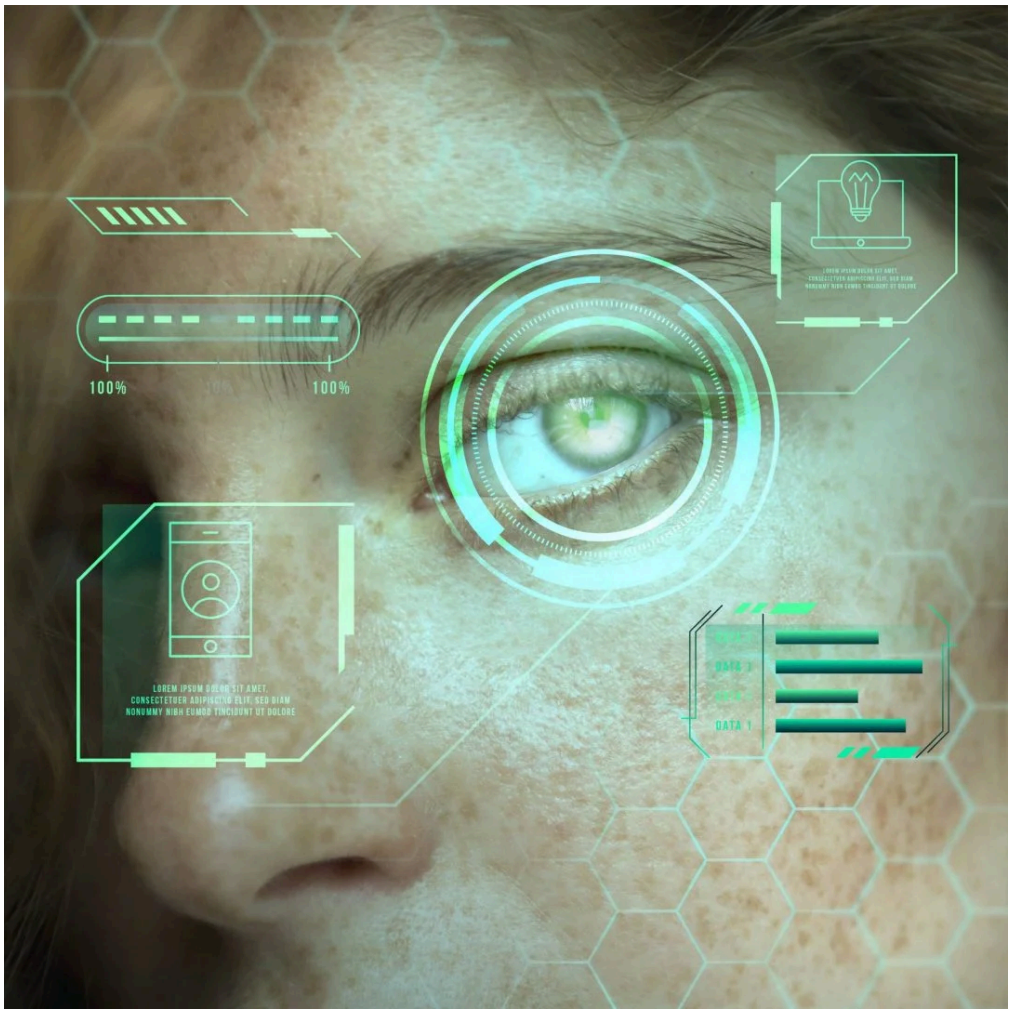
Detector de imperfecciones en madera



Miguel Estévez Díaz
Iria Vázquez Álvarez

Visión artificial
Grao en Robótica

Índice:

1. Introducción: **3**
2. Descriptor: **3**
3. Clasificadores: **6**
a. SVM: 6
b. Cascade: 7
c. MLP: 7
4. Resultados: **8**
a. SVM: 8
b. Cascade: 10
c. MLP: 11
5. Conclusiones: **14**
6. Bibliografía: **14**

1. Introducción:

La detección de imperfecciones en la madera es un problema crucial en la industria maderera, ya que permite identificar defectos que afectan la calidad del producto final.

