UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

SpotMyFM

Presentado por Jorge Ruiz Gómez
en Universidad de Burgos — 30 de mayo de 2022
Tutor: Bruno Baruque Zanón

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. nombre tutor, profesor del departamento de nombre departamento, área de nombre área.

Expone:

Que el alumno D. Jorge Ruiz Gómez, con DNI dni, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado título de TFG.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 30 de mayo de 2022

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. nombre tutor	D. nombre co-tutor

Resumen

En este primer apartado se hace una **breve** presentación del tema que se aborda en el proyecto.

Descriptores

Palabras separadas por comas que identifiquen el contenido del proyecto Ej: servidor web, buscador de vuelos, android ...

Abstract

A **brief** presentation of the topic addressed in the project.

Keywords

keywords separated by commas.

Índice general

Índice de figuras

Índice de tablas

Introducción

Descripción del contenido del trabajo y del estrucutra de la memoria y del resto de materiales entregados.

Objetivos del proyecto

Este apartado explica de forma precisa y concisa cuales son los objetivos que se persiguen con la realización del proyecto. Se puede distinguir entre los objetivos marcados por los requisitos del software a construir y los objetivos de carácter técnico que plantea a la hora de llevar a la práctica el proyecto.

Conceptos teóricos

3.1. Clasificación mediante Redes Neuronales Convolucionales

Las redes neuronales convolucionales son una de las arquitecturas neuronales más utilizadas debido a su poder a la hora de trabajar con imágenes. Estas redes suelen estar formadas por una serie de bloques convolucionales, seguidos de una o varias capas densas en la salida de la red.

Bloque Convolucional

Estos bloques, formados por capas, buscan resaltar los elementos más relevantes en los datos (normalmente imágenes) a partir de las operaciones de convolución y pooling.

La operación de convolución está formada por un Kernel o filtro que se aplica a través de los datos. En el caso de la capa de convolución, los pesos entrenables de dicha capa son el propio kernel.

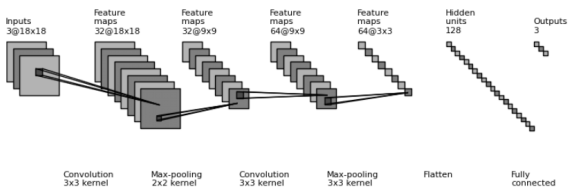


Figura 3.1: Arquitectura de una red convolucional

Figura 3.2: Kernel 3x3

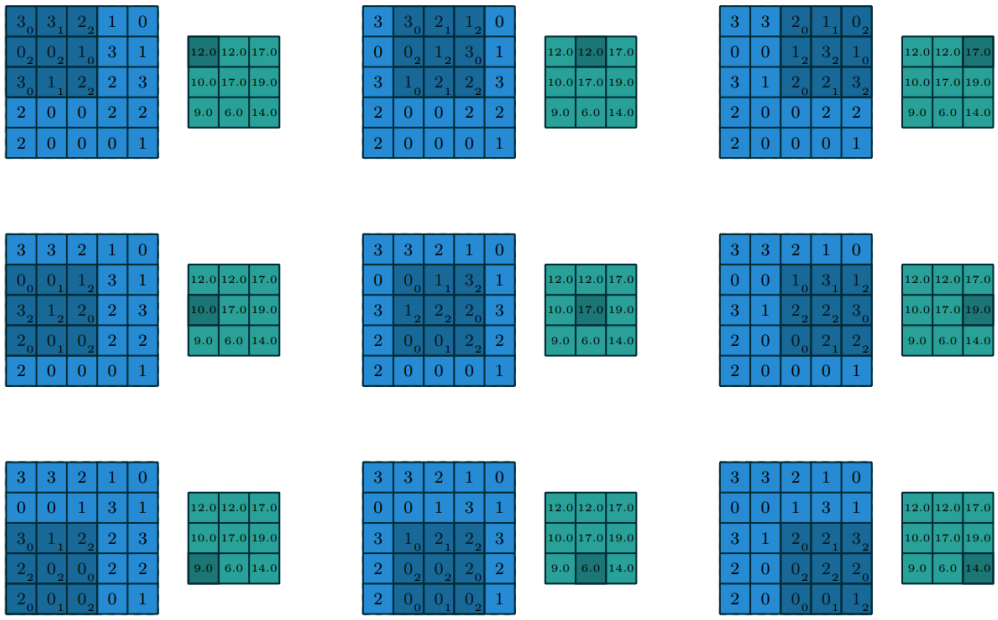


Figura 3.3: Aplicación del kernel 3x3 sobre un área 5x5

Esta operación puede estar acompañada por una función de activación para cada una de las neuronas, como por el ejemplo la función de activación RELU

$$Relu(x) = max(0, x)$$

Por otro lado, la operación de pooling intenta reducir el tamaño de la entrada, dejando solo los datos relevantes de un área. Se aplica de la misma manera que una convolución, pero en vez de aplicar un kernel, realizamos

una operación aritmética sobre el área, como obtener el valor más grande del área (Max Pooling) o la media de todos los valores (Average Pooling)

Un bloque convolucional puede contener otras capas, como capas de normalización, que normalizan de nuevo los datos para trabajar siempre en el intervalo [0,1].

