

---

# Past2Polygon: Un enfoque basado en Machine Learning para la digitalización de mapas históricos.

[GitHub](#)

**Facultad de Matemática y Computación. Universidad de La Habana.**

Ariel González Gómez  
[lenin46ariel@gmail.com](mailto:lenin46ariel@gmail.com)  
Leonardo Artiles Montero  
[leo16am@gmail.com](mailto:leo16am@gmail.com)

Alex S. Bas Beovides  
[abasbeovides@gmail.com](mailto:abasbeovides@gmail.com)

Paula Silva Lara  
[paula.silvalara030410@gmail.com](mailto:paula.silvalara030410@gmail.com)

Ricardo Cápiro Colomar  
[rikikpiro02@gmail.com](mailto:rikikpiro02@gmail.com)

Daniel Machado Pérez  
[daniel.machado.0206@gmail.com](mailto:daniel.machado.0206@gmail.com)

30 de enero de 2025

## Resumen

En este trabajo se presenta un procedimiento para la digitalización y modernización de mapas antiguos, con un enfoque particular en la ciudad de La Habana. El objetivo es transformar mapas históricos proporcionados por la Oficina del Historiador de la Ciudad de La Habana en representaciones digitales modernas, permitiendo el análisis de los cambios urbanos a lo largo del tiempo.

Inicialmente, los mapas son preprocesados mediante el recorte de bordes, la conversión a escala de grises y la aplicación de filtros para la eliminación de ruido. Posteriormente, se eliminan las etiquetas textuales y numéricas utilizando **PaddleOCR**, reemplazando las áreas detectadas con el color predominante mediante **K-means**. Luego, se emplea una **red neuronal convolucional** para generar un mapa de calor que diferencia calles y bloques de edificios. Las imágenes se segmentan con un algoritmo de **Flood Fill** y, utilizando la información del mapa de calor, se identifican y extraen los bloques urbanos.

Los bloques segmentados se convierten en polígonos, extrayendo sus vértices y aristas, y se simplifican con los algoritmos de **Ramer-Douglas-Peucker** (RDP) y **Shapely**. Para la georreferenciación, se identifican polígonos atípicos mediante **DBSCAN** y se comparan con bloques equivalentes en mapas actuales. Una vez encontrados los bloques correspondientes, se calculan sus centroides y se alinean los mapas históricos con los modernos mediante superposición geoespacial.

El método propuesto permite digitalizar y estructurar mapas antiguos en un formato compatible con herramientas de análisis geográfico moderno, facilitando el estudio de la evolución urbana de La Habana y su comparación con el entorno actual.

## Abstract

This work presents a procedure for the digitization and modernization of old maps, with a particular focus on the city of Havana. The objective is to transform historical maps provided by the Office of the Historian of

the City of Havana into modern digital representations, enabling the analysis of urban changes over time.

Initially, the maps undergo preprocessing, including border cropping, grayscale conversion, and noise removal using filtering techniques. Subsequently, textual and numerical labels are removed using **PaddleOCR**, replacing the detected areas with the predominant color through **K-means** clustering. A **convolutional neural network** is then used to generate a heatmap that distinguishes streets from building blocks. The images are segmented using a **Flood Fill** algorithm, and based on the heatmap information, urban blocks are identified and extracted.

The segmented blocks are converted into polygons, extracting their vertices and edges, and are simplified using the **Ramer-Douglas-Peucker** (RDP) and **Shapely** algorithms. For georeferencing, atypical polygons are identified through **DBSCAN** clustering and compared with equivalent blocks in modern maps. Once the corresponding blocks are found, their centroids are computed, and the historical maps are aligned with modern ones through geospatial overlay.

The proposed method enables the digitization and structuring of old maps into a format compatible with modern geographic analysis tools, facilitating the study of Havana's urban evolution and its comparison with the present-day environment.

---

**Palabras Clave:** Digitalización, Mapas Históricos, Georreferenciación, Segmentación de Imágenes, Aprendizaje Profundo, Visión por Computadora, PaddleOCR, DBSCAN, Heatmaps, Ramer-Douglas-Peucker, Optimización Bayesiana, Shapely, CNN, Índice de Jaccard

## ÍNDICE

<a href="#">1 Introducción</a>	<a href="#">2</a>
<a href="#">2 Preprocesamiento de Imágenes</a>	<a href="#">2</a>

