Práctica de Aprendizaje Profundo en la Asignatura de Tecnología para Sistemas Inteligentes

A. Molina-Picó[(1)], J. Jordán-Nuñez[(2)], B. Micó-Vicent[(2)]
(1) Departamento de Informàtica de Sistemas y Computadores,
Universitat Politècnica de València, Plaza Ferrándiz y Carbonell s/n, Alcoi, (Alicante)
e-mail: antoniomolina@disca.upv.es
(2) Departamento de Ingeniería Gráfica, Universitat Politècnica de València
Plaza Ferrándiz y Carbonell s/n, Alcoi, (Alicante)
e-mail: jorjornu@upv.es

RESUMEN

En este trabajo se presentan los resultados de una práctica llevada a cabo por los alumnos de la asignatura Tecnología para Sistemas Inteligentes, impartida en el Grado de Ingeniería Informática (Universitat Politècnica de València, Campus d'Alcoi). La práctica desarrolla una aplicación de reconocimiento de voz en el que un sistema Arduino se entrena para que pueda discriminar entre unos pocos comandos y actuar en consecuencia. Los alumnos tienen que grabar los ficheros con su propia voz para entrenar el modelo y compartir entre ellos las muestras con la intención de mantener una base de datos lo suficientemente representativa.

Palabras clave: Aprendizaje Profundo, TinyML, Arduino

INTRODUCCIÓN

Cada vez existen más dispositivos en el mercado que aplican algoritmos de Inteligencia Artificial sin que realmente seamos conscientes de las virtudes y riesgos que suponen. Los asistentes de voz son un claro ejemplo de este tipo de aplicaciones [1]. Todas las grandes tecnológicas han apostado por ofrecer su versión: Amazon ha desarrollado a Alexa, Microsoft a Cortana, Apple a Siri y Google a Google Assistant. Todos estos dispositivos que aplican reconocimiento de voz, ejecutan los algoritmos en la nube: envían el streaming de audio con la petición y reproducen la respuesta recibida por los servidores. Esta forma de operar hace que tengan que permanecer continuamente en escucha, pudiendo generar problemas serios de privacidad en caso de que sean hackeados. Para solucionar esta limitación, TinyML [2] permite ejecutar algoritmos de reconocimiento de voz en dispositivos que no consumen prácticamente nada y que sean estos los que tras reconocer algún comando activen a otro equipo de mayor consumo que ya sea el encargado de enviar la petición a la nube. El dispositivo que se utiliza para llevar a cabo esta práctica se llama Arduino Nano BLE 33 [3] y se trata de una placa de circuito impreso de bajo coste, bajo consumo y con una gran capacidad sensorial. En esta práctica se pretende que el alumno programe el equipo Arduino de tal manera que sea capaz de reconocer varios comandos y actuar en consecuencia: si la palabra sintetizada y reconocida es "verde", el equipo debe proceder a encender un led de color verde. Si la palabra es "rojo", se debe activar el led de ese color y si es "apágate", debe desactivarlos todos. La actuación sobre los leds no conlleva ninguna complejidad, pero abre un abanico de posibilidades en cuanto a posibles integraciones con motores u otro tipo de actuadores. Para conseguir este objetivo, es necesario recorrer todas las fases del proceso de Aprendizaje Profundo, desde la captación de muestras de audio, pasando por el entrenamiento y generación del modelo, hasta llegar a la ejecución final del modelo en el dispositivo [4]. La fase de captación de muestras de voz se plantea

como un trabajo colaborativo con todo el grupo de la clase para partir de una cantidad de muestras suficiente y variada para entrenar el modelo.

OBJETIVOS

Esta práctica tiene dos objetivos conceptuales en los que el alumno debe afianzar los conocimientos teóricos adquiridos (1 y 2) y dos objetivos procedimentales que pretenden que el alumno sepa aplicar la metodología correcta ante una aplicación de inteligencia artificial de este estilo (3 y 4). Integrar todas las etapas necesarias en el proceso de ML en una aplicación real de reconocimiento de voz. Evaluar el modelo de predicción generado citando términos estadísticos como la matriz de confusión, el parámetro de pérdidas o la probabilidad de aciertos. Combinar el uso en cascada de varias herramientas software en el que los resultados generados por una de ellas se utilizan como entrada de otra. Conocer herramientas colaborativas existentes como Google Collab o Edge Impulse que permiten ejecutar código en servidores remotos mucho más potentes que los equipos personales.

