CREACIÓN Y ANÁLISIS DE MODELOS PARA MUSIC OBJECTS SPOTTING

Pablo Mora Claros

Resumen– Mediante la técnica empleada en este proyecto de detección y reconocimiento de objetos musicales, es posible recuperar y buscar pasajes musicales en archivos de partituras para su preservación y tratamiento. El objetivo es crear un buscador de pasajes musicales para encontrar similitud entre diferentes partituras. En este proyecto se han desarrollado y analizados un modelo de Machine Learning y un modelo Deep Learning para dicha técnica (Music Spotting) mediante métodos de OMR (Reconocimiento Óptico de Música). El resultado de este trabajo sirve para poder planificar el desarrollo de una aplicación real para el uso cotidiano.
Palabras clave– OMR, Music Object Spotting, HOG, BSM, SVM, YOLO, Machine Learning, Deep Learning.

Abstract– Using the technique employed in this musical object detection and recognition project, it is possible to retrieve and search for musical passages in score archives for preservation and processing. The aim is to create a musical passage search engine to find similarities between different scores. In this project, a Machine Learning model and a Deep Learning model for this technique (Music Spotting) have been developed and analysed using OMR (Optical Music Recognition) methods. The result of this work serves to plan the development of a real application for everyday use.
Keywords– OMR, Music Object Spotting, HOG, BSM, SVM, YOLO, Machine Learning, Deep Learning.

1 INTRODUCCIÓN

El Reconocimiento Óptico de Música (OMR) es la tecnología del reconocimiento de objetos musicales dentro de una partitura para posteriormente tratarla y realizar alguna operación, como por ejemplo, poder editar las partituras escaneadas con el paso previo de convertirlo a objetos musicales, o directamente, a partir de una partitura escaneada, generar un archivo sonoro para el posterior tratamiento. Así, se puede preservar partituras clásicas antiguas para tenerlas en un formato digital y hacerlo más accesible y fácil de consultar. Pese a ello, sigue siendo una de las grandes dificultades. Una de las opciones para poder hacerlas más accesibles es hacer búsquedas mediante similitud visual. Es decir, hacer un buscador de pasajes musicales.

El propósito del trabajo es hacer un buscador de objetos musicales (Music Spotting) para que de esta manera los musicólogos e historiadores, o cualquier persona interesada en partituras, puedan buscar pasajes musicales en las partituras a través de la similitud visual entre una consulta y un colección de partituras. Esto se a llevado a cabo, creando un modelo de Machine Learning, desarrollado mediante descriptores HOG y BSM y un clasificador SVM, y un modelo de Deep Learning, desarrollado mediante un modelo YOLO entrenado con partituras sintéticas. Posteriormente se analizan los resultados que brindan cada uno de los modelos y así poder extraer unas conclusiones precisas.

Lo que hace novedoso a este proyecto es la aplicación de la idea de poder buscar similitudes visuales en objetos musicales en vez de en texto como ya se ha realizado en otros trabajos y por ende ya existe.

El artículo está organizado de la siguiente manera: en la sección 2 se describen cuales son los objetivos establecidos para el proyecto. En la sección 3 se comenta la metodología utilizada y la planificación que se ha llevado a cabo. En la sección 4 se resume el estado del arte sobre los temas que tocan de lleno a este proyecto (OMR y Spotting). En la sección 5 se describe el desarrollo general del proyecto empezando por el modelo de Machine Learning y todos sus componentes, y acabando por el modelo Deep Learning. En la sección 6 se explica los experimentos que se han llevado a cabo y los resultados obtenidos. Finalmente, en la sección 7 se resume el trabajo realizado junto a unas conclusiones y trabajos futuros.

2 OBJETIVOS

El objetivo principal de este proyecto es realizar Music Spotting mediante dos modelos diferentes y poder analizar los resultados. Un modelo Machine Learning clásico y un

• E-mail de contacto: pablo.mora@e-campus.uab.cat
• Mención realizada: Computació
• Trabajo tutorizado por: Alicia Fornés (Ciencias de la Computación)
• Curso 2021/22

modelo Deep Learning.

Una vez definido el objetivo global del proyecto, se puede subdividir en diferentes subobjetivos:

- **Estudio del arte:** Para ver como de desarrollado está el mundo del OMR actualmente y que métodos y técnicas se han utilizado, se realiza un estudio del arte en el que se repasan los trabajos relacionados con la detección y clasificación de objetos musicales y métodos de spotting para otros campos no relacionados con la música.
- **Creación modelo Machine Learning clásico:** Utilizar un método de Machine Learning que no precise de tanto entrenamiento, como SVM, para poder detectar pasajes musicales similares en diferentes partituras. Este tipo de modelo ya existe para Word Spotting. La idea es aplicar una estructura similar para objetos musicales y ver su rendimiento.
- **Creación modelo Deep Learning:** Utilizar método de Deep Learning de detección de objetos para adaptarlo a objetos musicales y poder buscar similitudes en una o varias partituras.
- **Generación de los datos para los diferentes modelos para poder testearlos:** Para poder evaluar los modelos generados, es necesario tener una base sólida de imágenes de partituras. Cada uno de los modelos necesita unos datos de entrada distintos, por eso, es necesario generar y procesar los datos de entrada.
- **Análisis y comparación entre métodos:** Una vez se obtengan los resultados de aplicar los diferentes modelos para un mismo problema, se realiza un análisis de los resultados obtenidos y una breve comparativa entre ellos.

3 METODOLOGÍA

Para la realización del proyecto se ha utilizado una metodología ágil caracterizada por proporcionar una estructura incremental, iterativa, susceptible a cambios y flexible para incorporar nuevas fases si fuera necesario. Esto facilita la idea de ir realizando el proyecto paso a paso de forma incremental en cuanto a funcionalidad y dificultad.

El proyecto se divide en las siguientes fases:

- **Estudio del arte:** Fase importante de preparación del proyecto para conocer los avances que se han hecho en el mundo del OMR, las técnicas que se han utilizado para detectar objetos musicales y entender como funciona y en que se basa el Spotting. Aparte de esto, también se ha hecho una búsqueda de información sobre los métodos utilizados para desarrollar los modelos propios de este proyecto.
- **Creación modelo Machine Learning:** En esta primera fase se crea la baseline del proyecto, llegando a implementar dos descriptores para las imágenes y un clasificador SVM. Se ha realizado de forma gradual en cuanto a detección de objetos musicales desde detección a nivel de compás hasta detección de pasajes musicales complejos en diferentes partituras.
- **Creación modelo Deep Learning:** Una vez existe la baseline del proyecto, es momento de crear el modelo de Deep Learning para poder compararlo con el modelo anterior. En dicha fase, se ha tenido que hacer un trabajo de generación y etiquetado de los datos superior al anterior.
- **Análisis de los modelos:** Una vez ambos modelos funcionan correctamente y se han testeado de forma separada, es momento de analizarlos para ver cual de los dos es más eficiente y sacar una conclusión de cual es mejor para desarrollo de un software para el usuario.

4 ESTADO DEL ARTE

Para poder comprender de mejor manera el desarrollo que se ha llevado a cabo de OMR, se ha hecho un estudio sobre aquellos trabajos o investigaciones que tienen un impacto sobre el trabajo que se expone en este artículo. Se va a empezar a hablar sobre proyectos relacionados con la detección de objetos musicales mediante diferentes modelos de Machine Learning y Deep Learning, para posteriormente nombrar aquellos proyectos más relacionados con el proyecto actual de Spotting.

Por una parte, se han realizado diferentes trabajos con el propósito de detectar objetos musicales mediantes modelos más sencillos de Machine Learning como el [1] en el cual mediante los descriptores HOG y un modelo SVM es capaz de detectar los objetos musicales establecidos. A su vez, en el proyecto [2] proponen 3 tipos de descriptores de características diferentes para la posterior clasificación mediantes SVM de partituras manuscritas. Por último, dentro del apartado de usos de modelos de Machine Learning se encuentra el trabajo realizado por Rebelo [3] en el que se comparan diferentes métodos como HMM, KNN y SVM para el mismo propósito que las anteriores.

