DISEÑO DE UN SISTEMA INTELIGENTE QUE EXTRAIGA INFORMACIÓN DE GRÁFICAS LINEALES ACADÉMICAS DEL ÁREA DE LA CIENCIA DE LOS MATERIALES

ESTEBAN FELIPE CÁCERES GELVEZ

UNIVERSIDAD FRANCISCO DE PAULA SANTANDER

FACULTAD DE INGENIERÍA

PLAN DE ESTUDIOS INGENIERÍA INDUSTRIAL

CÚCUTA

2019

DISEÑO DE UN SISTEMA INTELIGENTE QUE EXTRAIGA INFORMACIÓN DE GRÁFICAS LINEALES ACADÉMICAS DEL ÁREA DE LA CIENCIA DE LOS MATERIALES

ESTEBAN FELIPE CÁCERES GELVEZ

Directora
GAUDY CAROLINA PRADA BOTIA
Ingeniero Mecánico

UNIVERSIDAD FRANCISCO DE PAULA SANTANDER

FACULTAD DE INGENIERÍA

PLAN DE ESTUDIOS INGENIERÍA INDUSTRIAL

CÚCUTA

2019

Tabla de Contenidos

Lista de tablas

Lista de figuras

Introducción

La información ha sido nombrada por muchos como el nuevo petróleo del siglo XXI. Y no es para más, pues gracias a la alta generación de datos de los últimos años, y a la creciente capacidad computacional, ahora son factibles técnicas de sistemas inteligentes diseñadas en los años 60, que, por las condiciones de la época, no se habían podido explotar. Sin embargo, la no estandarización de los datos al momento de crearlos genera problemas y al mismo tiempo desafíos para investigadores que desean recuperar información, analizar datos o encontrar patrones. Un caso de estos, es el de las figuras académicas generadas en artículos científicos, pues los motores de búsqueda comunes no pueden acceder a la información que hay dentro de ellas, dificultando así, la tarea para el investigador.

Una forma de solucionar esta problemática es generando un sistema que sea capaz de extraer la información que está dentro de los gráficos y convertirla a tablas o a codificaciones fáciles de interpretar para una máquina. Por lo anterior, el presente modelo, a través de la ciencia de la computación, diseñó un sistema inteligente capaz de detectar texto y valores numéricos dentro de una gráfica lineal, para así, facilitar el análisis estadísticos y predictivos en el área de la ciencia de los materiales.

Este modelo, usando códigos ya existentes, extrajo las imágenes de los PDF’s del área de los materiales, para luego a través de sistemas inteligentes, obtener la información que hay en las mismas. Con esto se pretende ofrecer a los investigadores una herramienta fácil de usar y poderosa, capaz de exportar datos coherentes, correctos y precisos sobre materiales poliméricos, metálicos o cerámicos.

1. Problema

1.1. Título

Diseño de un sistema inteligente que extraiga información de gráficas lineales académicas del área de la ciencia de los materiales.

1.2. Planteamiento del problema

Los gráficos son ampliamente usados en reportes económicos, libros académicos y artículos científicos. La simplicidad y usabilidad de los mismos hace que en la literatura se hayan creado diferentes tipos de gráficos, inclusive. Sin embargo, para la mayoría de gráficos estáticos, no se tiene acceso a la data original (Dai, Wang, Niu, & Zhang, 2018). Esto es un problema enorme, especialmente en la academia, donde obtener los datos originales de los experimentos mecánicos, térmicos y de cualquier otra naturaleza, permitiría a un investigador encontrar patrones, correlaciones o rediseñar y repensar los gráficos. Asimismo, la mayoría de motores de búsqueda no pueden entender la forma en la que la mayoría de gráficas se presentan en los artículos pues estas no pueden decodificar la información numérica y textual de las gráficas. Esto impide un análisis profundo de las figuras y/o un rediseño o reúso de las visualizaciones.

Para ejemplificar la magnitud del problema el problema, (Sagnik Ray Choudhury, Wang, Mitra, & Giles, 2015) encontraron que, entre 10000 artículos publicados en las mejores 50 conferencias del mundo de la ciencia de la computación, entre 2004 y 2014, más del 70% contenía al menos una figura. A pesar de estas cifras, la literatura solo ha realizado trabajos sobre extracción y entendimiento de tablas, prestándole menos atención a las figuras (Carberry, Elzer, & Demir, 2006).

Por otro lado, existe una dificultad inherente en las gráficas al momento de procesarlas. Según (Falk Böschen, Beck, & Scherp, 2018), existen características que hacen que se presenten dificultades al momento de extraer su información, tales como el hecho de que tenga cuadrículas o una estructura de rejilla dentro de la misma, que no esté la leyenda dentro de la gráfica o que el fondo de la imagen no sea blanco. Según estos autores, aproximadamente el 42% de los artículos tienen al menos una de estas complicaciones. Existen otros desafíos, como la variedad en los colores, las curvas que se superponen, el tamaño y la orientación de los textos (F Böschen & Scherp, 2017). Estos problemas suponen desafíos de investigación interesantes que deberían abarcarse por la academia, sin embargo, no han sido tratados en la literatura.

La tarea, por ende, que se debe hacer con los gráficos, es un proceso de ingeniería inversa, donde los gráficos académicos sean convertidos en tablas, similares a las que se usaron para generarlos. Esto se hace con el fin de soportar búsquedas informativas por una máquina. Además, permitiría extraer información también y comparar métodos, mejorando así las búsquedas académicas, pues actualmente solo dos recientes páginas web (Viziomatriz y Semantic Scholar) permiten buscar figuras *per se* (Sagnik Ray Choudhury, 2017).

La forma de abarcar este problema ha sido a través de aprendizaje semi-supervisado. Sin embargo, estos enfoques no son escalables ni factibles con la cantidad de literatura científica que se publica hoy por hoy. Por esto, métodos no supervisados se necesitan para solucionar el problema de la extracción de texto. Asimismo, otra solución que se ha dado a este problema, es la extracción de las gráficas, sus textos y sus leyendas, para que pueda ser fácilmente accesible, como es el caso de Viziomatriz y Semantic Scholar. Sin embargo, se extraen las gráficas como tal, no los datos que están dentro de la misma.

Por lo anterior, se planteó la necesidad de diseñar un sistema inteligente capaz de extraer información de gráficas lineales en el área de las ciencias de los materiales. Esto permite a través de técnicas de inteligencia artificial, de algoritmos evolutivos, de procesamiento digital de imágenes y de visión computacional, obtener información de comportamientos mecánicos, térmicos y de otras naturalezas. En el largo plazo, los investigadores podrán predecir de manera más efectiva cómo cambian las propiedades de los materiales en función de ciertos métodos, procesos o añadidos, y así, se aportará en la aceleración de nuevas técnicas, compuestos.

1.3. Formulación del problema

¿A través de qué técnicas de las ciencias de la computación se pueden obtener grandes cantidades de información de gráficas académicas del área de la ciencia de los materiales?

1.4. Justificación

Debido al aumento en la generación de información en los artículos científicos, se ha acrecentado el número de gráficas. Sin embargo, esta información no es accesible a los motores de búsqueda. Una gran pérdida, pues sobre todo en el tema de la ciencia de los materiales, las gráficas contienen resultados experimentales que no están necesariamente en el texto (Choudhury & Giles, 2015). El aplicar técnicas de inteligencia artificial y de las ciencias de la computación, por ende, permite que la información se pueda recolectar fácilmente. Esto permitirá a investigadores acceder a la data original y hacer análisis estadísticos, aplicar técnicas de aprendizaje de máquina, para así, acelerar el conocimiento tecnológico que se tiene sobre el tema. Permite, además, que motores de búsqueda puedan acceder a esta información. Ya una vez la información esté codificada y accesible para los mismos, las técnicas que se pueden aplicar a estos datos son infinitas.

La pertinencia de este estudio, además, se ve reflejada en la tendencia del país y del mundo, de transitar hacia la cuarta revolución industrial. El trabajar con extracción y procesamiento de datos compete una de las áreas más interesantes y grandes de esta revolución. Asimismo, contribuye a expandir las áreas de actuación investigativas del ingeniero industrial y aporta a una de las líneas del grupo donde se realiza la investigación, Mindlab, que es la recuperación de información.

Por último, es importante recalcar la aplicación de técnicas propias de la ingeniería industrial, como lo es el área de la ciencia de los materiales, junto con una de las disciplinas científicas que más hincapié ha tenido en la industria 4.0, como lo es la visión computacional. La combinación de estas dos áreas del conocimiento permitió solucionar un problema poco abarcado en la literatura, como lo es la extracción de información de figuras académicas.

1.5.Objetivos

1.5.1. Objetivo general

Diseñar un sistema inteligente que extraiga información de gráficas lineales académicas del área de la ciencia de los materiales

1.5.2. Objetivos específicos

Extraer las figuras y su respectiva información de los archivos PDF que contienen los artículos académicos del área de la ciencia de los materiales.

Formular un modelo de clasificación binario de figuras académicas a través de técnicas de inteligencia artificial.

Procesar las líneas y la metadata que se encuentre dentro de las figuras académicas clasificadas.

1.6.Alcances y limitaciones

1.6.1. Alcances

Se planteó aplicar el sistema en artículos del área de la ciencia de los materiales, pues es un campo donde se generan altas cantidades de gráficos y de trabajos. En lo que confiere a la extracción de las imágenes de los archivos PDF se usaron diferentes alcances ya hechos previamente. Sin embargo, se realizó desde cero el modelo de clasificación de imágenes y el procesamiento de las líneas y de la metadata dentro de la imagen. Una vez el sistema esté completo, el código se dispondrá libremente de manera accesible.

1.6.2. Limitaciones

La temática del proyecto abarcó los siguientes temas: procesamiento digital de imágenes o visión por computador, aprendizaje de máquina y ciencia de los materiales. Existe, además, una posible delimitación de tipo temporal: los datos extraídos serán de los últimos diez años, de nuevo, del área de la ciencia de los materiales.

Se contempló, además, una limitación operativa en lo referente a la obtención de la información, pues no existe un dataset con el que se pueda entrenar el modelo de clasificación de imágenes. Por esto, se generó de manera manual un conjunto de datos que le permitió a un algoritmo inteligente aprender cuando una imagen es de tipo lineal o no. Asimismo, se planteó una limitación en lo que confiere al tipo de imagen que se haya incrustado en el PDF. Se recomendó que sea de tipo SVG, pues, por ejemplo, al existir líneas superpuestas, no habrá problema para separar cada una de ellas.

2. Marco Referencial

2.1. Antecedentes

Tabla 1
Antecedentes en la extracción de información en figuras académicas

Titulo	Descripción	Nivel	Autores	Fuente
A comparison of approaches for automated text extraction from scholarly figures	Evalúa y compara 32 configuraciones de extracción de texto de 4 datasets de figuras académicas. En total, se corrieron los experimentos en 400 figuras. Esto permite al presente trabajo entender cuáles algoritmos y métodos previos son mejores y sobre los cuales se puede basar el trabajo. Estos son: adaptive binarization, que tiene mejor rendimiento, pues se adapta a distintas configuraciones. En lo que compete a la OCR machine, la mejor según la evaluación es Ocropy.	Internacional	(F Böschen & Scherp, 2017)	Repositorio Kiel University, Alemania
Automated data extraction from scholarly line graphs	Explican el funcionamiento de la extracción de información de las líneas en un gráfico. Asimismo, proponen el sistema que implementarán, que primero, identifica gráficos que tengan líneas, luego extrae el texto y las curvas. Usan el aprendizaje de características no supervisada. Aporta al presente trabajo, pues expone la siguiente metodología: el primer filtro desarrollado lo que hace es extraer estas gráficas lineales de colores de gráficos de barras o de gráficos circulares. El procedimiento general siguiente consiste en extraer las palabras del gráfico, luego, clasificarlas según estén en el eje X o el eje Y, si son valores, leyendas o etiquetas. Para la extracción de los datos, cada punto necesita representarse en un par de valores que representen el valor en el eje X y el valor en el eje Y. Por último, se deben extraer las curvas y asociarlas con las leyendas. En este trabajo excluían curvas de color negro o gris, por lo que recomiendan para próximos trabajos incluirlas.		(Sagnik Ray Choudhury et al., 2015)	Repositorio Pennsylvania State University, Estados Unidos
Figure and caption extraction from biomedical documents	Los artículos previos a este intentan identificar imágenes analizando la codificación y estructura de PDF y los complejos objetos gráficos incrustados. En imágenes biomédicas no está bien esto. Proponen, por ende, PDFigCapX. Acá primero separan texto de gráficos y luego utilizan layout information para extraer. El usuario final recibe la leyenda y la descripción de la imagen por aparte. Este trabajo en sí no extrae información de la imagen. El método obtiene mejores resultados en la extracción de las figuras y de		(Li, Jiang, & Shatkay, 2019)	Repositorio University of Delaware, Estados Unidos

	las leyendas de las mismas. (En lo que confiere al dataset de computer science). Tiene este servicio una página web sencilla que sirve como guía para el presente trabajo.		
Chart decoder: generating textual and numeric information from chart images automatically	Crean un sistema que implementa la decodificación de características visuales y recupera datos de imágenes de gráficos. Identifica y clasifica los gráficos -a través de aprendizaje profundo- en gráfico de barras, gráfico de línea, gráfico circular, gráfico de dispersión y en tabla de radar. Exponen la metodología más completa, y es la base principal del presente trabajo, pues es el único encontrado que trabaja también imágenes raster.	(Dai et al., 2018)	Journal of Visual Languages and Computing Received, China
VizioMetrics: mining the scientific visual literature	Se creó una plataforma que extrae información visual de la literatura científica y que la hace disponible para nuevas ideas de recuperación de información. Procesa la información, clasifica las figuras, organiza los resultados de una manera visible. Permite brindar ideas en lo que confiere al diseño de la interfaz final del sistema.	(Lee, 2017)	Repositorio, University of Washington, Estados Unidos
An architecture for multimodal information extraction from scholarly documents	Es un sistema completo de extracción de información de artículos científicos. Extrae todas las figuras de un artículo con una precisión del 81%. Luego, se detectan imágenes compuestas con alrededor de 82% de precisión. Clasifica imágenes no compuestas como gráficos de líneas o como gráficos de barras con un 84% de precisión. Clasifica el texto dentro de una imagen con un 90%. Extraen los gráficos en formato de vectores, lo que permite separar las curvas en gráficos de líneas con una precisión del 75%. Finalmente, se muestra un método supervisado para extraer entidades académicas del texto. Aporta al trabajo la herramienta para convertir los PDF a SVG, que es InkScape.	(S R Choudhury, 2017)	Repositorio The Pennsylvania State University, Estados Unidos

Nota: elaboración propia.

