Universidad de Buenos Aires

Facultad de Ciencias Exactas y Naturales

Departamento de Computación

Aplicación de Redes Neuronales Convolucionales al Problema de Turn-Taking Usando Espectrogramas

Tesis de Licenciatura en Ciencias de la Computación

Sebastián Matías Sicardi

ssicardi@dc.uba.ar
L.U.: 42/13

Directores: Pablo Brusco y Pablo Riera

Buenos Aires, 2020

APLICACIÓN DE REDES NEURONALES CONVOLUCIONALES AL PROBLEMA DE TURN-TAKING USANDO ESPECTROGRAMAS

En una conversación hablada entre dos personas se da un intercambio implícito de señales que contribuyen a una mayor fluidez y naturalidad del diálogo. Este es el caso de las llamadas *pistas de transición de turno* (o *turn-taking cues* en inglés). Dichas pistas pueden entenderse como señales acústico-prosódicas, sintácticas o gestuales que anticipan eventos relacionados al manejo de turnos en una conversación y que permiten al interlocutor predecir cómo sucederá el turno actual.

En la actualidad, los problemas relacionados a la detección de estas pistas son mayormente analizados mediante técnicas de aprendizaje automático, más precisamente con redes neuronales recurrentes. Por otro lado, en otras tareas de procesamiento de audio se realizaron grandes avances mediante la combinación de *redes neuronales convolucionales* (CNNs) aplicadas sobre el *espectrograma* de la señal acústica. El objetivo de esta tesis es investigar la tarea de predicción de *transiciones de turno* utilizando CNNs.

Para ello, utilizamos datos provenientes de un corpus de juegos colaborativos en parejas, en donde 34 personas interactúan de a través de su voz para lograr posicionar objetos en sus pantallas. Este corpus se encuentra en español argentino y ha sido utilizado en trabajos anteriores para diversas tareas relacionadas al procesamiento de diálogos.

Nuestros resultados indican que los modelos basados en CNNs superan un puntaje f_1 macro-promediada de 0,45 — desempeño comparable con modelos anteriores y en donde los atributos han sido construidos mediante un trabajo más profundo de ingeniería de atributos. Por otra parte, obtuvimos buenos resultados al evaluar nuestro modelo sobre datos no utilizados para el entrenamiento del modelo y sobre datos provenientes de una segunda porción del corpus de juegos en español (f_1 macro-promediada de 0.47), resultados que demuestran el buen poder de generalización de estos modelos. Finalmente, estudiamos las posibilidades de mejora de nuestro modelos al agregar más datos mediante curvas de aprendizaje.

Concluimos que el uso de CNNs sobre espectrogramas constituye una herramienta competitiva y con gran potencial para la predicción de transiciones de turnos en conversaciones, lo cual es de gran importancia tanto para la tarea de análisis automático de conversaciones como para la mejorara del desempeño de los cada vez más utilizados asistentes virtuales.

Palabras claves: Manejo de Turno, Aprendizaje Automático, CNNs, Espectrogramas.

CONVOLUCIONAL NEURAL NETWORKS APPLIED TO THE TURN-TAKING PROBLEM USING SPECTROGRAMS

In any conversation between two individuals, an implicit exchange of signals occur that contributes to fluidity in the conversation. This is the case of the turn-taking cues — prosodic, syntactic or even gestural patterns that indicate events related to the turn management in a conversation and allows the listener to predict how the current turn will proceed.

In the present, tasks related to the detection of these cues are mostly performed with the use of machine learning techniques, more precisely with recurrent neural networks. On the other hand, in different tasks of speech processing, significant improvements have been made with the combination of *convolutional neural networks* (CNNs) applied to *spectrograms* of the acoustic signal. The main goal of this thesis is to research the *turn taking* prediction task using CNNs.

To do that, we used a dataset from a collaborative game in couples, where 34 people interact by voice to position an object on their screens. This corpus is available in Argentine Spanish and has been used in previous works for numerous tasks related to dialogue processing.

Our results show that models based on CNNs achieve a macro-averaged f_1 score of over 0,45 - which is comparable with previous models where features have been built by a more profound work of feature engineering. Meanwhile, we achieved good results evaluating our model over data not used in training and with data from a second part of the Spanish games corpus (*macro-average* f_1 score of 0.47). These results show the capacity of the generalization of these models. Finally, we studied the possibility of improving our models with training with more data by a learning curve.

In conclusion, the use of CNNs on top of spectrograms constitutes a competitive tool with great potential for turn-taking prediction in conversation, which is of great importance both for automatic analysis of conversations and for the improvement of virtual assistants.

Keywords: Turn-taking, Machine Learning, CNNs, Spectrograms.

AGRADECIMIENTOS

En primer lugar, quiero agradecer a mis directores, Pablo y Pablo, por su dedicación a la tarea que sobrepasó lo esperado y por presionar para que avance cuando más lo necesitaba.

A los jurados, por dedicarle un tiempo a este trabajo en estos complicados tiempos.

A mis padres, gracias por su cariño y su ejemplo de esfuerzo y dedicación. A mi hermana, Sol, por ayudarme a aguantar dichos ejemplos y por estar, siempre.

A mis amigos, Gus y Fran, por estar desde el comienzo y porque si ellos no estaría en esta carrera. A Juli, por los incontables trabajos prácticos juntos y las materias sufridas hasta las 10 de la noche de un Viernes. A Murga, Gastón, Espi, Rouli, Fosqui, Tomo y Nachito por las Dignis compartidas, las escaladas, los sukiyakis, los scalaguirres, por cada plan increíble y cada cursada compartida, porque la carrera no la hice solo.

A Gus (Cairo) y Guido (R, porque no voy a escribir todo tu apellido), por convertir una de las peores cursadas en un cago de risa. Y la mucha gente con la que disfruté cursar, desde AlgoI hasta Ingeniería.

A mis amigues del *Pelle*, que si me pongo a nombrarlos se duplican las páginas del trabajo. Por los juegos, las salidas, los grupos de Whatsapp, por *Pelle XVI* que me levantaba el finde luego de estar con un TP, por los asados, por los hermosos viajes compartidos y las anécdotas interminables. Por ser más que amigos, una familia.

A les chiques de Nebraska, que también son parte de esto. Por hacer que tres meses se recuerden para toda la vida.

A Ricardo, por todo esto, y mucho más.

Y finalmente, a la *UBA*. Porque todo lo anterior no habría sido posible sin esta institución. Porque la educación no es solo lo que se ve en el aula. Por cada asamblea y toma en el *Pelle* que me enseñaron que vale la pena luchar por lo que uno cree. Por hacerme tener ganas de ir a clases y encontrarme con la gente en los pasillos. Por la calidad y la dedicación increíble de los grandes docentes que tuve y por el privilegio de permitirme ser uno también.

A mis abuelos,

que siempre me apoyaron en este camino,
aunque nunca supe si entendían lo que hago

Índice general

1. Introducción 1

1.1 Definición del Problema 3

1.2 Estado del arte y problemas relacionados 4

1.3 Aprendizaje Automático 7

1.3.1 Conceptos Generales 7

1.3.2 Redes Neuronales 7

1.4 Espectrogramas como representación de un audio 8

2. Materiales y Métodos 11

2.1 UBA Games Corpus 11

2.2 Aprendizaje Automático 12

2.2.1 Métricas para Modelos de Aprendizaje Automático 12

2.2.2 Entrenamiento por medio de Validación Cruzada 13

2.2.3 Redes Neuronales Convolucionales 14

2.2.4 Redes Neuronales Long-Short Term Memory 15

2.2.5 Redes Neuronales con Capas de Atención 17

2.2.6 Tipos de capas utilizadas 18

3. Desarrollo y Selección de Modelos 21

3.1 Exploración de Posibles Modelos 21

3.1.1 Red Neuronal de Atención 22

3.1.2 Red Neuronal Convolucional de Tres Capas 23

3.1.3 Red Neuronal Convolucional de Cuatro Capas 24

3.1.4 Comparación de arquitecturas convolucionales 25

4. Predicción de Transiciones de Turno 29
4.1 Exploración del Modelo Seleccionado 29
4.2 Comparación con Modelo Externo 34
4.2.1 Definición del Modelo a Comparar 34
4.2.2 Comparación por Grupos 38
4.3 Comparación de resultados en nuevo corpus 41
4.4 Curva de Aprendizaje 43
4.4.1 Preparación del Experimento 43
4.4.2 Resultados 45

5. Conclusiones 48

Referencias 49

1. INTRODUCCIÓN

En una conversación hablada entre dos personas se da un intercambio implícito de señales que contribuyen a una mayor fluidez de la conversación. Esto se postula en [Duncan, 1974], trabajo en el cual se introduce el concepto de *pistas de transición de turno*. Dichas pistas pueden entenderse como señales prosódicas, sintácticas o gestuales que contribuyen a la fluidez de la conversación.

Una numerosa cantidad de trabajos a través de los años sugieren fuertes relaciones entre la presencia de pistas de transición de turno y la asignación de turnos en una conversación [Duncan, 1974, Ford and Thompson, 1996, Ferrer et al., 2002, Wennerstrom and Siegel, 2003, Gravano and Hirschberg, 2011, Hjalmarsson, 2011, Bögels and Torreira, 2015, Ward, 2019, entre otros]. En particular, algunos trabajos [Gravano and Hirschberg, 2011, Ward, 2019, Skantze, 2017, Roddy et al., 2018b] se enfocan especialmente en utilizar dichas pistas con el fin de **predecir** eventos que sucederán posteriormente, como en una pausa de la conversación. Ejemplos de estas situaciones son los *turn endings* (fin de turno), *interrupciones* (colaborativas y no colaborativas), entre otros eventos. Entender y detectar automáticamente las pistas pueden mejorar tanto el análisis automático de conversaciones, como la naturalidad y fluidez de los actuales *asistentes virtuales conversacionales*, cada vez más comunes en la vida cotidiana.

En la actualidad, los problemas relacionados al manejo de turnos son atacados mayormente mediante técnicas de aprendizaje automático. Bajo este paradigma, se analizan grandes cantidades de datos en busca de aprender patrones presentes en la señal de habla o en atributos acústico-prosódicos derivados tales como el tono de voz, la intensidad, características de la calidad de la voz, la velocidad del habla, etc. Estas técnicas tienen por objetivo construir modelos estadísticos que permitan predecir eventos en conversaciones no vistas en el momento de aprendizaje, es decir, generalizar a nuevos datos.

Recientemente han florecido trabajos relacionados al transición de turnos en conversaciones mediante la utilización de redes neuronales recurrentes. Por ejemplo, en los trabajos de [Skantze, 2017] y [Roddy et al., 2018b] los autores utilizan este tipo de arquitecturas para predecir la actividad de habla futura en diálogos, es decir, para cada momento en la conversación, predecir la probabilidad de que tanto el hablante actual como el interlocutor continúen hablando. Estos modelos están especialmente diseñados para aprender representaciones contextuales de series temporales, lo que quiere decir son capaces de recordar eventos pasados en la conversación mediante mecanismos de memoria.

Paralelamente, en el campo de investigación de detección de objetos en imágenes es muy común la utilización de las redes neuronales convolucionales (CNN), las cuales son comúnmente utilizadas en datos con alta relación espacial (como suelen ser las imágenes).

En la misma línea, un fragmento de audio puede ser parcialmente representado como una imagen mediante su *espectrograma* — la concatenación del espectro de una señal

acústica obtenida a lo largo del tiempo mediante un proceso de ventaneo (como puede verse en la Figura 1.1, en la cual mostramos un ejemplo de espectrograma proveniente del sonido de una persona diciendo "Hola a todos") —. Pese a que los espectrogramas no contienen toda la información presente en la señal acústica original, mantiene ciertas propiedades que los hacen útiles para tareas del procesamiento del habla. En esta dirección es cada vez más común encontrar trabajos que utilizan CNNs en el campo del procesamiento de sonidos. Ver por ejemplo trabajos como [Phan et al., 2019], en donde utilizan modelos con capas convolucionales para la tarea de clasificación de escenas acústicas, o [Coimbra de Andrade et al., 2018] en donde utilizan esta misma técnica para realizar reconocimiento de comandos de voz. Además, ambos trabajos mencionados presentan modelos donde se utilizan capas de *atención* [Vaswani et al., 2017], que están ganando terreno en una cantidad de problemas.

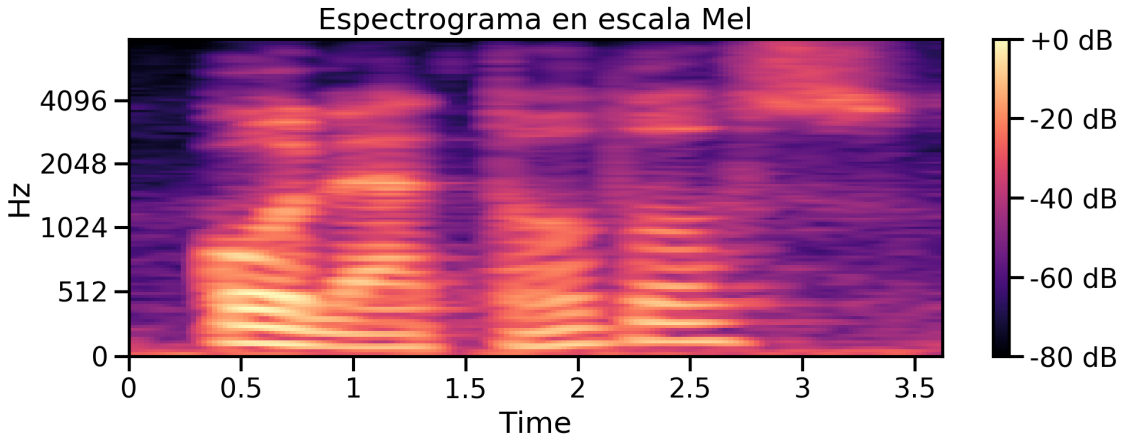


Fig. 1.1: Espectrograma de una persona diciendo "Hola a todos"

En el presente trabajo, estamos interesados en unir el mundo de redes neuronales convolucionales con el dominio de predicción de transiciones de turno, dado que estos modelos han sido poco utilizados para esta tarea en combinación con los espectrogramas. Para esto, probamos distintos modelos basados en los trabajos mencionados anteriormente, tanto convolucionales puros como convolucionales con capas de atención. En particular, nos interesará **predecir qué tipo de transición de turno** ocurrirá con la información disponible hasta el momento de una pausa en una conversación. Las clases a predecir serán (a) **Switch**: cuando el orador terminó de hablar y comienza a hablar la otra persona; (b) **Hold**: cuando el orador continuará hablando luego de una pausa; y por último (c) **Backchannel**: cuando el orador se detiene por un momento y el interlocutor produce una señal de atención o seguimiento de la charla como por ejemplo 'aha', 'claro', 'sí', 'okey'.

Para esta tarea, utilizamos la versión en español del "Columbia Objects Games Corpus"[1] — un juego colaborativo en parejas en donde los participantes se comunican naturalmente con el objetivo de posicionar objetos en sus pantallas y de esta manera ganar puntos [Gravano and Hirschberg, 2011] —. Para la versión en español, se recolectaron

[1] `http://www.cs.columbia.edu/speech/games-corpus/`

datos utilizando la misma metodología que para la base de datos en inglés. Esta tarea fue realizada por el grupo de *Procesamiento del Habla del Departamento de Computación* (FCEyN, UBA)[2]. Utilizaremos esta base de datos ya que contiene las anotaciones necesarias para tareas relacionadas al manejo de turnos y además permite la comparación de nuestros resultados con otros trabajos.

1.1 Definición del Problema

El problema que estudiamos en esta tesis puede definirse como: dada una pausa o silencio en una conversación entre dos personas, se quiere predecir qué tipo de **transición de turno** se dará a continuación en base a información inmediatamente anterior al evento.

Para poder definir el concepto de Turno, primero debemos definir lo que se conoce como *unidades interpausales* (o **IPU**s por sus siglas en inglés). Estas son segmentos continuos de habla sin silencios de duración mayor a 100 ms. Un **Turno** en una conversación se define como la secuencia maximal de unidades IPUs entre las cuales no se producen alocuciones del interlocutor. Es decir, en las pausas que conectan a dos IPUs consecutivas no hay rastros de alocuciones de la otra persona en la conversación. Para más detalles respecto a estas definiciones y al proceso de etiquetado de las mismas, consultar [Gravano and Hirschberg, 2011]. Luego, estudiamos algunos de los tipos de transiciones de turnos presentados en ese mismo trabajo. Estas son:

- **HOLD (H)**: Una transición entre dos IPUs dentro de un mismo turno (es decir, una transición intra-turno) en la cual el orador corta su discurso con un silencio y continua hablando.
- **SWITCH (S)**: Cuando el orador termina su turno, se produce un silencio, y luego el interlocutor comienza a hablar.
- **BACKCHANNEL (BC)**: Cuando el orador deja de hablar por un momento, el oyente emite una corta respuesta que indica su atención, del estilo *entiendo lo que decís* o *sigo escuchando*, como pueden ser 'aha', 'claro', 'sí', 'okey', 'mm-hh', etc. y luego el orador original continua hablando.

De aquí en adelante en este trabajo, diremos que una IPU pertenece a una categoría de transición de turno si la transición inmediatamente posterior a la IPU pertenece a dicha categoría.

En la Figura 1.2 puede verse un ejemplo de una conversación y como se darían las transiciones explicadas anteriormente, en este caso, podemos ver un ejemplo de cada transición utilizada en este trabajo. En base a lo mencionado anteriormente, definimos que una IPU es clasificada como un *SWITCH* si precede a una transición de tal tipo (caso del IPU1, representada por el primer segmento oscuro). Similarmente, denominamos una IPU como *HOLD* si la transición posterior es clasificada como tal (como se puede ver en la IPU2).

