

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



“SPEECH RECOGNITION EN ESPAÑOL USANDO DEEP LEARNING”

RODRIGO IGNACIO GONZÁLEZ SMITH

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Héctor Allende
Profesor Correferente: Carlos Valle

Noviembre - 2019

DEDICATORIA

Dedico este trabajo a mis hermanos y padres por el apoyo constante en mi formación.
También a mis abuelos Alicia y Gustavo, a quienes siempre recordaré por su cariño
incondicional a pesar de mis errores.

AGRADECIMIENTOS

Agradezco a mi familia por ayudarme en estos años de estudio, siempre con consejos y palabras adecuadas. A mi madre por su enorme paciencia en los malos momentos y ser en quien siempre pude encontrar el mejor apoyo emocional. También a mi padre por la preocupación constante en el proceso a pesar de la distancia, teniendo confianza en mí sin reparo alguno.

A mis profesores guías, por brindarme orientación en cada problema que tuve en el desarrollo del trabajo. Mencionando su buena voluntad para resolver dudas y encontrar espacios para tener varias reuniones.

A todos mis amigos tanto dentro como fuera de la universidad, quienes con su compañía hicieron de la etapa universitaria una mucho mejor experiencia y de los que pude aprender elementos valiosos en el transcurso.

Finalmente a la empresa Zeke la cual me aceptó como memorista, me ayudó en mis dudas y mostró en todo momento un excelente profesionalismo.

RESUMEN

El reconocimiento de habla se refiere a la capacidad que tiene un sistema para transcribir internamente una señal acústica, por medio de la identificación de palabras, fonemas o caracteres. En los últimos años, los modelos de esta área han logrado tasas de error muy bajas para el inglés, sin embargo para otros idiomas la cantidad de datos y trabajos disponibles disminuye drásticamente. Particularmente en español la mayoría de los acercamientos se realizan con modelos tradicionales basados en cadenas de Markov, usualmente centrándose en resolver la problemática para un acento en particular. En esta memoria se propone generar una estrategia distinta, basándose en trabajos con redes neuronales profundas con un enfoque español multi-acento. Agrupando fuentes gratuitas de datos de habla, por medio de un pre-procesamiento adecuado, se genera un conjunto de datos para entrenar de una duración de aproximadamente 200 horas. Se realiza una variación de hiperparámetros, a partir de una red neuronal recurrente con capas *BiLSTM* y la función de pérdida *CTC*. Explorando el desempeño con cambios en la normalización de la entrada, optimizador, número de capas, utilización de capas convolución y regularización. Se validan 3 modelos finales por medio de la tasa de error por carácter y palabra en 2 conjuntos de pruebas de distintas características, que permitan comparar con un modelo *baseline* de tipo *GMM-HMM*. Se logra superar las métricas obtenidas por este modelo, y se realiza un análisis de las diferencias en los resultados entre estos dos acercamientos.

Palabras Clave— Redes Neuronales Profundas, Reconocimiento de Habla, CTC.

ABSTRACT

Abstract—

Speech recognition refers to the ability of a system to internally transcribe an acoustic signal, through the identification of words, phonemes or characters. In recent years, models in this area have achieved very low error rates for English, however for other languages the amount of data and work available decreases drastically. Particularly in Spanish, most of the approaches are carried out with traditional models based on Markov chains, usually focusing on solving the problem for a particular accent. In this work it is proposed to generate a different strategy, based on studies with deep neural networks, with a Spanish multi-accent approach. Grouping free sources of speech data, by means of an appropriate preprocessing, a set of data for training of a duration of approximately 200 hours is generated. A variation of hyperparameters is made, from a recurrent neural network with layers *BiLSTM* and the loss function *CTC*. Exploring performance with changes in input normalization, optimizer, number of layers, use of convolution and regularization layers. The 3 final models are validated by means of the error rate per character and word in 2 sets of tests of different characteristics, which allow comparison with a model *baseline* of type *GMM-HMM*. It is possible to overcome the metrics obtained by this model, and an analysis of the differences in the results between these two approaches is carried out.

Keywords— Deep Neural Networks, Speech Recognition, CTC.

GLOSARIO

ANN: Artificial Neural Network o Red Neuronal Artificial.

ASR: Automatic Speech Recognition o Reconocimiento del habla automático.

Baseline: En un estudio es el resultado de un modelo referencial con el cual se puede comparar uno nuevo.

CER: Tasa de error por carácter.

CTC: Connectionist Temporal Classification.

Corpus: Conjunto de datos destinado a la investigación científica, usualmente textos.

DNN: Deep Neural Network o Red Neuronal Profunda.

HMM: Hidden Markov Models o Modelos Ocultos de Markov.

Input: Entrada de un modelo o sistema.

Loss: Costo de la función de pérdida.

LSTM: Long-Short Term Memory.

MFCC: Se refiere a Mel Frequency Cepstral Coefficients o Coeficientes Cepstrales en las Frecuencias de Mel.

Output: Salida de un modelo o sistema.

Pipeline: Flujo por el cual se realiza un proceso con entrada y salida definida. el cual puede contar con varias piezas.

Target: Clase la cual se desea predecir por medio de un modelo y su entrada respectiva.

UTFSM: Universidad Técnica Federico Santa María.

WER: Tasa de error por palabra.

ÍNDICE DE CONTENIDOS

RESUMEN	IV
ABSTRACT	V
GLOSARIO	VI
ÍNDICE DE FIGURAS	IX
ÍNDICE DE TABLAS	X
INTRODUCCIÓN	1
CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA	3
1.1 Contexto de la Memoria	3
1.2 Macro problema	3
1.3 Impacto inicial de solucionar el problema	4
1.4 Participantes de la problemática (<i>Stakeholders</i>)	4
1.5 Objetivos y alcances	6
1.5.1 Objetivo general	6
1.5.2 Alcance	6
1.5.3 Objetivos específicos	6
CAPÍTULO 2: MARCO CONCEPTUAL	7
2.1 Reconocimiento de habla	7
2.1.1 Componentes claves	8
2.1.2 Avances Speech Recognition	8
2.2 Extracción características de señales acústicas	11
2.3 Modelos Ocultos de Markov	14
2.3.1 HMM en <i>Speech Recognition</i>	14
2.3.2 Modelos de Mezclas Gaussianas	15
2.4 Redes neuronales	16
2.4.1 Redes neuronales recurrentes	18
2.4.2 Capas LSTM	19
2.4.3 Redes neuronales convolucionales	20
2.4.4 Optimización parámetros	21
2.4.5 Connectionist Temporal Classification	24
CAPÍTULO 3: PROPUESTA DE SOLUCIÓN	27
3.1 Metodología	27
3.2 Fuentes de datos	29
3.2.1 Ciempiess	29
3.2.2 Common Voice	29

3.2.3	Crowdsourced high-quality Spanish audios	30
3.2.4	Dimex-100	32
3.2.5	M-AILABS Speech Dataset	32
3.2.6	TEDx Spanish Corpus	33
3.2.7	Voxforge Spanish Corpus	33
3.2.8	Final	33
CAPÍTULO 4: CONSTRUCCIÓN E IMPLEMENTACIÓN DE LA SOLUCIÓN		36
4.1	Pre-Procesamiento	36
4.1.1	Modificación corpus	36
4.1.2	Distribuciones y partición corpus	37
4.2	Extracción de características	38
4.3	Arquitecturas redes neuronales	39
4.3.1	Transformación entrada	40
4.3.2	Optimizador	41
4.3.3	Número de capas	43
4.3.4	Convolución	44
4.3.5	Regularización	46
4.4	Modelos finales	48
CAPÍTULO 5: VALIDACIÓN DE LA SOLUCIÓN		52
5.1	Métricas de validación modelos	52
5.2	Modelo final de reconocimiento de habla	52
5.3	Comparación modelo <i>baseline</i>	54
CAPÍTULO 6: CONCLUSIONES		56
6.1	Aprendizaje y trabajo realizado	56
6.2	Objetivos	56
6.3	Trabajo futuro	57
REFERENCIAS BIBLIOGRÁFICAS		58

ÍNDICE DE FIGURAS

2.1	Línea cronológica respecto a trabajos sobre <i>Speech Recognition</i>	9
2.2	Esquema de un sistema de <i>Speech Recognition</i>	10
2.3	: Pipeline procesado de audios.	13
2.4	: Ejemplo de estados internos junto a sus observaciones para un HMM basado en fonemas. Aquí el vector Y es la señal acústica y a_{ij} son las probabilidades de transición del estado i a j . Finalmente $b_i(y_t)$ corresponde a la probabilidad de emisión del estado i respecto al <i>frame</i> y_t	16
2.5	Ejemplo de una red <i>feedfoward</i> simple.	17
2.6	: Capa recurrente desenrollada.	18
2.7	: Diagrama de unidades LSTM.	20
2.8	Red neuronal recurrente bidireccional.	21
2.9	Red neuronal con capas convolucionales, para clasificar dígitos.	22
2.10	Ejemplo de convolución 2D con kernel 2x2.	23
2.11	Matriz simple de caminos en CTC.	26
3.12	Distribución de la duración de los audios para el conjunto final.	34
3.13	Balance de géneros en <i>set</i> de entrenamiento final.	35
3.14	Distribución de acentos en la agrupación final de los corpus.	35
4.15	Diagrama de ejemplo respecto de codificación letra a número.	37
4.16	Distribución de la duración de los audios menores a 9[s], para el conjunto final.	38
4.17	Distribución de clases (letras) para la muestras de 100 [hrs] y 200[hrs].	39
4.18	Distribución de géneros para la muestra de 100 [hrs] (izquierda) y 200[hrs] (derecha).	39
4.19	Distribución de acentos para la muestra de 100 [hrs] y 200[hrs].	40
4.20	Comparación de la frase “no estamos solos” en su representación como señal de audio (arriba) y Espectrograma en la escala de Mel (abajo).	41

4.21	Diagrama respecto a la arquitectura base. La penúltima capa llamada <i>time distributed</i> , se agrega para referirse a que cada <i>timestep</i> producirá una distribución de probabilidad en las clases de la capa <i>softmax</i>	42
4.22	Comparación respecto a transformaciones del espectrograma de entrada. . .	43
4.23	Error de entrenamiento/validación según tipo de input [2-10 épocas].	44
4.24	Error de entrenamiento/validación según optimizador [2-10 épocas].	45
4.25	Error de entrenamiento/validación según cantidad de capas/ <i>layers</i> [2-10 épocas].	45
4.26	Diagrama respecto a la arquitectura base ocupando como entradas capas convolucionales, se debe notar el uso de <i>reshape</i> con el fin de concatenar los filtros resultantes en el eje de las frecuencias (mismo largo temporal). . .	46
4.27	Error de entrenamiento/validación según capas convolucionales [2-10 épocas].	47
4.28	Error de entrenamiento/validación según capas convolucionales (sin la arquitectura con 3 capas) [2-10 épocas].	48
4.29	Comparación respecto a técnicas de aumento de datos para el espectrograma de entrada.	49
4.30	Error de entrenamiento/validación según tipo de regularización [2-10 épocas].	50
4.31	Error de entrenamiento/validación según tipo de combinación de regularizaciones [2-10 épocas].	50
4.32	Error de entrenamiento/validación según modelo final [1-50 épocas].	51
5.33	Ejemplo de distribución de probabilidad para cada t respecto a las clases para la frase “pues”.	53

ÍNDICE DE TABLAS

1.1	Ejemplo interesados en solución de problemática. Fuente: Elaboración Propia.	5
3.2	Características generales corpus Ciempiess. Fuente: Elaboración Propia. . . .	29
3.3	Características generales corpus Common Voice. Fuente: Elaboración Propia.	30

3.4	Duración y cantidad de audios para <i>Argentinian speech dataset</i> . Fuente: Elaboración Propia.	30
3.5	Características generales corpus <i>Chilean speech dataset</i> . Fuente: Elaboración Propia.	31
3.6	Características generales corpus <i>Colombian speech data set</i> . Fuente: Elaboración Propia.	31
3.7	Características generales corpus <i>Peruvian speech data set</i> . Fuente: Elaboración Propia.	31
3.8	Características generales corpus <i>Puerto Rico speech data set</i> . Fuente: Elaboración Propia.	32
3.9	Características generales corpus <i>Venezuelan speech data set</i> . Fuente: Elaboración Propia.	32
3.10	Características generales corpus Dimex-100. Fuente: Elaboración Propia. . . .	32
3.11	Características generales corpus M-AILABS. Fuente: Elaboración Propia. . . .	33
3.12	Características generales corpus TEDx. Fuente: Elaboración Propia.	33
3.13	Características generales corpus <i>Voxforge Spanish</i> . Fuente: Elaboración Propia.	33
3.14	Características generales corpus Final. Fuente: Elaboración Propia.	34
4.15	Métricas de validación capas convolucionales. Fuente: Elaboración Propia. . .	46
4.16	Métricas de validación tipo de regularización. Fuente: Elaboración Propia. . .	48
5.17	Tasa de error por carácter (<i>CER</i>) modelos finales. Fuente: Elaboración Propia.	52
5.18	Tasa de error por palabra (<i>WER</i>) modelos finales. Fuente: Elaboración Propia.	52

INTRODUCCIÓN

Muchas cosas en nuestro mundo se pueden modelar por medio de señales continuas o discretas tal como imágenes, sonidos, etc. Por lo cual nuestras conversaciones, las palabras que usamos y todo nuestro discurso o *speech* también puede ser captado por medio de un aparato electrónico para ser digitalizado y posteriormente usado para un fin en particular. Aquí, nace el interesante campo del *speech recognition* o reconocimiento de habla, el cual se define usualmente como el proceso en el cual una máquina logra identificar letras, palabras o más bien crear una transcripción de la señal de voz. Por ello se comenta que el reconocimiento del habla hace posible que las computadoras puedan entender e identificar los lenguajes humanos.

Las soluciones actuales presentan ser eficientes y efectivas en los casos ideales, como por ejemplo los servicios que prestan grandes y reconocidas empresas dentro de las cuales se puede nombrar Amazon, Google, Microsoft entre otras. Sin embargo, basta con examinar el desempeño de los subtítulos automáticos en Youtube respecto a alguna persona que tenga una pronunciación propia de su país, tal como el español de Chile, para poder darse cuenta que no es totalmente efectivo.

En general para los estudios en el dominio del español, se suelen utilizar modelos híbridos en la resolución de la problemática en los cuales se debe calibrar y ajustar varias partes que terminan mostrando un *pipeline* poco intuitivo. Además los datos necesarios para crear estos modelos suelen ser limitados para cualquier lenguaje que no sea el inglés, por lo cual encontrar un *set* de datos adecuados se convierte en una tarea demorosa por sí misma. Es por ello, que en el trabajo presente es de interés plantear un acercamiento más directo y simple por medio de un modelo basado completamente en *Deep Learning*. En el cual se exploren fuentes de datos y alternativas en la arquitectura de la red neuronal para el problema, que aporte respecto a los corpus disponibles como también al uso de este acercamiento para el reconocimiento de habla en español.

El documento ha sido estructurado de la siguiente manera:

1. **Definición del Problema**, se planteará más a fondo la problemática que se desea solucionar, el impacto respectivo, la razón del actual trabajo y los objetivos de este.
2. **Marco Conceptual**, se explica conceptos teóricos base que se usarán en el desarrollo de la memoria además de una revisión de los aportes más importantes en la literatura respecto al *Speech Recognition*.
3. **Propuesta de solución**, se explicará la metodología a utilizar y el proceso de desarrollo del modelo. Entrando en detalle respecto a los datos que se juntaron para el trabajo en la memoria y la técnica a utilizar.
4. **Desarrollo de la solución**, se mostrará el trabajo realizado en el período de la memoria

abarcando el procesado de datos, la arquitectura y las mejoras respectivas. Todo esto con el objetivo de mejorar una métrica y realizar un proceso adecuado respecto a la herramienta.

5. **Validación de la solución**, aquí se realizará el análisis de los resultados de los modelos finales, comparando con un sistema base para los conjuntos de pruebas respectivos.
6. **Conclusiones**, se comentará los resultados obtenidos respecto a los objetivos, lo aprendido en el trabajo, posibilidades de continuación y mejoras respecto a la solución final.

CAPÍTULO 1

DEFINICIÓN DEL PROBLEMA

1.1. Contexto de la Memoria

La memoria se realiza en el contexto de un trabajo de título de pregrado del departamento de Informática UTFSM, por otra parte estará centrada principalmente en el área de *Machine Learning*. Esto dado a que las técnicas más efectivas para el reconocimiento de habla están contenidas por esta área de Inteligencia Artificial, notar que se trabajará con audios de hablantes de español. Los datos serán recopilados desde distintas fuentes gratuitas de internet, algunas de las cuales fueron previamente investigadas antes de iniciar el trabajo.

1.2. Macro problema

Actualmente la generación automática de subtítulos por ejemplo en Youtube, es creada por medio de sistemas de reconocimiento de habla automático. Tales sistemas son bastantes efectivos en casos ideales, con condiciones de audio adecuadas y una pronunciación genérica. Empeorando rápidamente sus métricas tales como el WER o tasa de error por palabra si se transcribe un discurso con una pronunciación particularmente mala o con condiciones ruidosas, tal como se puede ver en la charla [[Jyothi, 2017](#)], donde se ven varios ejemplos de subtitulado automático para muestras “difíciles”. Los principales factores que dificultan crear un sistema general [[Benzeghiba et al., 2007](#)] para el reconocimiento de habla son:

1. Acentos propios regionales.
2. Fisiología del hablante.
3. Estilo del habla y espontaneidad.
4. Velocidad del habla.

Sumado a esto para idiomas fuera del inglés, como el español, los corpus de datos gratis son de mucho menor volumen (duración), o no están formateados adecuadamente para el entrenamiento de un modelo de clasificación. Estas aristas limitan las posibilidades a la hora de crear sistemas que utilicen reconocimiento de habla o voz, más aún si se ocupan en un dominio que no ha sido abordado completamente por las soluciones actuales. Esto provoca que se tengan disponibles productos en los cuales los usuarios tengan que adaptarse a los sistemas y no al revés.

Centrando más el problema para el caso de nuestro país, la pronunciación que poseemos se suele comentar no es la más correcta, eso sin contar la cantidad de modismos que se

usan. En general los sistemas de ASR no logran tener los mismos resultados en dominios con pocos datos y/o condiciones poco favorables como se comentó previamente. Por lo cual, sigue siendo un problema el como acercar estas soluciones para contextos con pocos recursos.

1.3. Impacto inicial de solucionar el problema

Los trabajos sobre *Speech Recognition*, ya sean sobre el problema general de transcribir todo el vocabulario de un idioma o un subconjunto de este, como en el caso de un sistema de comandos pueden tener utilidad en muchas áreas. Por ejemplo, en la salud para sistemas de integración, servicios de ayuda al cliente, transcripción en tiempo real de charlas, encontrar palabras claves, etc. Más aún si es respecto a algún dominio en el cual no se poseen muchos estudios y/o productos como fue comentado previamente.

Por lo cual mostrar un acercamiento general, en el cual se analicen distintos factores que afecten la creación de un sistema de reconocimiento de habla puede ser un aporte en los trabajos y/o desarrollos de sistemas de este tipo para el español. En el presente caso, se utilizarán técnicas de *Deep Learning*, por lo cual se utilizarán distintas tecnologías y *frameworks* que están en continuo progreso para el área, de tipo *open source*. Por otra parte el enfoque será multi-acento, para de esta forma intentar abarcar variaciones no trabajadas directamente en la literatura.

1.4. Participantes de la problemática (Stakeholders)

Las características generales de la problemática hacen que sea interesante para una gran gama de aplicaciones y por lo tanto personas. Sin embargo, a modo de simplificación podemos considerar que la solución siempre estará asociada al desarrollo de un producto que aborde la problemática del ASR. Ya sea para todo el vocabulario del español, comandos, etc.

Dentro de la población interesada, los más importantes que podemos listar con su nivel de interés se pueden ver en la Tabla 1.1.

Tabla 1.1: Ejemplo interesados en solución de problemática. Fuente: Elaboración Propia.