Cabe destacar que en la tabla 1 solo se incluyen antecedentes internacionales pues este es un tema relativamente nuevo en la literatura que tiene, por ende, escaso trabajo académico.

2.2.Marco teórico

2.2.1. Sistema inteligente

Los sistemas inteligentes son aquellos que presentan un comportamiento similar, en algún aspecto, a la inteligencia humana. Tienen la capacidad de representar, procesar y modificar de forma explícita conocimiento sobre un problema, y de mejorar su desempeño con la experiencia. Esto les permite resolver problemas complejos y multidisciplinares concretos determinando las acciones a tomar para alcanzar los objetivos propuestos y de forma automática, dando soporte a las decisiones de un experto. Los sistemas inteligentes estudian el desarrollo de agentes inteligentes o racionales. Un agente racional debe ser capaz de actuar para maximizar el mejor resultado esperado.

A continuación, se citan algunos de los sistemas inteligentes más usados en la actualidad: clasificación lineal, regresión logística, Naive Bayes, k-nn, árboles de decisión, support vector machines, random forest, redes neuronales, deep learning, aprendizaje no supervisado.

2.2.2. Extracción y procesamiento de figuras académicas

Figuras académicas son aquellas visualizaciones que se encuentran en artículos científicos, como por ejemplo bar charts (gráfico de barras), line charts (gráficos de líneas) (Boschen & Scherip, 2017). En la figura 1 se ejemplifican los distintos tipos:

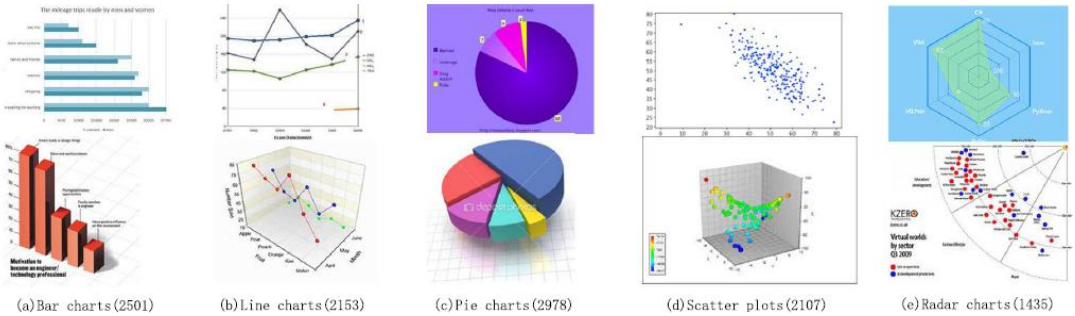

Figura 1. Diferentes ejemplos de figuras que se pueden encontrar en la literatura. (Dai et al., 2018)

Un enfoque realizable en la tarea de extraer información de gráficos es el de extraer el texto según el rol que cumple dentro de la imagen. En la figura 2 se muestra cómo se podría clasificar la potencial información a extraer.

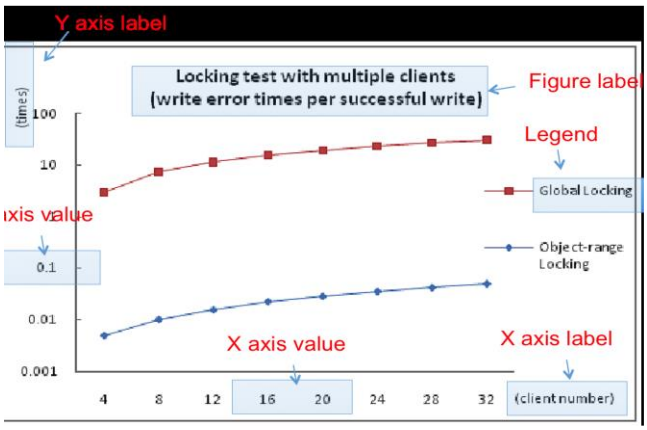

Figura 2. Diferentes tipos de roles que cumple el texto dentro de una gráfica. (Falk Böschen et al., 2018).

Las curvas en las gráficas se pueden clasificar en dos clases: puntos binarios o escala de grises, donde las curvas son dibujadas con pixeles negros o grises y son distinguidos por figuras. La segunda clase son los puntos de colores, que corresponden aproximadamente al 42% de los casos (Falk Böschen et al., 2018). Es en estas ocasiones, es donde extraer información es una tarea sencilla. Esto tiene unos desafíos, sin embargo: 1. Remover la estructura en cuadrícula y el fondo no blanco; 2. Identificar correctamente los colores y 3. La superposición de ciertas curvas.

Es muy beneficioso extraer las figuras en un formato de gráficos vectores (SVG). Pues permite lidiar mejor con curvas que se sobrepongan una sobre otra, o con fondos que aparentan ser blancos pero que no lo son. Por otro lado, convertir un PDF a una gráfica de vectores escalables (SVG) toma entre 5 y 6 segundos, mucho menos en comparación a los 50 – 60 segundos que tomaría si se hubiese rasterizado la imagen. (Sagnik Ray Choudhury, 2017). Se recomienda que, para construir sistemas de extracción de datos, la literatura debería enfocarse en gráficos vectoriales (SVG).

2.2.3. Clasificación de imágenes

La clasificación de imágenes se refiere a un proceso de visión por computador que puede clasificar una imagen de acuerdo con su contenido visual (Kaeli, Mistry, Schaa, & Zhang, 2015). Es una manera de describir el contenido de una imagen. El objetivo de la clasificación de imágenes es identificar la etiqueta de una imagen de entre un grupo de etiquetas predeterminadas (Foumani & Nickabadi, 2019), que, en este caso, serían los diferentes tipos de gráficas (lineales, de barras, circulares, etcétera). Existen técnicas que la suelen acompañar, como la anotación de imágenes. Esta se centra en describir el contenido de la imagen de entrada mediante un conjunto de palabras clave relacionadas. Asimismo, existe el subtitulado, que es la generación de una frase que describe el contenido observado en la imagen.

Por ejemplo, un algoritmo de clasificación de imágenes puede estar diseñado para decir si una imagen contiene o no una figura humana (Kaeli et al., 2015). Mientras que la detección de un objeto es trivial para los seres humanos, la clasificación robusta de las imágenes sigue siendo un reto en las aplicaciones de visión por computador.

2.2.4. Redes neuronales

Las redes neuronales son modelos simples del funcionamiento del sistema nervioso. Las unidades básicas son las neuronas, que generalmente se organizan en capas. Una red neuronal es un modelo simplificado que emula el modo en que el cerebro humano procesa la información. (IBM, s.f.). Las unidades de procesamiento se organizan en capas. Hay tres partes normalmente en una red neuronal: una capa de entrada, con unidades que representan los campos de entrada; una o varias capas ocultas; y una capa de salida, con una unidad o unidades que representa el campo o los campos de destino. Las unidades se conectan con fuerzas de conexión variables (o ponderaciones).

La red aprende examinando los registros de entrenamiento, generando una predicción para cada registro y realizando ajustes a las ponderaciones, a través de los datos de validación, cuando realiza una predicción incorrecta. Este proceso se repite muchas veces sobre todo el conjunto de datos (este número de veces se conoce como épocas) y la red sigue mejorando sus predicciones hasta haber alcanzado uno o varios criterios de parada (IBM, n.d.).

Al principio, todas las ponderaciones son aleatorias y las respuestas que resultan de la red son, posiblemente, disparatadas. La red aprende a través del entrenamiento. Continuamente se presentan a la red ejemplos para los que se conoce el resultado, y las respuestas que proporciona se comparan con los resultados conocidos. La información procedente de esta comparación se pasa hacia atrás a través de la red, cambiando las ponderaciones gradualmente. A medida que progresa el entrenamiento, la red se va haciendo cada vez más precisa en la replicación de resultados conocidos (IBM, s.f.). Una vez entrenada, la red se puede aplicar a casos futuros en los que se desconoce el resultado. Esto se conoce como transfer learning: existen redes preentreadas cuyo conocimiento se puede aplicar a otros problemas de clasificación de imágenes, por ejemplo.

Antes de comenzar el entrenamiento, es necesario establecer varios hiperparámetros de las redes neuronales, como el peso de las conexiones, el número de neuronas en la capa oculta, el número de características de entrada, la velocidad de aprendizaje, el impulso, las tasas de abandono y otros factores de regularización. La selección de estos parámetros determinará la tasa de éxito del modelo, pues afectan la capacidad y las características de aprendizaje del mismo (Sadiq, Faris, Al-Zoubi, Mirjalili, & Ghafoor, 2019). Aunque se puede abarcar desde la optimización, o simplemente, a través de ensayo y error, no existen reglas óptimas sobre cómo configurar exactamente estos parámetros.

2.2.5. Algoritmos evolutivos

La detección del texto que está en la figura se ha tratado con algoritmos evolutivos, mostrando muy buenos resultados en comparación a técnicas de predicción clásicas (Sagnik Ray Choudhury, 2017). Por ende, para este problema, puede ser necesario usar heurísticas o metaheurísticas que encuentren buenas soluciones en tiempo aceptable, como por ejemplo, los algoritmos evolutivos (Acuña & Arreche, 2014). Los algoritmos evolutivos son un conjunto de técnicas metaheurísticas que se resuelven problemas complejos y que están inspirados en la evolución natural. Utiliza mecanismos de selección, reproducción y técnicas para mantener la diversidad (Spears, 2000). Este algoritmo es iterativo y busca en cada paso mejorar las soluciones por medio de operadores de exploración y explotación, basado en un criterio predefinido a maximizar o minimizar. Se destacan cuatro etapas (Acuña & Arreche, 2014):

- Evaluación: para cada individuo de la población se determina un valor de aptitud en relación a su capacidad para resolver el problema.
- Selección: se eligen los individuos que sobrevivirán a la siguiente generación y sobre los cuales se aplicarán los operadores evolutivos.
- Operadores evolutivos: se aplican combinaciones entre individuos (cruzamiento) y modificaciones aleatorias de individuos (mutación). Los operadores generan nuevos individuos que sustituirán a los existentes en la población.
- Reemplazo: se produce un recambio generacional, sustituyendo a la antigua población por una nueva, que podría tener sobrevivientes de los anterior o solamente nuevos individuos generados en la etapa anterior.

Estas metaheurísticas son algoritmos aproximados destinados para problemas complejos que, por el sin número de operaciones combinatorias, hace muy costosa su solución computacional. Proporcionan, además, soluciones de alta calidad en un tiempo computacional

breve (Medina Nolasco, 2015). Heurístico per se es un procedimiento para que el que se tiene alto grado de confianza, con soluciones de alta calidad con un relativamente bajo coste computacional. Sin embargo, no se garantiza optimalidad. De hecho, heurístico está en contraposición a exacto (Melián & Pérez, 2003). Estos algoritmos están inspirados en otras ciencias como la física, en la evolución y en la biología (Medina Nolasco, 2015) y su aplicabilidad depende de las características del problema a resolver.

2.2.6. Visión por computador

Es el conjunto de técnicas y modelos que permiten la adquisición, procesamiento, análisis y explicación de cualquier tipo de información espacial del mundo real obtenida a través de imágenes digitales (Reinhard Klette, 2014). Consiste en analizar digitalmente imágenes por medio de un computador. Las etapas fundamentales son (Sisalima Ortega, 2018):

- Adquisición de una imagen: es usar el sensor que captura la luz emitida por una escena y almacenarlo en un archivo. Estos sensores están en cámaras de video, de fotografía digital o convencionales.
- Realce de una imagen: consiste en mejorar la apariencia visual de una imagen a través de técnicas que permiten mejorar contraste, brillo, eliminar distorsiones, falta de nitidez, etc.
- Procesamiento del color: consiste en la manipulación de la imagen a color, pues a color puede contener más o menos información que la imagen en nivel gris.
- Procesamiento morfológico: es aplicar una serie de operadores que permiten extraer los componentes útiles de una imagen, como los bordes, esqueletos, operaciones de llenado y lógicos, filtros, entre otros.
- Segmentación de una imagen: consiste en extraer de una imagen un objeto u objetos que son importantes o que se requieren para el proceso de representación y descripción.

