ISSN 1870-4069

Detección de comandos de voz con modelos compactos de aprendizaje profundo

Edoardo Bucheli-Susarrey[1], Miguel González-Mendoza[1], Oscar Herrera-Alcántara[2]

[1] Tecnologico de Monterrey, Escuela de Ingeniería y Ciencias, Monterrey, N.L., México
[2] Universidad Autónoma Metropolitana Azcapotzalco, Ciudad de México, México
edoardo.bucheli@tec.mx, mgonza@tec.mx, oha@azc.uam.mx

Resumen. El problema de Detección de Comandos consiste en localizar palabras de un vocabulario corto en un segmento de audio. La Detección de Comandos corre constantemente en el fondo de dispositivos móviles por lo que resulta necesario crear modelos económicos en cuanto a memoria y cómputo. Utilizando el conjunto de datos *Simple Speech Commands Detection* presentamos un estudio comparativo con dos tipos de capas. Las capas diseñadas consisten en métodos de extracción de representaciones de audio basadas en la Transformada de Fourier y los Bancos de Filtros Mel. Las capas diseñadas pertenecen al área de Aprendizaje Profundo e incluyen capas de perceptrones, capas recurrentes y capas convolucionales. Mediante el Canal de Aprendizaje profundo, organizamos estas capas para resolver el problema.

Palabras clave: detección de comandos, reconocimiento de voz, aprendizaje profundo, simple commands speech detection, preprocesamiento de audio.

Voice Command Detection with Compact Deep Learning Models

Abstract. The Keyword Detection problem consists in localizing a small vocabulary of words embedded in some stream of audio. Keyword Detection constantly runs in the background of many mobile devices and thus it becomes a requirement to create models with a small memory footprint and low computational power. Using the Simple Speech Commands Detection data set, we present a comparative study using two types of layers. Hand-Engineered layers are created from audio feature extraction models based on the Fourier Transform and Mel Filterbanks. Learned layers belong to the Deep Learning literature and include dense layers, recurrent layers and convolutional layers. Using the Deep Learning Pipeline, we organize these layers to solve the problem.

Keywords: keyword spotting, speech recognition, deep learning, simple commands speech detection, audio preprocessing.

1. Introducción

Con la omnipresencia de dispositivos móviles y pantallas en la vida diaria, surgen también riesgos relacionados con visión como pérdida de vista, saturación visual y distracción. Las interfaces de audio son una buena manera de reducir el uso de pantallas. Dichas interfaces contienen una variedad de módulos importantes incluyendo uno de Reconocimiento de Voz y un procedimiento de activación llevado a cabo por un módulo de Detección de Comandos.

El problema de Reconocimiento de Voz Automático (RVA) consiste en mapear una señal acústica que represente habla con la secuencia de palabras correspondiente [4]. La Detección de Comandos consiste en localizar palabras de un vocabulario corto en un segmento de audio [26].

Un módulo de RVA podría resolver el problema de Detección de Comandos sin problema. Sin embargo, como se menciona en [29,34], la Detección de Comandos se ejecuta usualmente en dispositivos móviles y surge la necesidad de crear modelos compactos en cuanto a memoria y procesamiento.

TensorFlow ha publicado el *Simple Speech Commands Dataset* [34] para auxiliar en el desarrollo de sistemas de Detección de Comandos. Además del conjunto de datos, la competencia *TensorFlow Speech Recognition Challenge* (2018) [17] fue publicada en Kaggle para alentar a gente de todo el mundo a proponer modelos de Aprendizaje Profundo que puedan resolver este problema.

La estructura de este documento es la siguiente. En la Sección 2 presentamos formalmente el problema de Detección de Comandos y explicamos nuestro acercamiento al desarrollo de arquitecturas. La Sección 3 presenta trabajo relacionado en los campos de Reconocimiento de Voz y Detección de Comandos. La Sección 4 describe nuestro acercamiento a la solución del problema. En la Sección 5, se hace una descripción del conjunto de datos *Simple Speech Commands Detection*. La Sección 6 detalla las arquitecturas seleccionadas para resolver el problema. La Sección 7 presenta nuestra metodología con los resultados presentados en la Sección 8 y, finalmente, discutimos conclusiones y trabajo posterior en la Sección 9.

2. El problema de detección de comandos

En Detección de Comandos, no se busca encontrar la secuencia más probable como sería el caso en RVA, sino que se desea localizar la ocurrencia de algún comando de voz perteneciente a un vocabulario pequeño en una señal de audio [26]. Sea $\mathbf{X} = (x^{(1)}, x^{(2)}, \ldots, x^{(T)})$ la secuencia de entrada, $\mathbf{V} = (v_1, v_2, \ldots, v_3)$ el vocabulario de comandos a detectar y $\mathbf{y} = (y_1, y_2, \ldots, y_N)$ una secuencia de salida que indica cada vez que un comando de $\mathbf{V}$ ha sido pronunciado. En Detección de Comandos, la meta es crear una función f_{DC} que para cada instante de tiempo en la secuencia $\mathbf{y}$ haga una predicción sobre la probabilidad que cada palabra de $\mathbf{V}$ haya sido pronunciada.

Como método de evaluación utilizaremos la métrica de Error *Top-One*, descrita para este problema en [34] que consiste en generar una predicción para un

segmento corto de audio de la probabilidad del habla de cada comando de **V**, otros comandos y silencio. Ésta fue la métrica utilizada en [17] y consecuentemente en la literatura [31,2,11,25].