Por otro lado, existen diferentes proyectos en los cuales mediante redes neuronales se ha conseguido detectar objectos musicales. Uno de los primeros trabajos fue el de Pacha y Jorge Calvo [4] en el cual mediante una red neuronal Faster-RCNN son capaces de detectar las clases a las que pertenecen los objetos musicales en partituras manuscritas. Otro de los trabajos, también desarrollado por Alexander Pacha y Jorge Calvo [5], muestra una comparación entre diferentes tipos de modelos para el mismo propósito de detección de objetos musicales. En él se utiliza un modelo de RetinaNet, similar al modelo que se presenta en este proyecto, YOLOV5. A diferencia de este proyecto, en ellos solo se detectan objetos musicales individualmente, mientras que en este proyecto se detectan y buscan pasajes musicales (p. ej. un compás entero o una línea de pentagrama entera).

Para acabar vamos a hacer más hincapié a trabajos más relacionados con lo que se pretende seguir en este proyecto, la de poder hacer una búsqueda de pasajes musicales mediante similitud visual. Para poder hacernos una idea, más gráfica e interactiva, de lo que se pretende conseguir con este proyecto nos podemos fijar en el trabajo desarrollado por P. Riba [6] en el cual mediante una aplicación móvil es capaz de seleccionar una palabra del manuscrito para posteriormente encontrar dentro del mismo documento palabras similares a la seleccionada. La base de este proyecto es el trabajo [7] en el cual mediante un descriptor HOG, un clasi-

ficador SVM y la idea de una ventana deslizante, son capaces de generar un modelo que encuentre las coincidencias visuales a una consulta pasada.

5 CREACIÓN DE MODELOS

En esta sección se expone el desarrollo y la estructura seguida para cada uno de los modelos que se han generado comentando los aspectos más importantes a tener en cuenta y los resultados de manera individual.

5.1. Modelo basado en Classical Machine Learning

Para poder crear una baseline del proyecto se ha desarrollado un modelo de Machine Learning capaz de encontrar mediante similitud visual un pasaje musical (conjunto de notas) seleccionado en otras partituras. Para ello, la estructura seguida para poder obtener resultados satisfactorios, se puede observar en la Figura 1 en la que se ven las diferentes etapas del modelo. Mediante el recorte de una imagen de una partitura, la extracción de características de la misma, la ventana deslizante a través de las líneas de pentagrama, la generación de muestras positivas y negativas y la posterior clasificación, el modelo es capaz de localizar las partes de la partitura que son similares a la consulta seleccionada.

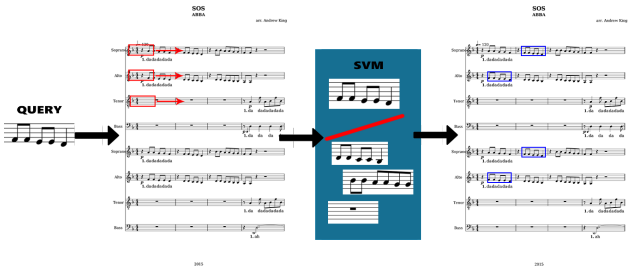


Fig. 1: Flujo de trabajo del desarrollo del modelo SVM

5.1.1. Detección líneas de pentagrama

Para poder desarrollar un modelo que sea capaz de detectar objetos musicales y que a su vez sea eficiente, es necesario hacer una detección de las líneas del pentagrama de una imagen para no tener que hacer ventana deslizante por regiones donde no hay información útil. Para ello, se ha creado un método capaz de detectar líneas de pentagrama de partituras sintéticas y manuscritas mediante métodos de morfología binaria.

A través de un conjunto de operaciones de OPEN y CLOSE se ha conseguido detectar líneas horizontales (Fig. 2) para posteriormente crear las áreas de cada uno de los pentagramas que será la zona de trabajo de la ventana deslizante.

Para finalizar, es necesario saber los puntos (coordenadas) en los que se encuentra cada rectángulo. Como no todos los rectángulos encontrados mediante las técnicas de morfología binaria son del mismo tamaño, se hace una estandarización con el tamaño medio de cada uno.

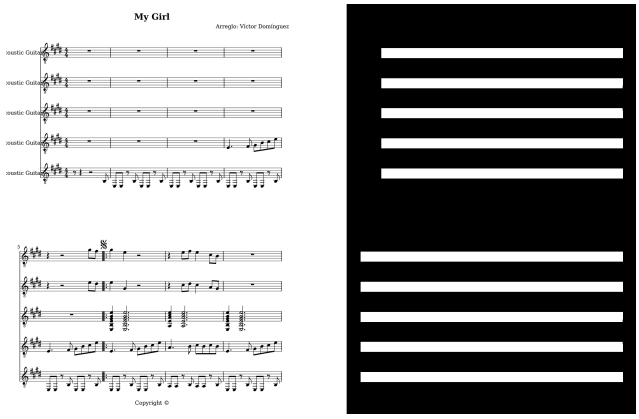


Fig. 2: Líneas de pentagrama detectadas mediante morfología binaria

5.1.2. Ventana deslizante

El siguiente paso a llevar a cabo antes de aplicar cualquier extractor de características es la ventana deslizante por las líneas de pentagrama. Para ello, antes de poder hacerlo, se selecciona una parte de la imagen que se quiera buscar en el resto de la partitura o partituras y que actuará como consulta. De esta manera, la ventana deslizante tendrá la altura y anchura que la consulta realizada por el usuario e igual que las ventanas finales.

Un problema que surge en este paso, es encontrar objetos musicales similares en diferentes imágenes y partituras de diferente tamaño. Este problema se soluciona haciendo ventana deslizante de la altura ideal de cada una de las líneas de pentagrama independientemente del tamaño de la imagen y de la anchura igual a la consulta realizada por el usuario. En la siguiente sección, se comenta como se ha podido resolver este problema para poder comparar vectores de características del mismo tamaño.

5.1.3. Descriptores

La siguiente fase para poder detectar objetos musicales mediante similitud visual es la extracción de características gracias a descriptores. Para ello, se han implementado dos tipos diferentes: HOG y BSM.

El primero de ellos, el descriptor HOG (Histogram of Oriented Gradients) [8], busca determinar la distribución de frecuencia de las orientaciones de los ángulos del gradiente (dirección en la que el cambio de intensidad es máximo tanto verticalmente como horizontalmente). Describe la forma del objeto que se encuentra dentro de la imagen a partir de los gradientes de los bordes que conforman el objeto. Los pasos que sigue este descriptor son los siguientes: cálculo del gradiente (vertical y horizontal) para todos los píxeles de la imagen, división en bloques de la imagen (8x8 píxeles), cálculo del histograma de 9 puntos (dividido por la orientación del gradiente en 20 grados cada uno), normalización de los histogramas y concatenación de histogramas para obtener el vector HOG final. Para este proyecto se ha utilizado una configuración de 16 píxeles por celda.

Para poder comparar un recorte de la imagen con otra, obtenemos, después de aplicar el descriptor, un vector. Dado que no todas las imágenes de partituras tienen el mismo tamaño total, antes de aplicar HOG se cambia el tamaño de

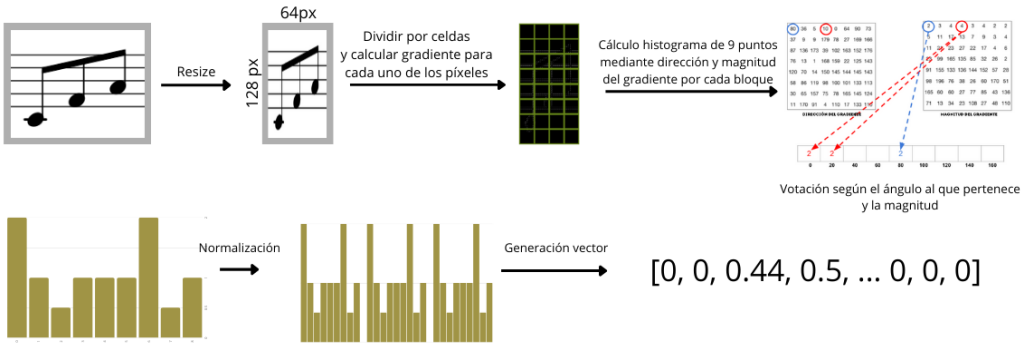
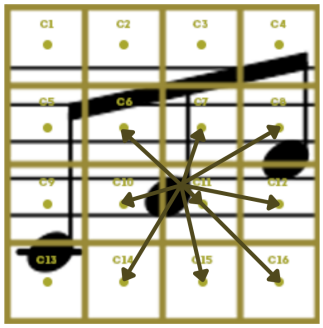


Fig. 3: Cálculo del vector HOG

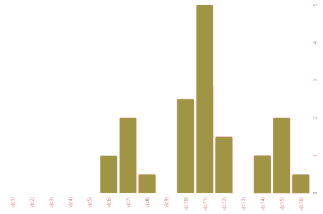
la imagen a 64x128 para que mediante un tamaño de celda de 8x8 los cálculos salgan enteros y sea más fácil de procesarlo. A su vez, hace que cada una de las ventanas, independientemente de su tamaño, acaben resultando en un mismo tamaño de vector HOG final. De esta manera la comparación se puede realizar correctamente siguiendo la estructura y pasos comentados (Fig 3).

