuronales. En este trabajos se utilizó una DNN para entrenar las probabilidades de observación, ya que muchos estudios en el área de reconocimiento de voz han demostrado que el sistema DNN-HMM es muy potente [121, 122], pero también es posible encontrar en la literatura otras arquitecturas que se pueden combinar con HMM, algunas de ellas son RNN [123], LSTM [124], CNN [125]. Por ejemplo [124] se reemplaza la DNN por una LSTM, ya que la red LSTM es capaz de comprimir toda la información histórica a diferencia de la DNN, la cual solo captura el contexto de un número fijo de elementos de información. Dado la gran variedad de pruebas que se puede realizar en esta dirección, sería interesante ahondar más en este trabajo.

- Definir las cotas de las restricciones de duración de estados y eventos en función de la magnitud del sismo y de la distancia al hipocentro. Las características de la señal de un sismo tienen intrínsecamente una amplia cantidad de variables. Por ejemplo la amplitud y duración de un terremoto especifico se observan distintas en las señales que fueron medidas cerca del hipocentro y en las que fueron capturadas a una gran distancia. Dado la variabilidad de estas señales emergentes, sería interesante modelar la duración de los sismos en función de la magnitud y la distancia al hipocentro para restringirlos de mejor forma en la decodificación.

- Implementar aprendizaje por transferencia. Dado que se en esta investigación se ha utilizado bases de datos limitadas, sería interesante estudiar si la utilización del aprendizaje

por transferencia con datos locales beneficia el rendimiento de la detección de eventos sísmicos. Por ejemplo, se podría utilizar como condición inicial en el entrenamiento con NorthChile, los pesos que se obtienen del entrenamiento con Iquique, o viceversa.

- Incorporar más métodos de robustez al ruido. Dependiendo del lugar en el que se ubiquen los sismógrafos, los sismogramas pueden verse perturbados por diferentes fuentes de ruido, como por ejemplo las condiciones climáticas, construcciones, actividad urbana, entre otros. Dado lo anterior, sería interesante aplicar métodos que puedan ser insensibles a los distintos factores de ruido que se ven sometidos los simografos.

En segundo lugar, se propone como trabajo futuro ahondar más en el nivel de detección que tiene el sistema, como por ejemplo:

- Evaluar el desempeño local del modelo sobre diferentes regiones de Chile. Dado el éxito obtenido en la implementación del sistema en el Norte de Chile, sería de gran aporte estudiar el nivel de detección en regiones que están ubicadas al Sur de Chile, donde la calidad de los registros sismológicos es más bajas debido a las condiciones climáticas.

- Evaluar al sistema con respecto a la selección de las fases P y S. Nuestro sistema es capaz de proporcionar la información de las fases, ya que la salida corresponde a una alineación óptima de estados, donde los estados 1-3 representan a la onda P, y los estados 2-6 a la onda S. Como el objetivo de esta investigación es solo realizar la detección del evento sísmico, no se realiza este estudio, pero con la información anterior es posible y de gran interés llevarlo a cabo como trabajo futuro.

Finalmente se propone encapsular el sistema para que su implementación pueda ser llevado a cabo por el Centro Sismológico Nacional Chileno. Considerando los buenos resultados que se tuvo sobre la microsismicidad local, sería interesante ponerlo en marcha de forma offline para ir monitoreando en zonas de interés a los sismos de baja magnitud, como por ejemplo, la falla de San Ramón.

Bibliografía

[1] H. Kanamori, "Importance of historical seismograms for geophysical research," 1988. Academic Press.

[2] S. M. Mousavi, W. L. Ellsworth, W. Zhu, L. Y. Chuang, and G. C. Beroza, "Earthquake transformer—an attentive deep-learning model for simultaneous earthquake detection and phase picking," *Nature communications*, vol. 11, no. 1, pp. 1–12, 2020.

[3] A. Cofré, M. Marín, O. V. Pino, N. Galleguillos, S. Riquelme, S. Barrientos, and N. B. Yoma, "End-to-end lstm based earthquake magnitude estimation with a single station," *IEEE Geoscience and Remote Sensing Letters*, 2022.

[4] S. M. Mousavi, C. A. Langston, and S. P. Horton, "Automatic microseismic denoising and onset detection using the synchrosqueezed continuous wavelet transform," *Geophysics*, vol. 81, no. 4, pp. V341–V355, 2016.

[5] X. Zhang, J. Zhang, C. Yuan, S. Liu, Z. Chen, and W. Li, "Locating induced earthquakes with a network of seismic stations in oklahoma via a deep learning method," *Scientific reports*, vol. 10, no. 1, pp. 1–12, 2020.

[6] N. B. Yoma, J. Wuth, A. Pinto, N. de Celis, J. Celis, F. Huenupan, and I. J. Fustos-Toribio, "End-to-end lstm based estimation of volcano event epicenter localization," *Journal of Volcanology and Geothermal Research*, vol. 429, p. 107615, 2022.

[7] K. L. Pankow, M. Stickney, J. Y. Ben-Horin, M. Litherland, S. Payne, K. D. Koper, S. L. Bilek, and K. Bogolub, "Regional seismic network monitoring in the eastern intermountain west," *Seismological Research Letters*, vol. 91, no. 2A, pp. 631–646, 2020.

[8] C. Lomnitz, "Major earthquakes of chile: a historical survey, 1535-1960," *Seismological Research Letters*, vol. 75, no. 3, pp. 368–378, 2004.

[9] S. Cesca, F. Grigoli, S. Heimann, T. Dahm, M. Kriegerowski, M. Sobiesiak, C. Tassara, and M. Olcay, "The m w 8.1 2014 iquique, chile, seismic sequence: a tale of foreshocks and aftershocks," *Geophysical Journal International*, vol. 204, no. 3, pp. 1766–1780, 2016.