Interesado	Tipo	Interés
Usuario	Con algún nivel de discapacidad auditiva.	Alto: Dado que un efectivo reconocimiento de habla puede brindar ayuda en la integración para distintos contextos.
	Casual, uso opcional de funcionalidad de reconocimiento de habla.	Bajo: Sólo en ciertas ocasiones le sería de verdadera utilidad.
Cliente	Tiene como requerimiento prioritario una funcionalidad con ASR.	Alto: Dado que recibir un producto que depende de esta funcionalidad, y está a la vez no logre ser efectiva puede causar una pérdida de recursos.
	El cual tiene como requerimiento no prioritario una funcionalidad con ASR.	Medio: Dado que a pesar de no ser prioritario, es un requerimiento que puede significar más tiempo y/o recursos para la obtención del producto en caso que deba ser desarrollado desde cero.
Empresa/Institución que ofrece un producto	Basado completamente en ASR.	Alto: Debido a la dificultad de comparación con respecto a otras implementaciones, por ejemplo el cambio de idioma.
	Funcionalidad complementaria con ASR.	Medio: Dado que una alternativa de este tipo puede entregar mucho valor para el cliente.
	Componente extra basado en ASR.	Bajo: Si es una funcionalidad la cual no tiene que ver con las primarias, desechable.

1.5. Objetivos y alcances

1.5.1. Objetivo general

Se desea implementar un modelo de *Speech Recognition* o reconocimiento de habla para el español. Orientado a reducir los errores relacionados a la transcripción de los audios de uno o más conjuntos de pruebas previamente seleccionado(s), utilizando redes neuronales artificiales profundas.

1.5.2. Alcance

A pesar de las técnicas que se puedan utilizar para resolver el problema, la mayor limitación son los datos que se lleguen a agrupar. Esto dado a que las mejores técnicas actuales dependen de poseer corpus del orden de las ~ 1000 horas [Chan et al., 2015, Amodei et al., 2016], es por ello que el alcance se verá limitado en parte al tiempo que se emplee en crear un *dataset* y/o la calidad de los existentes (gratuitos).

1.5.3. Objetivos específicos

1. Diseñar un modelo de *Speech Recognition* que este orientado a lograr un *CER* y *WER* o tasa de error por carácter y palabra respectivamente, menor o igual al de un modelo *baseline* para un set de pruebas en español.
2. En este contexto (idioma español) usar un acercamiento que no haya sido propuesto previamente, e implementarlo acorde a decisiones basadas en la teoría.
3. Crear un modelo que cumpla con ser eficaz y con un tiempo de predicción similar al del modelo *baseline*.

CAPÍTULO 2

MARCO CONCEPTUAL

En este capítulo se explicarán las bases conceptuales en las cuales estará basado el trabajo, también se van a enlazar ideas que estén conectadas en el desarrollo de la memoria y sus objetivos. Se comienza listando conceptos generales respecto al reconocimiento de habla, para proseguir a revisar los avances más importantes en la literatura. Luego, se detallan las técnicas que se utilizarán en el trabajo posterior.

2.1. Reconocimiento de habla

El reconocimiento de habla o *speech recognition* puede ser definido como el proceso en el cual se convierte la señal de habla en la secuencia de palabras correspondiente (o más probable) por medio de un algoritmo computacional. Puede ser definido como un problema de reconocimiento de patrones, según su alcance [Anusuya y Katti, 2010] se clasifica de las siguientes maneras:

- **Speaker Dependent:** Identificación de palabras pertenecientes a un solo hablante, por medio de un aprendizaje previo de las características propias en la señal de su voz.
- **Speaker Independent:** Identificación del discurso para cualquier hablante, generalmente enfocados en un idioma/acento.

En los modelos estadísticos de *speech recognition* se asume un simple modelo probabilístico de producción de habla o *speech* el cual para una secuencia de palabras W , existe una observación acústica Y con probabilidad $P(W, Y)$ [Anusuya y Katti, 2010]. Entonces, el objetivo del modelo es decodificar esta sentencia de palabras, basado en la secuencia acústica observada, tal que tenga mayor probabilidad a-posteriori. Lo que se puede ver como:

$$W = \operatorname{argmax}_{W \in V} P(W/Y).$$

Donde V es el conjunto de todas las secuencias de palabras respectivas a un vocabulario. Usando el teorema de Bayes y sabiendo de antemano que la señal $P(Y)$ es independiente de las palabras W , la ecuación queda como:

$$W = \operatorname{argmax}_{W \in V} P(Y/W)P(W).$$

Entonces se desea obtener la secuencia de palabras más probable, por medio del llamado modelo acústico $P(Y/W)$ y de lenguaje $P(W)$.

No debe ser confundido el reconocimiento de habla con el de voz, o *voice recognition* el cual esta orientado a la identificación del hablante y no las palabras en la señal de voz.

2.1.1. Componentes claves

Entre los sistemas de reconocimiento de habla más tradicionales, suelen existir componentes comunes los cuales son nombrados en [Benzeghiba *et al.*, 2007] y [Anusuya y Katti, 2010]:

1. **Front End:** En el cual se obtiene la señal digital y a partir de esta se realiza el proceso de extracción de características para obtener una representación interna de alto nivel. La más usada MFCC o Coeficientes Cepstrales en las Frecuencias de Mel, también para el caso de algunos trabajos con redes neuronales se utilizan Espectrogramas en la Frecuencia de Mel.
2. **Modelo Acústico:** Nombrado previamente, es el encargado de mapear el vector de características acústica a secuencias de fonemas, grafemas o palabras. Un claro ejemplo es explicado en [Jyothi, 2017]. Para crear este modelo el acercamiento más común era usar GMM o Modelos de Mezcla Gaussiana junto a HMM o Modelos Ocultos de Markov pero una estrategia más reciente se estableció con los modelos de DNN o *deep neural networks* permitiendo un resultado más flexible respecto al *Input*. Estos modelos en acercamientos modernos se entrenan con grandes cantidades de *speech data* o datos de habla, es decir grabaciones de conversaciones o discursos con sus respectivas transcripciones.
3. **Modelo de Lenguaje:** Este método es usado para la elección de la palabra/oración más probable, suele ensamblarse sobre un modelo acústico con el fin de corregir la transcripción interna a una más correcta.
4. **Modelo de pronunciación:** Viene principalmente a ser construido por conocimiento de expertos y lingüistas que estudien los fonemas que constituyen el sonido de las palabras. Estos en términos de *speech* vendrían a ser las sub-unidades que constituyen el sonido al pronunciar una palabra. Por ejemplo, en inglés la pronunciación del número 9 o *nine* viene a estar conformada por 3 fonemas: “n”, “ay” y “n”. Dado que para cada acento existen distintos fonemas al conformar una palabra, en los estudios más recientes se intenta dejar de depender de estas reglas impuestas “manualmente”. Creando modelos acústicos que implícitamente tengan uno de pronunciación, por medio del aprendizaje directo de los datos.

2.1.2. Avances Speech Recognition

Desde hace 70 años han existido muchos acercamientos y distintos trabajos asociados a mejorar los sistemas de *Speech Recognition* o también nombrados *Automatic Speech Recognition* (ASR). Un buen esquema es el entregado por Clark Boyd en la figura Figura 2.1. Los autores Juang y Rabiner en su revisión de estas tecnologías [Juang y Rabiner, 2005] comentan que en la primera mitad del siglo XX se estableció la importancia de la identificación del espectro de las señales para un confiable modelado de la naturaleza fonética de los discursos. Posteriormente, en los 50s se crearon sistemas que reconocerían dígitos y vocales, los cuales eran poco “maleables” y flexibles. Además estos necesitaban pausas entre cada palabra como también una pronunciación muy adecuada. En la década siguiente según

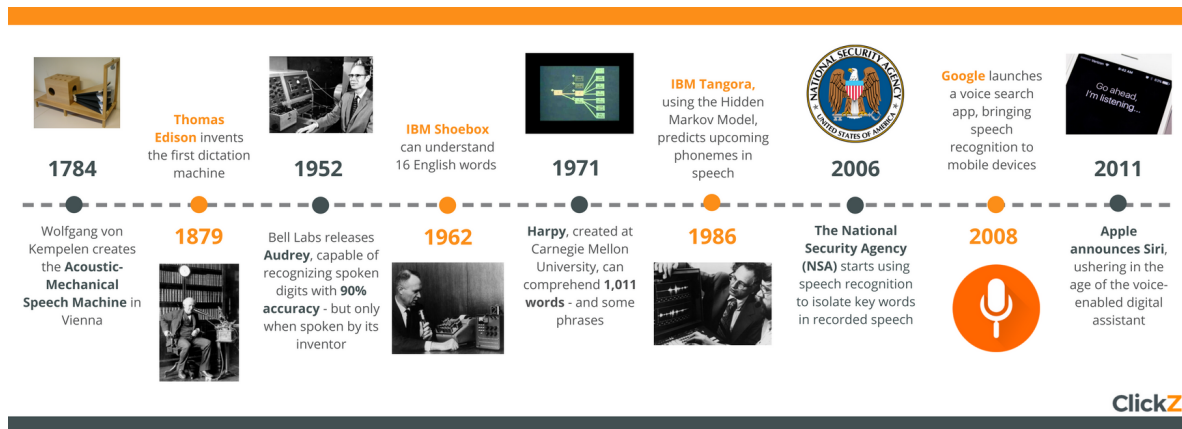


Figura 2.1: Línea cronológica respecto a trabajos sobre *Speech Recognition*.

Fuente: Obtenido de <https://medium.com/swlh/the-past-present-and-future-of-speech-recognition-technology-cf13c179aaf..>

los mismos autores se propusieron las primeras ideas respecto a utilizar tecnologías de reconocimiento de patrones para el problema de *Speech Recognition*. Como se muestra en la Figura 2.1 en 1971 la agencia de proyectos del departamento de defensa de Estados Unidos reveló Harpy el cual fue una innovación para lo realizado hasta el momento, dada la cantidad de vocabulario que podía manejar. Todos los aportes hasta el momento fueron convergiendo para cuando los *HMM* o *Hidden Markov Models* se comenzaron a aplicar a esta problemática en los 80s, generando un enfoque estadístico más riguroso y cambiando el paradigma *straightforward* con el cual se estaban abordando los sistemas ASR. Este nuevo modelo permitía ser resistente a la variabilidad intrínseca de las señales de la voz, tanto por factores de pronunciación y ruido ambiental, entre otros [Juang y Rabiner, 2005, Saini y Kaur, 2013, Graves y Jaitly, 2014].

En tales años también se propuso usar redes neuronales, siendo MLP [Juang y Rabiner, 2005] una alternativa eficaz para casos simples, como reconocer dígitos hablados. Pero solo se limitaban a ello, debido a la naturaleza temporal del problema y que para tales años en su forma original las *artificial neural networks* no probaban ser extensibles para tales tareas. Años después estaban en desarrollo estudios para integrar a las redes neuronales con las estructuras de un *HMM*, las cuales actualmente reemplazaron a los *gaussian mixture models* o Modelos de Mezcla Gaussiana en el modelamiento de la distribución de emisión de las señales de entrada. De tal modo se formulan las arquitecturas híbridas del tipo *DNN-HMM*.

Desde el 2007 en adelante, los sistemas de ASR mantenían muchos componentes en común debido a la utilización del mismo *pipeline*, una estructura general se puede representar con el esquema de la Figura 2.2. La señal de audio se procesa por medio del *front-end*, el modelo acústico genera un hipótesis y el modelo de lenguaje junto al de pronunciación generan la secuencia más probable [Yu y Deng, 2016].

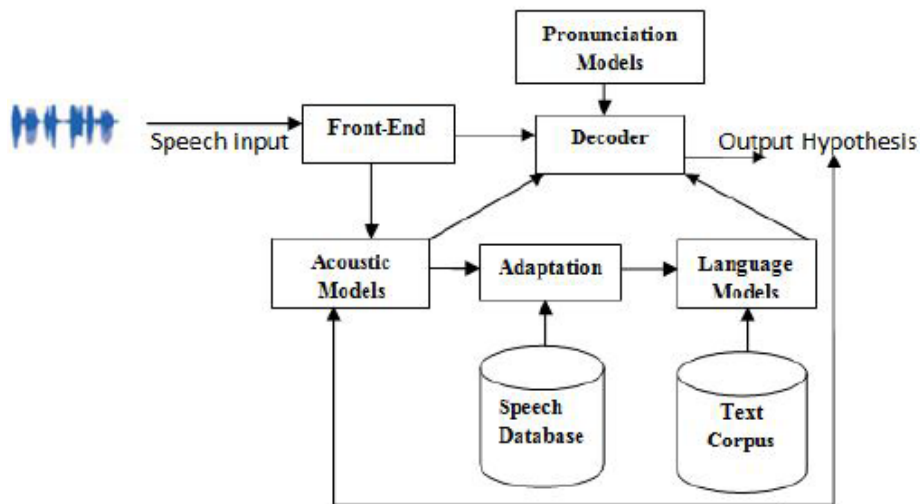


Figura 2.2: Esquema de un sistema de *Speech Recognition*.

Fuente: Obtenido de [Saini y Kaur, 2013].

Dada la simpleza y efectividad para resolver problemas de *machine learning*, las redes neuronales se comienzan a plantear como un reemplazo de los modelos de Markov en los modelos de reconocimiento de habla. Por medio de una orientación *end to end* [Graves et al., 2006, Graves et al., 2013, Graves y Jaitly, 2014, Amodei et al., 2016], es decir juntando el modelo acústico y de pronunciación en la generación de las palabras a través de la señal de entrada/input. Donde la primera formulación exitosa estaba compuesta por el uso de una capa bidireccional *LSTM* y la función objetivo *Connectionist Temporal Classification* que como se explicó previamente permite entrenar un modelo sin necesidad de “alinear” el *input* y *target*. Lo cual presentó ser una nueva innovación en la problemática, dado que el solo hecho de juntar transcripciones es una tarea costosa.

Del mismo modo en [Maas et al., 2015] se lograron mejores resultados al simplificar el modelo en la transcripción a caracteres, por medio de un modelo de lenguaje neuronal. Primero una red recurrente bidireccional profunda mapea el vector de características acústicas a una distribución de probabilidad de caracteres para cada *time-step*, con la *CTC* correspondiente. Posteriormente se enlaza a un modelo de lenguaje asociado a caracteres modelado con una red neuronal, una de las mayores ventajas de esto es que es mucho más rápido de computar en comparación a un modelo de lenguaje basado en palabras. Por otro lado permite manejar mejor las palabras fuera del vocabulario, siendo ideal para el habla espontánea o apresurada.

Finalmente se debe nombrar también a la arquitectura usada en los trabajos [Chan et al., 2015] y [Chiu et al., 2018], *LAS* o *Listen, Attend & Spell*. Esta hace uso del *Listener* que es un *encoder* recurrente piramidal (mejora tiempos de entrenamiento), específicamente un *BLSTM* o *bidirectional LSTM* y una parte *AttendAndSpell*, también de tipo *LSTM* que funciona como capa atencional y *decoder*. Actualmente la competencia por

los mejores resultados se realiza entre las arquitecturas híbridas *DNN-HMM* y los modelos *end-to-end* completamente neuronales.

2.2. Extracción características de señales acústicas

Para que los sistemas de reconocimiento de habla o voz puedan funcionar, necesitan poder tener una representación de la entrada adecuada respecto al *speech*. Esta entrada o *input* usualmente es una representación de alto nivel, que se obtiene después de una serie de transformaciones de la señal inicial [Anusuya y Katti, 2010, Muda et al., 2010]. Esto reduce la dimensionalidad de la entrada, a la vez que mantiene el poder de discriminación entre *frames*. Se prefiere no usar el audio *raw* o puro, dada la cantidad de variabilidad que contiene y el coste que conllevaría entrenar [Kamath y Whitaker, 2019].

Los **Coefficientes Cepstrales de la Frecuencia de Mel** [Muda et al., 2010, Saini y Kaur, 2013, Kamath y Whitaker, 2019] o *MFCC*, al extraerse en base a la frecuencia de Mel, están basados en la percepción auditiva humana, la cual no puede percibir frecuencias sobre los 1[Khz]. Es el método más comúnmente usado en trabajos relacionados a reconocimiento tanto de habla como de voz [Anusuya y Katti, 2010, Yu y Deng, 2016, Kamath y Whitaker, 2019]. El proceso para obtener esta representación se puede ver en la Figura 2.3 y se procede a explicar:

1. **Pre-emphasis ó pre-enfásis:** Paso que sirve como filtro para resaltar las frecuencias más altas, usualmente se ocupa un $\alpha = 0,95$ y la onda resultante se calcula con un filtro de primer orden:

$$Y(n) = X(n) - 0,95X(n - 1).$$

Con esto también se evita la distorsión provocada por el ruido cercano a los componentes de frecuencias más altas.

2. **Framing:** Se refiere a dividir la señal de audio en *frames* o cuadros entre los 20 a 40 [ms]. De esta manera en cada cuadro se asume que la señal es estacionaria, dividiendo con un *overlap* entre *frames* o cuadros contiguos.
3. **Hamming Windowing:** Integra todas las líneas de frecuencia cercana correspondiente a cada *frame*, de tal manera que se tenga una señal más adecuada para el siguiente paso, la ecuación esta definida como:

N = Número de muestras de cada *frame*.

$Y(n)$ = Señal de salida.

$X(n)$ = Señal de entrada.

$W(n)$ = Hamming window.

$$Y(n) = X(n) \times W(n),$$

$$W(n) = 0,54 - 0,46 \cos\left(\frac{2\pi n}{N-1}\right), 0 \leq n \leq N-1.$$

Notar que aquí se crearan ventanas solapadas, y es el principal motivo por el cual se dividen así los *frames*. Dada la forma de la *Hamming Window*, se tiende a perder información en los bordes a pesar de suavizar la curva de la señal inicial.

4. **Fast Fourier Transform:** Ahora se debe transformar cada *frame* de N muestras, a la representación en función de frecuencias por medio de la transformada discreta de *Fourier*. Con lo que se obtendrá un suma de funciones sinusoidales que representen la señal:

$$Y(k) = \sum_{n=0}^{N-1} x(n) e^{-j \frac{2\pi kn}{N}}.$$

Notar que para aumentar la eficiencia, se ocupa el algoritmo llamado *Fast Fourier Transform (FFT)* o Transformada Rápida de Fourier.

$$Y(w) = FFT[h(t) * X(t)] = H(w) * X(w).$$

Se calcula una estimación de la densidad espectral o *periodogram* para cada frecuencia, de esta forma se tiene una representación de que porcentaje de la señal esta en cada w . Su calculo viene dado por:

$$P_i(k) = \frac{1}{N} |Y_i(k)|^2.$$

5. **Mel Filter Bank Processing:** El rango de frecuencias en el espectro de *FFT* es muy grande, además que la señal de voz no esta correctamente modelada linealmente si se desea tener una semejanza con respecto a como el oído humano recibe las señales de voz. Por lo que por medio de los filtros en la escala de Mel [[Muda et al., 2010](#), [Benzeghiba et al., 2007](#), [Kamath y Whitaker, 2019](#)] se desea simular una mejor resolución en bajas frecuencias con respecto a las altas. De esta manera se hace un reducción de dimensión, usualmente con un número de filtros entre 20-40. La siguiente ecuación muestra la equivalencia entre la escala de Mel y el dominio de frecuencias:

$$F(Mel) = 2595 * \log_{10}\left(1 + \frac{f}{700}\right).$$

Además a modo de ilustración se puede ver en la Figura 2.3 el como se comportan estos filtros híbridos. Luego para cada triángulo, se multiplica su correspondencia con las energías calculadas en la transformada discreta de Fourier. El resultado también es

llamado espectrograma en la frecuencia de Mel de la señal, el cual se ve a la derecha. Bajo esta representación es donde existe la motivación de tratar la señal de voz con técnicas de clasificación de imágenes en algunos acercamientos [Amodei et al., 2016].