Por otra parte, se ha implementado el descriptor BSM en imágenes en escala de grises basado en el trabajo [8] donde gracias a ello, es posible conseguir un descriptor potente para reconocer formas en escala de grises. Dicho descriptor se basa en la división de la imagen en subregiones en la que cada una tiene un centro asociado. Dado que la imagen es en escala de grises, solo son importantes los píxeles de color negro que forman parte de la forma del objeto que se quiere detectar. De esta manera, se calcula, a grandes rasgos, la contribución (distancia) que tiene cada centro (a conectividad a 8 con la subregión a la que pertenece el píxel) con el píxel de color negro. Así se extrae por cada píxel, el "porcentaje" de contribución de las subregiones vecinas. Por último, se acumula toda la información de todos los píxeles que forman parte de la forma del objeto a detectar en un vector que será el resultado de la aplicación de dicho descriptor. En la Figura 4 se observa la idea detrás del cálculo de contribución de los centros asociados a cada uno de los píxeles negros de la imagen.

Para poder lidiar con el problema de diferente tamaño de imágenes, se han realizado los mismos pasos que con el descriptor HOG, ya que lo que se define en este método es el tamaño de las subregiones que queremos crear.



(a) Distancia de un píxel contorno a sus vecinos



(b) Vector descriptor con las distancias de (a)

Fig. 4: Cálculo BSM

5.1.4. SVM

El objetivo es crear un método de Machine Learning que facilite la detección mediante similitud visual entre objetos musicales. A su vez, mediante los descriptores comentados anteriormente y calculando la distancia euclidiana entre la consulta y cada una de las ventanas que forman parte de la partitura, se pueden extraer resultados correctos y eficientes.

Para poder clasificar correctamente cada una de las ventanas generadas y obtener sólo aquellas que se parecen a la consulta que se ha hecho, se ha aplicado un modelo SVM (Support Vector Machine) [10]. Con la premisa de obtener si la ventana es similar o no a la consulta, se han creado dos clases posibles: positiva (si que se parece) y negativa (no se parece). De esta forma, el objetivo del algoritmo SVM es encontrar un hiperplano que separe de la mejor forma posible estas dos clases diferentes de puntos de datos. Encontrando aquel hiperplano que maximice el margen (definido como distancia al hiperplano hasta la primera muestra de datos de cada una de las clases). Para poder obtener un mejor resultado se utilizan "kernel" que otorgan la posibilidad de añadir una dimensionalidad extra para poder separar mejor el conjunto de datos en las dos clases. En el caso de este proyecto, se ha utilizado el "kernel" sigmoidal.

Dado que las partituras no están previamente etiquetadas debido a que no se sabe cuales son las clases positivas y cuales son las clases negativas, es necesario realizar una cantidad suficiente de muestras positivas y muestras negativas en tiempo de ejecución. Esto quiere decir que el modelo SVM, cada vez que se quiera hacer una búsqueda, se va a entrenar con nuevas muestras positivas y nuevas muestras negativas dependiendo de la partitura que se esté visualizando.

Para la primera de ellas, se generan X imágenes iguales que la consulta pero con ligeros desplazamientos horizontales a ambos lados. De esta manera se obtienen muestras de la consulta desde diferentes vistas (Fig. 5).

Para el conjunto de muestras negativas se realiza un

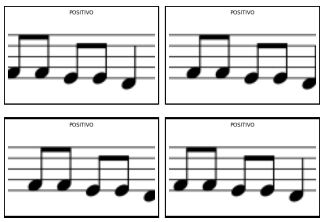


Fig. 5: Muestras positivas

método el cual genera X imágenes aleatorias de las ventanas de toda la partitura que previamente se han calculado. Para asegurar que se generan suficientes muestras negativas de manera aleatoria, se divide el total de muestras que queremos generar por la cantidad de líneas de pentagrama que tenemos, para poder tener la suficiente variedad de información (Fig. 6).

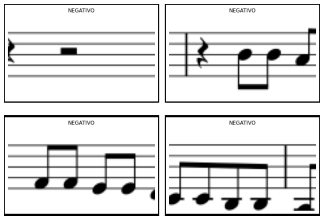


Fig. 6: Muestras negativas

Finalmente, después de entrenar el modelo SVM con las muestras generadas, se calcula el descriptor de cada una de las ventanas de las líneas de pentagrama de la cantidad de partituras que se desee, para poder predecir si pertenecen o no al conjunto de muestras positivas y, por lo tanto, si guardan similitud visual con la consulta realizada.

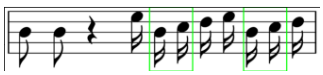
Para acabar de realizar un resultado más óptimo se realiza "Non Maximum Supression" [11] para no generar tantas ventanas finales y dejar un resultado final de la manera más clara y limpia posible.

5.1.5. Resultados individuales

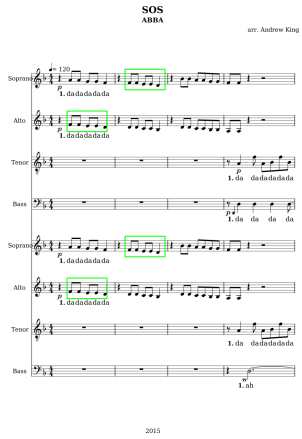
En esta sección se observan los resultados obtenidos por el modelo con los diferentes descriptores creados. Cabe mencionar que debido a que el descriptor BSM ha sido codificado desde cero no es tan eficiente como el HOG de la librería "Scikit-image" [12] pero no necesita tantos datos de entrenamiento como HOG, o incluso ninguno para extraer buenos resultados (es decir, no hace falta la intervención del SVM). Dado que existen estas dos opciones, se puede elegir entre más precisión o más velocidad.

En la Figura 7 (a) el modelo es capaz de reconocer el tono del pasaje musical a nivel de línea de compás. Una primera aproximación para obtener resultados visuales y validar que los métodos aplicados funcionan. Por otra parte en la Figura 7 (b) el modelo es capaz de detectar a nivel de partitura completa aquellos pasajes similares dada una consulta. De esta manera, el objetivo de realizarlo a nivel de partitura estaba realizado. Por último, en la Figura 8 se obtienen resultados de similitud visual en la misma partitura, pero diferente imagen de diferente tamaño, encontrando así los pasajes parecidos a la consulta realizada (cuadrado azul).