- Representación y descripción: permite extraer las características de los objetos que fueron seleccionados en el proceso de segmentación, como bordes, cálculo de áreas, longitudes, direcciones, formas, etc.
- Reconocimiento: permite identificar, teniendo en cuenta un patrón, la información contenida en la imagen. Se identifican pixeles agrupados, se comparan, y estos se asocian a un objeto ya conocido.

2.3. Marco conceptual

2.3.1. Aprendizaje de máquina

El aprendizaje automático o aprendizaje automatizado o aprendizaje de máquinas (del inglés, machine learning) es el subcampo de las ciencias de la computación y una rama de la inteligencia artificial, cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan. Se dice que un agente aprende cuando su desempeño mejora con la experiencia; es decir, cuando la habilidad no estaba presente en su genotipo o rasgos de nacimiento. una

2.3.2. Aprendizaje no supervisado

Aprendizaje no supervisado es un método del aprendizaje de máquina donde un modelo es ajustado a las observaciones. Se distingue del aprendizaje supervisado por el hecho de que no hay un conocimiento a priori. En el aprendizaje no supervisado, un conjunto de datos de objetos de entrada es tratado. Así, el aprendizaje no supervisado típicamente trata los objetos de entrada como un conjunto de variables aleatorias, siendo construido un modelo de densidad para el conjunto de datos.

2.3.3. Aprendizaje supervisado

En aprendizaje automático y minería de datos, el aprendizaje supervisado es una técnica para deducir una función a partir de datos de entrenamiento. Los datos de entrenamiento consisten de pares de objetos (normalmente vectores): una componente del par son los datos de entrada y el otro, los resultados deseados. La salida de la función puede ser un valor numérico (como en los problemas de regresión) o una etiqueta de clase (como en los de clasificación).

2.3.4. Metadata

Son datos que describen otros datos. En general, un grupo de metadatos se refiere a un grupo de datos que describen el contenido informativo de un objeto al que se denomina recurso.

2.3.5. Datos raster

Un ráster consta de una matriz de celdas (o píxeles) organizadas en filas y columnas (o una cuadrícula) en la que cada celda contiene un valor que representa información, como la temperatura. Los rásteres son fotografías aéreas digitales, imágenes de satélite, imágenes digitales o incluso mapas escaneados.

2.3.6. SVG

Los archivos SVG (Scalable Vector Graphics), Gráficos Vectoriales Redimensionables, son un formato de gráficos vectoriales bidimensionales, tanto estáticos como animados, en formato XML. A diferencia, de los mapas de bits, las imágenes vectoriales pueden ser redimensionada, tanto como se quiera, sin pérdida de calidad de imagen. Además, las imágenes vectoriales, pesan menos megas que las imágenes estándar.

2.4.Marco contextual

El presente trabajo de investigación se desarrolla en el grupo de investigación Mindlab. Es un grupo categoría A1 en Colciencias, dirigido por el profesor titular Fabio Augusto Gonzalez Osorio. Fue creado en el año 2001 y es uno de los mejores en aprendizaje de máquina del país. Consta de tres líneas de investigación: aprendizaje de máquina, recuperación de imágenes e imágenes médicas. Tiene tres objetivos:

- Impulsar el desarrollo de la ingeniería y ciencias biomédicas adelantando proyectos de investigación e innovación que contribuyan a la generación de nuevo conocimiento en el área.
- Promover la formación de alto nivel mediante la participación activa en programas de pregrado y postgrado y la vinculación de estudiantes a las líneas de investigación del grupo.
- Contribuir con el desarrollo del país, mediante el desarrollo de proyectos de I+D de alto impacto en el sector salud.

El grupo pertenece a la Universidad Nacional de Colombia sede Bogotá. El 22 de septiembre de 1867, mediante la Ley 66 expedida por el Congreso es oficialmente fundada como tal. Entre 1903 y 1940 se crearon en la Universidad más de 20 carreras, entre las que se encuentran: Arquitectura, Enfermería, Farmacia, Ingeniería Química, Medicina Veterinaria, Odontología y Química. A finales de la década de los 60 se les dio impulso a los programas de maestría en la Universidad Nacional y en el país. Los primeros programas, a nivel de maestría fueron creados entre 1967 y 1973 y en 1986 abrieron sus puertas los primeros programas doctorales del país en las áreas de Física y Matemáticas.

La Universidad Nacional de Colombia luego de 140 años de existencia, cuenta con 94 programas de pregrado, 86 especializaciones, 38 especialidades del área de la salud, 133 maestrías y 51 programas de doctorado.

2.5.Marco legal

Tabla 2
Referencias legales

Nombre	Definición	Fuente
Ley 23 de 1982 y la Decisión Andina 351 de 1993	Regulación de los derechos de autor. El derecho de autor protege como obras independientes, las traducciones, adaptaciones, arreglos musicales y demás transformaciones, sin perjuicio de los derechos de autor de las obras originales".	Congreso de la República
Ley de software libre en Colombia	Ley 11723: es una ley compuesta por 89 artículos, sancionada en 1933 (y todavía vigente), conocida como "Ley de Propiedad Intelectual" o también como "Ley de Propiedad Científica, Literaria y Artística". Esta ley regula todo lo referente a derecho de propiedad de una obra artística, científica o literaria, derechos de coautor, enajenación o cesión de una obra, licencias, etc. Además, establece sanciones tanto pecuniarias (multa) como privativas de la libertad (prisión) a quienes violen sus normas.	Congreso de la República

2.6.Delimitación espacial

La presente investigación se lleva a cabo en la ciudad de Bogotá, en las instalaciones de la Universidad Nacional de Colombia, específicamente, en el laboratorio del grupo de investigación Mindlab. Este laboratorio es el salón 205 de las Aulas de Ingeniería.

2.7.Delimitación temporal

El presente trabajo está contemplado para realizarse en 14 semanas. Siendo la fecha límite el 29 de noviembre del año 2019 y la fecha de inicio el 9 de septiembre.

3. Diseño Metodológico

3.1.Tipo de investigación

De acuerdo con los objetivos planteados para el desarrollo del proyecto, y según (Hernández Sampieri, Fernández Collado, & Baptista Lucio, 2014) la investigación se consideró de tipo exploratoria-descriptiva, pues se caracterizaron los artículos científicos de la ciencia de los materiales, así como las gráficas que en su mayoría componen esta área. Del mismo modo, se investigó un tema poco trabajado por la literatura, como lo es la extracción de información de gráficas académicas.

3.2.Población y muestra

3.2.1. Población

Se puede definir población como el "conjunto integrado por todas las mediciones u observaciones del universo de interés en la investigación." (Parra, 2003). Según lo anteriormente mencionado, la población de estudio para este trabajo fueron las figuras de todos los artículos científicos del área de la ciencia de los materiales.

3.2.2. Muestra

La muestra de este trabajo fueron las 10.053 figuras de los artículos científicos del área de la ciencia de los materiales que se pudieron extraer correctamente.

3.3.Instrumentos o técnicas para la recolección de información

3.3.1. Información primaria

Es toda aquella información ya sea de forma oral o escrita que pueden obtener las personas que realizan la investigación (Hernández Sampieri et al., 2014). En este caso se utilizó el análisis documental; específicamente, de los artículos científicos recolectados según la muestra

establecida.

3.3.2. Información secundaria

Para poder recolectar la información necesaria para este proyecto y así, poder realizar su respectivo análisis; se optó por utilizar librerías de Python, específicamente, Numpy y Pandas. Esta fue necesaria para organizar la información extraída de las imágenes. Asimismo, se usó el formato JSON para almacenar el texto de la imagen, su respectivo rol, y los valores extraídos de la gráfica para cada eje.

3.4. Análisis de la información

Se usó Python y sus librerías de aprendizaje de máquina para tratar la información. A continuación, se ejemplifica en la figura 3 un diagrama de flujo de la extracción y tratamiento de la información.

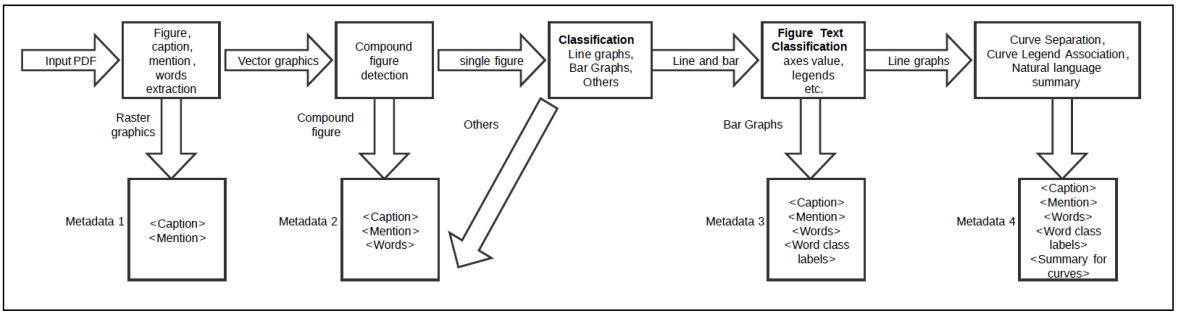

Figura 3. Diagrama de flujo para la extracción y procesamiento de los datos. (Sagnik Ray Choudhury, 2017).

4. Resultados y análisis

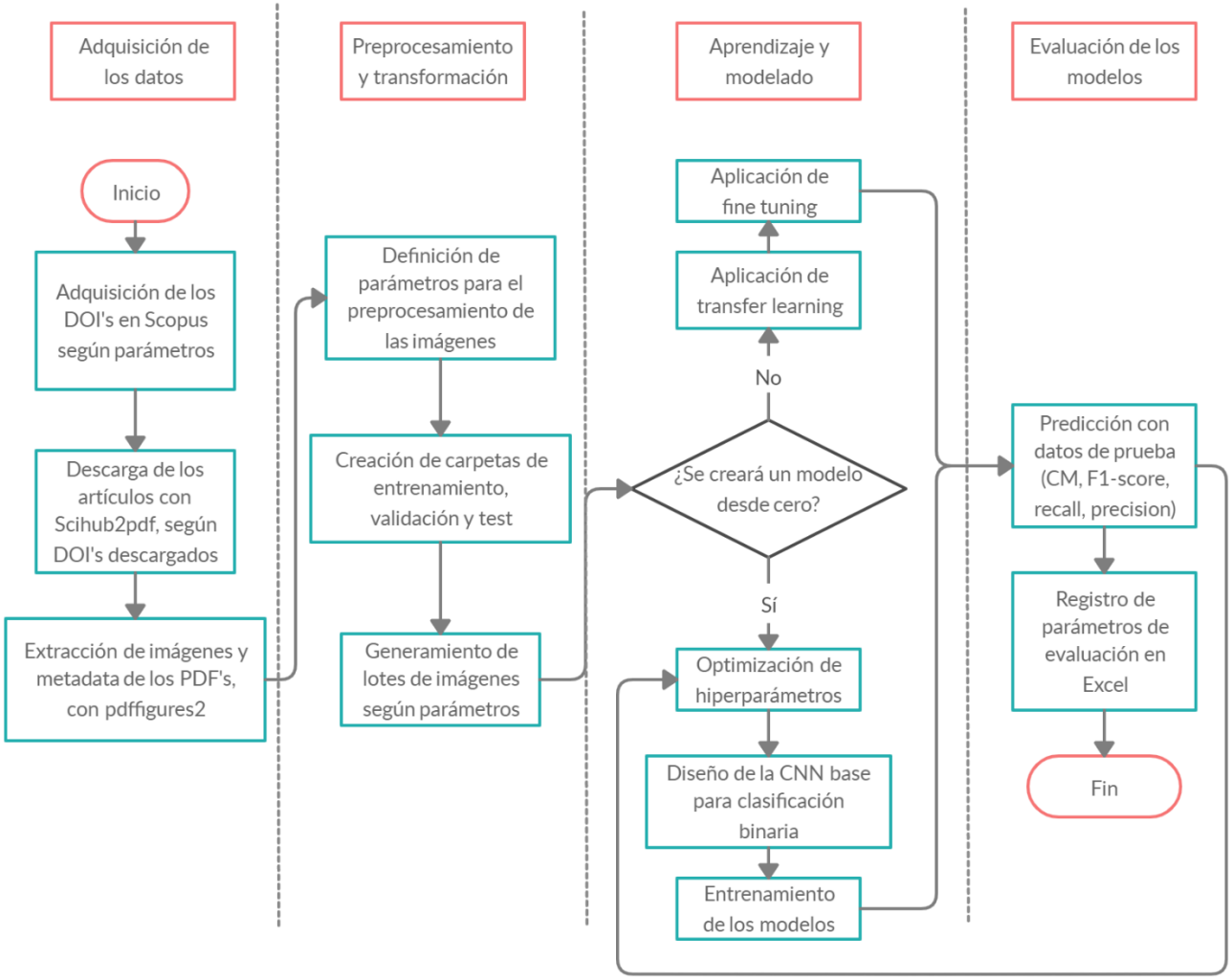

Figura 4. Diagrama de la primera parte del proceso

5. Explicación del proceso

5.1. Adquisición de los datos

5.1.1. Adquisición de los DOI's

Para obtener los PDF's lo primero que se hizo fue obtener una lista de los 2000 más relevantes artículos científicos en el área de las ciencias de los materiales en Scopus. Esta lista contenía los DOI de cada uno de los artículos. Se utilizó Scopus, precisamente, porque era la única base de datos que permitía exportar dicha cantidad en un archivo de texto. La ecuación de búsqueda para los artículos fue la siguiente: ("biodegradable film" OR "biodegradable composite" OR "nanocomposite film" OR "nanocomposite" OR "bionanocomposite" OR "biocomposite"). Los resultados fueron limitados a las áreas de las ciencias de los materiales, ingeniería, química e ingeniería química. También se filtraron documentos que solo fueran artículos, conference papers o revisiones. Finalmente, esta búsqueda se ordenó según la relevancia del artículo. Los resultados arrojaron un total de 104.457 documentos. Como se mencionó anteriormente, se escogieron los 2000 artículos más relevantes (según el filtro de la web de Scopus). Cabe destacar, que el énfasis en la producción y caracterización de cualquier tipo de compuesto biodegradable se debe a que es una de las pocas áreas de la ingeniería industrial de la que es factible encontrar grandes cantidades de datos fácilmente.