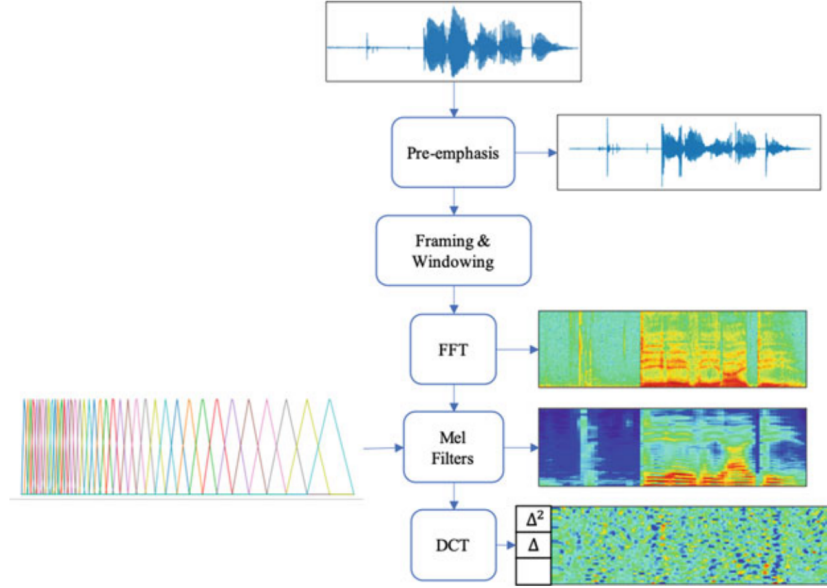


Figura 2.3: : Pipeline procesado de audios.

Fuente: Obtenido de [Kamath y Whitaker, 2019].

6. **Discrete Cosine Transform:** Se calcula el logaritmo de cada uno de los coeficientes obtenidos en el paso anterior y luego se utiliza la transformada discreta del coseno . El resultado es lo que se conoce como *Mel Frequency Cepstrum Coefficients* también llamados *MFCC*, corresponden al espectro o transformada inversa de Fourier de las energías logarítmicas del banco de filtros:

$$S_t = \ln\left(\sum_{n=0}^{N-1} |X_t(n)|^2 H_m(n)\right),$$

$$c_t[m] = \sum_{n=0}^{M-1} S_t(n) \cos\left(\pi m \left(\frac{n - 0,5}{M}\right)\right).$$

Donde $X_t(n)$ es la *Transformada Discreta de Fourier*, del input del frame del audio de habla t . Y $H_m(n)$ es la respuesta en la frecuencia del filtro m del banco de filtros híbridos. N vendría a ser el tamaño de la ventana de la transformada y M el número total de filtros. El resultado es un vector con menor cantidad de características, las cuales tienen menor correlación entre sí.

2.3. Modelos Ocultos de Markov

Siendo una técnica de reconocimiento de patrones [Singh et al., 2012], derivada de las cadenas y procesos estocásticos de Markov de primer orden. Permite modelar dependencias no-lineales de cada unidad de habla respecto a las adyacentes y posee un método eficiente para entrenar sus parámetros.

En [Singh et al., 2012] se define como una variante de una maquina de estado finita, la cual cuenta con:

- Un set Q de estados ocultos.
- Un set O de observaciones, para reconocimiento de habla corresponderán a la señal acústica.
- Un set A de probabilidades de transición.
- Un set B de probabilidades de salida o de emisión.
- Probabilidades iniciales de estado π .

Para cada tiempo t o *timestep* el modelo se encuentra en un estado oculto de Q , en el cual emite una observación de O según las probabilidades de salida B . Entonces para el siguiente *timestep* $t + 1$ se moverá según las probabilidades de transición A que siguen a su estado, con esta estructura base estos modelos permiten representar fenómenos probabilísticos [Gales et al., 2008, Yu y Deng, 2016].

2.3.1. HMM en *Speech Recognition*

De [Singh et al., 2012] los Modelos Ocultos de Markov (*HMM*) se pueden utilizar para modelar una unidad del *speech* o habla, tanto si es un fonema o palabra. Para lograr su efectividad en este problema, se asume lo siguiente:

- Observaciones adyacentes de *frames* en el habla, son independientes entre sí. Por lo cual la probabilidad de una secuencia de observaciones $P = (o_1, o_2, \dots, o_T)$, puede ser escrita como:

$$\prod_{i=1}^T P(O_i).$$

- Supuesto de Markov de orden 1, la probabilidad de estar en un estado en el tiempo t , depende solo del estado anterior en el tiempo $t - 1$.

Así se espera capturar la variabilidad respecto al contexto previo del input a la vez que puede crear relaciones entre las observaciones acústicas y los fonemas o palabras. Para un modelo HMM los parámetros que serán ajustados son $\lambda = (A, B, \pi)$, y de esta manera maximizar la probabilidad de la secuencia de observación $P(O|\lambda)$, esto de aquí sería un símil al **modelo acústico visto previamente**.

Un diagrama respectivo a un sistema de reconocimiento de habla basado en HMM se ve en la Figura 2.2. Cada palabra w puede ser descompuesta en fonemas, los cuales actúan como sonidos básicos del idioma que son utilizados en el diccionario de pronunciación para componer cada palabra. Así, para una vector acústico Y tenemos entonces que $P(Y|w)$ puede ser computado como:

$$P(Y|w) = \sum_Q P(Y|Q)P(Q|w).$$

Notamos entonces que $P(Q|w)$ correspondería al modelo de pronunciación, en la figura Figura 2.4 se puede ver un HMM para un fonema. Las palabras vendrían a ser representadas con concatenaciones de estos modelos, notar entonces la dependencia existente en los diccionarios de pronunciación que son creados a mano por lingüistas, es decir funcionan como parámetros ajustados por conocimiento experto. De la misma figura se puede ver que los estados ocultos Q definidos previamente corresponden a cada fonema, al decodificar el modelo se procede a generar una secuencia de estos. Por medio del modelo de pronunciación los fonemas Q son mapeados a palabras, las que luego serán corregidas por el modelo de lenguaje.

En resumen los sistemas ASR basados en HMM son una combinación de modelos acústicos, de lenguaje y pronunciación para los cuales un *decoder* o decodificador por medio de programación dinámica (Viterbi [Gales et al., 2008]) generan la transcripción con mayor probabilidad.

2.3.2. Modelos de Mezclas Gaussianas

Antes de que se utilizara redes neuronales en la sección de entrada de los modelos híbridos *DNN-HMM*, el modelado de la señal acústica se realizaba por medio de mezclas Gaussianas o *GMM*. Formalmente si deseamos modelar la distribución de una variable aleatoria x , por medio de esta técnica:

$$p(x) = \sum_{m=1}^M \frac{c_m}{(2\pi^{1/2})\sigma_m} \exp \left[-\frac{1}{2} \left(\frac{x - \mu_m}{\sigma_m} \right)^2 \right].$$

Notamos entonces que los parámetros a estimar son c_m , σ_m y μ_m que corresponden al peso, la desviación estándar y media de la distribución m . Este método no es bueno para modelar secuencias temporales, por ello para *speech recognition* los Modelos Ocultos de Markov se

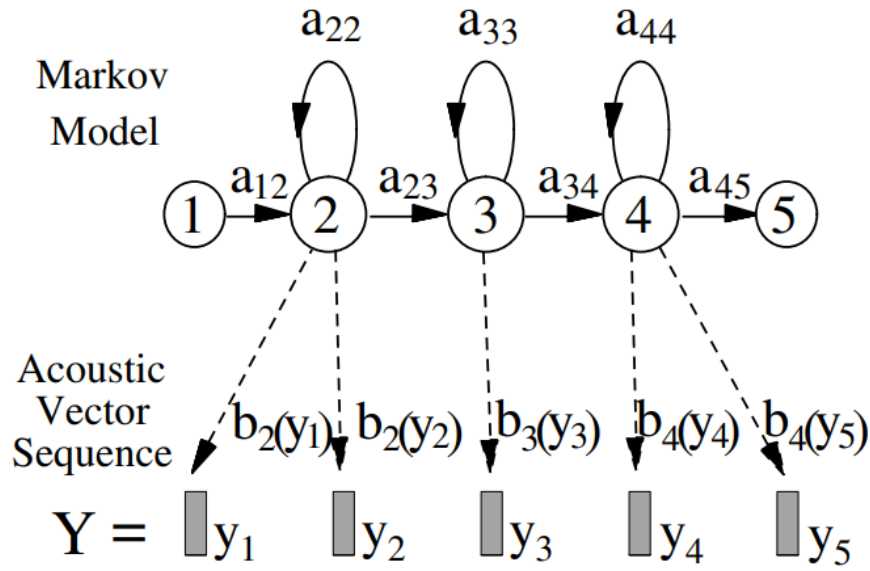


Figura 2.4: : Ejemplo de estados internos junto a sus observaciones para un HMM basado en fonemas. Aquí el vector Y es la señal acústica y a_{ij} son las probabilidades de transición del estado i a j . Finalmente $b_i(y_t)$ corresponde a la probabilidad de emisión del estado i respecto al *frame* y_t .

Fuente: Obtenido de [Gales et al., 2008].

ocupan de manejar las relaciones a través del tiempo de la entrada [Yu y Deng, 2016]. Como se mencionó previamente un estado interno de un modelo GMM-HMM se suele asociar con un fonema y las distribuciones Gaussianas son las encargadas de definir las probabilidades de emisión B :

$$b_j(x) = p(x|Q = q_j) = \sum_{m=1}^M c_{jm} \mathcal{N}(x; \mu_{jm}, \Sigma_{jm}).$$

Donde $b_j \in B$ es una probabilidad de emisión del estado j , μ_{jm} la media y Σ_{jm} la matriz de covarianza respectiva para la distribución m .

2.4. Redes neuronales

Las redes neuronales es una técnica del área de *Machine Learning*, estas nacen a partir de intentar imitar el procesamiento cognitivo del cerebro humano. Por medio de un aprendizaje a través de unidades conectadas y ordenadas en capas, se pueden dividir en 3 partes:

1. **Capa de entrada o Input Layer:** La entrada para los datos con los cuales se aprenderá y clasificará en el modelo.

2. **Capa escondida o *Hidden Layer*:** Donde se realiza la abstracción y paso de información hasta la capa de salida. Cada capa recibe un set de *weighted inputs* W o entradas con un peso asociado y produce una salida por medio de una función de activación. De esta manera, el output irá asociado a su vez a otro(s) peso(s) para llegar a la siguiente capa de abstracción.
3. **Capa de salida o *Output Layer*:** La última capa que indica la clasificación realizada por la red neuronal o bien el resultado obtenido después de procesar la señal de entrada en todo el modelo.

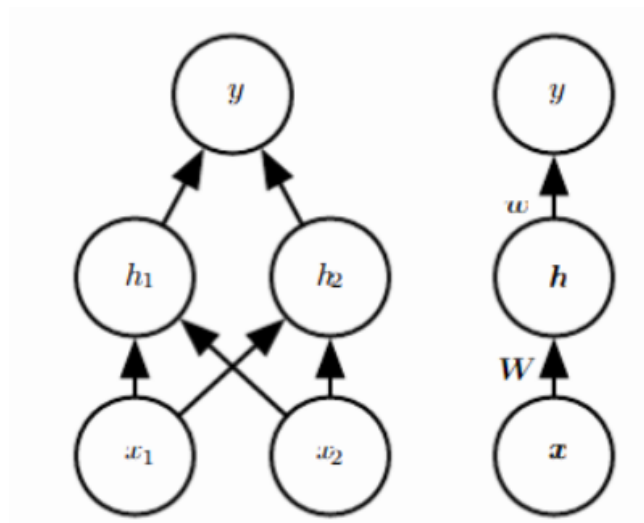


Figura 2.5: Ejemplo de una red *feedforward* simple.

Fuente: Obtenido de [[Goodfellow et al., 2016](#)].

Una simple ilustración puede ser vista en la Figura 2.5, esta red nos puede servir para realizar clasificación para vectores del tipo $X = (x_1, x_2)$, asociados a un *target* y . En las redes neuronales las salidas de un estado interno o neurona se computan por medio de una función de activación σ la cual es una función no-lineal:

$$\alpha = \sigma(WX - b).$$

Donde W es la matriz de los pesos respectivos multiplicados a cada x_i entrante a la unidad escondida y b corresponde al *bias*, aquí puede verse una similitud con una regresión lineal dado que estos serán los parámetros a ser estimados por el entrenamiento del modelo. Tomar en cuenta que esto corresponde a una unidad de la capa escondida, pudiendo ser miles en paralelo, las funciones de activación más conocidas son:

- Función ReLU:

$$ReLU(z) = \max(0, z).$$

- Función sigmoide:

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

- Función tangente hiperbólica o *tanh*, su uso es común en capas recurrentes:

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}.$$

2.4.1. Redes neuronales recurrentes

Las capas neuronales comunes con estados internos h como los de la Figura 2.5 al tener una arquitectura con un cantidad de celdas de entradas fijas, no son útiles para *inputs* de largo variable. Por otra parte este tipo de redes no tienen buen desempeño en el modelado de fenómenos temporales como series de tiempo [Kamath y Whitaker, 2019], dado que no tienen forma de generar inferencias respecto al contexto en una secuencia. A partir de este problema se estructuran las **redes neuronales recurrentes** que serán de interés para este trabajo, especializadas en modelar problemas de secuencias. Con entradas del tipo $X = x^{(1)}, \dots, x^{(t)}$, donde para este caso de estudio llamaremos a cada t un *timestep* y representa la posición del elemento x_i en la secuencia, este además puede ser un vector de características como por ejemplo las potencias en cada frecuencia de la señal de habla en un t específico. Matemáticamente un estado interno en la red recurrente vendría a ser representado como:

$$h^{(t)} = \sigma(h^{(t-1)}, x^{(t)}).$$

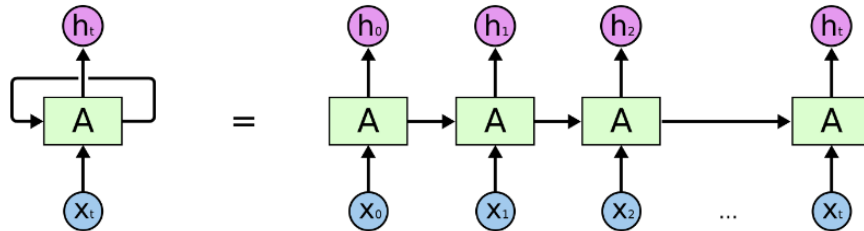


Figura 2.6: : Capa recurrente desenrollada.

Fuente: Obtenido de

<https://colah.github.io/posts/2015-08-Understanding-LSTMs/>.

Notamos entonces como el estado interno en el *timestep* t , depende del anterior y por lo tanto sería una recurrencia, un diagrama de esto se puede analizar en la Figura 2.6. Luego la ecuación anterior puede ser vista en términos de los parámetros a ajustar:

$$h_t = H(W_{xh}X_t + W_{hh}h_{t-1} + b_h),$$

$$y_t = W_{hy}h_t + b_y.$$

Donde nuevamente W son los pesos asociados a las conexiones en la red, en este caso abusamos de la notación designando a W_{xh} como los pesos asociados a la entrada a la capa oculta y W_{hh} los de la capa oculta respecto a si mismo en un *timestep* previo, recordando que se esta viendo una recurrencia. Notamos entonces como se mantiene la información contextual por medio de estos estados escondidos en el tiempo.

2.4.2. Capas LSTM

En [Hochreiter y Schmidhuber, 1997] los autores definen un nuevo tipo de arquitectura para redes recurrentes llamada *Long Short Term Memory Network*, esta tiene por idea permitir mantener conocimiento por varios *timesteps*. Esto lo logran resolviendo el problema del desvanecimiento del gradiente descendente a través del tiempo, un problema inherente del *back-propagation through time*. Aquí se ocupan bloques inteligentes que deciden cuando mantener u olvidar información útil para el aprendizaje, por medios de las *gates*. Se define entonces de la siguiente manera el paso de un *input* a través de una unidad *LSTM*:

$$\begin{aligned}i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i), \\f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f), \\c_t &= f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c), \\o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o), \\h_t &= o_t \tanh(c_t).\end{aligned}$$

Donde nuevamente σ representa la función de activación sigmoide, por otra parte i , f , c y o son respectivamente las *gates* o puertas de *input*, *forget*, *cell* y *output*. Luego cada una de estas puertas se encarga de:

- **Input Gate:** Controla la información nueva que sera añadida a la neurona. Aquí esta información es elegida por medio de la función de activación $\sigma(\cdot)$ dado que llevara ciertos valores a 0, los cuales serán multiplicados junto a los mismos valores después de pasar por la función de activación $\tanh(\cdot)$.
- **Forget Gate:** Mueve los parámetros asociados al *loop* interno, por lo cual decide la información que se mantiene o no en los siguientes *timesteps*.
- **Cell State:** El estado anterior de la celda es multiplicado por el vector *forget* calculado previamente, y el resultado se suma a la multiplicación resultante después de la *input gate*. Así el estado anterior sufre 2 transformaciones seguidas, y el resultado continuara al siguiente *timestep*.
- **Output Gate:** Aquí se calcula el siguiente h_t o estado escondido de la neurona, el cual se calcula por medio del h_{t-1} y el estado actual de la celda. Esta salida entonces influirá directamente en la predicción.

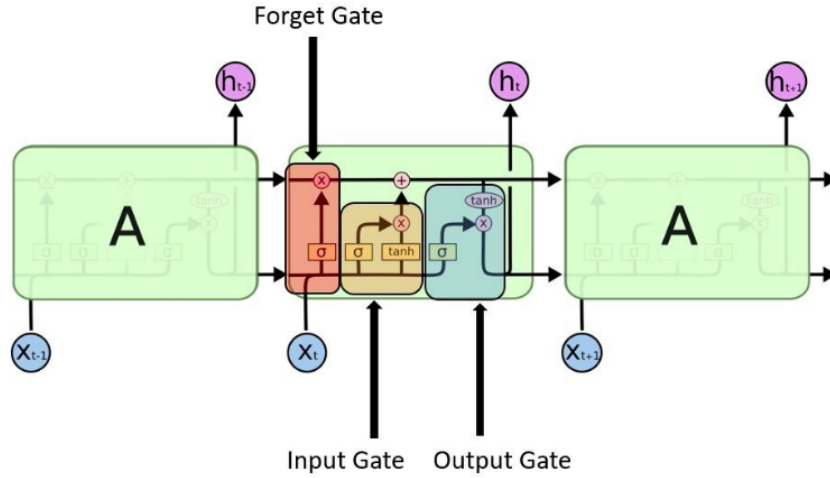


Figura 2.7: : Diagrama de unidades LSTM.

Fuente: Obtenido de <https://medium.com/@purnasaigudikandula/recurrent-neural-networks-and-lstm-explained-7f51c7f6bbb9>.

Una visualización de una unidad LSTM puede ser vista en la figura 2.7, por otra parte en este trabajo utilizaremos una versión que no solo utiliza información respectiva al contexto pasado si no también el futuro. Esto dado que es natural en el habla que las relaciones entre fonemas no provengan solo del anterior [Graves et al., 2006], entonces se desea también obtener información del contexto futuro. Para lograr esto en vez de computarse un solo estado escondido h_t , se utiliza:

$$(forward) \vec{h}_t = H(W_{x\vec{h}_t} x_t + W_{\vec{h}\vec{h}} \vec{h}_{t-1} + b_{\vec{h}}),$$

$$(backward) \overleftarrow{h}_t = H(W_{x\overleftarrow{h}_t} x_t + W_{\overleftarrow{h}\overleftarrow{h}} \overleftarrow{h}_{t-1} + b_{\overleftarrow{h}}),$$

$$y_t = W_{\vec{h}y} \vec{h}_t + W_{\overleftarrow{h}y} \overleftarrow{h}_t + b_y.$$

Así bien se computan los estados escondidos de 1 a T por medio de $\vec{h}_t$ y de T a 1 por medio de $\overleftarrow{h}_t$. Un diagrama de esta arquitectura puede ser visto en la figura 2.8, finalmente combinando esta estrategia de las redes recurrentes bidireccionales o *BRNNs* con las unidades *LSTM* se obtienen las capas que son de interés para este trabajo: *BiLSTM* o *bidirectional LSTM* [Graves y Schmidhuber, 2005].