Otro aspecto a comentar en la ejecución del modelo con el descriptor HOG es la cantidad de muestras positivas y

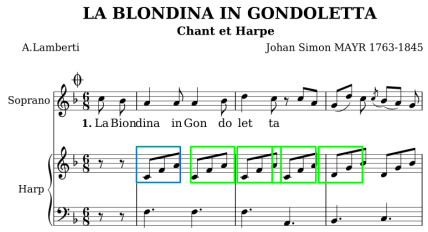


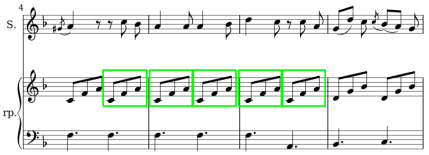
(a) HOG y SVM a nivel de compás



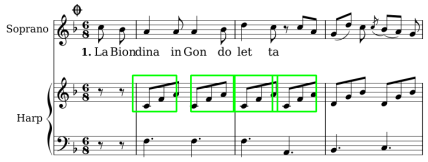
(b) HOG y SVM a nivel de partitura

Fig. 7: Resultados con descriptor HOG





(a) HOG y SVM con partitura original

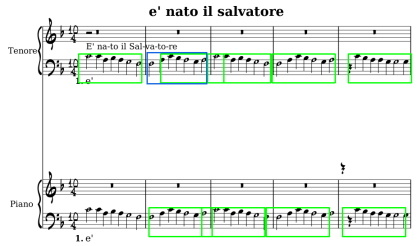


(b) HOG y SVM con misma partitura diferente imagen

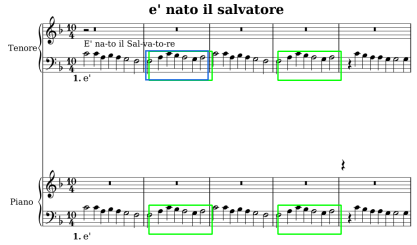
Fig. 8: Resultados con descriptor HOG y SVM en diferentes partituras

muestras negativas utilizadas. Esto puede alterar el resultado final obteniendo un mejor rendimiento cuantas más muestras de ambas clases se generen, pero resultando un tiempo de ejecución más elevado. En la Figura 9 se puede ver la diferencia en la detección del pasaje musical con la misma partitura donde en la Figura 9 (a) se han utilizado

1.500 muestras positivas y 3.000 negativas y ha resultado en un tiempo de ejecución de 30 segundos, mientras que en la Figura 9 (b) se han utilizado 7.000 muestras positivas y 15.000 muestras negativas con un tiempo de ejecución de 72 segundos.



(a) 1.500 muestras positivas y 3.000 muestras negativas



(b) 7.000 muestras positivas y 15.000 muestras negativas

Fig. 9: Diferencia entre la selección de cantidad de muestras

En cuanto a los resultados visuales mediante la utilización del descriptor BSM (Fig. 10)se puede observar que no es necesaria la intervención de la parte del SVM, dado que es tan potente que mediante la comparación entre vectores de la consulta y los generados, es capaz de extraer aquellos pasajes iguales a la consulta. Esto ocurre porque las partituras que se han utilizado para testearlo son impresas y por lo tanto son perfectas (no tienen imperfecciones como podría ser en partituras manuscritas). El resultado no sería el mismo en el caso de aplicarlo sobre partituras antiguas o escaneadas.

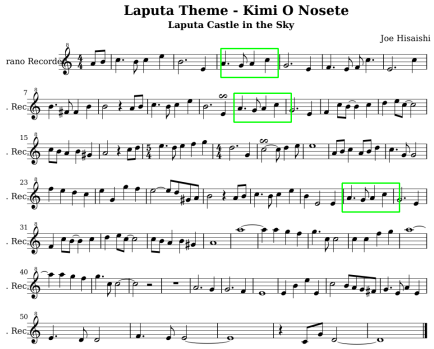


Fig. 10: Detección con descriptor BSM

5.2. Modelo basado en Deep Learning

En esta sección se describen cada uno de los pasos que se han llevado a cabo para poder desarrollar el modelo de Deep Learning con el objetivo de detectar objetos musicales y buscar a través de similitud visual.

5.2.1. YOLOV5

Después de revisar el trabajo [5] donde se hace una comparativa de modelos de Deep Learning para la detección de objetos musicales, se ha decidido utilizar un modelo YOLOV5 para poder implementar esta parte del proyecto. En los resultados en el trabajo comentado anteriormente, no se utiliza en ningún momento el modelo YOLOV5 [13], pero se utiliza RetinaNet (modelo similar). Los resultados no parecen ser muy prometedores para este tipo de modelos debido a que no es capaz de detectar las diferentes clases del resto. Aparte de esto, tampoco es capaz de detectar los objetos más pequeños como los silencios, o aún más importante, las cabezas de las notas. Aún así son los más rápidos entrenando y es un factor a tener en cuenta para poder mejorar y entrenar este tipo de modelo y se adapte a las circunstancias propuestas.

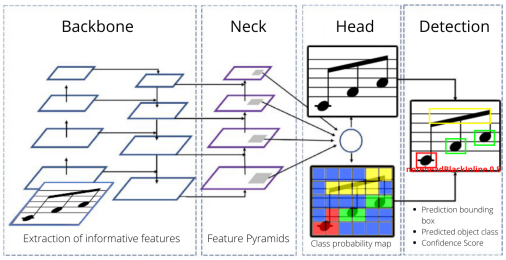


Fig. 11: Estructura YOLOV5

La idea de la utilización de este modelo es ser capaces de detectar las primitivas de objetos musicales (pentagrama, cabeza de nota negra en línea, cabeza de nota negra fuera de línea, cabeza de nota blanca en línea, cabeza de nota blanca fuera de línea, corcheas, silencio de negra,...) mediante las coordenadas de la bounding box que la contienen.

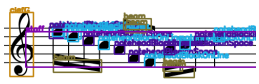


Fig. 12: Detección de primitivas en un pentagrama

5.2.2. Preprocesamiento de los datos

Para poder entrenar el modelo YOLOV5 es necesario preparar un dataset para que aprenda a detectar los objetos musicales que queremos. Para ello, se ha utilizado el dataset de Deepscores [14] el cual tiene una versión reducida con más de 1.700 imágenes y las anotaciones correspondientes a cada una de ellas.

Cuando se trabaja con datasets para la detección de objetos, normalmente, se trabaja con anotaciones en formato PASCAL-VOC donde, en formato XML, se incluye cada una de las coordenadas de la bounding box del objeto y la clase a la que pertenece.

El problema con el dataset escogido, es el tipo de notación que contiene (Oriented Bounding Box Annotations). De formato JSON, contiene todas las notaciones de todas las imágenes en un mismo archivo y con las coordenadas

calculadas de diferente forma. Por eso, ha sido necesario codificar un script que separe cada una de las notaciones en un archivo separado y que lo transforme al tipo de anotación que el modelo YOLO entiende (Fig. 13). Donde encontramos la clase a la que pertenece, coordenada x del centro, coordenada y del centro, ancho y altura de la bounding box normalizadas con el tamaño de la imagen (de izquierda a derecha).

```
122 0.536 0.085 0.833 0.001
35 0.536 0.097 0.833 0.024
24 0.952 0.151 0.002 0.024
24 0.952 0.097 0.002 0.024
```

Fig. 13: Notación formato YOLO

5.2.3. Entrenamiento

En primera instancia se generó un modelo entrenado con las más de 1.700 imágenes del dataset con un "batch size" de 32 y un total de 40 "epochs". El tiempo de entrenamiento fue de 16 horas y los resultados obtenidos dejaban mucho que desear. Debido a que se entrenó con las imágenes de partituras completas, en el momento de seleccionar un recuadro más pequeño de la imagen que actuase como consulta, no era capaz de detectar nada.

Por ello, se decidió realizar un segundo modelo utilizando las líneas de pentagrama de cada una de las partituras como imágenes de entrenamiento. Para llevarlo a cabo, se tuvo que preprocesar de nuevo todas las imágenes para poder dividirlas en líneas y que las anotaciones se adecuasen correctamente. Con esto, se obtuvo un total de más de 14.000 imágenes. De las cuales se utilizaron la mitad para reducir el tiempo de entrenamiento.

Otro aspecto que se ha tenido en cuenta antes de entrenar el nuevo modelo con el nuevo conjunto de imágenes es la reducción de número de clases a detectar. Para poder seleccionar las clases con las que se entrena el modelo, se ha observado la frecuencia de aparición en las 14.000 imágenes y únicamente se escogen aquellas que aparecen 2.000 veces. De esta forma se pasa de tener 132 clases a tener un total de 37, cosa que hizo que el tiempo de entrenamiento fuese inferior.

Con una configuración de "batch size" de 32 y 30 "epochs" y después de más de 24 horas de entrenamiento, se obtuvo un modelo capaz de detectar símbolos más pequeños y que nos sirve para la detección a nivel de línea y nivel de compás.