[10] A. G. Iaccarino, P. Gueguen, M. Picozzi, and S. Ghimire, "Earthquake early warning system for structural drift prediction using machine learning and linear regressors," *Frontiers in Earth Science*, vol. 9, p. 666444, 2021.

[11] K. Bergen and G. C. Beroza, "Automatic earthquake detection by active learning," in *AGU Fall Meeting Abstracts*, vol. 2017, pp. S43A–0828, 2017.

[12] B. Gutenberg, *Seismicity of the earth and associated phenomena*. Read Books Ltd, 2013.

[13] N. J. van der Elst, "B-positive: A robust estimator of aftershock magnitude distribution in transiently incomplete catalogs," *Journal of Geophysical Research: Solid Earth*, vol. 126, no. 2, p. e2020JB021027, 2021.

[14] J. A. Kintner, K. M. Cleveland, C. J. Ammon, and A. Nyblade, "Local-distance seismic event relocation and relative magnitude estimation, applications to mining related seismicity in the powder river basin, wyoming," *Bulletin of the Seismological Society of America*, vol. 111, no. 3, pp. 1347–1364, 2021.

[15] Q. Kong, D. T. Trugman, Z. E. Ross, M. J. Bianco, B. J. Meade, and P. Gerstoft, "Machine learning in seismology: Turning data into insights," *Seismological Research Letters*, vol. 90, no. 1, pp. 3–14, 2019.

[16] L. Zhou, S. Pan, J. Wang, and A. V. Vasilakos, "Machine learning on big data: Opportunities and challenges," *Neurocomputing*, vol. 237, pp. 350–361, 2017.

[17] W.-C. Kao, M.-C. Hsu, and Y.-Y. Yang, "Local contrast enhancement and adaptive feature extraction for illumination-invariant face recognition," *Pattern Recognition*, vol. 43, no. 5, pp. 1736–1747, 2010.

[18] T. H. Jordan, "Structure and formation of the continental tectosphere," *Journal of Petrology*, no. 1, pp. 11–37, 1988.

[19] E. E. Brodsky, "The importance of studying small earthquakes," *Science*, vol. 364, no. 6442, pp. 736–737, 2019.

[20] N. Xu, C. Tang, L. Li, Z. Zhou, C. Sha, Z. Liang, and J. Yang, "Microseismic monitoring and stability analysis of the left bank slope in jinping first stage hydropower station in southwestern china," *International Journal of Rock Mechanics and Mining Sciences*, vol. 48, no. 6, pp. 950–963, 2011.

[21] A. Wang, D. Song, X. He, L. Dou, Z. Li, Z. Zu, Q. Lou, and Y. Zhao, "Investigation of coal and gas outburst risk by microseismic monitoring," *Plos one*, vol. 14, no. 5, p. e0216464, 2019.

[22] T.-H. Ma, C.-A. Tang, S.-B. Tang, L. Kuang, Q. Yu, D.-Q. Kong, and X. Zhu, "Rockburst mechanism and prediction based on microseismic monitoring," *International Journal of Rock Mechanics and Mining Sciences*, vol. 110, pp. 177–188, 2018.

[23] S. Cheng, S.-c. Li, L.-p. Li, S.-s. Shi, Z.-q. Zhou, and J. Wang, "Study on energy band characteristic of microseismic signals in water inrush channel," *Journal of Geophysics and Engineering*, vol. 15, no. 5, pp. 1826–1834, 2018.

[24] V. Zaalishvili, D. Melkov, A. Kanukov, B. Dzeranov, and V. Shepelev, "Application of microseismic and calculational techniques in engineering-geological zonation," *GEOMATE Journal*, vol. 10, no. 19, pp. 1670–1674, 2016.

[25] R. Madariaga, "Sismicidad de chile.," *Física de la Tierra*, no. 10, p. 221, 1998.

[26] S. Blake, K. Burton, I. Parkinson, N. Harris, and M. Widdowson, *An introduction to our dynamic planet*. Cambridge University Press, 2008.

[27] A. Coburn and R. Spence, *Earthquake protection*. John Wiley & Sons, 2003.

[28] A. M. Suteau and J. H. Whitcomb, "A local earthquake coda magnitude and its relation to duration, moment mo, and local richter magnitude ml," *Bulletin of the Seismological*

Society of America, vol. 69, no. 2, pp. 353–368, 1979.

[29] L. Gutiérrez, J. Ibanez, G. Cortés, J. Ramírez, C. Benítez, V. Tenorio, and A. Isaac, "Volcano-seismic signal detection and classification processing using hidden markov models. application to san cristóbal volcano, nicaragua," in *2009 IEEE International Geoscience and Remote Sensing Symposium*, vol. 4, pp. IV–522, IEEE, 2009.

[30] R. Carniel, "Characterization of volcanic regimes and identification of significant transitions using geophysical data: a review," *Bulletin of Volcanology*, vol. 76, no. 8, pp. 1–22, 2014.

[31] R. N. Bracewell and R. N. Bracewell, *The Fourier transform and its applications*, vol. 31999. McGraw-Hill New York, 1986.

[32] S. Kumar, K. Singh, and R. Saxena, "Analysis of dirichlet and generalized "hamming" window functions in the fractional fourier transform domains," *signal processing*, vol. 91, no. 3, pp. 600–606, 2011.

[33] Z. S. Bojkovic, B. M. Bakmaz, and M. R. Bakmaz, "Hamming window to the digital world," *Proceedings of the IEEE*, vol. 105, no. 6, pp. 1185–1190, 2017.