2.4.3. Redes neuronales convolucionales

En los problemas que requieren extraer características de imágenes y por medio de estas crear un modelo de clasificación se necesita tener un acercamiento distinto. Aquí entran las **capas convolucionales** por medio de las cuales se filtran características de la entrada,

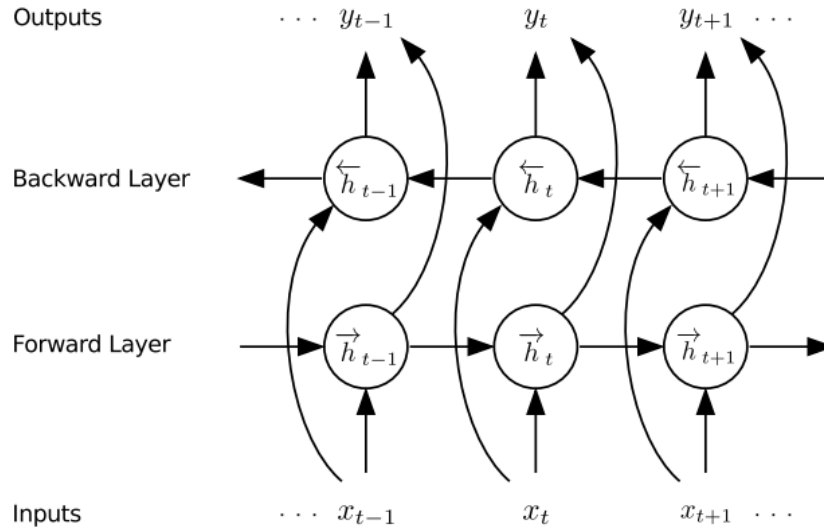


Figura 2.8: Red neuronal recurrente bidireccional.

Fuente: Obtenido de [Graves et al., 2013].

obteniendo características de mayor nivel según la complejidad del modelo.

Convolución: Por medio de un kernel/filtro se extraen productos punto de la imagen de la matriz de entrada de 3 dimensiones, una capa convolucional puede estar conformada por varios filtros los cuales en conjunto generan la salida de la misma dimensión. En la Figura 2.9 notamos que la entrada corresponde a la imagen de un dígito la cual posee 1 canal (escala de grises) por lo cual también es 3D.

Con mayor detalle en la Figura 2.10 se puede ver el funcionamiento del kernel convolucional en la entrada. Junto a las capas convolucionales suele ocuparse una capa de *pooling*, esta permite extraer información relevante respecto a cada filtro y además ayuda a evitar el sobre-ajuste del modelo, en otras palabras funciona como una extracción de características sobre la convolución. Se debe notar que para las capas convolucionales los parámetros a optimizar W corresponden a los valores de los filtros.

2.4.4. Optimización parámetros

Para poder ajustar los parámetros o pesos W de las capas neuronales, se hace uso de algoritmos de optimización guiados según funciones objetivo a minimizar. Por ejemplo para un problema de clasificación binaria tenemos una salida única de la red como $\hat{y} = \sigma(z)$, con valores entre 0 y 1 gracias a la función de activación *sigmoid* o sigmoide, este *output* vendría a ser la predicción. Para computar la pérdida o error respecto a esta salida y el valor real, podemos utilizar la función *Cross Entropy* binaria:

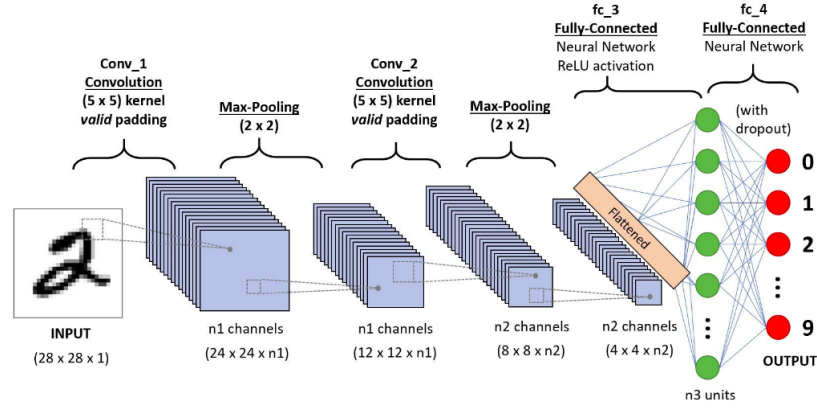


Figura 2.9: Red neuronal con capas convolucionales, para clasificar dígitos.

Fuente: Obtenido de <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>.

$$L = -\frac{1}{N} \sum_{i=1}^N [y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})].$$

Para el caso de múltiples categorías, tendremos una salida en la red para cada clase y el total de probabilidades sumará 1. Para lograr esto se utiliza la capa *softmax*, su cálculo según la salida viene dado como:

$$\hat{y}(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}.$$

Por medio de esta normalización se logra que las probabilidades de la última capa sumen 1, luego se generaliza la función *Cross Entropy* para K clases, obteniéndose el calculo de la pérdida:

$$L = -\frac{1}{m} \sum_{i=1}^m \sum_{k=1}^K [y_k^{(i)} \log(\hat{y}_k^{(i)}) + (1 - y_k^{(i)}) \log(1 - \hat{y}_k^{(i)})].$$

Por medio de esta función de error, la cual es diferenciable se pueden utilizar algoritmos como el gradiente descendente para el entrenamiento de la red y de esta manera ajustar los parámetros W de las capas $h(z)_i$ previamente explicados. Entonces la base de la optimización en las redes neuronales viene de la siguiente expresión:

$$\min_w L(w).$$

Donde la actualización de cada w en las capas viene dado por:

$$w^{(t+1)} = w^{(t)} - \eta \triangle_w L.$$

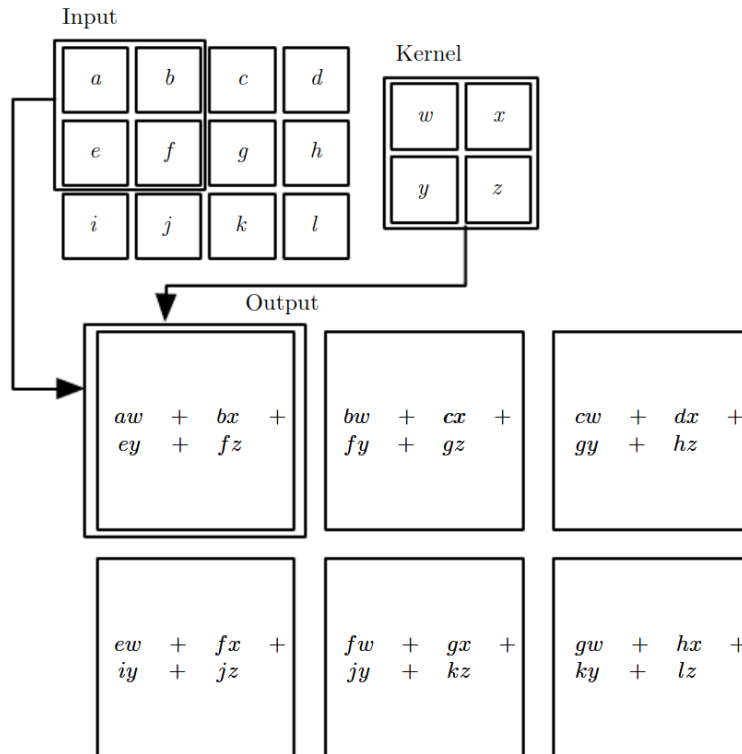


Figura 2.10: Ejemplo de convolución 2D con kernel 2x2.

Fuente: Obtenido de [Goodfellow et al., 2016].

Aquí η viene a representar la tasa de aprendizaje con la cual se actualizan los pesos. Así el entrenamiento se lleva a cabo primero calculando el error por medio de *forward-propagation*, es decir computando las entradas hasta el final de la red y obtener la predicción asociada. Luego utilizar *back-propagation*, calculando el error asociado a cada predicción y actualizando los pesos W como se mostró previamente, notar que para el caso de las redes recurrentes se ocupa una variación llamada *back-propagation through time* debido a que se procesan secuencias con dependencias temporales.

Existen distintos algoritmos para realizar una actualización de pesos más efectiva, dentro de los cuales se puede nombrar:

- **Stochastic gradient descent:** Usualmente llamado *SGD*, computa el promedio del gradiente descendente para un número fijo de muestras (*minibatch*). Aquí entra la importancia del η visto previamente dado que con el se regula la velocidad de convergencia de la optimización.
- **Adam:** Presentado en [Kingma y Ba, 2014] es un algoritmo mucho más avanzado respecto al anterior, dado que presenta ser adaptativo. Esto se refiere a que las tasas de aprendizaje η son adaptadas para cada parámetros [Kamath y Whitaker, 2019], su

definición más formal viene como:

$$\begin{aligned}m_t &= \beta_1 m_{t-1} + (1 - \beta_1) g_t, \\v_t &= \beta_2 v_{t-1} + (1 - \beta_2) g_t^2, \\ \hat{m}_t &= \frac{m_t}{1 - \beta_1^t}, \\ \hat{v}_t &= \frac{v_t}{1 - \beta_2^t}, \\ w^{t+1} &= w_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t.\end{aligned}$$

Donde m_t representa el promedio de los gradientes pasados y g_t el gradiente actual. Aquí entra el concepto del *momentum* el cual ayuda a aumentar la velocidad de convergencia por medio de un aumento de la velocidad v_t , *Adam* regulariza su aporte y el de la información acumulada de gradientes pasados m_t . Finalmente β_1 y β_2 funcionan simplemente como parámetros de ajuste respecto al aporte del nuevo gradiente en cada t .

Al momento de optimizar un modelo sobre un conjunto de datos llamado *set* de entrenamiento, se suele validar con otro conjunto correspondiente al de validación. Esto dado que nos interesa saber el grado de generalización para clasificar nuevos datos que logra nuestro modelo tras cada iteración, aquí es donde se puede presentar el *overfitting* o sobreajuste de los parámetros a los datos de entrenamiento. El caso contrario es conocido como *underfitting* y se refiere a cuando el modelo no posee la complejidad suficiente para modelar las relaciones intrínsecas de los datos con su *target*. Cuando se elige un modelo después de la validación, se debe proceder a probar con el conjunto de pruebas o *test* para tener una noción no sesgada respecto a la efectividad del clasificador en el caso más real posible.

2.4.5. Connectionist Temporal Classification

En [Graves et al., 2006] se presentó la función de pérdida *Connectionist Temporal Classification* o CTC, la cual permite entrenar redes neuronales para aprender un mapeo entre una secuencia de entrada y las clases de salida que le corresponden, optimizando la predicción del tipo secuencia-a-secuencia. De esta manera se pueden crear modelos de *Speech Recognition* utilizando *deep learning*, dado que los audios son de largo variable y se puede asociar a ellos una secuencia de caracteres, fonemas o palabras. El mayor aporte entonces, esta en que permite a una red neuronal aprender directamente este mapeo sin la necesidad de utilizar un *HMM*.

Para computar esta función de pérdida del modelo en una red neuronal, primero se hace uso de una capa de salida *softmax* la cual por ejemplo puede representar en cada clase (nodo de salida) un carácter o fonema y se agrega una clase más llamada *blank* que viene a corresponder al espacio “sin categoría”, es decir un *timestep* nulo en el *output*. De esta

manera se quiere definir una distribución de probabilidad $P(k|t)$ para cada *timestep*, donde k serían las clases de la salida. CTC ocupa un algoritmo *forward-backward* para calcular la suma respecto a todos los posibles alineamientos y determinar la probabilidad de la secuencia final dado el *input*. Dado que tenemos una secuencia de texto o transcripción l la cual deseamos obtener por medio de nuestra señal de *input* x , se define:

$$p(l|x) = \sum_{\pi \in \beta^{-1}(l)} p(\pi|x).$$

Donde π corresponde a un *path* o camino posible de asignación de clases a los *timestep*, un ejemplo se puede ver en la Figura 2.11 donde cada línea que conecta los nodos hacia la derecha es un camino posible. También se define la función $\beta(\cdot)$ la cual asocia *paths* equivalentes con una transcripción l . Esto es posible ya que $\beta(\cdot)$ colapsa los *labels* asignados en el camino gracias a la categoría “de más” previamente definida, *blank*. Tomando como ejemplo la misma Figura 2.11, mostramos el uso de β :

$$\beta(a-) = \beta(aa) = \beta(-a) = a.$$

Se puede ver que la clase nula — se omite en el mapeo, si bien estos caminos o *paths* llegan a la misma transcripción es posible que tengan distintas probabilidades. Entonces definimos el mapeo inverso β^{-1} , con esto la función de coste a minimizar queda como:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} - \sum_{i=1}^N \log \left[\sum_{\pi \in \beta^{-1}(z^{(i)})} p(\pi|x^{(i)}; \theta) \right].$$

De esta manera se computa la pérdida como el logaritmo negativo de la suma de las probabilidades de las secuencias [Kamath y Whitaker, 2019]. Donde la función objetivo busca los parámetros que maximizan la probabilidad de las clasificaciones correctas, lo cual es una optimización por máxima verosimilitud. Esta función es diferenciable, por lo cual podemos ocupar los algoritmos de optimización típicos de redes neuronales.

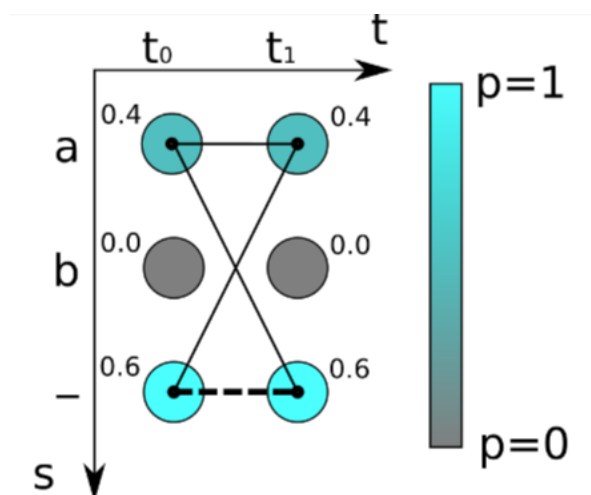


Figura 2.11: Matriz simple de caminos en CTC.

Fuente: Obtenido del siguiente <https://towardsdatascience.com/intuitively-understanding-connectionist-temporal-classification-3797e43a86c>.

CAPÍTULO 3

PROPUESTA DE SOLUCIÓN

La propuesta de solución para el problema de reconocimiento de habla en español de este trabajo, estará basada en el uso de arquitecturas de redes neuronales profundas. Es por ello que se utilizarán la mayoría de los corpus de audios en español que se encontraron, los cuales al presentar un español de distintos países dan una connotación de multi-pronunciación a la solución. A continuación se sentarán las bases respecto a la metodología a utilizar, como también la presentación de los datos que se entregaran al modelo para su entrenamiento.

3.1. Metodología

Para ordenar el trabajo y tener un plan de desarrollo, se realizará un flujo de trabajo común en proyectos de *Machine Learning* pero realizando modificaciones necesarias para nuestro caso en particular:

1. **Pre-procesamiento:** Corresponde al análisis y modificaciones previos a la generación de los set de *train/val/test*. Esta etapa es de mucha importancia dado que son los filtros de los datos que son usados por parte del modelo, por lo cual se listara el *pipeline* utilizado para limpiar los datos provenientes de las distinta fuentes. Se visualizarán las distribuciones y tendencias que sean de importancia para este caso de estudio, tales como la duración, acentos y género.
2. **Extracción de características:** Partición del *dataset* y transformaciones necesarias en los datos para usar como entrada al modelo, lo cual dependerá de la arquitectura base. Además se verán técnicas de utilidad en el manejo de los datos para la realización del proceso de ajuste. Se debe notar que existen técnicas de extracción de características comunes en la literatura para el reconocimiento de habla, las cuales dependen de la cantidad de datos disponibles y de la complejidad del modelo.
3. **Modelado:** La elección de la arquitectura base a utilizar en la cual se realiza el entrenamiento. Para el trabajo presente corresponderá a una red neuronal, la cual será elegida según lo revisado en la literatura entregando un detalle más específico de su implementación.
4. **Entrenamiento/Variación del modelo:** Utilización de la arquitectura elegida con los datos de entrenamientos y conjunto de validación correspondientes. Se analizará el desempeño para distintos casos, variando cuidadosamente los hiper-parámetros dado lo costoso que presenta ser en tiempo el ajustar un modelo de *deep learning*. Para cada modificación de hiper-parámetros se entrenarán 3 modelos con semillas distintas para obtener resultados más robustos.

5. **Evaluación:** Después del proceso de elección del mejor modelo con el set de validación, se procederá a realizar una prueba de la arquitectura entrenada con los set de test o pruebas elegidos y de esta manera verificar la generalización del modelo final. Notar que este es el paso final para la revisión de la calidad del modelo, para la cual se utilizaran 2 métricas:

- a) **WER:** Tasa de error por palabra o *word error rate*, se define como el porcentaje de error en la transcripción realizada por el modelo en base a la sentencia original [Anusuya y Katti, 2010], su origen viene de la distancia de Levenshtein. Matemáticamente se define como:

$$WER = \frac{S + D + I}{N} * 100.$$

- S es el número de substituciones.
- D es el número de eliminaciones.
- I es el número de inserciones.
- N es el número de palabras de referencia.

Por ejemplo si la frase *target* es "hola como estas", y el modelo entrega la transcripción "hola esas"(notar el uso de texto normalizado), el calculo viene dado como:

$$WER = \frac{1 + 1}{3} * 100 = 66,6\%.$$

- b) **CER:** También se utilizará la tasa de error por carácter, dado que el trabajo vendrá a estar centrado en los modelos acústicos. Los cuales optimizan directamente sobre los *label* o salidas del modelo, que para el problema presente corresponden a los caracteres. De esta manera se define la tasa de error por carácter o *character error rate* como:

$$CER = \frac{S + D + I}{N} * 100.$$

- S es el número de substituciones.
- D es el número de eliminaciones.
- I es el número de inserciones.
- N es el número de caracteres presentes en la oración original.

Notar que estas métricas serán comparadas respecto a un modelo *baseline*, el cual en nuestro caso corresponde a uno obtenido de <https://sourceforge.net/projects/cmusphinx/files/Acoustic%20and%20Language%20Models/Mexican%20Spanish/>. La estructura de este modelo es del tipo híbrido utilizando mezclas Gaussianas junto a un modelo oculto de Markov, por otra parte según los autores fue entrenado en 581 horas de audios.

Provenientes tanto de fuentes gratuitas como pagadas y principalmente para el acento mexicano pero también con un porcentaje de pronunciación española y de distintos lugares latinoamericanos. Por todo esto, presenta ser un buen modelo base para la comparación respecto al que será generado en este caso de estudio, dado que también es multi-pronunciación y permitirá realizar un mejor análisis respecto a los resultados.

3.2. Fuentes de datos

A continuación se procede a listar las distintas fuentes de datos que fueron recolectadas, con sus detalles específicos y duración respectiva. Notar que se listaran la cantidad/duración de los subconjuntos a utilizar de estos set de datos, es decir después del pre-procesamiento realizado.

3.2.1. Ciempiess

El nombre proviene de un acrónimo para “Corpus de Investigación en Español de México del Posgrado de Ingeniería Eléctrica y Servicio Social”, creado en la facultad de ingeniería de la Universidad Nacional Autónoma de México [Hernández-Mena y Herrera-Camacho, 2014]. Su dominio es principalmente un acento mexicano céntrico.

Posee una duración total de 17[hrs], con solo 3 audios más largos que 20 segundos por lo cual se decide descartarlos. Este posee un des-balance de géneros con 77.8 % de voces masculinas contra el 22.1 % de las femeninas. Las características respecto al tamaño del corpus pueden verse en la Tabla 3.2.

Tabla 3.2: Características generales corpus Ciempiess. Fuente: Elaboración Propia.

Set	Duración [hrs]	Cantidad
Train	17.1	16.014
Test	0.5	1.000

Los archivos .wav tienen una frecuencia de muestreo (*samplerate*) de 16.000 [Hz] y tasa de bits (*bitrate*) de 16[bps]. El set de pruebas o *test* incluido en el corpus fue previamente extraído por los autores y se utilizará como uno de los set de pruebas.

3.2.2. Common Voice

Es un corpus libre formado mediante *crowd-sourcing* en la pagina de mozilla <https://voice.mozilla.org/>, es decir cualquier usuario interesado puede ingresar y grabarse leyendo una oración y/o validar las existentes. Esto permite tener un corpus gratis, de distintas fuentes y con distintas características que lo hacen interesante para el trabajo actual en el dominio del español. Por otro lado cuenta con un set de validación y otro de pruebas ya creados previamente, que evitan el solapamiento de hablantes. Otro punto a favor es que es multi-acento, por lo cual validar y probar en estos datos entrega una visión más real de la generalización del modelo. Notar que por lo mismo se vuelve un

set de datos en el cual tener buenos resultados es una tarea difícil, sin nombrar que son audios que suelen tener ruido intrínseco dada la calidad de microfonos y/o ruido presente al realizar las grabaciones. Según la misma pagina de Mozilla pose un des-balance de géneros con un 74 % audios de hombres y 9 % de mujeres (resto faltante no registrado), por otra parte posee gran cantidad de audios con acento de España. Las características respecto al tamaño del corpus pueden verse en la Tabla 3.3.

