Proyecto 2: Reconocimiento de comandos de voz mediante espectrogramas y aprendizaje profundo

Daniel A. Rojas, *Estudiante, ITCR*
Christopher R. Russell, *Member*

***Abstract*—As an exercise in familiarization with deep learning techniques, a convolutional neural network was programmed with the Keras library for Python and trained with the TensorFlow voice command sample set to recognize selected spoken words. The model was observed to converge well during training, and was successful in recognizing commands from a newly-recorded data set.**

***Index Terms*—Speech Recognition, Deep Learning, Convolutional Neural Network**

I. Introducción

La civilización humana esta produciendo datos a un ritmo sin precedentes. Esto es a la vez un resultado de y un impulso para el desarrollo de la tecnología informática. La cantidad de datos disponibles en el mundo es demasiado grande para que los analice un solo ser humano, o incluso grupos de seres humanos. El uso de técnicas de inteligencia artificial es esencial para convertir estos datos en información sobre la que se pueda actuar. No solo es imposible manejar la gran cantidad de datos, sino que la complejidad de esos datos también suele es mas allá de la capacidad humana de comprender, con relaciones que pueden ni siquiera ser aparentes. por esta razón, el aprendizaje automático, y en particular las técnicas de aprendizaje profundo con su capacidad para extraer características de grandes cantidades de datos no estructurados sin que los humanos les indiquen específicamente como hacerlo, hace que el aprendizaje profundo sea muy poderoso para grandes problemas.

Un ejemplo de la aplicabilidad del aprendizaje automático es el problema del automóvil autónomo. Si uno le preguntara a un ser humano exactamente como analiza el campo visual mientras orienta el automóvil en la carretera, no podría hacerlo mas que en los términos mas simples. Pregúnteles como reconocen un peligro en la carretera y lo distinguen de todo lo que ven en ese momento, y su única respuesta seria que ”simplemente lo hacen”. Si bien los problemas de los automóviles autónomos pueden estar actualmente mas allá de las capacidades de la inteligencia artificial, serian completamente imposibles de superar con inteligencia programada utilizando algoritmos generados por humanos, ya que los humanos no somos lo suficientemente conscientes de nuestros propios procesos de pensamiento y aprendizaje.

Sin embargo, el aprendizaje automático no es prometedor solo para vehículos autónomos. ya se utiliza a diario en aplicaciones de inteligencia empresarial y minería de datos, encontrando relaciones entre variables hasta ahora desconocidas. Se utiliza en la publicidad dirigida para que los proveedores puedan inferir que es probable que un cliente potencial este interesado en comprar en funcion no solo de su historial de compras, sino también de sus hábitos generales en el web. Quizás lo mas importante y aterrador es que también se utiliza en la prevención del delito y la lucha contra el terrorismo a través del análisis automatizado de datos de seguridad y vigilancia.

Una breve revisión de unos conceptos importantes utilizados en este proyecto es util.

A. Transformada de Fourier de Corto Plazo

La transformada de Fourier de corto plazo (”Short-Time Fourier Transform”, STFT) es un método para estimar la densidad espectral de señales no periódicas. Consiste en dividir la señal muestreada en segmentos de tiempo y calcular la transformada de Fourier de cada uno como si fueran periódicos. Tratarlos como periódicos provoca efectos de discontinuidad en los bordes del segmento. El efecto de estas discontinuidades se conocen como ”fuga espectral” y aparecen como errores en la estimación espectral. Estos efectos se pueden reducir, pero no eliminar, seleccionando una función de ventana que quita énfasis a las muestras cerca de los bordes del segmento.

B. Espectrogramas

Para muestras mas largas, la estimación espectral de cada segmento puede mostrarse en un imagen bidimensional, mostrando el cambio en el uso espectral a lo largo del tiempo. Los gráficos de este tipo se conocen como espectrogramas.

En un espectrograma, uno de los ejes representa el espectro de frecuencias y el otro representa una escala de tiempo. Cada linea de la imagen es la estimación del espectro de un segmento de la señal muestreada, con el contenido de energía del contenedor representado por un valor (intensidad o tono) de un píxel. El ”apilamiento” de cada linea en la otra dirección brinda una representación visual del cambio en la densidad espectral a lo largo de la duración de la muestra. El uso de segmentos superpuestos en lugar de contiguos tiene un efecto de ”filtrado”, lo que reduce la visibilidad de las discontinuidades en el espectrograma.