---

2.1 Datos Útiles . . . . .	3	y precisión. En este trabajo, se propone un enfoque automatizado basado en visión por computadora y aprendizaje profundo para la segmentación, extracción y georreferenciación de mapas antiguos.
2.2 Datos Descartados . . . . .	3	
<b>3 Extracción y Eliminación de Etiquetas (Labels )</b>	<b>4</b>	El proceso comienza con el preprocesamiento de los mapas, eliminando ruido y mejorando la calidad de las imágenes mediante filtros y transformaciones.
3.1 Detección de Texto con PP-OCR . .	4	
3.2 Manejo de Imágenes de Mapas Históricos de Gran Tamaño . . . . .	5	La eliminación de etiquetas textuales se lleva a cabo con PaddleOCR, con optimización adicional basada en cortes y rotaciones de la imagen para mejorar la detección. Posteriormente, se emplea una red neuronal convolucional para generar un mapa de calor que distingue entre calles y bloques de edificios.
3.3 Eliminación de Texto con Agrupamiento K-Means . . . . .	5	
<b>4 Generación de <i>Heatmaps</i> de Calles</b>	<b>6</b>	Para la segmentación de los elementos urbanos, se utiliza un algoritmo de Flood Fill, integrando la información del mapa de calor para mejorar la identificación de bloques urbanos. Estos bloques se convierten en polígonos vectorizados, aplicando simplificación geométrica con los algoritmos de Ramer-Douglas-Peucker (RDP) y Shapely.
4.1 Creación del dataset . . . . .	6	
4.2 Arquitectura del Modelo . . . . .	6	
4.3 Proceso de Entrenamiento . . . . .	6	
4.4 Resultados . . . . .	7	
<b>5 Generación del <i>Heatmap</i></b>	<b>7</b>	
<b>6 Segmentación de Imágenes</b>	<b>8</b>	Una de las innovaciones clave de este trabajo es la optimización en la identificación de polígonos atípicos mediante DBSCAN, con la selección óptima de hiperparámetros a través de optimización bayesiana. Además, se introduce un algoritmo novedoso para la comparación de polígonos topológicamente similares, basado en la normalización a área unitaria, rotaciones alineadas por pares de lados trasladados al origen y la evaluación del Índice de Jaccard o Intersection over Union (IoU). Finalmente, los mapas históricos se alinean con mapas actuales mediante georreferenciación, permitiendo su integración en plataformas SIG modernas.
6.1 Metodología . . . . .	8	
6.1.1 Algoritmo de Flood Fill Modificado . . . . .	8	
6.1.2 Clasificación de Componentes	8	
6.2 Hiperparámetros . . . . .	9	
6.3 Salidas . . . . .	9	
<b>7 Vectorización de Polígonos</b>	<b>9</b>	
7.1 Vectorización del Raster . . . . .	9	
7.2 Simplificación de Polígonos . . . . .	10	
7.3 Corrección de Orientación . . . . .	10	
<b>8 Identificación de Polígonos Atípicos (<i>Outliers</i>)</b>	<b>10</b>	Este trabajo representa un avance significativo en la digitalización automatizada de mapas históricos, contribuyendo a la preservación de información geográfica valiosa y proporcionando nuevas herramientas para el análisis de la evolución urbana.
8.1 Métricas utilizadas . . . . .	10	
8.2 Detección de <i>Outliers</i> . . . . .	10	
<b>9 Georreferenciación</b>	<b>11</b>	
<b>10 Conclusiones</b>	<b>11</b>	

## 1 INTRODUCCIÓN

El estudio de mapas históricos constituye una herramienta fundamental en la investigación sobre la evolución urbana, ya que permite analizar patrones de crecimiento, cambios en la infraestructura y la conservación patrimonial. En el caso de La Habana, una ciudad con una rica historia urbanística, la digitalización de mapas antiguos ofrece una oportunidad única para evaluar su transformación a lo largo del tiempo.

Los enfoques tradicionales de digitalización de mapas han recurrido a la vectorización manual o a técnicas semiautomatizadas para la detección de estructuras urbanas. Sin embargo, estos métodos presentan limitaciones en términos de escalabilidad

## 2 PREPROCESAMIENTO DE IMÁGENES

Antes del procesamiento y análisis de los mapas, fue necesario realizar una etapa de preprocesamiento con el objetivo de mejorar la calidad de los datos y eliminar aquellos elementos que pudieran afectar el desempeño de los algoritmos de visión por computadora.

El preprocesamiento de las imágenes consistió en las siguientes tareas:

- **Recorte de bordes:** Algunos mapas presentaban daños físicos como roturas o pedazos faltantes de papel, lo que generaba ruido en la aplicación de filtros de OpenCV. Se eliminaron estas áreas afectadas para mejorar la uniformidad del dataset.

- **Selección de datos útiles:** Se identificaron mapas en mal estado, con representaciones desproporcionadas o ilegibles debido a su antigüedad. Estos fueron descartados para evitar problemas en el análisis posterior.

- **Mejora de imágenes:** Algunas imágenes fueron editadas utilizando herramientas como Photoshop para optimizar su calidad. Entre las modificaciones realizadas se encuentran ajustes de color, conversión a blanco y negro y eliminación de elementos no cartográficos como carteles o anotaciones manuscritas.

Tras el proceso de filtrado y clasificación, el conjunto de datos quedó organizado de la siguiente manera:

## 2.1 DATOS ÚTILES

- **La Habana:** 30 imágenes de mapas generales de la ciudad. 1
- **Barrios:** 41 imágenes de mapas detallados de distintos barrios. 2
- **Azules:** 50 imágenes de planos específicos de barrios de La Habana. 3