Tabla 3.3: Características generales corpus Common Voice. Fuente: Elaboración Propia.

Set	Duración [hrs]	Cantidad
Train	20.8	18.308
Val	3.14	2.718
Test	3.5	3.53

Dada que los archivos de audios son creados directamente desde la grabación de los usuarios, las muestras poseen distintos *samplerate* y *bitrate*. Pero en general son 48.000[hz] y 16[bps] respectivamente.

3.2.3. Crowdsourced high-quality Spanish audios

A continuación se lista una serie de conjuntos de audios junto a sus transcripciones obtenidas a través de voluntarios, para distintos países de Latinoamérica. Existen oraciones que se repiten entre los hablantes de distintos países, sin embargo al ser varios los acentos de igual manera generan un aporte como parte de los datos de entrenamiento. Los datos fueron obtenidos de <https://www.openslr.org/index.html>.

■ *Argentinian speech data set*

Consiste de oraciones con el acento argentino, especialmente el *rioplatense*. Además de audios con pronunciaciones de oraciones respectivas al pronóstico del tiempo. Se debe notar que posee más audios femeninos, con un desbalanceo del 32/68 % en audios de hombres/mujeres. Las características respecto al tamaño del corpus pueden verse en la Tabla 3.4.

Tabla 3.4: Duración y cantidad de audios para *Argentinian speech dataset*. Fuente: Elaboración Propia.

Set	Duración [hrs]	Cantidad
Train	6.3	4.541

■ *Chilean speech data set*

Consiste de oraciones con el acento chileno. Posee una relación del 60 % de audios masculinos contra el 40 % de audios femeninos. Las características respecto al tamaño del corpus pueden verse en la Tabla 3.5.

Tabla 3.5: Características generales corpus *Chilean speech dataset*. Fuente: Elaboración Propia.

Set	Duración [hrs]	Cantidad
Train	7.1	4374

■ ***Colombian speech data set***

Consiste de oraciones con el acento colombiano. Posee una relación del 52 % de audios masculinos contra el 48 % de audios femeninos. Las características respecto al tamaño del corpus pueden verse en la Tabla 3.6.

Tabla 3.6: Características generales corpus *Colombian speech data set*. Fuente: Elaboración Propia.

Set	Duración [hrs]	Cantidad
Train	7.5	4.903

■ ***Peruvian speech data set***

Consiste de oraciones con el acento peruano. Posee una relación del 54 % de audios masculinos contra el 46 % de audios femeninos. Las características respecto al tamaño del corpus pueden verse en la Tabla 3.7.

Tabla 3.7: Características generales corpus *Peruvian speech data set*. Fuente: Elaboración Propia.

Set	Duración [hrs]	Cantidad
Train	9.2	5.447

■ ***Puerto Rico speech data set***

Consiste de oraciones con el acento puertorriqueño. Se debe notar que para este corpus solo se poseen audios de población femenina. Las características respecto al tamaño del corpus pueden verse en la Tabla 3.8.

■ ***Venezuelan speech data set***

Consiste de oraciones con el acento venezolano. Posee una relación del 52 % de audios masculinos contra el 48 % de audios femeninos. Las características respecto al tamaño del corpus pueden verse en la Tabla 3.9.

Tabla 3.8: Características generales corpus *Puerto Rico speech data set*. Fuente: Elaboración Propia.

Set	Duración [hrs]	Cantidad
Train	1.0	617

Tabla 3.9: Características generales corpus *Venezuelan speech data set*. Fuente: Elaboración Propia.

Set	Duración [hrs]	Cantidad
Train	4.8	3.557

En general los audios poseen una tasa de muestreo de 48.000[hz] y tasa de bits del 16[bps]. Y no se presentan des-balanceos importantes.

3.2.4. Dimex-100

Creado en el departamento de Ciencias de la Computación del Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas (IIMAS), UNAM [Castellanos y Gamboa,]. El nombre viene del proyecto DIME, también esta centrado en la pronunciación mexicana del español y posee distintos niveles de segmentación respecto a los audios con las transcripciones. Para el caso de este trabajo se decide por usar el nivel de oración. Posee una relación balanceada con 51 % de audios masculinos contra el 49 % de audios femeninos. Las características respecto al tamaño del corpus pueden verse en la Tabla 3.10.

Tabla 3.10: Características generales corpus Dimex-100. Fuente: Elaboración Propia.

Set	Duración [hrs]	Cantidad
Train	5.8	5.667

Los archivos wav presentan tener un *samplerate* o frecuencia de muestreo de 44.100 [Hz] y tasa de bits de 16[bps].

3.2.5. M-AILABS Speech Dataset

Este corpus de gran tamaño corresponde a lecturas de audiolibros, provenientes de [LibriVox](#) y [Project Gutenberg](#). Corresponde principalmente a hablantes de pronunciación española en conjunto con una menor proporción de hablantes de Latinoamérica. Obtenido de <https://www.caito.de/2019/01/the-m-ailabs-speech-dataset/>, posee una relación del 64 % de audios masculinos contra el 12 % de audios femeninos, el porcentaje faltante no esta categorizado. Las características respecto al tamaño del corpus pueden verse en la Tabla 3.11.

Los archivos wav presentan tener un *samplerate* o frecuencia de muestreo de 16.000 [Hz] y tasa de bits de 16[bps].

Tabla 3.11: Características generales corpus M-AILABS. Fuente: Elaboración Propia.

Set	Duración [hrs]	Cantidad
Train	108.4	59.667

3.2.6. TEDx Spanish Corpus

Corpus proveniente de charlas TEDx de hablantes mexicanos, el proceso de creación fue en parte gracias al programa de servicio social “Desarrollo de Tecnologías del Habla” de la Universidad Nacional Autónoma de México. Posee una relación del 74% de audios masculinos contra el 26% de audios femeninos. Los datos fueron obtenidos de <https://www.openslr.org/index.html>. Las características respecto al tamaño del corpus pueden verse en la Tabla 3.12.

Tabla 3.12: Características generales corpus TEDx. Fuente: Elaboración Propia.

Set	Duración [hrs]	Cantidad
Train	24.5	11.243

3.2.7. Voxforge Spanish Corpus

Obtenido de <http://www.voxforge.org/home>, es similar al corpus de *Common Voice* corresponde a un repositorio de audios con sus transcripciones y modelos de reconocimiento de habla, de aquí se obtiene el *baseline*. El corpus fue formado principalmente con aportes de distintos hablantes a través de un *software open source* y subidos a la página. También es de multi-pronunciación y posee entradas con distintas condiciones entre sí, las características respecto al tamaño del corpus pueden verse en la Tabla 3.13.

Tabla 3.13: Características generales corpus *Voxforge Spanish*. Fuente: Elaboración Propia.

Set	Duración [hrs]	Cantidad
Train	50	22.674

3.2.8. Final

Recolectando todos estos corpus, y juntándolos en un solo conjunto de datos final corresponde al total de datos que serán usados para el desarrollo de la memoria. Notar que debido a que corresponde a un conjunto de datos recolectado de distintas fuentes, posee condiciones distintas entre los audios. Tales como:

- Medio de entrada.
- Calidad del audio.

- Pronunciación.
- Tasa de muestreo o *samplerate*.
- Volumen.

La duración total del set de datos (sin los conjuntos de validación y pruebas) como también cantidad de audios se muestra en la Tabla 3.14, además la distribución respectiva a estos se puede ver en la Figura 3.12. El balance de géneros puede ser visualizado en la Figura 3.13, aquí se puede notar la acumulación de las diferencias entre la cantidad de hablantes masculinos respecto a los femeninos. Por otra parte, como se menciono previamente el corpus de entrenamiento es de tipo multi-acento, sin embargo en la Figura 3.14 podemos evidenciar que existe un sesgo hacia acentos de España y México.

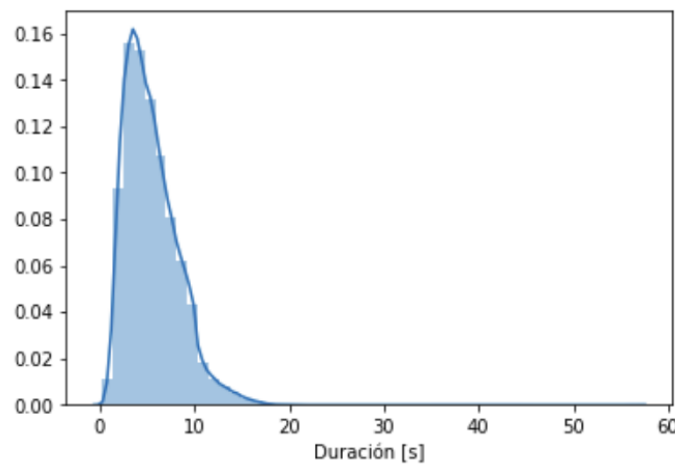


Figura 3.12: Distribución de la duración de los audios para el conjunto final.

Fuente: Elaboración Propia.

Tabla 3.14: Características generales corpus Final. Fuente: Elaboración Propia.

Set	Duración [hrs]	Cantidad
Train	263	156.373

El conjunto que será utilizado para la **validación** en la variación de hiperparámetros corresponderá al pre-seleccionado en Mozilla Common Voice de la Tabla 3.3. Esto dado que es de tipo multi-acentos, presenta audios con ruido y finalmente se modificó con tal de que no tuviera un solapamiento de hablantes respecto al de entrenamiento, lo cual lo hace ideal para analizar la generalización de los modelos a ajustar. Como conjunto de **pruebas** para la comparación del modelo final contra el *baseline*, se utilizará un set de pruebas proveniente de Mozilla Common Voice y también el set de datos de pruebas de *Ciempies* para una comparación más diversa. Notar que existe un desbalance de géneros similar al set de entrenamiento, por otro lado para el caso de los conjuntos provenientes de Mozilla Common Voice el balance de acentos es más equilibrado.

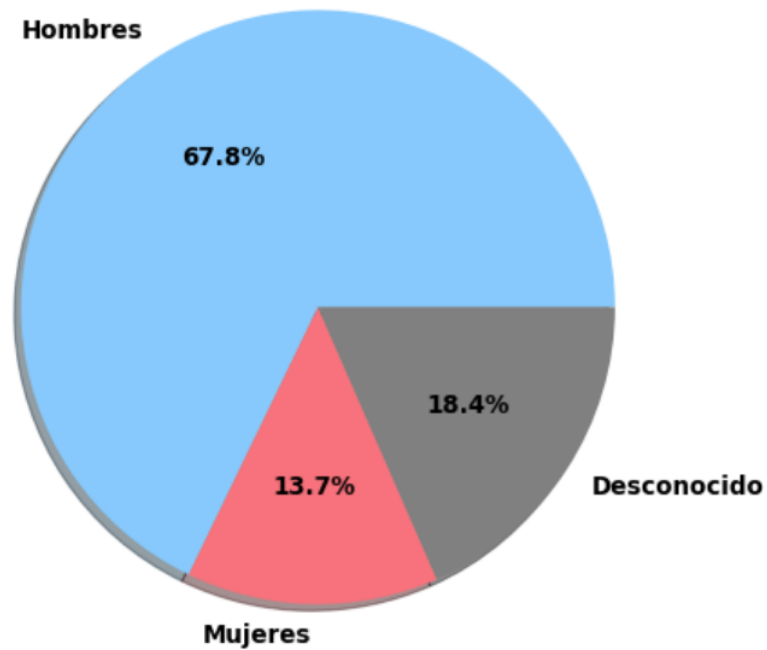


Figura 3.13: Balance de géneros en set de entrenamiento final.
Fuente: Elaboración propia.

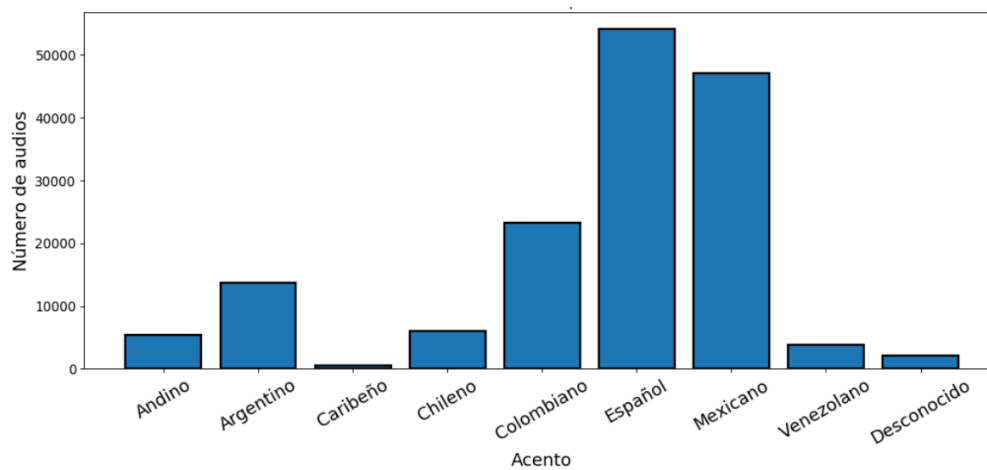


Figura 3.14: Distribución de acentos en la agrupación final de los corpus.
Fuente: Elaboración propia.

CAPÍTULO 4

CONSTRUCCIÓN E IMPLEMENTACIÓN DE LA SOLUCIÓN

Se procede a mostrar el desarrollo de la solución utilizando la estructura previamente explicada. Siguiendo los pasos para formar el conjunto de datos desde las distintas fuentes, se discutirán las decisiones a tomar respecto a la arquitectura, además de mostrar el progreso y resultados respectivo en cada paso de la modificación de los modelos.

4.1. Pre-Procesamiento

4.1.1. Modificación corpus

Los datos totales que se pudieron juntar corresponden alrededor de 263[hrs], la calidad de los audios depende de cada fuente y existen casos en los cuales hay transcripciones o *labels* erróneos. El formato estándar por el cual se procesan los audios es el de archivos WAV o *Waveform Audio Format*. El pre-procesamiento para cada corpus por separado se dividió en los siguientes pasos:

- **Agrupación de audios:** Para cada corpus se ordenó los audios en una carpeta según las transcripciones, permitiendo filtrar audios sin su correspondiente *target* y eliminando aquellos con transcripciones vacías. Además se realiza el *re-sampling*, cambiando la tasa de muestreo a 16.000[khz] para el caso de audios con un mayor *samplerate* de manera que todas las entradas tengan las mismas condiciones.
- **Limpieza de transcripciones:** Se revisan los caracteres especiales incluidos en las transcripciones, y de notaciones que puedan ser problemáticas para la limpieza posterior. Por ejemplo para el *set* de *Ciempies* la “ñ” es reemplazada por una “N” según lo explicado en [\[Hernández-Mena y Herrera-Camacho, 2014\]](#).
- **Revisión pronunciaciones problemáticas:** Se realizó una revisión de general de pronunciaciones para cada hablante que puedan marcar una diferencias respecto a la relación audio/transcripción. Por ejemplo en zonas de Argentina existe el fenómeno del “yeísmo rehilado”, en el cual el fonema usual para “ll” o “y” vendría a ser reemplazado por un “sh” al momento de la pronunciación. Este paso fue distinto según el corpus que se analizó, pero en general el mayor cuidado que se tuvo fue respecto al reemplazo de fonemas muy específicos.
- **Revisión de calidad de audios:** También al revisar cada hablante se tuvo cuidado con descartar aquellos que debido a un mal micrófono y/o ruido externo poseían grabaciones demasiado ruidosas. Donde más se pudo notar este problema fue en *Common Voice*, esto ya que los usuarios directamente realizaban la grabación desde su computador.

Por medio de estos pasos se limpió en primera fase los conjuntos de audios y transcripciones para cada corpus, posteriormente se agruparon todos para formar el set final. Las transcripciones pasaron por una segunda iteración de limpieza, en la que se descartó acentos y mayúsculas además de realizar la conversión de dígitos a palabras. Finalmente cada transcripción paso a codificarse numéricamente según un mapeo de carácter a número, esto con el fin de poder realizar el entrenamiento posterior asociado cada grafema a una clase. Como ilustración de la codificación o *encoding* se adjunta la Figura 4.15.

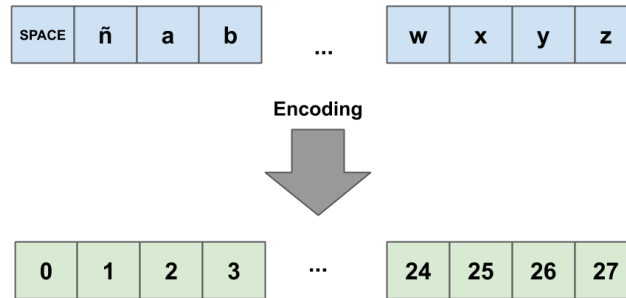


Figura 4.15: Diagrama de ejemplo respecto de codificación letra a número.
Fuente: Elaboración Propia.

4.1.2. Distribuciones y partición corpus

Ahora bien con el corpus ya agrupado se procede a analizar las distribuciones respectivas que posee tanto en duración, clases, géneros y acentos. Dado que el reconocimiento de habla presenta ser un problema de mapeo de secuencias, se debe tener cuidado con la duración de los audios dado que este factor puede afectar la precisión del modelo en caso de ser muy prolongados. Por otro lado, el coste en memoria de utilizar CTC puede aumentar en demasía con caminos o *paths* muy largos (numerosos *timesteps*).

Podemos notar en la Figura 3.12 que existe una concentración de muestras con una duración entre los 3[s] y 9[s]. Sin embargo, después de los 9[s] la proporción de audios disminuye demasiado, algunos presentando errores en su transcripción al ser muchos caracteres (casos erróneos). Se decide dejar solo 200 [hrs], la distribución resultante puede ser vista en la Figura 4.16 que correspondería a los audios menores a 9[s], esto principalmente a 2 factores:

- **Relleno:** O *padding*, corresponde al paso de prolongar las secuencias que se encuentren en un mismo bloque de entrenamiento para que todas tengan el mismo largo. Esto puede ser problemático si en un bloque una secuencia es notoriamente más larga que las demás, dado que significaría generar mucho relleno lo cual es información falsa y puede empeorar el aprendizaje del modelo.

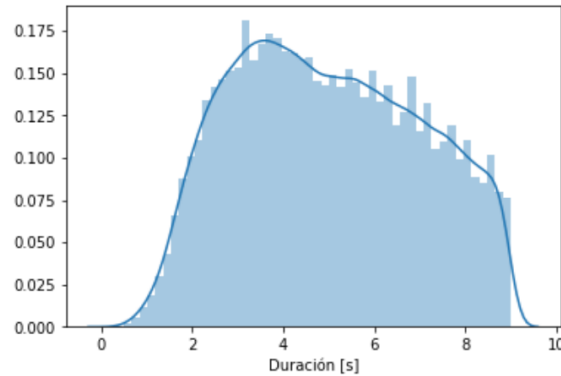


Figura 4.16: Distribución de la duración de los audios menores a 9[s], para el conjunto final.
Fuente: Elaboración Propia.

- **Memoria:** El coste computacional de entrenar con CTC puede crecer exponencialmente con secuencias muy largas.

Además de lo anterior, dado que entrenar con este volumen de datos todavía resulta ser costoso en el tiempo se decide utilizar una muestra de 100 [hrs] extraída del conjunto anterior. De esta forma se hace factible utilizar más de una semilla para la obtención de resultados más robustos y poder realizar más variaciones dentro de un plazo adecuado. En la Figura 4.17 vemos que las clases o *target* se mantienen similares en las 2 distribuciones, lo cual también se aprecia para los géneros en la Figura 4.18 y acentos en la Figura 4.19. Debido a la similitud entre las muestras, se concluye que utilizar el conjunto de 100 [hrs] puede entregar información relevante respecto a que decisiones en la arquitectura mejorarían una clasificación para el conjunto de 200[hrs], lo cual será el *pipeline* que se utilizará en el trabajo.

Se realizará la evaluación de los hiperparámetros con el set de validación de Common Voice. Esto, dado que es un conjunto ruidoso y multi-acento, por lo cual la dificultad inherente respecto a lograr buenos resultados en el lo hacen ideal para comprobar la generalización de los modelos.