En la Figura 14 se puede observar en formato de curva de Precision (porcentaje de los que hemos dicho que son de una clase, en realidad lo son)/Recall (porcentaje de cierta clase hemos sido capaces de identificar) el resultado de la detección de objetos musicales del modelo. Precision calculado como $\frac{TP}{TP+FP}$ y Recall calculado como $\frac{TP}{TP+FN}$

5.2.4. Spotting

La siguiente etapa que se ha llevado a cabo es la de adaptar los resultados obtenidos por el modelo (detecta primitivas musicales) para poder encontrar similitud visual entre la un pasaje musical seleccionado y el resto de objetos que hay en la partitura. Para ello, y de forma gradual, igual que

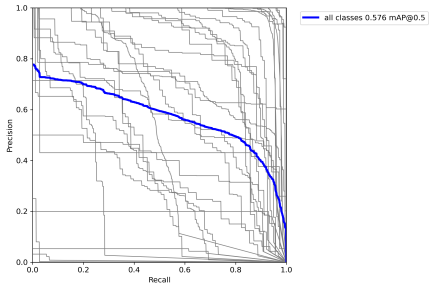


Fig. 14: Precision/Recall Curve

se hizo en el modelo de Machine Learning, primero se ha realizado a nivel de línea de partitura para posteriormente pasar a nivel de partitura completa y finalmente diferentes partituras.

Pese a que el modelo YOLO es capaz de detectar donde se encuentran las líneas de pentagrama, no siempre las encuentra correctamente y difieren en tamaño las líneas de una misma partitura. Para solucionar este inconveniente se ha creado un módulo de detección de líneas horizontales, donde con la imagen binarizada y la detección de las coordenadas de aquellos objetos negros con una longitud más grande que un threshold se detectan como líneas de pentagrama.

Una vez detectadas las líneas de pentagrama se agrupan en bloques de 5 para conocer el área por donde va a operar la ventana deslizante detectando los objetos musicales.

En este punto se han creado dos variaciones para comparar los resultados de detección. La primera de ellas es detectar los objetos musicales a página completa, es decir, toda la partitura. Donde posteriormente la ventana deslizante pasa y guarda los objetos musicales que se encuentren dentro de la ventana. Y la segunda de ellas que consiste en generar una imagen por cada una de las líneas (dejando margen por arriba y por abajo para asegurar que se tienen en cuenta todos los objetos) y detectar los objetos musicales a nivel de línea. Luego, la ventana deslizante pasa por cada una de las líneas igual que en la aproximación anterior.

Los resultados finales son bastante similares, pero dado que el modelo se ha entrenado a nivel de partitura, es más propenso a detectar los objetos musicales más pequeños cuando se hace a nivel de línea (segunda aproximación) que no cuando se hace a página completa (primera aproximación). De este modo, todos los resultados comentados en este artículo se han realizado siguiendo la segunda aproximación.

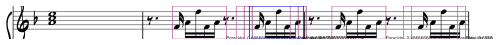
Por último, para poder comparar la consulta generada con las ventanas que se han seleccionado, no solo se tiene que tener en cuenta la clase a la que pertenece la primitiva musical. También se tiene que tener en cuenta el tono de la nota. Para ello, sabiendo la posición de la línea de pentagrama, se calcula la distancia entre el centro de la bounding box del objeto musical y el centro del pentagrama al que pertenece (solo en el eje Y, que es el que importa para detectar el tono).

Con todas estas premisas juntas, para poder comparar la consulta y la ventana generada, se comparan de manera que en primera instancia contengan los mismos objetos musicales en el mismo orden (de izquierda a derecha) y que dichos objetos musicales estén a la misma altura (mismo tono). De

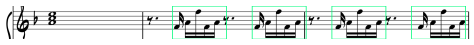
esta manera, el modelo es capaz de sacar como resultado el porcentaje según la cantidad de objetos musicales que sean iguales al pasaje musical seleccionado.

5.2.5. Resultados individuales

En esta sección se exponen los resultados individuales de forma visual desde el ejemplo más fácil (nivel de línea) hasta el más complejo (nivel de partitura completa). Se pueden diferenciar ambas aproximaciones que se ha hecho para el tipo de partitura completa donde la primera detecta zonas erróneas de más (Falsos positivos) y la segunda deja sin detectar zonas que si que son (Falsos negativos).



(a) Porcentaje de parecido con la consulta



(b) Aplicación de NMS

Fig. 15: Resultados a nivel de línea

En la Figura 15 (a) se observa la detección del pasaje musical, marcado mediante un cuadro azul, donde se han generado diferentes cuadros en color rosa en las zonas donde más se parecen al pasaje. A estos cuadros se le asignan un porcentaje según el parecido para que posteriormente en la Figura 15 (b) mediante la aplicación de NMS solo quede como resultado los cuadros verdes finales de la detección de este modelo.

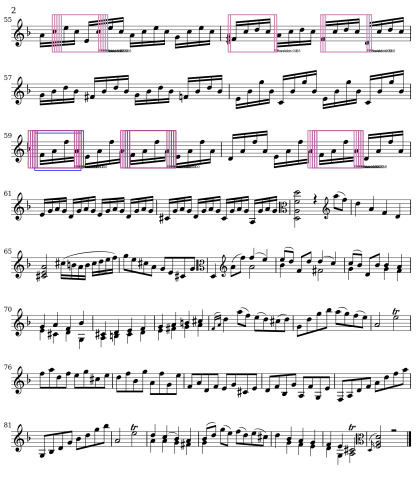
En la Figura 16 se observan los resultados generados a nivel de partitura completa mediante la primera aproximación (partitura completa total). En la subfigura (a) se encuentran los resultados generados por el parecido según el porcentaje para después poder aplicar NMS y generar únicamente las ventanas finales. Se puede ver que esta primera aproximación no es capaz de descartar resultados incorrectos y detectar como positivos aquellos que no lo son (False Positive) cosa que hace que la precisión disminuya considerablemente.

Por otra parte, se encuentran los resultados generados por la segunda aproximación (Fig. 22), en la cual las ventanas resultantes son menores y donde la precisión es mayor que la anterior aproximación detectando únicamente 2 cuadros correctos. Pese a ello, al no detectar todos los posibles, hace que el Recall disminuya.

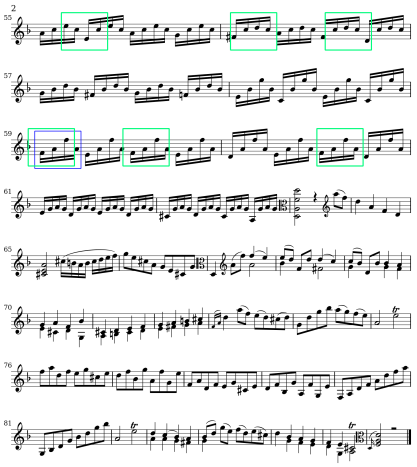
6 ANÁLISIS RESULTADOS

En este apartado se comentan los resultados cuantitativos obtenidos de ejecutar los diferentes modelos con diferentes pasajes musicales. Debido a que el usuario final puede escoger en las diferentes partituras que se le ofrecen el pasaje musical que desee, para poder evaluar de forma correcta los modelos, se han extraído un total de 25 pasajes musicales de diferentes complejidad que afectan y aparecen en más de 100 imágenes distintas de partituras.

La división en diferentes pasajes musicales ha estado mediante la dificultad que tienen. Subdividiendo así en tres grupos diferentes: sencillo, medio y complejo. En ellos aparecen desde un par de notas juntas hasta uno o dos compases enteros. De todas maneras, pese a que haya un grupo



(a) Porcentaje de parecido con la consulta



(b) Aplicación de NMS

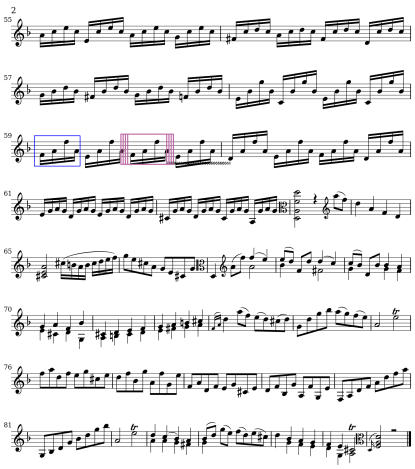
Fig. 16: Partitura completa primera aproximación

denominado como sencillo, se ha intentado escoger pasajes que contengan más de 3 notas seguidas para poder evaluar también el rendimiento de los modelos.