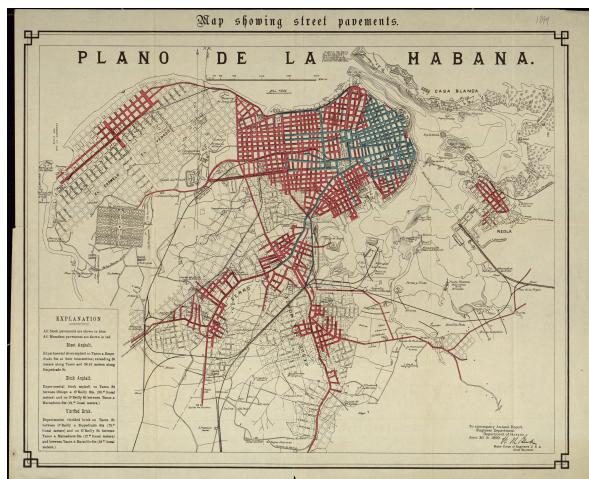


Figure 1: Mapa de La Habana

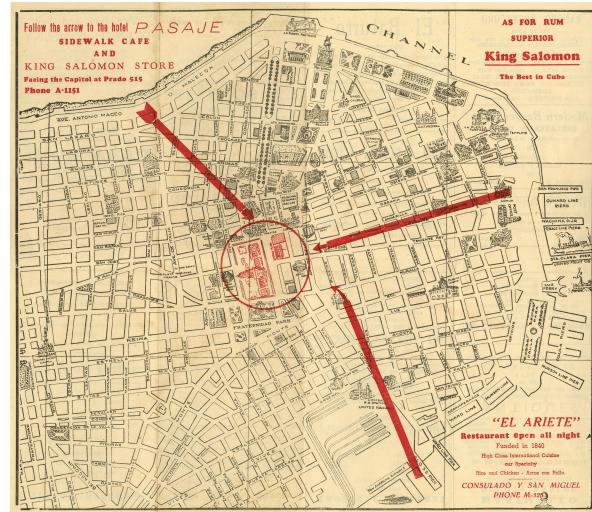


Figure 2: Mapa de un barrio de La Habana

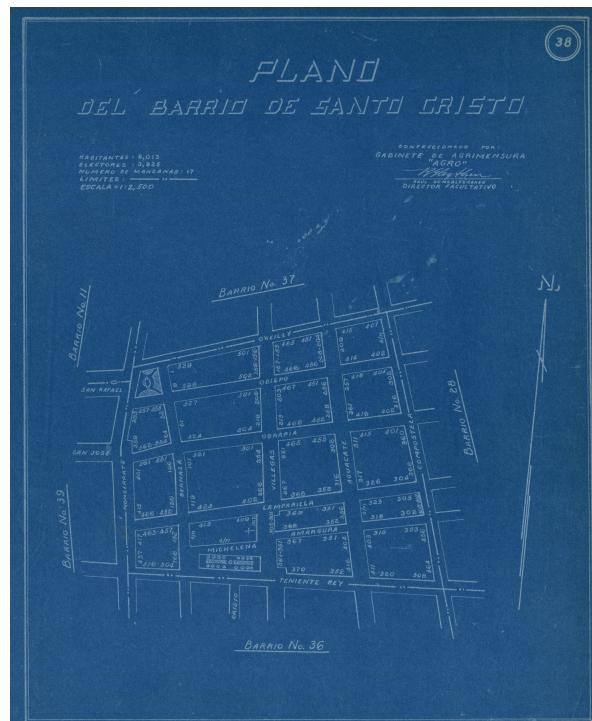


Figure 3: Plano azul de un barrio de La Habana

## 2.2 DATOS DESCARTADOS

- **Antiguos:** 19 imágenes de mapas demasiado antiguos con poca utilidad para el análisis. 4
- **Deficientes:** 90 imágenes en mal estado o con deficiencias graves. 5
- **3D:** 5 imágenes con representaciones tridimensionales que no eran compatibles con el modelo. 6



Figure 4: Mapa antiguo de La Habana



Figure 5: Mapa de la Playa Santa María, La Habana  
Mapa 3D de La Habana



Figure 6: Mapa 3D de La Habana

En la Figura 7 se muestra un ejemplo de una imagen antes y después de preprocesarla

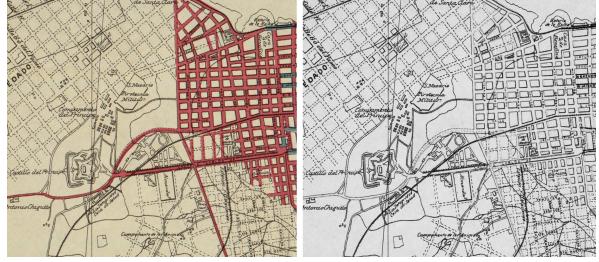


Figure 7: Mapa antes y después de preprocesar

### 3 EXTRACCIÓN Y ELIMINACIÓN DE ETIQUETAS (Labels)

Los mapas históricos contienen etiquetas textuales que identifican calles y lugares. Sin embargo, estas etiquetas pueden introducir ruido en los sistemas automatizados de análisis de mapas, ya que los trazos de las letras pueden ser malinterpretados como líneas que representan polígonos. La detección y eliminación de estas etiquetas es un paso crucial en el preprocesamiento para mejorar la interpretación automatizada de mapas históricos.



Figure 8: Imagen preprocesada.

#### 3.1 DETECCIÓN DE TEXTO CON PP-OCR

PP-OCR [2], es un sistema OCR ultraligero diseñado para eficiencia y precisión. Este modelo emplea *Differentiable Binarization* (DB) [4] para una segmentación robusta del texto y está basado en MobileNetV3 [3], lo que optimiza la velocidad de procesamiento y minimiza el consumo de memoria. Su entrenamiento se ha realizado en un conjunto diverso de 97,000 imágenes, incluyendo:

- 68,000 imágenes reales obtenidas de conjuntos de datos públicos y búsquedas en Baidu.
- 29,000 imágenes sintéticas con diversos estilos y orientaciones de texto.

Los conjuntos de datos empleados incluyen:

- **LSVT**: Texto en vistas urbanas a gran escala.
- **RCTW-17**: Texto en documentos escaneados.

- **MTWI 2018**: Imágenes con texto mixto.
- **CASIA-10K**: Texto en documentos oficiales.
- **SROIE**: OCR en recibos y facturas.
- **MLT 2019**: Detección de texto en múltiples idiomas.
- **MSRA-TD500**: Señales de tráfico y carteles.
- **CCPD 2019**: Reconocimiento de matrículas vehiculares.