[34] A. A. Hannah and G. K. Agordzo, "A design of a low-pass fir filter using hamming window functions in matlab," 2020.

[35] J. Price and T. Goble, "Signals and noise," in *Telecommunications Engineer's Reference Book*, pp. 10–1, Elsevier, 1993.

[36] R. V. Allen, "Automatic earthquake recognition and timing from single traces," *Bulletin of the seismological society of America*, vol. 68, no. 5, pp. 1521–1532, 1978.

[37] A. Trnkoczy, "Understanding and parameter setting of sta/lta trigger algorithm," in *New Manual of Seismological Observatory Practice (NMSOP)*, pp. 1–20, Deutsches GeoForschungsZentrum GFZ, 2009.

[38] S. M. Mousavi, W. Zhu, Y. Sheng, and G. C. Beroza, "Cred: A deep residual network of convolutional and recurrent units for earthquake signal detection," *Scientific reports*, vol. 9, no. 1, pp. 1–14, 2019.

[39] C. E. Yoon, O. O'Reilly, K. J. Bergen, and G. C. Beroza, "Earthquake detection through computationally efficient similarity search," *Science advances*, vol. 1, no. 11, p. e1501057, 2015.

[40] S. Baluja and M. Covell, "Waveprint: Efficient wavelet-based audio fingerprinting," *Pattern recognition*, vol. 41, no. 11, pp. 3467–3480, 2008.

[41] T. Perol, M. Gharbi, and M. Denolle, "Convolutional neural network for earthquake detection and location," *Science Advances*, vol. 4, no. 2, p. e1700578, 2018.

[42] K. J. Bergen and G. C. Beroza, "Detecting earthquakes over a seismic network using single-station similarity measures," *Geophysical Journal International*, vol. 213, no. 3, pp. 1984–1998, 2018.

[43] I. El Naqa and M. J. Murphy, "What is machine learning?," in *machine learning in radiation oncology*, pp. 3–11, Springer, 2015.

[44] Y. Long, H. Li, Z. Du, M. Geng, and Z. Liu, "Confined gaussian-distributed electro-

magnetic field of tin (ii) chloride-sensitized surface-enhanced raman scattering (sers) optical fiber probe: From localized surface plasmon resonance (lspr) to waveguide propagation," *Journal of Colloid and Interface Science*, vol. 581, pp. 698–708, 2021.

[45] R. Dürichen, M. A. Pimentel, L. Clifton, A. Schweikard, and D. A. Clifton, "Multitask gaussian processes for multivariate physiological time-series analysis," *IEEE Transactions on Biomedical Engineering*, vol. 62, no. 1, pp. 314–322, 2014.

[46] R. Marcelpoil and Y. Usson, "Methods for the study of cellular sociology: Voronoi diagrams and parametrization of the spatial relationships," *Journal of theoretical biology*, vol. 154, no. 3, pp. 359–369, 1992.

[47] D. A. Reynolds, "Gaussian mixture models.," *Encyclopedia of biometrics*, vol. 741, no. 659-663, 2009.

[48] H. H. Aghdam and E. J. Heravi, "Guide to convolutional neural networks," *New York, NY: Springer*, vol. 10, no. 978-973, p. 51, 2017.

[49] K. Yasaka, H. Akai, A. Kunimatsu, S. Kiryu, and O. Abe, "Deep learning with convolutional neural network in radiology," *Japanese journal of radiology*, vol. 36, no. 4, pp. 257–272, 2018.

[50] Y. Ge, S. Jiang, Q. Xu, C. Jiang, and F. Ye, "Exploiting representations from pretrained convolutional neural networks for high-resolution remote sensing image retrieval," *Multimedia Tools and Applications*, vol. 77, no. 13, pp. 17489–17515, 2018.

[51] A. Kamilaris and F. X. Prenafeta-Boldú, "A review of the use of convolutional neural networks in agriculture," *The Journal of Agricultural Science*, vol. 156, no. 3, pp. 312–322, 2018.

[52] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1d convolutional neural networks and applications: A survey," *Mechanical systems and signal processing*, vol. 151, p. 107398, 2021.

[53] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[54] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with lstm recurrent networks," *Journal of machine learning research*, vol. 3, no. Aug, pp. 115–143, 2002.

[55] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, $\mathcal{L}$. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[56] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*, pp. 213–229, Springer, 2020.

[57] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[58] C. Lüscher, E. Beck, K. Irie, M. Kitza, W. Michel, A. Zeyer, R. Schlüter, and H. Ney, "Rwth asr systems for librispeech: Hybrid vs attention–w/o data augmentation," *arXiv*

preprint arXiv:1905.03072, 2019.

[59] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, and D. Tran, "Image transformer," in *International conference on machine learning*, pp. 4055–4064, PMLR, 2018.

[60] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[61] J. A. Bilmes *et al.*, "A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models," *International computer science institute*, vol. 4, no. 510, p. 126, 1998.

[62] G. Synnaeve, Q. Xu, J. Kahn, T. Likhomanenko, E. Grave, V. Pratap, A. Sriram, V. Liptchinsky, and R. Collobert, "End-to-end asr: from supervised to semi-supervised learning with modern architectures," *arXiv preprint arXiv:1911.08460*, 2019.

[63] J. Benesty, J. Chen, and Y. Huang, "Automatic speech recognition: A deep learning approach," 2008. Springer Berlin Heidelberg, New York, USA.

[64] Y. Bengio, Y. LeCun, *et al.*, "Scaling learning algorithms towards ai," *Large-scale kernel machines*, vol. 34, no. 5, pp. 1–41, 2007.