Densa

La salida de una red neuronal está formada por capas densas, estas capas tienen una entrada unidimensional, por lo que si la salida del último bloque convolucional no es unidimensional, es necesario utilizar una operación de Flatten que transforma la matriz de entrada en una única fila concatenando cada una de las filas de la matriz.

En este tipo de arquitectura, cada una de las neuronas de una capa se interconectan con todas las neuronas de la capa siguiente.

Cada neurona está formada por:

- Entradas ($[x_0 \dots x_n]$)
- Salida (y)
- Vector de pesos entrenables ($[w_0 \dots w_n]$)
- Función de activación ($fn(x)$)
- Bias de activación entrenable (b)

Y la salida (o salidas) de una neurona se calculan de la siguiente manera

$$y = fn(x \cdot w + b)$$

La salida de cada una de estas neuronas se propaga a la siguiente capa como la entrada de cada neurona.

Capa de Salida

En tareas de clasificación, la capa de salida está formada por una o varias neuronas (dependiendo del número de clases), y una función de activación.

En tareas de clasificación con **varias clases**, utilizamos la función de activación Softmax, que aplica una normalización en la salida para resaltar la salida con la clase resultante.

Conociendo la salida de cada una de las neuronas, obtenemos la neurona con la salida más cercana a 1 (la salida más grande)

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ i = 1, 2, \ldots, n$$

Para obtener la salida más grande:

$$clase = argmax([y_0 \ldots y_n])$$

En tareas de clasificación con **varias etiquetas**, utilizamos la función de activación Sigmoide, que permite a cada neurona tener una salida entre 0 y 1, siendo este la confianza de que la clase contenga dicha etiqueta.

$$sigmoide(x) = \frac{1}{1 + e^{-}x}$$

Consideramos que una entrada contiene una etiqueta si el resultado de la sigmoide es mayor o igual que 0.5

3.2. Cuantización de Redes Neuronales

La cuantización de redes neuronales es una de las técnicas de optimización de redes neuronales entrenadas más utilizada.

El proceso de cuantización consiste en convertir los distintos pesos (y por tanto operaciones) de la red neuronal a un tipo de datos con menos bits, lo que permite ahorrar memoria y acelerar las operaciones.

Normalmente, la cuantización convierte los tensores de FLOAT32 (4 bytes por peso) a INT8 (1 byte por peso)

En el proceso de cuantización tenemos presente dos parámetros para cada uno de los tensores:

Escala (S): Es un número decimal que nos permite escalar los valores entre un número real y uno cuantizado.

Zero Point (Z): Es un número dentro del rango de cuantización. Nos indica el valor del 0 real en valores cuantizados.

Estos valores son calculados a partir de la distribución de entrada y la de salida.

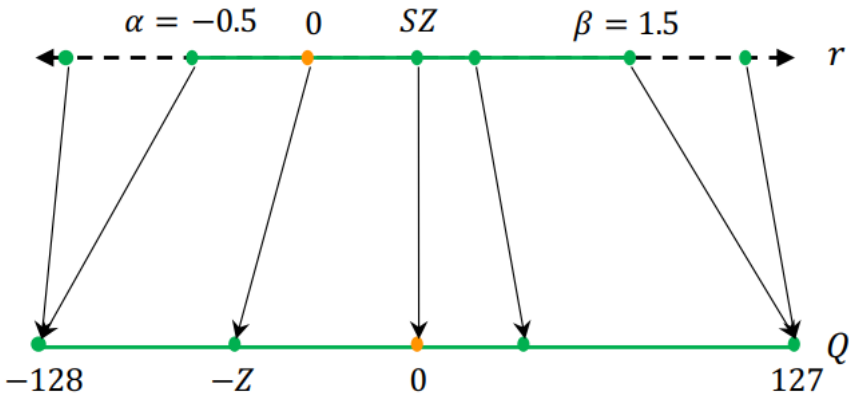


Figura 3.4: Cuantización INT8 en el intervalo $[-0{,}5, 1{,}5]$

$$S = \frac{\beta - \alpha}{\beta_q - \alpha_q}$$

$$Z = round(\frac{\beta\alpha_q - \alpha\beta_q}{\beta - \alpha})$$

En el caso de la distribución de valores reales, debemos conocer el valor mínimo (α) y el máximo (β) del tensor.

Para los tensores cuantizados, α_q y β_q se corresponden con el mínimo y el máximo de los tipos de datos, para INT8, $\alpha_q = -128$ y $\beta_q = 127$.