Figure 9: Imagen con etiquetas detectadas.

### 3.2 MANEJO DE IMÁGENES DE MAPAS HISTÓRICOS DE GRAN TAMAÑO

Los mapas históricos suelen tener alta resolución, lo que plantea desafíos en la detección de texto con PP-OCR debido a su tamaño de entrada fijo. El redimensionamiento automático de imágenes grandes puede comprometer la resolución y afectar la detección de texto pequeño. Para abordar este problema, la imagen se segmenta en regiones más pequeñas antes del procesamiento, permitiendo detectar el texto con mayor precisión. Posteriormente, los polígonos de texto detectados se mapean a sus coordenadas originales en la imagen completa.

### 3.3 ELIMINACIÓN DE TEXTO CON AGRUPAMIENTO K-MEANS

Después de detectar el texto, su eliminación implica reemplazar los píxeles correspondientes con el color del fondo circundante. Métodos como la moda o el promedio de color no resultan efectivos debido a la variabilidad cromática del fondo. Por ello, se utiliza el algoritmo de agrupamiento K-Means para identificar el color predominante dentro del polígono de texto. El procedimiento es el siguiente:

- Se agrupan los colores dentro del polígono en 3 *clusters*.
- Se selecciona el *cluster* dominante como color representativo del fondo.

- Los píxeles del texto se reemplazan por este color, integrando visualmente el área editada con el resto de la imagen.

Este método ha demostrado ser efectivo sin requerir ajustes adicionales del hiperparámetro **n\_clusters**.



Figure 10: Imagen con etiquetas eliminadas.

## 4 GENERACIÓN DE Heatmaps DE CALLES

Se entrenó un modelo basado en redes neuronales convolucionales (CNN) para la detección de calles en mapas antiguos. El objetivo es generar un mapa de calor que permita identificar las regiones correspondientes a calles y diferenciarlas de bloques de casas. Este proceso es un paso fundamental para la posterior selección de los bloques tras la segmentación.

### 4.1 CREACIÓN DEL *dataset*

Para entrenar el modelo, se construyó un *dataset* a partir de recortes de mapas. Se intentó también generar un *dataset* sintético, pero este enfoque no resultó efectivo, ya que las imágenes generadas no lograban capturar la variabilidad y complejidad de los mapas reales, lo que llevó a un desempeño inferior en la detección de calles. Las imágenes fueron clasificadas en cuatro categorías 11:

- **Intersecciones de calles**: aproximadamente 1400 imágenes.
- **Calles (segmentos entre intersecciones)**: 1500 imágenes.
- **Partes del mapa que no son ni calles ni intersecciones**: 1500 imágenes.
- **Recortes aleatorios de los mapas**: cantidad variable.

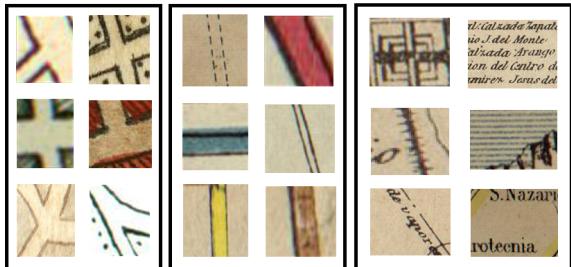


Figure 11: Ejemplos de imágenes del *dataset*.

Para mejorar la capacidad de generalización del modelo, se realizó data augmentation sobre la categoría de intersecciones de calles. En particular, cada imagen se rotó en tres ángulos diferentes ( $90^\circ$ ,  $180^\circ$  y  $270^\circ$ ), cuadruplicando así el número de muestras en esta clase.

## 4.2 ARQUITECTURA DEL MODELO

En la Figura 12 se muestra un esquema gráfico de la arquitectura de la CNN utilizada para la detección de calles:

El modelo utilizado es una CNN con la siguiente configuración:

- **Capas convolucionales:** Cuatro capas convolucionales con 32, 64, 128 y 256 filtros respectivamente, con kernel de tamaño  $3 \times 3$  y padding de 1.
  - **Normalización y activación:** Se emplea *Batch Normalization* después de cada capa convolucional y ReLU como función de activación.
  - **Regularización:** Se utiliza una capa de *dropout* con una tasa del 50% para evitar el sobreajuste.
  - **Pooling:** Se aplica *max pooling* con un tamaño de ventana  $2 \times 2$  y *stride* de 2, reduciendo progresivamente la dimensionalidad espacial de las características extraídas.
  - **Clasificador:** Dos capas completamente conectadas, una con 512 neuronas y otra de salida con 2 neuronas para la clasificación binaria (calles vs. no calles).

#### 4.3 PROCESO DE ENTRENAMIENTO

Se empleó la función de pérdida de entropía cruzada y el optimizador AdamW con una tasa de aprendizaje ajustable mediante un *scheduler* que reducía la tasa si la pérdida de validación dejaba de mejorar.

La base de datos se dividió en 80% para entrenamiento y 20% para validación. Se utilizó un tamaño de batch de 32 y se entrenó por 10 épocas.

#### 4.4 RESULTADOS

Durante el entrenamiento, se observó un comportamiento inusual en las primeras épocas: la pérdida en el conjunto de validación era menor que la pérdida en el conjunto de entrenamiento, y el *accuracy* de validación también era mayor que el de entrenamiento. Este fenómeno puede deberse a varios factores:

- **Regularización efectiva:** Es posible que el modelo generalizara bien desde el principio debido al uso de *Batch Normalization* y *Dropout*, lo que impide un sobreajuste rápido a los datos de entrenamiento.
  - **Efecto del data augmentation:** Como el conjunto de entrenamiento incluye imágenes con transformaciones adicionales, esto puede haber aumentado su complejidad inicial en comparación con las imágenes del conjunto de validación, que no sufrieron las mismas alteraciones.