C. Escalas de Mel

La escala de Mel es la transformación de una escala lineal de frecuencia a una logarítmica. Es diseñado para imitar la percepción del tono del oído humano. Así como el oído tiene una sensibilidad logarítmica a los cambios de amplitud, su

sensibilidad a los cambios de tono también es logarítmica. para aplicaciones como el reconocimiento de voz, la escala de Mel se usa típicamente para redistribuir la densidad espectral de un algoritmo de frecuencia lineal como el STFT en una distribución que mapea mas de cerca la respuesta del oído, de modo que los algoritmos no sean demasiado sensibles a información de tono que no es distinguible por los humanos.

D. Conceptos básicos sobre aprendizaje profundo

El aprendizaje profundo es una rama de la inteligencia artificial y el aprendizaje automático que busca emular el proceso de aprendizaje humano. Así como los humanos aprenden típicamente en un proceso de prueba y error, un sistema de aprendizaje profundo utiliza un proceso conocido como ”entrenamiento”, donde primero se le suministran conjuntos seleccionados de entradas que representan varios ejemplos del estado deseado que se aprenderá, junto con entradas representando contraejemplos que no representan el estado deseado. Con cada entrada, el sistema recibe la etiqueta verdadera del estado de entrada, ya sea deseado o no deseado. A través de capas de análisis de las entradas, de las que la técnica adquiere su nombre ”profundo”, el sistema ”aprende” que características indican los estados deseados y no deseados sin que se le diga explícitamente.

Al igual que con el aprendizaje humano, los resultados iniciales del entrenamiento pueden ser relativamente pobres. Durante cada entrenamiento repetido, la salida del sistema se conoce como una ”predicción”, ya que predice el etiquetado correcto de la entrada en función de lo que ya ha aprendido. La comparación de la predicción con el etiquetado verdadero proporciona una retroalimentación, que el sistema puede usar como métrica para un algoritmo de optimización para ajustar los parámetros de sus capas internas para mejorar la exactitud de sus predicciones. Para hacer esto, el optimizador busca minimizar la métrica de perdida del modelo a medida que itera. La métrica de perdida es una función del error estadístico que refleja cuan indeseable es un determinado error. Por ejemplo, si un amigo confunde la declaración de uno con ”lunch is at my place”, eso podría generar vergüenza. Por otro lado, malinterpretando la misma afirmación por ”punch me in the face”, los resultados serian mucho peores, a pesar de que ambos son errores.

E. Redes Neuronales Convolucionales

Las redes neuronales convolucionales son una subclase de redes neuronales artificiales, que se basan en estructuras diseñadas para imitar las estructuras biológicas del cerebro. El elemento básico de una red neuronal es la neurona artificial. Cada neurona tiene conexiones con otras neuronas, lo que le permite recibir señales, procesarlas y transmitir los resultados a otras neuronas. De esta manera, las redes de neuronas artificiales se parecen vagamente a las estructuras de la neuronas biológicas en el cerebro.

En lugar de potenciales eléctricos, las neuronas artificiales utilizan valores numéricos. Cada entrada y salida tiene una ponderación asociada que escala la señal en las conexiones. Los valores de salida de la neurona son una función de la suma de las entradas. Las salidas se transmiten solo si la suma calculada supera el valor umbral para la neurona.

Las neuronas generalmente se organizan en capas, y cada capa realiza diferentes cálculos de suma en sus entradas. En tales sistemas, las señales se propagan a través del sistema desde una capa de entrada, a través de varias capas, hasta que alcanzan una capa de salida. En las redes neuronales convolucionales, las capas están organizadas de manera que todas las neuronas de una capa están conectadas a todas las de la siguiente capa. Como sugiere el nombre, la aplicación realizada por cada neurona en una red neuronal convolucional es una convolución de las entradas. la convolución permite una reducción significativa en el numero de entradas a través del escalado y la extracción de características dominantes, y una reducción correspondiente en la potencia informática requerida para el análisis. Las redes convolucionales son particularmente adecuado para aplicaciones que involucran imágenes bidimensionales, como los espectrogramas utilizados en este proyecto.