Una vez obtenidos S, Z, α_q y β_q, ya podemos cuantizar o decuantizar los distintos tensores con la siguiente ecuación.

$$x_q = clip(round(\frac{1}{s}x + z), \alpha_q, \beta_q)$$

Siendo clip una operación de saturación que impide que resultado de la cuantización sea un valor sea menor que α_q o mayor que β_q. En la figura 3.4 esta operación se ve reflejada en los valores que están fuera del intervalo $[\alpha, \beta]$

Técnicas de Cuantización

Existen dos técnicas de cuantización post entrenamiento:

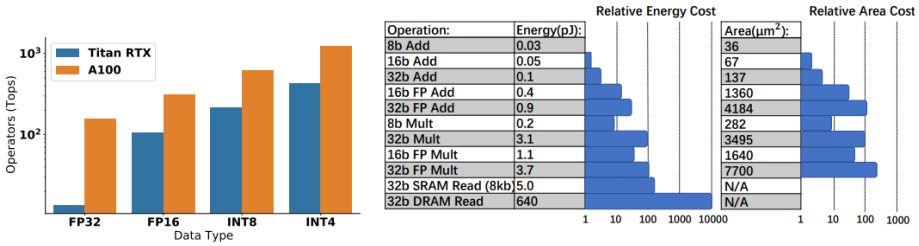


Figura 3.5: Resultados del coste de la operaciones dependiendo del tipo de dato de una red neuronal

Fake Quantization o Cuantización Dinámica:

Los pesos se almacenan en un tipo de datos inferior, pero los parámetros de cuantización se calculan en tiempo de inferencia.

Fixed Quantization:

En este caso los parámetros de la cuantización se calculan durante el proceso de cuantización. Este tipo de cuantización requiere utilizar un **conjunto de datos representativo de entrada** durante el proceso de cuantizado, con el que se calcularán los parámetros de cuantización de cada tensor.

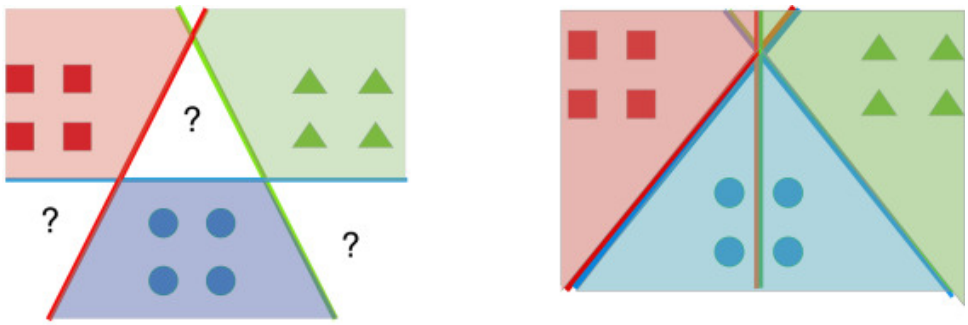
Resultados de la cuantización

Si bien este proceso es muy eficiente, tiene un inconveniente muy importante, y es que la salida de una neurona está limitada a tantos valores como como el tipo de dato que utilicemos, en caso de int8 solo tenemos 256 valores.

Por otro lado, esto no debería ser ningún problema si trabajamos con tareas de clasificación, pero es importante comprobar con un conjunto de prueba que la precisión de la red no se haya visto afectada por la cuantización.

3.3. Ensembles, OVA y OVO

Existen múltiples técnicas para clasificar conjuntos de datos con múltiples clases. En este aparatado vamos a comentar dos tipos de ensembles que unen varios clasificadores binarios para obtener un único clasificador multiclase. En algunas situaciones, los ensembles binarios pueden ser mucho más robustos que un único clasificador multietiqueta.



(a) Separation with OvA. (b) Separation with OvO.

Figura 3.6: Comparación entre OVA y OVO

One vs All

El ensemble One vs All (OVA) divide un problema con N clases en N clasificadores binarios. Cada uno de estos clasificadores se encarga de identificar si una entrada se corresponde con una clase (1) o no (0). En este caso, es importante que el clasificador esté calibrado, es decir, su salida debe asemejarse con la confianza del resultado, ya que, si nos encontramos con un conflicto entre varios clasificadores base por tener la misma salida, no vamos a poder predecir la clase correctamente.

Ejemplo Red Neuronal Binaria

Si utilizamos la **función de activación escalón** en la neurona de salida de nuestra red neuronal, nos encontramos con que la salida únicamente tiene dos valores, 0 y 1, por lo que no conocemos la confianza de la predicción.

Por otro lado, si utilizamos la **función sigmoide** como función de activación, nos encontramos con una función cuyo recurrido son valores reales en el intervalo [0, 1], por lo que podríamos considerar que la salida es la confianza que tiene la red ante una predicción.

One vs One

El ensemble One vs One (OVO), a diferencia del ensemble OVA, busca una clasificación más robusta entrenando un mayor número de clasificadores.

Si en el ensemble OVA únicamente lidiábamos con las colisiones teniendo en cuenta la confianza, el ensemble OVO es capaz de conocer si un dato pertenece a una clase o a otra, ya que cada uno de los clasificadores se entrena con un subconjunto de datos formado solo por dos clases.

$$M = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 1 & 0 \\ 0 & -1 & 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{bmatrix}$$

Figura 3.7: Matriz para N=4

En este caso, tenemos que entrenar $L = K(K-1)/2$ clasificadores binarios, un número mucho más grande que en el ensemble OVA, por lo que el coste en memoria y tiempo de entrenamiento es mucho mayor.

En el caso del ensemble OVO, el clasificador base no solo tiene que estar correctamente calibrado, sino que además las salidas deben estar comprendidas en el intervalo [-1, 1], siendo -1 la clase A, 1 la clase B, y 0 el valor correspondiente la clase desconocida.

