Zubía 40 177-196 Logroño 2022

DIGITALIZACIÓN DEL LIBRO *"LA MÚSICA EN LA CATEDRAL DE SANTO DOMINGO DE LA CALZADA"**

GONZALO SANTAMARÍA[1], TERESA CASCUDO[2,3], CÉSAR DOMÍNGUEZ[1], JÓNATHAN HERAS[1], ELOY MATA[1], VICO PASCUAL[1], MARÍA VILLOTA[1]

RESUMEN

El archivo musical de la Catedral de Santo Domingo, de José López Calo, contiene una catalogación de obras musicales de carácter religioso como misas, villancicos o salmos de diferentes autores. Dicho libro no contaba con una versión digital, por lo que la única forma de acceder a su contenido era disponiendo de uno de los 500 ejemplares que fueron impresos. El objetivo de este trabajo ha consistido en digitalizar el libro, y estructurarlo de una forma sencilla para poder realizar búsquedas de texto, o consultar las obras de diferentes autores y/o diferentes géneros musicales mediante una página web, además de poder reproducir la melodía de las mismas. En este artículo se explica el proceso seguido para lograr dicho objetivo. El resultado final se puede encontrar accediendo al siguiente enlace: https://domingo.unirioja.es/.

Palabras clave: Archivo musicales riojanos, Inteligencia Artificial, Reconocimiento Óptico de Caracteres, Reconocimiento Óptico de notación musical.

El archivo musical de la Catedral de Santo Domingo, by José López Calo, contains a catalogue of sacred musical compositions such as masses, carols, or psalms composed by different authors. This book did not have a digital version until now; so, its contents could only be accessed through one of the 500 printed copies. The aim of this work is to produce a digital and structured version of the book, in order to conduct text searches, or to consult the works of different authors and/or different musical genres through a web page, as well as to be able to reproduce the melody of these works. This article explains the process followed to reach such an objective. The final result can be found in the following link: https://domingo.unirioja.es/.

Keywords: *Rioja musical archive, Artificial Intelligence, Optical Character Recognition, Optical Music Recognition.*

---

* Registrado el 21 de diciembre de 2021. Aprobado el 15 de septiembre de 2022.

[1] Departamento de Matemáticas y Computación, Universidad de La Rioja {gonzalo.santamaria, cesar.dominguez, jonathan.heras, eloy.mata, vico.pascual, maria.villota}@unirioja.es

[2] Departamento de Ciencias Humanas, Universidad de La Rioja

[3]Área Patrimonio Regional, Instituto de Estudios Riojanos. tcascudo@larioja.org

1. INTRODUCCIÓN

El archivo musical de la Catedral de Santo Domingo es un libro impreso en 1988 que contiene la catalogación de obras musicales de carácter religioso como misas, villancicos o salmos de diferentes autores. Este libro no contaba con una versión digital, por lo que la única forma de acceder a su contenido era disponiendo de uno de los 500 ejemplares que fueron impresos. Este libro fue el primer volumen que se publicó dentro del proyecto: "La Música en La Rioja", cuyo objetivo era catalogar fondos musicales conservados en archivos riojanos. La labor de catalogación del archivo de Santo Domingo fue realizada por el musicólogo José López Calo, a partir de un conjunto de partituras históricas en su mayoría manuscritas.

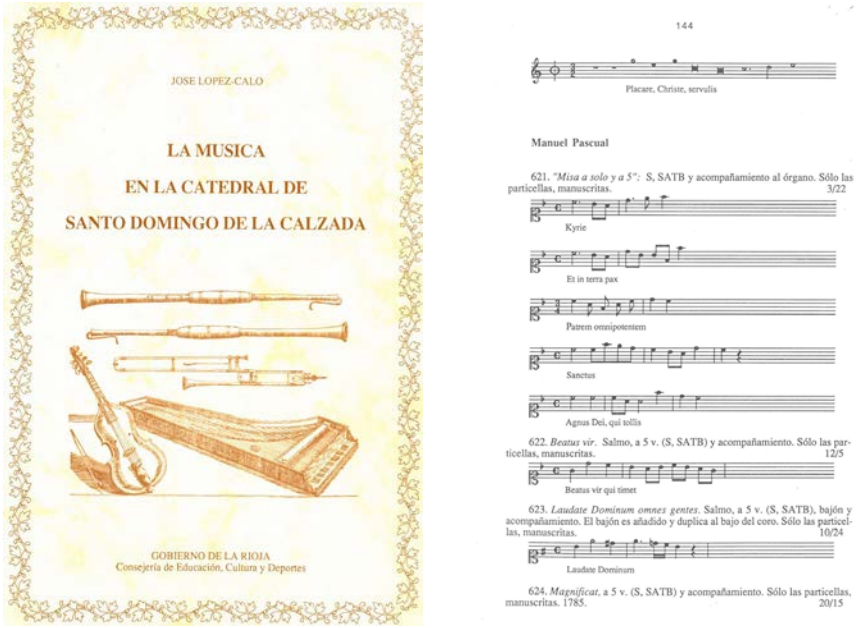

Figura 1. Portada a la izquierda y página 144 del libro a la derecha.

En la figura 1 podemos ver la portada y la página 144 del archivo musical de la Catedral de Santo Domingo de la Calzada. Como puede verse en la página del libro mostrada en dicha figura 1, las páginas contienen diferente información, como por ejemplo *bloques de texto* y *pentagramas* (o íncipits), que se pueden combinar para formar fichas, aquí referidas en ocasiones como obras, que son la estructura más importante dentro del libro. En concreto, en dicha página 144, podemos visualizar un pentagrama de la ficha 620, las fichas 621, 622, 623 completas y el principio de la ficha 624, estas cuatro últimas relativas al autor *Manuel Pascual*. También podemos ver, por ejemplo, que la ficha 621 es una *Misa*.

El objetivo de este trabajo consiste en tener una versión del libro completamente digitalizada y estructurada de una forma sencilla para poder

realizar búsquedas de texto, o consultar las obras de diferentes autores y/o diferentes géneros musicales mediante una página web, además de poder reproducir la melodía de las mismas. Se pretende detectar y estructurar la información anterior, teniendo en cuenta características como el número de obra, el tipo de obra y autor, la partitura, etc. Esto presenta algunas dificultades, como por ejemplo que no todas las obras empiezan o terminan en la misma página, o que el texto contiene saltos de línea, y deberemos solventar todos estos problemas a la hora de cumplir nuestro objetivo.

