Entrenamiento de Modelo de Voz para Hablantes Hispanos Aplicando Redes Neuronales Convolucionales

Voice Model Training for Hispanic Speakers Applying Convolutional Neural Networks

Mariana Areiza [1], Jaime Palacios [2], Anthony Castillo [3], José Mendoza V. [4]*
[1,2,3,4] Laboratorio de Investigación y Desarrollo, Facultad de Ingeniería de Sistemas Computacionales
[1] mariana.areiza@utp.ac.pa, [2] jaime.palacios@utp.ac.pa, [3] anthony.castillo@utp.ac.pa, [4] jose.mendoza11@utp.ac.pa

Resumen– *El presente artículo documenta el diseño y desarrollo de un modelo de voz, entrenado mediante redes neuronales convolucionales utilizando TensorFlow, con diez comandos grabados en español hispanohablantes.*

Palabras claves– *Aprendizaje profundo, Red neuronal convolucional, TensorFlow.*

Abstract– *This article documents the design and development of a voice model, trained through convolutional neural networks using TensorFlow, with ten commands recorded in Spanish-speaking Spanish.*

Keywords– *Convolutional neuronal network, Deep learning, TensorFlow.*

1. Introducción

El reconocimiento del habla es la tecnología que ha permitido a las computadoras identificar comandos de voz y en los casos más complejos, como los asistentes virtuales, entender el contexto de las palabras.

Existen proyectos que buscan crear modelos Open Source como VoxForce que incluye 15 modelos acústicos. Entre los que están el español, inglés, ruso y chino. Sin embargo, los modelos ocultos de Márkov son sensibles a la velocidad del habla y acento.

La tecnología más reciente que se utiliza en esta área es el Deep Learning especialmente las redes neuronales recurrentes.

En este proyecto se explora el uso de redes neuronales convoluciones, usadas habitualmente en reconocimiento de imágenes, a través del proyecto Simple Speech Recognition [1] de Google para desarrollar un modelo que permita identificar diez comandos en español.

Debido a la falta de soporte para los sistemas de reconocimiento del habla, su escasez para la lengua española y la complejidad de obtener un algoritmo funcional, es necesario el desarrollo e investigación de algoritmos y modelos que permitan poner a disposición esta tecnología. Es por lo que se explora una de las posibilidades, más específicamente, el uso de Deep Learning para el reconocimiento del habla.

2. Antecedentes

A. CMU Sphinx

CMU Sphinx es un conjunto de bibliotecas y herramientas de uso libre para el desarrollo de reconocimiento de voz, desarrollado por la Universidad de Carnegie Mellon. Su principal función es la decodificación de la voz con la ayuda del modelo oculto de Márkov; actualmente se encuentra en la cuarta versión el cual tiene nuevas características tales como: la optimización del entrenador que además de agilizar la obtención de datos, mejora la gestión de configuración de sistema creando una interfaz gráfica que permite la visualización de los datos, y una mejor interacción humano computador [2].

Julius es un software que permite la decodificación de voz casi en tiempo real de hasta sesenta mil palabras sin necesidad del uso de internet, usando aplicaciones como Word 3-gram y el modelo oculto de Márkov, desarrollado por Lee Akinobu del Instituto Tecnológico de Nagoya. Tiene como función la decodificación de voz por medio de modelo de mezclas vinculadas con el

HMM; se utiliza principalmente en el sistema operativo Linux donde se descarga los repositorios y complementos de esta. Cuenta con una licencia de código abierto y se distribuye por el estilo BSD (Berkeley Software Distribution) revisada [3].

Google Speech API es una aplicación desarrollada por Google que permite el reconocimiento de voz basado en redes neuronales, esto quiere decir que necesita siempre del internet para su funcionamiento; acepta diversos formatos de audio como FLAC, PCMU, AMR y Linear-16. Este programa utiliza como base el Google Cloud Storage para ejecutar todas sus funciones, consta con un amplio vocabulario de más de 80 idiomas, una de las desventajas es que a la hora de que Google Speech responde una pregunta sólo responderá en Ingles, porque es el único idioma habilitado por el momento y se piensa habilitar el habla en otros idiomas.

Tiene un uso gratuito de hasta 60 minutos, después de los 60 minutos tiene una Tarifa de por cada 15 segundos seria 0,006 $, el uso mensual está limitado a un millón de minutos [4].