A medida que avanzaron las épocas, esta diferencia se redujo y el modelo terminó mostrando una tendencia más esperada, con la pérdida de entrenamiento disminuyendo más rápidamente 13 y el *accuracy* de entrenamiento alcanzando valores superiores a los de validación. 14

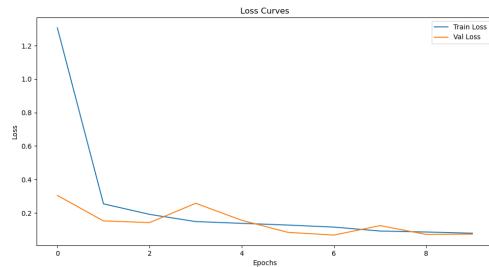


Figure 13: Gráfica de la función de pérdida.

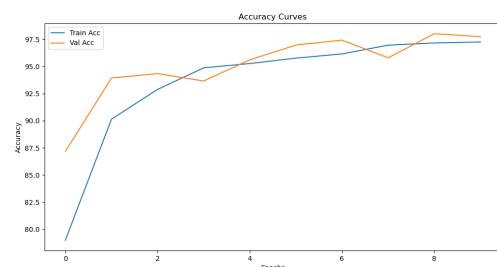


Figure 14: Gráfica de la función de *accuracy*.

Se probaron diferentes enfoques para el entrenamiento del modelo. El mejor desempeño se obtuvo cuando se entrenó el modelo exclusivamente con la categoría de intersecciones de calles.

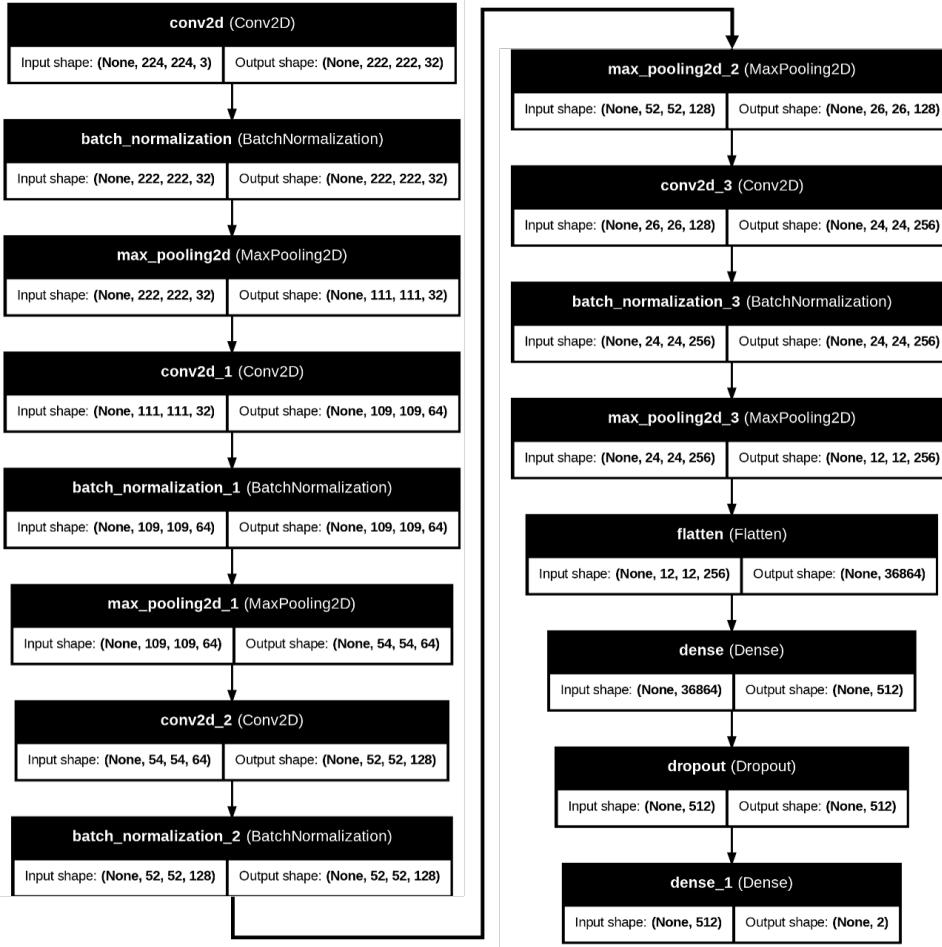


Figure 12: Esquema de la arquitectura de la CNN utilizada.

aplicando data augmentation mediante rotaciones. Este enfoque permitió mejorar significativamente la capacidad del modelo para reconocer patrones en mapas antiguos y distinguir calles de bloques de casas.

## 5 GENERACIÓN DEL *Heatmap*

Para generar un mapa de calor que permita identificar las zonas correspondientes a calles en los mapas antiguos, se utilizó un enfoque basado en una ventana deslizante sobre la imagen. Se aplicó el modelo entrenado a pequeñas secciones de la imagen original, obteniendo una probabilidad de presencia de calles en cada región. Los resultados se combinaron en un mapa de calor normalizado para representar gráficamente las zonas con mayor probabilidad de ser calles.