[65] D. E. Rumelhart, "Learning representations by back-propagation errors," *nature*, vol. 323, pp. 533–536, 1986.

[66] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[67] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *The annals of mathematical statistics*, vol. 37, no. 6, pp. 1554–1563, 1966.

[68] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains," *The annals of mathematical statistics*, vol. 41, no. 1, pp. 164–171, 1970.

[69] L. E. Baum *et al.*, "An inequality and associated maximization technique in statistical estimation for probabilistic functions of markov processes," *Inequalities*, vol. 3, no. 1, pp. 1–8, 1972.

[70] L. Rabiner and B. Juang, "An introduction to hidden markov models," *IEEE ASSP MAGAZINE*, vol. 3, no. 1, pp. 4–16, 1986.

[71] L. Rabiner and B.-H. Juang, *Fundamentals of speech recognition*. Prentice-Hall, Inc., 1993.

[72] A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler, "Hidden markov models in computational biology: Applications to protein modeling," *Journal of molecular biology*, vol. 235, no. 5, pp. 1501–1531, 1994.

[73] J. Yang and Y. Xu, "Hidden markov model for gesture recognition," tech. rep., Carnegie mellon University Pittsburgh Robotics Institute, 1994.

[74] H. Bunke, M. Roth, and E. G. Schukat-Talamazzini, "Off-line cursive handwriting recognition using hidden markov models," *Pattern recognition*, vol. 28, no. 9, pp. 1399–

1413, 1995.

[75] I. Visser, M. E. Raijmakers, and P. Molenaar, "Fitting hidden markov models to psychological data," *Scientific Programming*, vol. 10, no. 3, pp. 185–199, 2002.

[76] T. Oates, L. Firoiu, and P. R. Cohen, "Clustering time series with hidden markov models and dynamic time warping," in *Proceedings of the IJCAI-99 workshop on neural, symbolic and reinforcement learning methods for sequence learning*, vol. 17, p. 21, Citeseer, 1999.

[77] P. Baldi, Y. Chauvin, T. Hunkapiller, and M. A. McClure, "Hidden markov models of biological primary sequence information.," *Proceedings of the National Academy of Sciences*, vol. 91, no. 3, pp. 1059–1063, 1994.

[78] N. Li and M. Stephens, "Modeling linkage disequilibrium and identifying recombination hotspots using single-nucleotide polymorphism data," *Genetics*, vol. 165, no. 4, pp. 2213–2233, 2003.

[79] B.-J. Yoon, "Hidden markov models and their applications in biological sequence analysis," *Current genomics*, vol. 10, no. 6, pp. 402–415, 2009.

[80] M. Kritzman, S. Page, and D. Turkington, "Regime shifts: Implications for dynamic strategies (corrected)," *Financial Analysts Journal*, vol. 68, no. 3, pp. 22–39, 2012.

[81] N. Nguyen and D. Nguyen, "Hidden markov model for stock selection," *Risks*, vol. 3, no. 4, pp. 455–473, 2015.

[82] D. N. Nguyet Nguyen and T. P. Wakefield, "Using the hidden markov model to improve the hull-white model for short rate," *International Journal of Trade, Economics and Finance*, vol. 9, no. 2, 2018.

[83] A. A. Markov, "Extension of the law of large numbers to dependent quantities," *Izv. Fiz.-Matem. Obsch. Kazan Univ.(2nd Ser)*, vol. 15, no. 1, pp. 135–156, 1906.

[84] F. Jelinek, "Interpolated estimation of markov source parameters from sparse data," in *Proc. Workshop on Pattern Recognition in Practice, 1980*, 1980.

[85] N. Morgan and H. A. Bourlard, "Neural networks for statistical recognition of continuous speech," *Proceedings of the IEEE*, vol. 83, no. 5, pp. 742–772, 1995.

[86] M. Heck, C. Hammer, A. van Herwijnen, J. Schweizer, and D. Fäh, "Automatic detection of snow avalanches in continuous seismic data using hidden markov models," *Natural Hazards and Earth System Sciences*, vol. 18, no. 1, pp. 383–396, 2018.

[87] P. Peng, Z. He, and L. Wang, "Automatic classification of microseismic signals based on mfcc and gmm-hmm in underground mines," *Shock and Vibration*, vol. 2019, 2019.

[88] F. Dammeier, J. R. Moore, C. Hammer, F. Haslinger, and S. Loew, "Automatic detection of alpine rockslides in continuous seismic data using hidden markov models," *Journal of Geophysical Research: Earth Surface*, vol. 121, no. 2, pp. 351–371, 2016.

[89] G. Cortés, L. García, I. Álvarez, C. Benítez, Á. de la Torre, and J. Ibáñez, "Parallel system architecture (psa): An efficient approach for automatic recognition of volcano-seismic events," *Journal of Volcanology and Geothermal Research*, vol. 271, pp. 1–10, 2014.

[90] M. Ohrnberger, *Continuous automatic classification of seismic signals of volcanic origin at Mt. Merapi, Java, Indonesia*. PhD thesis, Potsdam, Univ., Diss., 2001, 2001.

[91] M. Bicego, C. Acosta-Munoz, and M. Orozco-Alzate, "Classification of seismic volcanic signals using hidden-markov-model-based generative embeddings," *IEEE transactions on geoscience and remote sensing*, vol. 51, no. 6, pp. 3400–3409, 2013.

[92] S. M. Bhatti, M. S. Khan, J. Wuth, F. Huenupan, M. Curilem, L. Franco, and N. B. Yoma, "Automatic detection of volcano-seismic events by modeling state and event duration in hidden markov models," *Journal of Volcanology and Geothermal Research*, vol. 324, pp. 134–143, 2016.