La investigación se ha realizado en la Universidad de La Rioja y las técnicas utilizadas en el mismo forman parte de los métodos de la inteligencia artificial y la visión por computador. Además, es importante resaltar que dicha investigación precisó de la colaboración de dos disciplinas aparentemente distantes: las humanidades y la ingeniería informática, ya que era necesario un conocimiento de música bastante amplio para llevar a cabo una correcta digitalización del libro.

A su vez, la mayor parte del trabajo ha girado en torno a dos campos dentro de la visión por computador y la inteligencia artificial. El primero de ellos es el Reconocimiento Óptico de Caracteres, *OCR* a partir de ahora (del inglés, *Optical Character Recognition*), el cual se utiliza para extraer el texto contenido en imágenes de documentos escaneados [Adrian Rosebrock y Haase, 2020a, Adrian Rosebrock y Haase, 2020b]. El OCR es una tecnología muy consolidada que se lleva utilizando muchos años, por lo que existe una gran variedad de librerías disponibles que nos realizan esta función correctamente. Las mayores dificultades las encontramos en el segundo campo, cuando tratamos extraer la música contenida en las obras del libro. Para ello se puede utilizar una tecnología que recibe un nombre análogo al anterior: Reconocimiento Óptico de Música, *OMR* a partir de ahora (del inglés, *Optical Music Recognition*) [Calvo-Zaragoza et al., 2021]. El principal problema es que el *OMR* no tiene unas bases tan asentadas como el *OCR* y, a día de hoy, permanecen abiertos muchos problemas para los que no hay una solución que prevalezca sobre las demás [Rebelo et al., 2012, Huang et al., 2019].

El resultado final de este proyecto es una aplicación web donde se pueden consultar todas las fichas del libro, organizadas por autor y género musical. Este portal web está accesible desde el siguiente enlace: https://domingo.unirioja.es/. La aplicación web también permite realizar búsquedas de texto, almacenándolo con un programa de *OCR*, o reproducir el sonido de las obras utilizando un modelo de *OMR* propio. Por último, todo el desarrollo técnico se puede encontrar en el siguiente repositorio de GitHub: https://github.com/joheras/MusicaCatedralStoDomingoIER.

2. MATERIALES Y MÉTODOS

El primer paso de este proyecto fue el escaneado del libro. Para ello, se utilizó una impresora *Bizhub c452* situada en la Oficina Técnica 2 del

Zubía | *núm. 40 (2022), pp. 177-196*
ISSN 0213-4306

Departamento de Matemáticas y Computación de la Universidad de La Rioja. Al final del proceso, obtuvimos 377 imágenes (una por cada página) de dimensiones 1200 píxeles de ancho por 1825 píxeles de alto. En la figura 1 se puede ver una de las paginas escaneadas.

Una vez realizado el escaneado, fue necesario detectar los distintos bloques que componen cada página del libro (es decir, bloques de texto, pentagramas y fichas) y extraer la información que contienen (bien texto, bien símbolos de notación musical) y almacenar todo ello de forma conveniente. Para dichos bloques, se utilizaron diferentes técnicas de procesamiento de imagen e inteligencia artificial, con el fin de extraer la información que contenían. El entorno de trabajo fue *Google Colab* [Bisong, 2019] y el lenguaje de programación *Python* [Van Rossum y Drake, 2009].

Para detectar los bloques, se utilizó la librería *OpenCV* [Bradski, 2008, Bradski, 2000]. Para extraer el texto, se usó la librería *pytesseract* [Adrian Rosebrock y Haase, 2020a, Adrian Rosebrock y Haase, 2020b]. Por último, para extraer la música, como ya hemos comentado, se construyó un modelo de *OMR*, ayudándonos de las librerías *OpenCV*, *IceVision* [Vázquez, 2020a, Vázquez, 2020b], *Darknet* [Bochkovskiy, 2020], y *FastAI* [Howard et al., 2018]. En la figura 2 se puede ver un resumen de las tareas realizadas. Aunque parecen independientes, veremos que en muchas ocasiones tuvimos que respaldarnos en tareas anteriores para resolver una nueva.

Tarea 1 → Detección de pentagramas

Tarea 2 → Detección de bloques de texto

Tarea 3 → Identificación de obras

Tarea 4 → Aplicación de OCR a los bloques de texto

Tarea 5 → Aplicación de OMR a los pentagramas

Tarea 6 → Almacenamiento de la información

Figura 2. Procesos para detectar y extraer la información del libro.

2.1. Detección de bloques

El primer paso consistió en la detección y diferenciación de bloques. La figura 3 es un buen ejemplo para confirmar los tres tipos de bloques hay. Observamos que una ficha es una combinación de un bloque de texto y uno o varios pentagramas.

Las herramientas que hemos utilizado para detectar los diferentes bloques son *OpenCV* y *pytesseract*. Para el texto y los pentagramas hemos seguido un proceso de umbralización (transformar los píxeles a 0 o 1 bajo cierta regla establecida) utilizando el método *Otsu* [Bangare et al., 2015, Yousefi, 2015], operaciones morfológicas (deformar la imagen binaria) y detección de contornos. Para las obras nos hemos ayudado, además, del texto de las páginas del libro, pues todas las obras comienzan con un nú-

Zubía | núm. 40 (2022), pp. 177-196
ISSN 0213-4306

mero (por ejemplo, en la figura 3 se muestra la obra 822) y terminan con la siguiente obra o con el inicio de una nueva sección.

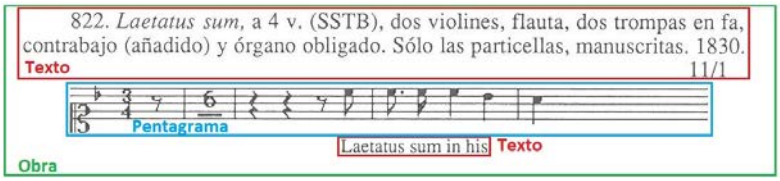

Figura 3. Tipos de bloques dentro de la obra.

Comenzamos detallando el proceso para detectar los pentagramas en una página, dicho proceso se ilustra en la figura 4. A partir de la imagen de una página, ver imagen de la izquierda en la figura 4, se aplica una operación morfológica conocida como dilatación vertical, produciendo una imagen donde cada uno de los pentagramas forma un bloque, ver imagen central de la figura 4. A partir de dicha imagen dilatada se pueden detectar los pentagramas tomando aquellos bloques de la imagen cuya área supera un umbral dado, ver imagen de la derecha en la figura 4.