4.2. Extracción de características

Varias son las alternativas para extraer características de la señal de voz, se opta por utilizar espectrogramas en la escala de Mel o *Log Mel filterbank*, a pesar que en el estudio en el cual se basa la arquitectura inicial [Graves et al., 2006] se utilizan coeficientes cepstrales o *Mfcc*. Esto dado que en trabajos más recientes con redes neuronales convolucionales [Amodei et al., 2016] son de utilidad dado que no se omite información y permiten que la propia capa interna realice la abstracción de características. Una comparación de la señal de habla con su representación como espectrograma se puede ver en la Figura 4.20.

Además dado que los audios ya se encuentran ordenados de manera descendente según su

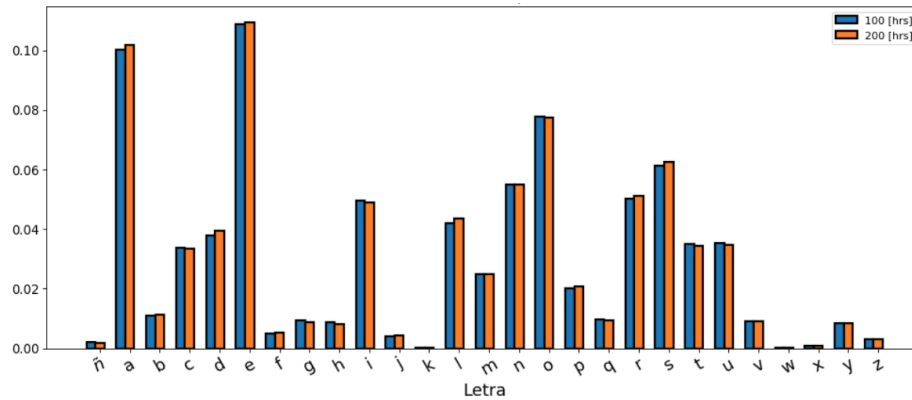


Figura 4.17: Distribución de clases (letras) para la muestras de 100 [hrs] y 200[hrs].
Fuente: Elaboración Propia.

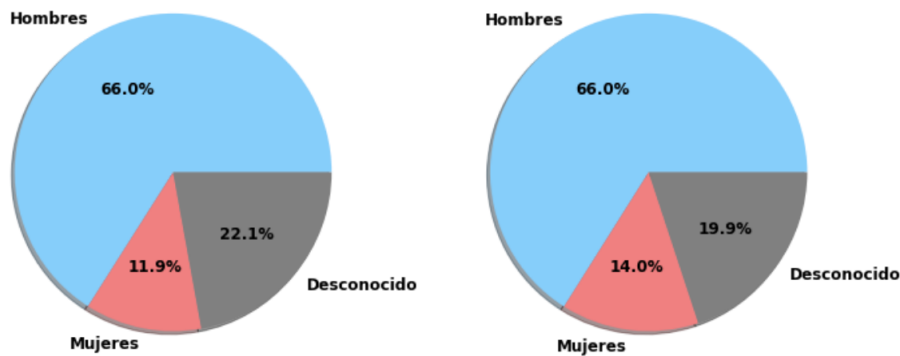


Figura 4.18: Distribución de géneros para la muestra de 100 [hrs] (izquierda) y 200[hrs] (derecha).

Fuente: Elaboración Propia.

largo, se realiza el *padding* o relleno previo a entrenar con el fin de ahorrar tiempo y memoria en la ejecución.

4.3. Arquitecturas redes neuronales

Empezamos con un modelo basado principalmente en los trabajos [Graves et al., 2006, Graves et al., 2013], el cual vendría a corresponder al uso de la función de pérdida CTC junto a redes recurrentes BiLSTM (bidireccionales). Un esquema adecuado se puede ver en la Figura 4.21. Estas capas permiten encontrar relaciones de contexto entre *timesteps* previos y futuros en una secuencia de datos, lo cual es ideal para modelos de reconocimiento de habla.

Se comienza con una configuración de 3 capas recurrentes BiLSTM de 208 unidades o neuronas en cada una ($\sim 2.5M$ parámetros). Con una entrada de dimensión 40 para los espectrogramas en la escala de Mel o *Log-Mel Filterbank* de los audios y una capa de salida

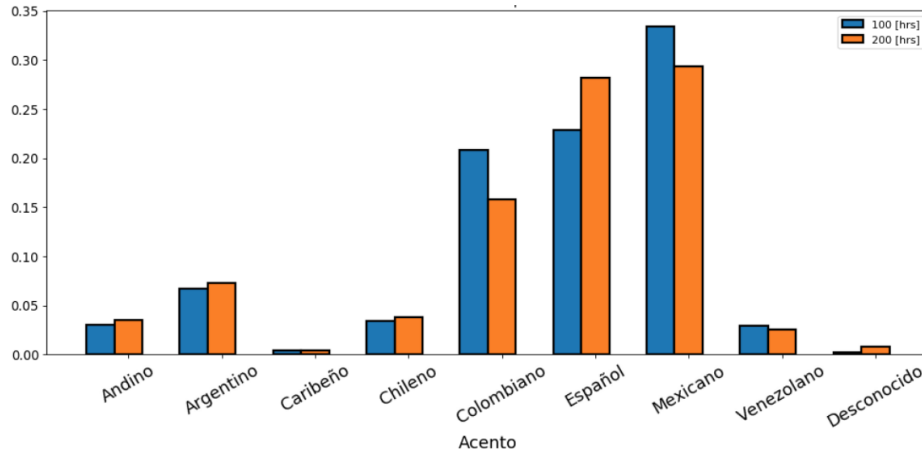


Figura 4.19: Distribución de acentos para la muestra de 100 [hrs] y 200[hrs].
Fuente: Elaboración Propia.

softmax con 29 clases correspondientes al alfabeto español más una clase nula o *blank* y otra para los silencios/espacios " ". A diferencia de los trabajos mencionados, como optimizador inicial se ocupará Adam debido a su adaptación respecto a la tasa de aprendizaje para cada parámetro y efectividad en problemas con gradiente ruidoso [Kingma y Ba, 2014].

En los entrenamientos a menos que se indique lo contrario, se utilizará un *mini-batch* de tamaño 80 para cada actualización del modelo en las épocas y **3 semillas** distintas con el fin de poder comparar resultados más robustos. El hecho de no añadir más semillas, se debe a los largos tiempos de entrenamiento que conlleva ajustar estos modelos. Lo cual, también es comentado en la literatura [Saini y Kaur, 2013, Graves y Jaitly, 2014, Amodei et al., 2016].

4.3.1. Transformación entrada

En [Yu y Deng, 2016] se explica que en algunos trabajos se realiza una transformación a la entrada previa a entrenar después de procesar los audios. Con el fin de eliminar variaciones muy grandes de valores entre entradas debido al ruido. Sobre todo, en casos como el de este trabajo con un corpus conformado de distintas fuentes. Esta transformación, puede ser una normalización de la media para cada entrada, que viene de la forma:

$$X'_i = X_i - \frac{1}{N} \sum_j X_j.$$

Donde X_i corresponde a cada señal procesada, notar que si bien esta normalización debería reducir el ruido puede provocar una pérdida de información general en los datos. La segunda transformación viene dada por una estandarización global de los datos:

$$X'_i = \frac{X_i - \mu}{\sigma}.$$

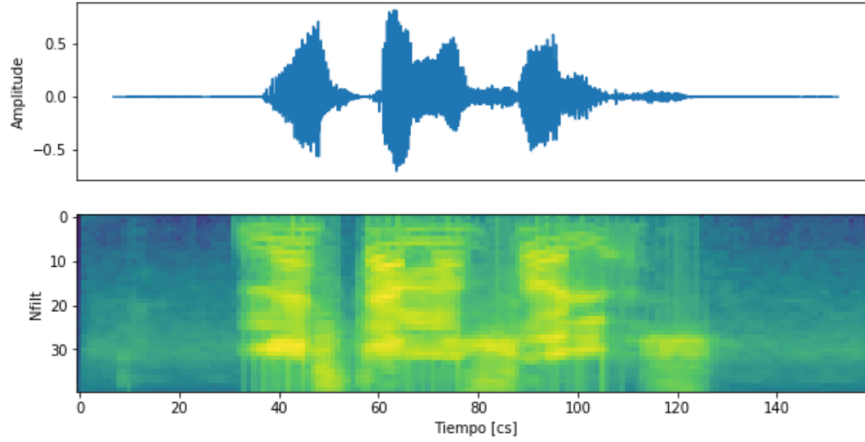


Figura 4.20: Comparación de la frase “no estamos solos” en su representación como **señal de audio** (arriba) y **Espectrograma en la escala de Mel** (abajo).

Fuente: Elaboración Propia.

Donde μ y σ son la media y desviación estándar global respectivamente, una visualización respecto al resultado de estas transformaciones se puede ver en la Figura 4.22. Se procede a entrenar por **10 épocas** para cada opción, los resultados respecto al error de entrenamiento y validación se pueden ver en la Figura 4.23. Notamos que para obtener una mejor comparación cualitativa se gráfica desde la segunda época, debido a que la magnitud de la primera difiere mucho respecto a las siguientes. Se puede apreciar un claro *overfitting* en los 3 modelos, sin embargo se opta por elegir la entrada estandarizada dado que obtiene el menor error de validación, 33,3.

4.3.2. Optimizador

Se desea comparar Adam con otras 2 variaciones de el mismo optimizador para el problema, los cuales serán:

- **Adamax:** Se diferencia de Adam en la actualización de la variable v_t . En el algoritmo original esta encargada de escalar el gradiente de manera inversamente proporcional a la norma l_2 de los anteriores:

$$g_t = \nabla_{\theta_t} L(\theta_t),$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) |g_t|^2.$$

Recordando que g_t es el gradiente y θ_t los parámetros del modelo, podemos generalizar esta expresión para la norma l_p :

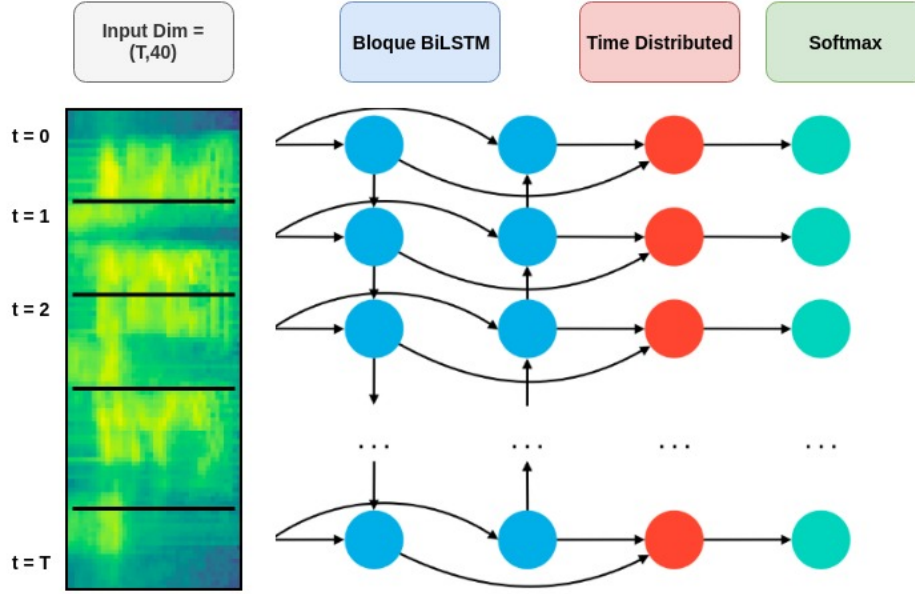


Figura 4.21: Diagrama respecto a la arquitectura base. La penúltima capa llamada *time distributed*, se agrega para referirse a que cada *timestep* producirá una distribución de probabilidad en las clases de la capa *softmax*.

Fuente: Elaboración Propia.

$$v_t = \beta_2^p v_{t-1} + (1 - \beta_2^p) |g_t|^p.$$

Según lo comentado en [Kingma y Ba, 2014] para valores grandes de p se exhibe un comportamiento numéricamente inestable en la optimización, sin embargo con $p \rightarrow \infty$ se obtiene un algoritmo estable.

- **Nadam:** Se refiere a la implementación del momentum de Nesterov en Adam:

$$g_t = \nabla_{\theta_t} L(\theta_t - \gamma m_{t-1}),$$

$$m_t = \gamma m_{t-1} + \eta g_t,$$

$$\theta_{t+1} = \theta_t - m_t.$$

La idea es mover los parámetros más precisamente en la dirección del gradiente, por medio de actualizarlos antes de computar g_t . Adaptando esto en Adam la ecuación del gradiente descendente queda como:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \left(\beta_1 \hat{m}_1 + \frac{(1 - \beta_1) g_t}{1 - \beta_1^t} \right).$$

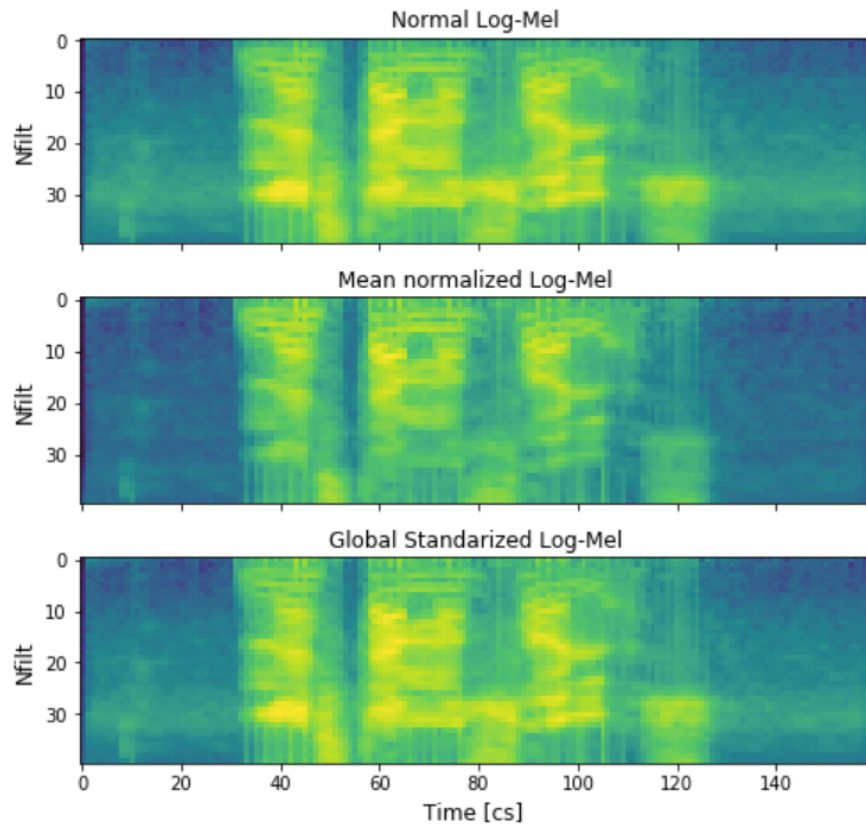


Figura 4.22: Comparación respecto a transformaciones del espectrograma de entrada.

Fuente: Elaboración Propia.

Se entrena con estos 2 algoritmos utilizando la representación elegida previamente. Los resultados pueden ser vistos en la Figura 4.24, donde podemos notar *overfitting* para el algoritmo Nadam con una divergencia del error de validación en épocas posteriores. Esto puede deberse a que el ajuste de *Nesterov* acelera demasiado el *momentum* empeorando los resultados. En general los resultados para Adam y Adamax son muy similares pero se decide seguir ocupado el primero dado que obtiene una mayor precisión en la validación $0,8\% > 0,4\%$. Además Adam teniendo el mismo error de validación posee una menor pérdida respecto a los datos de entrenamiento.

4.3.3. Número de capas

En [Graves et al., 2013] se comenta que el tener redes más profundas entrega mejores resultados para *speech recognition*, sin embargo esto conlleva un costo mayor en memoria y tiempo al computar el entrenamiento, inclusive si se mantiene el número de parámetros. Por ello se probará el desempeño para bloques de **1,2,3,4 y 5** capas *BiLSTM*, manteniendo aproximadamente la cantidad de parámetros de entrenamiento entre experimentos ($\sim 2.5M$).

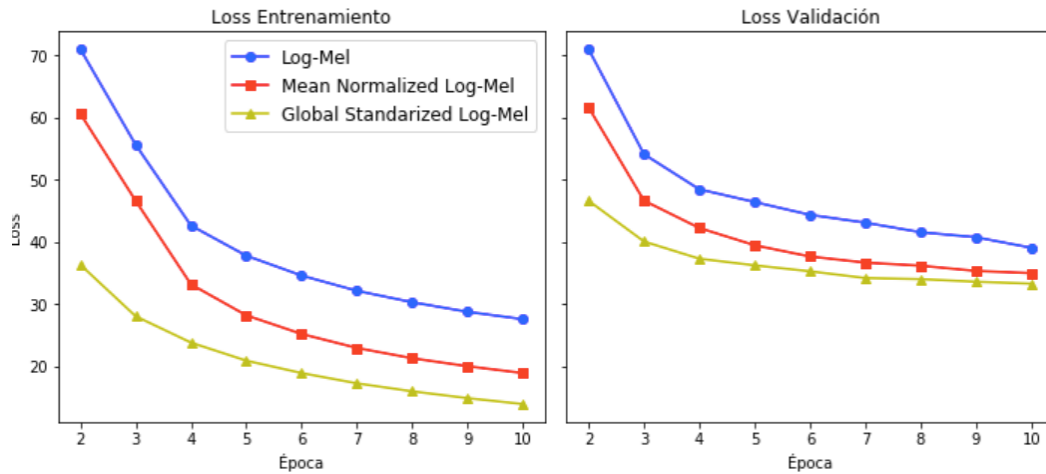


Figura 4.23: Error de entrenamiento/validación según tipo de input [2-10 épocas].

Fuente: Elaboración Propia.

El resultado para 10 épocas puede ser visto en la figura 4.25, donde notamos que al aumentar las capas hasta 4 también mejora el aprendizaje de la red. A lo cual podemos atribuir una mayor capacidad de abstracción interna respecto a la entrada para formar relaciones entre los parámetros, a pesar de mantener la misma cantidad de parámetros. Sin embargo, esto también conlleva a un mayor costo en tiempo en el entrenamiento de la red. Elegimos entonces la arquitectura con 4 capas *BiLSTM* para los siguientes experimentos, dado que a pesar que con 5 capas se puede notar una convergencia en épocas posteriores en general presenta un peor rendimiento.

4.3.4. Convolución

En [Amodei et al., 2016] se utilizan capas convolucionales de 1 y 2 dimensiones como entrada en la red neuronal, haciendo un simil al *front-end* en acercamiento tradicionales. Es de interés entonces verificar si para el caso presente añadir estas capas logra mejorar los resultados, un diagrama de la arquitectura resultante se puede ver en la Figura 4.26. Además se deben tomar en cuenta los siguientes factores:

- Se utilizará un relleno o *padding* al momento de aplicar el filtro convolucional con el fin de no disminuir el largo de la entrada, algo esencial para el correcto funcionamiento de la función de pérdida CTC.
- Una capa de *max pooling* filtrará los resultados de la última convolución con el fin de disminuir el *overfitting* y la dimensionalidad.

En la Tabla 4.15 se entra en detalle respecto a las configuraciones de la entrada convolucional de la red, donde notamos la disminución de tamaño del kernel para el eje temporal a medida que se aumenta el número de capas. La razón de ello es para evitar tener demasiada cantidad

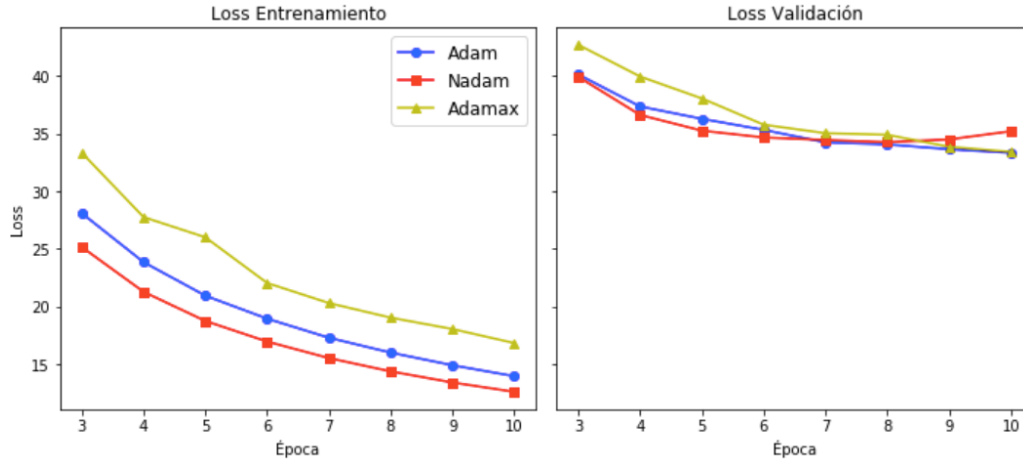


Figura 4.24: Error de entrenamiento/validación según optimizador [2-10 épocas].
Fuente: Elaboración Propia.