Este trabajo se centra en el análisis del desempeño de diferentes técnicas de detección, utilizando descriptores HOG (Histogram of Oriented Gradients) en combinación con dos clasificadores diseñados específicamente para esta tarea: un detector HOG multiescala basado en un clasificador SVM y un clasificador MLP (Perceptrón Multicapa), y un tercer clasificador que trabaja directamente sobre las imágenes, un clasificador Cascade. Todos estos clasificadores han sido implementados y ajustados manualmente como parte del desarrollo de este proyecto.

El trabajo incluye la extracción de características HOG a partir de imágenes de madera, el entrenamiento de cada uno de los clasificadores y la evaluación de su rendimiento sobre varios conjuntos de pruebas. Finalmente, estas técnicas se aplican para la detección de imperfecciones en secuencias de vídeo, lo que representa un desafío adicional debido a factores como las variaciones de iluminación, movimiento de la cámara y complejidad del fondo.

El objetivo principal es comparar la efectividad de los clasificadores mencionados en términos de precisión, recall, F1-Score y capacidad de generalización en el contexto de detección en tiempo real. Este enfoque permite identificar las fortalezas y limitaciones de cada técnica, proporcionando una base sólida para futuras mejoras e implementaciones en aplicaciones prácticas.

2. Descriptor:

HoG (Histogram of Oriented Gradients) es una técnica utilizada para describir las características locales de una imagen mediante la distribución de los gradientes de intensidad, que incluyen tanto las direcciones como las magnitudes, en regiones específicas de la imagen. El objetivo principal de HoG es capturar las formas y bordes presentes, que son fundamentales para la identificación de objetos.

En este caso, utilizamos HoG como descriptor para transformar las imágenes de entrada (ya sean de tablas con o sin defectos) en vectores de características. Estos vectores se utilizan luego como entradas para los clasificadores (SVM y MLP).

El proceso para calcular el descriptor HoG seguimos los siguientes pasos:

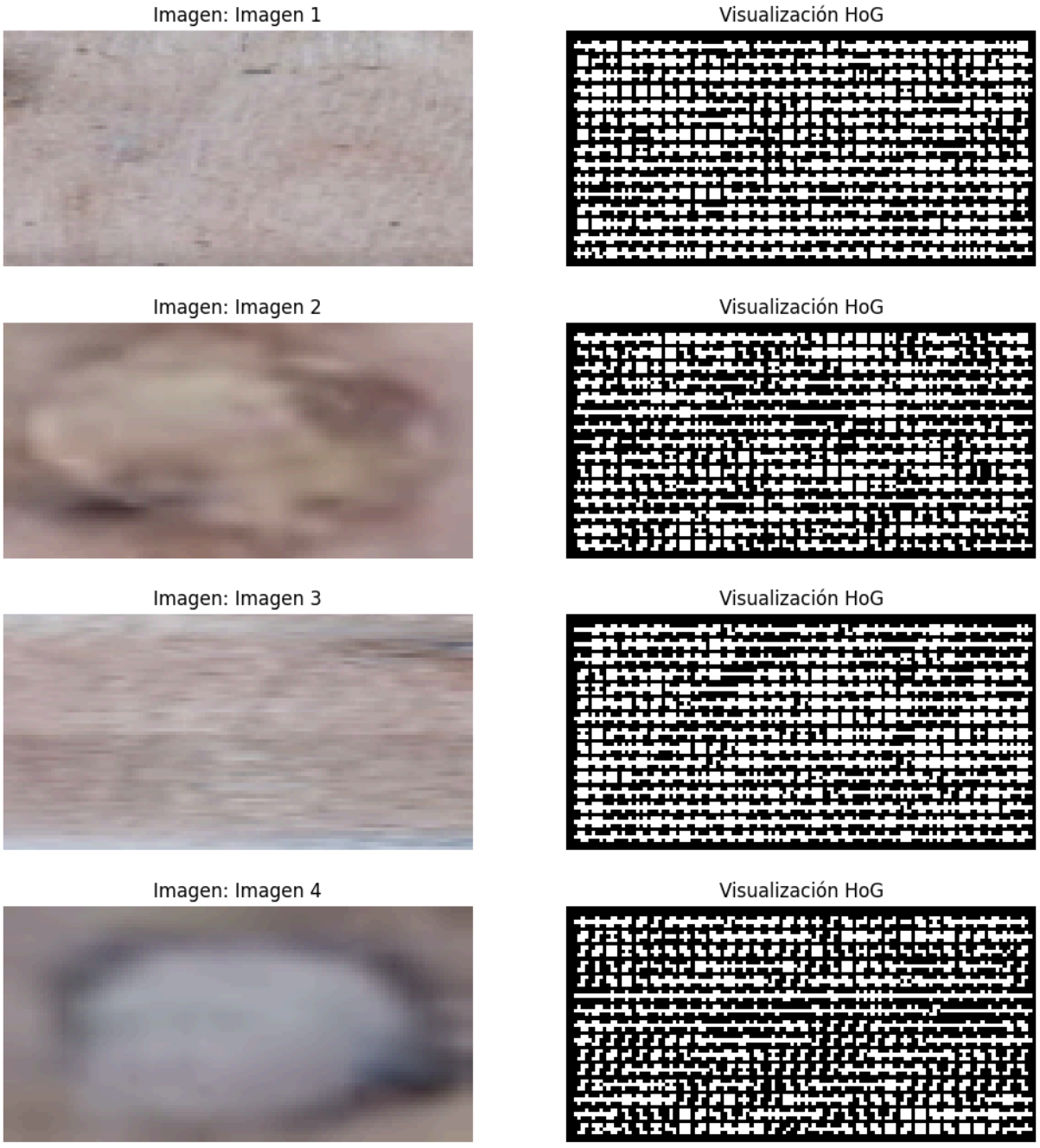
1. Preprocesamiento de la imagen:
    - Convertir la imagen a escala de grises (si no está ya en esa escala), ya que la intensidad de los píxeles es el dato que necesitamos para calcular los gradientes.
    - Reducción del ruido en la imagen mediante un filtro bilateral para no perder los detalles importantes.
    - Normalización de los valores de la imagen, para la potenciación de máximos y discriminación de mínimos.
2. Cálculo de gradientes:
    - Se calcula el gradiente de la intensidad en cada píxel de la imagen utilizando operadores Sobel. Esto genera dos componentes para cada píxel:
        - Magnitud del gradiente: $\sqrt{Gx^2 + Gy^2}$
        - Orientación del gradiente: arctan(Gy/Gx)