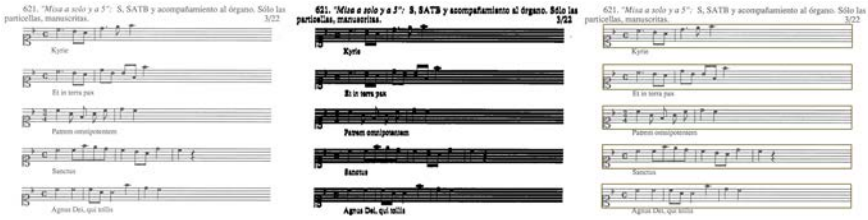

Figura 4. Detección de pentagramas. Izquierda: imagen original; centro: imagen después de la dilatación vertical; derecha: pentagramas detectados a partir de seleccionar los bloques más grandes de la imagen central.

Una vez se han detectado los pentagramas pasamos a detectar los bloques de texto dentro de la imagen, un ejemplo del proceso se puede ver en la figura 5. La idea aquí es, en primer lugar, eliminar los pentagramas que detectamos anteriormente, para después aplicar una dilatación horizontal en la imagen binaria y que se unan las palabras formando bloques. Finalmente, se extrae el mínimo rectángulo que abarque cada uno de los bloques para obtener cada uno de los bloques de texto de la página.

Zubía | *núm. 40 (2022), pp. 177-196*
ISSN 0213-4306

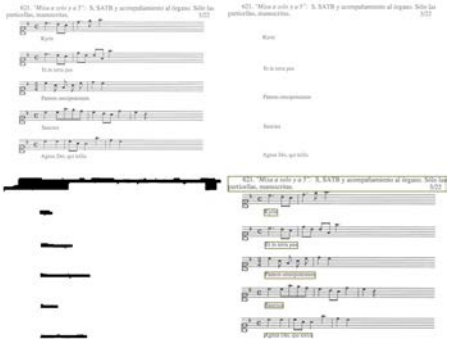

Figura 5. Detección de texto. Arriba-izquierda: imagen original; arriba-derecha: imagen sin pentagramas; abajo-izquierda: imagen sin pentagramas tras aplicar una dilatación horizontal; abajo-derecha: bloques de texto detectados.

Tras detectar los pentagramas y los bloques de texto, se realiza al paso de identificación de las obras, ver figura 6. Vemos en la imagen de la izquierda de la figura 6 que hay algunas palabras marcadas en verde. Estas palabras pueden ser el número de obra, nombres de autores, o bien tipos de obra; y son detectadas con *OCR*. Dichas palabras son muy importantes ya que nos van a marcar el principio y el final de cada obra. Algunas partes de las obras pertenecen a páginas anteriores o posteriores y, en esas situaciones, debemos ser capaces de enlazar bloques de una página con bloques de la siguiente hasta completar la obra.

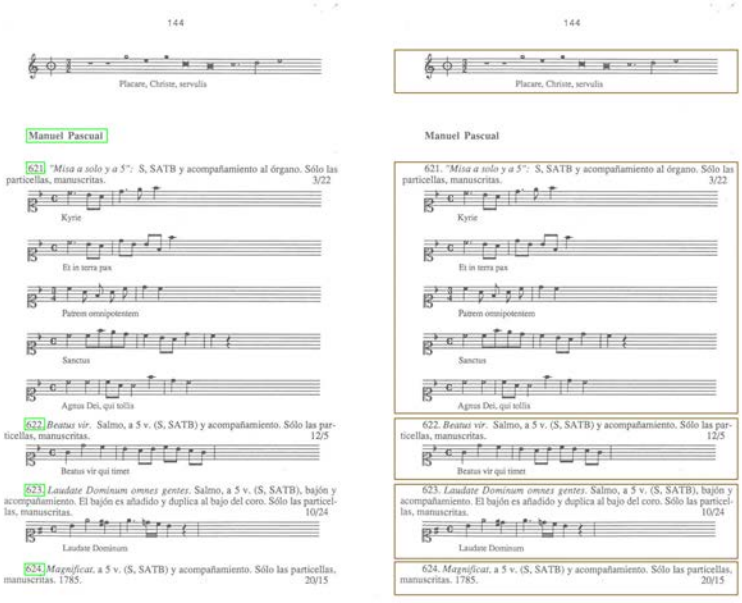

Figura 6. Detección de obras.

2.2. OCR

Para extraer el texto de los bloques etiquetados como texto en el paso anterior, haremos uso de la librería *pytesseract*. Dicha librería ofrece dos funcionalidades diferentes, tal y como se muestra en la figura 7. La primera, dada la imagen de un texto devuelve un diccionario clave-valor, donde las claves son cada una de las palabras del texto, y el valor sus coordenadas dentro de la imagen. La segunda funcionalidad consiste en directamente extraer el texto contenido en una imagen como una cadena de caracteres. En nuestro proyecto, hemos empleado ambas funcionalidades. La primera funcionalidad ha sido útil para proporcionar búsquedas de texto; mientras que la segunda se ha usado para almacenar el texto contenido dentro de las obras.

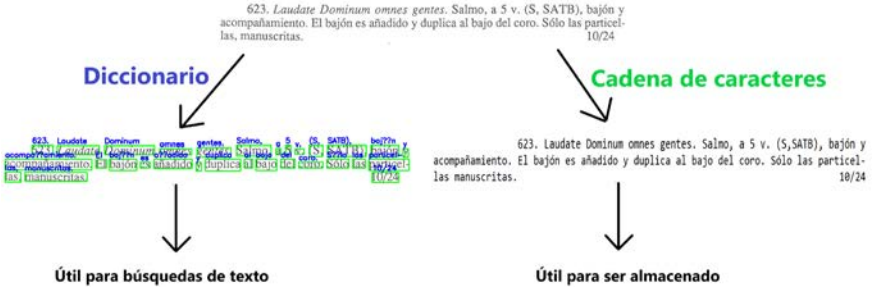

Figura 7. Funcionalidades de la librería pytesseract.