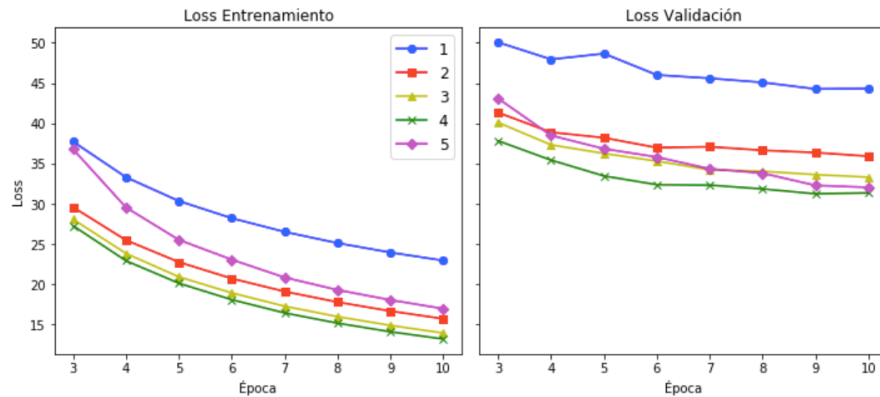


Figura 4.25: Error de entrenamiento/validación según cantidad de capas/layers [2-10 épocas].

Fuente: Elaboración Propia.

de información con *padding*. La Figura 4.27 muestra las curvas de pérdida respectivas, donde podemos notar el mal desempeño al utilizar la arquitectura con 3 capas. Esto posiblemente a un problema de gradiente desvaneciente, se crea una segunda visualización en la Figura 4.28 que muestra un desempeño muy similar entre las otras arquitecturas. Por lo cual se decide por elegir utilizando la Tabla 4.15 donde claramente el modelo con 2 capas convolucionales posee mejores resultados. Una de las razones se puede atribuir a que permite tener una mejor abstracción de las características del espectrograma, sin tener problemas respecto a la convergencia del gradiente.

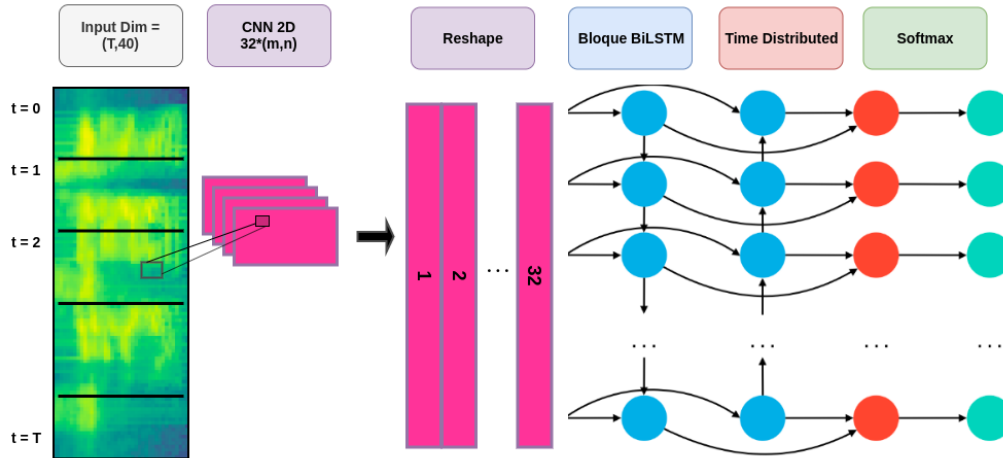


Figura 4.26: Diagrama respecto a la arquitectura base ocupando como entradas capas convolucionales, se debe notar el uso de *reshape* con el fin de concatenar los filtros resultantes en el eje de las frecuencias (mismo largo temporal).

Fuente: Elaboración Propia.

Tabla 4.15: Métricas de validación capas convolucionales. Fuente: Elaboración Propia.

Filtros	Kernel	Stride	Menor error val.	Mayor precisión val.
-	-	-	31.2	1.1 %
32	(41,5)	(1,1)	30.5	0.9 %
32,32	(20,5),(21,5)	(1,1),(1,1)	29.1	2.0 %
32,32,32	(10,5),(10,5),(10,5)	(1,1),(1,2),(1,2)	111.3	-

4.3.5. Regularización

En todos los casos anteriores se pudo notar una tendencia al *overfitting* en los modelos, es decir, clasificadores que no están generalizando lo suficiente y poseen un evidente mejor desempeño en el error del conjunto de entrenamiento. Por lo cual, analizaremos si la alternativa de ocupar técnicas de regularización puede mejorar el error de validación respecto al de entrenamiento. Las estrategias a ocupar corresponden a las siguientes:

- **Gaussian Noise:** Una técnica común en procesamiento de señales, corresponde a sumar un vector de muestra respectivas a una distribución normal con una variación estándar asignada. Así el coeficiente del espectrograma que entra a la red es transformado en una versión ruidosa, con lo cual se espera que el modelo logre generalizar el aprendizaje respecto a las características más importantes. Para este caso se ocupará una desviación estándar $\sigma = 0,4$. Su efecto en el espectrograma puede ser visto en la Figura 4.29.
- **SpecAugment:** Presentada en [Park et al., 2019] se trata de reemplazar por medio de franjas horizontales y verticales secciones aleatorias del espectrograma de entrada,

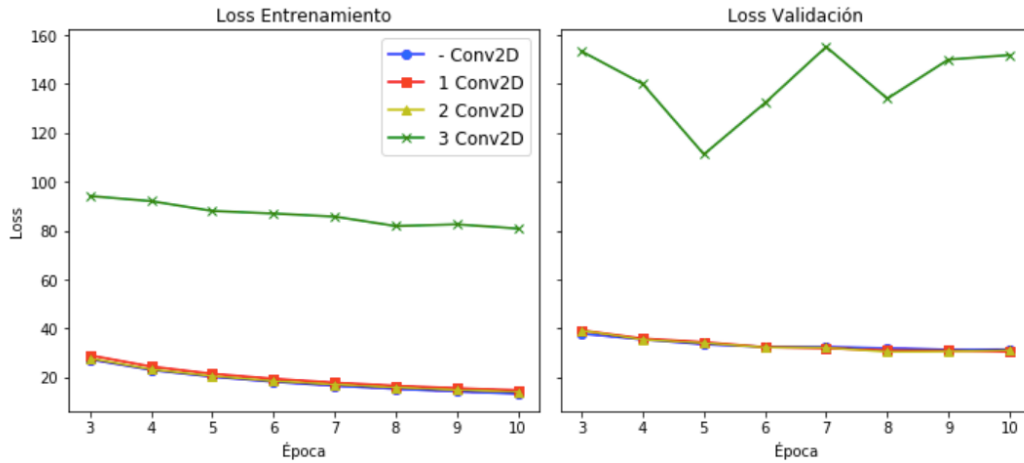


Figura 4.27: Error de entrenamiento/validación según capas convolucionales [2-10 épocas].
Fuente: Elaboración Propia.

puede ser visto en la Figura 4.29. Notar que la motivación principal de esta técnica es viene de ver el espectrograma y la transcripción como un problema de clasificación de imágenes.

- **Dropout:** Corresponde a bloquear ciertas salidas de las capas internas en la red durante el ajuste de parámetros, con el fin de forzar el entrenamiento sobre nuevos caminos y que todas las unidades participen en la clasificación. De esta manera se aumenta la redundancia y se dificulta el sobre-ajuste respecto al conjunto de entrenamiento.

Se inicia entrenando con cada una de estas técnicas por separado, los resultados se pueden ver en la Figura 4.30. Las técnicas de regularización muestran ser efectivas en dificultar el sobreajuste del modelo con el conjunto de entrenamiento, sin embargo para el caso del ruido Gaussiano se puede notar *underfitting*. Por lo cual se descarta esta técnica, para los otros métodos se obtiene una mejora de la pérdida en el conjunto de validación.

También es de interés comparar otro acercamiento combinando *SpecAugment* con *Dropout* y verificar si mejora los resultados en la validación, las curvas de pérdida respectiva se ven en la Figura 4.31. Se obtiene desempeño similar respecto al error, pero tal y como se puede esperar al aumentar la tasa de *dropout* también se dificulta más el ajuste en el set de entrenamiento. Para un análisis más fino se cuenta con la Tabla 4.16, donde podemos ver que en general al ocupar regularización, disminuye la pérdida en el set de validación. Pero a la vez también baja la precisión, esto se puede asociar a que existan algunas muestras en el conjunto de validación muy parecidas al set de entrenamiento. Por lo que al sobre-ajustar se obtiene una clasificación correcta en ellas, pero a la vez un aprendizaje respecto a relaciones más específicas en los espectrogramas.

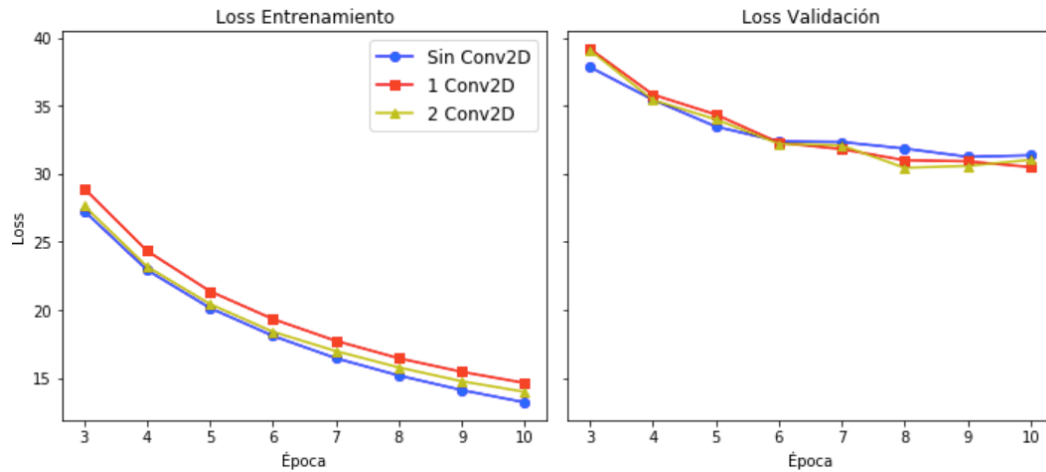


Figura 4.28: Error de entrenamiento/validación según capas convolucionales (sin la arquitectura con 3 capas) [2-10 épocas].

Fuente: Elaboración Propia.

Tabla 4.16: Métricas de validación tipo de regularización. Fuente: Elaboración Propia.

Tipo de regularización	Menor error val.	Mayor precisión val.
-	29.1	2 %
<i>Dropout</i> 10 %	27.4	1.5 %
<i>Dropout</i> 10 % + <i>SpecAugment</i>	26.9	0.7 %
<i>Dropout</i> 20 % + <i>SpecAugment</i>	27.5	0.5 %

4.4. Modelos finales

Se seleccionarán 3 modelos con el fin de validar la arquitectura y *pipeline* propuesto, los cuales corresponderán a:

- **Modelo 1:** Se elije el modelo inicial, correspondiente a la arquitectura sin capas convolucionales y con 3 *BiLSTM*. Con entrada el espectrograma sin estandarizar, esto con el fin de verificar si las mejoras propuestas en las variaciones de hiperparámetros suponen una mejora relativa.
- **Modelo 2:** A partir de la Tabla 4.16, seleccionamos el modelo que obtuvo el mejor error de validación. Se espera que la regularización mejore todavía más al utilizar una muestra más grande de datos.
- **Modelo 3:** Se decide entrenar un modelo con más unidades recurrentes, basándose en la arquitectura del modelo 2. Sin embargo, dado que aumentar las neuronas de la red puede provocar un peor desempeño en la generalización al tender a sobreajustar también se aumentará la tasa de *dropout* de 10 % a 30 %.

Estos 3 modelos se entrenarán con la muestra de 200[hrs], también se utilizarán más épocas

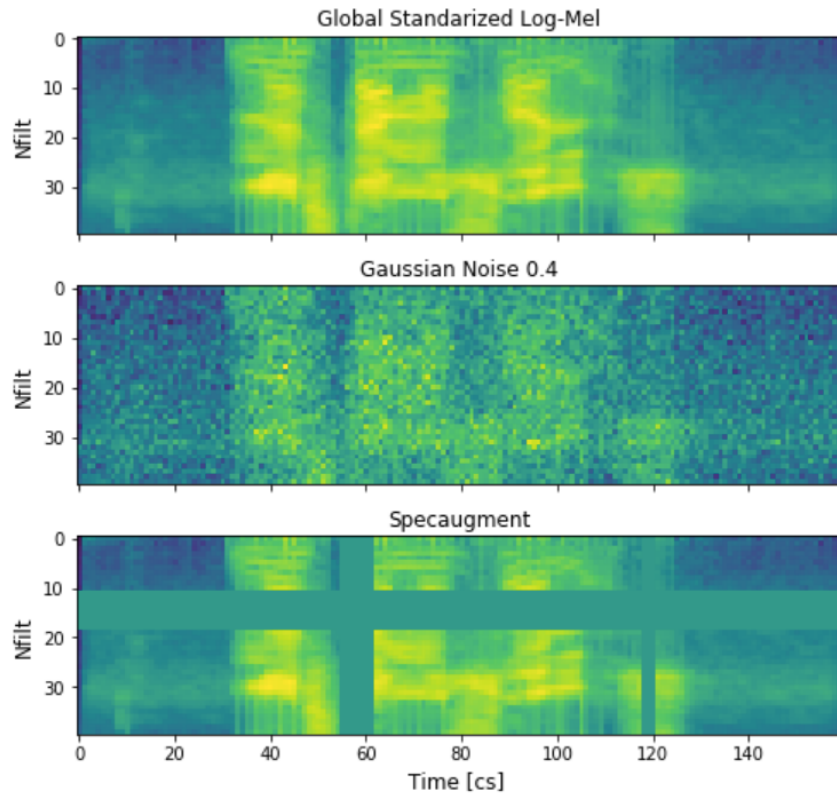


Figura 4.29: Comparación respecto a técnicas de aumento de datos para el espectrograma de entrada.

Fuente: Elaboración Propia.

que en los entrenamientos previos dado que se desea llevar a la convergencia los parámetros de las redes. En la Figura 4.32 se puede ver el resultado respecto a las pérdidas de entrenamiento para estos modelos. Notamos como la regularización es efectiva también al utilizar la muestra más grande, y el como los modelos obtenidos con la variación de hiperparámetros presentan una mejora relativa respecto al inicial.

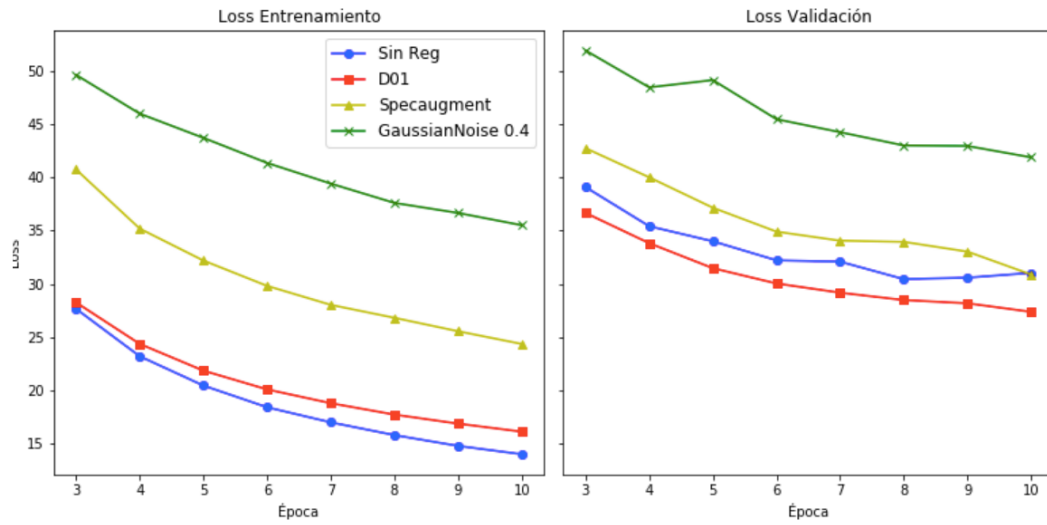


Figura 4.30: Error de entrenamiento/validación según tipo de regularización [2-10 épocas].
Fuente: Elaboración Propia.

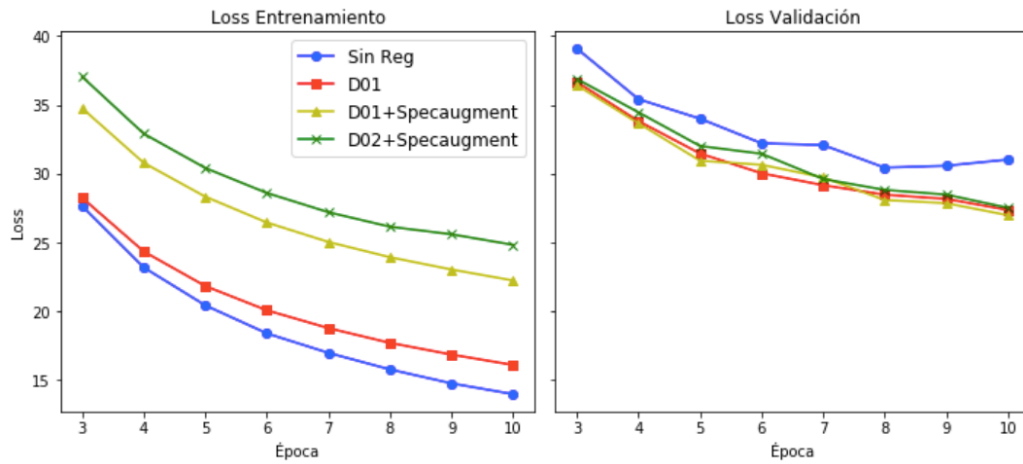


Figura 4.31: Error de entrenamiento/validación según tipo de combinación de regularizaciones [2-10 épocas].
Fuente: Elaboración Propia.

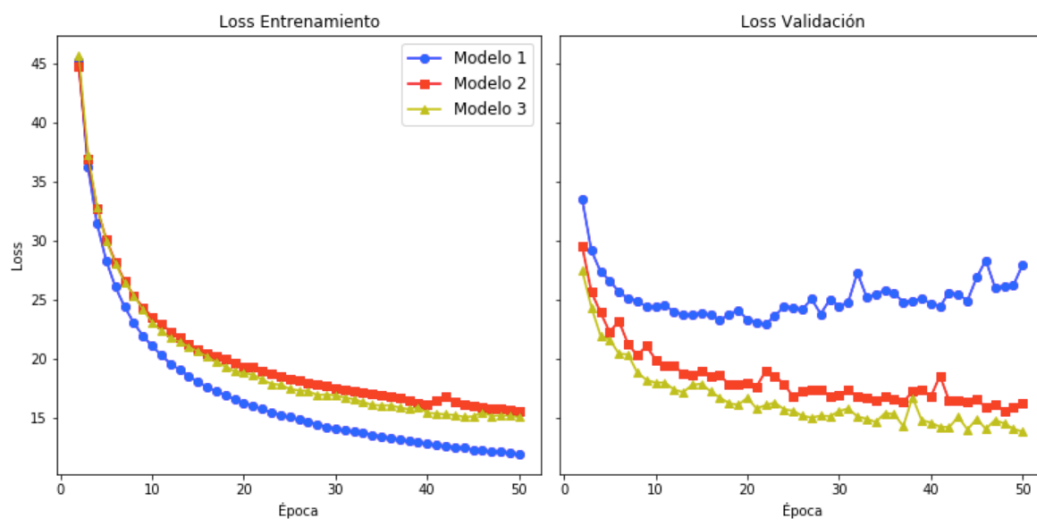


Figura 4.32: Error de entrenamiento/validación según modelo final [1-50 épocas].
Fuente: Elaboración Propia.