II. METODOLOGÍA

El sistema consta de una red neuronal convolucional entrenada para reconocer palabras habladas seleccionadas del conjunto de datos de comandos de voz de TensorFlow [10]. las palabras seleccionadas para el entrenamiento fueron ”down”, ”go”, ”left”, ”no”, ”off”, ”on”, ”right”, ”stop”, ”up”, y ”yes”. Todas las demás palabras en el conjunto de datos de TensorFlow se etiquetaron como ”unknown” para fines de entrenamiento. Además, se produjeron muestras de ruido consistentes en sonidos de fondo y ruido blanco al dividir grabaciones mas largas en intervalos de 1 segundo. Los tamaños de estos conjuntos se resumen en la Fig 1 y la Fig 2.

Down	2095
Go	2112
Left	2106
No	2105
Off	2101
On	2110
Right	2111
Stop	2134
Up	2115
Yes	2116
Unknown	2736
Background	6471

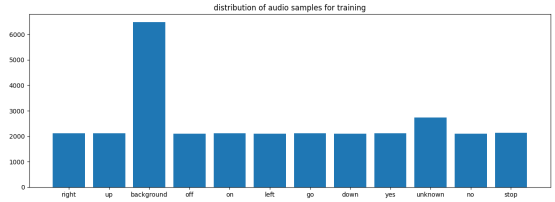


Fig. 1. Distribución de clases en el conjunto de entrenamiento

Además, se produjo un conjunto de muestras de comprobación que consiste en 10 muestras de cada clase pronunciadas

Down	264
Go	260
Left	247
No	270
Off	256
On	257
Right	256
Stop	246
Up	260
Yes	261
Unknown	2737
Background	657

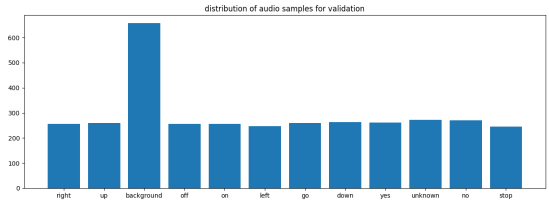


Fig. 2. Distribución de clases en el conjunto de validación

por el mismo orados, con las muestras de prueba desconocidas restantes del conjunto de TensorFlow. Los tamaños del conjuntos de comprobación se resumen en la Fig 3

Down	10
Go	10
Left	10
No	10
Off	10
On	10
Right	10
Stop	10
Up	10
Yes	10
Unknown	10
Background	68

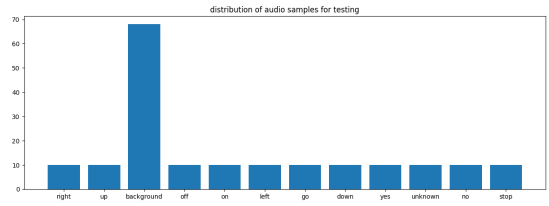


Fig. 3. Muestras en el conjunto de comprobación

Para la entrada a la CNN, las muestras de audio primero se recortaron o se rellenaron con ceros según se requería hasta un segundo de longitud, y se produjeron imágenes de espectrograma doble-logarítmico con las funciones melspectrogram y power_to_db del modulo librosa [11] para análisis de audio con Python. En la Fig 5 se muestra un espectrograma de muestra del conjunto de validación de la palabra ”right” con su forma de onda correspondiente en la Fig 4. Tenga en cuenta que los ejes y las etiquetas son ilustrativos y no están incluidos en el espectrograma utilizado internamente.