2.3. OMR

A diferencia de la técnica de *OCR*, que está bastante consolidada, las técnicas de *OMR* no están completamente definidas y son un campo de investigación abierto [Rebelo et al., 2012, Huang et al., 2019]. Existen algunos trabajos en el campo del *OMR* que nos han servido de ayuda o inspiración, como ha sido el caso de [Calvo-Zaragoza y Rizo, 2018]. En él se ofrece una solución para transformar imágenes de pentagramas individuales en datos estructurados, en concreto una lista con todos los símbolos musicales ordenados, basándose en un modelo *End-to-End* [Roza, 2019]. De esta manera, se puede simplificar mucho la futura traducción a música o la conversión a otros formatos bien conocidos, como por ejemplo el *Mp3*. El principal problema era que las conversiones no eran lo suficientemente precisas, seguramente porque la estructura general de las imágenes del libro que queríamos digitalizar era diferente a la de las imágenes con las que se construyó dicho conversor.

La poca capacidad de generalización es uno de los principales problemas en todos los modelos de *OMR*, al contrario que el *OCR*, el cual generaliza bastante bien y, en consecuencia, es capaz de extraer con bastante precisión el texto de casi cualquier documento escaneado. Existe un programa de código abierto que es ampliamente utilizado para la tarea de transformar imágenes de partituras en música. Se llama *Audiveris* [Bitteur, 2004], y es

capaz de codificar las imágenes de partituras al formato *MusicXML* [Good, 2001b, Good, 2013, Good, 2001a] (una forma de configurar la música en un archivo *XML* que permite transformar las canciones a cualquier otro tipo de formato con facilidad). Lo bueno de este programa es que es capaz de extraer tanto texto como música, combinando *OCR* y *OMR*, pero, tras probar *Audiveris* en las obras de nuestro libro, vimos que tampoco era lo suficientemente bueno. Es por ello que decidimos utilizar una aproximación propia que se adaptara a las características concretas de nuestro proyecto.

En la figura 8 se puede encontrar un esquema de los pasos que hemos seguido para crear un modelo que, dada una obra del libro, extraiga y codifique el texto y la música que contiene. El resultado final de nuestra aproximación es un modelo *M* que combina *OMR* y *OCR* para transformar una partitura a dato estructurado, basándonos en el diccionario clave-valor que devolvía *pytesseract* (ver figura 7); y un algoritmo *A* que transforma dicho dato estructurado al formato *MusicXML*.

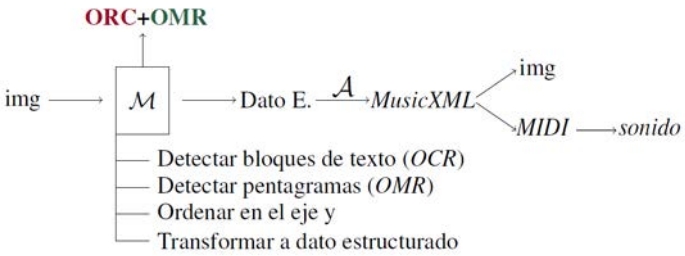

Figura 8. Esquema de nuestro modelo para transformar partituras en música.

Mientras que la codificación del texto en castellano está bien establecida a través del tipo de dato cadena de caracteres, no ocurre lo mismo con la música. Para codificar la música existen muchas alternativas [Hankinson et al., 2011], como por ejemplo: formato imagen (como es el caso de nuestro libro), sonido (*Mp3*, *Mp4*) u otros menos intuitivos como el *MIDI* [Britannica, 2009] o el ya mencionado *MusicXML* [Good, 2001b, Good, 2013, Good, 2001a]. Nosotros hemos optado por almacenar las obras en el formato *MusicXML*, al igual que *Audiveris*, ya que es el más general y admite una conversión al resto de los formatos. Además, es capaz albergar texto dentro de su cuerpo.

Vamos a explicar, por ejemplo, por qué no se puede transformar una melodía en formato *MIDI* a imagen o *MusicXML*. En la figura 9 se encuentra la misma nota musical representada de dos formas diferentes. En *MIDI* las notas se representan por tonos, por lo tanto, al tratarse de la misma nota, son el mismo tono. Es decir, son el mismo archivo *MIDI*. Sin embargo, vemos que un mismo tono puede generar dos o más imágenes diferentes; o dos archivos *MusicXML*, ya que se tienen en cuenta más elementos musicales como la *clave*. Este hecho de tener información menos completa almacenada, hace que no podamos convertir de formato *MIDI* a imagen.

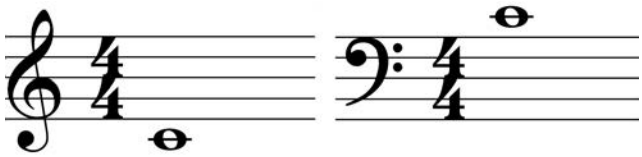

Figura 9. Cuarta octava de la nota Do escrita en Clave de Sol y Clave de Fa en cuarta.

Una vez tenemos clara la forma en la que queremos codificar la música, explicaremos la manera de transformar las imágenes de nuestras obras al formato *MusicXML*. Para ello, hace falta entender cómo se lee un pentagrama musical. En un pentagrama hay elementos como la *clave* o la *armadura* que afectan a toda la secuencia y determinan la altura y la tonalidad de las notas musicales. La figura 9 sirve como ejemplo de cómo una misma nota (C4) se puede notar con dos claves diferentes, así como de una armadura sin alteraciones que se podría corresponder con las tonalidades de Do mayor o La menor. En los tres primeros pentagramas de la figura 11, encontramos otros tipos diferentes de armaduras, con un sostenido, con un bemol y con tres bemoles. Vemos también que hay diferentes figuras, como por ejemplo *redondas*, *negras* o *corcheas*, y otros símbolos musicales como los *silencios* o las *líneas* divisorias que atraviesan verticalmente el pentagrama separando los compases. Todos estos símbolos se leen de izquierda a derecha y el orden en el que aparecen es muy importante. Es por eso por lo que decidimos crear dos modelos de inteligencia artificial, uno para detectar símbolos musicales y otro para clasificar la altura de las notas (las notas detectadas previamente con el anterior modelo). Si una ficha del catálogo tiene más de un pentagrama, se puede realizar este proceso de forma individual a cada uno de ellos y luego juntar las predicciones.