5.1.2. Descarga de los artículos PDF

Una vez obtenido este archivo de texto de Scopus, se descargaron los artículos científicos de la plataforma de datos abierta, Libgen. Se usó la herramienta Scihub2pdf para esto. Es un paquete que se corre desde la terminal, en Linux. De los 2000 artículos, la herramienta descargó correctamente 1673 PDF's. Se tuvieron dificultades, pues Libgen bloqueaba la IP según un número de descargas, por lo que había que esperar un día para continuar con la descarga de los artículos.

5.1.3. Extracción de imágenes y metadata de los PDF

Para esta actividad se usó pdffigures2. Es un software escrito en Scala que permite extraer las imágenes en formato PNG, el título y cualquier texto que esté alrededor de las mismas. Esta metadata se guarda en un archivo JSON. Es necesario correrlo en Ubuntu e instalar Java y SBT. Un tutorial completo fue realizado por el autor y está disponible en GitHub, en este link. Una vez instalado lo necesario, este software almacena las imágenes en una carpeta, y su metadata en otra. Ocurrieron ciertos problemas en la ejecución. Por ejemplo, paraba cada vez que no podía extraer las imágenes. En total, se extrajeron 10.053 imágenes. Se pueden vincular las imágenes y su metadata a través del nombre del artículo del que se fue extraído.

5.2. Preprocesamiento y transformación

5.2.1. Definición de parámetros para el procesamiento de las imágenes

Se definió que para el entrenamiento y la validación se usaría el batch generator, que pudiese generar lotes de imágenes con las siguientes características:

- Longitud y altura de cada imagen de 224 pixeles.
- El tamaño del batch de 32.
- Para el generador de imágenes de entrenamiento, se dividió cada imagen sobre 255 para estandarizar y escalar; y se establecieron unos parámetros para aumentar los datos: shear_range = 0.2; zoom_range = 0.2; y horizontal_flip.
- Para el generador de imágenes de validación solamente se dividió cada imagen sobre 255.
- Para ambos generadores el class_mode se estableció como binario.
- De las 10.053 imágenes extraídas, se estableció que el 70% fueran para entrenamiento, 15% para validación y 15% para test.

5.2.2. Creación de carpetas de entrenamiento, validación y test

Las imágenes se clasificaron como lineares (linear) o no lineales (nonlinear) de forma manual. Se encontraron, luego de la clasificación manual, 4.320 imágenes lineales (43%) y 5.733 imágenes no lineales (57%). Luego, según la repartición de las imágenes establecida en el punto anterior, se crearon las carpetas y se movieron las imágenes usando un código escrito en Python que aleatoriamente movía la cantidad de imágenes establecida a la carpeta correspondiente. Se crearon 3 carpetas principales: “train”, “val” y “test”. Cada una de estas carpetas contiene dos carpetas (las dos clases): “linear” y “nonlinear”. Por ende, cada una de las carpetas tiene la siguiente cantidad de imágenes:

```
/home/jovyan/dataset/
0
/home/jovyan/dataset/val
0
/home/jovyan/dataset/val/linear
648
/home/jovyan/dataset/val/nonlinear
860
/home/jovyan/dataset/test
0
/home/jovyan/dataset/test/linear
648
/home/jovyan/dataset/test/nonlinear
860
/home/jovyan/dataset/train
0
/home/jovyan/dataset/train/linear
3024
/home/jovyan/dataset/train/nonlinear
4013
```

Figura 5. Descripción de carpetas de entrenamiento, validación y test

De esta forma, los datos totales de entrenamiento son 7037 imágenes, los de validación y test, 1508.

5.2.3. Creación de carpetas de entrenamiento, validación y test

Los lotes de imágenes se generaron con la herramienta de procesamiento de Keras: Image Data Generator, usando los parámetros establecidos anteriormente.

6. Modelado y aprendizaje

6.1. Modelos generados desde cero

6.1.1. Exploración/optimización de hiperparámetros

La arquitectura básica de la red fue hecha en Keras, usando el lenguaje Python. La primera forma en la que se abarcó la aplicación de redes neuronales convolucionales consistió en generar modelos a través de ciclos for. Cada posible valor que podía tomar un hiperparámetro se guardó una lista, sobre la que luego se iteraba. La arquitectura básica de la red está compuesta de la siguiente forma:

1. Capa inicial: padding ="same", kernel_regularizer=regularizers.l1_l2(0.01)
2. Segunda capa inicial: padding ="same"
3. Primer bloque:
    a. Capa convolucional
    b. MaxPooling
    c. Dropout
4. Segundo bloque:
    a. Capa convolucional
    b. MaxPooling
    c. Dropout
    d. Flatten
5. Bloque de capas densas
6. Capa de salida

A continuación, están los hiperparámetros que se exploraban:

dense_layers = [1,2] #Número de capas densas

layer_sizes = [64,256] #Número de nodos de las capas densas

conv_layers = [1,2] #Número de capas convolucionales

lr = [1e-3, 1e-4, 1e-5, 1e-6] #Learning rate

drop_rate = [0.4, 0.5, 0.6, 0.7] #Dropout

batch_norm = Si se añadía o no normalización.

Por otro lado, los siguientes son los hiperparámetros que permanecieron constantes:

filtrosConv1 = 32 #Filtros del primer bloque de capas convolucionales

filtrosConv2 = 64 #Filtros del segundo bloque de capas convolucionales

tamano_filtro1 = (3, 3) #Tamaño del filtro para el primer bloque de capas

tamano_filtro2 = (2, 2) #Tamaño del filtro para el segundo bloque de capas

tamano_pool = (2, 2) #Tamaño del pool para todas las capas

función de activación = relu

A medida que se terminaban experimentos, se ajustaba el rango de valores de cada hiperparámetro, para obtener mejores resultados. Estos experimentos se corrían durante 20 épocas. El optimizador que se usó fue Adam. Se añadió un checkpoint de Keras que guardaba el mejor modelo, un early stopping y se reducía el learning rate a través de ReduceLROnPlateau. Asimismo, el log de cada modelo se guardó para visualizarse en Tensorbard. Se corrieron 122 modelos. Los

hiperparámetros de los mejores modelos se usaron para luego generar 8 modelos, con 60 épocas, que se probaron en los datos de test.

6.2. Aplicación de transfer learning y fine tuning

Para este punto, se usaron 4 modelos preentrenados disponibles en Keras: Inception, InceptionResNet, MobileNet y VGG19. Primero, se importaron estos con los pesos de Imagenet, sin incluir el tope de la capa (las que realizan la clasificación) e introduciendo la dimensión de entrada, que para el caso de InceptionResNet tuvo que aumentar de 224, 224,3 a 299, 299, 3, pues el modelo solo así lo permitía. A estos modelos importados se les congelaron las capas base, y se le añadieron capas personalizadas que sí se entrenarían. Dependiendo de cada modelo, se añadían distintas arquitecturas. La última capa, sin embargo, siempre fue una capa densa, con un nodo, y con la función de activación sigmoide.

Luego, usando los datos de entrenamiento ya generados, y con la misma configuración de compilación y de *fit* que en el punto 4.1, se procedía a correr el modelo durante 50 épocas, y con una taza de aprendizaje de 0.001.

Una vez finalizado, se descongelaban las capas previamente congeladas y se disminuía la tasa de aprendizaje a 0.00001. Se aumentó también el número de épocas a 70. Esto en todos los casos permitió aumentar considerablemente las métricas de cada modelo. Por cada arquitectura se generaban dos modelos: uno, con la mayor parte de la red congelada, y el segundo, con el modelo generado después de descongelar.

7. Evaluación de los modelos

En lo que confiere a la evaluación de los modelos, se generó un notebook a manera de plantilla, donde todos los mejores modelos (en formato .h5) se evaluaban. Para esto, se creó una función en Python que identificara la carpeta, que recibía las clases y el tamaño al que se trataría la imagen, es decir, 224 x 224 pixeles. Esta función unía el directorio, la clase, y el nombre de la imagen. A cada imagen, según el nombre de la carpeta, le asignaba un 0 o un 1 como etiqueta. De esta forma, cada imagen leída se añadía a una lista. Esta lista luego se separó en dos, para tener en dos listas diferentes las imágenes y las etiquetas. Las imágenes se escalaron entre 0 y 1. Ambas listas, al finalizar, se convertían a arrays, con Numpy.

Luego, se cargaba el modelo, y con model.evaluate y model.evalue_generator se generaron la pérdida y la precisión en las 1508 imágenes del dataset de test. Luego, a través de model.predict, se creó la matriz de confusión, que a su vez, permitía obtener las 3 métricas restantes: precision, recall y F1-score.

7.1. Resultados

En lo que confiere a la exploración de hiperparámetros, se corrieron 122 modelos con diferentes configuraciones y arquitecturas. En la siguiente tabla, a modo de seleccionar un rango de hiperparámetros, se escogen los mejores 10 y mejores 20 modelos, según el validation accuracy.

Tabla 3.
Resultados de la exploración de hiperparámetros

	Promedio de los mejores de 10 modelos (de entrenamiento)	Promedio de los mejores 20 modelos (de entrenamiento)
Capas_conv	1,2	1,35
Nodos_dense	204,8	182,4
Capas_dense	1,1	1,15
Learning_rate	0,00016	0,00037
Dropout	0,49	0,51
Batch_norm (número de veces añadida)	1,77	1,1
Epochs	35	30,5
Batch_size	32	32
Train_accuracy	0,84506	0,83415

Train_loss	0,93265	0,98511
Val_accuracy	0,833055	0,8259225
Val_loss	1,13708	1,18564

Con estos rangos de valores se propone en un futuro trabajo realizar optimización de hiperparámetros a través de Grid Search, Random Search u optimización Bayesiana. Sin embargo, dado que no es necesario que el modelo final tenga una baja cantidad de parámetros (y por ende, que sea liviano), se procede a experimentar con transfer learning y con fine tuning, obteniendo mejores resultados, tal y como se verá a continuación. Aun así, con base a los rangos de la anterior tabla, se generaron 8 modelos desde cero (scratch), para ver cómo se comportaban en los datos de test. En la siguiente tabla se aprecian los resultados tanta para arquitecturas pre-entrenadas (con fine tuning) y no pre-entrenadas (desde cero).