3. Trabajo relacionado

A partir de los años ochenta y hasta los años 2010-2012 el estado del arte en RVA consistía principalmente en Modelos Ocultos de Markov (MOM) combinados con Modelos de Mezcla Gausiana (MMG). [4]. El campo eventualmente transitó hacia Redes Neuronales a partir de [21,22] donde se utilizó una Máquina de Boltzmann Restringida (MBR) como sustituto al MMG. Otros avances durante estos años [14] incluyeron redes con activaciones ReLU [9] y Dropout [32][35,7], Redes Neuronales Convolucionales (CNN) [28] y Redes Neuronales Recurrentes (RNN) [12].

El problema de Detección de Comandos fue descrito por primera vez en [26]. Al igual que en RVA, los modelos clásicos utilizaron MOM pero el campo transitó hacia el aprendizaje profundo manteniendo énfasis en los modelos compactos. En [29] se proponen varios modelos de CNN. Otros enfoques incluyen [3] que utiliza una Red Neuronal Recurrente Convolutional (CRNN) y [2] que propone un CRNN con unidades Bidireccionales-LSTM y un módulo de atención.

Como resultado del *TensorFlow Speech Recognition Challenge* se publicaron varias propuestas; entre ellas [31,11,25] que utilizan modelos CNN basados en redes propuestas para visión por computadora como VGG [30] y Resnet [13] o las redes compactas de [29]. El primer lugar de la competencia obtuvo un puntaje en exactitud de 91.06 % sin embargo no existen anotaciones disponibles de clase para el conjunto de prueba utilizado en Kaggle por lo que los resultados obtenidos en esta investigación no son directamente comparables.

4. Hacia un sistema *End-to-End* de detección de comandos

Dado el anális del estado del arte, algunas de las preguntas que nos hemos planteado para esta investigación son: (1) ¿Podemos crear un verdadero modelo *End-to-End* utilizando solo capas entrenadas? (2) Para cada tipo de representación, ¿cuál es el *trade-off* entre resolución, exactitud y eficiencia? (3) ¿Qué capas diseñadas son más adecuadas para esta tarea considerando la exactitud y la eficiencia? (4) ¿Qué combinación de capas da el mejor rendimiento en nuestras métricas de evaluación? (5) ¿Cómo manejan los modelos la precisión y la exhaustividad en sus diferentes clases? Para contestar estas preguntas, en esta sección detallamos el modelo propuesto.

4.1. El canal de aprendizaje profundo

Sistemas clasificadores para datos de alta dimensión como el audio, pueden modelarse como un canal muy general (Figura 1) que incluye una etapa de

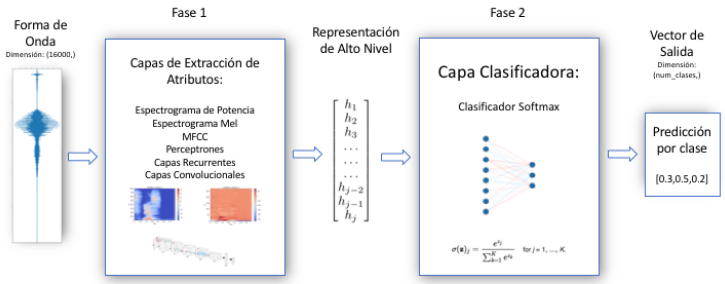

Fig. 1. Nuestro canal de aprendizaje profundo para la clasificación de audio.

extracción de representación (Fase 1) seguida de una etapa de clasificación (Fase 2) que dada la representación obtenida hace una predicción de clase. La Fase 1 obtiene como entrada una forma de onda de audio y sus capas transforman los datos en algo que es más fácil de separar linealmente. Un modelo para la Fase 1 puede ser compuesto de una combinación de capas entrenadas o diseñadas, aunque las capas diseñadas aparecerán invariablemente antes de las capas entrenadas.

Esta sección presenta una visión general de tales capas. Las capas diseñadas incluyen la Transformada de Fourier de Tiempo Reducido (STFT) [23], un Banco de Filtros Mel [8] y la Transformada Discreta de Cosenos (DCT) [8,24]. Las capas entrenadas incluyen las Capas de Perceptrones [27], las Capas Recurrentes [16,15,5] y las Capas Convolucionales [10].

4.2. Las capas diseñadas: Extracción de atributos de audio

Una forma de onda es un arreglo unidimensional de valores de punto flotante dentro del rango $[-1, 1]$. Este arreglo representa una señal en el tiempo $x[n]$ donde n es una marca de tiempo y $x[n]$ es el nivel de amplitud de la onda en el tiempo n. A cada uno de los valores del arreglo se le conoce como muestra. La frecuencia de muestreo f_s representa cuántas muestras deben tomarse en un segundo.

Las formas de onda proporcionan toda la información necesaria para reproducir una señal. Sin embargo, la cantidad de información que podemos extraer fácilmente de ellas es limitada. Esta es la razón por la que es muy común aplicar preprocesamianto a las señales de audio en sistemas de reconocimiento.

Los atributos más comúnmente extraídos de las formas de onda son las representaciones frecuencia-tiempo. Estos incluyen el Espectrograma de Potencia (EP), el Espectrograma Mel (EM) y los Coeficientes Cepstrales de Frecuencia

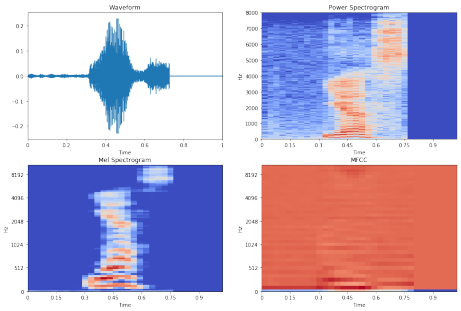

Fig. 2. Cuatro Representaciones de Audio; Forma de Onda (superior izquierda), Espectrograma de Potencia (superior derecha), Espectrograma Mel (inferior izquierda), MFCC (inferior derecha)

Mel (MFCC). En la Figura 2, se presenta una comparación de todas las representaciones mencionadas hasta ahora.