Para entrenar modelos de inteligencia artificial son necesarios los conjuntos de datos (generalmente conocidos, del inglés, como *datasets*). Aunque existen una gran variedad de conjuntos de datos relacionados con el *OMR* [Pacha, 2017], nosotros decidimos crear nuestro propio conjunto de datos utilizando algunas imágenes del libro (este conjunto de datos se encuentra disponible en el portal web del proyecto y puede ser utilizado por otros autores para sus propios proyectos). En concreto creamos dos conjuntos de datos, uno para detección de símbolos musicales dentro de pentagramas y otro a parte para la clasificación de las alturas de las notas; y en ambos casos tuvimos que realizar un proceso de anotación de imágenes.

En el proceso de anotación se siguió un procedimiento semi-automático y en la figura 10 se puede encontrar un esquema donde se explica cómo se creó el conjunto de datos de detección. Anotamos 300 de 2308 pentagramas disponibles en el libro. Para el conjunto de datos de clasificación de notas se siguió un procedimiento análogo.

Zubía | *núm. 40 (2022), pp. 177-196*
ISSN 0213-4306

Paso 1 → Anotar 20 pentagramas manualmente

Paso 2 → Entrenar un modelo de detección

Paso 3 → Elegir una muestra aleatoria de 20 pentagramas no anotados

Paso 4 → Calcular las predicciones

Paso 5 → Corregir los errores para completar la anotación

Paso 6 → Volver al **Paso 2** hasta tener anotados 300 pentagramas

Figura 10. Procesos para creación del conjunto de datos de detección.

En el *paso 2* de la figura 10 vemos que entrenamos un modelo de detección con el fin de que el proceso de anotación fuera más rápido. El proceso de entrenamiento consiste en proporcionar a un algoritmo una serie de ejemplos y el resultado que debe devolver para ellos (en nuestro caso los ejemplos serían los pentagramas, y el resultado a producir la posición de cada una de las figuras del pentagrama) e ir modificando poco a poco el comportamiento del algoritmo hasta que los resultados producidos sean los adecuados. El resultado final de este proceso se conoce como modelo.

En nuestro caso, el algoritmo que utilizamos en el paso 2 fue el *Faster R-CNN* [Ren et al., 2015], de la librería *IceVision* [Vázquez, 2020b,Vázquez, 2020a], y lo elegimos por ser el modelo que menos tiempo tardaba en entrenarse, ya que en este momento solo nos importaba la rapidez con la que anotábamos, aunque esto supusiera que tuviéramos que corregir parte de las anotaciones. Para el etiquetado de la altura de las notas utilizamos el algoritmo *ResNet-18* [Kaiming He, 2015], de la librería *FastAI* [Howard et al., 2018]. En la figura 11 se encuentran algunos pentagramas anotados del conjunto de datos para la detección de símbolos musicales y en la figura 12 las imágenes para la clasificación de la altura de las notas.

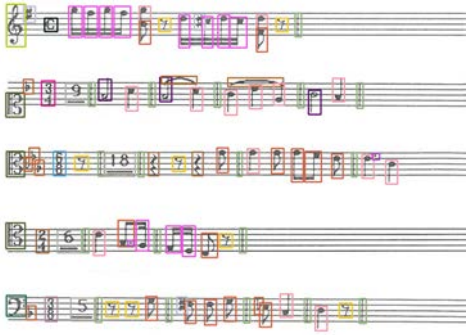

Figura 11. conjunto de datos de detección de símbolos.

Zubía | núm. 40 (2022), pp. 177-196
ISSN 0213-4306

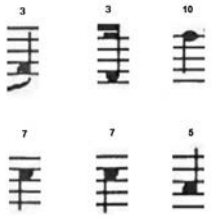

Figura 12. conjunto de datos de clasificación.

En la tabla 1 podemos ver un resumen de los conjuntos de datos que hemos creado. Estos conjuntos de datos están publicados en el repositorio de GitHub del proyecto.

Los modelos de detección que probamos dentro de la librería de *IceVision*, en su versión 0.5.1, son: *EfficientDet* [Tan et al., 2019], el ya mencionado *Faster R-CNN* [Ren et al., 2015] y *RetinaNet* [Lin et al., 2018]. En la librería *Darknet* [Bochkovskiy, 2020], entrenamos *YOLO v4* [Bochkovskiy et al., 2020]. Para la clasificación de la altura de las notas solamente entrenamos el ya mencionado *ResNet-18* [Kaiming He, 2015].

El entorno donde se han entrenado dichos modelos ha sido *Google Colaboratory* [Bisong, 2019] para poder utilizar las GPUs que ofrece de forma gratuita, salvo para entrenar el modelo *Yolo v4* (que necesitaba más recursos), que decidimos utilizar el servidor *Simba* de la Universidad de La Rioja, el cual dispone de cuatro GPUs *Nvidia RTX 2080 Ti*.

Tabla 1. Información de los conjuntos de datos.

Conjunto de datos	Train	Test	Clases
Detección símbolos	225	75	34
Clasificación alturas	2015	504	17

Otra herramienta que hemos utilizado en estos conjuntos de datos, con el fin de crear modelos capaces de generalizar mejor, es el *Data Augmentation* [Perez y Wang, 2017]. Esta técnica consiste en aumentar el número de imágenes de nuestro conjunto de datos por medio de rotaciones, añadiendo ruido, recortando, etc. Algunas de estas transformaciones no tenían sentido en nuestro contexto, por lo que han sido descartadas. Por ejemplo, no tiene sentido rotar un pentagrama 180 grados, ya que, con esa transformación, las notas que antes representaban un "mi" en clave de Sol pasarían a representar un "fa", y eso es un error.

2.4. Almacenamiento de la información

Toda la información requiere de un almacenamiento para su posterior utilización. Es por ello que el texto lo hemos almacenado de dos formas, una pensada para realizar búsquedas dentro del portal web y otra pensada para los archivos *MusicXML*. Evidentemente, en el segundo caso, solo se

almacena el texto que esté relacionado con la obra musical. De esta manera, los archivos *MusicXML* serán muy completos y podremos hacer cosas como reconvertirlo a un *PDF* limpio sin el ruido que se produce en el proceso de escaneado. El texto utilizado para realizar búsquedas se ha almacenado dentro de un archivo *JSON* [Pezoa et al., 2016]. Este archivo almacena un diccionario donde las claves principales son las páginas del libro y, en cada una de ellas, se encuentra el diccionario que devuelve *pytesseract*.