Para evaluar la salida de todos los clasificadores, debemos construir una matriz NxL (3.3), que permita convertir el vector L a un vector con N elementos con la confianza de cada clase. Cada fila de la matriz se corresponde con cada clase, y debemos asignar el mismo valor (1 o -1) que hemos utilizado como etiqueta durante el entrenamiento

Haremos una multiplicación de matrices entre la matriz y la salida de los clasificadores, obteniendo una matriz N x L con la probabilidad de cada clase. Como hemos utilizado valores en el intervalo [-1, 1] como salida de cada clasificador, al realizar la multiplicación de matrices nos encontraremos con que únicamente tenemos valores positivos.

Si sumamos todas las columnas, de la matriz anterior obtendremos la confianza del ensemble para cada uno de los datos.

3.4. Coeficientes Cepstrales en las Frecuencias de Mel

Los coeficientes cepstrales en las frecuencias de mel o MFCC son un tipo de representación de la densidad espectral del sonido que intentan representar la percepción auditiva humana.

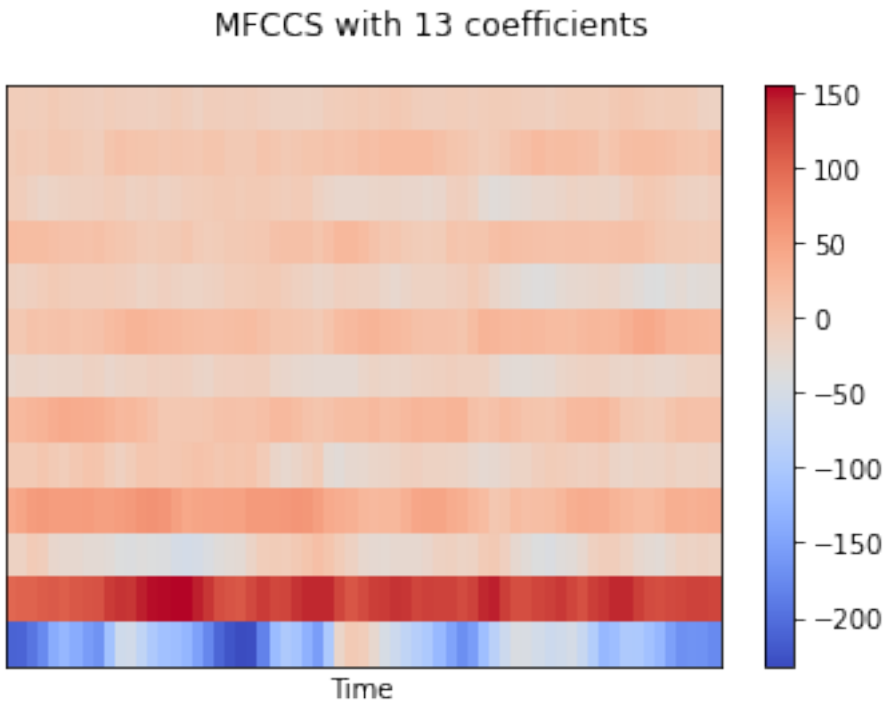


Figura 3.8: MFCC de un fragmento de 30s de una canción, extraido de spotmyfm/Ludwig/notebooks/gtzan/mfcc.ipynb

Son muy utilizados en tareas de reconocimiento del habla y recuperación de información musical.

Estos coeficientes están ajustados sobre la **escala de mel**, una escala que aproxima la escala de tono a una escala similar a la percepción humana, mejorando la representación del sonido.

Para obtener los MFCC partimos del espectrograma de mel, un espectrograma ajustado a la escala de mel. En este caso, estamos intentando obtener el espectrograma de un espectrograma, un cepstrum.

Un método para calcular los MFCCs puede ser el siguiente:

1. Separar la señal en segmentos (hop_length)
2. Aplicar la Transformada Discreta de Fourier a cada segmento para obtener la **potencia espectral**.
3. Transformar los resultados del paso 2 a la escala de mel.

4. Aplicar el logaritmo a cada frecuencia de mel, (escala logarítmica en **decibelios**).

5. Aplicar la Transformada Discreta del Coseno al resultado del paso 4

Los pasos 1-3 se corresponden con la obtención de un espectrograma de mel.

Técnicas y herramientas

Este capítulo tiene la finalidad de resaltar las distintas tecnologías, técnicas y herramientas que han sido utilizadas a lo largo del desarrollo del proyecto.

4.1. Tecnologías y Dependencias

Frontend

Typescript

Typescript es un superset del lenguaje de programación Javascript, por lo que cualquier código escrito en Javascript puede ser utilizado como código Typescript. La principal característica de este superset es la inclusión de un sistema de tipado estático fuerte.

Para el desarrollo de los componentes se ha utilizado TSX (Typescript Syntax Extensions), esta extensión de Typescript permite generar HTML junto con Typescript, facilitando la generación de HTML dinámico.

Estas extensiones de lenguaje no están reflejadas en el estándar de ECMAscript, por lo que es necesario utilizar un transpilador que convierta estas extensiones en código Javascript.