A continuación, se describen los parámetros utilizados para la configuración del descriptor HoG:

- **winSize (64, 64):**
  Define el tamaño de la ventana de detección donde se calculan los descriptores HoG. La imagen se divide en varias ventanas de este tamaño, lo que asegura que se capturen características relevantes sin perder información clave sobre bordes y formas.
- **blockSize (8, 8):**
  Establece el tamaño de los bloques, que son agrupaciones de varias celdas. Cada bloque es la unidad básica normalizada en el descriptor HoG. Bloques pequeños, como en este caso 8×8, permiten una mejor resolución para capturar detalles locales.
- **blockStride (4, 4):**
  Representa el desplazamiento entre bloques consecutivos. Un desplazamiento menor que el tamaño del bloque, como en este caso, introduce solapamiento entre ellos, lo que mejora la robustez frente a variaciones en la imagen.
- **cellSize (8, 8):**
  Indica el tamaño de las celdas, que son las unidades donde se calculan los histogramas de los gradientes. Celdas pequeñas, como en este caso 8 x 8, capturan detalles más finos, pero aumentan el costo computacional.
- **nbins (9):**
  Determina el número de bins en los histogramas de las orientaciones de los gradientes. Con 9 bins, el rango de 0° a 180° se divide en 9 intervalos de 20°, lo que permite capturar precisamente las

direcciones de los bordes.

A continuación se muestran parejas de imágenes y su descriptor de Hog correspondiente:



En las cuales podemos observar que el descriptor de Hog de la madera está conformado por histogramas más o menos estables, pero cuando se trata de una imperfección presentan cierta irregularidad representando la forma de la imperfección.

3. Clasificadores:

a. SVM:

Support Vector Machine (SVM) es un clasificador supervisado que tiene como objetivo separar los datos en diferentes clases utilizando un hiperplano óptimo en un espacio de características multidimensionales. La distancia de este hiperplano a los puntos más cercanos de cada clase (conocidos como vectores de soporte), es el margen que SVM trata de maximizar. Al maximizar este margen, el modelo logra una mejor capacidad para generalizar, es decir, para clasificar correctamente nuevos datos no vistos. En el caso de que los datos no sean separables linealmente, los SVM utilizan kernels, que son funciones de transformación, para proyectar los datos a un espacio de mayor dimensión donde sí pueden separarse. El kernel define cómo se transforman los datos para proyectarlos a un espacio de características donde sea posible separarlos, en nuestro caso usamos un kernel lineal para un problema de clasificación binaria (positivos y negativos).