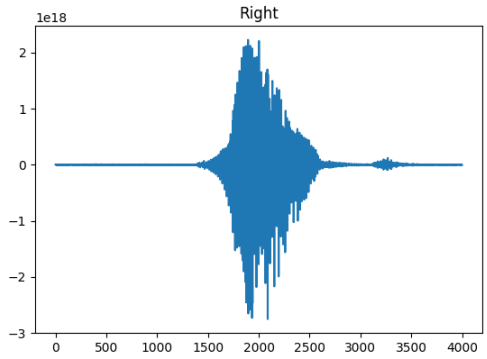


Fig. 4. Ejemplo de forma de onda de la palabra ”right”

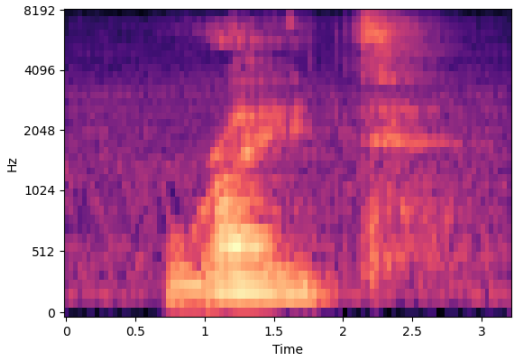


Fig. 5. Ejemplo de espectrograma de la palabra ”right”

Las imágenes en escala de grises mas similares al formato interno real se muestran en la Fig 6, con varios ejemplos de los espectrogramas de muestras de la palabra ”right”. En la Fig 7, se muestra un espectrograma de ejemplo de cada clase.

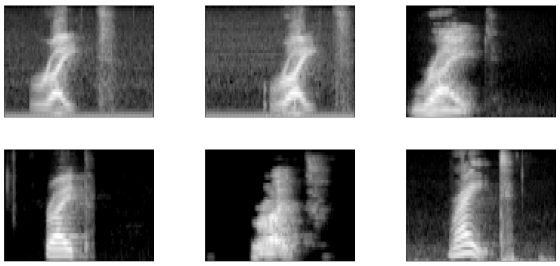


Fig. 6. Comparación de espectrogramas de la palabra ”right”

El histograma de píxeles calculado sobre los conjuntos de entrenamiento y validación se calculo utilizando técnicas de procesamiento de imágenes en los espectrogramas guardados como archivos png, en lugar del formato de objeto serializado utilizado para el entrenamiento. Sin embargo, en la Fig 8 se muestra la suavidad esperada de la curva de probabilidad, además de un pico en un solo bin que se sospecha que se

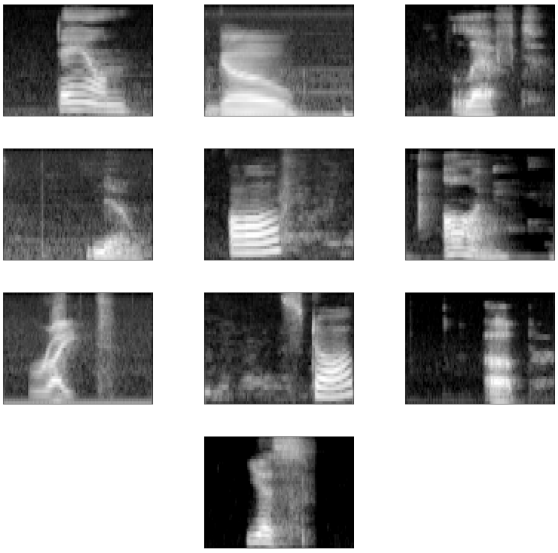


Fig. 7. Comparación de espectrogramas de las distintas clases

debe a algún defecto desconocido tal vez en las muestras de ruido de fondo generadas.

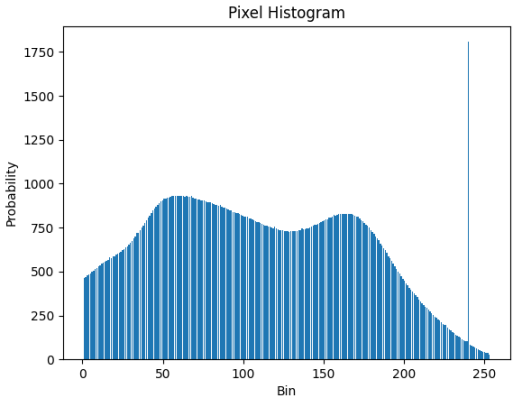


Fig. 8. Histograma de valores de los espectrogramas

La red convolucional se construyo utilizando el modulo Keras para Python [5]. la estructura de capas fue la misma que la recomendad por el profesor, y su implementación en Python se muestra en la Fig 9