CAPÍTULO 5

VALIDACIÓN DE LA SOLUCIÓN

En esta sección se validará los modelos creados, tomando en cuenta los objetivos específicos iniciales. La comparación principal será realizada con el modelo *baseline*, analizando las causas en las diferencias de resultados.

5.1. Métricas de validación modelos

Por medio del entrenamiento de los modelos finales seleccionados en la sección anterior, se calculan las métricas *CER* y *WER*. Tanto para el conjunto de validación como para los 2 de pruebas ya definidos, además se evaluará también el sistema *baseline* en las mismas condiciones. Se debe destacar que este sistema fue entrenado con más del doble de horas, respecto a los modelos finales de este trabajo. Los resultados asociados a las arquitecturas y sus detalles, se presentan en la Tabla 5.17 y la Tabla 5.18.

A partir de los resultados se puede notar que la arquitectura final de redes neuronales profundas del presente trabajo supera al *baseline*, tanto para el *CER* como el *WER*. Si se compara la arquitectura inicial 1 respecto al modelo *baseline* se puede ver el efecto de optimizar respecto a la clasificación de letras y no palabras, obteniéndose una brecha entre el *CER* y *WER* mayor que en el caso del modelo base. También de los resultados se verifica que las modificaciones del modelo 1 para la muestra pequeña, logran generalizar una mejora del modelo con el entrenamiento de 200 [hrs] dado los buenos resultados para el 2 y 3.

Tabla 5.17: Tasa de error por carácter (*CER*) modelos finales. Fuente: Elaboración Propia.

Modelo	CV Val	Ciempies Test	CV Test
Baseline (HMM+GMM)	29.7	25.7	34.7
Modelo 1 inicial (Adam + 3xBiLSTM)	17.6	12.6	19.5
Modelo 2 (Adam + 2xConv2D + 3xBiLSTM + Reg.)	9.2	6.4	11.7
Modelo 3 (Adam + 2xConv2D + 3xBiLSTM (triple complejidad) + Reg.)	7.9	5.6	9.8

Tabla 5.18: Tasa de error por palabra (*WER*) modelos finales. Fuente: Elaboración Propia.

Modelo	CV Val	Ciempies Test	CV Test
Baseline (HMM+GMM)	58.5	46.3	66.5
Modelo 1 inicial (Adam + 3xBiLSTM)	57.2	48.9	59
Modelo 2 (Adam + 2xConv2D + 3xBiLSTM + Reg.)	35.9	24.7	38.4
Modelo 3 (Adam + 2xConv2D + 3xBiLSTM (triple complejidad) + Reg.)	28.8	21.2	31.4

5.2. Modelo final de reconocimiento de habla

Este modelo presenta ser un acercamiento distinto al tradicional para el idioma español, de tipo *end-to-end* con la utilización de redes neuronales junto a *CTC*. Esta función de

pérdida se encarga de asociar cada *timestep* de la entrada con un carácter, en la Figura 5.33 se ejemplifica este proceso. Al realizar la predicción, si se ocupa el algoritmo *greedy* se selecciona para cada t de la entrada la clase más probable según lo indicado por la capa *softmax* de la red neuronal. Esto hace presente la utilidad de este modelo para identificar y segmentar secciones de audio con su correspondiente *target*. Entregando flexibilidad en caso de implementarse en una solución más general que requiera la identificación de palabras con su marca temporal.

Notamos que no depende de un modelo de pronunciación, por lo cual no se utilizan reglas creadas manualmente. Sin embargo tampoco se utiliza uno de lenguaje, el cual puede ser añadido para mejorar la transcripción final. Esto provoca que se puedan generar palabras fuera del vocabulario en la transcripción, aspecto que según la necesidad puede disminuir la eficacia del modelo. Como es el caso de un sistema de comandos, en el cual una letra errónea puede llevar al error.

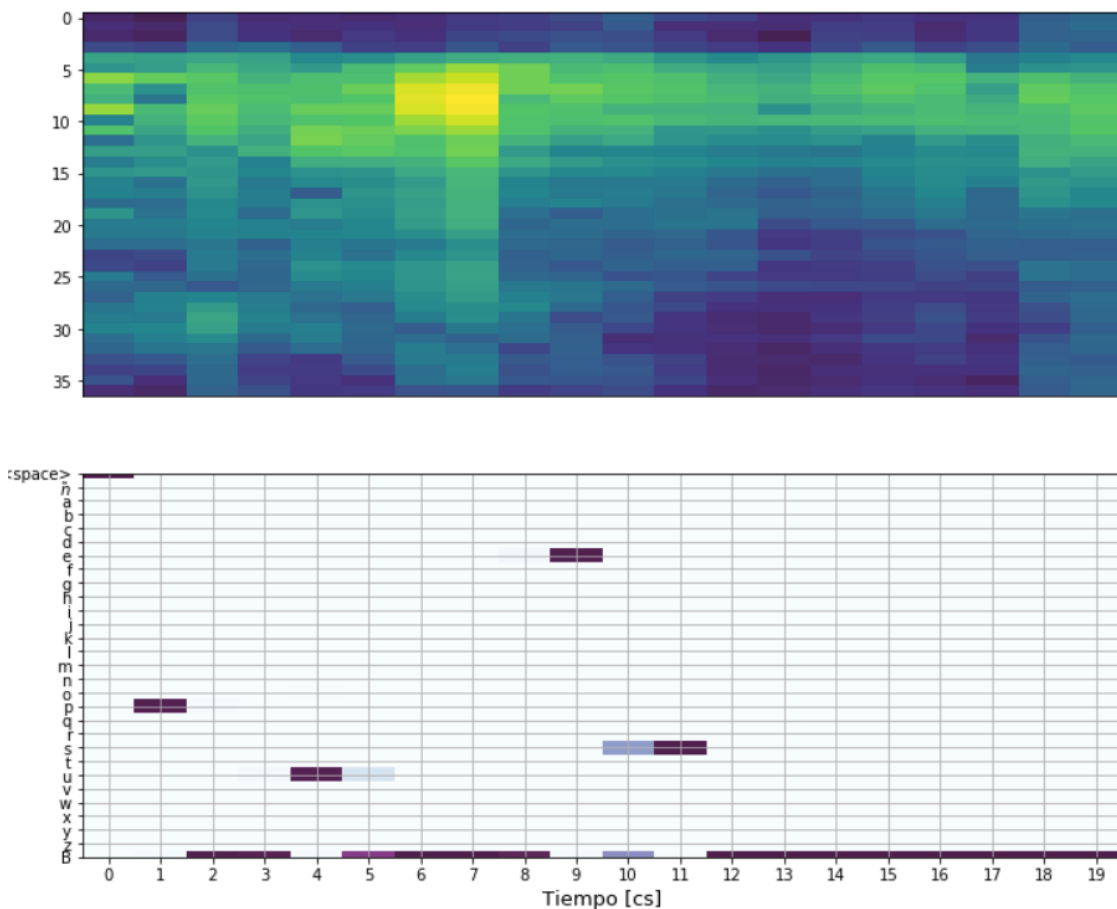


Figura 5.33: Ejemplo de distribución de probabilidad para cada t respecto a las clases para la frase “pues”.

Fuente: Elaboración Propia.

5.3. Comparación modelo *baseline*

El modelo *baseline* al tener una arquitectura tradicional basada en cadenas ocultas de Markov, utiliza un modelo acústico, de lenguaje y pronunciación. Esto puede ser útil para dominios con un vocabulario limitado, tal como asistentes automáticos de dictado, sistemas de comandos, etc. Pero en casos donde se tengan palabras fuera del vocabulario no acotadas por el modelo de lenguaje, se vuelven ineficaces para generar una transcripción correcta. Por otra parte, el uso de un modelo de pronunciación generado por conocimiento experto y no implícitamente a través de los datos puede ser problemática para una solución multi-acento. Finalmente ocupar la arquitectura GMM-HMM para modelar la señal acústica no logra ser lo suficientemente robusto ante el ruido en las entradas [Yu y Deng, 2016], lo cual es presente para los conjuntos de validación y pruebas. Estos factores acumulados generan un modelo que no puede generalizar correctamente, en contraste a un acercamiento con *deep learning* al problema y puede explicar el porque obtiene peores resultados al comparar.

Se procede a generar transcripciones de los conjuntos de pruebas para ejemplificar:

1. **Frase real:** “El origen de estas croquetas es una receta de la abuela”.
 - **Modelo *baseline*:** “gemir el origen está creo que hace es una receta de la abuela”.
 - **Modelo 3:** “el origen destas croques dases una receta de la abuela”.
2. **Frase real:** “por las mismas universidades este objetivo que esta”.
 - **Modelo *baseline*:** “por los universidades este objetivo que es”.
 - **Modelo 3:** “por las mias universidades en este betivo que esta”.
3. **Frase real:** “ni de peces, solo semillas”.
 - **Modelo *baseline*:** “mide trece sólo semillas”.
 - **Modelo 3:** “ni de peces solo cemillas”.
4. **Frase real:** “el indio ya se recogía, como un gato montés , dispuesto a saltar sobre mí”.
 - **Modelo *baseline*:** “el bien y ahí hace recogía como el gato montés dispuesto saltar sobre mi”.
 - **Modelo 3:** “el india ya se recogia como un gato montes dispuestoasaltar sobre a mi”.

A pesar de que el **modelo 3** pueda poseer errores debido a que no usa un modelo de lenguaje, los resultados de la Tabla 5.18 validan su efectividad como modelo de reconocimiento de habla. Además, al tener una baja tasa de error por carácter indica que existe la posibilidad de acoplar un modelo de lenguaje de caracteres. El cual se debería utilizar dentro del algoritmo de búsqueda del camino más probable en la decodificación CTC, con el fin de mejorar la tasa de error por palabra.

Finalmente se realiza una comparación del tiempo para obtener una transcripción a partir de un audio respecto al *set Ciempiess Test*. El **modelo 3** logra un mejor tiempo de respuesta en comparación al modelo *baseline* tomando 55[*min*] y 125[*min*] respectivamente. El modelo que ocupa *HMM* es más lento debido a la utilización de varios modelos para realizar la inferencia, lo cual también necesita un proceso de decodificación final.

CAPÍTULO 6

CONCLUSIONES

6.1. Aprendizaje y trabajo realizado

Se realizó un acercamiento directo al reconocimiento de habla en español de tipo hablante independiente y vocabulario abierto. Fue de vital importancia tener un proceso ordenado y consistente respecto al manejo de los corpus, que permitiera filtrar y ordenar adecuadamente las muestras. Otro aspecto clave fue el estudio de la literatura de distintos acercamientos con redes neuronales, con el fin de analizar los alcances y factibilidad de la técnica utilizada. El trabajo permitió mostrar el uso de conocimientos generales en la creación y mejora de un modelo de *Machine Learning*. A la vez que se tuvo que profundizar respecto a las variaciones posibles de una arquitectura con redes neuronales tanto convolucionales como recurrentes. Analizando para el problema de reconocimiento de habla, cuales hiperparámetros eran más adecuados. Guiándose por resultados empíricos y siguiendo un *pipeline* definido previamente con el fin de ordenar el avance hacia la solución.

6.2. Objetivos

Se construyó un modelo de redes neuronales utilizado previamente para el inglés [[Graves et al., 2006](#), [Graves et al., 2013](#)], para el español multi-acento con un modelo de identificación de caracteres. Completando el objetivo principal, utilizando *deep learning* con un modelo creado desde cero sin pre-entrenamiento.

Respecto al primer objetivo específico se superó la precisión del modelo *baseline* respecto a las métricas CER y WER. Se debe tomar en cuenta que este modelo se suele definir solo como acústico, por lo cual presenta limitaciones respecto a la ortografía en las transcripciones. Tales clasificaciones erróneas no empeoran mucho el CER pero el WER aumenta drásticamente en estos casos según lo estudiado en la literatura, dado que un carácter mal clasificado produce una palabra errónea. Se debe agregar que al utilizar este modelo que obtiene un aprendizaje intrínseco de los fonemas y sus caracteres, no se hace necesario el depender de conocimiento experto con un modelo de pronunciación. Lo cual presenta ser ideal para el español, donde estos modelos suelen estar enfocados en un solo acento.

Este acercamiento presenta ser nuevo para el español multi-acento, con el uso de CTC. Las modificaciones de la arquitectura se eligieron tanto por su uso en trabajos similares como también otras por ser decisiones comunes para los trabajos de *Machine Learning*. Tales como la normalización de la entrada y regularización. De esta manera se pudo analizar empíricamente sus efectos para esta arquitectura, en el contexto del español.

Respecto a la eficacia del modelo para lograr su objetivo, se puede ver en la Tabla 5.18 los rangos del 20-30 % para la tasa de error por palabra (WER). Lo cual en comparación al *ba-*

seline presenta ser una gran mejora, sin embargo si se compara con trabajos más actuales en la literatura [Amodei et al., 2016, Chiu et al., 2018] podemos evidenciar una brecha con resultados del estado del arte en inglés. Las causas principales además de la complejidad de las arquitecturas, son la utilización de modelos pre entrenados y un volumen de datos mucho mayor. Se obtiene un mejor tiempo de predicción para el modelo más complejo de este trabajo en comparación al *baseline*, lo cual lo hace más apto para una aplicación que requiera una respuesta en tiempo real.

6.3. Trabajo futuro

A pesar de que los resultados superan el *baseline* propuesto, como se mencionó previamente en comparación a los resultados en inglés existe una diferencia notable. Una de las causas es el poco volumen de datos disponible de manera gratuita para el español. Es por ello que como posibilidad extra se pueden investigar más métodos de *data augmentation*, con el fin de generar un modelo más robusto. En este trabajo se modificó directamente la imagen del espectrograma, sin embargo el acercamiento más común es crear nuevos audios por medio del cambio de tono de voz, velocidad y combinación con ruido (tráfico, música, etc).

Por otro lado, la función de pérdida utilizada presenta ser una de varias para modelar un mapeo de secuencias, para el reconocimiento de habla en inglés. Por lo cual sería de interés el ocupar otra arquitectura de redes neuronales, con un acercamiento *end-to-end*. Como las arquitecturas con una capa atencional, que suelen obtener buenos resultados cuando se utilizan estas funciones de pérdida, pero conllevan un modelo más pesado. Otro aspecto interesante que se podría mejorar tiene que ver con el uso de un modelo de lenguaje como se mencionó previamente, y de que algoritmo utilizar para su implementación. Cada vez es más común el uso de modelos de secuencias neuronales en estos problemas, además que los corpus existentes para textos superan en demasía a los de *speech*. Específicamente para la arquitectura utilizada en este trabajo, un opción posible vendría a ser un modelo de lenguaje de caracteres el cual entregue información en cada paso de la decodificación con CTC.

Por último, un problema que no fue cubierto en esta memoria tiene que ver con el manejo de transcripciones erróneas en el conjunto de entrenamiento. En la literatura dado que se utilizan gran cantidad de datos del orden de las ~ 1000 horas, no se suele hacer hincapié en esto. Sin embargo, para el caso del español al poseer un menor volumen de datos, un análisis e implementación respecto a técnicas para mejorar la calidad de los datos podría causar un impacto favorable.

REFERENCIAS BIBLIOGRÁFICAS

- [Amodei *et al.*, 2016] Amodei, Dario and Ananthanarayanan, Sundaram and Anubhai, Rishita and Bai, Jingliang and Battenberg, Eric and Case, Carl and Casper, Jared and Catanzaro, Bryan and Cheng, Qiang and Chen, Guoliang and others (2016). Deep speech 2: End-to-end speech recognition in english and mandarin. En *International conference on machine learning*, pp. 173–182.
- [Anusuya y Katti, 2010] Anusuya, M. y Katti, S. K. (2010). Speech recognition by machine, a review. *arXiv preprint arXiv:1001.2267*.
- [Benzeghiba *et al.*, 2007] Benzeghiba, Mohamed and De Mori, Renato and Deroo, Olivier and Dupont, Stephane and Erbes, Teodora and Juvet, Denis and Fissore, Luciano and Laface, Pietro and Mertins, Alfred and Ris, Christophe and others (2007). Automatic speech recognition and speech variability: A review. *Speech communication*, 49(10-11):763–786.
- [Castellanos y Gamboa,] Castellanos, H. y Gamboa, C. Corpus dimex100.
- [Chan *et al.*, 2015] Chan, W., Jaitly, N., Le, Q. V., y Vinyals, O. (2015). Listen, attend and spell. *arXiv preprint arXiv:1508.01211*.
- [Chiu *et al.*, 2018] Chiu, Chung-Cheng and Sainath, Tara N and Wu, Yonghui and Prabhavalkar, Rohit and Nguyen, Patrick and Chen, Zhifeng and Kannan, Anjuli and Weiss, Ron J and Rao, Kanishka and Gonina, Ekaterina and others (2018). State-of-the-art speech recognition with sequence-to-sequence models. En *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4774–4778. IEEE.
- [Gales *et al.*, 2008] Gales, Mark and Young, Steve and others (2008). The application of hidden markov models in speech recognition. *Foundations and Trends® in Signal Processing*, 1(3):195–304.
- [Goodfellow *et al.*, 2016] Goodfellow, I., Bengio, Y., y Courville, A. (2016). *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>.
- [Graves *et al.*, 2006] Graves, A., Fernández, S., Gomez, F., y Schmidhuber, J. (2006). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. En *Proceedings of the 23rd international conference on Machine learning*, pp. 369–376. ACM.
- [Graves y Jaitly, 2014] Graves, A. y Jaitly, N. (2014). Towards end-to-end speech recognition with recurrent neural networks. En *International Conference on Machine Learning*, pp. 1764–1772.
- [Graves *et al.*, 2013] Graves, A., Mohamed, A.-r., y Hinton, G. (2013). Speech recognition with deep recurrent neural networks. En *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6645–6649. IEEE.

- [Graves y Schmidhuber, 2005] Graves, A. y Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks*, 18(5-6):602–610.
- [Hernández-Mena y Herrera-Camacho, 2014] Hernández-Mena, C. D. y Herrera-Camacho, J. (2014). Ciempiess: A new open-sourced mexican spanish radio corpus. En *Proceedings of the ninth international conference on language resources and evaluation (LREC'14)*, pp. 371–375. European Language Resources Association (ELRA) Reykjavik, Iceland.
- [Hochreiter y Schmidhuber, 1997] Hochreiter, S. y Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.
- [Juang y Rabiner, 2005] Juang, B.-H. y Rabiner, L. R. (2005). Automatic speech recognition—a brief history of the technology development. *Georgia Institute of Technology. Atlanta Rutgers University and the University of California. Santa Barbara*, 1:67.
- [Jyothi, 2017] Jyothi, P. (2017). Automatic speech recognition - an overview. obtenido de <https://www.youtube.com/watch?v=q67z7PTGRi8&t=940s>.
- [Kamath y Whitaker, 2019] Kamath, U. L. y Whitaker, J. (2019). *DEEP LEARNING FOR NLP AND SPEECH RECOGNITION*. Springer.
- [Kingma y Ba, 2014] Kingma, D. P. y Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- [Maas et al., 2015] Maas, A., Xie, Z., Jurafsky, D., y Ng, A. (2015). Lexicon-free conversational speech recognition with neural networks. En *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 345–354.
- [Muda et al., 2010] Muda, L., Begam, M., y Elamvazuthi, I. (2010). Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw) techniques. *arXiv preprint arXiv:1003.4083*.
- [Park et al., 2019] Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., y Le, Q. V. (2019). SpecAugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*.
- [Saini y Kaur, 2013] Saini, P. y Kaur, P. (2013). Automatic speech recognition: A review. *International Journal of Engineering Trends and Technology*, 4(2):1–5.
- [Singh et al., 2012] Singh, B., Kapur, N., y Kaur, P. (2012). Speech recognition with hidden markov model: a review. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(3).
- [Yu y Deng, 2016] Yu, D. y Deng, L. (2016). Automatic speech recognition: A deep learning approach.