Cuando aplicamos la capa de STFT se obtiene una matriz de valores complejos donde cada columna representa una ventana en el tiempo con valores para cada componente de frecuencia. Tomando la magnitud de cada valor complejo y aplicando una escala logarítmica se obtiene el EP. Al aplicar un Banco de Filtros Mel se obtiene la potencia del EP a través de bandas de frecuencia que están linealmente espaciadas y tienen el mismo ancho de banda en la escala Mel [33], también conocido como el EM. Finalmente, podemos aplicar una DCT al EM para de-correlacionar los vectores obteniendo así el MFCC. En [14] se argumenta que los MFCC eran necesarios cuando se utilizaban MMG, pero para el Aprendizaje Profundo esto ya no es una necesidad.

4.3. Las capas entrenadas: Redes neuronales artificiales

Una red neuronal típica se puede pensar como un extractor de atributos con un clasificador lineal adjunto al final. Para clasificar correctamente, la red debe primero aprender cómo crear alguna representación que sea linealmente separable. Las redes crean representaciones ocultas o codificaciones $\mathbf{h}^{(l)}$, donde l es la profundidad de la capa que generó $\mathbf{h}^{(l)}$. A medida que crezca l, $\mathbf{h}^{(l)}$ debería ser más sencilla de separar.

Las capas de Redes Neuronales Artificiales a utilizar son: (1) Capas de Perceptrones [27] (2) Capas Recurrentes [16] que incluyen LSTM [15] y GRU [5] y (3) Capas Convolucionales [10].

5. El conjunto de datos: Simple Speech Commands Detection

El conjunto de datos *Simple Speech Commands Detection* [34] fue publicado por TensorFlow y está compuesto de varios miles de ejemplos por miles de

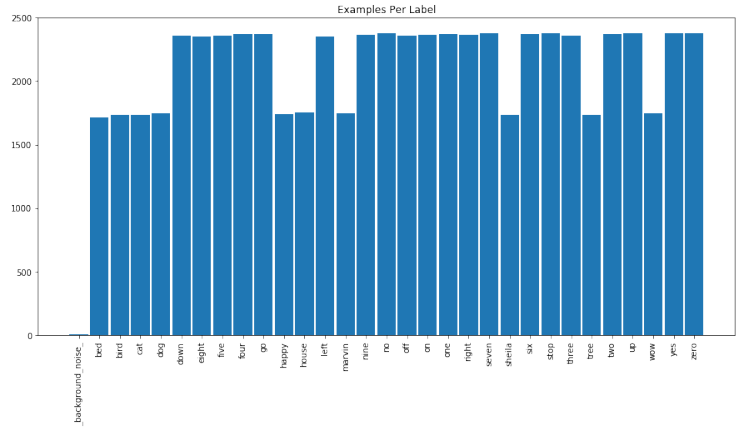

Fig. 3. Distribución del conjunto de datos Simple Commands Speech Detection v0.01. El eje horizontal representa todos los comandos presentes y el eje vertical representa el número de ejemplos para el comando correspondiente.

diferentes oradores de más de 30 palabras y algunas señales de ruido de fondo. El objetivo es crear un modelo que pueda predecir un subconjunto de esta lista de palabras clasificando cualquier otro comando como desconocido o la falta de un comando como silencio.

Existen dos versiones de este conjunto de datos. La primera (v0.01) contiene 64,727 archivos de audio que representan 30 comandos y 6 archivos de ruido de fondo, ésta fue la versión que se utilizó para el *TensorFlow Speech Recognition Challenge*. El conjunto de datos fue eventualmente actualizado a la versión 0.02 que contiene 105,829 archivos de audio que representan 35 comandos y 6 archivos de ruido de fondo. La figura 3 muestra la distribución de archivos para la versión 0.01 del conjunto de datos.

5.1. Subproblemas

Hemos identificado dos versiones del problema. La primera se utilizará para medir el rendimiento general y la segunda para analizar de manera más cercana las métricas de Precisión y Exhaustividad así como la capacidad de los modelos para identificar un comando como *desconocido*.

- Clasificación de 10 palabras (12 clases): *'yes'*, *'no'*, *'up'*, *'down'*, *'left'*, *'right'*, *'on'*, *'off'*, *'stop'* y *'go'*. Cualquier otra palabra deberá ser clasificada como *'desconocido'* y la ausencia de palabras deberá ser clasificada como *'silencio'*. Estos modelos serán entrenados con la versión 0.01 del conjunto de datos.
- Clasificación *"Left/Right"* (3 clases): *'left'*, *'right'* y *'unknown'*. Estos modelos serán entrenados con la versión 0.02 del conjunto de datos.

5.2. Datos de entrada

Los archivos de entrada se presentan en formato WAV con una frecuencia de muestreo de 16,000 Hz y una profundidad de palabra de 16 bits. Los clips duran aproximadamente un segundo pero pueden variar ligeramente, en estos casos se agregan ceros al final de la cadena o se corta para que cada clip tenga una longitud exacta de 16,000 muestras.

5.3. Preprocesamiento de datos

Como parte de la Fase 1 del canal de aprendizaje, queremos comparar cuatro tipos de representaciones de audio. Formas de onda, EP, EM y MFCC obtenidos mediante el uso de las capas diseñadas correspondientes.

Para los EP, EM y MFCC podemos obtener diferentes resoluciones ya sea en la frecuencia o en el dominio del tiempo. Para nuestros experimentos hemos fijado la resolución de tiempo a 32 pasos de tiempo, lo que significa que cada paso de tiempo representa exactamente 31.25 milisegundos de audio.

6. Algoritmos de aprendizaje

Basados en la literatura y los resultados de la competencia de Kaggle, hemos hecho una selección de arquitecturas en cuatro categorías; (1) Perceptrones Multi-Capa, (2) CNN para una dimensión, (3) CNN para dos dimensiones y (4) CRNN.