Las obras se han almacenado de varias formas. Una de ellas en el formato imagen *JPG* (comentar que algunas obras empiezan en una página y terminan en la siguiente, por lo que hemos tenido que concatenar ambas imágenes para tener la obra completa); otra en el formato ya mencionado *MusicXML* (las predicciones); y otra derivada de la anterior en el formato de audio *WAV*. También hemos creado dos archivos *JSON* donde podemos encontrar clasificadas las obras por su autor y su tipo de obra, con el fin de realizar consultas en la página web. Por ejemplo, podemos hacer la consulta: "Muéstrame todas las misas del autor Pedro Estorcui". Nuestro programa accederá a la clave "misas" del diccionario de las obras clasificadas por género y a la clave "Pedro Estorcui" del diccionario de las obras clasificadas por autor y mostrará la intersección de los dos resultados. En la figura 13 podemos ver un ejemplo de qué criterios seguimos para clasificarlas.

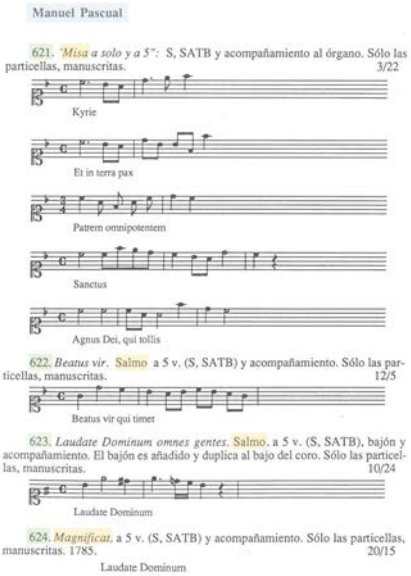

Figura 13. Criterios para clasificar las obras.

Por último, comentar que también se han almacenado las propias páginas en el formato *JPG* para poder mostrarlas en la página web que se ha construido como se describe a continuación.

2.5. Portal Web

Para el portal web hemos utilizado *HTML*, para mostrar toda la información almacenada, y el lenguaje de programación *JavaScript* para definir las funciones de búsqueda de texto y obras. Es interesante mencionar el elemento *Canvas* [Jeff Fulton, 2011], el cual fue incorporado a partir de *HTML5*. Con este elemento se puede generar cualquier tipo de gráfico en 2D, así como animaciones o ventanas interactivas dentro de la página web. Gracias a *Canvas* pudimos renderizar las imágenes de nuestro libro en tiempo real para así poder mostrar el resultado de las consultas.

3. RESULTADOS

A continuación, se muestran los resultados obtenidos en el conjunto de *test* por los modelos que hemos construido sin tener en cuenta el clasificador de las alturas de las notas. Vamos a comparar los cuatro modelos de detección utilizando las siguientes *métricas*: *Precison*, *Recall*, *F1-score*, *mAP* (del inglés, *mean Average Precision*) y *COCO-metric* [Hui, 2018], todas ellas dependen de un indicador conocido como el *IoU* (del inglés, *Intersection over Union*) [Rosebrock, 2016]. Dichas métricas se han obtenido tras aplicar la técnica de *Non-Maximum Suppression* [Rosebrock, 2014] para eliminar las predicciones que se superponen.

En la tabla 2 se encuentran los resultados finales de los modelos de detección que hemos probado.

Tabla 2. Resultados de la detección de símbolos musicales.

	Precision	Recall	F1-score	mAP	COCO	Tiempo
EfficientDet	0.28	0.14	0.19	13.86%	0.16	0.37 h
Faster R-CNN	0.86	0.77	0.82	76.73%	0.59	0.21 h
RetinaNet	0.73	0.15	0.25	14.73%	0.19	5.49 h
YOLO v4	0.89	0.90	0.90	68.25%	0.48	72 h

En vista de los resultados de la tabla 2, los modelos *EfficientDet* y *RetinaNet* quedan descartados, vemos que sus resultados son bastante bajos en comparación con los otros dos modelos. También, cabe destacar que a ambos modelos les ha llevado mucho más tiempo entrenarlos que al *Faster R-CNN* dentro de la librería *IceVision*, por lo que obtenemos peores resultados invirtiendo más tiempo. Para los otros dos modelos tenemos unos resultados más que aceptables en la detección. El modelo *YOLO v4* ha sido el que más tiempo de entrenamiento ha necesitado, pero también es el que mejores valores de *Precision*, *Recall* y *F1-score* nos ofrece. Las métricas *mAP* y *COCO-metric* son algo más altas en el *Faster R-CNN*, pero las del *YOLO v4* son bastante correctas. También hemos realizado algunas pruebas manuales

Zubía | núm. 40 (2022), pp. 177-196
ISSN 0213-4306

para ver cómo son las predicciones de cada modelo de forma visual y al final hemos decidido utilizar el modelo *YOLO v4*, ya que es el que menos se confunde de los cuatro modelos estudiados y, en la mayoría de casos, detecta todo de forma correcta. El modelo *Faster R-CNN* es bastante bueno pero comete algunos errores, como por ejemplo, confundir *corcheas* con *semicorcheas*.

Para medir la fiabilidad del clasificador de las alturas de las notas utilizamos la *precisión*, que es una proporción de aciertos y mide el número de aciertos entre el número total de imágenes. En el conjunto de *test* obtuvimos una precisión del 98.4% con el modelo *ResNet-18*, el entrenamiento duró 56 segundos y es por eso por lo que no entrenamos más clasificadores, ya que consideramos que el resultado era lo suficientemente bueno para las necesidades de nuestro proyecto.

Finalmente, para medir lo parecido que sonarán las canciones originales con respecto a las predichas podemos usar como métrica el valor de multiplicar el *F1-score* del modelo de detección ganador *YOLO v4* por la *precisión* del *ResNet-18*:

$$E = 0.9 \cdot 0.984 = 0.8856 \sim 88.56\ \%$$

siendo E la eficacia final de nuestro modelo de OMR. Es decir, hemos conseguido crear un modelo que, dada una imagen con una obra del libro, sea capaz de reproducir la música que representa con un 88.56 % de eficacia.

En la figura 14 se pueden encontrar dos predicciones de YOLO v4. La primera imagen pertenece al conjunto de entrenamiento y la segunda al de *test*.