3. Metodología

Para llevar a cabo esta investigación, se utiliza las siguientes herramientas:

TensorFlow es una biblioteca de código libre desarrollado y liberado por Google. Su función es el aprendizaje automático y continuo de sistemas a través de tareas asignadas capaz de construir una red neuronal [5].

Python Dev como su nombre lo indica es el Python para desarrolladores que contiene una serie de paquetes para construir extensiones en Python que permiten compilar [6].

Para el proyecto se utilizó una Laptop ASUS GL-503VD el cual cuenta con las siguientes especificaciones:

- Procesador: Intel® Core™ i7 7700HQ.
- Sistema Operativo: Windows 10 Home o Versión: 1709. Compilación del sistema operativo: 16229.214.
- Placa Madre: Intel® HM175 Express Chipset.
- Memoria RAM: 16 GB DDR4 2400MHz SDRAM.
- Tarjeta Gráfica: NVIDIA GeForce GTX 1050 con 4GB GDDR5 VRAM.

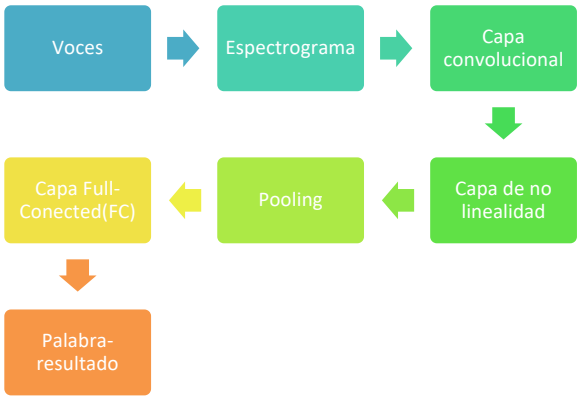

Figura 1. Proceso del ordenador al analizar los audios.

Redes Neuronales Convolucionales.

Las redes neuronales reciben como entrada un vector y a través de diversas capas ocultas obtiene aproximaciones por clase.

Como cualquier otra red neuronal la Convulutional Neural Network (CNN) es representada por grafos.

Esta configuración es idónea para estructuras 2D como imágenes. Lo cual se logra con conexiones locales y pesos vinculados [7].

Las CNN se dividen en cuatro capas las cuales son:

La capa convolucional consiste en un grupo de núcleos o filtros que se pueden aprender [8]. Estos filtros escanean conjuntos de pixeles para determinar la composición RGB de cada uno y calculan los productos de punto entre la entrada del filtro y la data de entrada en cualquier otra posición. A medida que se analiza el volumen de la data de entrada se obtiene un mapa de activación bidimensional, son la respuesta del filtro a cada posición espacial [9]. Después del entrenamiento la red aprenderá los filtros al reconocer ciertos patrones.

La capa de no linealidad en esta capa se aplica la siguiente formula a cada una de las neuronas.

$$RELU(x) = \begin{Bmatrix} x \leq 0 \rightarrow x = 0 \\ x > 0 \rightarrow x = 0 \end{Bmatrix} \qquad (1)$$

La fórmula de activación tiene el nombre RELU por sus siglas en ingles Rectified Linear Units.

Pooling es un algoritmo reduce progresivamente el tamaño espacial de la representación lo que reduce los datos a computar, lo que a su vez controla el sobre ajuste.

La Capa Full-Conected(FC) son nodos o neuronas de esta capa que están conectadas por completo a las neuronas de la capa anterior, también analiza el vector extraído y determina a que clase pertenece.

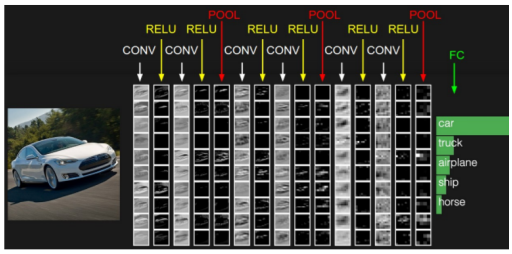

Figura 2. Muestra de funcionamiento en reconocimientos. de imágenes con CNN [9].

Funcionamiento del algoritmo para entrenar el modelo.