El funcionamiento de la estructura de la red neuronal esta fuera del alcance de esta clase, pero es posible hacer algunas generalizaciones:

Las operaciones repetidas que siguen la capa de Input son operaciones de extracción de características:

- Las capas Conv2D representan una operación de filtrado convolucional bidimensional.
- Las capas BatchNormalization son una normalización de valores para mantener una salida dentro de un rango aceptable (media cercana a 0 y desviación estándar cercana a 1)
- La capa ReLU es una función de rectificación que descarta valores negativos.

```
model = Sequential()
model.add(InputLayer(input_shape=\
  x_train.shape[1:]))
model.add(Conv2D(12,3,padding="same"))
model.add(BatchNormalization())
model.add(ReLU())
model.add(MaxPool2D(3,padding = "same",\
  strides = (2,2)))
model.add(Conv2D(24,3,padding="same"))
model.add(BatchNormalization())
model.add(ReLU())
model.add(MaxPool2D(3,padding = "same",\
  strides = (2,2)))
model.add(Conv2D(48,3,padding="same"))
model.add(BatchNormalization())
model.add(ReLU())
model.add(MaxPool2D(3,padding = "same",\
  strides = (2,2)))
model.add(Conv2D(48,3,padding="same"))
model.add(BatchNormalization())
model.add(ReLU())
model.add(Conv2D(48,3,padding="same"))
model.add(BatchNormalization())
model.add(ReLU())
model.add(MaxPool2D((1,13),\
  strides = (1,1)))
model.add(Flatten())
model.add(Dropout(.2))
model.add(Dense(12))
model.add(Softmax())
```

Fig. 9. Implementación de red en Python

- MaxPool2D toma el valor máximo bajo su núcleo. Con un núcleo de 3x3 y un stride de 2x2 esto da como resultado una reducción del tamaño de la imagen.

Después de la extracción de características, la capa Flatten reforma las imágenes bidimensionales en imágenes lineales unidimensionales. La siguiente capa, Dropout, inserta valores cero aleatoriamente mientras escala los valores restantes para mantener el mismo valor promedio, pero solo durante el entrenamiento. Su propósito es evitar el overfitting añadiendo ruido durante el entrenamiento.

La capa Dense es la capa profundamente conectada donde cada neurona recibe entradas de todas las neuronas de las capas anteriores. Esta capa realiza una multiplicación de matrices donde los valores de la matriz son los parámetros que varían a medida que el modelo ”aprende”. En efecto, estos parámetros son la ”memoria aprendida” del modelo.

Finalmente, la capa Softmax implementa la función softmax, una función de normalización de distribución de probabilidad, que es la matriz de probabilidad de que la entrada corresponda a cada clase. La clase correspondiente al lugar de mayor valor en la matriz de probabilidad es efectivamente la predicción de la red.

III. Resultados

La red se entrenó en una maquina virtual que se ejecuta en un cuatro núcleos de un Intel i9-12900KF a 5.2 GHz. A la maquina virtual se le asignaron 16 Gb de memoria para permitir suficiente memoria para que todos los datos permanezcan en la memoria física en todo momento. Las bibliotecas de Keras no usaban aceleración de GPU y la aplicación no se escribió teniendo en cuenta la operación multiprocesador, pero se esperaba que las bibliotecas subyacentes se ejecutaran en tareas múltiples siempre que fuera posible. Durante el entrenamiento, se observo que la aplicación consumía aproximadamente el 31.7% del 400% del ancho de banda disponible de la CPU.

El entrenamiento procedió a través de una secuencia de 25 épocas de 3032 batches cada una. El tiempo de ejecución total fue de 1230 segundos, para un tiempo de época promedio de 49.2 segundos y un tiempo de batch promedio de 16.2 ms. Esto es equivalente a 617.3 espectrogramas que se propagan a traves del modelo por segundo.

Despues de los 25 epocas, el modelo logro una exactitud de entrenamiento del 98.50% y una exactitud de validación de 98.93%. La evolución de la perdida y exactitud a lo largo de todos los batches se representa gráficamente en la Fig 10.