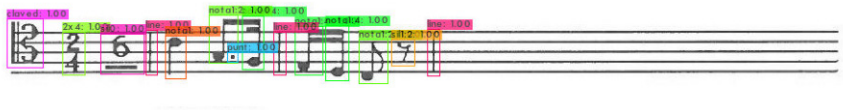

Figura 14. Predicciones en dos pentagramas del modelo Yolo v4.

En la figura 15 se puede ver una obra de nuestro libro procesada con *Audiveris* y nuestro modelo propio. En el caso de nuestro modelo, se detectan los bloques de texto y los pentagramas por separado. A los bloques de texto les aplicamos el *ORC* utilizando la librería *pytesseract* y a los pentagramas se les aplica nuestro *OMR*, el cual consiste en detectar los símbolos musicales con el modelo *YOLO v4*, predecir la altura de aquellos símbolos que sean notas musicales con el modelo *ResNet-18* y determinar la altura de aquellos símbolos que sean claves utilizando *OpenCV*. Finalmente, combinamos la información de cada bloque, ordenándolos verticalmente, para general el archivo *MusicXML*.

Vemos que, en las predicciones de la figura 15, nuestro modelo lo hace mejor que *Audiveris*.

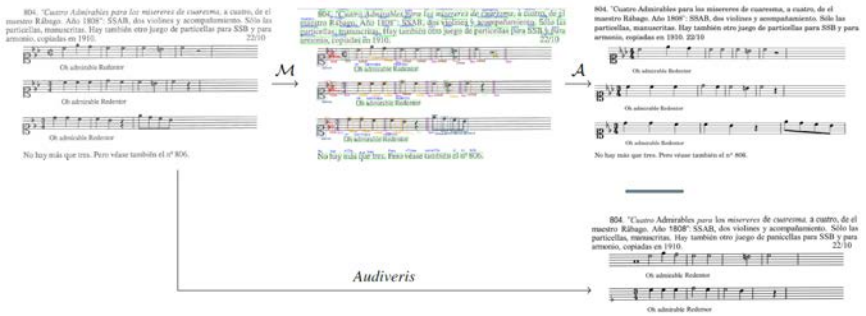

Figura 15. Resultado de aplicar OMR a la obra 804 del libro utilizando nuestro modelo propio y Audiveris.

Una vez visto el rendimiento de nuestros algoritmos, pasamos a explicar la funcionalidad a la que se puede acceder desde nuestra aplicación web. En concreto se ofrecen tres tipos de búsquedas. La primera de ellas sirve para consultar autores y/o tipos de obra, tiene dos ventanas desplegables que nos permiten acceder a los diferentes compositores y géneros musicales respectivamente. La segunda sirve para seleccionar una obra en concreto y también cuenta con una ventana desplegable. La tercera es para las búsquedas de texto. Para la búsqueda de texto hemos tenido en cuenta los saltos de línea, además de permitir ciertos errores de ortografía gracias a una función que normaliza las cadenas de caracteres. Así, por ejemplo, la frase: "Santo Domingo de La Calzada" será equivalente a: "säNtò DomigÓ delacalzAda".

4. DISCUSIÓN

A lo largo de este trabajo se ha llevado a cabo la digitalización, utilizando inteligencia artificial y visión por computador, de un libro que entremezcla texto y música. Cabe resaltar que el proceso de digitalización podría haberse realizado de manera manual; aunque esto hubiese sido una tarea tediosa, muy costosa en tiempo, y difícil de generalizar a otras obras.

El proceso que hemos seguido nos ha permitido definir una serie de técnicas de localización de objetos y extracción de información dentro de las páginas del libro. También, hemos almacenado los datos en distintos formatos y creado una página web para poder consultarlos. En consecuencia, no solo hemos cumplido con el objetivo que se pretendía inicialmente, sino que también hemos facilitado la futura digitalización de otros catálogos de archivos musicales en los que se hayan incluido íncipits o incluso de monografías de temas musicales que incluyan numerosos ejemplos musicales.

Es importante remarcar que las técnicas empleadas en este proyecto son innovadoras, ya que el volumen impreso al que nos enfrentamos era algo peculiar. Generalmente, los íncipits están formados por pentagramas y pequeños textos para el título, las voces o los instrumentos, y los sistemas

Zubía | *núm. 40 (2022), pp. 177-196*
ISSN 0213-4306

de *OMR* como el *Audiveris* [Bitteur, 2004], aunque normalmente funcionan bien, no están preparados para el tipo de formato presente en el libro, ya que no tienen en cuenta la diferenciación entre bloques de texto y pentagramas, o el hecho de que una página tenga que ser dividida en varias fichas que han de ser tratadas de manera independiente.

La metodología que hemos definido en este trabajo para aplicar *OMR*, una vez que tenemos la obra localizada, también ha sido diferente a la propuesta por los modelos disponibles en la literatura. Por ejemplo, en [Calvo-Zaragoza y Rizo, 2018] proponen un modelo *End-to-End* [Roza, 2019] para resolver el problema de transformar pentagramas individuales en música. En nuestro caso, nos basamos en modelos de detección y clasificación de imágenes digitales. En [Pacha, 2017] podemos encontrar algunos conjuntos de datos de *OMR* para la detección o la clasificación de símbolos musicales, pero no existe esa diferenciación entre primero la detección y luego la clasificación de las alturas tal y como se ha hecho en este proyecto. La idea detrás de todo esto reside en la manera en la que se leen las partituras musicales: de arriba hacia abajo y de izquierda a derecha, observando qué símbolos van apareciendo y comprobando de qué nota se trata o en qué clave estamos según la altura de los pentagramas.

Por último, conjeturamos al principio que el *OMR* era una tecnología que todavía no estaba consolidada y que había muchas líneas de investigación abiertas. Uno de los problemas era el de encontrar modelos que fuesen capaces de generalizar mejor. Nuestro modelo de *OMR* ha resultado ser más efectivo que los modelos ya existentes para transformar en música las imágenes de las obras del libro, tal y como se puede apreciar en la comparativa de la figura 15. Comparamos con *Audiveris* justamente por ser uno de los más utilizados y que, en principio, mejores resultados ofrecen. Esto permite afirmar que todavía queda trabajo por hacer en cuanto al problema de la generalización de los modelos de *OMR*.

5. CONCLUSIONES

En este trabajo se ha digitalizado el catálogo impreso del archivo musical de la Catedral de Santo Domingo. En dicho libro aparecen entremezclados párrafos de texto junto con obras musicales, que a su vez aparecen distribuidas en distintos pentagramas. Esta característica supone que el libro es diferente de una partitura musical, lo que origina que los programas de OMR desarrollados en la literatura no sean aplicables para su digitalización.