Simple Audio Recognition [1] en TensorFlow está basado en Convolutional Neural Networks for Small-footprint Keyword Spotting [13]. Esto se debe a que redes neuronales convolucionales son más fáciles de entender y entrenar que las redes neuronales recurrentes.

Debido a que el audio es una señal unidireccional, continua a través del tiempo genera un problema ya que las redes convolucionales se utilizan para en imágenes. Este problema se resuelve al transformar la onda de audio en una imagen. Primero se segmenta la muestra de audio, de unos pocos milisegundos de largo, y se calcula intensidad de la frecuencia en un conjunto de bandas. Cada conjunto de intensidad de frecuencias se utiliza como vector a través del tiempo formando una matriz bidimensional. Esta matriz es la que se transforma en imagen conocida como espectrograma [1].

4. Data

La data es la información traducida a una forma que lo hace eficiente para el procesamiento y el movimiento de este dentro de una computadora.

En este caso la data a utilizar serán varias voces de personas pronunciando palabras escogidas, lo cual se graba repetidamente hasta tener una cantidad considerable, en este caso, se graba diez palabras por persona lo cual cada palabra lo repiten 10 veces, y al final habrá 100 grabaciones por persona, pero con formato .WAV, ya que TensorFlow y Python necesitan un formato de audio compatible con sus sistemas y que tenga un peso liviano para evitar la pérdida de datos y aumentar la precisión.

A. Característica de la data

Como se menciona anteriormente, la data recopilada debe tener una cierta estructura para que el sistema que lo va a compilar lo valide, lo cuales serial el tipo de extensión, que sería .wav.

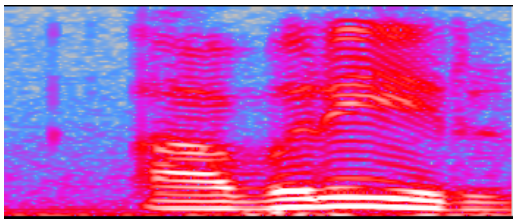

Figura 3. Espectrograma de frecuencias de la palabra abrir.

Se utiliza el .WAV debido a que es un formato puro, no agrega data ni reguladores dentro del archivo, lo cual, aumenta la fidelidad y es usado mundialmente para la edición de audio a diferencia del MP3 que es el más conocido por su abreviación MPEG Layer 3, el cual comprime el archivo y regula el volumen del audio perdiendo características del audio.

Frecuencia de muestreo

De forma teórica la frecuencia es la cantidad de ondas generadas en un segundo.

El teorema de NyQuist-Shannon dice que para poder digitalizar una señal análoga y transmitirla por un medio eléctrico a grandes distancias y poder recuperarla en el extremo distante con la máxima fidelidad posible, se requiere que la señal análoga sea muestreada al menos dos veces su frecuencia máxima [10].

En el caso de la voz, el ancho de banda es de 4000 Hz, entonces, su razón de muestreo es 2B=2x que sería equivalente a 8000 Hz muestras por segundo. Entonces la razón del muestreo de la voz debe ser de al menos 8000 Hz muestras para que pueda regenerarse sin error.

Profundidad de bit

Hace referencia a la resolución de captura de una señal de audio en relación con la amplitud(volumen) [11].

Para este proyecto se emplea 16 bits debido a que permite una calidad de audio nítida.

B. Características de las muestras

Estas muestras se obtienen de forma voluntaria de trabajadores y estudiantes de la UTP (Universidad Tecnológica de Panamá) Centro Regional de Chiriquí.

5. Entrenar el modelo

Para entrenar el modelo se utilizó el algoritmo Simple Audio Recognition de TensorFlow [1].

Originalmente el algoritmo utiliza el DataSet [12] colectado por Google en Ingles para crear una red neuronal y se compila a través de Bazel.

No obstante, en búsqueda de simplificar el proceso, el compilador utilizado fue Python 3.5 debido a que Bazel causo ciertas incompatibilidades con el lenguaje de bajo nivel del procesador, además que su uso requiere diferentes librerías.

Como algoritmo Simple Audio Recognition no se encuentra en la librería de TensorFlow, es necesario descargarlo del repositorio de GitHub.