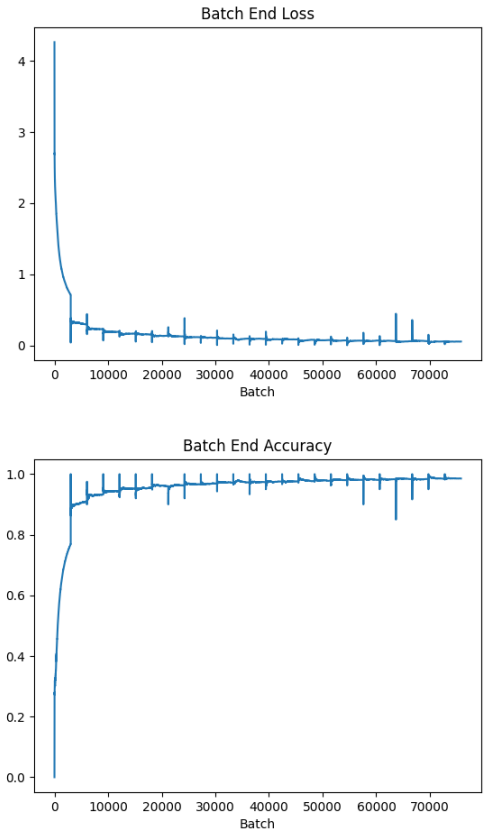


Fig. 10. Evolución de perdida y exactitud de entrenamiento al final de cada batch

La evolución de perdida y exactitud al final de cada época se representa gráficamente en la Fig 11, que muestra la evolución mas suavemente con el tiempo.

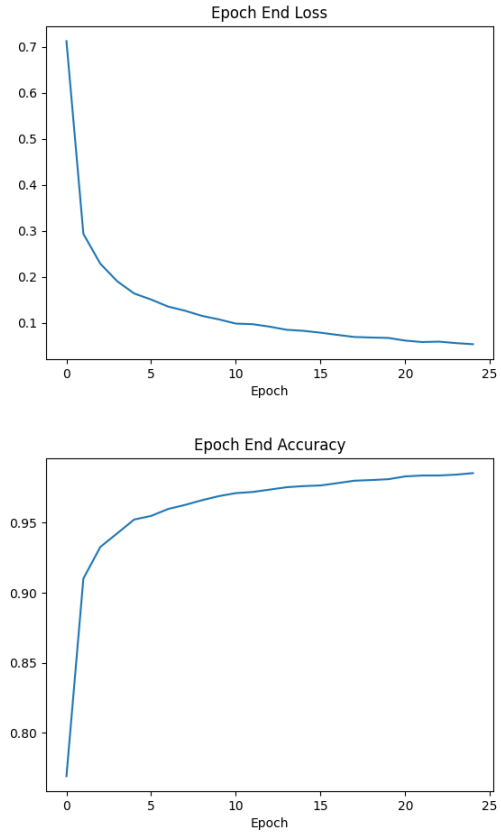


Fig. 11. Evolución de perdida y exactitud de entrenamiento al final de cada época

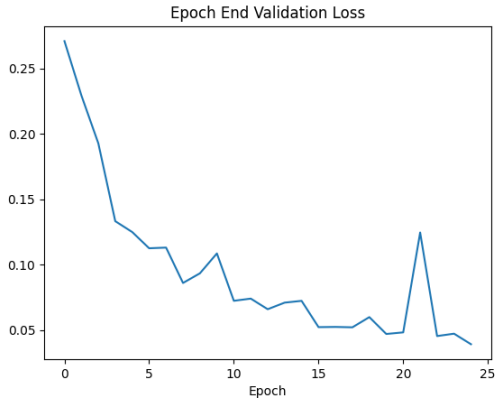
La evolución de perdida y exactitud del conjunto de datos de validación al final de cada época se representa gráficamente en la la Fig 12

IV. Discusión

En los gráficos, parece que después de 25 épocas, la perdida y exactitud de entrenamiento todavía experimentan una tasa de mejor significativa. Sin embargo, los gráficos de perdida de validación y exactitud pueden tener mesetas, habiendo llegado a sus limites. mas ciclos de entrenamiento pueden dar como resultado un modelo que esta ”sobreentrenado” y que se ha ajustado demasiado a las características especificas del conjunto de entrenamiento, perdiendo la capacidad de generalizar. Esta condición estaría indicada por una inflexión en la exactitud de la validación al mismo tiempo que la exactitud de entrenamiento sigue mejorando.