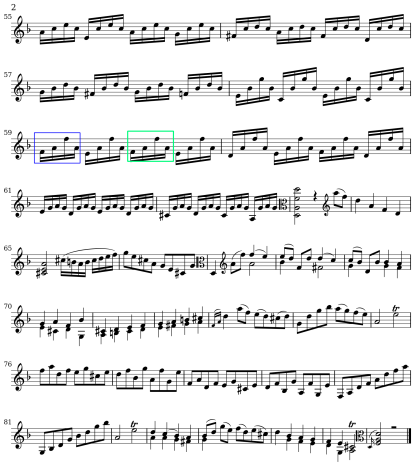
Cada uno de los pasajes extraídos forman parte de una partitura completa que se ha probado en diferentes imágenes de la misma partitura pero con cambios en el "tipo de letra" utilizado (Fig. **??**). De esta manera, los resultados pueden ir variando y ser distintos en todas las partituras. Posteriormente, para evaluar la eficacia del modelo conforme un pasaje musical dado, se realiza la media del "Precision" y del "Recall" (calculados de la misma manera que se explican en la sección 5.2.3).

En la Figura 18 se observan los resultados de los porcentajes obtenidos de Precision y Recall de la ejecución de los diferentes modelos. En ellos se ve la separación entre los diferentes grupos de pasajes musicales (sencillo, medio y complejo).

Con el modelo de HOG+SVM se generan un Precision y Recall por encima del 0.7 en todos los grupos, quedando un mejor equilibrio en el grupo de dificultad media. Esto es debido a que los pasajes musicales catalogados como sencillos contienen dos o tres objetos musicales, pero que aparecen más de tres veces en la partitura y hace que el modelo no sea capaz de detectar con totalidad las ventanas correspondientes. Por otra parte, el Recall en el grupo de dificultad compleja es más alto que la Precision debido a que los

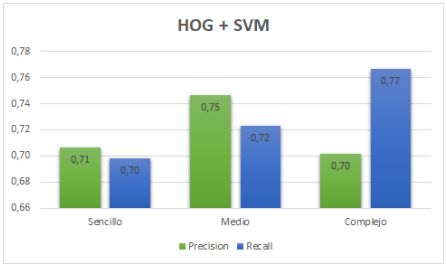


(a) Porcentaje de parecido con la consulta

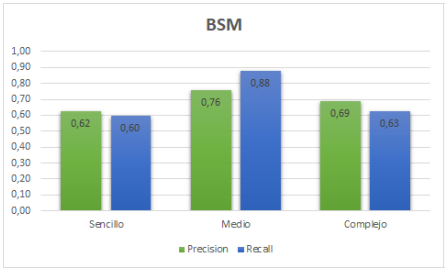


(b) Aplicación de NMS

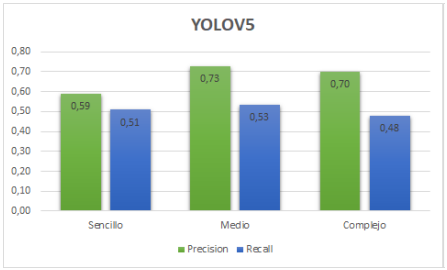
Fig. 17: Partitura completa segunda aproximación



(a) Porcentaje de Precision y Recall con el modelo HOG+SVM



(b) Porcentaje de Precision y Recall con el modelo BSM



(c) Porcentaje de Precision y Recall con el modelo YOLOV5

Fig. 18: Gráficos de los porcentajes de Precision y Recall de los diferentes modelos

pasajes musicales seleccionados, contienen uno o dos compases que aparecen pocas veces en la partitura y el modelo únicamente es capaz de detectar uno. Aún así los resultados obtenidos son favorables dado que como se ha comentado antes, los pasajes musicales seleccionados son ya de alta complejidad pese a ser catalogados como sencillos.

Con la utilización del descriptor BSM (sin el uso de SVM), los resultados son equilibrados. Todos los grupos están por encima del 0.6 tanto en Precision y Recall destacando otra vez el grupo de dificultad media debido a lo que se ha comentado anteriormente. Se puede observar que los resultados de este grupo son mejores que en el primer modelo comentado. Aún así, los resultados en líneas generales no son mejores, y como se ha comentado en la sección 5.1.5 no resultarían igual si se probase con partituras más complejas(escaneadas, manuscritas o antiguas).

Cabe mencionar que se ha decidido no utilizar SVM en este modelo ya que los resultados obtenidos ya son lo suficientemente buenos como para tener que ejecutar ese módulo y obtener resultados similares y, por lo tanto, incrementar el tiempo de ejecución.

Por último se encuentran los resultados generados por el modelo YOLOV5, donde nuevamente, el grupo perteneciente a la dificultad media vuelve a tener un Precision y Recall más alto que el resto. Pese a ello, los resultados en líneas generales son peores que el modelo de Machine Learning con los dos descriptores. Esto es debido a que el modelo no se ha entrenado suficiente tiempo. La configuración seleccionada para entrenar el modelo parece no ser suficiente para obtener mejores resultados (o más parecidos al del modelo anterior). Esto se refleja a la hora de detectar los objetos musicales individualmente. El modelo, para diferentes compases donde aparecen los mismo objetos musicales, detecta elementos diferentes sobreponiéndose uno encima de otro, y por lo tanto, el modelo final de Spotting no es capaz de detectar como similar un compás idéntico al del pasaje seleccionado.

Otro aspecto a tener muy en cuenta para poder comparar mínimamente los modelos generados es el tiempo de ejecución de cada uno (Tabla 1). El modelo que utiliza HOG, el tiempo de ejecución se eleva debido a la generación de muestras positivos y negativos. Este tiempo puede variar según la cantidad que se precisen, pero para obtener un buen resultado, las ejecuciones se han realizado con 1.500 muestras positivas y 3.000 negativas. Aún así, es el modelo más rápido de los tres. Por otra parte se encuentra el modelo que utiliza BSM, el cual, debido a que ha sido configurado y creado desde cero sin la utilización de ninguna librería, no está optimizado del todo. Por último, los tiempos generados con el modelo YOLOV5 son ligeramente superiores al del primer modelo debido a que existen pasos previos (como

TABLA 1: TIEMPOS DE EJECUCIÓN

Tiempos de ejecución (en segundos)			
Clase	HOG + SVM	BSM	YOLOV5
Sencillo	36	163	51
Medio	22	125	29
Complejo	33	153	47

la separación de la imagen original en recortes de líneas de pentagrama) que hacen que se eleve.

7 CONCLUSIONES

El objetivo principal de este proyecto sobre la creación de dos modelos diferentes mediante técnicas distintas capaces de buscar mediante similitud visual un pasaje musical en las partituras se ha conseguido satisfactoriamente. Se ha creado dos aproximaciones para el modelo de Machine Learning con dos descriptores distintos y se han podido evaluar mediante consultas de complejidad diferente. Por otra parte, se ha creado y entrenado un modelo de Deep Learning capaz de detectar primitivas musicales individualmente para posteriormente ejecutar el módulo de Spotting y buscar similitudes. Igualmente se ha podido probar su eficacia mediante la generación de consultas individuales.

Los resultados en líneas generales son buenos, pero siempre hay margen para la mejora. Aún así, como baseline de Music Object Spotting, algo que se había hecho pero con texto manuscrito, es un buen comienzo e invita a poder seguir con su desarrollo y mejora para poder obtener resultados más prometedores y aplicativos para desarrollos para el uso cotidiano y atreverse a realizarlo con partituras manuscritas para conservar su historia.

Para poder probar que los modelos funcionan correctamente se ha realizado una interfaz gráfica sencilla de modo que el usuario pueda recortar de la partitura el pasaje musical que desee. Como trabajo futuro para este proyecto, se podría realizar a nivel aplicativo real, donde desde una aplicación web "conectada" a una colección de partituras o biblioteca a la que se pueda tener acceso, el sistema sea capaz de buscar lo que el usuario desee entre todas las partituras.

Este proyecto ha servido para poder adentrarme más en el mundo musical y ver la complejidad que tiene poder realizar detecciones y procesamiento que con texto, a simple vista, parece más sencillo. He descubierto la técnica de Spotting que antes desconocía y he ampliado los conocimientos que previamente tenía sobre Machine Learning y Deep Learning, y todo lo que ello conlleva.