Se ha tenido en cuenta utilizar JavaScript y JSX como alternativa.

NextJS

NextJS es un framework de Javascript compatible con Typescript que permite construir interfaces. NextJS está construido sobre la biblioteca ReactJS, y añade una gran variedad de características como:

- Rutas o Páginas (/home, /about, etc)
- Server Side Rendering
- Rutas de API basadas en NodeJS
- Empaquetamiento sin configuración
- Api de internacionalización

ReactJS es una biblioteca de desarrollo de interfaces web que incluye un Virtual DOM que permite montar y desmontar componentes JSX en el DOM de manera transparente.

Se ha tenido en cuenta utilizar ReactJS como una alternativa.

Tailwind CSS

Tailwind CSS es una biblioteca de generación de hojas de estilo CSS de código abierto muy interesante. Su principal filosofía es la de la creación de clases de utilidad dinámicas con las que podemos seguir buenas prácticas de CSS. Esta biblioteca está acompañada con un preprocesador que se encarga de generar y optimizar las stylesheets a partir de las clases que hemos utilizado.

Una de sus características más interesantes es la generación de clases de utilizado con variantes que permiten cambiar el estilo dependiendo de algunos condicionales

Algunas de estas variantes de estado pueden ser:

- hover:<clases de utilidad> estilo cuando el ratón está encima de un elemento
- sm:<clases de utilidad> estilo cuando el ancho del viewport es mayor a 640px
- dark:<clases de utilidad> estilo a aplicar cuando estamos utilizando el modo oscuro
- ...

Por último, esta biblioteca permite declarar nuestras propias paletas de colores, variantes de estado, clases de utilidad, etc, por lo que podemos expandirla manualmente.

Twin Macro y Styled Components

En la actualidad, debido a las limitaciones de las hojas de estilo en cascada, existen múltiples métodos de escribir estilos CSS

- Ficheros .css
- Estilos en línea
- Estilos como componentes
- CSS in JS

Para esta aplicación hemos utilizado el paradigma CSS in JS, que nos permite declarar los estilos CSS en el propio código de la aplicación. La principal ventaja de este paradigma es que podemos modificar los atributos CSS directamente desde nuestro código.

Twin macro es un adaptador de distintas bibliotecas **CSS in JS**, que permite estilar mediante TailwindCSS con bibliotecas CSS in JS como PostCSS, Emotion o Styled Componets.

En este caso hemos escogido la biblioteca **Styled Coponentes**, que nos permite declarar o estilar componentes de ReactJS.

Zustand

Zustand es una biblioteca de código abierto para la gestión de estado de React basada en hooks. Esta biblioteca permite declarar stores (espacios de almacenaje de estados) en los que podemos almacenar datos y sus operaciones que deseamos compartir entre varios componentes.

En el caso de la aplicación, hemos utilizado esta biblioteca para almacenar estados globales, como el estilo de la web, la pestaña que tenemos activa o instancias únicas de los distintos clientes REST de la aplicación.

DexieDB

DexieDB es un wrapper de IndexedDB, una **base de datos** NO-SQL que se encuentra en todos los navegadores modernos.

Este wrapper, a parte de contener toda la funcionalidad de DexieDB, nos permite declarar los esquemas de la base de datos junto sus distintas operaciones y comprobaciones.
DexieDB es compatible con Typescript, por lo que permite operar las tablas con supervisión de tipos.

Jest

Jest es una biblioteca de código abierto de pruebas unitarias desarrollada por Facebook. Esta biblioteca tiene una gran variedad de características, como Mocking, compatibilidad con código asíncrono, compatibilidad con React, Typescript, Node, etc.

Una de las principales ventajas de Jest son las extensiones de su comunidad. En este caso se ha utilizado la extensión **React Testing Library**, una extensión que permite renderizar internamente componentes de React y hacer comprobaciones sobre el DOM Virtual.

Cypress

Cypress es un framework de pruebas punto a punto de código abierto. Cypress es muy similar a Selenium, ambas realizan pruebas automatizadas sobre un navegador físico, pero Cypress es mucho más sencillo de configurar.

Algunas de las características más relevantes de Cypress son:

- Mayor velocidad de ejecución
- Tests escritos en Javascript
- Dashboard y grabación de ejecución de pruebas
- Tamaño de viewport variable (Tests para tablets, móviles, escritorio, etc)
- Ejecución headless
- Compatibilidad con Github Actions

Otras:

- **Recharts**: Es una biblioteca de código abierto para React que permite generar gráficas interactivas.
- **React Icons**: Una colección de iconos vectoriales que junta múltiples bibliotecas de iconos.
- **Framer Motion**: Biblioteca de código abierto que permite añadir animaciones que dependen de los estados internos de un componente.

Backend NextJS

NextJS API REST

NextJS incluye una API REST funcional basada muy simple basada en Node/Express. Esta API tiene como principal objetivo apoyar al funcionamiento del cliente en dos situaciones.

- Renderizar en el servidor (SSR) elementos dinámicos de la web. Como por el ejemplo vistas individuales para usuarios registrados.
- Trabajar con peticiones que requieran autenticación.