Las claras diferencias en el gráfico de exactitud de entrenamiento en comparación con el gráfico de la de validación muestra aun mas la importancia de la validación utilizando datos fuera del conjunto de entrenamiento. Si bien la exactictud de entrenamiento muestra una mejora continua, la de validación muestra una evolución mucho mas ruidosa, lo que sugiere como es posible el sobreentrenamiento.

También es interesante comparar los gráficos de error cuadrático medio de la Fig 13 con sus correspondientes gráficos de perdida. El valor del error es una cantidad pura-



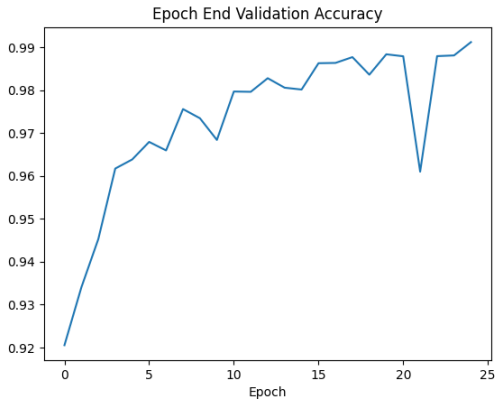
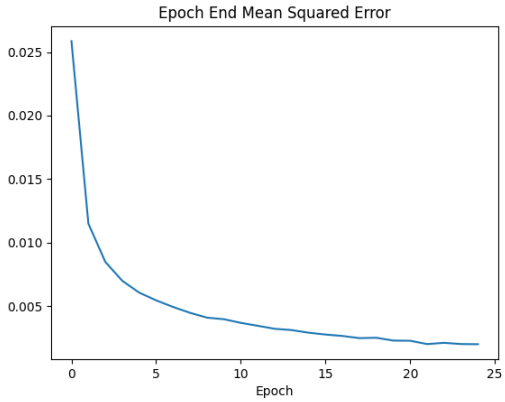


Fig. 12. Evolución de perdida y exactitud de validación al final de cada época



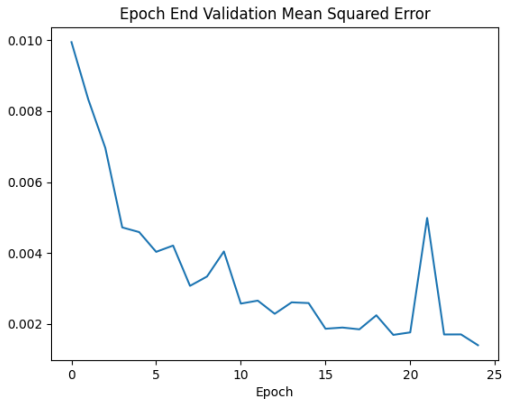


Fig. 13. Error cuadrático medio de entrenamiento y validación al final de cada época

mente estadística mientras que la perdida aplica una función que intenta incorporar una estimación de cuan indeseable es el error. Dado que el error cuadrático medio y las curvas de perdida tienen una forma casi idéntica, parece que la función de perdida de entropía cruzada utilizada aqui puede no contribuir significativamente al entrenamiento del modelo en comparación con métodos menos sofisticados.

La matriz de confusión en la Fig 14 muestra el rendimiento del modelo entrenado al hacer predicciones sobre el conjunto de validación. Es de destacar la tasa relativamente alta de confusión de la palabra ”off” por la palabra ”up”. Si bien se puede decir que las palabras suenan de manera similar, el error inverso de confundir ”off” por ”up” no parece ocurrir con la misma frecuencia. Mirando mas de cerca, parece que el modelo tiene una propensión a predecir incorrectamente ”up” para otras clases validas. Esto puede indicar algún defecto en el conjunto de entrenamiento.