Para el programa funcione con la data de este proyecto es necesario alterar las líneas, en train.py, 291 y eliminar el URL, 297 agregar la ruta de la carpeta donde esta almacena la data y en 399 remplazar por la lista de palabras usadas en el proyecto en este caso (abrir, atrás, avanzar, cerrar, derecha, detener, hola, izquierda, maximizar, minimizar).

6. Resultados

Una vez colectada la data, se separa en carpetas para cada palabra.

Para evitar que la red memorice sus entradas durante el entrenamiento es recomendable dividir el banco de datos en tres categorías. El 80% se utiliza para entrenar la red el 10% llamado se emplea para validar la red y el ultimo 10% para probar la red [1].

En la figura 4, se muestra la gráfica de precisión vs pasos. En naranja se muestra los valores para la data de entrenamiento y en azul los resultados para la data de validación. Cabe destacar que al final del entrenamiento, a los 5200 pasos, existe una precisión de 90 %.

En la figura 5, se muestra la gráfica de error relativo vs pasos. Al igual que la figura 3 en naranja se muestra los valores para la data de entrenamiento y en azul los resultados para la data de validación. Al final del entrenamiento existe un error relativo de 0.5%. Para reducir el error es necesario aumentar la cantidad de data de entrenamiento y el número de pasos.

El modelo generado se prueba con 3 audios de la carpeta de prueba, que no fue utilizada para entrenar o

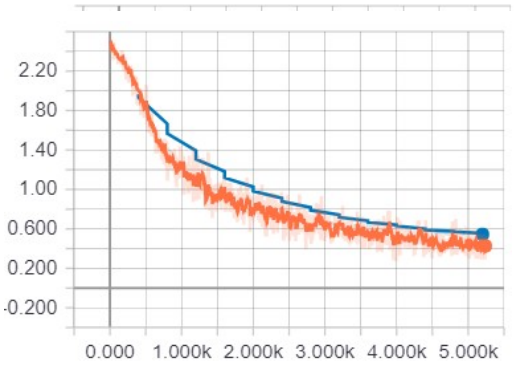

Figura 5. Error relativo vs la cantidad de pasos.

validar el modelo.

En la figura 6 se muestra la taza de acierto del modelo. Aparece en rojo cuando la palabra que se reconoce es errónea y en verde cuando es acertada.

Los valores se obtienen a través de un programa de la librería descargada de GitHub, label_wav.py. Este programa ejecuta el modelo entrenado previamente y da una salida, en orden descendente respecto al porcentaje, de tres posibles palabras a las que puede pertenecer el audio. También contempla si el audio es desconocido o es silencio.

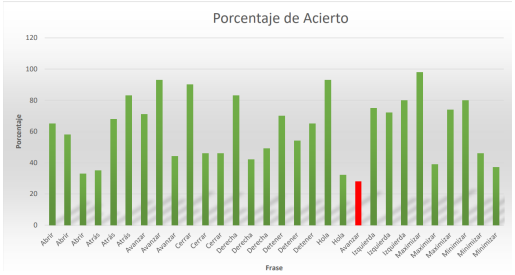

Figura 6. tasa de acierto.

7. Conclusiones

Se comprobó que, el algoritmo desarrollado por Google en Simple Audio Recognition puede ser aplicado para el desarrollo de modelos con data en español.

El número de muestras directamente en calidad del modelo, ya que, a mayor cantidad de participantes mejor será la respuesta del sistema.

El modelo obtenido posee una precisión del 90% para los hablantes cuyas voces fueron parte del entrenamiento del modelo.

8. Recomendaciones

La data de entrenamiento debe ser grabada con un micrófono unidireccional y en ambientes libres de ruido para evitar la contaminación de la muestra. También debe contener gran variedad de formas de pronunciar las palabras, variando la entonación, por lo que es conveniente obtener muestras de voz procedentes de personas que el español no sea su idioma nativo.

9. Trabajo futuro

Los planes a futuro son aplicar una configuración distinta de red neuronal convolucional y utilizar la data de este proyecto, para disminuir el error relativo y aumentar la precisión de respuesta del modelo.

10. Reconocimiento

Un agradecimiento a las personas que prestaron su voz para el desarrollo de esta investigación.

11. Referencias

[1] Google, "Simple Audio Recognition | TensorFlow," january 13, 2018. [Online]. Available: https://www.tensorflow.org/versions/master/tutorials/audio_recognition. [Accessed: 31-Jan-2018].