Dada la necesidad de utilizar modelos compactos, como pauta utilizaremos la restricción colocada en Kaggle que un modelo no debe sobrepasar los 5 MB. Trataremos esta restricción como una restricción suave sujeta al rendimiento de los modelos en las métricas de evaluación. No obstante, no se incluirán las arquitecturas que superen en gran medida esta limitación como es el caso de VGG y Resnet.

Todas las arquitecturas utilizan activaciones ReLU y *Dropout* para evitar sobre-entrenamiento a menos que se especifique de otra manera. En la salida todas las arquitecturas tienen una capa *Softmax* de 12 o 3 unidades (una por cada clase) dependiendo de la versión del problema.

6.1. Perceptrón Multi-Capa

De manera similar a [29], como modelo base entrenaremos un Perceptrón Multi-Capa (PCM) con 3 Capas Ocultas, 128 Unidades Ocultas. Esta red podría aplicarse a formas de onda, pero el tamaño de la entrada hace muy difícil obtener un resultado que sea si quiera cercano a aceptable.

6.2. Redes convolucionales de una dimensión

Hemos elegido el modelo detallado en [19] para formas de onda que se presentó para anotación general de audio al cual nos referimos como *Conv1D*. Ese modelo se compone de cuatro bloques cada uno con dos capas convolucionales (filtros de tamaño [9,3,3,3] respectiamente), *Dropout* y *MaxPooling* (filtros de tamaño [16,4,4,Global]). Seguido de dos capas de Perceptrones con 64 y 128 unidades respectivamente.

6.3. Redes convolucionales de dos dimensiones

Una gran parte de la literatura está centrada alrededor de modelos CNN de dos dimensiones utilizando las representaciones de EP, EM ó MFCC. Por eso la mayoría de nuestros modelos encajan en esta categoría.

Para los modelos CNN de dos dimensiones, nos inspiramos en [29] que explora el uso de CNN compactas. De las arquitecturas presentadas en este trabajo, utilizaremos *cnn-one-fstride4* y *cnn-trad-fpool3*.

Por último, existe una red propuesta en el *Tensor Flow Speech Recognition Challenge* por Thomas O'Malley y que es detallada en [25] y nos referimos a ella como *O'Malley*. La idea principal de esta red es aplicar capas de convolución que afectan a la dimensión de frecuencia pero no a la dimensión temporal. Cuenta con cuatro capas convolucionales y una capa oculta de perceptrones.

6.4. Redes convolucionales recurrentes

Utilizamos tres redes CRNN para EP, EM y MFCC y dos para formas de onda. Todas ellas utilizan capas convolucionales para transformar la entrada antes de ser alimentada a las unidades recurrentes. Las dos primeras se basan en [3] y nos referimos a ellas como CRNN1 y CRNN2. Ambas tienen dos capas convolucionales a la entrada, la primera con 32 filtros de tamaño 5×5 y la segunda con un filtro de tamaño 1×1 seguida de dos capas recurrentes de 32 unidades, la diferencia entre CRNN1 y CRNN2 radica en que CRNN1 utiliza unidades GRU y CRNN2 utiliza unidades LSTM bidireccionales. Utilizamos las inferencias de todas las unidades en el tiempo y alimentamos la salida a una capa de perceptrones con 64 unidades. Una variación de CRNN1 fue creada para manejar las formas de onda utilizando convoluciones de una dimensión en lugar de las convoluciones de dos dimensiones al principio.

El último modelo a utilizar fue tomado de [2] y nos referimos a él como *attRNN*. En él se propone un módulo de atención que indica a la red dónde se producen los datos más importantes. La arquitectura se detalla en el documento mencionado anteriormente. Al igual que con CRNN1, se ha propuesto una versión para forma de onda de esta arquitectura utilizando convoluciones de una dimensión.

7. Metodología

Tenemos un total de 10 arquitecturas de capas entrenadas y 9 representaciones de entrada obtenidas con las capas diseñadas. Sin embargo, no todas las arquitecturas y representaciones coinciden. De las diez arquitecturas, tres sólo funcionan con formas de onda y siete sólo trabajan con EP, EM o MFCC. Esta sección proporciona información sobre las combinaciones aplicadas, el método de evaluación y los pasos seguidos durante la implementación.

7.1. Representaciones de entrada

Usando 4 representaciones y sus variaciones obtenemos 9 representaciones de entrada, todas con una resolución en el tiempo de 32 a excepción de la forma de onda:

- Forma de onda muestreada a 16,000Hz.
- Espectrograma de potencia con resoluciones de frecuencia de 257 y 129.
- Espectrograma Mel con resoluciones de frecuencia de 120, 80 y 40.
- MFCCs con resoluciones de frecuencia de 120, 80 y 40.

7.2. Arquitecturas de redes neuronales

La Sección 6 ha detallado las arquitecturas a utilizar. La siguiente lista las presenta de manera reducida.

- Conv1D: descrita en la sección 6.2 sólo compatible con formas de onda.
- CRNN 1D: descrita en la sección 6.4 sólo compatible con formas de onda.
- attRNN 1D: descrita en la sección 6.4 sólo compatible con formas de onda.
- PMC: descrita en la sección 6.1 compatible con EP, EM y MFCC.
- trad_fpool3: descrita en la sección 6.3 compatible con EP, EM y MFCC.
- one_fstride4: descrita en la sección 6.3 compatible con EP, EM y MFCC.
- CRNN1 2D: descrita en la sección 6.4 compatible con EP, EM y MFCC.
- CRNN2 2D: descrita en la sección 6.4 compatible con EP, EM y MFCC.
- attRNN 2D: descrita en la sección 6.4 compatible con EP, EM y MFCC.
- O'Malley: descrita en la sección 6.3 compatible con EP, EM y MFCC.