Tabla 4.
Comparación de métricas para modelos pre-entrenados y para modelos creados desde cero

Métrica	Promedio de arquitecturas pre-entrenadas	Promedio de arquitecturas creadas desde cero
Confusion_matrix(1,1) (era lineal y se predijo lineal)	616,875	583,125
Confusion_matrix(1,2) (era lineal y se predijo no lineal)	25,125	64,875
Confusion_matrix(2,1) (era no lineal y se predijo lineal)	93,125	146,75
Confusion_matrix(2,2) (era no lineal y se predijo no lineal)	753,875	713,25
Test_loss	0,343797661	1,07954
Test_accuracy	0,920313463	0.8596625
Test_loss_eval_gen	0,283423249	1,333538953
Test_acc_eval_gen	0,920378986	0,864029261
Promedio_loss (entre test y eval_gen)	0,313610455	1,206538
Promedio_acc (entre test y eval_gen)	0,920346224	0,861845881
F1-score	0,91875	0,86
Precision	0,925	0,87
Recall	0,92375	0,86

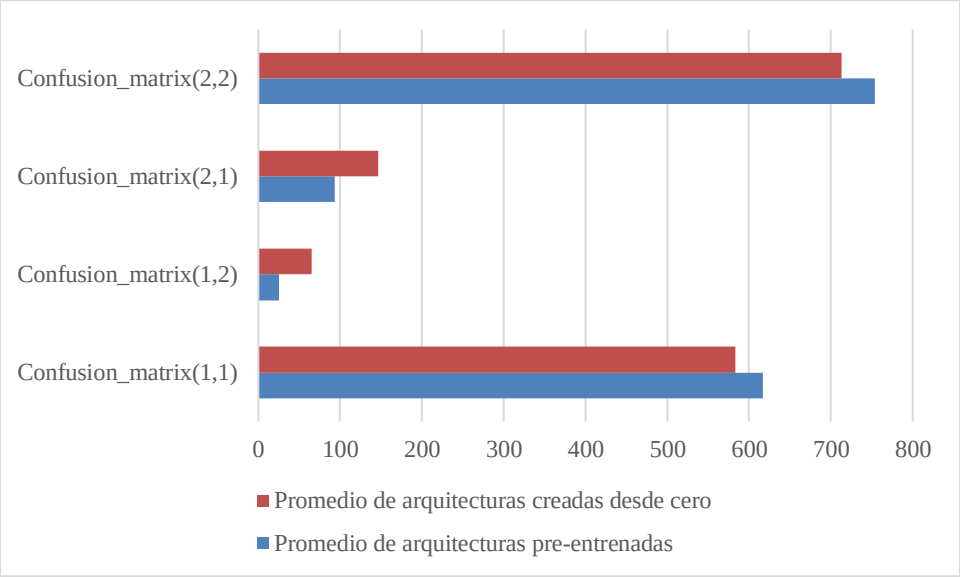

Figura 6. Comparación de matrices de confusión para modelos scratch y pre-entrenados

En la matriz de confusión, el (1,1) representa las veces que se era 0 y que se predijo 0 (correctamente). Donde 0 representa las figuras lineales y 1 las figuras no lineales. Con base a lo reportado en la tabla 2, el porcentaje de error en las predicciones para la clase de figuras lineales fue de 4,0729%, mientras que para las figuras no lineales fue del 12,35%. Con esto se puede concluir que las redes identifican mejor una imagen lineal que una imagen no lineal. Esto tiene sentido, pues una imagen no lineal puede ser desde una gráfica de barras, pasando por una fotografía, hasta una ecuación química.

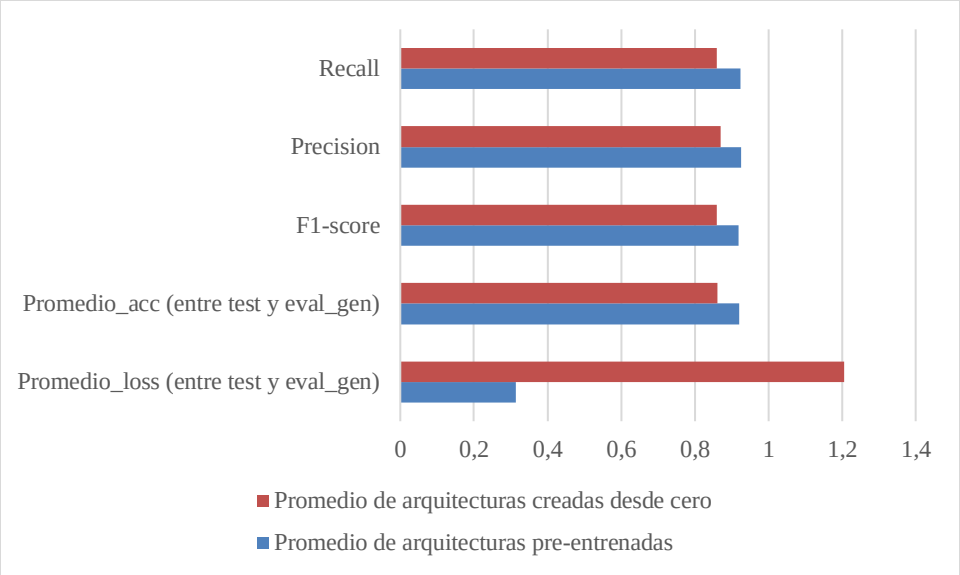

Figura 7. Comparación de métricas para modelos pre-entrados y creados desde cero

Con base en la figura 2 se puede apreciar como el uso de transfer learning y fine tuning, influyó en la mejora de las métricas del modelo. Esta mejora se ve en la pérdida, donde, en los modelos creados desde cero, esta fue 4 veces mayor que en modelos pre-entrenados. El mejor resultado, en términos de test accuracy y test loss (en ambas es el mejor), es el modelo realizado con Inception. En aquellos modelos realizados desde cero (scratch), el mejor modelo, también según test accuracy y según test loss, fue "KK1-1-conv-256-nodes-1-dense-0,0004-learning-best". A continuación, las métricas para cada uno de estos dos modelos:

Tabla 5.
Métricas de los mejores modelos obtenidos (pre-entrenados y desde cero)

Tipo de modelo	Mejor modelo con fine tuning	Mejor modelo scratch
Nombre	TT-Inception2_transfer-lr-0,001-epochs-best	KK1-1-conv-256-nodes-1-dense-0,0004-learning-best
Confusion_matrix(1,1)	612	610
Confusion_matrix(1,2)	27	38
Confusion_matrix(2,1)	56	135

Confusion_matrix(2,2)	785	725
Test_loss	0,0828	0,5171
Test_accuracy	0,9439	0,8853
Test_loss_eval_gen	0,158086984	0,637608961
Test_acc_eval_gen	0,94547874	0,8843085
Promedio_loss (entre test y eval_gen)	0,120443492	0,57735448
Promedio_acc (entre test y eval_gen)	0,94468937	0,88480425
F1-score	0,94	0,89
Precision	0,94	0,89
Recall	0,94	0,89

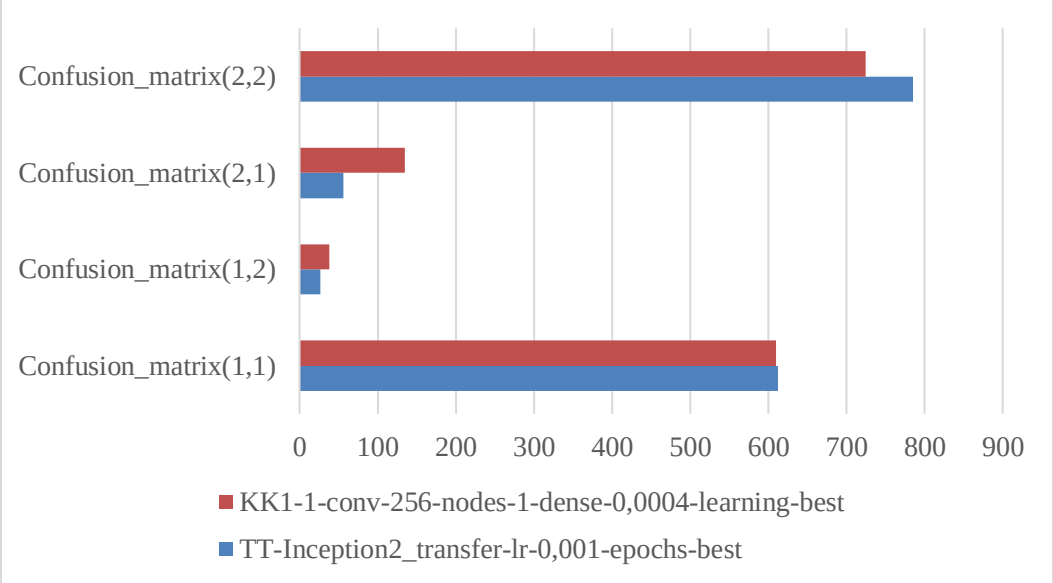

Figura 8. Análisis de las matrices de confusión de los dos mejores modelos

La figura 3 permite concluir que la principal ventaja del modelo pre-entrenado (en azul) y el modelo creado desde cero (en rojo) es que el primero se desempeña mejor en la difícil tarea de detectar imágenes no lineales: el modelo creado desde cero confunde muchas veces una imagen no lineal, con una lineal. En cambio, si se analizan las imágenes lineales, ambos modelos se desempeñan de forma muy similar.

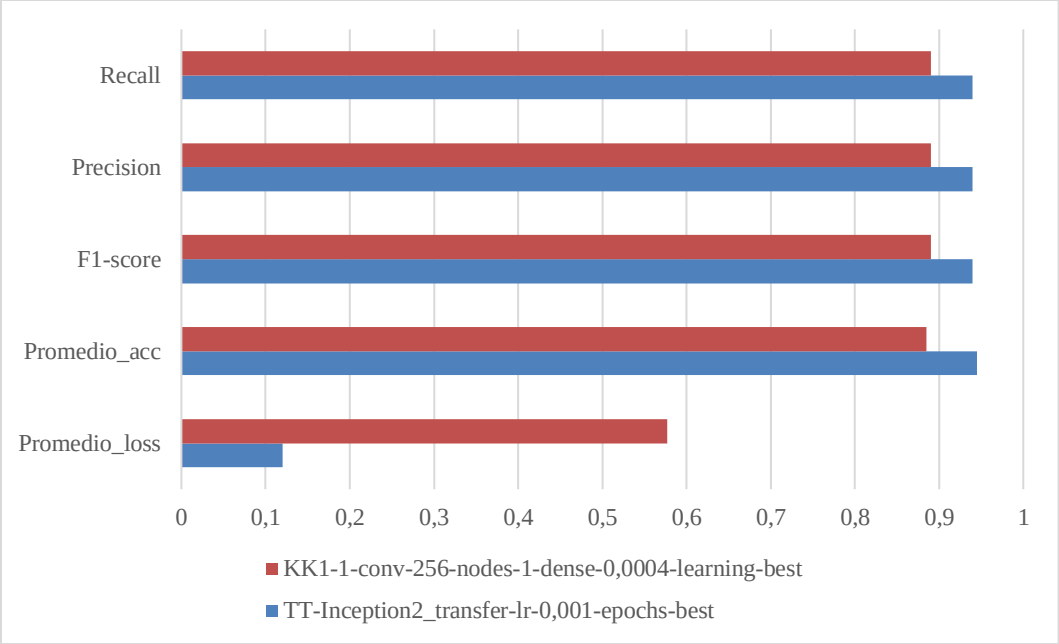

Figura 9. Análisis de las métricas obtenidas en los dos mejores modelos

La figura 4 permite, de nuevo, concluir cómo la diferencia entre los dos modelos radica en los valores de pérdida que se obtienen al correr los mismos en los datos de test.

Tabla 6.
Comparación del modelo generado con el estado del arte

Trabajo	Loss	Accuracy	F1-score	Precision	Recall
Presente trabajo	0,12044	94,46%	94	94	94
(Sagnik Ray Choudhury et al., 2015)	-	80%	-	-	-
(Dai et al., 2018)*	-	99%	-	-	-
(S R Choudhury, 2017)	-	-	84	84	85
(Sagnik Ray Choudhury & Giles, 2015)	-	85%	-	-	-

*Este trabajo entrena su modelo con imágenes descargadas del buscador de Google, que de por sí, ya las clasifica previamente. Por ende, puede que exista un sesgo en el resultado reportado en el artículo.

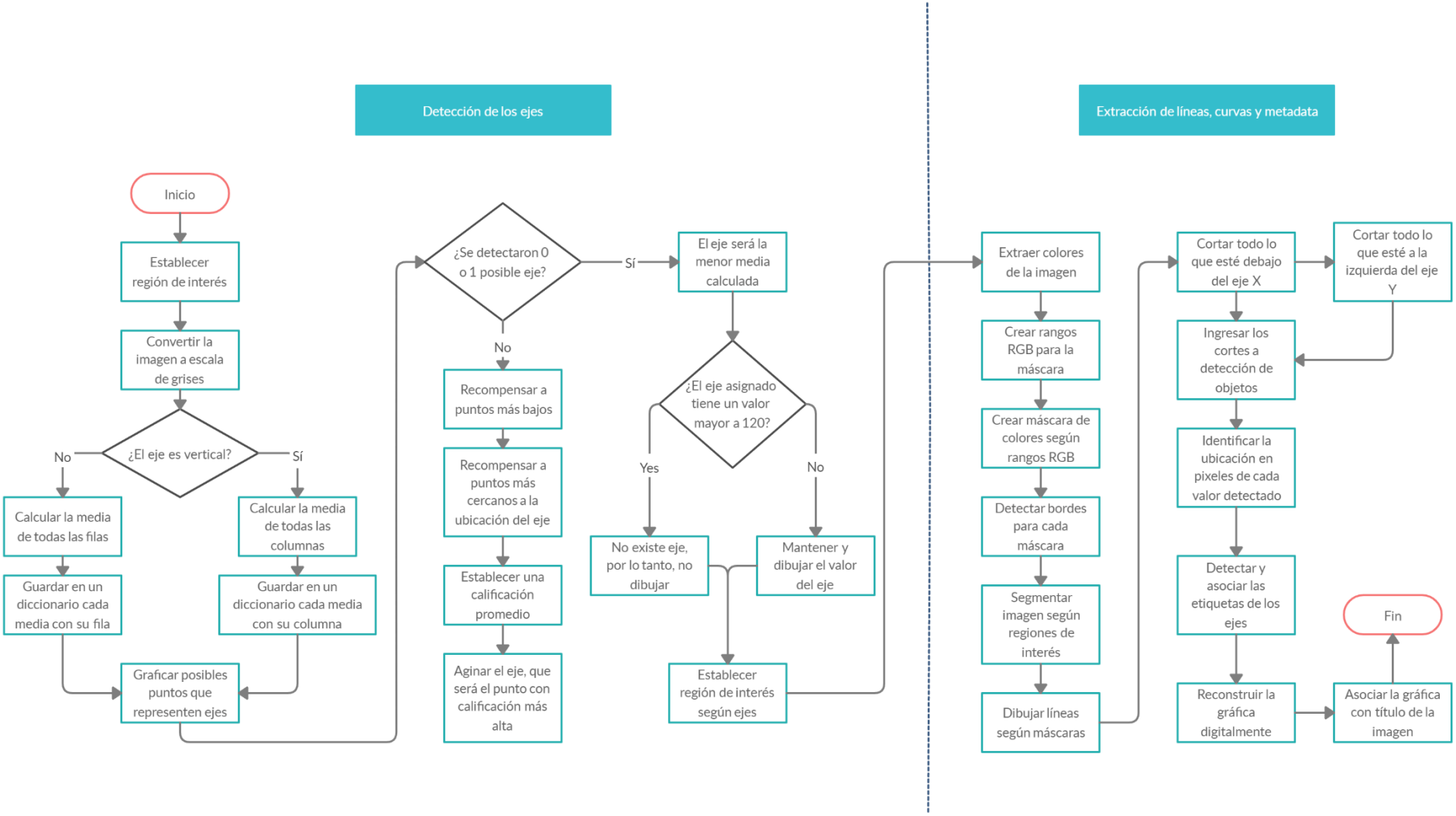

Figura 10. Descripción general del proceso de extracción de información de las gráficas

8. Detección de los ejes y bordes

Lo primero que se debe realizar para poder extraer correctamente la información de las gráficas es detectar las líneas verticales y horizontales que están en las mismas. Cabe destacar, que adicionales a los ejes X y Y, obligatorios, algunas imágenes incluyen un cuadrado completo, tal como se aprecia en la figura 6.