8 AGRADECIMIENTOS

Quisiera agradecer en primera instancia a mi tutora de este proyecto, Alicia Fornés, por saber guiarme, ayudarme y tener tiempo en cualquier momento para resolverme todas las dudas que he podido tener y por enseñarme todo lo que he aprendido con este trabajo. También quería agradecer a todas esas personas que están en mi día a día diario y han sabido aguantarme en estos meses tan complicados.

REFERENCIAS

[1] Apparao, S. & Hangarge, Ma., Optical Recognition of Old Handwritten Music Symbols Based on Texture Descriptors, International Journal of Advanced Research in Engineering and Technology.

[2] Oh, Jiyong & Son, Sung & Lee, Sangkuk & Kwon, Ji-Won & Kwak, Nojun. (2017). Online recognition of handwritten music symbols. International Journal on Document Analysis and Recognition (IJDAR).

[3] Rebelo, A. & Capela, G. & Cardoso, J. (2010). Optical recognition of music symbols - A comparative study. IJDAR.

[4] Pacha, A. & Calvo-Zaragoza, J. (2018). Optical Music Recognition in Mensural Notation with Region-Based Convolutional Neural Networks.

[5] Pacha, A. & Hajič, jr, Jan & Calvo-Zaragoza, J. (2018). A Baseline for General Music Object Detection with Deep Learning. Applied Sciences.

[6] Riba, P. & Almazan, J. & Fornés, A. & Fernandez, D. & Valveny, E. & Lladós, J. (2014). e-Crowds: a mobile platform for browsing and searching in historical demography-related manuscripts.

[7] Almazan, J. & Gordo, A. & Fornés, A. & Valveny, E.. (2014). Segmentation-free word spotting with exemplar SVMs. Pattern Recognition.

[8] Dalal, Na. & Triggs, B. (2005). Histograms of oriented gradients for human detection. IEEE Computer Vision and Pattern Recognition. pp. 886-893.

[9] Escalera, S. & Fornés, A. & Pujol, O. & Radeva, P. & Sánchez, G. & Lladós, J. (2009). Blurred Shape Model for binary and grey-level symbol recognition. Pattern Recognit. Lett., 30, 1424-1433.

[10] Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273–297.

[11] Neubeck, A. & Van Gool, L. (2006). Efficient Non-Maximum Suppression. Proceedings of the 18th International Conference on Pattern Recognition - Volume 03 (p./pp. 850–855)

[12] Van der Walt, S. & Schonberger, Johannes L & Nunez-Iglesias, J.& Boulogne & Franccois & Warner & J. D. & Yager& N.& Yu. (2014). scikit-image: image processing in Python. PeerJ, 2, e453.

[13] Jocher, G. & Stoken, A. & Borovec, J. (2020). ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements.

[14] Tuggener, L. & Satyawan, Y. & Pacha, A. & Schmidhuber, J. & Stadelmann, T. (2021). The DeepScoresV2 Dataset and Benchmark for Music Object Detection.

APÉNDICE

A.1. Planificación seguida

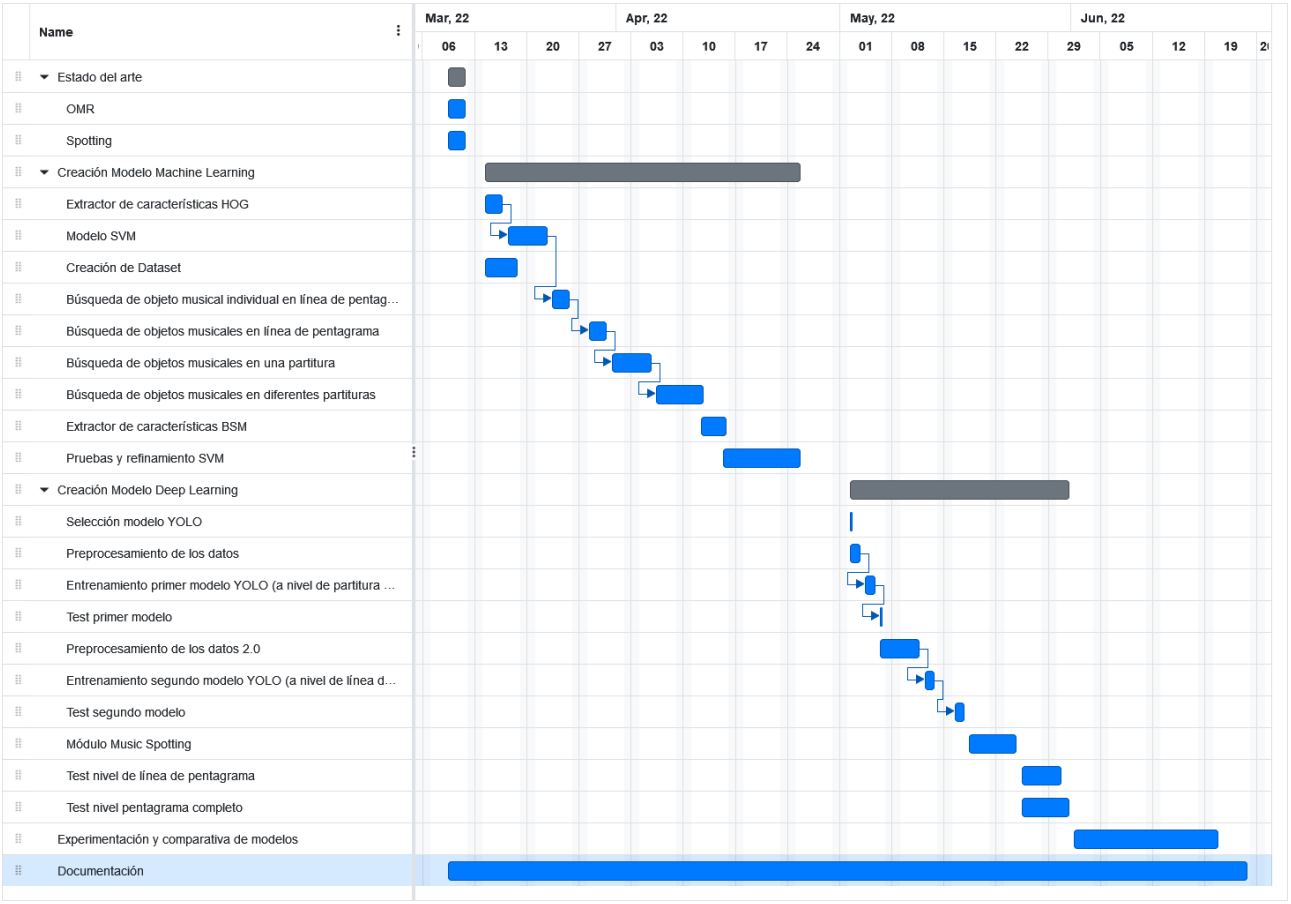
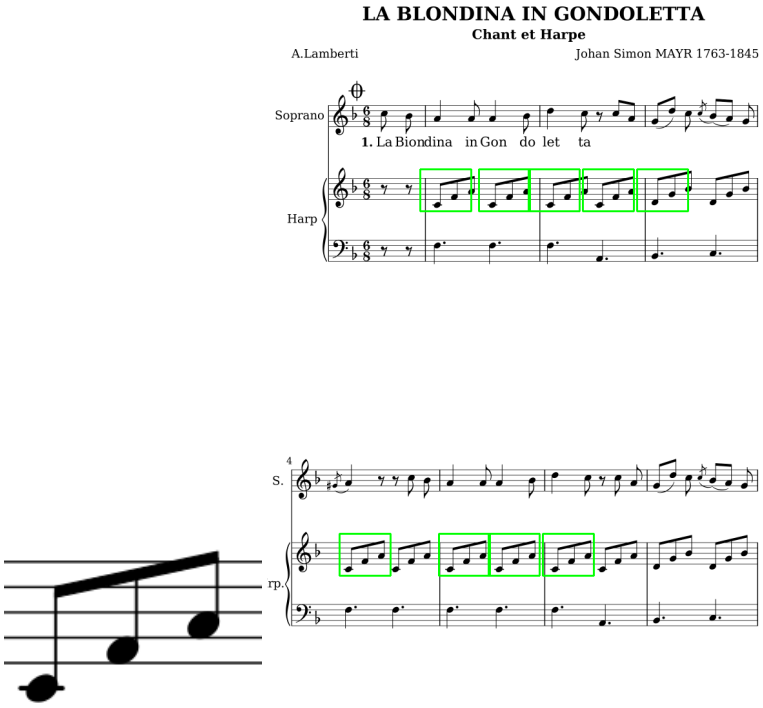