En este api, un endpoint se define como un fichero .ts/.js que recibe dos objetos, la petición HTTP y la respuesta HTTP. Como cada endpoint se corresponde con un fichero único, la API permite el paradigma de cloud computing FaaS (Functions as a Service), en el que cada función puede actuar como un pequeño microservicio.

Cada endpoint puede hostearse de manera independiente en plataformas como Cloudflare Workers o AWS Lambda.

JWT

JWT (Json Web Tokens) es un método de representación de tokens en formato json basado en el estándar RFC 7519. Los tokens están formados por tres segmentos:

- Cabecera: Es un json serializado en base64 que incluye el tipo del token y el algoritmode firma utilizado (RSA, SHA256, etc)
- Payload: Es otro json serializado en base 64 que incluye los contenidos del token.
- Firma de verificación: Firma hash generada a partir de una clave privada y los campos anteriores. Permite comprobar que el payload no ha sido modificado.

Una vez construidos y serializados los 3 segmentos, estos son concatenados mediante un punto (“.”) y pueden ser deserializados por cualquier receptor.

DynamoDB y Dynamoose

DynamoDB es una base de datos NO-SQL basada en documentos clave valor desarrollada por Amazon para AWS, su plataforma de Cloud Computing.
Algunas características de esta base de datos son:

- Capacidad bajo demanda
- Base de datos sin servidor (No utiliza sockets / conexiones físicas, por lo que es apta para ser accedida por la arquitectura sin servidor, como CaaS o FaaS.
- Compatibilidad con AWS OpenSearch (ElasticSearch)
- Copias de seguridad
- Replicas
- Transacciones ACID

Además, DynamoDB está incluida en el plan gratuito de AWS, y ofrece 25GB de por vida.

Dynamoose es una herramienta de modelado muy similar a Mongoose, que permite utilizar las distintas características de AWS desde un api de alto nivel.

La principal ventaja de Dynamoose es que permite modelar los esquemas de la base de datos en Typescript/Javascript. Estos esquemas serán validados por Dynamoose antes de crear, modificar o leer documentos.

Otras Bases de datos planteadas

	Firestore	MongoDB	Supabase	DynamoDB	Heroku
Tipo	Documentos	Documentos	Relacional	Documentos	Relacional
Capacidad	1GB	0.5GB	0.5GB	25GB	1000 filas
Lim. Peticiones	50k R/doc/day 20k W/ doc/day	No hay	No Hay	200M R+W /mes	No
Self-hosted	No	Si	Si	NO	Si
SSL	Si	Si	Si	Si	Si
Escalable	Si	Si	Si	Si	Si
Pausa	No	No	Si (7 días)	No	Si (7 días)
ORM/ Bussiness Logic / Model / Validation	No	Si (Mongoose)	No	Si (Dynamoose)	Si (Postgres)
Editor	Si	Si	Si	Si (De pago)	Si

Backend MIR

El backend de **Music Information Retrieval (MIR)** está basando en Python 3.10

FASTAPI

FastAPI es un framework web diseñado para construir APIs con Python 3.6+.

La principal característica de FastAPI es que utiliza el sistema de sugerencia de tipos de Python para validar las distintas peticiones HTTP.

Este sistema permite asignar un tipo de forma estática a una variable, estos tipos son ignorados en tiempo de ejecución, pero pueden se utilizados para documentar o como comprobación en tiempo de desarrollo mediante un linter.

```
foo: int = 1 # No error
foo: int = "bar" # typechcking error
```

Además de validar las peticiones de nuestra aplicación, FastAPI genera automáticamente **documentación interactiva** para cada endpoint con **OpenAPI y Redocs**.

Por último, FastAPI utiliza el api de **corrutinas** de Python basada en la sintaxis Async/Await. Cada uno de los endpoints puede ser definido como una corrutina asíncrona con la que podemos esperar a ejecución de otras corrutinas utilizando la palabra clave await, facilitando el uso de bibliotecas concurrentes.

AIOHTTP

aiohttp es una biblioteca de Python basada en **requests** que permite realizar **peticiones http concurrentes**. aiohttp utiliza el api de corrutinas, por lo que puede sen consumida directamente desde un endpoint de FastAPI.

ONNX Runtime Python

ONNX Runtime es una biblioteca de código abierto mantenida por Facebook y Microsoft. Esta biblioteca implementa múltiples motores de inferencias compatibles con el estándar **Open Neural Network eXchange**

(ONNX), un estándar que busca la interoperabilidad de modelos neuronales a partir de una **representación intermedia (IR).**

La filosofía de ONNX es que un modelo neuronal pueda ser utilizado en cualquier hardware independientemente de su representación interna, para ello existen una serie herramientas para convertir y optimizar modelos basados en Pytorch, Keras, Caffé, etc en representación intermedia.

Un modelo IR puede ser utilizado en múltiples **plataformas**

- Windows
- Linux
- Android
- IOS
- Navegador
- ...

Independientemente de la **arquitectura**

- X86
- X64
- ARM64
- ...

Desde múltiples **lenguajes de programación** o **entornos de ejcución**

- Python
- NodeJS / Javascript / Typescript
- C++
- Java
- ...

Y compatible con una gran variedad de APIs para la aceleración por Hardware

- CPU
- Nvidia Cuda
- Nvidia TensorRT
- Intel OpenVINO
- Windows DirectML
- ...