Además del enfoque basado en la ventana deslizante, se experimentó con técnicas como Grad-CAM y cuadriculado para visualizar la relevancia de las activaciones del modelo. Sin embargo, estos métodos no ofrecieron resultados tan precisos como el enfoque de ventana deslizante, por lo que se optó por este último en la generación final del *heatmap*.



Figure 15: Ejemplo de *heatmap* aprendido.

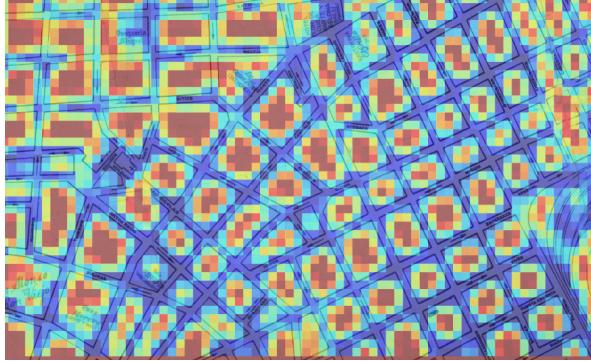


Figure 16: Ejemplo de *heatmap* aprendido.

## 6 SEGMENTACIÓN DE IMÁGENES

En esta fase del proyecto, se aborda la segmentación de las imágenes con el objetivo de identificar y aislar componentes conexas cromáticamente similares. Este proceso es crucial para la posterior vectorización de bloques de edificios y generación de puntos de control. La segmentación se realiza sobre las imágenes obtenidas de la fase de eliminación de etiquetas, generando como resultado un conjunto de máscaras binarias que representan los contornos de las componentes identificadas, además de información asociada a cada componente.

### 6.1 METODOLOGÍA

La segmentación se lleva a cabo mediante un algoritmo de *Flood Fill* modificado, similar al funcionamiento de la herramienta "cubo de pintura" en programas de edición de imágenes. Este algoritmo se aplica de forma iterativa a cada píxel no visitado, explorando sus vecinos y agrupando aquellos que comparten características de color similares.

#### 6.1.1 ALGORITMO DE FLOOD FILL MODIFICADO

El algoritmo de *Flood Fill* se adapta para la segmentación considerando la similitud cromática entre píxeles. El proceso se describe como sigue:

1. Se inicia con un píxel no visitado como semilla.
2. Se expande a los píxeles vecinos, incluyendo los píxeles adyacentes por los lados, y opcionalmente por las diagonales, verificando si la diferencia de color con respecto al píxel inicial (o al último píxel visitado) es menor que un umbral ( $k$ ).
3. Si la diferencia de color es menor que  $k$ , el píxel se agrega a la componente actual y se marca como visitado.
4. El proceso se repite iterativamente hasta que no queden píxeles conectados que cumplan con el umbral de diferencia de color.

5. Se obtienen los píxeles que forman la componente segmentada.

La diferencia de color entre dos píxeles se calcula mediante una distancia, la cual puede ser la diferencia absoluta de los valores RGB, o la distancia Euclídea en el espacio de color RGB. Sean  $c_1 = (r_1, g_1, b_1)$  y  $c_2 = (r_2, g_2, b_2)$  los colores de dos píxeles, la diferencia de color se calcula como:

- Distancia Absoluta:

$$\text{diff}(c_1, c_2) = |r_1 - r_2| + |g_1 - g_2| + |b_1 - b_2|$$

- Distancia Euclídea:

$$\text{diff}(c_1, c_2) = \sqrt{(r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b_2)^2}$$

Una vez completado el proceso de segmentación de una componente, se calcula el *bounding box* que la engloba, almacenando su ubicación (píxel de la esquina superior izquierda) y dimensiones (ancho y alto).

#### 6.1.2 CLASIFICACIÓN DE COMPONENTES

Tras segmentar las imágenes en componentes, se realiza una clasificación para distinguir entre componentes que representan bloques de edificios y aquellas que no. Esta clasificación se basa en el análisis de un mapa de calor previamente generado en la fase anterior. Para cada componente, se calcula el promedio de las probabilidades de ser parte de un bloque de edificios de todos sus píxeles.

Si el promedio de probabilidades supera un umbral determinado, la componente se considera un bloque de edificios; de lo contrario, se clasifica como un componente no relacionado. Además, se descartan componentes de tamaño inferior a un umbral mínimo, ya que se consideran ruido o texto en la imagen.

## 6.2 HIPERPARÁMETROS

La precisión de la segmentación depende de la configuración de varios hiperparámetros, los cuales se ajustan de forma empírica:

- $k$ : Umbral de similitud de color. Determina la tolerancia en la diferencia de color para considerar dos píxeles como parte de la misma componente.
- *use8Way*: Booleano que activa o desactiva el uso de los 8 vecinos en la búsqueda de componentes conexas. Si es falso, solo los 4 vecinos ortogonales.
- *euclidif*: Booleano que activa o desactiva el uso de la distancia euclídea en vez de la absoluta para calcular la diferencia de color.

- *adj*: Booleano que indica si la comparación se hace con respecto al ultimo pixel visitado o el inicial.
- *minComponentSize*: Tamaño mínimo (en número de píxeles) para que una componente sea considerada válida.
- *buildingBlockThreshold*: Umbral de probabilidad promedio para clasificar una componente como un bloque de edificios.