Fig. 19: Diagrama de Gantt

A.2. Ejemplo pasaje musical sencillo



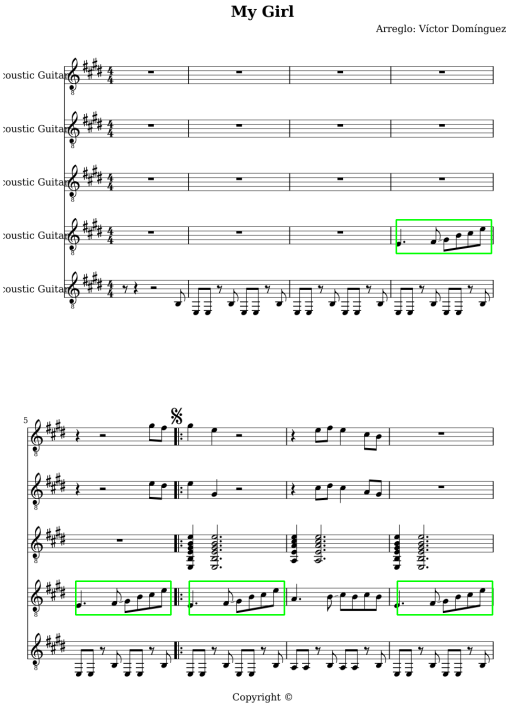
(a) Pasaje musical sencillo (b) Resultado HOG + SVM

(c) Resultado BSM (d) Resultado YOLO

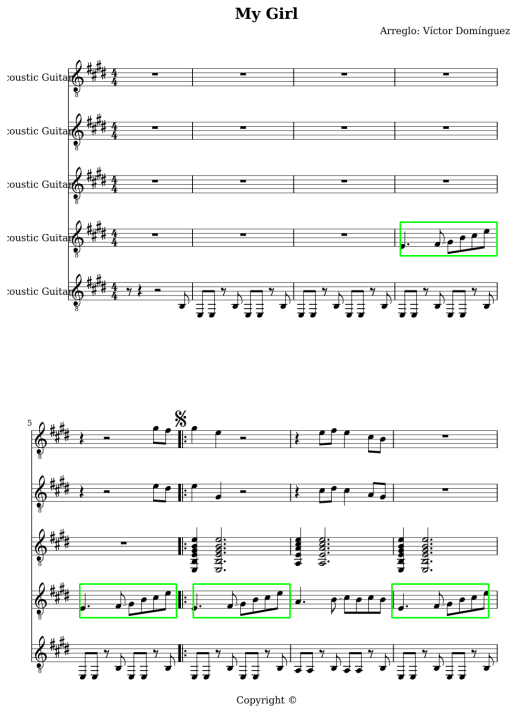
Fig. 20: Resultados pasaje musical sencillo

A.3. Ejemplo pasaje musical medio

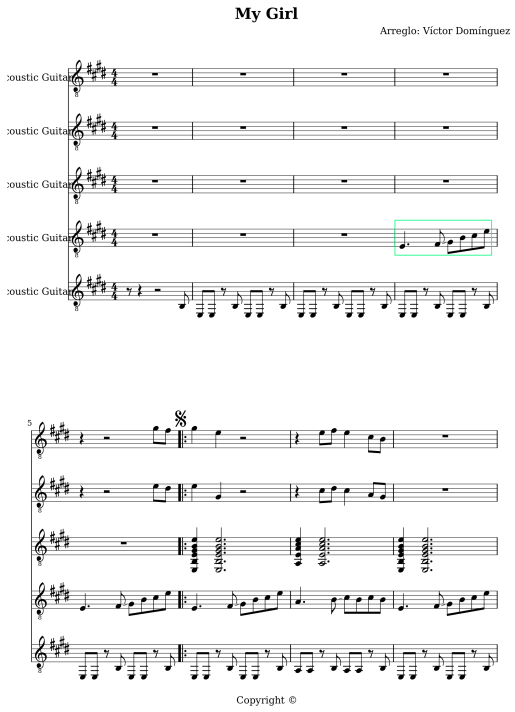
(a) Pasaje musical medio



(b) Resultado HOG + SVM



(c) Resultado BSM

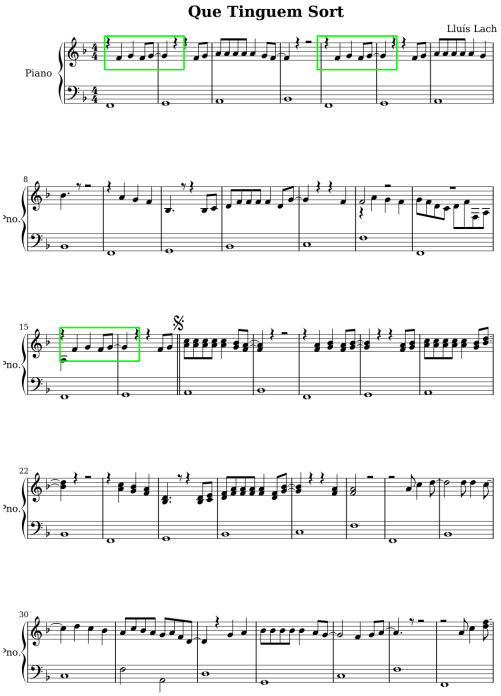


(d) Resultado YOLO

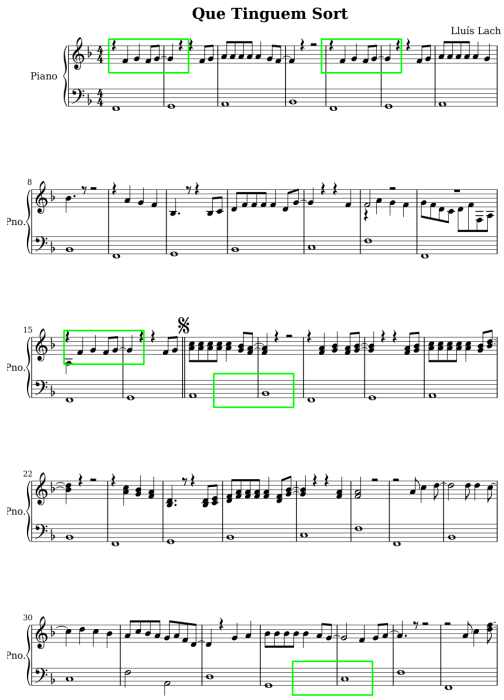
Fig. 21: Resultados pasaje musical medio

A.4. Ejemplo pasaje musical complejo

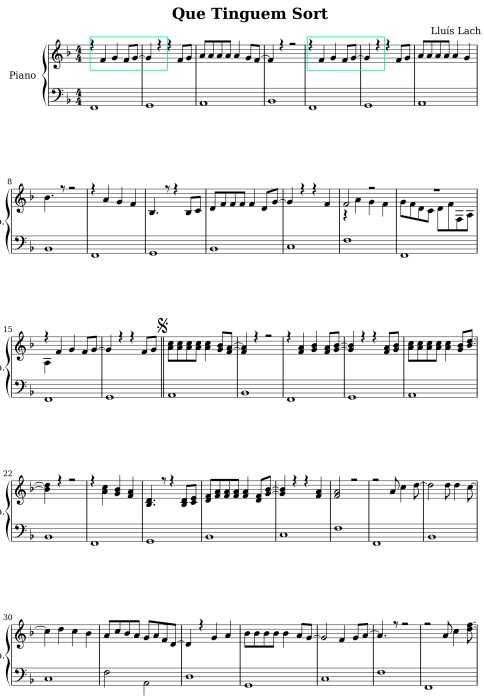
(a) Pasaje musical complejo



(b) Resultado HOG + SVM



(c) Resultado BSM



(d) Resultado YOLO

Fig. 22: Resultados pasaje musical complejo