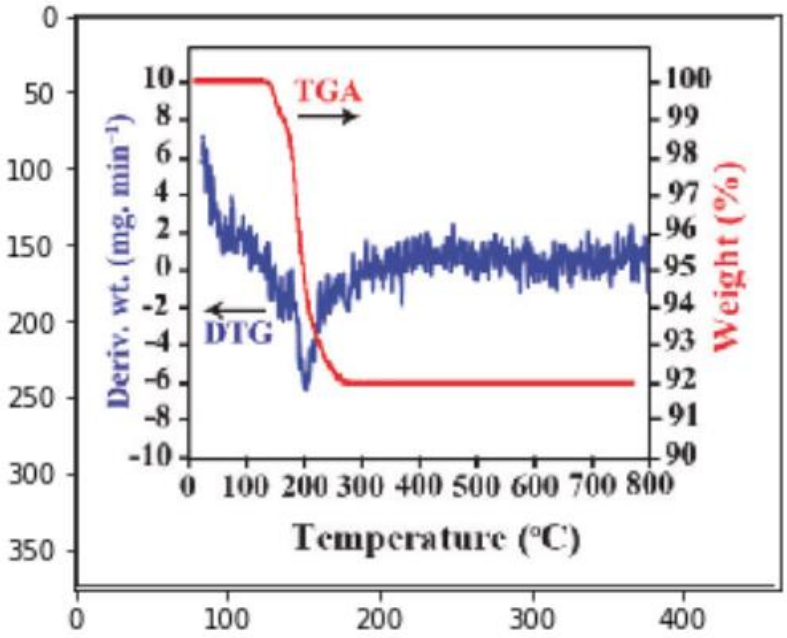

Figura 11. Ejemplo de gráfica que tiene un cuadrado alrededor de las curvas

Para la obtención de los ejes lo que se hizo fue establecer un área de interés. Esta asume que cada eje está entre el borde de la imagen y entre 1/3 de la misma. Luego de establecer el área, se convirtió la imagen a escala de grises, para poder analizar fila por fila o columna por columna el tamaño de los pixeles. Es decir, se obtuvo el promedio de cada fila o de cada columna de la región seleccionada, para luego graficarla a través de un histograma: donde hubiese una fila o columna con un valor extremadamente bajo, era donde estaba ubicada algún eje del cuadrado. (Un valor bajo representa una línea negra, que es en la gran mayoría de los casos el color de los ejes.

La representación del histograma se puede apreciar en la figura 7. Como se puede ver, el valle de la gráfica representa el eje X, en este caso, pues es el análisis de las filas.

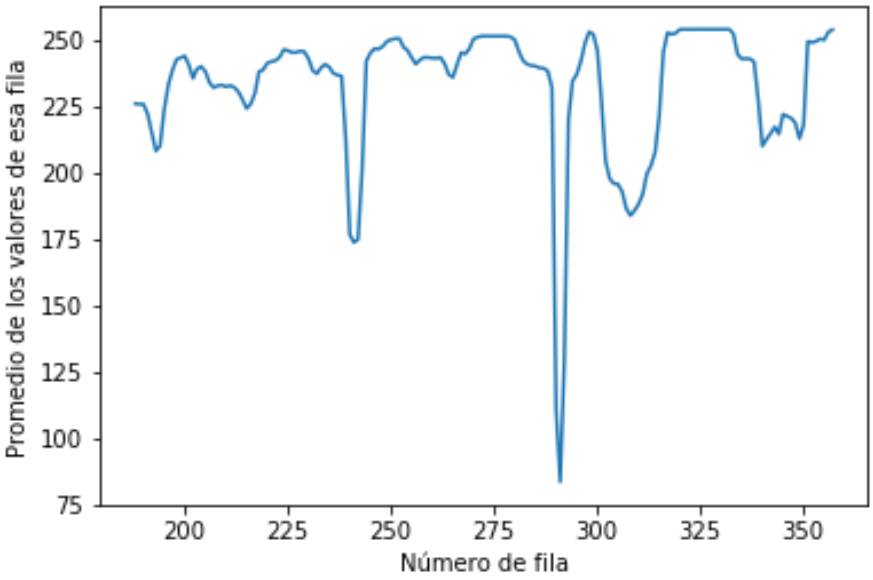

Figura 12. Representación de los valores de todas las filas de una gráfica

Cada que se realizaba un cálculo de media se guardaba en un diccionario con su respectiva fila o columna. Luego, se procedió a invertir la lista que contenía los promedios, con el fin de poder graficar y a través de la función find_peaks detectar los promedios más grandes, que, en realidad, son los más bajos, es decir, donde hay grandes cantidades de pixeles negros (posibles ejes). Si se detectaba un punto o ninguno, se asignaba como eje el menor valor la lista que contenía los promedios, siempre y cuando este valor fuera menor a 100. Cuando un valor fuese mayor a 100, significaba que no era un eje, o que tal vez estaba en otro color. En la gráfica 8 se ve el resultado de los posibles ejes detectados.

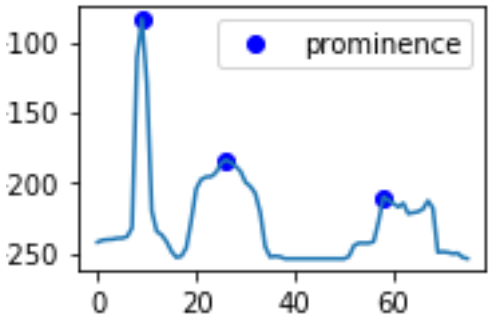

Figura 13. Histograma invertido con el fin de detectar los promedios más grandes

Por otro lado, si se detectaba más de un eje, como en la figura 8 -por ejemplo- el procedimiento era diferente. Se generó una calificación a cada potencial punto, para que al final, se pudiese elegir como eje a aquel punto con mayor puntuación. Este puntaje fue el promedio de dos más: el primero recompensaba ser un punto alto (o pequeño, si la matriz no se había invertido), mientras que el segundo recompensaba más a aquel punto que estuviese más cerca al eje en cuestión; es decir, si por ejemplo en la figura 8 se estuviese detectando el eje Y vertical, el primer punto, de izquierda a derecha, tendría el primer puesto, pues es el que está más cerca a la izquierda y -a su vez- es el más alto de todos. La altura corresponde al 60% de la calificación, mientras que la posición corresponde al 40% restante.

Este sistema permite, que -por ejemplo- el sistema sea robusto cuando se tope con una gráfica que tenga grid, o cuadrícula, ya sea vertical u horizontal. Por ejemplo, en la figura 9, el sistema ha detectado dos puntos, pues -como se ve en la figura 10-, la gráfica tiene cuadrículas horizontales. En este caso, se está detectando el borde superior de la imagen. Sin el sistema de calificación, el sistema asignaría el punto de la derecha como el eje, por ser el mayor, así sea por una leve diferencia. Acá es donde tener en cuenta la posición interviene en la elección: al estar el punto izquierdo mucho más cerca de la posición ideal que el punto de la derecha, se le asigna una calificación mayor, que termina escogiéndolo, correctamente, como el eje.

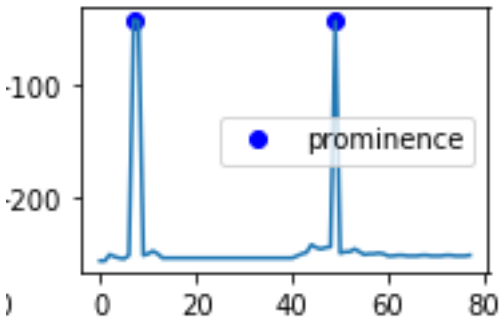

Figura 14. Representación de gráfica con cuadrículas horizontales

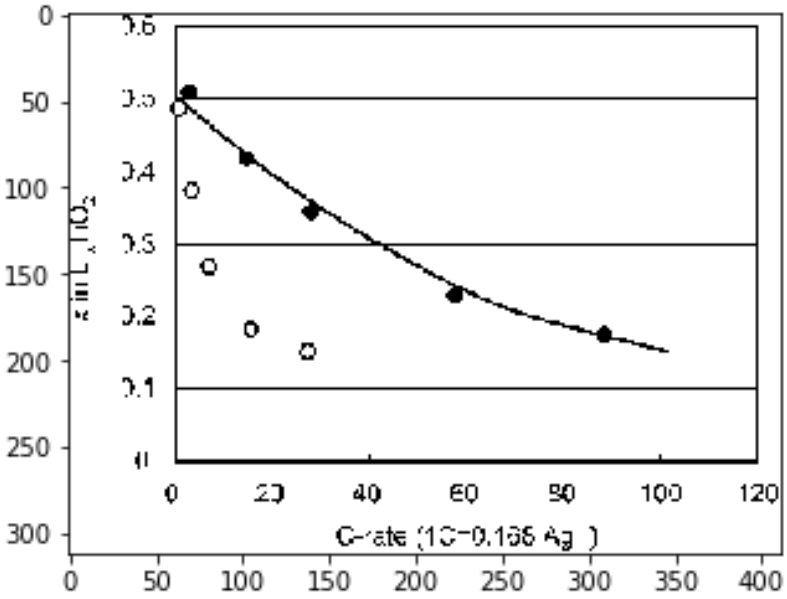

Figura 15. Imagen con cuadrículas horizontales

A continuación, se aprecia el resultado de la identificación de los ejes X, Y y de los bordes de la parte derecha y superior de la imagen. Esta detección permite que luego el algoritmo de extracción de líneas se concentre en el área que debe, y, además, permite extraer los valores de los ejes X, Y, así como sus leyendas.

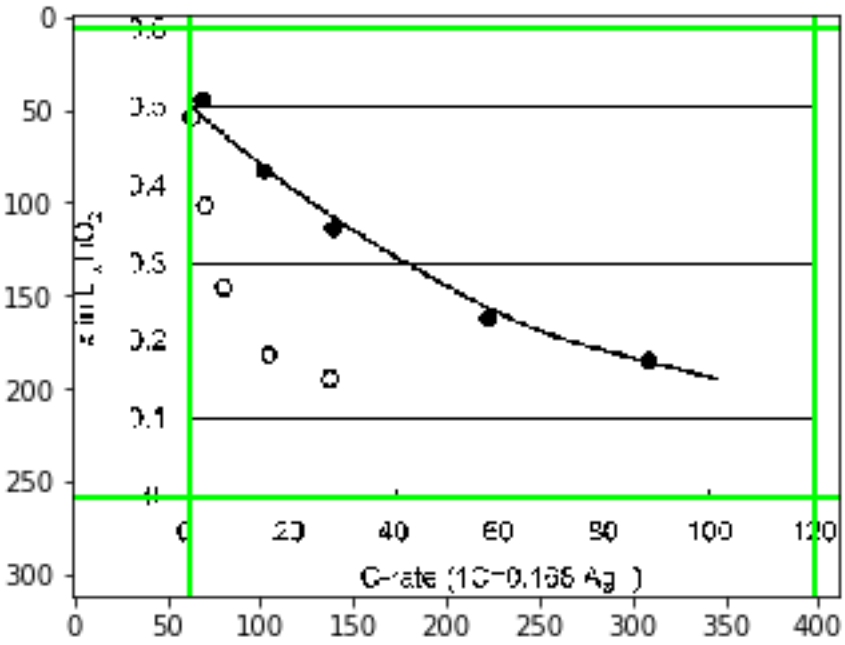

Figura 16. Representación de la detección de los ejes y bordes en la imagen original

En el caso de figuras con cuadrícula vertical, el modelo satisfactoriamente también reconoce los ejes y bordes, tal como se puede apreciar en la figura 12 y 13.