[2] habla.dc.uba.ar

Finalmente, denominamos *BACKCHANNEL* a la IPU que precede a *la primer transición* correspondiente a un *BACKCHANNEL* (como se puede ver en la IPU3). Es importante notar que se necesitan las IPUs posteriores (IPU4 e IPU5) para que una transición sea clasificada como *BACKCHANNEL*, por lo que la secuencia IP3-IP4-IP5 puede ser considerada un *BACKCHANNEL* en su totalidad, pero en este trabajo utilizaremos solo la IPU3 para la tarea.

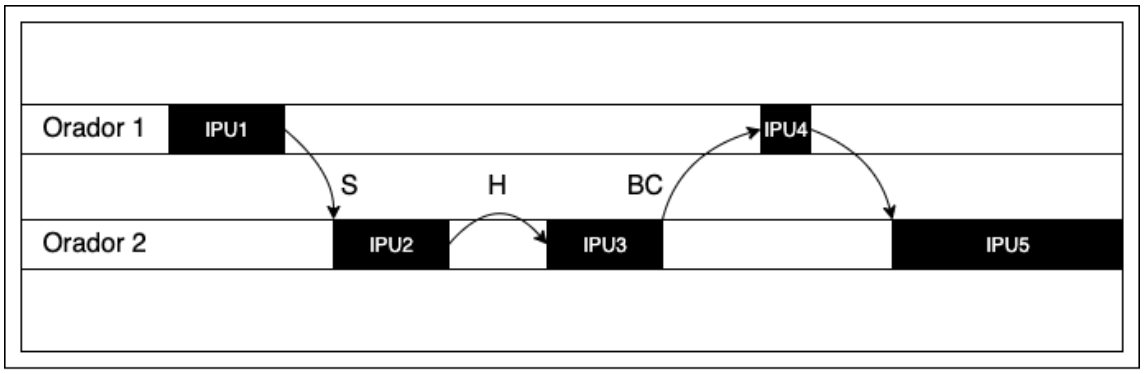


Fig. 1.2: Figura ilustrando las distintas transiciones mencionadas anteriormente en una conversación entre dos interlocutores

Es importante notar que estos son solo algunos de los tipos de transiciones que se estudian en [Gravano and Hirschberg, 2011]. Por ejemplo, estamos estamos ignorando casos con *solapamiento*, es decir, cuando dos personas se superponen durante un breve momento u otras transiciones, como las *interrupciones* o los *comienzos en falso*.

1.2 Estado del arte y problemas relacionados

A lo largo de una conversación entre dos personas las transiciones mencionadas anteriormente suceden de manera natural y fluida. Más aún, [Sacks et al., 1974] reporta que el intervalo de silencio entre que una persona termina de hablar y el oyente comienza es lo suficientemente cercano al tiempo de respuesta humano ante un estímulo cualquiera. Esto respalda la *teoría de la proyección* que postula que una comunicación eficiente necesita la formación de *predicciones* con respecto a cuando se va a realizar una transición de turno. De esta manera surge la necesidad de estudiar las llamadas **pistas de transición de turno**, es decir, cualquier indicio por parte del orador (consciente o inconsciente) que de el indicio que realizará una transición de turno dentro de poco tiempo.

En base a esta definición podemos mencionar una numerosa cantidad de trabajos en la cual se estudian las distintas pistas y su relación con transiciones de turno específicas. Por ejemplo, en el trabajo [Brusco et al., 2017], los autores exploran las diferencias entre las pistas de cambio de turno entre el inglés estadounidense y el español argentino, mediante una serie de experimentos utilizando modelos de aprendizaje automático. Sus resultados sugieren que, en el idioma español, mantener la entonación al final de la frase (en vez

elevarla como es común en el caso de algunos tipos de preguntas) es un buen indicador de que la IPU será precedida por una transición de tipo *HOLD* (indicio que también encuentran en inglés). Por otro lado, el trabajo también explica como en español sucede que la presencia de una entonación decreciente correlaciona en gran medida a las transiciones de tipo *SWITCH*. De la misma manera, en el trabajo de [Skantze, 2017] se menciona el uso expresiones para rellenar una pausa como "uhm" para indicar el mantenimiento del turno (*HOLD*) en el idioma inglés.

Similarmente, en [Gravano and Hirschberg, 2011] los autores realizan un amplio estudio de las pistas de transiciones de turno en el inglés americano y analizan su importancia al preceder a las transiciones de turno. Para esto realizan una serie de experimentos con el objetivo de establecer las pistas de transiciones de turno que obtengan una mejor correlación con la transición realizada. A partir de estas definiciones de pistas los autores analizan la combinación de las mismas y como correlaciona con las transiciones observadas y concluyen que la combinación de las pistas de transiciones de turno aumentan la probabilidad de que suceda posteriormente dicha transición.

Por otro lado, existen una gran cantidad de trabajos atacando problemas relacionados a la transición de turnos en conversaciones, aunque no todos definen el problema de la misma manera, y cómo se define el problema condiciona a las técnicas a utilizar en los trabajos. Las definiciones explicadas en la Sección 1.1 son solo una forma de dividir el problema.

Por ejemplo, en el trabajo de [Skantze, 2017] podemos ver como es posible separarse de modelos basados en IPUs y generan modelos de aprendizaje automático del tipo LSTM para que prediga en cada momento la posibilidad de que el orador esté hablando. Además, combina información acústica (propiedades del audio como el tono o la fluctuación del audio) con etiquetas de *Part-Of-Speech* (etiquetas gramaticales que indican qué tipo de palabra se esta diciendo).

Siguiendo con el mismo autor, podemos mencionar otro trabajo más reciente en [Roddy et al., 2018a]. En este trabajo podemos ver que sigue en la misma línea que el anterior y profundiza en ciertos aspectos, ya que utiliza el mismo modelo para realizar el mismo tipo de predicciones continuas. Además, divide en tres el tipo de información a ingresar en el modelo: Lingüísticas, Acústicas y Fonéticas. Las acústicas y lingüísticas anteriores se mantienen y se agregan las palabras que se están diciendo (100 ms después de haberlas dicho para simular un escenario real en el que el interlocutor reacciona a la palabra emitida). Además, también incluyen información fonética, que obtienen desde la primera parte de un modelo de reconocimiento automático del habla.

Los mismos autores presentan en [Roddy et al., 2018b] una nueva forma de extraer información de los audios de las conversaciones. Al realizar extracciones en distintas ventanas de tiempo de manera continua logran obtener una mejor percepción del contexto en distintos plazos (dependiendo de la extensión de la ventana de tiempo). Lo innovador de esta técnica es que el modelo del trabajo procesa la información con distintos componentes dependiendo de la ventana de tiempo que se requiere para procesarla. De esta manera, pueden separar la información lingüística y acústica y procesarlas en distintas ventanas de tiempo, según se requiera.

Pasando a otros autores, vemos que en [Maier et al., 2017] también se utiliza modelos de LSTM de manera similar, con el objetivo de detección binaria de fin de turno del orador. Para esta tarea utilizan información lingüística y acústica, de la misma manera que los trabajos anteriores. Utilizando esta tarea, comparan la utilidad de la información lingüística y acústica para ver qué modelos obtienen mejores resultados, y se comparan con modelos de detección de fin de turno a partir de un umbral de silencio de cierta cantidad de tiempo (muy utilizado en sistemas de asistentes virtuales).

Por otro lado, en [Hara et al., 2018] utilizan, también, un modelo de tipo LSTM para predecir de manera conjunta tres tipos distintos de eventos binarios: SWITCH vs HOLD (aquí lo llaman KEEP), BC vs NOT (es decir, que no es un BC) y FILLER vs NOT (es decir, que no es un FILLER). El trabajo define un FILLER como el momento en el que el orador rellena un silencio incómodo (como dice explícitamente el trabajo) con una palabra como por ejemplo "*uh*" o "*so*" en inglés. Lo novedoso de este trabajo es que presenta un modelo que logra predecir los tres eventos binarios a la vez, basándose en modelos preexistentes de LSTM, y utilizando información prosódica principalmente.

Hasta el momento, pocos trabajos han realizado tareas similares a transición de turnos utilizando redes neuronales convolucionales, con la excepción de [Liu et al., 2019]. En dicho trabajo se utilizan una combinación de información léxica con acústica, esta última tomada de una manera muy similar a la que utilizamos en este trabajo, mediante coeficientes espectrales en escala mel. Los autores de este trabajo utilizan un set de datos en el cual distintos individuos interactúan con un robot que imita expresiones faciales humanas controlado por un operador en otra sala (protocolo Mago de Oz), las conversaciones se dividen en cuatro situaciones que tienen que ser actuadas por el operador y el individuo. Este trabajo utiliza dos tipos de transiciones, *dentro de turno* y *fin de turno*, dichas etiquetas equivalen a lo que en este trabajo denominamos HOLD y SWITCH, respectivamente. El trabajo mejora los resultados presentados como comparación, dos modelos basados en LSTMs. Esto resulta una buena referencia para reafirmar la posibilidad de seguir utilizando este tipo de redes en combinación con información acústica de espectros en escala mel para trabajos de transiciones de turno y similares.

En conclusión, hay evidencia de distintas propiedades de las IPUs que preceden a pausas en la conversación respecto de su rol en la transición de turno que sucederá inmediatamente después. Si bien en su mayoría se han mencionado pistas *acústico-prosódicas*, existen otros tipos de pistas, como lingüísticas o gestuales, las cuales se suelen utilizar en otros trabajos en el estado del arte. Con respecto a qué técnicas se utilizan para estos problemas, los trabajos anteriormente mencionados utilizan casi unánimemente modelos de LSTM para resolver estos problemas (a excepción de trabajos como el de [Brusco et al., 2017] o [Liu et al., 2019]). Por otro lado, vemos una división de trabajos con respecto a como resolver los problemas, dividiéndose entre los basados en IPUs en el cual se trata de predecir el tipo de transición al próximo turno y los trabajos de predicción continua, los cuales presentan, a cada momento una probabilidad de que se presente un cambio de turno.

1.3 Aprendizaje Automático

Para este trabajo se utilizaron técnicas de aprendizaje automático, a continuación se introduce una breve explicación de tema y luego profundizaremos en las técnicas específicas utilizadas en la Sección 2.2.

1.3.1 Conceptos Generales

El objetivo de un clasificador de aprendizaje automático es el de aprender a realizar una tarea en base a la *experiencia*, es decir, información de la tarea asociada a su resultado deseable. Este tipo de aprendizaje, el cual utilizamos en el trabajo, se denomina *aprendizaje supervisado*, en el cual el clasificador aprende a partir de datos ya etiquetados (asociación entre los datos iniciales con los resultados esperados) para poder clasificar correctamente nuevos datos. Para más detalles respecto a estas definiciones consultar [Bishop, 2006].

1.3.2 Redes Neuronales

Dentro del área del aprendizaje automático, existen diversos tipos de algoritmos para entrenar un modelo. Un subconjunto de estos son las redes neuronales, que han cobrado mucha fuerza en los últimos años, obteniendo resultados superiores al aprendizaje automático clásico en problemas como detección de objetos en imágenes, detección de escenas acústicas o detección de enfermedades.

Entre los beneficios de las redes neuronales se puede destacar su gran capacidad de generalización, siempre y cuando se tengan la cantidad suficiente de datos. Esto permite que se apliquen a una gran cantidad de tareas. Por otro lado, podemos mencionar que el costo computacional se suele concentrar principalmente en su entrenamiento, y una vez entrenada la red es de gran velocidad a la hora de la clasificación. La proliferación en los últimos años de utilizar GPUs para el entrenamiento y hardware especializado hizo posible la paralelización del entrenamiento, así reduciendo sus tiempo de entrenamiento.

En esta Sección introducimos los conceptos básicos del tema, mientras que en el Capítulo 2 profundizamos sobre las técnicas utilizadas, estas siendo *Redes Neuronales Convolucionales* y *Redes Neuronales con Capas de Atención* explicadas en la Sección 2.2.3 y 2.2.5 y utilizadas en la Sección 3.1.

Componente base: El perceptrón simple

Como su nombre lo indica, el concepto de redes neuronales fue tomado de la biología, más específicamente del funcionamiento de una neurona. Inicialmente el concepto más sencillo (y único que mantiene su analogía con una neurona biológica) es el llamado **perceptrón simple**. Este se compone de una entrada $x_1 \dots x_n$ con $x_i \in \mathbb{R}$ multiplicados por sus pesos $w_1 \dots w_n$ con $w_i \in \mathbb{R}$. De esta manera, a cada una de las entradas x_i se le puede

dar una importancia determinada según nos aporte a la función, el cual se lo denomina *peso*.

La salida de un perceptrón simple es la aplicación de una *función de activación* a la suma de los valores explicados anteriormente. Una función de activación es una función que retorna un valor final de un perceptrón simple, dicha función es una transformación no lineal que restringe los valores a un rango específico, según la función utilizada.

Existen diversos ejemplos de función de activación, la más clásica es la función sigmoidea, pero a lo largo del tiempo se fueron separando de la teoría y se descubrieron nuevas funciones que mejoraban el funcionamiento práctico de las redes neuronales, separándose de su analogía biológica. Varias de ellas son explicadas en la Sección 2.2.6.

Perceptrón multicapa

Dada una neurona como la presentada anteriormente, es posible conectar distintas neuronas entre sí, donde el input de una proviene del output de la otra, así formando un grafo o *red neuronal*. Usualmente estas redes suelen ser del tipo *feedforward* donde se puede separar por niveles o *capas* y el resultado de cada capa es la entrada de la siguiente. La totalidad de estas capas, cómo se conforman y cómo se conectan se denomina *arquitectura* de una red neuronal.

Un perceptrón múltiple consta de varias capas de neuronas, donde las capas internas son llamadas *ocultas*. Además, todas las neuronas están conectadas entre sí, con todas las salidas de una capa yendo a todas las entradas de la siguiente. Esto se denomina *capa densa* y, si bien es una representación clásica de una red neuronal, no es la única en utilización hoy en día.

1.4 Espectrogramas como representación de un audio

Digitalmente, un audio es representado como una serie de valores, llamados *samples* o muestras, que son obtenidos mediante el muestreo de las ondas sonoras al momento de la grabación. Un muestreo de una señal continua tiene asociado un *sample rate* o frecuencia de muestreo, es decir, la cantidad de valores por segundo que fueron tomados.

Dado un audio digital, se denomina *espectro* a la descomposición en las frecuencias que la forman mediante la aplicación de transformadas de Fourier a una determinada ventana de tiempo como se puede ver en el ejemplo de la Figura 1.3, siendo el eje X las frecuencias que aparecen en la ventana de tiempo y el eje Y la amplitud de la frecuencia.

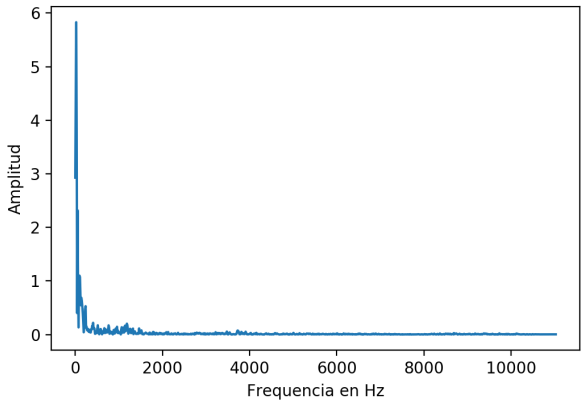


Fig. 1.3: Gráfico representando la descomposición en frecuencias de una ventana tiempo

Para aplicar esta transformación a todo un audio se aplica estableciendo una ventana de tiempo y un desplazamiento, de esta forma se realiza la descomposición en frecuencias en dicha ventana y se corre la ventana en una cantidad de samples establecida para el desplazamiento. De esta manera podemos generar una matriz de valores, donde cada ventana es representada por las frecuencias que se escuchan en dicha ventana. Sin embargo, al representar esta matriz en Hz, es muy poco distinguible unas frecuencias entre otras, debido a la escala que estamos viendo ya que no se pueden apreciar correctamente el rango de la voz humana, un ejemplo de esto se puede ver en la Figura 1.4, donde se aplica la escala logarítmica. Como se puede ver en la Figura se realzan las frecuencias bajas, las cuales no son tan escuchadas por el oído humano. Es por esto que se aplica la *escala mel* en la que se puede apreciar de una manera más clara estas diferencias, ya que se define esta escala de manera que los oyentes escuchen tonos equi-espaciados. Como podemos ver en la Figura 1.5 la representación es mucho más visual.

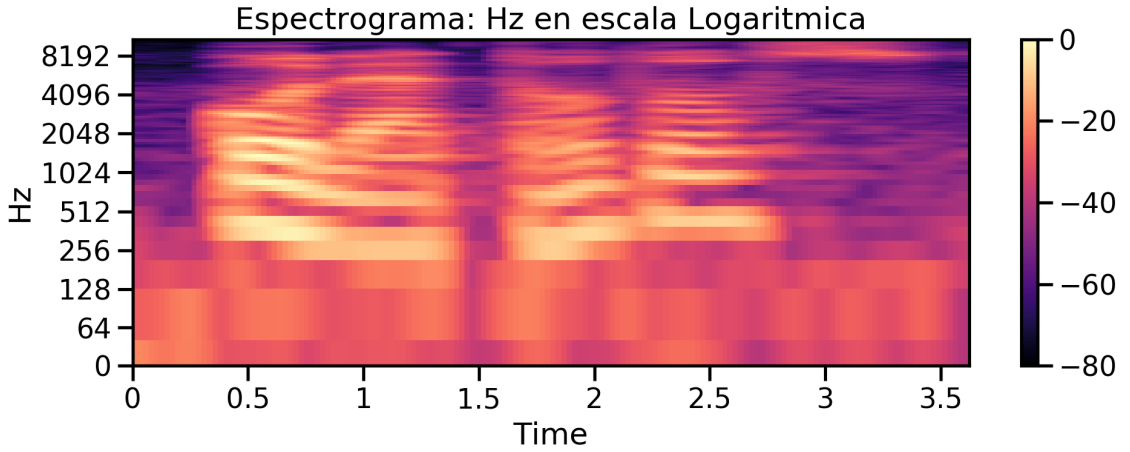


Fig. 1.4: Espectrograma en escala logarítmica de una persona diciendo “Hola a todos”

De esta manera podemos ver un audio digital representado como una matriz de intensidades en cada una de las frecuencias que la componen, separando el audio en ventanas de tiempo y aplicando una transformada de Fourier a cada una.