Librosa

Librosa es una biblioteca de Python enfocada al análisis de audio y música. Esta biblioteca está basada en Scipy y es muy utilizada a la hora de trabajar con sonido en Python.

Esta biblioteca incluye todos los componentes necesarios para poder trabajar con audio, desde la lectura / creación de ficheros de audio, conversión entre formatos, muestreo, visualización y extracción de características.

En este apartado del proyecto utilizamos librosa para leer ficheros wav y extraer algunos datos como:

- MFCCs
- Tempo
- Beat
- Descomposición del espectrograma

Se tuvo en cuenta el uso de Torch Audio para las tareas de preprocesamiento.

FFMPEG

FFMPEG es conjunto de bibliotecas de **alto rendimiento** que permiten trabajar con datos multimedia (ficheros y streams de vídeo, audio, imágenes, etc). FFMPEG permite grabar, editar, convertir, transcodificar o escalar datos multimedia.

Este backend utiliza FFMPEG para convertir y muestrear ficheros .mp3 a .wav en 22050HZ rápidamente. Esto es debido a que Librosa no es compatible de forma nativa con ficheros .mp3.

Sklearn

Sklearn es una biblioteca basada en scipy diseñada para tareas de aprendizaje automático.

Incluye una gran variedad de algoritmos para tareas de clasificación, regresión, clustering, reducción de dimensionalidad, selección de modelo o preprocesado.

Este backend utiliza dos componentes de Sklearn:

- **BallTree**: Utilizado para encontrar los vecinos más cercanos a un dato.
- **MinMaxScaler**: Utilizado para normalizar la entrada al BallTree

Scikit Surprise

Surprise es una biblioteca de código abierto especializada en sistemas de recomendación colaborativos.

El funcionamiento de esta biblioteca es muy simple, ya que permite generar la matriz de recomendación a partir de una lista formada por:

- Id Usuario
- Id Elemento
- Votación numérica del Usuario

Surprise implementa una gran variedad de algoritmos, como SVD o NMF utilizados en descomposición de matrices, y otros más tradicionales como KNN, Co Clustering o Slope One.

Por otro lado, Surprise además implementa un sistema de evaluación de recomendaciones, que valora la precisión del modelo a partir de intentar completar recomendaciones para usuarios incompletos.

4.2. Herramientas y Servicios

Kaggle

Kaggle es una comunidad de ciencia de datos en la que se publican conjuntos de datos y competiciones.

Uno de los servicios que incluye Kaggle es **Code**, un servicio para publicar y ejecutar notebooks que utilicen el Dataset, esta ejecución se realiza en unos contenedores virtuales con las siguientes características:

- 4 Núcleos
- 12GB de RAM
- NVIDIA TESLA P100 de 16GB (opcional)
- Google TPU (ocional)

En este caso se ha utilizado Kaggle para publicar los datasets y entrenar los modelos de forma remota.

Se ha planteado el uso de Google Cloud Notebooks y Google Colab para el entrenamiento de los modelos.

Tensorflow Keras

Tensorflow es una biblioteca de código abierto destinada a el entrenamiento e inferencia de modelos neuronales. Tensorflow está pensada para ser usada con Python, pero es compatible con otros lenguajes como C++ o incluso JavaScript.

Keras es una API de Tensorflow para Python que simplifica la implementación y entrenamiento de redes neuronales mediante Tensorflow.

Se ha planteado el uso de Pytorch como alternativa a Tensorflow Keras.

ONNX Tensorflow Converter y ONNX Quantizer

ONNX Tensorflow Converter es una herramienta de línea de comandos que permite convertir un modelo de Tensorflow en un modelo del estándar ONNX.

ONNX Quantizer es una biblioteca de Python que permite cuantizar modelos de ONNX. Esta herramienta es compatible tanto con la cuantización dinámica como con la estática.

Ambas herramientas forman parte del proyecto de código abierto ONNX, mantenido por Microsoft y Facebook.

Se tuvo en cuenta el uso de TFLITE para cuantizar e inferir los modelos, pero este motor únicamente está optimizado para dispositivos móviles (ARM).

Github

Github es una plataforma para hostear repositorios git. Esta plataforma permite crear y compartir repositorios públicos y privados, utilizar ramas, crear y gestionar pull requests, y muchas otras características como la creación de flujos de trabajo con Github Actions o el control de dependencias con Dependabot.

Se ha utilizado Github y su sistema de Issues para el control de versiones del proyecto y la planificación temporal del proyecto.

Zenhub

Zenhub es una herramienta de gestión de proyectos que intenta integrarse con Github. Esta biblioteca presenta un tablero Kanban en el que podemos encontrarnos cada una de las tareas del propio repositorio.

Zenhub, además tiene un sistema de sprints, épicas y puntos de historia con el que podemos gestionar proyectos utilizando la metodología Scrum.

Dependabot

Dependabot es un servicio integrado en Github que permite analizar un respositorio en busca de brechas de seguridad.

Este servicio analiza el código en busca de **filtraciones de claves o tokens secretos**, y como su propio nombre indica, analiza las **dependencias del proyecto** en busca de dependencias vulnerables.