[93] G. Cortés, R. Carniel, P. Lesage, M. Á. Mendoza, and I. Della Lucia, "Practical volcano-independent recognition of seismic events: Vulcan. ears project," *Frontiers in Earth Science*, vol. 8, p. 616676, 2021.

[94] C. Benitez, A. Alos, J. Prudencio, I. Alvarez, A. de la Torre, *et al.*, "A comparative study of classifiers based on hmm, gmm and svm for the vt, lp and noises discrimination task.," in *EGU General Assembly Conference Abstracts*, p. 11783, 2014.

[95] M. Beyreuther, R. Carniel, and J. Wassermann, "Continuous hidden markov models: Application to automatic earthquake detection and classification at las canãdas caldera, tenerife," *Journal of Volcanology and Geothermal Research*, vol. 176, no. 4, pp. 513–518, 2008.

[96] M. Beyreuther, C. Hammer, J. Wassermann, M. Ohrnberger, and T. Megies, "Constructing a hidden markov model based earthquake detector: application to induced seismicity," *Geophysical Journal International*, vol. 189, no. 1, pp. 602–610, 2012.

[97] M. Beyreuther and J. Wassermann, "Continuous earthquake detection and classification using discrete hidden markov models," *Geophysical Journal International*, vol. 175, no. 3, pp. 1055–1066, 2008.

[98] M. Beyreuther and J. Wassermann, "Hidden semi-markov model based earthquake classification system using weighted finite-state transducers," *Nonlinear Processes in Geophysics*, vol. 18, no. 1, pp. 81–89, 2011.

[99] P. Alasonati, J. Wassermann, M. Ohrnberger, H. Mader, C. Connor, and S. Coles, "Signal classification by wavelet-based hidden markov models: application to seismic signals of volcanic origin," *Statistics in Volcanology*, vol. 1, pp. 161–174, 2006.

[100] M. C. Benítez, J. Ramírez, J. C. Segura, J. M. Ibanez, J. Almendros, A. García-Yeguas, and G. Cortes, "Continuous hmm-based seismic-event classification at deception island, antarctica," *IEEE Transactions on Geoscience and remote sensing*, vol. 45, no. 1, pp. 138–146, 2007.

[101] M. Joswig, "Pattern recognition for earthquake detection," *Bulletin of the Seismological Society of America*, vol. 80, no. 1, pp. 170–186, 1990.

[102] J. Woollam, A. Rietbrock, A. Bueno, and S. De Angelis, "Convolutional neural network for seismic phase classification, performance demonstration over a local seismic network," *Seismological Research Letters*, vol. 90, no. 2A, pp. 491–502, 2019.

[103] H. Soto and B. Schurr, "Deepphasepick: A method for detecting and picking seismic phases from local earthquakes based on highly optimized convolutional and recurrent

deep neural networks," *Geophysical Journal International*, vol. 227, no. 2, pp. 1268–1294, 2021.

[104] Z. E. Ross, M.-A. Meier, E. Hauksson, and T. H. Heaton, "Generalized seismic phase detection with deep learningshort note," *Bulletin of the Seismological Society of America*, vol. 108, no. 5A, pp. 2894–2901, 2018.

[105] W. Zhu and G. C. Beroza, "Phasenet: a deep-neural-network-based seismic arrival-time picking method," *Geophysical Journal International*, vol. 216, no. 1, pp. 261–273, 2019.

[106] J. Wang, Z. Xiao, C. Liu, D. Zhao, and Z. Yao, "Deep learning for picking seismic arrival times," *Journal of Geophysical Research: Solid Earth*, vol. 124, no. 7, pp. 6612–6624, 2019.

[107] Y. Zhou, H. Yue, Q. Kong, and S. Zhou, "Hybrid event detection and phase-picking algorithm using convolutional and recurrent neural networks," *Seismological Research Letters*, vol. 90, no. 3, pp. 1079–1087, 2019.

[108] A. Lomax, A. Michelini, and D. Jozinović, "An investigation of rapid earthquake characterization using single-station waveforms and a convolutional neural network," *Seismological Research Letters*, vol. 90, no. 2A, pp. 517–529, 2019.

[109] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[110] A. Rosenfeld, R. Zemel, and J. K. Tsotsos, "The elephant in the room," *arXiv preprint arXiv:1808.03305*, 2018.

[111] H. Shen and Y. Shen, "Array-based convolutional neural networks for automatic detection and 4d localization of earthquakes in hawai 'i," *Seismological Research Letters*, vol. 92, no. 5, pp. 2961–2971, 2021.

[112] O. M. Saad, G. Huang, Y. Chen, A. Savvaidis, S. Fomel, N. Pham, and Y. Chen, "Scalodeep: A highly generalized deep learning framework for real-time earthquake detection," *Journal of Geophysical Research: Solid Earth*, vol. 126, no. 4, p. e2020JB021473, 2021.

[113] W. Zhu, K. S. Tai, S. M. Mousavi, P. Bailis, and G. C. Beroza, "An end-to-end earthquake detection method for joint phase picking and association using deep learning," *Journal of Geophysical Research: Solid Earth*, vol. 127, no. 3, p. e2021JB023283, 2022.

[114] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. CONF, IEEE Signal Processing Society, 2011.

[115] B. H. Huang and L. R. Rabiner, "Hidden markov models for speech recognition," *Technometrics*, vol. 33, no. 3, pp. 251–272, 1990.

[116] D. Yu and L. Deng, *Automatic speech recognition*, vol. 1. Springer, 2016.

[117] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[118] N. B. Yoma, F. R. McInnes, M. A. Jack, S. D. Stump, and L. L. Ling, "On including temporal constraints in viterbi alignment for speech recognition in noise," *IEEE*

transactions on speech and audio processing, vol. 9, no. 2, pp. 179–182, 2001.