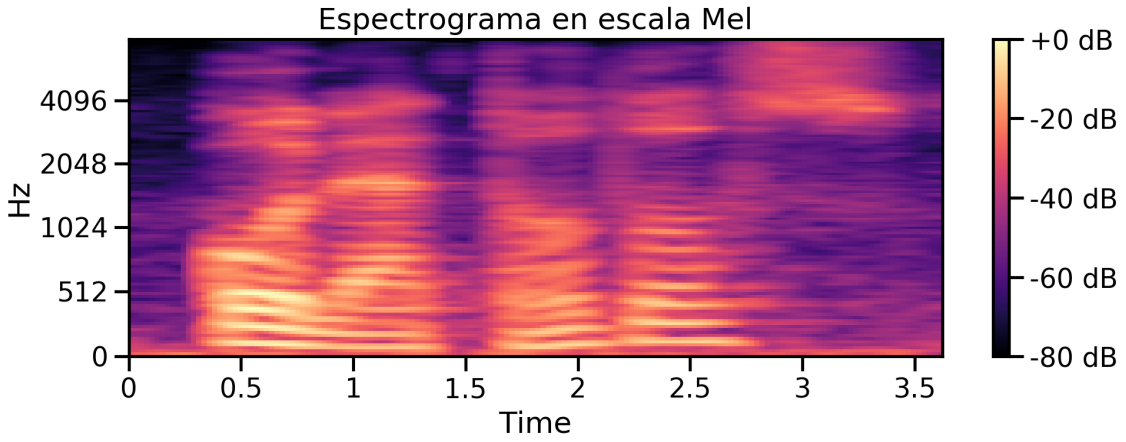


Fig. 1.5: Espectrograma en escala mel de una persona diciendo "Hola a todos"

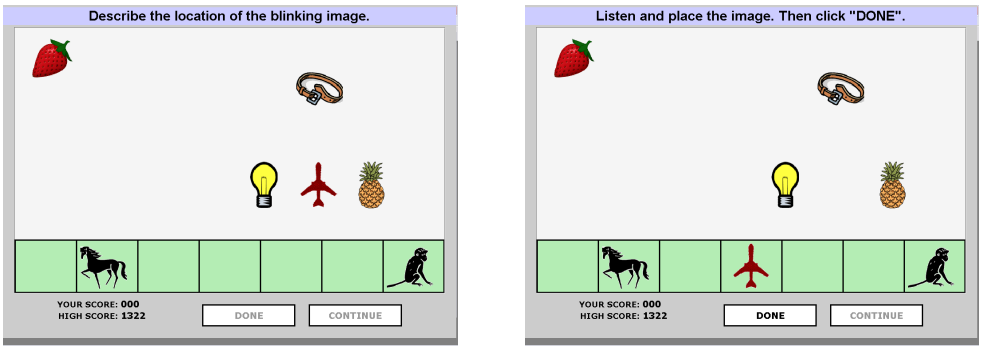
2. MATERIALES Y MÉTODOS

2.1 UBA Games Corpus

Para este trabajo, utilizamos el llamado *UBA Games Corpus* — una base de datos en español argentino que contiene grabaciones de voz de sujetos participando en *el Juego de Objetos* [Gravano and Hirschberg, 2011], un juego colaborativo en el cual pares de sujetos interactúan, de manera hablada y sin contacto visual, para lograr ubicar objetos en su pantalla. La recolección fue realizada en dos lotes. El *Lote A* fue recolectado durante el año 2014 consistió en 7 sesiones en las cuales participaron 14 sujetos (7 hombres, 7 mujeres). El *Lote B* fue recolectado durante el 2016 con registro simultaneo de electroencefalograma (EEG) consta de un total de 10 sesiones entre 20 sujetos (10 hombres, 10 mujeres). El *Lote A* fue utilizado como datos de entrenamiento para todos los experimentos realizados, mientras que el *Lote B* lo utilizamos en el experimento de la Sección 4.3

Los participantes recibieron una pequeña suma de dinero fija y luego un adicional según el puntaje final obtenido.

Cada sujeto se encontraba frente a una computadora, separado de su pareja mediante una cortina opaca, de tal manera que todas las comunicaciones fuesen a través de la voz.



(a) Pantalla del jugador 1, donde tiene como tarea indicar la posición del avión rojo, que estaría parpadeando

(b) Pantalla del jugador 2

Fig. 2.1: Pantallas de los jugadores en el segundo juego del experimento

En el juego de objetos visto en la Figura 2.1, se presentan varias imágenes distribuidas en la pantalla de los jugadores, el sujeto 1 (el *Descriptor*) tiene un objeto de más (el *objetivo*), mientras que el sujeto 2 (el *Seguidor*) tiene ese mismo objeto en la parte inferior de la pantalla. El juego consiste en que el Descriptor describa la posición del objetivo al Seguidor, que tiene que posicionarlo en la pantalla de la mejor manera posible en relación a los demás objetos. Se entregan de 1 a 100 puntos en base a qué tan bien posicionado está

el objeto. El juego consiste de 14 turnos, en los cuales los jugadores se intercambiaban los roles por turno.

A partir del audio de los sujetos, se extrajeron IPUs y cada transición inmediatamente posterior fue anotada como se describió en la Sección 1.1. Vemos en la tabla 2.1 la cantidad de IPUs de cada una de las etiquetas que se anotaron en base a este corpus, para cada lote.

Etiqueta	**Lote A**	**Lote B**
HOLD	5299	4057
SWITCH	1935	2462
BACKCHANNEL	842	711
Total	**8076**	**7230**

Tab. 2.1: Tabla de cantidades de IPUs divididas en sus etiquetas utilizadas en el trabajo

2.2 Aprendizaje Automático

Con el objetivo de realizar una distinción entre las distintas categorías a clasificar las IPUs corpus de datos, se utilizaron técnicas de aprendizaje automático. A continuación explicaremos algunos conceptos y técnicas utilizadas en el trabajo, que nos sirvieron para obtener los mejores resultados.

2.2.1 Métricas para Modelos de Aprendizaje Automático

Para este trabajo utilizaremos la métrica *macro* F_1 ya que nos permite tener el cuenta el promedio de todos los valores F_1 de cada clase y ser promediadas todas por igual.

$$macroF_1 = \frac{F_1^H + F_1^{BC} + F_1^S}{3} \tag{2.1}$$

Donde cada submétrica F_1^i es calculada de la misma manera y por separado para cada una de las clases:

$$F_1^i = 2 \times \frac{precision_i \times recall_i}{precision_i + recall_i} \tag{2.2}$$

Se utiliza la métrica F_1 ya que con ella se obtiene una combinación de las métricas de *Precisión* y *Recall*. Con el objetivo de definir dichas métricas más formalmente, se utiliza *Precisión*, que es definida como:

$$precision_i = \frac{TP_i}{TP_i + FP_i} \tag{2.3}$$

y Recall como:

$$recall_i = \frac{TP_i}{TP_i + FN_i} \tag{2.4}$$

Siendo:

- TP_i *True Positive* (Verdaderos Positivos): La cantidad de instancias de la clase i clasificadas correctamente como clase i.
- TN_i *True Negative* (Verdaderos Negativos): La cantidad de instancias no pertenecientes a la clase i clasificadas correctamente como no pertenecientes a la clase i.
- FP_i *False Positive* (Falso Positivos): La cantidad de instancias no pertenecientes a la clase i clasificadas *incorrectamente* como clase i.
- FN_i *False Negative* (Falso Negativo): La cantidad de instancias pertenecientes a la clase i clasificadas *incorrectamente* como clase i.

2.2.2 Entrenamiento por medio de Validación Cruzada

Para poder medir el desempeño de un modelo, es común utilizar la técnica de *Validación Cruzada*, en la cual se dividen los datos en k grupos de igual tamaño, y se realizan k entrenamientos para realizar la evaluación. Esta técnica permite tener medidas que no se encuentren sobre-estimadas debido al sobreajuste sobre los datos particulares utilizados para el entrenamiento. En la Figura 2.2 podemos ver un ejemplo de validación cruzada con $k = 5$.

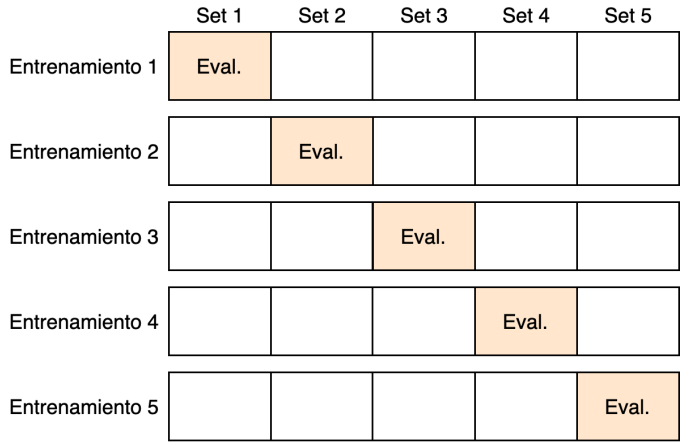


Fig. 2.2: Figura ilustrando un ejemplo de validación cruzada, donde se entrena con los grupos de datos en blanco y se valida con el grupo naranja

De esta forma podemos ver que estamos entrenando k veces, y cada uno de los grupos de datos es evaluado en una de esas k iteraciones, mientras que en las restantes es parte de los datos de entrenamiento. Este sistema nos da una mejor visión del desempeño del modelo en sí mismo, independientemente de los datos específicos que se estén usando para entrenarlo.

En el presente trabajo se realiza validación cruzada con $k = 10$ para una mayor granularidad y obtener más datos de entrenamiento en base al total (cada ronda de entrenamiento tiene una división del 90 % de los datos asignados a entrenamiento y 10 % a evaluación). Además, dada la naturaleza de los datos es importante incluir que se realiza una versión especial de validación cruzada. Con el objetivo de una mejor generalización hacia nuevos sujetos, son separados los audios de cada sujeto y puestos todos en el mismo grupo, de tal forma que ningún sujeto está en dos grupos de datos distintos. Como resultado, ningún sujeto está, en ningún momento, tanto en los datos de evaluación como en los datos de entrenamiento, resultando en una tarea más difícil pero con mejores resultados para un caso general. Esta técnica es llama *Group-K-Fold Cross Validation*.

2.2.3 Redes Neuronales Convolucionales

Como su nombre lo indica, las redes neuronales convolucionales provienen de la aplicación de la operación matemática llamada *convolución*. Dadas dos funciones, se define la convolución como el la integral del producto entre ambas, donde una es desplazada y reflejada.

Aplicando este concepto a matrices, podemos utilizar la definición de una convolución discreta. Dadas dos matrices (que pueden ser consideradas funciones discretas de dos variables $f : \mathbb{N} \times \mathbb{N} \to \mathbb{R}$), podemos definir la operación por medio de la multiplicación matricial de las submatrices de la entrada y un *kernel* o filtro (matriz fija con la que se realizan todas las operaciones).

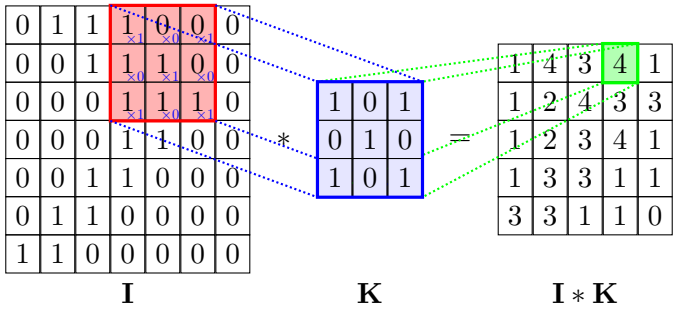


Fig. 2.3: Convolución aplicada a la matriz I con filtro K

Al realizar la traslación de la matriz kernel hay que decidir cuantos *saltos* realiza, en términos de valores de la matriz, a esta cantidad de saltos se le llama *stride*. Por último, podemos notar en la Figura 2.3 como se reduce la dimensión de la matriz resultante. Esto se debe a que la matriz kernel es de dimensión mayor a 1×1. Si se desea que la matriz resultante tenga dimensiones iguales a la original, se puede extender la matriz por medio de un *padding*, o relleno en los límites de la matriz.

Para dos dimensiones, se define formalmente al resultado de una convolución g entre una matriz f y un kernel h de la manera:

$$g(i,j) = \sum_{k,l} f(i-k, j-l)h(k,l) \tag{2.5}$$

En el campo de investigación de clasificación de imágenes es muy común la utilización de estas redes. También son utilizadas en el campo de detección de escenas acústicas, convirtiendo los audios en matrices mediante espectrogramas (explicados en la Sección 1.4).

Volviendo al contexto de las redes neuronales, se utilizan estas capas convolucionales donde se optimizan los filtros con el objetivo de extraer información de la mejor manera para resolver el problema.

La gran ventaja de las capas convolucionales frente a las capas densas es que sus nodos comparten los mismos pesos, lo que termina siendo muy beneficioso a la hora de entrenar, ya que esto resulta en menos parámetros para ajustar en el entrenamiento.

2.2.4 Redes Neuronales Long-Short Term Memory

A lo largo de la Sección 1.2 estuvimos nombrando una variedad de trabajos en los cuales se utiliza la misma base de red neuronal para realizar las arquitecturas completas. Las llamadas *Long-Short Term Memory* o *LSTM*.

Las redes neuronales LSTM son un tipo de red neuronal recurrente. Este tipo de redes se suele utilizar para operar con secuencias de vectores, sea de entrada, salida o su

combinación. De esta manera, podemos ver por qué resultan tan utilizadas en el estado del arte de la disciplina, ya que estamos tratando con secuencias temporales.

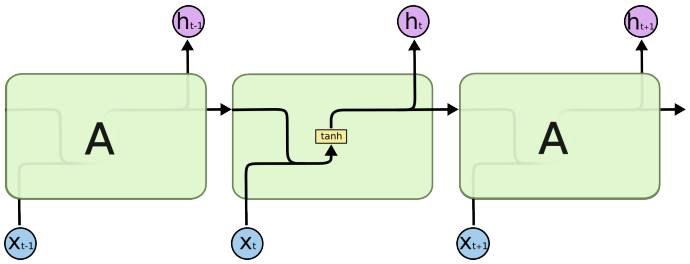


Fig. 2.4: Diagrama de una unidad interna de una red neuronal recurrente simple

Una de las componentes principales de una red neuronal recurrente es el *estado oculto* o *hidden state*, como se puede ver en el diagrama de una red neuronal recurrente en la Figura 2.4, siendo las h_i en violeta los estados ocultos. Este estado permite a la red recurrente poder aprender de su *contexto*. Este estado es retro-alimentado, y se toma como entrada en combinación a la entrada de la red. Esto hace que la salida de la capa sea una combinación de la entrada con h, el vector de estados ocultos de la red, cuyo valor es influenciado por todas las entradas que hubo en el pasado.

El problema de las redes recurrentes clásicas es que tiene un acotado límite en su capacidad para aprender el contexto. Este tipo de problema es muy común en las redes neuronales y se conoce como *vanishing gradient* o desvanecimiento del gradiente, en el cual el gradiente sucesivo se va reduciendo o se *desvanece* hasta llegar a cero, de manera que los pesos no son actualizados.

Las LSTM vienen a resolver este problema en el mundo de las redes recurrentes, ya que debido a su arquitectura son capaces de aprender del contexto general de toda la entrada.

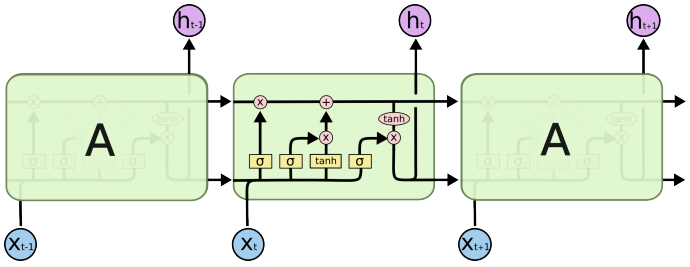


Fig. 2.5: Diagrama de una unidad interna de una red neuronal recurrente de tipo LSTM

La gran diferencia de las LSTM es la inclusión de una *celda de memoria*, como se puede ver en la Figura 2.5. Dicha celda tiene la capacidad de mantener su estado en el tiempo, así resolviendo el problema del vanishing gradient para esta red.

Además, las LSTM constan de *compuertas* con la capacidad de regular el flujo de información de una unidad hacia otra. De esta manera, se puede elegir dejar pasar o no la

información hacia las unidades siguientes.