7.3. Implementación

Tomando las combinaciones descritas anteriormente, tenemos un total de 59 redes. Cada red fue programada usando la API de Keras [6] junto con TensorFlow [1]. Las capas diseñadas que incluyen; la STFT, el Banco de Filtros Mel y la DCT fueron creadas usando el paquete Librosa [20].

Cada modelo fue entrenado por 30 épocas con una tasa de aprendizaje de $1e-3$ y el algoritmo de optimización *Adam* [18]. La única excepción fueron los Perceptrones Multicapa que se entrenaron por 100 épocas cada uno. Las redes

fueron entrenadas en un sistema de escritorio usando una GPU GTX 970 con 3.5 GB de Memoria.

Para evaluar los modelos se utilizó validación cruzada de 10 pliegues. Dado que hay demasiados ejemplos para la clase "desconocido", muestreamos aleatoriamente el 20 % de los ejemplos y los mantuvimos consistentes a través de las 10 particiones. Para todas las separaciones, nos aseguramos de que ningún hablante apareciera a través de las particiones, asegurando así que los modelos sean independientes del hablante.

8. Resultados

Experimento 1: 10 Palabras Las Tablas 1 y 2 muestran los resultados de exactitud media, desviación estándar y tamaño del modelo para el problema "10 palabras". En la Tabla 1 se presentan los resultados de los modelos para formas de onda, hemos destacado en negrita el modelo con la máxima exactitud media. En la Tabla 2 se presentan los resultados de las representaciones del EP, el EM y el MFCC. Para cada arquitectura hemos puesto en negritas la representación con la máxima exactitud media.

Tabla 1. Resultados de modelos entrenados con formas de onda.

	Conv1D	CRNN1D	attRNN 1D
Exactitud WF-16k	**93.08 %**	91.61 %	92.83 %
Des. Est. WF-16k	**1.59 %**	1.51 %	1.64 %
Memoria WF-16k	**3.2 MB**	1.41MB	2.4 MB

En respuesta a la pregunta de investigación (1) dados los resultados de la Tabla 1 podemos ver claramente que se pueden lograr resultados aceptables sin ninguna capa diseñada, Conv1D logró una exactitud promedio de 93.08 % y un tamaño de memoria muy por debajo de la restricción de 5 MB. CRNN 1D y attRNN logran resultados similares y por lo tanto concluimos que los verdaderos sistemas de aprendizaje profundo *End-to-End* por lo menos para la Detección de Comandos son más que plausibles.

En cuanto a las preguntas (2) y (3) en la Tabla 2, podemos ver que el PMC, trad_fpool3 y on_fstride4 funcionan mejor con representaciones más simples y sin necesidad de los vectores de-correlacionados del MFCC. Es por eso que la representación EM-40 domina claramente en estos ejemplos. Sin embargo, la diferencia se vuelve menos clara con las redes CRNN1 2D, CRNN2 2D, attRNN 2D y O'Malley donde las representaciones más grandes obtienen una ligera ventaja sobre EM-40 y los resultados obtenidos con MFCCs son muy similares a los obtenidos con EM.

Teniendo en cuenta estos resultados, nuestra conclusión es que para sistemas compactos, es probablemente mejor utilizar EM-40, pero pequeñas mejoras podrían lograrse en ciertas redes utilizando representaciones más grandes como

Tabla 2. Resultados de models entrenados con EM, MFCC y EP.

	PMC	trad_fpool3	one_fstride4	CRNN1 2D	CRNN2 2D	attRNN 2D	O'Malley
Exactitud EM-40	**77.75 %**	**87.71 %**	**84.18 %**	**89.62 %**	90.13 %	93.36 %	94.94 %
Des. Est. EM-40	**3.44 %**	**3.72 %**	**3.60 %**	**2.45 %**	2.55 %	2.48 %	2.07 %
Memoria EM-40	**2.4 MB**	**3.0 MB**	**1.5 MB**	**556.1 kB**	1.2 MB	2.3 MB	17.0 MB
Exactitud EM-80	74.93 %	85.94 %	82.44 %	89.39 %	90.15 %	92.55 %	95.19 %
Des. Est. EM-80	5.04 %	3.54 %	4.73 %	2.43 %	2.79 %	2.56 %	2.15 %
Memoria EM-80	4.4 MB	2.8 MB	2.4 MB	579.1 kB	1.3 MB	2.5 MB	18.9 MB
Exactitud EM-120	73.01 %	85.89 %	81.11 %	88.72 %	89.70 %	91.48 %	94.77 %
Des. Est. EM-120	5.32 %	3.54 %	5.58 %	3.15 %	2.94 %	2.57 %	2.17 %
Memoria EM-120	6.4 MB	3.1 MB	2.7 MB	602.2 kB	1.4 MB	2.8 MB	18.9 MB
Exactitud MFCC-40	73.58 %	81.90 %	76.32 %	71.75 %	68.74 %	90.00 %	93.82 %
Des. Est. MFCC-40	4.11 %	1.58 %	2.29 %	2.51 %	3.95 %	1.37 %	1.41 %
Memoria MFCC-40	2.4 MB	3.0 MB	1.5 MB	556.1 kB	1.2 MB	2.3 MB	17.0 MB
Exactitud MFCC-80	69.65 %	79.48 %	74.70 %	70.25 %	83.95 %	92.85 %	**95.36 %**
Des. Est. MFCC-80	6.84 %	1.53 %	3.33 %	2.88 %	3.58 %	2.50 %	**1.85 %**
Memoriae MFCC-80	4.4 MB	2.8 MB	2.4 MB	579.1 kB	1.3 MB	2.5 MB	**18.9 MB**
Exactitud MFCC-120	64.51 %	83.46 %	77.66 %	85.41 %	84.72 %	92.47 %	94.84 %
Des. Est. MFCC-120	6.70 %	5.46 %	8.29 %	2.74 %	4.53 %	2.39 %	2.05 %
Memoria MFCC-120	6.4 MB	3.1 MB	2.7 MB	602.2 kB	1.4 MB	2.8 MB	18.9 MB
Exactitud EP-129	57.60 %	81.60 %	78.12 %	89.33 %	**90.33 %**	**93.37 %**	95.31 %
Des. Est. EP-129	3.41 %	3.40 %	5.12 %	1.61 %	**2.15 %**	**2.14 %**	2.05 %
Memoria EP-129	6.8 MB	3.2 MB	2.7 MB	607.9 kB	**1.4 MB**	**2.8 MB**	18.9 MB
Exactitud EP-257	50.05 %	82.43 %	77.19 %	89.56 %	90.31 %	92.83 %	95.34 %
Des. Est. EP-257	3.84 %	3.84 %	5.40 %	2.13 %	3.03 %	2.26 %	1.95 %
Memoria EP-257	13.1 MB	4.7 MB	3.5 MB	681.6 kB	1.6 MB	3.6 MB	18.9 MB