[2] "CMUSphinx Documentation – CMUSphinx Open Source Speech Recognition." [Online]. Available: https://cmusphinx.github.io/wiki/. [Accessed: 10-Feb-2018].

[3] "Open-Source Large Vocabulary CSR Engine Julius." [Online]. Available: http://julius.osdn.jp/en_index.php. [Accessed: 10-Feb-2018].

[4] Google, "API Speech: reconocimiento de voz | Google Cloud Platform." [Online]. Available: https://cloud.google.com/speech/?hl=es. [Accessed: 10-Feb-2018].

[5] Google, "Getting Started | TensorFlow," Enero 13, 2018. [Online]. Available: https://www.tensorflow.org/versions/master/get_started/. [Accessed: 15-Jan-2018].

[6] python, "dev 0.4.0 : Python Package Index." [Online]. Available: https://pypi.python.org/pypi/dev/0.4.0. [Accessed: 30-Jan-2018].

[7] UFLDL, "Tutorial de aprendizaje de funciones y aprendizaje profundo sin supervisión." .

[8] A. Bhandare, M. Bhide, P. Gokhale, and R. Chandavarkar, "Applications of Convolutional Neural Networks."

[9] "CS231n Convolutional Neural Networks for Visual Recognition." [Online]. Available: http://cs231n.github.io/convolutional-networks/. [Accessed: 10-Jan-2018].

[10] J. M. Sociedad Mexicana de Física and C. . Stern Forgach, Revista mexicana de física. E, Publicación de enseñanza, historia y filosofía de la Sociedad Mexicana de Física., vol. 56, no. 2. Sociedad Mexicana de Física, 2010.

[11] Academia UNIMUSICA, "La Profundidad de Bits - Clases de Produccion Musical en Lima Metropolitana." [Online]. Available: https://www.unimusica-peru.com/produccion_musical_profundidad_bits.htm. [Accessed: 15-Feb-2018].

[12] Google AIY, "Google launches Speech Commands Dataset - voxforge.org," agosto 25, 2017. [Online]. Available: http://www.voxforge.org/home/forums/message-boards/speech-recognition-in-the-news/google-launches-speech-commands-dataset. [Accessed: 16-Jan-2018].

[13] S. O. Arik et al., "Convolutional Recurrent Neural Networks for Small-Footprint Keyword Spotting," Mar. 2017.

[14] S. B. Hannes Schaulz, «Springer,» 17 Mayo 2012. [En línea]. Available: https://link.springer.com/article/10.1007/s13218-012-0198-z. [Último acceso: 10 Mayo 2018].

[15] Y. T. M.-J. H. J.-Y. C. Y.-H. L. F.-C. L. C.-T. W. Shih-Hau Fang, «ScienceDirect,» 19 Marzzo 2018. [En línea]. Available: https://www.sciencedirect.com/science/article/pii/S089219971730509X. [Último acceso: 15 Mayo 2018].

[16] L. L. W. H. L. Z. N. Z. D. B. Ji Yan, «ScienceDirect,» Abril 2018. [En línea]. Available: https://www.sciencedirect.com/science/article/pii/S0167639317300961. [Último acceso: 15 Mayo 2018].

[17] N. L. S. Scotty D. Craig, «ScienceDirect,» Noviembre 2017. [En línea]. Available: https://www.sciencedirect.com/science/article/pii/S0360131517301653. [Último acceso: 13 Mayo 2018].

[18] D. D. I. C. Ido Ariav, «ScienceDirect,» Enero 2018. [En línea]. Available: https://www.sciencedirect.com/science/article/pii/S0165168417302529. [Último acceso: 13 Mayo 2018].

[19] D. Q. W. Z. W. Z. Xu'Kui Yang, «ScienceDirect,» 9 Marzo 2018. [En línea]. Available: https://www.sciencedirect.com/science/article/pii/S1051200418300745. [Último acceso: 14 Mayo 2018].

[20] H. P. J. C. Inyoung Hwang, «ScienceDirect,» Julio 2016. [En línea]. Available: https://www.sciencedirect.com/science/article/pii/S0885230815001072. [Último acceso: 14 Mayo 2018].