2.2.5 Redes Neuronales con Capas de Atención

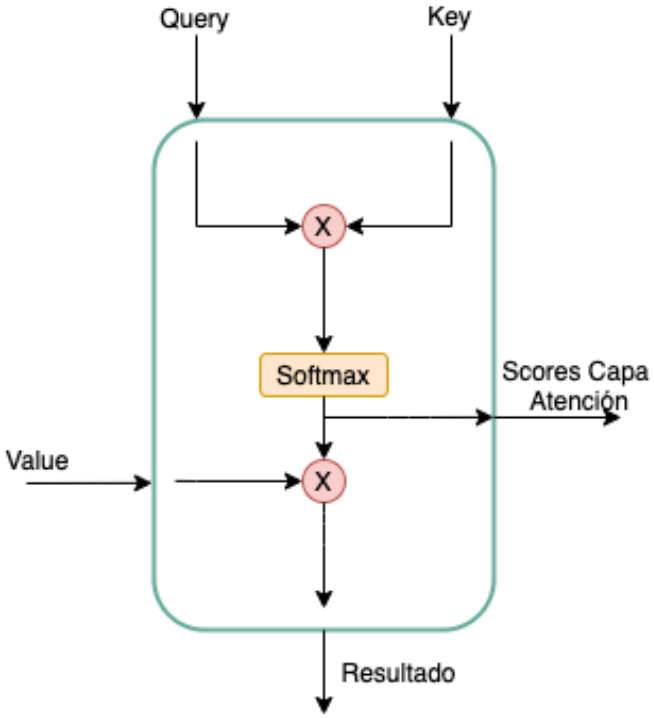


Fig. 2.6: Diagrama de una capa de atención

Otra de las capas utilizadas en este trabajo consiste en la utilización de *capas de atención* para obtener los resultados. Dichas capas consisten de tres entradas: *Query*, *Key* y *Value*. En la Figura 2.6 se puede ver como interactúan las entradas de la capa y cual es el resultado. Como se puede ver, una capa de atención no es más que el producto matricial de la *Query* y la *Key*, mientras que al resultado se le aplica una capa de activación de tipo *Softmax* y luego se le aplica el producto matricial con la entrada denominada *Value*.

La principal característica de la capa de atención es la capacidad de generar una *máscara* la cual es aplicada a la entrada *Value*. Dicha máscara, que es una secuencia resultado de la capa *Softmax* nos indica a qué sectores de la entrada *Value* más nos conviene *prestarle atención*, o que más nos aportan para los resultados, de esta manera se resaltan los valores más importantes de *Value* y se minimizan los de menor importancia.

En el modelo utilizado en este trabajo, la capa de atención es utilizada luego de una capa de tipo LSTM, mencionada anteriormente. *Query* y *Value* son alimentadas por la totalidad de los resultados de la capa LSTM mientras que la *Key* es una de las columnas de dichos resultados, siendo pasada antes por una capa densa, de esta manera vemos como los parámetros optimizables de la capa densa de la *Key* serán los principales encargados

de resaltar los valores de mayor interés de las matrices *Query* y *Value*.

2.2.6 Tipos de capas utilizadas

A lo largo de esta Sección hemos presentado diversas arquitecturas de redes neuronales compuestas por distintas capas. Sin embargo, estas capas no pueden actuar solas, principalmente por problemas clásicos en el aprendizaje automático como el sobreajuste. Además, existen otras capas muy útiles a la hora de entrenar los modelos, que mejoran el desempeño por diversos motivos. Aquí nombraremos las principales capas utilizadas en el modelo y como afectan en las arquitecturas en general.

Normalization

A medida que las redes crecen en profundidad, crece la sensibilidad de los pesos de los nodos a la variación de distribución de los datos ingresados. Sin embargo, si bien es muy común normalizar los datos en su totalidad antes de realizar experimentos, la sensibilidad de los pesos sigue siendo importante a la hora de tratar con un *batch* de entrenamiento (subset de datos utilizados para entrenar a la misma vez). Esto puede causar que la red no logre converger y generalizar correctamente.

Este problema suele ser mitigado utilizando una capa de *Batch Normalization*. Dicha capa realiza una normalización de la totalidad del batch, centrando la media en 0 y la varianza en 1. La aplicación de este tipo de capas mejora la rapidez, eficiencia y estabilidad de una red neuronal.

Dropout

Se denomina capa de *dropout* a una capa en la cual se desactivan ciertas neuronas al azar durante entrenamiento. Este porcentaje de neuronas es un hiperparametro de la capa, la cual sirve para prevenir el sobreajuste. Esto sucede ya que es muy común que una poca cantidad de neuronas concentren todo el aprendizaje, lo cual termina siendo un detrimento para la red, que pierde generalidad.

El gran beneficio de la capa de dropout es su simplicidad de implementación y su bajo costo a la hora de entrenar. Por otro lado, es importante notar que solo se desactivan las neuronas en la fase de entrenamiento, mientras que en el modelo final están todas activadas.

En este trabajo también se utiliza una versión más moderna del dropout llamado *spatial dropout* en la cual se utiliza el mismo concepto de desactivar nodos, con la diferencia de que lo que se termina desactivando son filtros completos de los resultados de las capas convolucionales.

Regularización

Otra de las técnicas utilizadas para la reducción de sobreajuste es la de *Regularización*. Dada una capa de un modelo con los pesos $w_1...w_n$, dichos modelos son optimizados mediante la minimización de la *función de pérdida*. Sin embargo, es posible que con el objetivo de la minimización, los pesos mencionados terminen siendo valores con valor absoluto muy alto, lo cual es la indicación de una mayor complejidad del modelo, lo que indica un posible sobreajuste. Con el objetivo de obtener un modelo más *simple* (es decir, una reducción de los valores absolutos de los pesos) se agrega una penalidad a los pesos $w_1...w_n$ de la siguiente forma:

$$Reg.L2 = \|w\| = w_1^2 + w_2^2 + \ldots + w_{n-1}^2 + w_n^2 \tag{2.6}$$

De esta manera podemos ver que el entrenamiento del modelo favorecería a los resultados con valores más bajos, así manteniendo el modelo más *simple*.

Pooling

A la hora de realizar una red neuronal, es posible que nuestra entrada sea demasiado grande como para procesarla completa en cada cada de la red. Al reducir las dimensiones de nuestros datos estamos ganando en varios aspectos con el objetivo de un mejor desempeño de la red. Por un lado, al reducir las dimensiones ganamos tiempo de cómputo que hubiese sido gastado en el procesamiento de nuestra entrada gigante. Por otro lado, reducimos la cantidad de parámetros de las capas siguientes, por este motivo reduciendo la probabilidad de overfitting.

Esta reducción de dimensiones es realizada mediante capas de *pooling*. Estas capas realizan una función de agregación simple en la entrada, aplicada de alguna manera, ya sea a una vecindad de la matriz de entrada o a su totalidad.

De esta manera podemos filtrar activaciones no deseadas de capas previas, y quedarnos con algún valor representativo. Algunas capas utilizadas comúnmente son:

- Average Pooling: Obtiene el promedio de la submatriz elegida
- Max Pooling: Obtiene el máximo de la submatriz elegida
- Global Average Pooling: Se realiza un Average Pooling de toda la dimensión de cada una de las entradas, quedándose entonces con un valor por input que se le ingresó a la capa.

Capas de Activación

En el trabajo se utilizan distintas capas de activación dependiendo la necesidad de la capa anterior. A continuación se describen las mismas y como funcionan.

Una de las capas de activación más conocidas en la denominada **sigmoidea**, la cual está definida por la siguiente ecuación:

$$sigmoid(v) = \frac{1}{1 + e^{-v}} \tag{2.7}$$

Siendo v la entrada de la capa. Los valores de esta función van de 0 a 1.

Por otro lado tenemos la función de activación de *tangente hiperbólica* o *tanh* como se le suele llamar, cuyo rango va de -1 a 1 y su función es la siguiente:

$$tanh(v) = \frac{2}{1 + e^{-2v}} + 1 \tag{2.8}$$

Luego otra función de activación muy utilizada es la llamada $ReLU$ cuyas siglas corresponden a *Rectified Linear Unit*:

$$ReLU(v) = \begin{cases} 0 & si \quad v < 0 \\ v & si \quad v \geq 0 \end{cases} \tag{2.9}$$

ReLU resulta muy eficiente en términos de tiempo pero es susceptible al problema de la *dying* $ReLU$ en la cual la capa deja de activarse, es decir, devuelve 0 sin importar el valor de la entrada. Es posible que se le asignen pesos a la activación de manera excesiva y los valores terminen en la parte negativa de la activación. Esto sucede ya que el gradiente de la función en su intervalo donde es cero *también* es cero, por lo que no puede modificarse fácilmente.

Por estos motivos se puede usar una modificación de ReLU llamado *Leaky* $ReLU$ en la cual la parte negativa no es cero sin pendiente sino que tiene una pendiente muy baja de valor 0,01.

$$LeakyReLU(v) = \begin{cases} 0{,}01v & si \quad v < 0 \\ v & si \quad v \geq 0 \end{cases} \tag{2.10}$$

3. DESARROLLO Y SELECCIÓN DE MODELOS

En este capítulo presentaremos los modelos utilizados para la investigación. Evaluaremos su desempeño y los compararemos, para elegir así el mejor modelo. Para el presente trabajo utilizamos como punto de partida los modelos preexistentes utilizados para tareas similares. Dichos modelos tienen la propiedad de estar hechos para tareas de clasificación con audio tomando como entrada el espectrograma del mismo.

Como primer paso del desarrollo de nuestro sistema, exploramos la capacidad de estos modelos para extraer información de audios a partir de espectrogramas, utilizando el dataset de *Google Speech Commands* [Warden, 2018]. Los resultados preliminares de este experimento indicaron que los modelos eran capaces de realizar la tarea sin muchas modificaciones, por lo tanto resultan viables como posibles modelos para la tarea de este trabajo.

En segundo lugar, una vez definidos los posibles modelos a utilizar nos propusimos adaptarlos a la tarea de *predicción de transiciones de turno*. El objetivo de esta tarea es: Dados los espectrogramas de las IPUs previas a las transiciones de turno, se desea predecir qué tipo de transición ocurrirá a continuación. Para esto preparamos el siguiente experimento que consiste en realizar validación cruzada del modelo con 10 grupos de usuarios, de tal manera que los audios de un mismo usuario no estén en dos grupos distintos. La división de grupos por usuario fue realizada de esta manera con el objetivo de que el modelo no sea capaz de extraer características particulares de los individuos que tengan audios tanto en el set de entrenamiento como en el de evaluación.

3.1 Exploración de Posibles Modelos

Para el presente trabajo exploramos tres posibles arquitecturas. Inicialmente tomamos como base los modelos utilizados en [Coimbra de Andrade et al., 2018] y un modelo extra basado en redes neuronales convolucionales, los cuales presentamos a continuación.

3.1.1 Red Neuronal de Atención

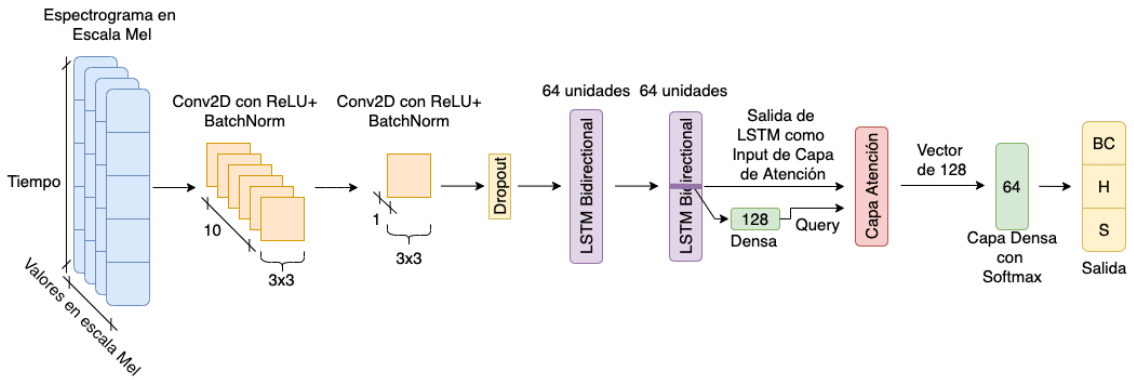


Fig. 3.1: Diagrama de la arquitectura de la red neuronal de atención

Podemos en la Figura 3.1 la arquitectura utilizada para el modelo basado en capas de Atención, tomado del trabajo en [Coimbra de Andrade et al., 2018]. Luego del input, consta de 5 capas, dos convolucionales, luego dos LSTM's y luego la salida de la LSTM es tomada como entrada en la capa de Atención.

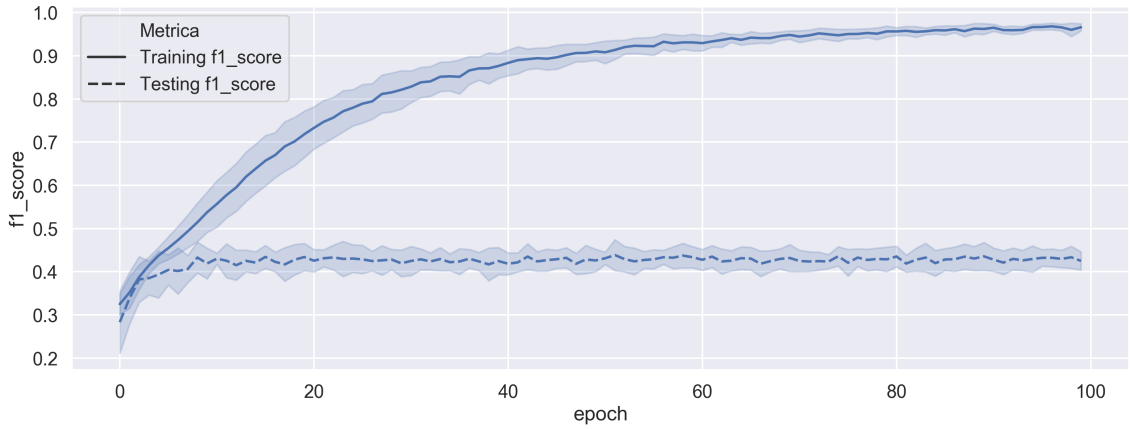


Fig. 3.2: Evolución de la métrica Score F_1 del modelo de atención, promediada entre los 10 folds a medida que avanzan los epochs.

En la Figura 3.2 vemos la métrica Score F_1, tanto en entrenamiento como en validación, siendo la sombra el intervalo de confianza y las lineas el promedio de cada evaluación de la partición de los datos por medio de validación cruzada. Podemos ver que el modelo de atención posee una divergencia entre los resultados de la métrica Score F_1 en validación y en entrenamiento, por lo que estamos viendo un caso de *sobreajuste*, ya que vemos como el modelo sigue mejorando sus resultados en los datos de entrenamiento pero no sucede lo

mismo en el set de validación, que obtiene iguales resultados sin importar el epoch en el que se encuentre.

Luego de varios intentos de mejora no encontramos mejores resultados para un modelo basado en atención, por lo que desistimos en seguir por este camino.

3.1.2 Red Neuronal Convolucional de Tres Capas

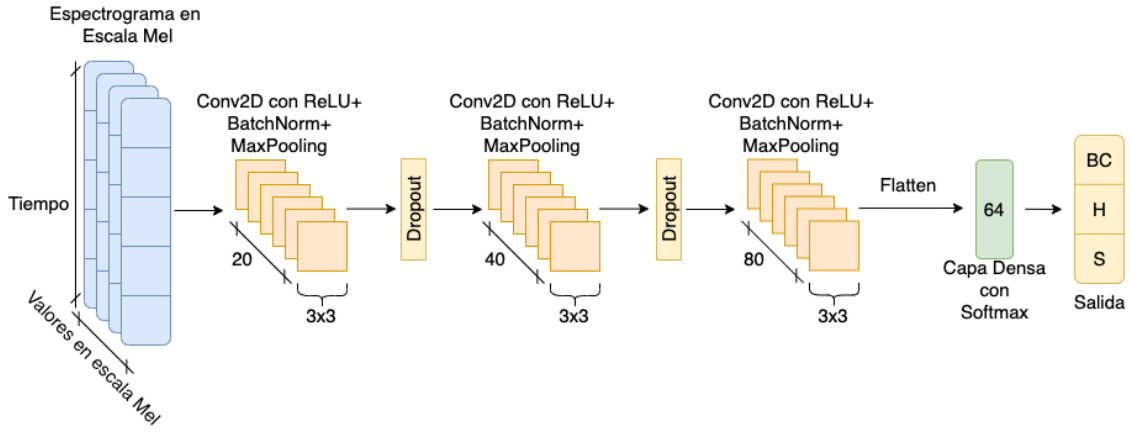


Fig. 3.3: Diagrama de la arquitectura de la red neuronal convolucional básica

En la Figura 3.3 podemos ver la representación de la arquitectura convolucional utilizada para esta sección, a la que *denominaremos* ***Modelo Base de Tres Capas***, originalmente utilizada en [Coimbra de Andrade et al., 2018]. Consta de dos capas convolucionales con Batch Normalization, Max Pooling y Dropout y una capa final sin Dropout. La cantidad de filtro de las capas convolucionales aumentan en cantidad para poder extraer features más finas a medida que la entrada es procesada. Luego la capa *Flatten aplana* el resultado y por último una capa densa con activación de tipo Softmax para obtener la clasificación final.

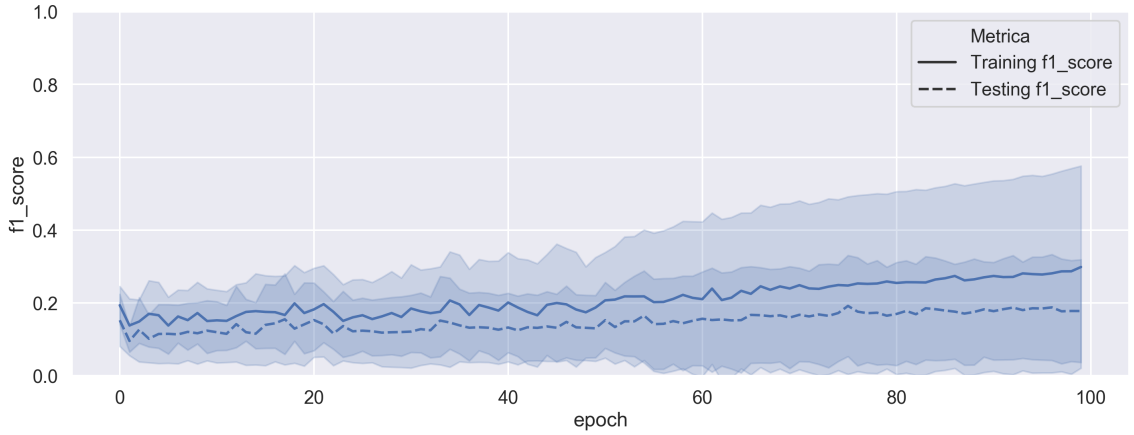


Fig. 3.4: Evolución de la métrica Score F_1 del modelo convolucional básico, promediada entre los 10 folds a medida que avanzan los epochs.