DESARROLLO DE LA PRÁCTICA

Los alumnos de la asignatura que vayan a realizar la práctica necesitan disponer de un guion en el que se vayan listando todos los pasos que tienen que ir aplicando. Esta memoria se le debe facilitar al alumno con anterioridad a la práctica y recomendar una lectura previa. Se trata sin duda alguna de un ejercicio complejo, en el que principalmente se pide al alumno que vaya realizando todas y cada una de las tareas establecidas de manera ordenada. Se deben ejecutar diferentes fases.

La primera tarea que deben abordar los alumnos es la de recopilar las muestras necesarias para entrenar el modelo de aprendizaje. Estas muestras de audio deben ser grabadas de manera individual, pero compartidas por todo el grupo de clase. Aun así, es conveniente tener presente que la mayoría de alumnos comparten el mismo género y perfil sociodemográfico, y que si se ejecuta el algoritmo con un perfil muy distinto, es probable que el algoritmo no sea capaz de discriminar correctamente. Todos los ficheros de audio deben compartir las mismas características en cuanto a duración (1 segundo) y frecuencia de muestreo (16 kHz). El número de muestras que se aconseja crear a cada alumno es de 20 elementos para cada palabra: *rojo, verde y apágate.*

La fase de preprocesado es necesaria en la mayoría de aplicaciones de aprendizaje automático. Cuando se procesan imágenes es casi inevitable aplicar alguna conversión, modificar la resolución de la imagen o modificar parámetros como el contraste o el brillo. Pues en esta práctica también es necesario realizar un preprocesado de los ficheros de audio capturados que nos permitan preparar las muestras para el modelo de aprendizaje. En una primera fase de preprocesado se van a añadir distintos sonidos de fondo a las muestras para que el algoritmo pueda ser discriminantes en entornos con barullo o interferencias. En una segunda fase se balancea el número de muestras y finalmente se separan en dos conjuntos: entrenamiento y testeo. La separación en estos grupos se realiza con una ratio 80/20, es decir, el 80% de las muestras se van a utilizar para entrenar el modelo y el 20% restante para testearlo. El programa en python que ejecuta esta fase de preprocesado se le proporciona al alumno, de manera que éste solo debe analizar detalladamente cada línea de código y realizar una pequeña adaptación en algunas variables para que la ejecución sea correcta.

Para generar el modelo vamos a utilizar una plataforma online del estilo de Google Colab, llamada Edge Impulse. Este servicio web basado en TinyML permite generar

modelos de aprendizaje automático de una manera muy intuitiva para que sean ejecutados en dispositivos de bajo consumo. El alumno debe importar los datos en la plataforma, y se le plantea al alumno dos formas para hacerlo: una forma manual o ejecutando un pequeño programa que lo hace de manera automática. Seguidamente se debe diseñar el modelo de aprendizaje, en el que el alumno debe elegir entre una serie de opciones los mecanismos tanto para el bloque de procesado (la plataforma recomienda aplicar los Coeficientes Cepstrales en las Frecuencias de Mel [5]), como para el propio bloque de aprendizaje (se recomienda aplicar una Red Neuronal Convolucional).
El siguiente paso consiste en entrenar el modelo que va mostrando los resultados parciales en cada simulación. Finalmente, en la fase de ejecución, el alumno debe seleccionar el dispositivo en el que va a ejecutar el modelo (Arduino Nano BLE 33 en este caso) y automáticamente se genera un fichero comprimido que el alumno debe descargar.
La librería exportada en Edge Impulse debe abrirse con la aplicación Arduino y realizar unos pequeños ajustes respecto a la presentación de los resultados. La aplicación debe obtener los resultados de predicción y actuar consecuentemente sobre el led verde o rojo, según el comando recibido.

RESULTADOS

Los resultados proporcionados al realizar el entrenamiento del modelo son un buen indicador de la bondad del modelo. Estos resultados se representan en algunos parámetros de salida como las pérdidas (*loss*), que indica cómo de alejados están las predicciones de los valores reales, o la precisión (*accuracy*) que indica la probabilidad existente que una nueva muestra se clasifique correctamente. Además de estos parámetros, los resultados del modelo también se representan gráficamente mediante la matriz de confusión (Tabla 1) y la distribución de las muestras clasificadas correcta e incorrectamente (Figura 1).

Tabla 1. Matriz de confusión obtenida al aplicar el conjunto de test sobre el modelo. Las filas corresponden a las etiquetas reales y las columnas representan las predicciones.