[119] J. Münchmeyer, J. Woollam, A. Rietbrock, F. Tilmann, D. Lange, T. Bornstein, T. Diehl, C. Giunchi, F. Haslinger, D. Jozinović, *et al.*, "Which picker fits my data? a quantitative evaluation of deep learning based seismic pickers," *Journal of Geophysical Research: Solid Earth*, vol. 127, no. 1, p. e2021JB023499, 2022.

[120] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural computation*, vol. 10, no. 7, pp. 1895–1923, 1998.

[121] Y. Huang, M. Slaney, M. L. Seltzer, and Y. Gong, "Towards better performance with heterogeneous training data in acoustic modeling using deep neural networks," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[122] V. Manohar, D. Povey, and S. Khudanpur, "Semi-supervised maximum mutual information training of deep neural network acoustic models," in *Sixteenth annual conference of the international speech communication association*, 2015.

[123] H. Kuehne, A. Richard, and J. Gall, "A hybrid rnn-hmm approach for weakly supervised temporal action segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 4, pp. 765–779, 2018.

[124] W. Ying, L. Zhang, and H. Deng, "Sichuan dialect speech recognition with deep lstm network," *Frontiers of Computer Science*, vol. 14, pp. 378–387, 2020.

[125] L. Liu, T. Hueber, G. Feng, and D. Beautemps, "Visual recognition of continuous cued speech using a tandem cnn-hmm approach.," in *Interspeech*, pp. 2643–2647, 2018.

[126] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[127] C. J. Wu, "On the convergence properties of the em algorithm," *The Annals of statistics*, pp. 95–103, 1983.

Anexo A

Algoritmos implementados en GMM-HMM

A continuación se presentarán dos algoritmos que hacen uso de técnicas de programación dinámica: *Algoritmo Forward* y *Algoritmo Backward*.

A.1. Algoritmo Forward

Considérese la siguiente variable $\alpha_t(i)$, definida de la siguiente manera

$$\alpha_t(i) = \mathbb{P}\left(o_1, o_2, \ldots, o_t, q_t = i \mid \lambda\right) \tag{A.1}$$

Esta variable representa la probabilidad de observar la secuencia o_1,o_2,...,o_T, y estar en el estado i en el instante t. Esta variable se puede calcular iterativamente para todos los estados del modelo en los diferentes instantes de tiempo, como se puede observar en la siguiente figura A.1.

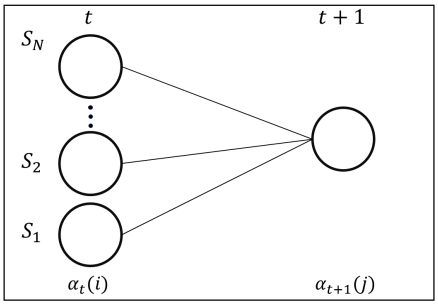


Figura A.1: Algoritmo Forward

El proceso que se observa en la figura A.1 se describe en el siguiente algoritmo:

Algoritmo A.1: Algoritmo Forward

1 Inicialización.

$$\alpha_1(i) = \pi_i b_i(o_1), \quad 1 \leq i \leq N \tag{A.2}$$

2 Inducción.

$$\alpha_{t+1}(j) = b_j(\mathbf{o}_{t+1}) \sum_{i=1}^{N} \alpha_t(i) a_{ij}, \quad 1 \leq j \leq N \tag{A.3}$$

3 Finalización.

$$\mathbb{P}(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(si) \tag{A.4}$$

En cuanto a cálculos numéricos, este algoritmo necesita $N(N+1)(T-1)+N$ multiplicaciones y $N(N-1)(T-1)$ adiciones. Al utilizar la inducción, se reduce en forma significativa el proceso de computación que se observó con anterioridad, lo cual lo convierte en un método bastante conveniente.

A.2. Algoritmo Backward

En forma análoga al algoritmo *Forward*, se define la variable $\beta_t(i)$ como:

$$\beta_t(i) = \mathbb{P}(o_{t+1}, o_{t+2}, \ldots, o_T \mid q_t = i, \lambda) \tag{A.5}$$

Esta variable representa la probabilidad de observar la secuencia o_1,o_2,...,o_T desde $t+1$ hasta el final (T), dado el estado i en el instante de tiempo t y el modelo λ. Es importante destacar que la definición de la variable hacia adelante es una probabilidad conjunta, mientras que la probabilidad hacia atrás es una probabilidad condicional. De manera similar, la variable $\beta_t(i)$ se puede calcular inductivamente, como se puede observar en la figura A.2.

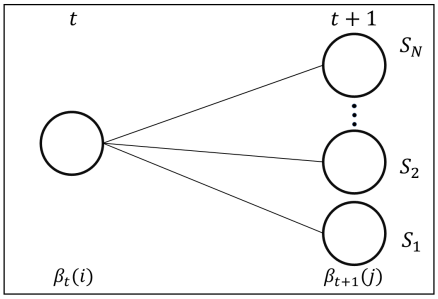


Figura A.2: Algoritmo Backward

El proceso que se observa en la figura A.2 se describe en el siguiente algoritmo:

Algoritmo A.2: Algoritmo Backward

1 Inicialización.

$$\beta_T(i) = 1/N_F \quad \forall 1 \leq i \leq N \tag{A.6}$$

2 Inducción.

$$\beta_t(j) = \left[\sum_{i=1}^{N} a_{ji} \cdot b_i\left(o_{t+1}\right) \cdot \beta_{t+1}(i)\right] \quad \begin{matrix} t = T-1, T-2, \ldots, 1 \\ 1 \leq j \leq N \end{matrix} \tag{A.7}$$

3 Finalización.

$$\mathbb{P}(O \mid \lambda) = \sum_{i=1}^{N} \beta_1(i)\pi_i b_i\left(o_1\right) \tag{A.8}$$

El costo computacional que presenta este algoritmo es del orden de $O(N^2T)$, al igual que el algoritmo *Forward*.