Los resultados de la red convolucional de tres capas terminan siendo considerablemente peores a los de la red de atención, como podemos ver en la Figura 3.4. A diferencia de la red de Atención vemos que no logra ajustarse a los datos de entrenamiento (cosa que si sucede en el entrenamiento de dicha red ya que el Score F_1 alcanzaba casi 1). No solo eso sino que presenta una gran varianza entre los distintos sets de entrenamiento de la validación cruzada. Por otro lado vemos que los resultados en validación apenas mejoran con el entrenamiento, y alcanzan peores resultados que la red anterior.

3.1.3 Red Neuronal Convolucional de Cuatro Capas

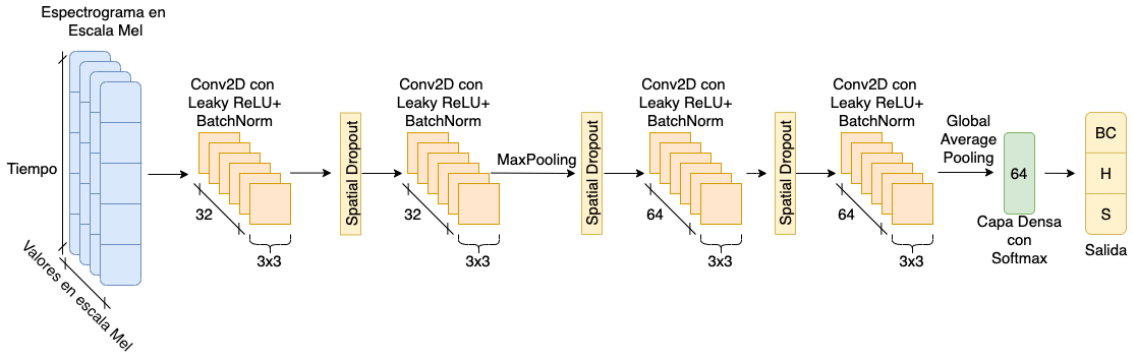


Fig. 3.5: Diagrama de la arquitectura de la red neuronal convolucional utilizada

En la Figura 3.5 vemos a la arquitectura que *denominaremos* ***Modelo Final de Cuatro Capas***. Podemos notar unas cuantas diferencias con la arquitectura anterior. Agregamos una capa convolucional adicional, las cuales son activadas con Leaky ReLU y luego una Batch Normalization. Es necesario notar además la presencia de capas de Spatial

Dropout entre cada capa convolucional. Además de un solo Max Pooling en la mitad de la arquitectura.

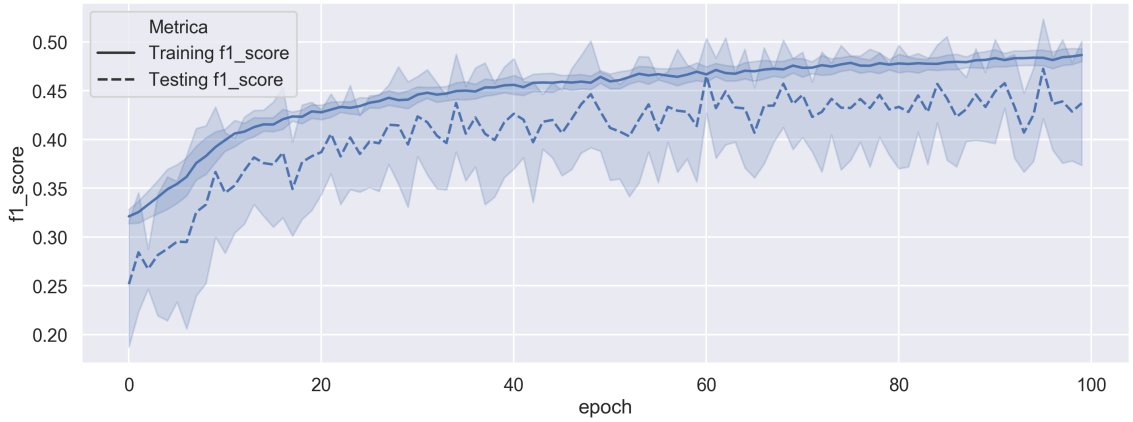


Fig. 3.6: Evolución de la métrica Score F_1 del modelo convolucional utilizado, promediada entre los 10 folds a medida que avanzan los epochs. Es importante notar el cambio de eje Y en relación a los anteriores.

En la Figura 3.6 correspondiente al tercer modelo vemos algo particular. El modelo no presenta signos de divergencia considerable entre entrenamiento y validación, y además ambas métricas obtienen valores similares en cada epoch, por lo que podemos ver que este modelo carece de los problemas de sobreajuste presentados anteriormente.

A partir de lo investigado y explorado con el modelo convolucional de cuatro capas procedimos a comparar los resultados, agregando algunas características del modelo elegido al modelo tres capas, con el objetivo de obtener un mayor entendimiento de como mejora el modelo convolucional de tres capas a medida que le agregamos estas características.

3.1.4 Comparación de arquitecturas convolucionales

En esta sección evaluaremos las diferencias entre los dos modelos convolucionales. Particularmente, distinguimos cuatro características principales que las diferencian y evaluamos como impactan en su desempeño:

- La inclusión de regularización de tipo L2
- La inclusión de capas de activación de tipo Leaky ReLU
- El cambio de las capas de Dropout por capas de Spatial Dropout
- El agregado de una capa convolucional adicional a las ya presentes

Tabla de Comparación de Características

	Score F_1			
	Set de Entrenamiento		**Set de Validación**	
Modelo	**Promedio**	**Std**	**Promedio**	**Std**
Base de Tres Capas	0.195	0.124	0.130	0.085
Regularización L2	0.237	0.111	0.192	0.120
Spatial Dropout	0.178	0.077	0.136	0.082
Leaky ReLU	0.168	0.076	0.133	0.073
Agregando Cuarta Capa	0.321	0.103	0.282	0.108
4Capas+L2	0.397	0.089	0.356	0.097
Final de 4 Capas	**0.473**	**0.014**	**0.459**	**0.040**

Tab. 3.1: Comparación de los Scores F_1 de los modelos estudiados para el experimento

En la Tabla 3.1 podemos observar cómo fueron modificándose los resultados al agregarle al *Modelo Base de Tres Capas* cada una de las características del *Modelo Final de Cuatro Capas*. Para obtener estos resultados entrenamos cada modelo de la misma forma: Validación Cruzada de 10 *folds*. Luego, obtuvimos el mejor resultado para el set de validación según el Score F_1 para cada entrenamiento y tomamos el promedio y el desvío estándar para comparar los resultados.

Regularización L2

Comenzando por agregar regularización de tipo L2, podemos ver en la Tabla 3.1 que el modelo presenta una leve mejora al realizar la comparación con el *Modelo Base de Tres Capas*. Vemos que en el set de entrenamiento el Score F_1 mejora levemente y vemos una reducción del desvío estándar, aunque sea en una cantidad casi despreciable. Similarmente, vemos una mejora en el Score F_1 en validación, aunque además un aumento, también despreciable, del desvío estándar.

Lo visto es sorprendente ya que al observar los entrenamientos (como podemos ver en la Figura 3.4) no estábamos ante una situación de sobreajuste en el modelo original (o al menos no era el principal problema, ya que vemos una sutil divergencia entre las curvas al final del gráfico), por lo que aplicar una modificación al modelo cuyo objetivo es reducir el sobreajuste no esperábamos mejoras en los resultados, aunque sean marginales.

Spatial Dropout

Similarmente a lo ocurrido en el caso anterior, vemos que para el caso de Spatial Dropout su inclusión no hace más que perjudicar al aprendizaje. La inclusión de este tipo de capas empeora los resultados en entrenamiento y no es una mejora considerable la del set de validación, aunque podemos ver una mejora en el desvío estándar de los resultados.

Sin embargo, esto no es algo considerable como positivo debido a los pobres resultados en el promedio del Score F_1.

LeakyReLU

Una de las hipótesis principales a la hora de entender el bajo desempeño del modelo convolucional básico era el problema denominado *dying ReLU* (explicado en la Sección 2.2.6) en el cual las capas de activación de tipo ReLU retornan siempre 0 como resultado sin importar la entrada, comportamiento que resulta imposible de modificar con mayor entrenamiento. Una de las soluciones a este problema es la capa de activación de tipo LeakyReLU la cual solo retorna cero para input cero y valores cercanos a cero para valores negativos.

En la Tabla 3.1 podemos ver que este no era el caso ya que que la inclusión de esta capa no hace más que empeorar los resultados. Contrario a la hipótesis explicada en el párrafo anterior, los resultados son incluso peores en comparación al *Modelo Base de Tres Capas*, al menos en entrenamiento. Cabe destacar que en el gráfico de entrenamiento análogo a la Figura 3.4, vimos que todo el aprendizaje del modelo durante todos los epochs resulta nulo ya que no cambia en todo el entrenamiento su valor del Score F_1 posterior a la segunda epoch. Más allá de si el problema original era o no de *dying ReLU*, queda claro que las Leaky ReLU por sí solas no realizaron mejoras al modelo.

Cuarta Capa Convolucional

En contraste con los resultados anteriores, podemos ver en la Tabla 3.1 como mejora considerablemente el rendimiento del modelo al agregar otra capa convolucional. Esto resulta interesante ya que no había evidencia considerable que indique que el modelo carecía de complejidad, más aún teniendo en cuenta la baja cantidad de datos de entrenamiento.

En base a lo visto en los experimentos, observamos que agregar una nueva capa presentaba *sobreajuste* y una mayor varianza en los resultados para cada epoch, por lo que realizamos un nuevo entrenamiento combinando dos características que vimos en estos experimentos: Agregar una cuarta capa y aplicar regularización L2, ya que esto haría que se reduzca el sobreajuste visto en el experimento y, por lo visto en esta sección, podría hasta mejorar marginalmente los resultados ya que son las únicas dos características con mejoras considerables en el Score F_1 de los modelos.

Cuarta Capa Convolucional con Regularización L2

Como experimento final podemos ver en la Tabla 3.1 que al aplicar regularización L2 en el modelo básico con 4 capas convolucionales vemos una mejora del rendimiento considerable como era de esperar, y se reduce el desvío estándar entre los folds. Si bien es una reducción marginal, es consistente ya que podemos verla tanto en entrenamiento como en validación. Por lo tanto, estos resultados nos dan un mejor entendimiento del

funcionamiento del modelo final y como las distintas partes contribuyen al desempeño total.

En conclusión, hemos mostrado y comparado los distintos modelos con los que iniciamos el trabajo, realizando una exploración más profunda con el modelo convolucional básico con el objetivo de *alcanzar* el rendimiento final del modelo elegido.

4. PREDICCIÓN DE TRANSICIONES DE TURNO

En este capítulo nos concentramos en realizar experimentos de predicción de transiciones de turno utilizando un modelo de red neuronal convolucional. El modelo fue obtenido a partir de la exploración de hiperparámetros y comparándolo con otros modelos candidatos, ya utilizados para tareas similares. Una vez obtenido el modelo final (que se puede ver en la Figura 4.1) exploramos qué propiedades más contribuyeron al desempeño del mismo, los detalles de esta búsqueda se encuentran en la Sección 3.

Dado un conjunto de IPUs donde cada una fue etiquetada con la transición posterior correspondiente, las cuales pueden ser *HOLD*, *SWITCH*, o *BACKCHANNEL*, se entrenó el modelo seleccionado con estos datos mediante *Validación Cruzada* de diez *folds*, donde ningún sujeto tiene IPUs en más de un *fold* a la vez.

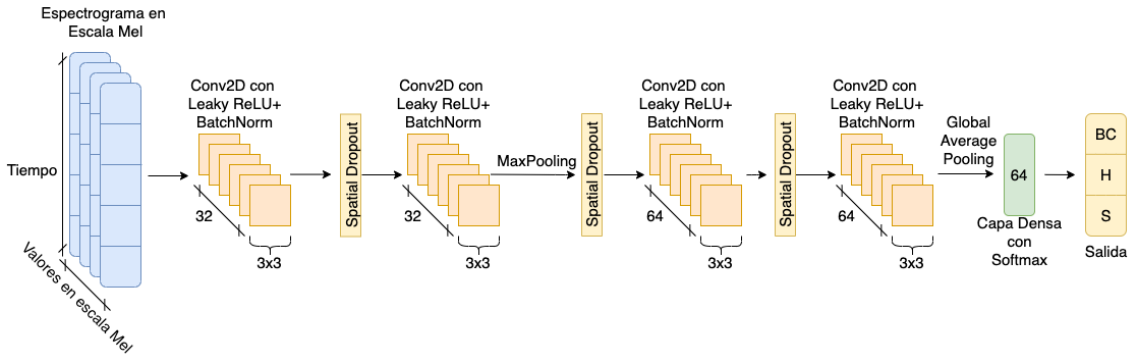


Fig. 4.1: Diagrama de la arquitectura de la red neuronal convolucional utilizada

A partir de este esquema de entrenamiento, realizamos comparaciones con modelos de azar y modelos externos, agrupando los audios en diferentes grupos y evaluamos el desempeño del modelo en un nuevo set de datos. Por último, vimos si aumentaría el desempeño del modelo al obtener una mayor cantidad de datos.

4.1 Exploración del Modelo Seleccionado

Con el objetivo de analizar el desempeño del modelo elegido observamos el modelo a partir de distintas métricas. En los resultados iniciales queremos ver como fue aprendiendo el modelo. En la Figura 4.2 podemos ver como avanza el valor de la función de pérdida durante el entrenamiento, siendo la linea punteada los valores en validación y la linea sólida los valores en entrenamiento, mientras que las sombras al rededor de las mismas son el desvío estándar. Como podemos ver en la Figura 4.2, los valores de pérdida promedio de entrenamiento y validación se mantienen cercanos, algo muy deseable al entrenar un

modelo de redes neuronales, para evitar el sobreajuste. Es importante notar la sombra al rededor de los valores promedio, el cual es el desvío estándar de la muestra (los 10 folds), el cual podemos ver que es muy ruidoso y cambia considerablemente.

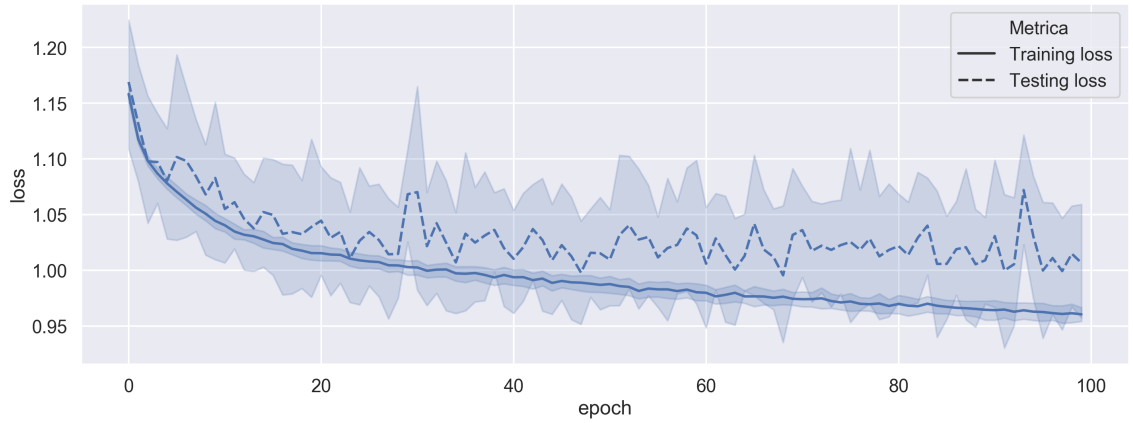


Fig. 4.2: Gráfico mostrando la evolución de la función de pérdida, promediada entre los 10 folds a medida que avanzan los epochs.

Similarmente, podemos ver que sucede lo mismo al observar el gráfico con nuestra métrica objetivo. Las métricas crecen a la par en ambos casos e incluso el desvío estándar se ve reducido al compararlo con la función de pérdida. Es importante notar que el modelo no lograr clasificar correctamente todas las instancias de set de datos de entrenamiento, sin embargo este es el mejor resultado al que pudimos llegar en la etapa de experimentación con los hiperparámetros.

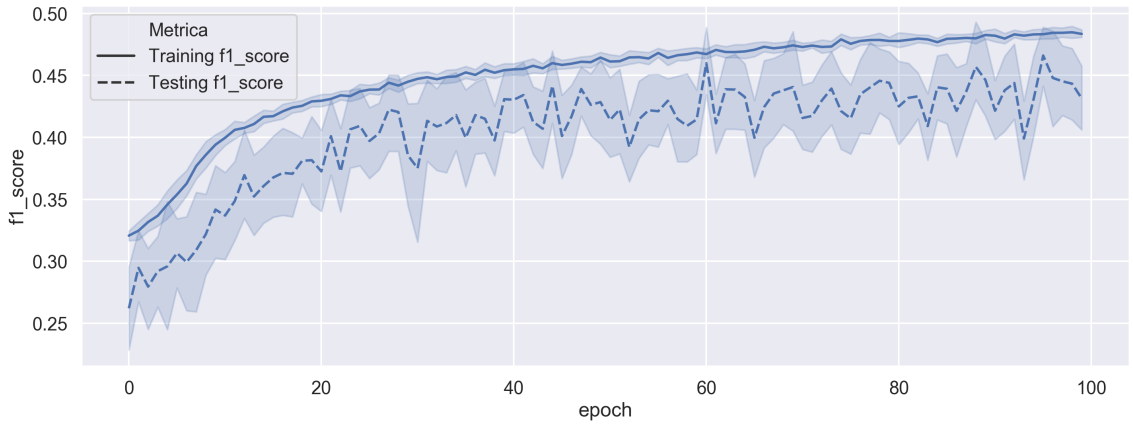


Fig. 4.3: Gráfico mostrando la evolución de la métrica F_1, promediada entre los 10 folds a medida que avanzan los epochs.