	_ambiente	**_desconocido**	**rojo**	**apágate**	**verde**
_ambiente	98.5%	0.5%	0.5%	0.5%	0%
_desconocido	3.6%	74.7%	3.1%	16.5%	2.1%
rojo	0%	1.0%	99.0%	0%	0%
apágate	2.1%	5.2%	0%	92.7%	0%
verde	0%	1.0%	0.5%	0%	98.4%

Es destacable que los valores de clasificación correcta en los comandos “rojo” y “verde” es muy elevado. Es muy probable que la razón sea porque el grupo de palabras que se clasifican como “_desconocido” son palabras de habla inglesa muy diferentes a las palabras en cuestión. Esta diferencia es más notable con la palabra “rojo” y puede que sea por esta misma razón: el fonema *erre* vibrante con el que empieza la palabra es poco común en lengua inglesa. Sin embargo, es importante tener en cuenta que los resultados estadísticos obtenidos son teóricos y que, en la ejecución final, es posible que nuestro modelo no se comporte tan bien como se espera. La razón de este comportamiento podría ser porque las muestras de entrenamiento no son tan representativas como debería, pero también podría deberse a diferencias significativas entre el micrófono con el que se han tomado las muestras de entrenamiento y el

micrófono del Arduino en el que se ejecuta el programa final. Para mejorar estos resultados, el alumno puede ajustar de manera experimental el umbral de sensibilidad y establecer las actuaciones sobre los leds con valores inferiores de predicción, aunque esto pueda ocasionar la aparición de algún falso positivo.

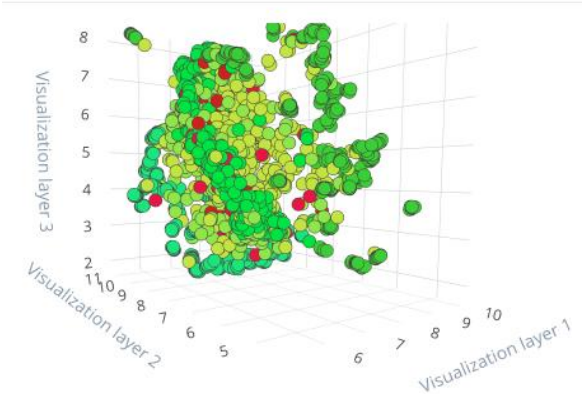

Figura 1. Distribución de los datos de entrenamiento clasificados por la red neuronal. En verde se representan las muestras clasificadas correctamente y en rojo las clasificadas erróneamente

CONCLUSIONES

En este artículo se ha propuesto una práctica de Aprendizaje Profundo a los alumnos en la asignatura de Tecnologías para Sistemas Inteligentes del Grado de Ingeniería Informática. Desde un punto de vista académico, se trata de un ejercicio completo en el que el alumno atraviesa por todo el proceso necesario para llevar a cabo una aplicación de reconocimiento de voz, desde la obtención de muestras de audio hasta la ejecución final en una pequeña placa de circuito impreso. No es objetivo de esta práctica que el alumno recuerde parámetros propios de la red neuronal utilizada o el algoritmo de procesado utilizado, pero sí que tenga una visión global de todos los pasos necesarios para desarrollar una aplicación de aprendizaje profundo y que conozca que es algo que hoy en día puede aplicarse en un dispositivo de muy bajo coste.

Desde un punto de vista docente, el retorno que esta experiencia ha supuesto para la asignatura ha sido enteramente gratificante: el hecho de que se trata de una práctica en la que el alumno obtiene resultados tangibles y que además puede compartir con sus compañeros o su familia, ha sido muy motivante. Además, los alumnos han tenido que compartir su trabajo con el objetivo de mejorar el comportamiento de la aplicación porque sencillamente el timbre vocal de cada persona es diferente.

REFERENCIAS

[1] Rivero Santana, Claudia Raquel (2020). “Plataforma para la integración de asistentes de voz en dispositivos deintercomunicación de terminales de vivienda.” Universitat Politècnica de València.

[2] Warden, Pete (2019). *TINYML : machine learning with tensorflow on arduino, and ultra-low power micro-controllers*, First edition. Beijing: O’Reilly.

[3] Kurniawan, Agus. (2021). *IoT Projects with Arduino Nano 33 BLE Sense Step-By-Step Projects for Beginners*, 1st ed. Berkeley, CA: Apress. doi: 10.1007/978-1-4842-6458-4.

[4] Ketkar, Nikhil (2017). *Deep learning with Python : a hands-on introduction*, 1st ed. 2017. Apress. doi: 10.1007/978-1-4842-2766-4.

[5] Aguirre Martín, Fabián (2017). “Desarrollo y análisis de clasificadores de señales de audio.” Universitat Politècnica de València.