A.3. Algoritmo de Viterbi

EL Algoritmo de Viterbi fue propuesto en 1967 como un método para decodificar códigos convolucionales [117]. El objetivo de este algoritmo es descubrir la secuencia de estados más probable para una secuencia de estados observados y un modelo λ. Para ello se considera la variable $\delta_t(i)$, que se define como:

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} \mathbb{P}\left(q_1, q_2, \ldots, q_t = i, o_1, o_2, .., o_t \mid \lambda\right) \tag{A.9}$$

Esta nueva variable corresponde a la máxima probabilidad que representa una secuencia de estados que finaliza en el estado i, teniendo en cuenta las primeras t observaciones. Por inducción $\delta_{t+1}(i)$ puede ser encontrado como

$$\delta_{t+1}(i) = b_i\left(\mathbf{o}_{t+1}\right) \max_{1 \leq i \leq N} \left[\delta_t(i) a_{ij}\right] \tag{A.10}$$

Tras definir las N trayectorias más probables, se tiene que recuperar la secuencia de estados *óptima* mediante un proceso de retroceso. Para lo anterior, se requiere realizar un seguimiento del argumento que maximiza la ecuación A.10, por lo que se define una nueva variable, $\psi_t(i)$, como la matriz en la que se almacenaría las secuencias de estados de todas las N trayectorias. A continuación se presenta el desarrollo de este algoritmo para la construcción de la mejor secuencia de estados.

Algoritmo A.3: Algoritmo de Viterbi

1 Inicialización.

$$\begin{aligned} \delta_1(i) &= \pi_i b_i\left(o_1\right), \quad 1 \leq i \leq N \\ \psi_1(i) &= 0, \quad 1 \leq i \leq N \end{aligned} \tag{A.11}$$

2 Inducción.

$$\begin{aligned} \delta_t(i) &= b_j\left(o_t\right) \max_{1 \leq i \leq N} \delta_{t-1}(i) a_{ij}, \quad 1 \leq j \leq N \\ \psi_t(i) &= \arg \max_{1 \leq i \leq N} \left[\delta_{t-1}(i) a_{ij}\right], \quad 1 \leq j \leq N \end{aligned} \tag{A.12}$$

3 Finalización.

$$\begin{aligned} \mathbb{P}\left(O q^* \mid \lambda\right) &= \max_{1 \leq i \leq N} \left[\delta_T(i)\right] \\ q_T^* &= \arg \max_{1 \leq i \leq N} \left[\delta_T(i)\right] \end{aligned} \tag{A.13}$$

4 Reconstrucción secuencia de estados.

$$q_t^* = \psi_{\mathrm{t}+1}\left(q_{t+1}^*\right) \quad \begin{array}{c} t = T-1, T-2, \ldots, 1 \\ 1 \leq q_{t+1}^* \leq N \end{array} \tag{A.14}$$

A.4. Algoritmo de Baum-Welch

El algoritmo de Baum-Welch que se aplica sobre el modelo estocástico representado por los HMM se deriva del algoritmo de Maximización de Expectativa (EM) [61]. El algoritmo EM es un método iterativo utilizado para encontrar estimaciones de máxima verosimilitud (local) de parámetros en modelos estadísticos, donde el modelo depende de variables latentes no observadas [126]. Este algoritmo itera en dos pasos:

I. **Paso E**: Calcula la esperanza de la verosimilitud mediante la inclusión de variables latentes como si fueran observables a partir de la estimación actual de los parámetros del modelo

II. **Paso M**: Calculan los parámetros del modelo que maximizan la verosimilitud a partir de la maximización de la verosimilitud esperada del paso E

Estos dos pasos se repiten hasta cumplir algún criterio de convergencia. En el caso del algoritmo Baum-Welch, en cada iteración se calcula un nuevo conjunto de $\lambda^* = (A^*, B^*, \pi^*)$ a partir de los parámetros actuales $\lambda = (A, B, \pi)$, para seguir aumentando la verosimilitud hasta el punto en el que el algoritmo converja o el incremento de verosimilitud sea mínimo mediante la siguiente expresión

$$\mathbb{P}\left(O \mid \lambda^*\right) \geq \mathbb{P}(O \mid \lambda) \tag{A.15}$$

Estará garantizada la convergencia del algoritmo a un óptimo local [127], mientras que el espacio de parámetros sea continuo y no presente restricciones de transición. Para describir el procedimiento de reestimación de manera formal, en primer lugar se procederá a describir los elementos necesarios para estimar los parámetros del modelo en el paso E.

En primer lugar se define la variable $\xi_t(i,j)$ como

$$\xi_t(i,j) = \mathbb{P}\left(q_t = i, q_{t+1} = j \mid O, \lambda\right) \tag{A.16}$$

la cual corresponde a la probabilidad de transición de un estado i a un estado j dado un modelo λ y una secuencia de observación O. En función de las variables *Forward* y *Backward* previamente calculadas, la variable $\xi_t(i,j)$ queda de la forma

$$\xi_t(i,j) = \frac{\alpha_t(i) \cdot a_{ij} b_j\left(o_{t+1}\right) \cdot \beta_{t+1}(j)}{\mathbb{P}(O \mid \lambda)} \tag{A.17}$$

Considérese además la variable $\gamma_t(i)$, definida como

$$\gamma_t(i) = \mathbb{P}\left(q_t = i \mid O, \lambda\right) = \sum_{j=1}^{N} P\left(q_t = i, q_{t+1} = j \mid O, \lambda\right) = \sum_{j=1}^{N} \xi_t(i,j) \tag{A.18}$$

La expresión anterior corresponde a las probabilidades a posteriori de que el proceso se encuentre en el estado i en el instante t, dada una secuencia de observaciones O y un modelo λ.