Tomando el modelo con el mejor desempeño a lo largo del aprendizaje en cuanto a la métrica *Score* F_1 obtuvimos las predicciones completas de cada categoría. Recordemos que dividimos los usuarios en 10 grupos, entrenamos con 9 y evaluamos en el grupo restante,

haciendo esto 10 veces podemos obtener predicciones en las cuales todos los datos están en el grupo de validación. Como podemos ver en la Figura 4.4 realizamos la comparación de los resultados, teniendo el cuenta la clasificación real y la predicción del modelo, de esta manera podemos ver lo que se denomina *matriz de confusión*.

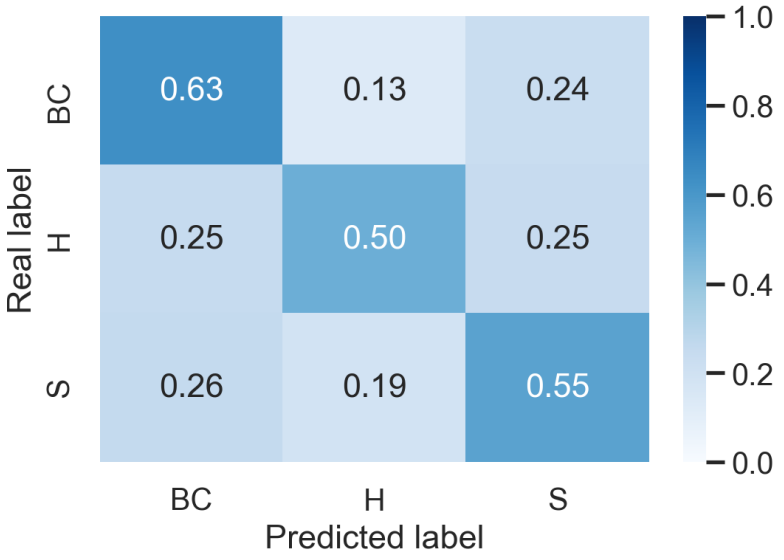


Fig. 4.4: Matriz de confusión comparando las predicciones del modelo con las categorías reales, los totales fueron normalizados por categoría (cada fila suma 1)

Observando los resultados podemos notar que clasifica correctamente más de la mitad de cada una de las instancias de cada clase. Luego es interesante notar que las clases que más se confunde el modelo al clasificar son las minoritarias, es decir, *SWITCH* y *BACKCHANNEL*, donde hay una diferencia considerable al comparar con las predicciones erróneas de esas clases cuando predice un *HOLD*.

Una vez analizados los resultados por separado procedimos a comparar los resultados con un modelo que elija azarosamente las categorías. Para ello, implementamos un modelo el cual retorna al azar una de clases como resultado, con la particularidad de que elige de manera *pesada*, según pesos establecidos en base a la proporción de las clases sobre el total. De esta manera, el nuevo modelo de azar predice al rededor de 65 % de las veces *HOLD*, 24 % de las veces *SWITCH* y 11 % de las veces *BACKCHANNEL*.

En la Figura 4.5 podemos ver los resultados de la comparación de los Score F_1 del modelo elegido y el azar, tomamos el promedio del Score F_1 para cada *sujeto* y lo graficamos, siendo las barras el desvío estándar.

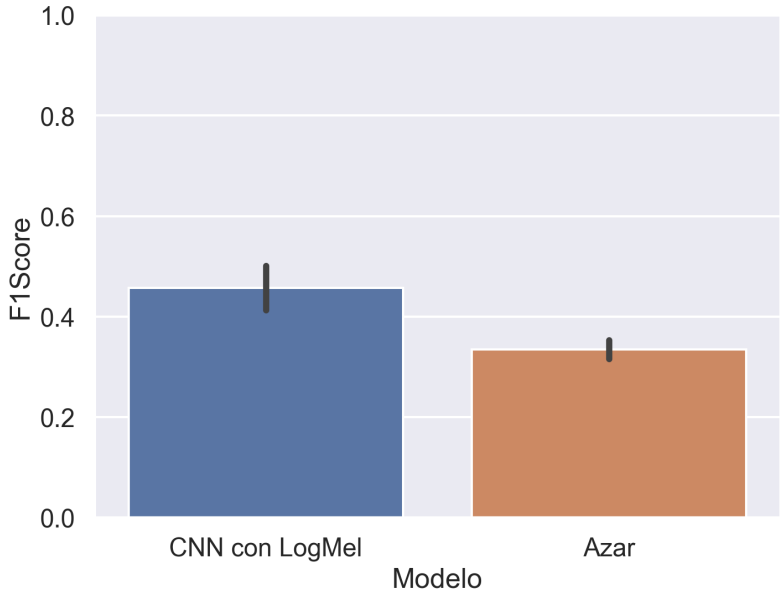


Fig. 4.5: Comparación de los Score F_1 de ambos modelos

Podemos ver una clara diferencia entre ambos modelos, lo cual, si bien es un resultado positivo para el modelo convolucional es necesario profundizar en estos resultados para obtener un mejor entendimiento de los mismos. Con este objetivo decidimos comparar también los Score F_1 de cada modelo divididos por clase:

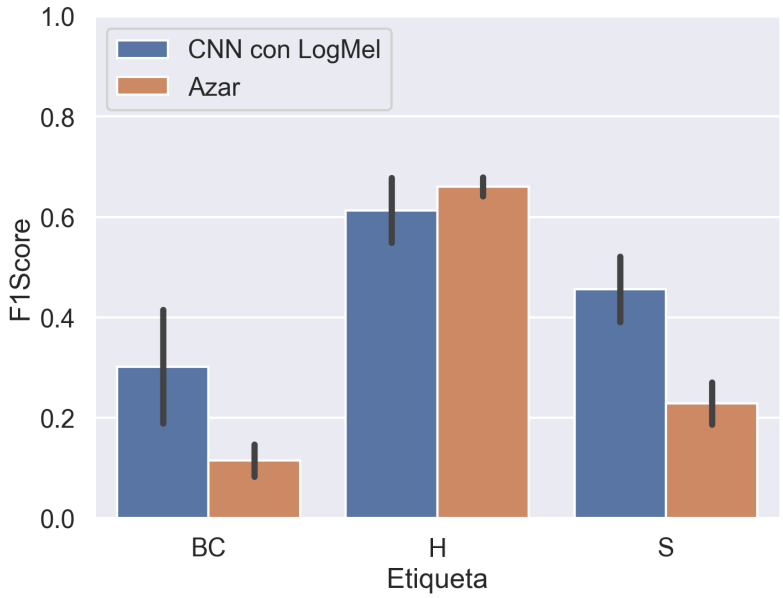
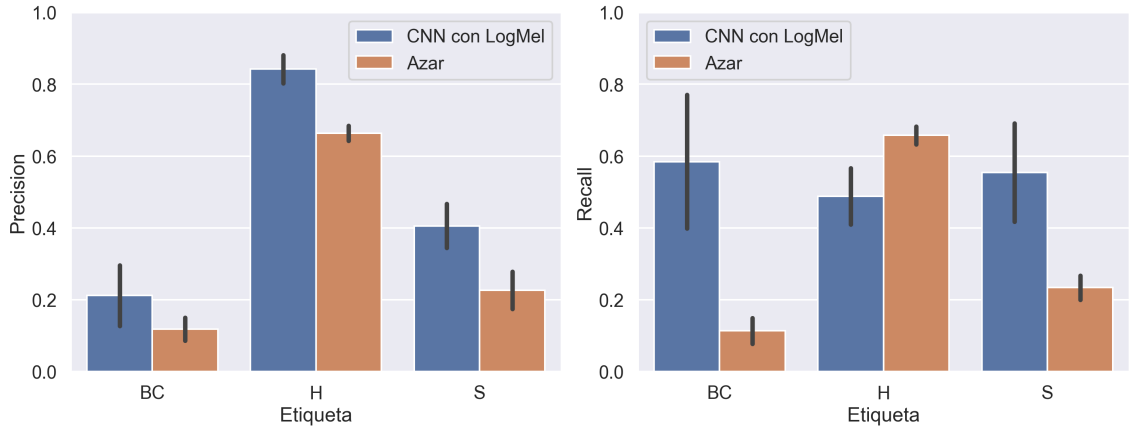


Fig. 4.6: Comparación de los Score F_1 de ambos modelos, divididos por etiqueta de clase

Como podemos ver en la Figura 4.6, hay una clara diferencia de resultados en las clases minoritarias (*SWITCH* y *BACKCHANNEL*) con la clase *HOLD*, ya que ésta no logra superar los resultados del Azar. Sin embargo, es importante notar que nuestro objetivo es obtener buenos resultados en las tres métricas en conjunto, y es por esto que tomamos el promedio de los tres resultados. De esta manera podemos entender el *bajo* rendimiento en la clase *HOLD*.

Además, podemos tratar de entender el origen de estos resultados analizando la composición de los mismos, es decir, observando los gráficos de precisión y recall.



(a) Precisión *(b)* Recall

Fig. 4.7: Comparación de la Precisión y Recall de los modelos de este trabajo (CNN con Logmel) y el Azar, divididos por etiqueta de clase

Podemos ver en la Figura 4.7 (a) cómo se desempeña el modelo al predecir cada una de las clases. Como podemos ver, notamos bajos rendimientos en las clases minoritarias, aunque un gran rendimiento en la clase *HOLD*. Podemos pensar que el modelo, dado un audio clasificado como *HOLD*, es muy seguro que sean realmente de esa etiqueta. No se puede decir lo mismo con las etiquetas restantes, por ejemplo, *BACKCHANNEL*, si bien más que duplica el desempeño del azar, sigue siendo un resultado muy pobre, ya que casi 80 % de las etiquetas elegidas como tal no pertenecen a dicha clase.

En contrapartida, podemos ver en la Figura 4.7 (b) una imagen completa de la situación. Sabemos que el 60 % de las etiquetas de *BACKCHANNEL* están dentro de la clase correcta, y al comparar con el gráfico anterior entendemos que esto se debe a una sobreclasificación de audios a esta clase, muy posiblemente de audios de clase *HOLD*. Esto se puede ver comparando el Recall de *HOLD* contra el Azar. Algo similar es posible que suceda en la clase *SWITCH* pero no es tan significativo como para que se note.

En conclusión, al comparar ambos gráficos se puede observar lo siguiente: Las clases minoritarias mejoran sus resultados a partir de agrupar audios de su clase con audios de la clase *HOLD*, de esta manera asegurándose de clasificar correctamente sus propios audios, pero a la vez reduciendo el desempeño en la clase *HOLD*. Esto se puede entender como una consecuencia de nuestros objetivos, ya que queríamos el mejor modelo que obtenga

clasificaciones balanceadas de las tres clases (por eso el promedio de los Score F_1) y por el otro lado es consecuencia de establecer *pesos de clases* dentro del modelo. Esto hace que las clasificaciones erróneas de las clases minoritarias sean penalizadas de una mayor manera que la clase *HOLD*. En definitiva, es importante ver que el modelo es mejor que el azar de manera consistente a nivel general, lo cual teníamos como objetivo a la hora de realizar este trabajo. Profundizando un poco podemos ver que el modelo falla en ciertas ocasiones más que el azar (como en las etiquetas *HOLD*) y tratamos de esbozar una teoría del por qué de estos resultados.

4.2 Comparación con Modelo Externo

En esta sección analizamos como se desempeña el modelo en comparación con otros similares. Para esta sección comparamos los resultados con un modelo de tipo Random Forest.

4.2.1 Definición del Modelo a Comparar

A diferencia de nuestro trabajo, para utilizar el modelo de Random Forest se debe realizar un proceso inicial de *extracción de features*, es decir, obtener propiedades de los audios a clasificar. Para esto, se realiza para cada instancia un vector de características que se toman como la entrada del modelo.

El experimento contra el que comparamos, obtuvo los vectores de features realizando el siguiente procedimiento. Inicialmente, se extrajeron propiedades de la totalidad de la IPU como son la duración y la cantidad de sílabas por segundo. Luego se computaron diversas propiedades a través de ventanas temporales de 50 milisegundos para el último segundo de cada IPU. Dicha ventana se toma cada 50 milisegundos de salto, de manera que no hay solapamiento entre las características. Para cada ventana se tomó el promedio y su *pendiente* como características. A partir de esta definición, para cada ventana se obtuvo una variedad de parámetros acústicos como el tono, la intensidad y el *jitter*, entre otros. De esta manera se obtiene un vector de tamaño 202 que definen una IPU para el modelo.

Estos vectores de características son la entrada del modelo de Random Forest elegido para realizar la comparación, el cual se entrenó con la misma técnica utilizada en este trabajo, de validación cruzada de 10 *folds*. A su vez, los hiperparámetros del modelo fueron optimizados para esta misma tarea, y con la misma métrica objetivo (macro F_1-Score).

A continuación, comparamos los resultados obtenidos mediante esta técnica, en contraste con nuestro modelo basado en redes convolucionales y el modelo al azar.

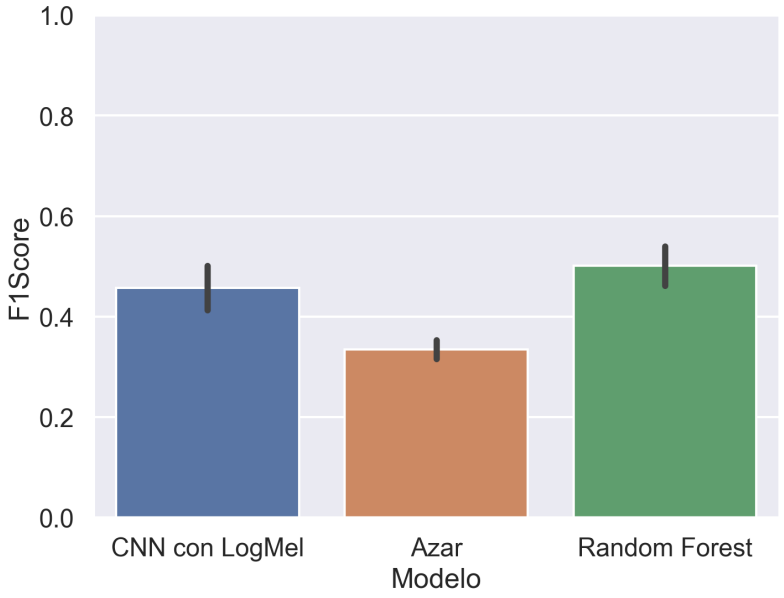


Fig. 4.8: Score F_1 de los modelos de este trabajo (CNN con Logmel) el azar y el Random Forest

Como podemos ver en la Figura 4.8, mirando los resultados del Score F_1 total, los resultados son muy similares, con una diferencia marginal en favor del modelo de *Random Forest*. Sin embargo, es interesante observar el desglose de los números, ya que encontramos resultados importantes de analizar.