Para llevar a cabo la tarea de digitalización ha sido necesario el desarrollo de una metodología que conlleva la detección y clasificación de los distintos bloques (bien de texto, bien de música) que componen el libro, la organización de dichos bloques para configurar las obras que contiene, el empleo coordinado de técnicas de *OCR* para la interpretación de los bloques de texto, junto con el desarrollo de nuevos modelos de aprendizaje

profundo para realizar el *OMR* para la interpretación de las música. Además, todos los contenidos se han almacenado en una base de datos, para finalmente poder ser recuperadas y suministradas a los usuarios a través de una página web según distintos criterios de búsqueda. El resultado final permite que cualquier persona interesada en consultar el archivo musical acceda a él desde la página web del libro, donde se pueden consultar las obras de los diferentes autores, los distintos tipos de obras e incluso el sonido musical que representan. También hemos creado un sistema capaz de reconstruir las imágenes escaneadas a pdf.

La realización de este trabajo nos ha abierto más puertas y, como posible trabajo futuro sería interesante conseguir un conjunto de datos más amplio, teniendo en cuenta partituras que no pertenezcan a los volúmenes de "La Música en La Rioja", para construir un modelo de *OMR* basado en nuestra metodología que nos permita estudiar su capacidad de generalización, para así poder compararlo con el resto de modelos de la literatura.

AGRADECIMIENTOS

Trabajo cofinanciado por las áreas de Patrimonio Regional y de Ciencias del Instituto de Estudios Riojanos, por la Ayuda PID2020-115225RB-I00 financiada por MCIN/AEI/ 10.13039/501100011033 y por la Ayuda RTC-2017-6640-7.

REFERENCIAS BIBLIOGRÁFICAS

Adrian Rosebrock, Abhishek Thanki, S. P., y Haase, J. (2020a). *OCR with OpenCV, Tesseract and Python - Intro to OCR*. PyImageSearch.

Adrian Rosebrock, Abhishek Thanki, S. P., y Haase, J. (2020b). *OCR with OpenCV, Tesseract and Python - OCR Practitioner Bundle*. PyImageSearch.

Bangare, S., Dubal, A., Bangare, P., y Patil, S. (2015). Reviewing Otsu's method for image thresholding. *International Journal of Applied Engineering Research*, 10:21777–21783.

Bisong, E. (2019). *Google Colaboratory*, pages 59–64. Apress, Berkeley, CA.

Bitteur, H. (2004). Audiveris. https://github.com/audiveris.

Bochkovskiy, A. (2020). Yolo v4, v3 and v2 for Windows and Linux. https://github.com/AlexeyAB/darknet.

Bochkovskiy, A., Wang, C., y Liao, H. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934.

Bradski, A. (2008). *Learning OpenCV, Computer Vision with OpenCV Library; software that sees*. O'Reilly Media, 1 ed. edition.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Britannica, T. E. (2009). Midi: music technology. *Encyclopedia Britannica*.

Calvo-Zaragoza, J., Hajič Jr. J., y Pacha, A. (2021). Understanding optical music recognition. *ACM Computing Surveys*, 53(4), 77, 1–35

Calvo-Zaragoza, J. y Rizo, D. (2018). End-to-end neural optical music recognition of monophonic scores. *Applied Sciences*, 8(4).

Good, M. (2001a). MusicXML: An internet-friendly format for sheet music. *Recordare LLC*.

Good, M. (2001b). MusicXML for notation and analysis. *Recordare LLC*.

Good, M. (2013). MusicXML: The first decade. *MakeMusic, Inc., USA*.

Hankinson, A., Roland, P., y Fujinaga, I. (2011). The music encoding initiative as a document-encoding framework. pages 293–298.

Howard, J. et al. (2018). fastai. https://github.com/fastai/fastai.

Huang, Z., Jia, X., y Guo, Y. (2019). State-of-the-art model for music object recognition with deep learning. *Applied Sciences*, 9(13):2645–2665.

Hui, J. (2018). mAP (mean average precision) for object detection. https://jonathan-hui. medium.com/map-mean-average-precision-for-object-detection-45c121a31173.

Jeff Fulton, S. F. (2011). *HTML5 Canvas*. O'Reilly Media.

Kaiming He, Xiangyu Zhang, S. R. (2015). Resnet-18: Deep residual learning for image recognition. https://www.kaggle.com/pytorch/resnet18.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., y Dollár, P. (2018). Focal loss for dense object detection. https://arxiv.org/abs/1708.02002

Pacha, A. (2017). Optical music recognition datasets. https://github.com/apacha/OMR-Datasets.

Perez, L. y Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. *CoRR*, abs/1712.04621.

Pezoa, F., Reutter, J. L., Suarez, F., Ugarte, M., y Vrgoč, D. (2016). Foundations of JSON schema. In *Proceedings of the 25th International Conference on World Wide Web*, pages 263–273. International World Wide Web Conferences Steering Committee.

Rebelo, A., Fujinaga, I., Paszkiewicz, F., Marcal, A. R., Guedes, C., y Cardoso, J. d. S. (2012). Optical music recognition: state-of-the-art and open issues. *International Journal of Multimedia Information Retrieval*, 1(3):173–190.

Ren, S., He, K., Girshick, R. B., y Sun, J. (2015). Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497.

Rosebrock, A. (2014). Non-maximum suppression for object detection in python. https://www.pyimagesearch.com/2014/11/17/ non-maximum-suppression-object-detection-python/.

Rosebrock, A. (2016). Intersection over union (IoU) for object detection. https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/.

Roza, F. (2019). End-to-end learning, the (almost) every purpose ml method. https://towardsdatascience.com/e2e-the-every-purpose-ml-method-5d4f-20dafee4

Tan, M., Pang, R., y Le, Q. V. (2019). EfficientDet: Scalable and efficient object detection. *CoRR*, abs/1911.09070.

Van Rossum, G. y Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.

Vázquez, L. (2020a). IceVision. https://airctic.com/0.8.0/.

Vázquez, L. (2020b). IceVision: An agnostic object detection framework. https://github.com/airctic/icevision.

Yousefi, J. (2015). Image binarization using Otsu thresholding algorithm. https://www.researchgate.net/publication/277076039_Image_Binarization_using_Otsu_Thresholding_Algorithm