La capacidad del modelo para generalizar aun mas se probo usándolo para hacer predicciones para muestras de audio no incluidas en el conjunto de TensorFlow. Dos oradores hablantes nativos de ingles realizaron un pequeño conjunto de veinte grabaciones (diez por cada orador) para cada clase. La matriz de confusión para esta prueba se muestra en la Fig 15. Si bien este conjunto de muestras es demasiado pequeño para que el análisis estadístico sea útil, se hicieron muy pocas predicciones incorrectas, con una sola instancia de la palabra ”right” que se predijo incorrectamente para una muestra desconocida, y 4

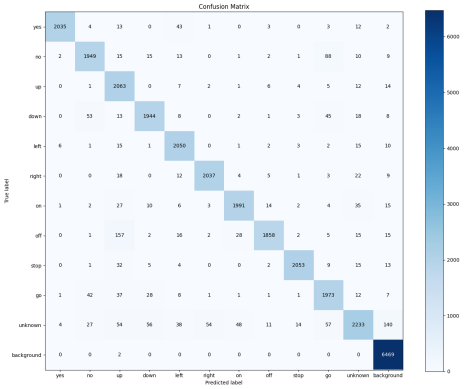


Fig. 14. Matriz de confusión del modelo con el conjunto de validación

instancias donde el ruido de fondo se clasifico incorrectamente como ”unknown”.

V. Conclusiones

El modelo se entreno con éxito utilizando los conjuntos de muestras de entrenamiento y validación. Después del entrenamiento, el modelo logro reconocer comandos pronunciados por nuevos hablantes no incluidos en los conjuntos originales, al menos con un conjunto relativamente pequeño de muestras

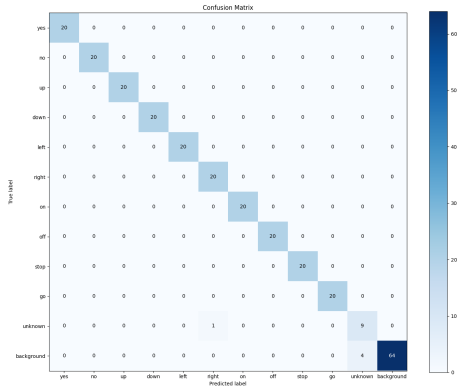


Fig. 15. Matriz de confusión del modelo con las muestras nuevamente grabadas

de alta calidad. Se requieren mas pruebas con un conjunto mas grande de muestras y oradores para caracterizar aun mas el rendimiento con muestras novedosas.

Otras posibles áreas de investigación serian estudiar técnicas de segmentación y análisis para su uso con muestras de menos o mas de un segundo. Además, se deben estudiar los efectos de ciclos de entrenamiento adicionales y diferentes algoritmos de optimización y métricas de perdida en el rendimiento del modelo.

REFERENCIAS

[1] S. Smith, *The Scientist & engineer's Guide to Digital Signal Processing*, 1st Ed. California Technical Pub., 1996.

[2] Becoming Human, Becoming Human: Exploring Artificial Intelligence & What it Means to be Human, https://becominghuman.ai/

[3] S. Saha *"A Comprehensive Guide to Convolutional Neural Networks"*, Towards Data Science, https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[4] B. Burns, K. Brush *"Deep learning"*, TechTarget Enterprise AI, https://www.techtarget.com/searchenterpriseai/definition/deep-learning-deep-neural-network

[5] Keras, *"Keras API reference"*, Keras, https://keras.io/api/

[6] D. Mwiti, *"Keras Metrics: Everything You need to Know*, MLOps Blog, https://neptune.ai/blog/keras-metrics

[7] J. Brownlee, *"How to Use Metrics for Deep Laerning with Keras in Python"*, Machine Learning Mastery, https://machinelearningmastery.com/custom-metrics-deep-learning-keras-python/

[8] L. Roberts, *"Understanding the Mel Spectrogram"*, Analytics Vidhya, https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53

[9] S. Kuo, B. Lee, W. Tian, *Real-Time Digital Signal Processing: Fundamentals, Implementations and Applications*, 3rd Ed. Wiley, 2013.

[10] TensorFlow, *"speech_commands"*, TensorFlow, https://www.tensorflow.org/datasets/catalog/speech_commands

[11] librosa, *"librosa: audio and music processing in Python"*, librosa, https://librosa.org/