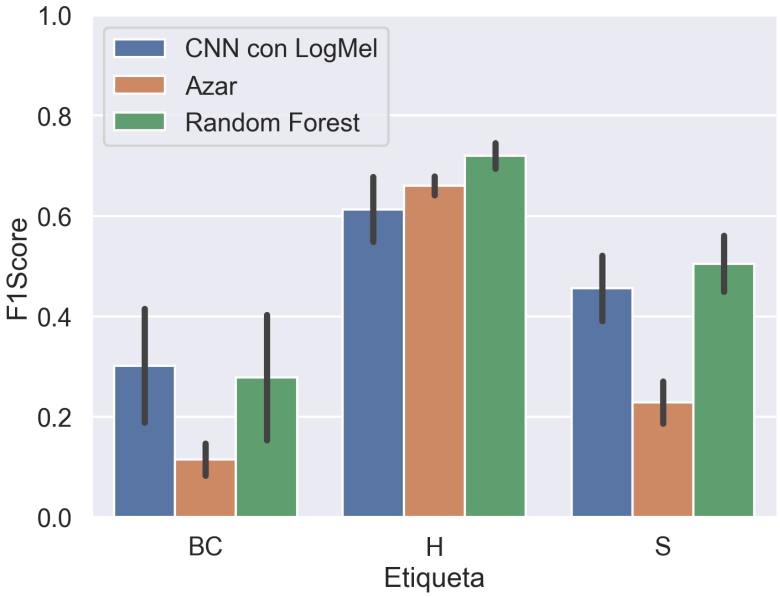
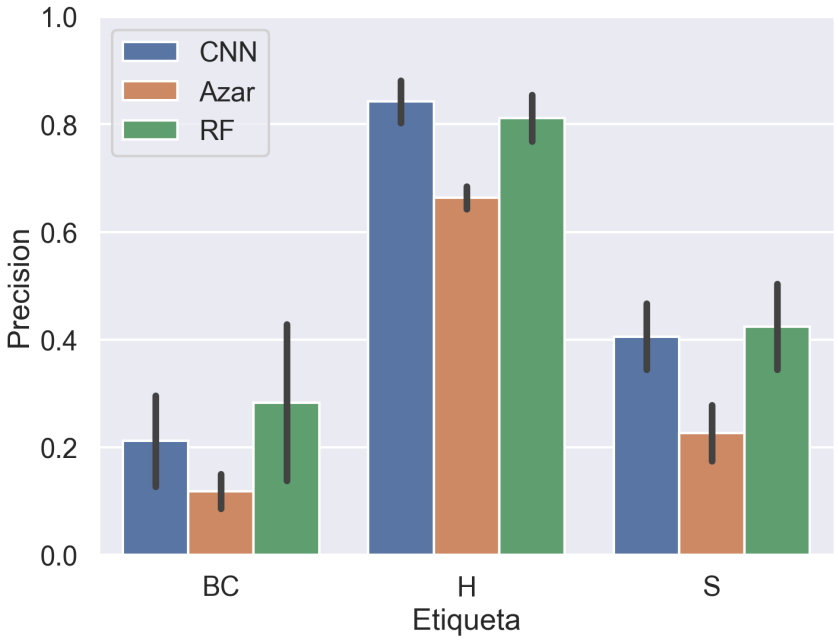
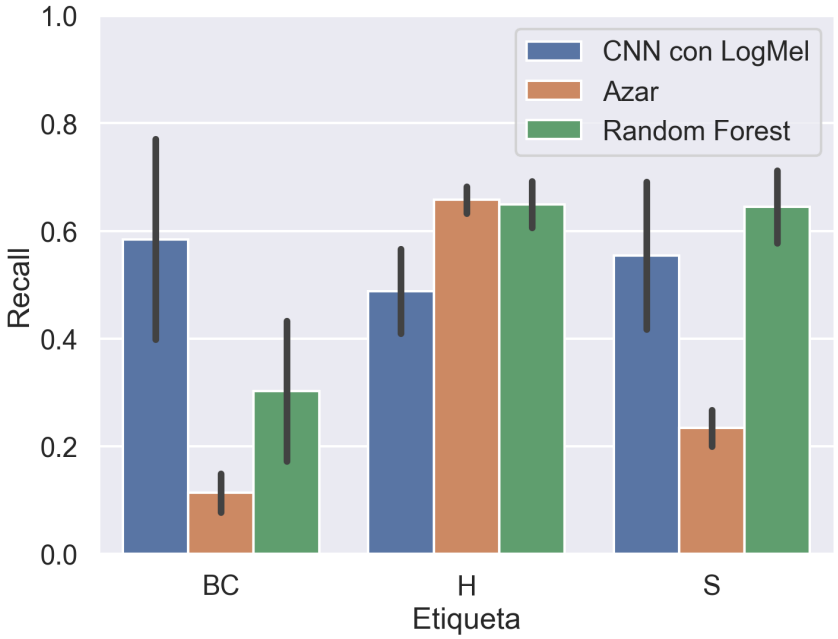


Fig. 4.9: Score F_1 de los modelos de este trabajo (CNN con Logmel) el azar y el Random Forest, divididos por etiqueta de clase

En la Figura 4.8 notamos como se modifica la diferencia de resultados entre el Random Forest y nuestro modelo convolucional a medida que nos movemos hacia las clases con menor cantidad de datos. Mientras que vemos una clara diferencia en *HOLD*, *SWITCH* reduce esta diferencia mientras que en la etiqueta *BACKCHANNEL* el modelo de este trabajo supera (aunque sea por una diferencia mínima) el desempeño del modelo de Random Forest.



(a) Precisión



(b) Recall

Fig. 4.10: Precisión y Recall de los modelos de este trabajo (CNN con Logmel), Azar y Random Forest

Al profundizar en las métricas Precisión y Recall, podemos ver en la Figura 4.10 cómo se conforman los resultados de la Figura anterior. El modelo de este trabajo se destaca por su Recall equilibrado entre las tres clases, sin embargo tiene un notable menor Recall en la etiqueta *HOLD* frente a los otros dos modelos.

Podemos ver que el origen de los mejores resultados con respecto al Random Forest en la etiqueta *BACKCHANNEL* provienen de los resultados positivos en el Recall de esta etiqueta. Todo esto evidencia que es posible que el modelo haya tenido dificultad a la hora de separar un grupo de audios, el cual estaban repartidos entre *HOLD* y *BACKCHANNEL*. Al clasificarlos todos como *BACKCHANNEL*, nos aseguramos un gran Recall de dicha etiqueta, perjudicando así su Precisión y además el Recall de *HOLD*. Esta situación hace que la métrica elegida termine mejorando.

En conclusión, hemos visto que el modelo presentado en este trabajo supera el desempeño de un modelo de azar, mientras que es comparable en resultados con un modelo de features expertos. Además, observamos como influye en los resultados el desempeño por cada categoría profundizando además en las métricas de Precisión y Recall.

4.2.2 Comparación por Grupos

Con el objetivo de realizar una comparación más profunda de los resultados, nos propusimos en esta sección comparar de nuevo los resultados del Random Forest con los resultados del modelo de este trabajo, con la diferencia de hacer distinciones, dividiendo en categorías los audios, según sus propiedades originales. De esta sección nos interesa particularmente descubrir propiedades de la predicción del modelo realizado, que nos otorgue un mayor entendimiento del funcionamiento interno.

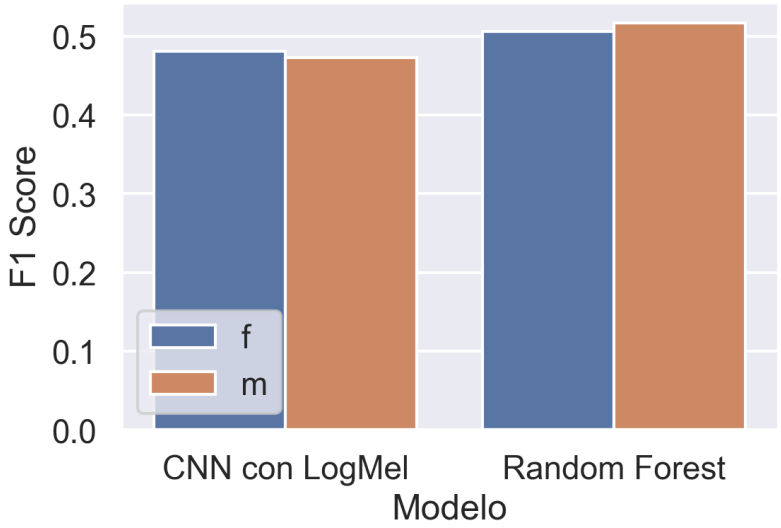


Fig. 4.11: Score F_1 de los modelos de este trabajo (CNN con Logmel) y el Random Forest, divididos por género

Comenzando con la Figura 4.11, podemos ver que no hay diferencias significativas en los resultados al dividir a los usuarios según su género, de todas formas es importante notar esto para verificar que no haya un sesgo particular a la hora de escuchar audio de personas

de distintos géneros. Esto tiene sentido ya que el corpus tiene una cantidad balanceada de hombres y mujeres, como explicamos en la Sección 2.1.

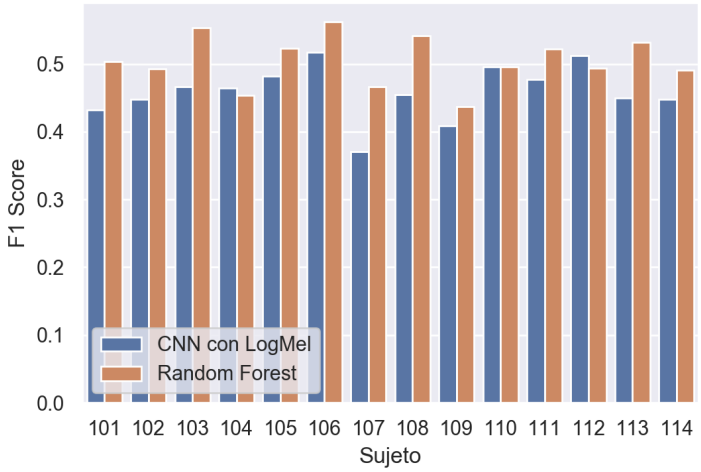
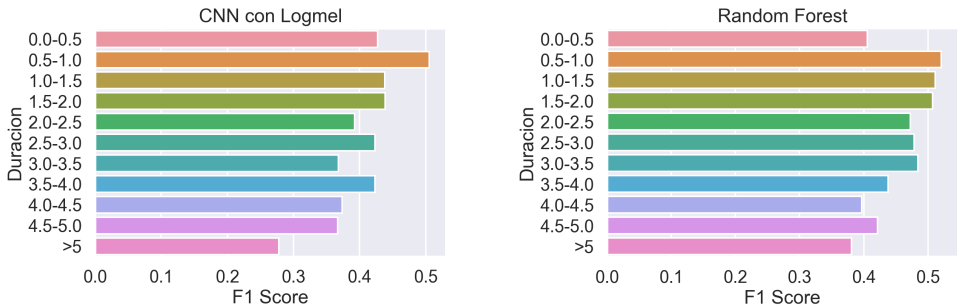


Fig. 4.12: Score F_1 de los modelos de este trabajo (CNN con Logmel) y el Random Forest, divididos por usuario

Al realizar una división similar pero por sujeto notamos en la Figura 4.12 una tendencia similar, ya que a grandes rasgos no hay sesgos particulares que diferencien un modelo del otro. Podemos ver como los sujetos que peor resultados obtienen con un modelo, muy seguramente obtengan resultados con un bajo Score F_1 en el otro, lo cual nos indica una dificultad general por los audios de estos sujetos en particular, por lo que probablemente sean los más difíciles de generalizar.



(a) Precisión *(b)* Recall

Fig. 4.13: F1 Score de los modelos de CNN y Random Forest, separados por duración, agrupados en bloques de 0.5 segundos

En las Figuras 4.13 (a) y (b) sí podemos notar un patrón interesante para el análisis. Dividimos los audios del dataset según su duración, es decir, la cantidad de segundos

que hablan sin silencios. Dicho de otra manera, son la cantidad de segundos previos a la transición de turno que nos interesa.

Notamos en el modelo convolucional (Figura 4.13 (a)) una considerable baja del rendimiento a partir de que la longitud del audio supera los dos segundos. Esto se puede explicar por el hecho de que justamente, como estamos tratando con una arquitectura convolucional que toma como entrada matrices de tamaño fijo, decidimos cortar los audios si superasen los dos segundos. Visto de esta manera, podemos pensar que el modelo se vería beneficiado de aumentar la ventana elegida de dos segundos, ya que podría obtener más información que se pierde.

Sin embargo, podemos ver que algo similar sucede en la Figura 4.13 (b) correspondiente al modelo de Random Forest. Podemos ver como su desempeño se reduce a medida que se incrementa la duración, algo inesperado si tenemos en cuenta lo postulado anteriormente.

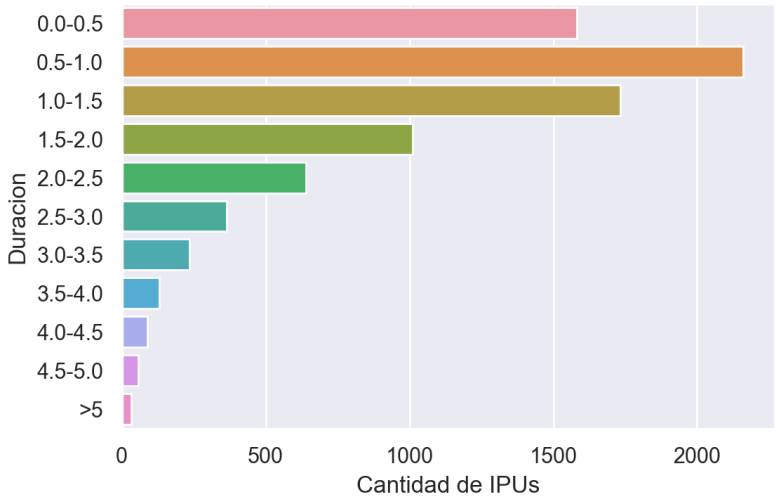


Fig. 4.14: Cantidad de audios en el dataset, por grupo de duración de 0.5 segundos

Finalmente, podemos entender un poco más los resultados de las Figuras 4.13 (a) y (b) viendo la Figura 4.14. Podemos ver que el 80 % de los datos del dataset tienen una duración menor a los 2 segundos, por lo que este grupo es el que más datos concentra. De esta manera es fácil ver que los resultados son una combinación de estas dos propiedades: Al haber menos audios mayores a dos segundos ambos modelos tienen menos información para generalizar y por lo tantos su desempeño es peor, además, el modelo convolucional presenta un marcado deterioro de su rendimiento en audios estrictamente mayores a los dos segundos ya que estamos obteniendo menor información de la total en este grupo.

4.3 Comparación de resultados en nuevo corpus

A la hora de la elección de un modelo final para poder realizar predicciones fuera de la experimentación, es importante evaluar mediante un dataset que no sea parte del entrenamiento de los modelos. En nuestro caso hemos utilizado la totalidad del *Lote A* para el entrenamiento, y realizaremos la comparación con *Lote B*.

Para realizar las predicciones se tomó al azar un modelo entrenado de los 10 *folds* de la validación cruzada, a partir de estos resultados se obtuvieron las métricas de interés para el trabajo y se realizaron las comparaciones con los resultados obtenidos con los datos originales.

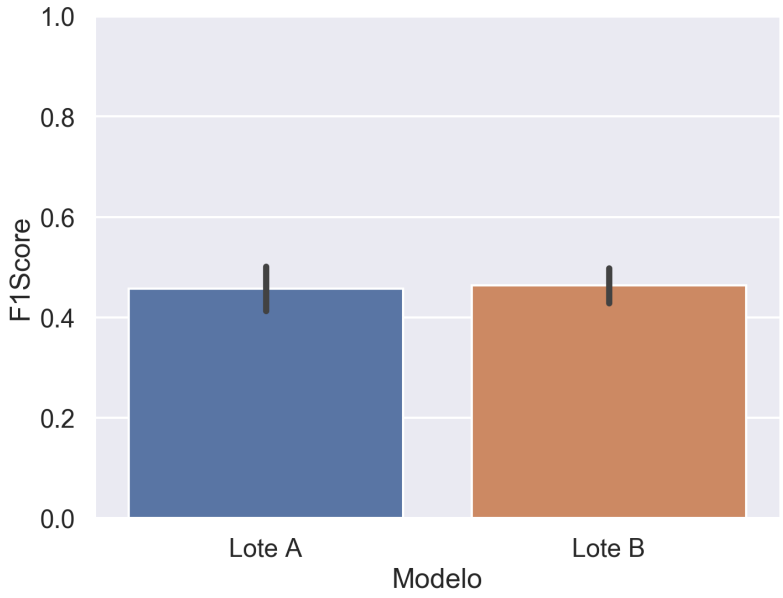


Fig. 4.15: Score F_1 de los modelos de este trabajo (cuyos resultados se ven en el Lote UBA A) evaluando en el nuevo dataset (Lote UBA B)

Los resultados son los siguientes: En la Figura 4.15 podemos ver como el modelo se comporta de manera similar en el nuevo dataset, con un resultado de 0.46 de Score F_1, en comparación con un 0.45 en los datos de prueba.

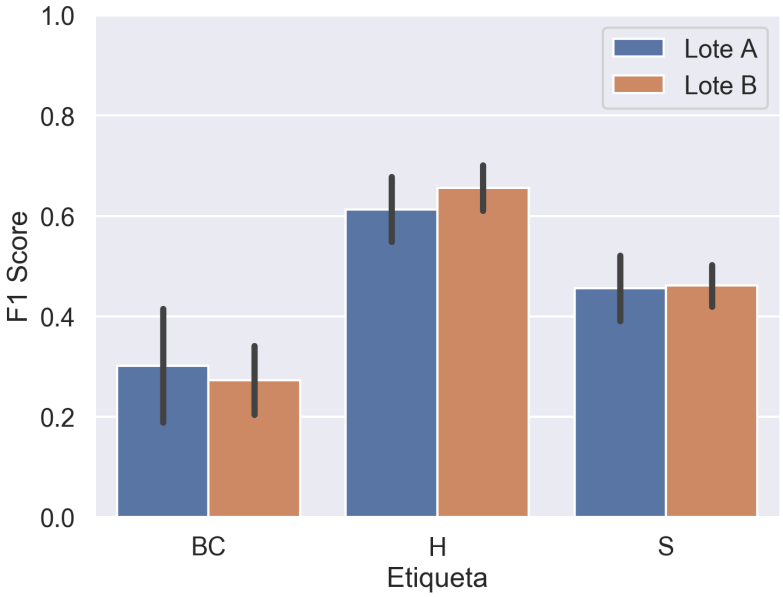
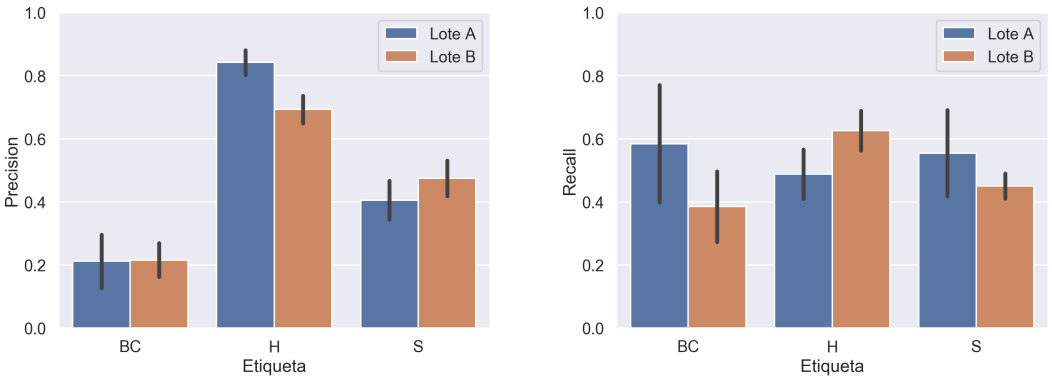


Fig. 4.16: Score F_1 de los modelos de este trabajo (CNN con Logmel) evaluando en el nuevo dataset

Profundizando un poco en los resultados podemos ver en la Figura 4.16 como se dividen los resultados por etiquetas, vemos que los resultados son similares a lo que ya veníamos viendo en comparaciones anteriores y se repite el mismo patrón: Un mayor Score F_1 de la etiqueta principal viene acompañada de un descenso de las etiquetas restantes. El mejor resultado en la clase *HOLD* puede deberse a que, como el modelo está entrenado con una mayor cantidad de datos con esta etiqueta, es mejor a la hora de generalizar ante nuevos usuarios y nuevos audios, como es este caso.