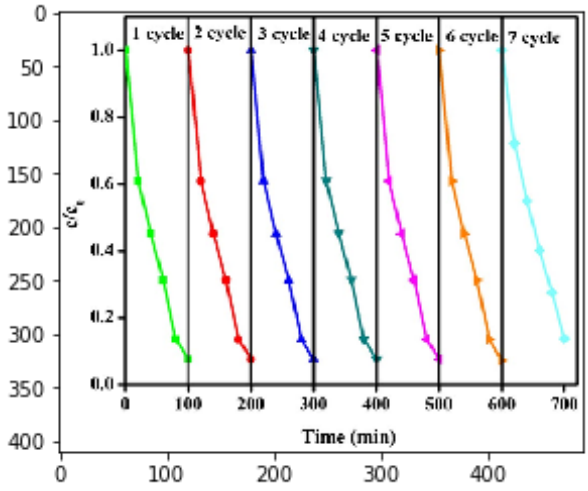

Figura 17. Gráfica con cuadrículas verticales

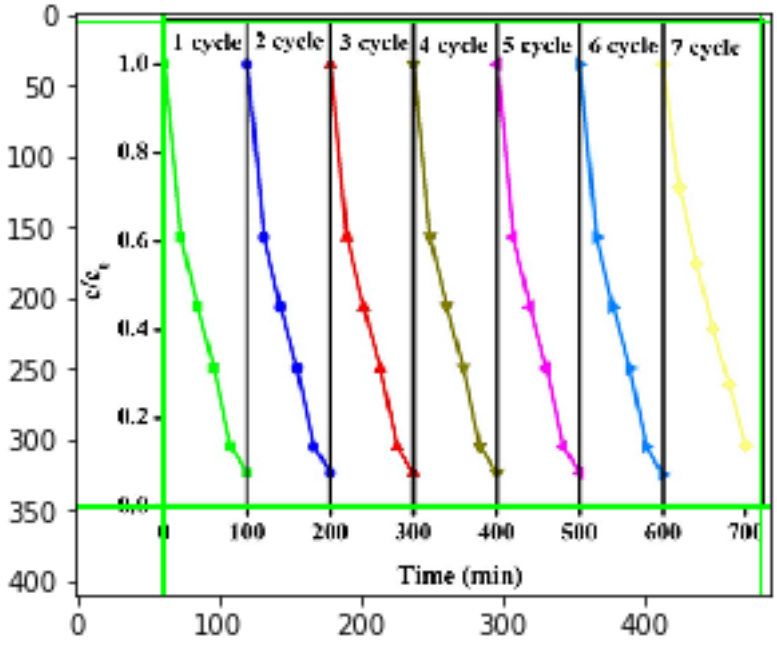

Figura 18. Detección de ejes y bordes en figura con líneas verticales

9. Detección y extracción de líneas, curvas y metadata

9.1. Identificación de colores y creación de máscaras

Para la extracción de colores de la imagen se usó un paquete de Python llamado extcolors, que, como se espera, extrae los valores RGB que más detecta en la imagen. Con esta información se generaron unos límites inferiores y superiores por cada color detectado. Esto, para luego crear una máscara que segmentara, según estos rangos, en la imagen. Lo anterior permite en la mayoría de los casos extraer una a una cada curva o línea, según su color. Una vez aplicada esta máscara la imagen, a través de cv2.canny se detectaron los bordes que se encontrasen, por cada color. Esto permite, básicamente, encontrar el contorno de cada curva. Por último, estas dos operaciones se realizaron solo dentro del cuadrado que se generó en el paso anterior. Esto se hizo con el fin de que el algoritmo no fuese ejecutado en áreas innecesarias. Una vez se generaba el contorno de la imagen, se procedía a dibujar la línea con una línea verde, tal como se aprecia en la figura 14.

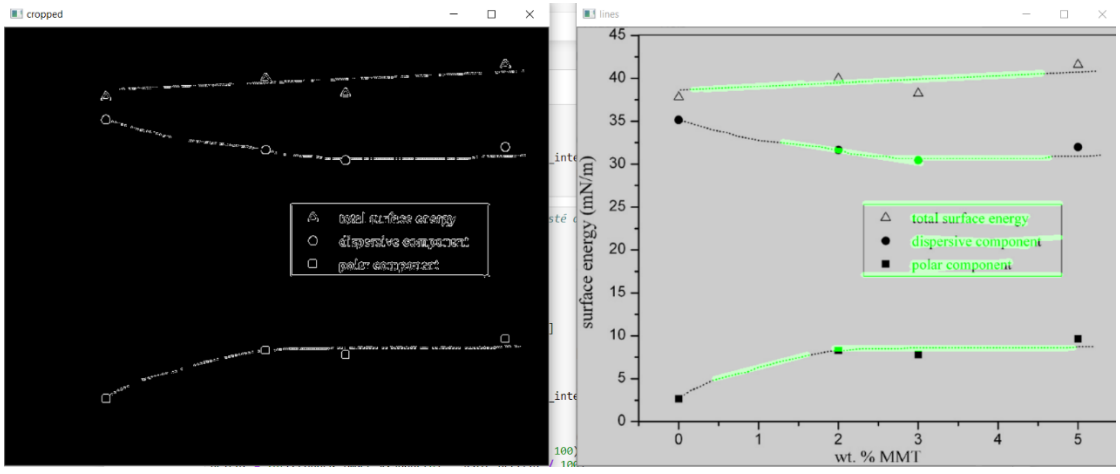

Figura 19. Segmentación y extracción de gráficas de color negro

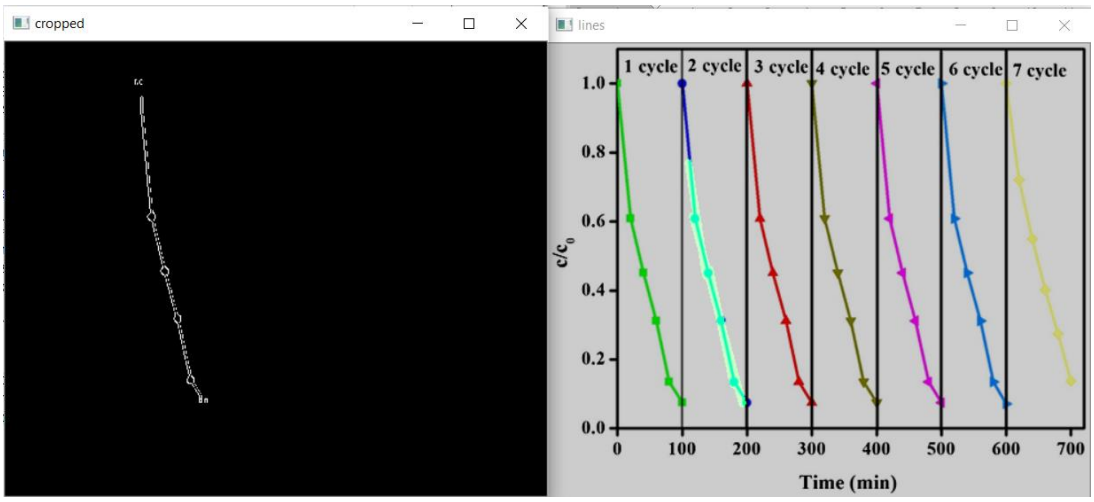

Figura 20. Ejemplo de detección de líneas de diferentes colores

Existen varios desafíos en la extracción de estas líneas. Se deben probar distintas configuraciones, por ejemplo, en el método de Hough Lines, que es el usado para dibujar las líneas verdes. Asimismo, en muchas de las figuras no se representa el significado de cada color de cada línea.

9.2. Detección y extracción de valores de los ejes

Una vez detectado, por ejemplo, el eje X, es muy sencillo cortar o segmentar los valores del eje X y la leyenda de su significado, tal como se ve en la figura 16. Luego, una vez esto se ha hecho, es posible extraer solamente los valores numéricos, para luego, pasarlos al modelo de detección de objetos. Este paso se aprecia en la figura 17. Esto se hizo, analizando las filas de la figura 16, por ejemplo, y detectando un largo de valores con pixeles blancos: donde hubiese valores seguidos y mayores a 252, se podía concluir que existía un gran espacio blanco, es decir, la separación entre los números y la leyenda del eje. Una vez reconocido este espacio, se trazaba una línea de separación en la mitad del rango, para así, separar los números de la leyenda.

La idea es que estos números sean reconocidos a través de un modelo de detección de objetos (dígitos) y que su ubicación se obtenga, para así, poder reconstruir la imagen digitalmente. Esto permitirá aterrizar las curvas extraídas y escalarlas según el rango de los ejes.

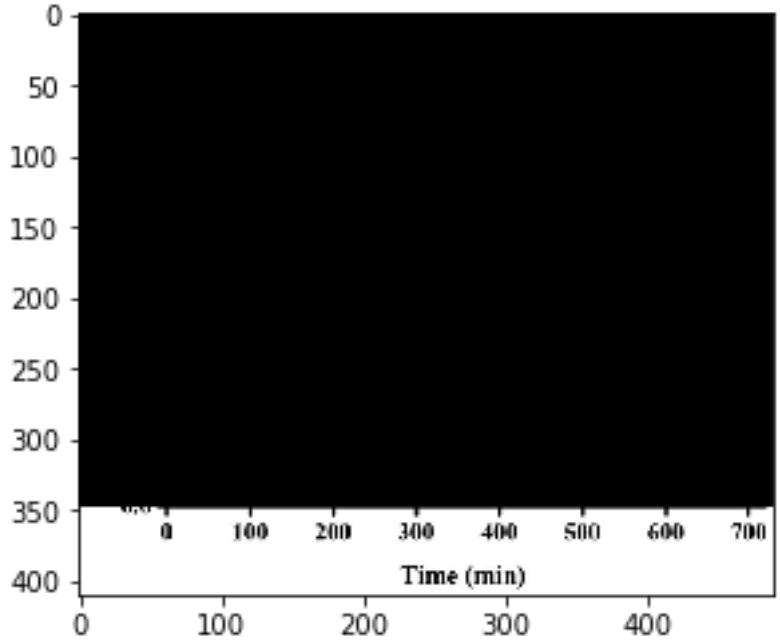

Figura 21. Extracción de los valores del eje X y de la leyenda

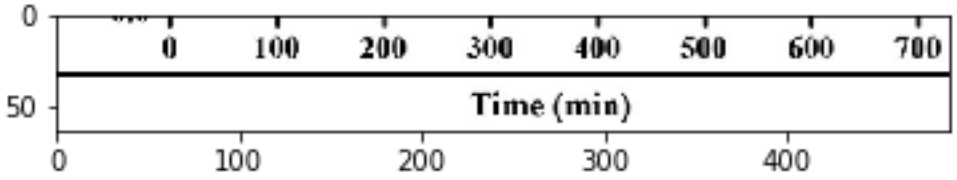

Figura 22. Línea de separación entre los números y la leyenda del eje

9.3. Detección de texto y de roles

Para llevar a cabo la detección del texto que estaba dentro de la imagen se usó la API de Visión Computacional de Google. Esta recibe imágenes y devuelve un archivo json con: el elemento detectado, las coordenadas de ubicación de la caja que encierra el elemento y el

nombre de la imagen a la cual hacía referencia. Una vez se generó el archivo json con toda la información de cada imagen, esta se pasó a un dataframe de Pandas, con el objetivo de limpiar los datos. La figura 23 permite ver que, por ejemplo, se detectaban guiones con los números. Por ende, a través de regex se excluyeron caracteres que estuvieran a la derecha de cada digito. Luego, cada elemento se convertía de string a float.

	value	v1	v2	v3	v4	img_name
0	13-	[29, 1]	[44, 1]	[44, 13]	[29, 13]	Mondal201357-Figure5-1.png
1	12-	[29, 43]	[57, 43]	[57, 60]	[29, 60]	Mondal201357-Figure5-1.png
2	11	[29, 88]	[42, 88]	[42, 100]	[29, 100]	Mondal201357-Figure5-1.png
3	10-	[27, 129]	[48, 129]	[48, 146]	[27, 146]	Mondal201357-Figure5-1.png
4	8	[37, 219]	[44, 219]	[44, 231]	[37, 231]	Mondal201357-Figure5-1.png

Figura 23. Ejemplos con elementos alfanuméricos que no corresponden

Una vez se limpiaban los datos, se procedía a calcular el centro de ese rectángulo que encierra a cada elemento detectado. Esto se realizó haciendo restas entre cada punto en el eje X y en el eje Y. Con base a estos centroides, se procedió a identificar el rol de cada elemento. Esto se realizó gracias a la identificación que se hizo en los pasos anteriores de los valores del eje X, Y y de las leyendas de los mismos. Por ejemplo, si un elemento se detectó en una coordenada en el eje Y inferior a la línea de separación dibujada en la figura 22, significa que ese elemento representa la leyenda del eje Y. Esto se hizo para cada elemento y se creó un nuevo dataframe con toda la información, tal como se ve en la figura 24.

	value	v1	v2	v3	v4	img_name	centroides_x	centroides_y	rol
7982	200	[64, 355]	[88, 355]	[88, 366]	[64, 366]	Venkatesan201861-Figure10-1.png	76.0	360.5	valor_x
7983	300	[163, 355]	[189, 356]	[189, 367]	[163, 366]	Venkatesan201861-Figure10-1.png	176.0	361.5	valor_x
7984	400	[266, 356]	[291, 356]	[291, 367]	[266, 367]	Venkatesan201861-Figure10-1.png	278.5	361.5	valor_x
7985	500	[371, 357]	[396, 357]	[396, 367]	[371, 367]	Venkatesan201861-Figure10-1.png	383.5	362.0	valor_x
7986	600	[473, 357]	[498, 357]	[498, 367]	[473, 367]	Venkatesan201861-Figure10-1.png	485.5	362.0	valor_x

Figura 24. Dataframe con la información de los centros de los rectángulos y con el rol de cada elemento

Finalizado lo anterior, se extrajeron las coordenadas de cada curva detectada en el paso 9.1. Esta matriz, con la ubicación de cada punto de la curva, se interpoló, con base a la columna centroides_x o centroides_y. Esto, para convertir los pixeles a la escala real de la figura. De esta manera, se procedía a graficar el resultado para comprobar que todo estuviera siendo detectado correctamente; resultado apreciado en la figura 25. En la figura 26 se recuperaron las leyendas detectadas y se insertó el título de la imagen, usando un diccionario que contenía el título de la imagen, con el nombre del archivo al cual pertenecía. También se le asignó a la curva el color original. Por último, en la imagen 27 se aprecia la imagen del ejemplo.