EM-80. EM-120 no parece dar ninguna ventaja adicional más allá de EM-80. Lo mismo parece ser válido para el MFCC. En cuanto a la comparación entre EM-120, MFCC-120 y EP-129 la respuesta parece depender del modelo, pero ninguna representación tiene una victoria clara en ese sentido.

En cuanto a la pregunta (4) el mejor resultado de exactitud promedio en general se obtuvo usando MFCC-80 y O'Malley, pero otras arquitecturas producen resultados muy similares como O'Malley y EM-80, O'Malley y EP-129 y O'Malley y EP-257 por lo que podrían ser intercambiables. Si aplicamos la restricción de 5 MB, el modelo con la mayor precisión media es attRNN 2D y EP-129 pero al igual que con O'Malley, este resultado es prácticamente idéntico al obtenido con attRNN y EM-40.

Experimento 2: *Left/Right* Las Tablas 3 y 4 presentan la precisión y exhaustividad medias y el tamaño en memoria para cada una de las tres clases en este problema. Debido a su incompatibilidad una vez más hemos separado ciertas arquitecturas y representaciones. También hemos destacado en ambas tablas la mejor precisión y exhaustividad para cada clase y representación.

En la Tabla 4 vemos en general resultados que parecen ser coherentes con los resultados del experimento anterior. O'Malley y attRNN 2D aún poseen las puntuaciones más altas en las métricas seleccionadas.

Si contamos el número de victorias y empates para cada representación en cada uno de los seis parámetros de la Tabla 4 (Precisión y Exahustividad para las tres clases) EP-127 con 12 victorias y 1 empate y EM-40 con 11 victorias

Tabla 3. Precisión y Exhaustividad para cada clase del problema *Left/Right* con formas de onda.

	Conv1D	CRNN 1D	AttRNN 1D
Prec/Exh 'Left'	**95.3 %/93.1 %**	95.1 %/89.1 %	94.5 %/91.6 %
Prec/Exh 'Right'	94.7 %/**93.5 %**	95.1 %/90.1 %	**95.5 %**/92.6 %
Prec/Exh 'Desconocido'	**97.8 %**/98.5 %	96.7 %/98.7 %	97.4 %/**98.6 %**

Tabla 4. Precisión y Exhaustividad para cada clase del problema *Left/Right* con EM, MFCC y EP.