(a) Precisión *(b)* Recall

Fig. 4.17: Precisión y Recall de las curvas de aprendizaje, divididas por etiqueta

Al realizar de nuevo la división de los resultados por etiqueta y por Precisión y Recall

podemos ver de nuevo similitudes entre los resultados. Lo primero que notamos es un mayor Recall en la etiqueta *HOLD* y un considerable peor Recall en la etiqueta *BACKCHANNEL*. Podemos entender estos resultados por la cantidad de datos con los que fue entrenado el modelo: Los mejores resultados se observan en la clase mayoritaria, la cual es la clase en la que más generalización se puede hacer.

En conclusión, propusimos en el análisis de los resultados en un nuevo grupo de datos, el cual nunca utilizamos en los entrenamientos, para evaluar el posible desempeño en un entorno real. Vemos que los resultados conservan una gran similitud con los datos de entrenamiento y el modelo obtiene mejor desempeño donde más información tiene. Estos resultados son muy positivos ya que indican que los resultados vistos anteriormente son confiables y podemos esperar resultados similares ante nuevos grupos de usuarios que no hayan sido utilizados para el entrenamiento.

4.4 Curva de Aprendizaje

A la hora de utilizar redes neuronales para el entrenamiento de un modelo, es imperativa la necesidad de grandes cantidades de datos, ya que su desempeño crece considerablemente a medida que dataset crece, en comparación con modelos clásicos de Aprendizaje Automático como un Random Forest.

Es por esto que nos propusimos analizar los resultados de la *curva de aprendizaje* del modelo, para saber si es posible que mejoren los resultados con un dataset mayor. Para esto generamos distintos grupos de los datos del corpus original de manera que cada nuevo grupo incluye más elementos que el anterior. Analizando gráficamente el desempeño para cada grupo de datos podemos ver cómo cambia la métrica elegida (*Score* F_1 en este caso) a medida que se agregan más datos, y posiblemente considerar observando las curvas si con más datos fuera del corpus original se podrían obtener mejores resultados.

4.4.1 Preparación del Experimento

Para este experimento hay que tener en cuenta que necesitamos entrenar el modelo una gran cantidad de veces y generar datasets de distinta cantidad de audios para el entrenamiento.

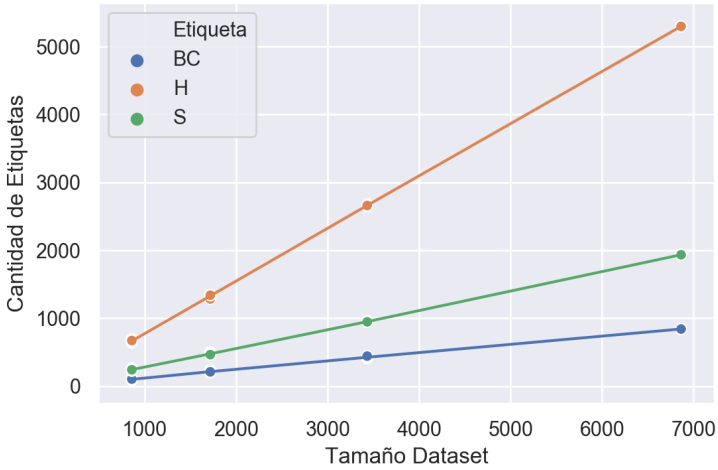


Fig. 4.18: Comparación de la cantidad de audios divididos por su etiqueta

Para obtener una buena medida de como va creciendo el rendimiento decidimos crear los dataset con espaciado logarítmico. Es decir, comenzamos con el dataset completo, luego la mitad, luego la mitad, y así sucesivamente. Realizamos estos pasos cuatro veces, de manera que terminamos con cuatro lotes de datasets, de al rededor de 1000, 2000, 4000 y 8000 datos cada uno. Decidimos mantener la cantidad de usuarios distintos estable, por lo que los datasets fueron generados de tal manera que todos tengan todos los usuarios posibles. Entrenamos con Validación Cruzada de 7 grupos y para cada tamaño generamos cinco datasets para una mayor robusteza con cinco semillas distintas para inicializar el generador de números aleatorios. En la Figura 4.18 vemos como varía la composición de etiquetas a medida que aumenta el tamaño del dataset generado. Es importante notar que cada punto es un dataset generado y que las diferencias en cantidades son mínimas, por lo que no se llega a apreciar en el gráfico.

4.4.2 Resultados

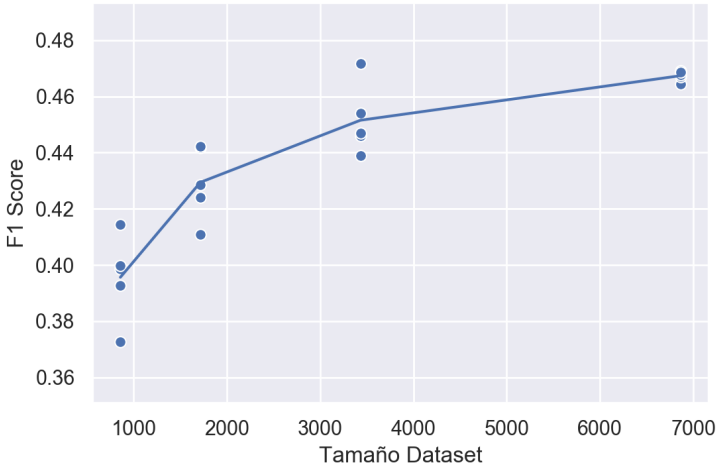


Fig. 4.19: Curva aprendizaje de la métrica F_1 Score

Podemos ver en la Figura 4.19 un notable aumento del Score F_1 a medida que aumentamos el tamaño del dataset. No solo eso, podemos ver también como se reduce la varianza entre las distintas corridas, sin embargo harían falta un poco más de pruebas para verificar esto, dado que solo se puede notar la reducción de varianza desde el cambio de los 4000 a 8000 datos. Estos resultados nos dan evidencia de que sería beneficioso para el modelo una mayor cantidad de datos, evidenciado por la clara curva ascendente del Score F_1. Más aún, veamos los resultados divididos por etiquetas, como estuvimos haciendo hasta ahora.

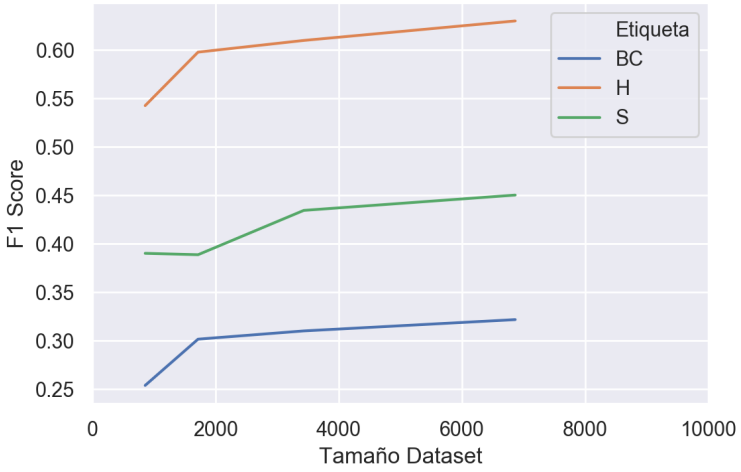
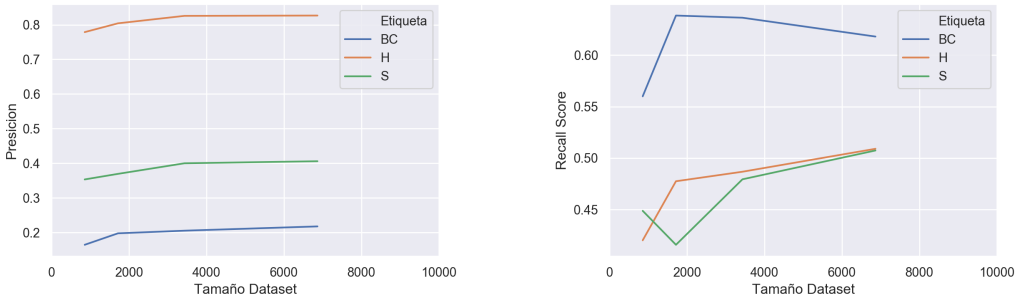


Fig. 4.20: Curva aprendizaje de la métrica F_1 Score

De la misma manera que vimos en la Figura 4.19, podemos ver en la Figura 4.20 como mantiene la tendencia creciente, principalmente para las etiquetas *SWITCH* y *HOLD*, las cuales son justamente las de mayor incremento proporcional en el dataset, como vimos en la Figura 4.18.



(a) Precisión *(b)* Recall

Fig. 4.21: Precisión y Recall de las curvas de aprendizaje, divididas por etiqueta

Podemos ver en la Figura 4.21 (a) como crecen los resultados de las etiquetas *SWITCH* y *BACKCHANNEL* mientras que *HOLD* se mantiene similar durante todos los datasets. Sin embargo, podemos ver unos grandes cambios en los resultados de la Figura 4.21 (b), en la que vemos el Recall. Mientras que en las etiquetas *SWITCH* y *HOLD* podemos ver un crecimiento considerable, no pasa lo mismo con *BACKCHANNEL* el cual ve su Recall reducido a medida que aumenta la cantidad de datos.

De la misma manera que hemos visto en experimentos anteriores, es posible que el

modelo encuentre dificultad en diferenciar un grupo de audios que no corresponden a la categoría *BACKCHANNEL*. A su vez, como se incluyen muchos más nuevos audios de las categorías restantes que de *BACKCHANNEL*, podemos pensar que es mayor el porcentaje de etiquetas nuevas mal clasificadas en *BACKCHANNEL* que clasificadas correctamente.

En conclusión presentamos el experimento de curvas de aprendizaje y nos propusimos ver si nos ayudaría a obtener mejores resultados una mayor cantidad de datos. Los resultados muestran que obtener más datos para entrenar el modelo podrían ayudar a obtener un mayor *Score* F_1. Además profundizamos en los resultados y vimos como afecta a cada una de las etiquetas el aumento de los datos.

5. CONCLUSIONES

En el presente trabajo investigamos la construcción de modelos de redes neuronales profundas de tipo convolucionales para el problema de *Turn Taking* a partir de espectrogramas extraídos de la señal acústica. En particular, construimos modelos con el objetivo de predecir, a partir de un pequeño segmento de audio justo antes de cada pausa, cómo continuará la conversación.

Con este objetivo en mente, y ante la ausencia de información sobre arquitecturas convolucionales aplicadas al problema de cambios de turno, exploramos arquitecturas que hayan sido utilizadas en problemas similares. De los cuales escogimos tres: un modelo basado en capas de atención; un modelo convolucional, orientado a la tarea de reconocimiento de comandos de voz; y un segundo modelo convolucional previamente utilizado para la tarea de reconocimiento de escenas acústicas.

A través de diversos experimentos, seleccionamos la mejor arquitectura y comparamos su desempeño con un modelo basado en Random Forest (técnica que requiere atributos construidos mediante un trabajo profundo de ingeniería de atributos). Al realizar la comparación, observamos que nuestro modelo no logra superar el desempeño obtenido por éste, pero se acerca lo suficiente como para considerarlo un candidato competitivo — debido a la simplicidad de la extracción de atributos (comparado contra el trabajo requerido en el caso de Random Forest) y a la gran velocidad de procesamiento (comparándolo contra modelos de redes neuronales recurrentes).

Otro de los puntos fuertes del modelo fue su capacidad de generalización. Los resultados en un corpus completamente nuevo han sido de una gran similitud, y si bien el corpus consta del mismo experimento con nuevos sujetos, los resultados son positivos y se podrían esperar resultados similares en otras conversaciones en dominios similares (conversaciones colaborativas entre dos personas).

En base a los experimentos realizados comparando los resultados por grupos, vimos que el modelo no presenta sesgos considerables con respecto al género del sujeto. Además, vimos que el modelo presenta una considerable baja de rendimiento cuando se toman los IPUs de longitud mayor a dos segundos, lo cual es positivo ya que los IPUs para entrenamiento y clasificación deben ser cortados para ser tomados como entrada para el modelo y elegimos como límite esa longitud de tiempo y esto indica que aumentando la ventana de tiempo permitida podríamos mejorar los resultados.

Finalmente, por medio de las curvas de aprendizaje, mostramos evidencia que sugiere que el modelo elegido presentaría mejores resultados si se agregara una mayor cantidad de datos — algo esperable este tipo de modelos.

En el futuro será importante realizar una mayor investigación a partir del modelo presentado, con el objetivo de profundizar su rendimiento, ya que por lo mostrado en el trabajo se cree que es una meta posible de lograr. Además, es importante destacar que

se puede investigar con mayor profundidad los modelos con capas de atención, ya que lo presentado en el trabajo indica que es posible como una alternativa a estos problemas.

Por otro lado, cabe destacar la otra gran corriente no investigada en este trabajo, la cual es la idea de utilizar modelos *pre entrenados* en corpus de datos de magnitudes mayores y luego realizar optimizaciones para lograr resultados ajustados a esta tarea. Dichos modelos se han aplicado en otras áreas con grandes resultados, por lo que no sería raro obtener desempeños comparables o superiores con esta técnica.

Finalmente, una de las áreas que no se pudo explorar en este trabajo es el de la *interpretabilidad*, es decir, la capacidad entender los resultados del modelo, por medio de visualizaciones. Esto tiene como principal objetivo obtener un más profundo entendimiento de la relación de las pistas de cambio de turno con sus resultados, y eventualmente contribuir a avances en esta área.

Debido a la falta de literatura relacionado a este problema, la búsqueda de arquitecturas e hiperparámetros de la red resultó ser una tarea muy desafiante. De todas maneras, pensamos que el objetivo principal de este trabajo fue cumplido debido a que pudimos mostrar la factibilidad de utilizar arquitecturas convolucionales sobre la tarea de detección de transiciones de turno, lo cual creemos que puede ser de gran utilidad para decisiones con impacto en la naturalidad y fluidez de futuros asistentes virtuales hablados.

Bibliografía

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

[Bögels and Torreira, 2015] Bögels, S. and Torreira, F. (2015). Listeners use intonational phrase boundaries to project turn ends in spoken interaction. *Journal of Phonetics*, 52:46–57.

[Brusco et al., 2017] Brusco, P., Pérez, J. M., and Gravano, A. (2017). Cross-linguistic study of the production of turn-taking cues in american english and argentine spanish. In *Interspeech*, pages 2351–2355.

[Coimbra de Andrade et al., 2018] Coimbra de Andrade, D., Leo, S., Loesener Da Silva Viana, M., and Bernkopf, C. (2018). A neural attention model for speech command recognition. *ArXiv e-prints*.

[Duncan, 1974] Duncan, S. (1974). On the structure of speaker–auditor interaction during speaking turns. *Language in society*, 3(2):161–180.

[Ferrer et al., 2002] Ferrer, L., Shriberg, E., and Stolcke, A. (2002). Is the speaker done yet? faster and more accurate end-of-utterance detection using prosody. In *Seventh International Conference on Spoken Language Processing*.

[Ford and Thompson, 1996] Ford, C. E. and Thompson, S. A. (1996). Interactional units in conversation: syntactic, intonational, and pragmatic resources for the management of turns. *Studies in interactional sociolinguistics*, 13:134–184.

[Gravano and Hirschberg, 2011] Gravano, A. and Hirschberg, J. (2011). Turn-taking cues in task-oriented dialogue. *Computer Speech & Language*, 25(3):601–634.

[Hara et al., 2018] Hara, K., Inoue, K., Takanashi, K., and Kawahara, T. (2018). Prediction of turn-taking using multitask learning with prediction of backchannels and fillers. *Listener*, 162:364.

[Hjalmarsson, 2011] Hjalmarsson, A. (2011). The additive effect of turn-taking cues in human and synthetic voice. *Speech Communication*, 53:23–25.

[Liu et al., 2019] Liu, C., Ishi, C., and Ishiguro, H. (2019). A neural turn-taking model without rnn. In *Interspeech*, pages 4150–4154.

[Maier et al., 2017] Maier, A., Hough, J., and Schlangen, D. (2017). Towards deep end-of-turn prediction for situated spoken dialogue systems. *Proceedings of INTERSPEECH 2017*.

[Phan et al., 2019] Phan, H., Chén, O. Y., Pham, L., Koch, P., Vos, M. D., McLoughlin, I., and Mertins, A. (2019). Spatio-temporal attention pooling for audio scene classification.

[Roddy et al., 2018a] Roddy, M., Skantze, G., and Harte, N. (2018a). Investigating speech features for continuous turn-taking prediction using lstms. *arXiv preprint arXiv:1806.11461*.

[Roddy et al., 2018b] Roddy, M., Skantze, G., and Harte, N. (2018b). Multimodal continuous turn-taking prediction using multiscale rnns. In *Proceedings of the 2018 on International Conference on Multimodal Interaction*, pages 186–190. ACM.

[Sacks et al., 1974] Sacks, H., Schegloff, E. A., and Jefferson, G. (1974). A simplest systematics for the organization of turn-taking for conversation. *language*, pages 696–735.

[Skantze, 2017] Skantze, G. (2017). Towards a general, continuous model of turn-taking in spoken dialogue using lstm recurrent neural networks. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 220–230.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need.

[Ward, 2019] Ward, N. G. (2019). *Prosodic patterns in English conversation*. Cambridge University Press.

[Warden, 2018] Warden, P. (2018). Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*.

[Wennerstrom and Siegel, 2003] Wennerstrom, A. and Siegel, A. F. (2003). Keeping the floor in multiparty conversations: Intonation, syntax, and pause. *Discourse Processes*, 36(2):77–107.