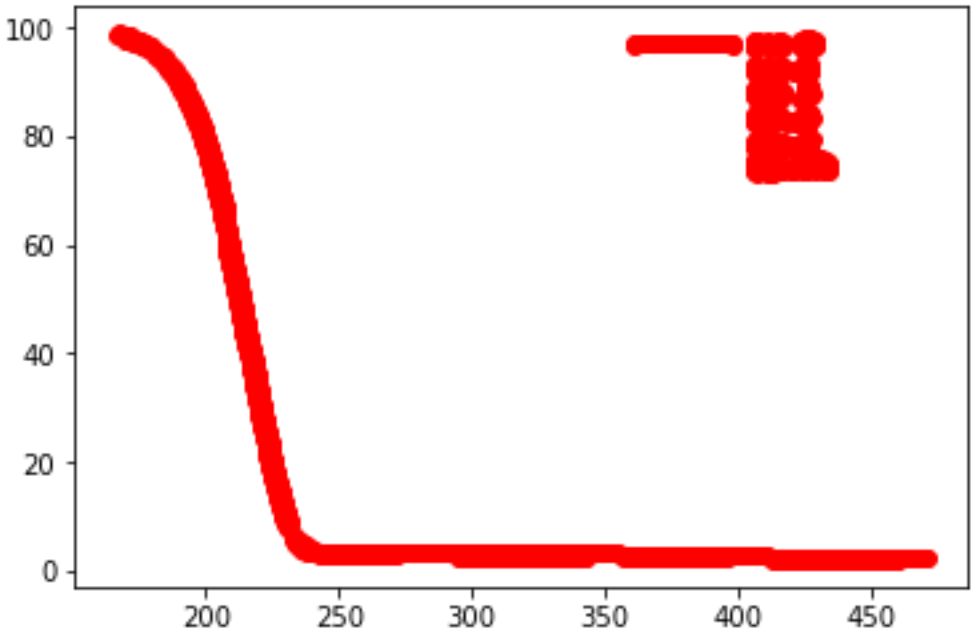

Figura 25. Graficación de matriz interpolada con valores reales

FIGURE 10 Thermogravimetric analysis curves of poly (butylene adipate-co-terephthalate) and poly (butylene adipate-co-terephthalate)/ silver oxide nanocomposites

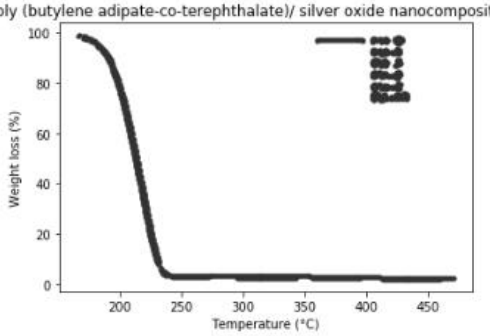

Figura 26. Resultado final con las leyendas y el título de la imagen

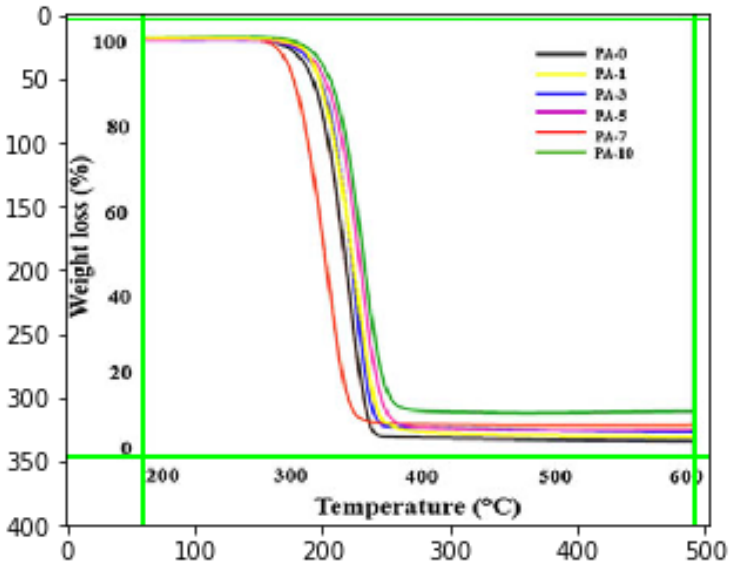

Figura 27. Imagen original

El código se puede encontrar en el siguiente repositorio:

https://github.com/felipec23/Information-extraction-from-scholar-figures

10. Conclusiones y recomendaciones

Según lo adelantado del proyecto, se puede concluir la importancia de realizar un análisis exploratorio previo a clasificar y procesar las imágenes. Por ejemplo, para el área escogida, la ciencia de los materiales, un poco menos de la mitad de las figuras corresponde solo a imágenes lineales. Por lo que es conveniente, si se quiere trabajar en esta área, enfocarse en esta clase de imágenes. Sin embargo, para otras áreas del conocimiento puede ser propicio analizar gráficas de barras, o fotografías, como es el caso de la medicina. Se presenta, entonces, la disyuntiva de: o generar un gran modelo que abarque distintas áreas y enfoques, o bien, generar pequeños modelos, específicos para cada área del conocimiento.

Se propone, según los resultados de la red convolucional, implementar un modelo multiclase y no binario. De esta forma, es probable que las grandes arquitecturas pre-entrenadas obtengan mejores métricas. Sin embargo, para esta tarea, habría que ampliar el dataset y el área del conocimiento, pues casi la mitad de las imágenes, para la ciencia de los materiales y para este conjunto de datos en específico, son gráficas lineales, por lo que habría un desbalance de clases considerablemente alto.

Es preciso, para la accesibilidad del proyecto, desplegar el código en una interfaz amigable con el usuario, ya sea a través de una página web o una aplicación móvil. Esto, en aras de disponer de sus funcionalidades a la mayor cantidad de investigadores y usuarios posible.

Se recomienda probar el algoritmo, tanto de clasificación de imágenes, como de extracción de información, en imágenes fuera de las ciencias de los materiales, para robustecer el modelo y encontrar errores que tal vez no se presentaban en esta área del conocimiento.

Se recomienda, por último, para futuros trabajos, que se asocie cada curva a su respectiva leyenda, pues este aspecto no se tuvo en cuenta en el presente trabajo, debido a que las leyendas muchas veces, no se encuentran dentro de la imagen, como tal, por lo que era imposible asociarlas.

11. Bibliografía

Acuña, A., & Arreche, E. (2014). *Algoritmos evolutos aplicados a la sincronización de semáforos en el Corredor Garzón*. Universidad de la República.

Boschen, F., & Scherip, A. (2017). A Comparison of Approaches for Automated Text Extraction from Scholarly Figures. *Spring International Publishing*, *1*, 478–489. https://doi.org/10.1007/978-3-319-51811-4

Böschen, F, & Scherp, A. (2017). A comparison of approaches for automated text extraction from scholarly figures. *International Conference on Multimedia Modeling*. Retrieved from https://link.springer.com/chapter/10.1007/978-3-319-51811-4_2

Böschen, Falk, Beck, T., & Scherp, A. (2018). Survey and empirical comparison of different approaches for text extraction from scholarly figures. *Multimedia Tools and Applications*, *77*(22), 29475–29505. https://doi.org/10.1007/s11042-018-6162-7

Carberry, S., Elzer, S., & Demir, S. (2006). Information graphics: An untapped resource for digital libraries. *Proceedings of the Twenty-Ninth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, *2006*, 581–588.

Choudhury, S R. (2017). *An Architecture For Multimodal Information Extraction From Scholarly Documents*. Retrieved from http://search.proquest.com/openview/61d5598dccbd2367c74ee4e8d2856f36/1?pq-origsite=gscholar&cbl=18750&diss=y

Choudhury, Sagnik Ray. (2017). *An arquitecture for multimodal information extraction from scholarly documents* (The Pennsylvania State University). https://doi.org/10.1017/CBO9781107415324.004

Choudhury, Sagnik Ray, & Giles, C. L. (2015). An architecture for information extraction from figures in digital libraries. *WWW 2015 Companion - Proceedings of the 24th International Conference on World Wide Web*, 667–672. https://doi.org/10.1145/2740908.2741712

Choudhury, Sagnik Ray, Wang, S., Mitra, P., & Giles, C. L. (2015). Automated Data Extraction from Scholarly Line Graphs. *Grec*.

Dai, W., Wang, M., Niu, Z., & Zhang, J. (2018). Chart decoder: Generating textual and numeric information from chart images automatically. *Journal of Visual Languages and Computing*, *48*, 101–109. https://doi.org/10.1016/j.jvlc.2018.08.005

Foumani, S. N. M., & Nickabadi, A. (2019). A probabilistic topic model using deep visual word representation for simultaneous image classification and annotation. *Journal of Visual Communication and Image Representation*, *59*, 195–203. https://doi.org/https://doi.org/10.1016/j.jvcir.2019.01.009

Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, P. (2014). *Metodología de la investigación* (6ta ed.). Mc Graw Hill Education.

IBM. (n.d.). El modelo de redes neuronales. Retrieved from https://www.ibm.com/support/knowledgecenter/es/SS3RA7_sub/modeler_mainhelp_client_ddita/components/neuralnet/neuralnet_model.html

Kaeli, D., Mistry, P., Schaa, D., & Zhang, D. P. (2015). Chapter 9 - Case study: Image clustering. In D. Kaeli, P. Mistry, D. Schaa, & D. P. Zhang (Eds.), *Heterogeneous Computing with OpenCL 2.0* (pp. 213–228). https://doi.org/https://doi.org/10.1016/B978-0-12-801414-1.00009-0

Lee, P. (2017). VizioMetrics: Mining the Scientific Visual Literature. *ProQuest Dissertations and Theses*, 125. Retrieved from

https://login.pallas2.tcl.sc.edu/login?url=https://search.proquest.com/docview/1943998626?accountid=13965%0Ahttp://resolver.ebscohost.com/openurl?ctx_ver=Z39.88-2004&ctx_enc=info:ofi/enc:UTF-8&rfr_id=info:sid/ProQuest+Dissertations+%26+Theses+Global&rft_v

Li, P., Jiang, X., & Shatkay, H. (2019). Figure and caption extraction from biomedical documents. *Bioinformatics*, (April), 1–8. https://doi.org/10.1093/bioinformatics/btz228

Medina Nolasco, J. D. (2015). *Implementación de un algoritmo genético para la optimización de flujo vehicular aplicado a la fase de tiempos en las intersecciones de un Corredor Vial*. Pontificia Universidad Católica del Perú.

Sadiq, A. S., Faris, H., Al-Zoubi, A. M., Mirjalili, S., & Ghafoor, K. Z. (2019). Chapter 17 - Fraud Detection Model Based on Multi-Verse Features Extraction Approach for Smart City Applications. In D. B. Rawat & K. Z. Ghafoor (Eds.), *Smart Cities Cybersecurity and Privacy* (pp. 241–251). https://doi.org/https://doi.org/10.1016/B978-0-12-815032-0.00017-2

Savva, M., Kong, N., Chhajta, A., Li, F. F., Agrawala, M., & Heer, J. (2011). ReVision: Automated classification, analysis and redesign of chart images. *UIST'11 - Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, 393–402. https://doi.org/10.1145/2047196.2047247

Sisalima Ortega, F. R. (2018). *Sistema para detección y conteo vehicular aplicando técnicas de visión artificial.* Retrieved from http://dspace.unl.edu.ec/jspui/handle/123456789/20892

Tang, B., Liu, X., Lei, J., Song, M., Tao, D., Sun, S., & Dong, F. (2016). DeepChart: Combining deep convolutional networks and deep belief networks in chart classification. *Signal Processing*, *124*, 156–161. https://doi.org/10.1016/j.sigpro.2015.09.027