### 6.3 SALIDAS

Como resultado de esta fase, se obtienen los siguientes archivos por cada imagen procesada:

- Máscaras binarias: Imágenes en formato JPG que contienen las máscaras binarias de cada componente identificada, separadas en dos carpetas: una para componentes clasificadas como bloques de edificios y otra para las no relacionadas.
- Imagen de segmentación: Una imagen en formato JPG que visualiza la segmentación, donde cada componente se muestra con un color aleatorio.
- Imagen de componentes de edificios: Una imagen en formato JPG que visualiza las componentes clasificadas como bloques de edificios.
- Archivo de información: Un archivo en formato JSON con información detallada sobre cada componente extraída, incluyendo su bounding box (coordenadas de la esquina superior izquierda, ancho y alto) y la probabilidad de que sea un bloque de edificios.

Estos resultados constituyen la entrada para las siguientes etapas del proyecto, en particular, la detección y clasificación de bloques de edificios a partir de las componentes segmentadas.

## 7 VECTORIZACIÓN DE POLÍGONOS

La vectorización constituyó un paso esencial en el procesamiento de los datos, permitiendo la conversión de estructuras raster a representaciones geométricas vectoriales. Este proceso comenzó con la recepción de imágenes que contenían los polígonos de interés, junto con la información complementaria almacenada en el archivo `components.info.json`. La combinación de estas fuentes de datos permitió garantizar una correspondencia precisa entre los elementos geoespaciales y sus atributos asociados.

Inicialmente, cada imagen fue procesada para su preparación antes de la vectorización. El procedimiento incluyó la conversión a escala de grises y la binarización mediante la aplicación de un umbral fijo. Posteriormente, la imagen binarizada fue almacenada en formato GeoTIFF con la proyección espacial EPSG:4326. Este formato permitió conservar las referencias espaciales y facilitó la integración con herramientas SIG, como QGIS, para su visualización y análisis posterior.

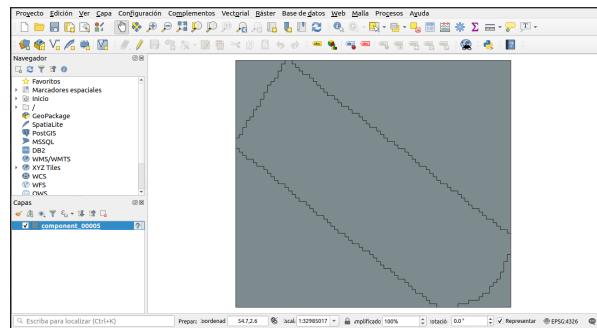


Figure 17: Ejemplo de uso de QGIS.

### 7.1 VECTORIZACIÓN DEL RASTER

Una vez obtenida la imagen binarizada, se empleó el módulo `gdal_polygonize.py` para la vectorización. Este proceso generó un conjunto de polígonos almacenados en formato Shapefile. En esta etapa, se aplicó un filtrado para seleccionar el polígono principal basado en su área como criterio de prioridad. Este filtrado permitió eliminar elementos espurios y garantizar que la información extraída fuera relevante para el análisis.

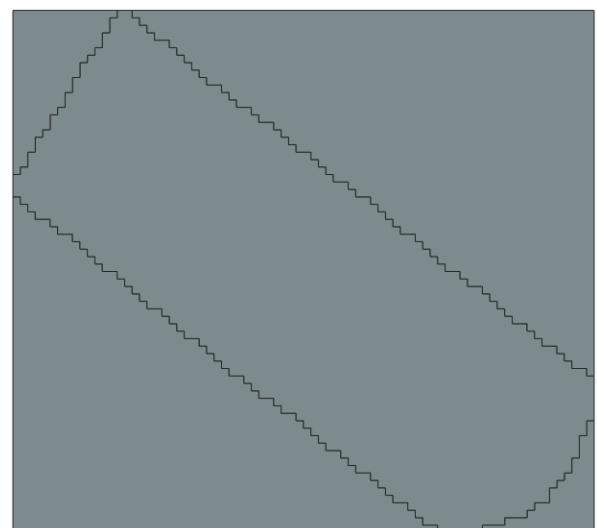


Figure 18: Resultado de vectorizar el polígono.

## 7.2 SIMPLIFICACIÓN DE POLÍGONOS

Para reducir la complejidad geométrica de los polígonos vectorizados, se implementó una fase de simplificación mediante dos métodos distintos: el algoritmo de Ramer-Douglas-Peucker (RDP) [5] y el método de simplificación de Shapely [6]. Se realizaron comparaciones entre ambos enfoques para evaluar su efectividad en la preservación de la estructura original del polígono mientras se minimizaba la cantidad de vértices. Las principales diferencias en cuanto a los resultados se sustentan en lo siguiente:

- **rdp:** Implementa únicamente Douglas-Peucker, lo que lo hace específico para simplificación basada en la distancia.
- **shapely:** Es más general, basado en un algoritmo de topología-preservación y puede ser más preciso en algunos casos.

Una diferencia evidente se puede observar en la Figura ??

## 7.3 CORRECCIÓN DE ORIENTACIÓN

Dado que el proceso de vectorización podía generar una inversión en la orientación de los polígonos, se aplicó una transformación geométrica (*flip*) para corregir esta situación. Esto aseguró la correcta orientación de los datos conforme a la configuración esperada en el sistema de referencia.

## 8 IDENTIFICACIÓN DE POLÍGONOS ATÍPICOS (*Outliers*)

Para detectar polígonos con características significativamente diferentes al resto, se utilizó el algoritmo DBSCAN. Primero, a partir de las coordenadas de los polígonos identificados en las fases anteriores, se calcularon diversas métricas relevantes para el análisis.