		PMC	trad_fpool3	one_fstride4	CRNN1 2D	CRNN2 2D	AttRNN 2D	O'Malley
		Prec/Exh	Prec/Exh	Prec/Exh	Prec/Exh	Prec/Exh	Prec/Exh	Prec/Exh
MS-40	'Left'	86.7 %/**79.7 %**	88.1 %/**88.5 %**	86.5 %/**83.3 %**	92.0 %/89.5 %	91.2 %/**91.1 %**	**94.8 %**/91.6 %	95.8 %/94.8 %
	'Right'	86.1 %/**84.1 %**	91.3 %/87.8 %	**89.1 %**/**83.5 %**	94.3 %/**92.7 %**	94.4 %/91.5 %	95.7 %/92.9 %	95.7 %/95.3 %
	'Desc'	**94.4 %**/96.3 %	96.2 %/96.8 %	**94.8 %**/96.6 %	**97.1 %**/97.9 %	97.2 %/97.8 %	97.5 %/**98.6 %**	**98.4 %**/98.6 %
MS-80	'Left'	86.5 %/77.0 %	89.4 %/85.0 %	84.8 %/81.3 %	91.0 %/88.9 %	92.3 %/89.3 %	92.3 %/91.4 %	95.3 %/94.3 %
	'Right'	**84.9 %**/80.3 %	89.8 %/87.5 %	87.0 %/82.4 %	94.3 %/90.3 %	94.0 %/91.1 %	95.8 %/91.8 %	97.7 %/94.9 %
	'Desc'	93.2 %/96.1 %	95.6 %/96.9 %	94.2 %/95.8 %	96.6 %/97.8 %	96.8 %/98.0 %	97.3 %/98.2 %	98.1 %/98.8 %
MS-120	'Left'	**86.8 %**/74.6 %	87.0 %/85.5 %	85.2 %/78.8 %	91.4 %/89.3 %	90.3 %/88.7 %	90.9 %/91.5 %	95.4 %/94.7 %
	'Right'	84.0 %/78.9 %	90.1 %/85.5 %	86.9 %/81.9 %	93.1 %/91.0 %	94.0 %/90.1 %	94.5 %/91.6 %	97.9 %/94.6 %
	'Desc'	92.6 %/96.2 %	95.4 %/96.7 %	93.8 %/96.1 %	96.8 %/97.6 %	96.6 %/97.7 %	97.4 %/97.8 %	98.1 %/98.9 %
MFCC-40	'Left'	71.3 %/77.2 %	89.3 %/86.4 %	85.3 %/78.6 %	90.8 %/84.4 %	88.2 %/85.6 %	94.3 %/93.3 %	96.1 %/**94.9 %**
	'Right'	74.2 %/77.7 %	89.6 %/**90.0 %**	86.5 %/82.3 %	93.8 %/86.8 %	92.3 %/89.3 %	95.7 %/94.3 %	**98.0 %**/94.7 %
	'Desc'	94.2 %/91.8 %	**96.4 %**/96.9 %	94.1 %/96.3 %	95.5 %/98.1 %	95.9 %/97.1 %	97.9 %/98.5 %	98.2 %/99.0 %
MFCC-80	'Left'	65.5 %/70.5 %	88.8 %/85.2 %	85.9 %/75.4 %	89.7 %/86.0 %	88.5 %/87.3 %	91.5 %/92.3 %	96.8 %/94.8 %
	'Right'	65.1 %/74.1 %	**91.6 %**/87.5 %	84.5 %/79.7 %	93.6 %/88.5 %	92.8 %/88.5 %	94.8 %/92.6 %	97.7 %/94.8 %
	'Desc'	93.2 %/89.4 %	95.6 %/97.1 %	93.0 %/96.1 %	95.9 %/97.6 %	96.2 %/97.3 %	97.5 %/97.7 %	98.2 %/**99.2 %**
MFCC-120	'Left'	63.2 %/67.0 %	89.1 %/81.9 %	86.7 %/72.4 %	90.8 %/85.5 %	90.1 %/85.6 %	93.3 %/90.5 %	94.3 %/94.2 %
	'Right'	61.3 %/64.3 %	89.1 %/87.2 %	85.2 %/79.4 %	93.3 %/88.8 %	91.4 %/89.2 %	94.3 %/92/7 %	97.1 %/95.4 %
	'Desc'	92.1 %/90.1 %	95.2 %/97.0 %	92.4 %/96.5 %	95.8 %/97.7 %	96.1 %/97.4 %	97.3 %/98.1 %	98.3 %/98.5 %
PS-127	'Left'	83.0 %/64.5 %	**89.6 %**/84.9 %	**87.6 %**/74.3 %	**94.0 %**/89.9 %	**93.3 %**/90.7 %	94.4 %/**93.4 %**	96.3 %/94.8 %
	'Right'	80.3 %/71.6 %	90.6 %/85.9 %	86.5 %/79.6 %	94.3 %/91.9 %	93.7 %/**93.3 %**	95.7 %/**94.4 %**	96.6 %/**96.3 %**
	'Desc'	90.5 %/96.2 %	95.5 %/**97.4 %**	92.6 %/**96.7 %**	97.0 %/98.2 %	**97.4 %**/98.0 %	**98.0 %**/98.4 %	**98.4 %**/98.8 %
PS-257	'Left'	70.9 %/58.8 %	88.4 %/83.3 %	83.9 %/78.2 %	93.6 %/**90.4 %**	92.8 %/89.7 %	94.4 %/92.6 %	**97.4 %**/94.0 %
	'Right'	76.7 %/60.8 %	88.7 %/84.2 %	**89.1 %**/75.7 %	**95.0 %**/91.7 %	**94.9 %**/90.0 %	**96.5 %**/93.4 %	97.5 %/95.4 %
	'Desc'	89.3 %/**95.5 %**	94.9 %/96.8 %	92.8 %/96.6 %	97.0 %/**98.3 %**	96.8 %/**98.2 %**	97.6 %/**98.6 %**	98.2 %/**99.2 %**

y 3 empates tienen los mejores resultados. PS-257 con 8 victorias y 3 empates vendría después.

En respuesta a la pregunta de investigación (5), podemos ver que en la mayoría de los casos, la precisión es mejor que la exhaustividad para los comandos *Left/Right*, pero para los comandos *"desconocido"* la exhaustividad es generalmente mejor que precisión. Esto sucede porque cada vez que los algoritmos predicen un comando *Left/Right* como positivo suelen ser correctos (de ahí la alta precisión), pero hay algunos comandos que deben ser clasificados como positivos que se clasifican como *"desconocido"* (esta es la razón por la que la precisión no es tan buena para *"desconocido"* y la exhaustividad no es tan buena para los comandos).

9. Conclusiones

Hemos presentado un análisis de varias arquitecturas y representaciones posibles para el Problema de Detección de Comandos con un énfasis en sistemas *End-to-End* compactos. También hemos presentado un Canal de Aprendizaje

Profundo enfocado en extraer mejores representaciones para el problema usando una mezcla de capas diseñadas y entrenadas.

Se ha observado también que ciertas CNN compactas pueden ser eficaces con las representaciones más pequeñas pero sufren al aumentar el tamaño de los datos. Por otro lado, los CRNN en general utilizan menos recursos de memoria y presentan mejoras al aumentar la resolución de frecuencia.

Asegurarse de que los hablantes no aparezcan en más de una partición del conjunto de datos y tener muchos equipos de grabación diferentes nos ayudó a hacer un modelo más robusto. Sin embargo, estos resultados se pueden mejorar si se terminan de entrenar con ejemplos específicos del usuario.

Nos hemos centrado en la Detección de Comandos para dispositivos móviles en este documento, pero muchos de estos principios pueden aplicarse a la anotación de audio en general y trabajo futuros podría hacer una exploración en esta línea.