Las variables $\xi_t(i,j)$ y $\gamma_t(i)$ son indispensables para calcular las frecuencias en las transiciones hacia un estado y entre dos estados. Por consiguiente, se tiene que

$$\gamma_1(i) = \text{Probabilidad de iniciar en el estado } i \tag{A.19}$$

$$\sum_{t}^{T-1} \gamma_t(i) = \text{Número esperado de transiciones desde el estado } i \text{ en } O \tag{A.20}$$

$$\sum_{t=1}^{T-1} \xi_t(i,j) = \text{Número esperado de transiciones del estado } i \text{ al estado } j \text{ en } O \tag{A.21}$$

Una vez calculados estos parámetros se procede a calcular el paso M, el cual maximiza la función $Q(\lambda, \bar{\lambda})$, definida como:

$$Q(\lambda, \bar{\lambda}) = \sum_{Q} \mathbb{P}(Q \mid O, \lambda) \log\Big(\mathbb{P}(O, Q \mid \bar{\lambda})\Big) \tag{A.22}$$

donde λ corresponde a los parámetros actuales del HMM y $\bar{\lambda}$ a los parámetros del modelo resultante de la reestimación. Entonces, maximizar esta función dará lugar a los nuevos parámetros de la estructura $\bar{\lambda}$ del HMM, determinados por las siguientes fórmulas:

$$\pi_i = \gamma_1(i) \tag{A.23}$$

$$a_{i,j} = \frac{\sum_{t=1}^{L-1} \xi_{i,j}(t)}{\sum_{t=1}^{L-1} \gamma_i(t)} \tag{A.24}$$

$$b_i(k) = \frac{\sum_{t=1}^{L-1} \delta_{o_t,v_k} \gamma_i(t)}{\sum_{t=1}^{L-1} \gamma_i(t)} \tag{A.25}$$

El algoritmo de EM garantiza que maximizando la función A.22 respecto a los nuevos parámetros, se obtendrá una mayor verosimilitud en la siguiente iteración, cumpliéndose que

$$\max_{\delta}[Q(\delta \mid \bar{\delta})] \Rightarrow \mathbb{P}O \mid \bar{\delta}) \geq \mathbb{P}(O \mid \delta) \tag{A.26}$$

Lo anterior implica que si se repite el proceso considerando $\lambda = \bar{\lambda}$ en cada iteración, la verosimilitud irá aumentando hasta el punto en el que el algoritmo converja o el incremento de verosimilitud sea mínimo.

Anexo B

Aplicación de EM al aprendizaje de la GMM-HMM

A continuación se verá la estimación de los parámetros de máxima verosimilitud y, en particular, la aplicación del algoritmo EM para resolver el problema de aprendizaje del HMM. Se considerará un HMM gaussiano para describir los pasos E y M.

Comenzando por el paso E, se simplificará la expectativa condicional $Q(\lambda, \bar{\lambda})$ (ecuación A.22) de una forma adecuada para que posteriormente sea maximizada en el paso M. Para simplificar la escritura, denotaremos

$$N_t(i) = -\frac{D}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma_i| - \frac{1}{2}(o_t - \mu_i)^T \Sigma_i^{-1}(o_t - \mu_i) \tag{B.1}$$

La ecuación B.1 corresponde al logaritmo de la PDF gaussiana (ecuación 2.39) asociada al estado i. A continuación se define dos elementos necesarios para estimar los parámetros del modelo:

I. Las probabilidades a posteriori de transición de estado.

$$\xi_t(i,j) = \frac{\alpha_t(i)\beta_{t+1}(j)a_{ij}\exp\left(N_{t+1}(j)\right)}{\mathbb{P}\left(o_1^T \mid \bar{\lambda}\right)} \tag{B.2}$$

para $t = 1, 2, \ldots, T-1$. Las α y β variables corresponden a las variables *Forward* y **Backward**, respectivamente.

II. Probabilidades a posteriori de que el proceso se encuentre en el estado i en el instante t.

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{B.3}$$

para $t = 1, 2, \ldots, T-1$

Una vez calculados estos parámetros se procede a calcular el paso M, el cual maximiza la función $Q(\lambda, \bar{\lambda})$. A continuación se presentan los resultados que se obtienen tras la maximización, los cuales utilizan los parámetros de las ecuaciones B.2 y B.3, como se puede ver en las siguientes ecuaciones:

I. Reestimación de las probabilidad de transición.

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{B.4}$$

II. Fórmula de reestimación compacta para la matriz de covarianza en el estado i.

$$\hat{\Sigma}_i = \frac{\sum_{t=1}^{T} \gamma_t(i) \left(\mathbf{o}_t - \hat{\mu}_i\right) \left(\mathbf{o}_t - \hat{\mu}_i\right)^\top}{\sum_{t=1}^{T} \gamma_t(i)} \tag{B.5}$$

para cada estado $i = 1, 2, \ldots, N$

III. Reestimación de la media del estado i.

$$\hat{\mu}_i = \frac{\sum_{t=1}^{T} \gamma_t(i) \mathbf{o}_t}{\sum_{t=1}^{T} \gamma_t(i)} \tag{B.6}$$