### 8.1 MÉTRICAS UTILIZADAS

- **Número de lados:** Los polígonos con mayor cantidad de lados tienden a ser más atípicos.
- **Área:** Es una característica fundamental para otras métricas.

$$A = \frac{1}{2} \left| \sum_{i=1}^n (x_i y_{i+1} - x_{i+1} y_i) \right|$$

[1]

- **Perímetro:** También es una característica importante para otras métricas.

$$P = \sum_{i=1}^n \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}$$

- **Compacidad:** Evalúa qué tan "compacto" es un polígono. Para un polígono dado, un valor más alto indica que tiene un área grande en relación con su perímetro.

$$\text{Compacidad} = \frac{\text{Area}}{\text{Perimeter}^2}$$

- **Circularidad:** La circularidad compara un polígono con un círculo perfecto, que tiene el valor máximo posible de circularidad (1). Los polígonos más cercanos a un círculo tendrán valores altos, mientras que formas más irregulares (estrechas, alargadas o dentadas) tendrán valores bajos.

$$\text{Circularidad} = \frac{4\pi \text{Area}}{\text{Perimeter}^2}$$

- **Convexidad:** Esta métrica mide qué fracción del área del *convex hull* está ocupada por el polígono. Un valor de convexidad muy cercano a 1, significa que el polígono es muy cercano a ser convexo.

- **Coeficiente de variación:** Esta métrica permite identificar cuánto varían las longitudes de los lados en relación con su media.

$$CV = \frac{\sigma}{\mu}$$

Donde  $\sigma$  representa la desviación estándar y  $\mu$  la media de los datos.

### 8.2 DETECCIÓN DE *Outliers*

Después de calcular las métricas seleccionadas como características para cada polígono, se procede a normalizar o estandarizar los datos. Si los valores siguen una distribución normal, se realiza una estandarización (restando la media y dividiendo por la desviación estándar); en caso contrario, se aplica normalización (ajustando los valores a un rango definido, como [0,1]).

Luego, se utiliza la optimización bayesiana con el objetivo de encontrar los mejores hiperparámetros para próximamente aplicar el algoritmo DBSCAN. La optimización bayesiana es un método para encontrar los valores óptimos de hiperparámetros en funciones costosas de evaluar. Se basa en la construcción de un modelo probabilístico (usualmente un proceso gaussiano) para estimar la función objetivo y seleccionar los próximos puntos a evaluar de manera eficiente.

Utiliza una función de adquisición para equilibrar la exploración y explotación, reduciendo la cantidad de evaluaciones necesarias en comparación con métodos de búsqueda exhaustivos. Con esta técnica son hallados los hiperparámetros **eps** y

`min_samples` de DBSCAN, maximizando la detección de outliers en los polígonos analizados.

Función objetivo:

$$\arg \min_{\epsilon \in [0.1, 1.0], \text{ min\_samples} \in \{2, \dots, 20\}} -\text{Outliers}(\epsilon, \text{min\_samples})$$

Posteriormente, se aplica el algoritmo DBSCAN para detectar los *outliers*, los cuales corresponden a aquellos polígonos considerados atípicos en relación con el resto de la muestra.

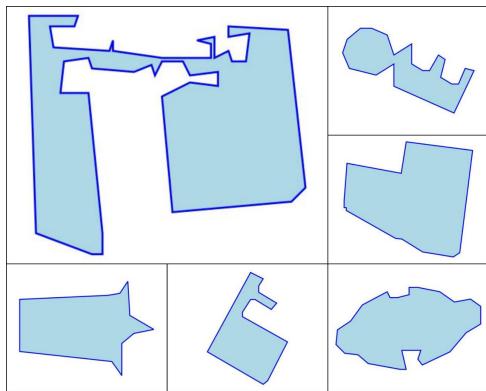


Figure 19: Polígonos *Outliers*.

## 9 GEORREFERENCIACIÓN

La georreferenciación se basa en el uso de puntos de control previamente identificados para calcular una matriz de transformación. En este caso, se emplea una matriz de homografía, la cual permite convertir coordenadas de píxeles de una imagen a coordenadas geográficas (latitud y longitud). A partir de esta transformación, se determinan los límites geográficos de la imagen, definiendo así su extensión en el mapa de *Open Street Maps*. Finalmente, las imágenes georreferenciadas se superponen como capas sobre un mapa interactivo, permitiendo su visualización y análisis.



Figure 20: Mapa superpuesto.

## 10 CONCLUSIONES

### REFERENCIAS

- [1] Arteaga Moreno, F. J. (2009). Cálculo del área de un polígono simple: una demostración personal. Lección inaugural, septiembre de 2009.
- [2] Du, Y., Li, C., Guo, R., Yin, X., Liu, W., Zhou, J., ... & Wang, H. (2020). Pp-ocr: A practical ultra lightweight ocr system. arXiv preprint arXiv:2009.09941.
- [3] Koonce, B., & Koonce, B. (2021). MobileNetV3. Convolutional Neural Networks with Swift for Tensorflow: Image Recognition and Dataset Categorization, 125-144.
- [4] Liao, M., Zou, Z., Wan, Z., Yao, C., & Bai, X. (2022). Real-time scene text detection with differentiable binarization and adaptive scale fusion. IEEE transactions on pattern analysis and machine intelligence, 45(1), 919-931.
- [5] Ehret, U., & Neuper, M. (2014, May). Applying the Ramer-Douglas-Peucker algorithm to compress and characterize time-series and spatial fields of precipitation. In EGU General Assembly Conference Abstracts (p. 13537).
- [6] Gillies, S. (2013). The shapely user manual. URL <https://pypi.org/project/Shapely>.