Si Dependabot detecta una vulnerabilidad en el proyecto, este intentará corregirla y publicará una **Pull Request** con la vulnerabilidad solucionada.

Docker

Docker es una plataforma que permite crear y desplegar contenedores. Un contenedor es un "paquete" que contiene todos los requisitos para que una aplicación se pueda ejecutar en un entorno aislado, como el binario de la aplicación, dependencias, configuraciones de red, etc.

Estos contenedores pueden ser considerados una pequeña máquina virtual, ya que cada contenedor tiene su propio sistema de ficheros, unidades, o LAN, pero a diferencia de una máquina virtual, el contenedor se ejecuta sobre el **Docker Engine** y no sobre un **hipervisor.**

En este caso se ha utilizado Docker para crear los contenedores con los que poder desplegar las aplicaciones de NextJS y FastAPI.

Github Copilot

Github Copilot es un servicio online que permite sugerir cambios o implementaciones de código basado en el contexto actual del fichero que estamos editando. Copilot se integra con el editor (VSCODE) o IDE (Webstorm) y necesita conexión a internet para procesar las sugerencias.

La principal ventaja de Copilot frente a otros servicios de autocompletado como Intellisense es que Copilot utiliza los tipos de datos, comentarios, estilo del código, nombres de variables y nombres de funciones para sugerir cambios o implementaciones.

Github Actions

Github Actions es un servicio integrado en Github que permite ejecutar flujos de trabajo sobre un repositorio a partir de distintos estados.

En este caso, se han programado las siguientes acciones para cada push al repositorio:

- **Prettier**: Aplica el formateador de código Prettier a todos los ficheros compatibles (Typescript, Javascript, JSON, CSS, etc)

- **Integración Continúa**: Asegura que el proyecto se compila correctamente y ejecuta las pruebas unitarias sobre las distintas clases del proyecto.

- **Cypress:** Ejecuta las pruebas punto a punto sobre el despliegue de la aplicación.

Se ha considerado el uso de los servicios TravisCI y CircleCI

Vercel (Plataforma)

Vercel es una plataforma para publicar proyectos de frontend y webs estáticas. Está desarrollado por la empresa Vercel, la misma empresa que desarrolla NextJS, por lo que es compatible con todas las funcionalidades de NextJS.

Vercel ofrece muchas otras características como:

- Compatibilidad con **Edge Functions** (FaaS)

- Despliegue Continuo desde un repositorio

- SSL

- Dominios Personalizados

Se ha planteado el uso de la plataforma Netlify pero se ha escogido Vercel por su integración con NextJS.

Google Cloud Run

Google Cloud Run es un servicio de Google Cloud que permite el despliegue de contenedores (CaaS) de forma escalable.
Su funcionamiento es muy simple, a partir de un contenedor con un puerto expuesto, Cloud Run creará un proxy para dicho puerto y dependiendo del tráfico el orquestador instanciará o mantendrá activos un número variable de contenedores.

Visual Studio Code

Visual Studio Code es un editor de código de código abierto desarrollado por Microsoft. Este editor está construido sobre Chromium y Electron, con una filosofía en la que la interfaz está totalmente separada del editor, por lo que el editor en si puede encontrarse en un servidor, máquina virtual o local, mientras que editamos desde un navegador web o la propia aplicación de escritorio.

La principal ventaja de Visual Studio Code es su gran comunidad, que ha desarrollado un gran número de extensiones para ajustar el desarrollo del editor para cada uno de los casos de uso.

OpenAPI

La especificación OpenAPI es una especificación para definir la interfaz de un servicio REST. Esta interfaz puede declararse mediante un fichero. json o .yaml, y permite declarar los distintos endpoints de un servidor, así como las peticiones y respuestas esperados.

La principal ventaja de OpenAPI es que puede ser integrada con un middleware para ofrecer validación de la API.

Por otro lado, la especificación OpenAPI permite generar una interfaz interactiva con la que poder generar automáticamente peticiones de muestra a partir de la especificación de peticiones.

Aspectos relevantes del desarrollo del proyecto

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto, comentados por los autores del mismo. Debe incluir desde la exposición del ciclo de vida utilizado, hasta los detalles de mayor relevancia de las fases de análisis, diseño e implementación. Se busca que no sea una mera operación de copiar y pegar diagramas y extractos del código fuente, sino que realmente se justifiquen los caminos de solución que se han tomado, especialmente aquellos que no sean triviales. Puede ser el lugar más adecuado para documentar los aspectos más interesantes del diseño y de la implementación, con un mayor hincapié en aspectos tales como el tipo de arquitectura elegido, los índices de las tablas de la base de datos, normalización y desnormalización, distribución en ficheros3, reglas de negocio dentro de las bases de datos (EDVHV GH GDWRV DFWLYDV), aspectos de desarrollo relacionados con el WWW... Este apartado, debe convertirse en el resumen de la experiencia práctica del proyecto, y por sí mismo justifica que la memoria se convierta en un documento útil, fuente de referencia para los autores, los tutores y futuros alumnos.

Trabajos relacionados

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

Conclusiones y Líneas de trabajo futuras

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

Bibliografía