Referencias

1. Abadi, M., et al.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), https://www.tensorflow.org/, software available from tensorflow.org
2. de Andrade, D.C., Leo, S., Viana, M.L.D.S., Bernkopf, C.: A neural attention model for speech command recognition. In: undefined (2018), http://arxiv.org/abs/1808.08929
3. Arik, S.O., et al.: Convolutional recurrent neural networks for small-footprint keyword spotting. In: Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH. vol. 2017-Augus, pp. 1606–1610 (mar 2017)
4. Bengio, Y., Goodfellow, I., Courville, A.: Applications: Speech Recognition, pp. 446–448 (2016), https://mitpress.mit.edu/books/deep-learning
5. Cho, K., et al.: Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) pp. 1724–1734 (2014)
6. Chollet, F., et al.: Keras. https://keras.io (2015)
7. Dahl, G.E., Sainath, T.N., Hinton, G.E.: Improving deep neural networks for LVCSR using rectified linear units and dropout. In: ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings. pp. 8609–8613. IEEE (may 2013)
8. Davis, S.B., Mermelstein, P.: Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences (aug 1980)
9. Fred Agarap, A.M.: Deep Learning using Rectified Linear Units (ReLU). Tech. rep., https://github.com/AFAgarap/relu-classifier.
10. Fukushima, K.: Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics 36(4), 193–202 (apr 1980)
11. Gouda, S.K., Kanetkar, S., Harrison, D., Warmuth, M.K.: Speech Recognition: Keyword Spotting Through Image Recognition (mar 2018), http://arxiv.org/abs/1803.03759
12. Graves, A., Mohamed, A.R., Hinton, G.: Speech recognition with deep recurrent neural networks. In: ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings. pp. 6645–6649. IEEE (may 2013)

13. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition pp. 770–778 (2015), http://arxiv.org/abs/1512.03385
14. Hinton, G., et al.: Deep Neural Networks for Acoustic Modeling in Speech Recognition. IEEE Signal Processing Magazine 29(November), 82–97 (nov 2012), https://www.microsoft.com/en-us/research/publication/deep-neural-networks-for-acoustic-modeling-in-speech-recognition/
15. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Computation 9(8), 1735–1780 (nov 1997)
16. Hopfield, J.J.: Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences of the United States of America 79(8), 2554–8 (apr 1982)
17. Kaggle: Tensorflow speech recognition challenge (2018), kaggle Challenge, https://www.kaggle.com/c/tensorflow-speech-recognition-challenge/
18. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization (dec 2014), http://arxiv.org/abs/1412.6980
19. Mahmood, Z.: Beginner's guide to audio data (2018), python notebook posted in Freesound General-Purpose Audio Tagging Challenge (2018) https://www.kaggle.com/fizzbuzz/beginner-s-guide-to-audio-data
20. McFee, B., Raffel, C., Liang, D., Ellis, D., McVicar, M., Battenberg, E., Nieto, O.: librosa: Audio and Music Signal Analysis in Python. In: Proceedings of the 14th Python in Science Conference. pp. 18–24 (2018)
21. Mohamed, A.r., Dahl, G., Hinton, G.: Deep belief networks for phone recognition. Scholarpedia 4(5), 5947 (2009)
22. Mohamed, A.R., Dahl, G.E., Hinton, G.: Acoustic modeling using deep belief networks. IEEE Transactions on Audio, Speech and Language Processing 20(1), 14–22 (jan 2012)
23. Müller, M.: Fourier Analysis of Signals: Short-Time Fourier Transform, pp. 93–105. Springer International Publishing, Cham (2015), http://link.springer.com/10.1007/978-3-319-21945-5
24. Narasimha, M.J., Peterson, A.M.: On the Computation of the Discrete Cosine Transform. IEEE Transactions on Communications 26(6), 934–936 (jun 1978)
25. O'Malley, T.: Second place solution (2018), discussion posted in TensorFlow Speech Recognition Challenge (2018) https://www.kaggle.com/c/tensorflow-speech-recognition-challenge/discussion/47715
26. Rohlicek, J., et al.: Continuous hidden Markov modeling for speaker-independent word spotting. In: International Conference on Acoustics, Speech, and Signal Processing. pp. 627–630. IEEE (1989)
27. Rumelhart, David E. Hinton, G.E., Williams, R.J.: Learning Internal Representations by Error Propagation. In: Parallel Distributed Processing Explorations in the Microstructure of Cognition Volume 1 Foundations, vol. 1, p. 567. MIT Press (1986), https://ieeexplore.ieee.org/document/6302929
28. Sainath, T.N., Mohamed, A.R., Kingsbury, B., Ramabhadran, B.: Deep convolutional neural networks for LVCSR. In: ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings. pp. 8614–8618. IEEE (may 2013)
29. Sainath, T.N., Parada, C.: Convolutional Neural Networks for Small-footprint Keyword Spotting. In: Proceedings INTERSPEECH. pp. 1478–1482 (2015), https://storage.googleapis.com/pub-tools-public-publication-data/pdf/43969.pdf
30. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition (sep 2014), http://arxiv.org/abs/1409.1556

31. Solovyev, R.A., Vakhrushev, M., Radionov, A., Aliev, V., Shvets, A.A.: Deep Learning Approaches for Understanding Simple Speech Commands. Tech. rep. (oct 2018), http://arxiv.org/abs/1810.02364
32. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15, 1929–1958 (2014), http://jmlr.org/papers/v15/srivastava14a.html
33. Stevens, S.S., Volkmann, J.: The Relation of Pitch to Frequency: A Revised Scale. The American Journal of Psychology 53(3), 329 (jul 1940)
34. Warden, P.: Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition (apr 2018), http://arxiv.org/abs/1804.03209
35. Zeiler, M.D., et al.: On rectified linear units for speech processing. In: ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings. pp. 3517–3521. IEEE (may 2013)