En este clasificador también aplicamos minería de negativos difíciles, la cuál consiste en identificar los falsos positivos en las predicciones de los datos de entrenamiento, después de haber entrenado el clasificador, para volver a entrenarlo con estos datos, denominados "negativos difíciles", junto con los datos positivos del entrenamiento. El objetivo principal de esta técnica es mejorar la precisión del modelo en las predicciones finales. Al centrarse en los casos donde el clasificador está fallando, se ajusta mejor a las características críticas que diferencian los datos positivos de los negativos, haciendo que las predicciones sean más precisas.

El parámetro C (0.01) o parámetro de penalización controla el equilibrio entre maximizar el margen y minimizar los errores de clasificación. Valores bajos de C permiten márgenes más amplios pero con más errores, mientras que valores altos reducen los errores a costa de un margen más estrecho, lo que podría llevar al sobreajuste.

Finalmente conectamos el modelo SVM entrenado con el descriptor HoG para que pueda identificar defectos en las imágenes. Primero, tomamos lo que el SVM aprendió al entrenarse, es decir, qué características son importantes para diferenciar entre defectos y no defectos. Después, adaptamos esa información para que HoG pueda usarla como una especie de "regla" que le permita detectar esas mismas características en nuevas imágenes.

b. Cascade:

El clasificador en cascada es una técnica que combina múltiples clasificadores simples en una secuencia, con el objetivo de realizar una detección eficiente. El clasificador en cascada busca la detección de los objetos mediante el entrenamiento de múltiples clasificadores débiles, basándose en este caso en descriptores HAAR. Cada uno de los clasificadores se centra en la detección de un conjunto de características diferentes, los cuales se auto ajustan durante la fase de entrenamiento. El proceso de detección consiste en la clasificación de la imagen mediante todos los clasificadores, descartando esta en caso de alguno de ellos dar una respuesta negativa.

Generamos nuestro modelo utilizando Cascade Trainer GUI, una herramienta que simplifica la creación de clasificadores en cascada entrenados con nuestros propios datos. Esta herramienta automatiza el proceso de entrenamiento mediante el uso de imágenes positivas (que contienen el objeto de interés, las imperfecciones de la madera) e imágenes negativas (que no lo contienen). Internamente, se emplea Haar para describir las imágenes y entrenar los clasificadores. El modelo final, contenido en el archivo cascade.xml, incorpora toda la información necesaria, incluyendo umbrales, parámetros y características seleccionadas, lo que nos permite cargarlo y utilizarlo directamente con OpenCV para detectar objetos en imágenes o videos nuevos.

c. MLP:

El Perceptrón Multicapa (MLP) es una red neuronal artificial supervisada diseñada para aprender relaciones no lineales entre las entradas y las salidas mediante capas de neuronas interconectadas. Consta de tres capas principales: una capa de entrada, que recibe las características (en este caso, los descriptores HoG); capas ocultas, encargadas de realizar transformaciones no lineales para identificar patrones complejos; y una capa de salida, que genera la predicción final. Las neuronas de cada capa están conectadas a las de la siguiente mediante pesos, los cuales se ajustan durante el entrenamiento para minimizar los errores de predicción..

Estructura de nuestra red -> en este caso nuestra red está compuesta por:

- Una capa de entrada con tamaño 4185, debido a que este es el

tamaño de los descriptores usados.
- Una capa oculta con 20 neuronas, adecuada para aprender los patrones relevantes.
- Una capa de salida con 2 neuronas, una para cada clase.

4. Resultados:

En este apartado analizaremos las métricas obtenidas (precisión, recall, f1-score y matriz de confusión) en los diferentes tests realizados con los distintos clasificadores. Para ello, nos situamos en el punto de que los clasificadores ya están entrenados, para esto, a partir de una base de datos que extrajimos de la página *kaggle*, adaptada para nuestro proyecto, la cuál dividimos entre datos de entrenamiento y datos de test, estos últimos pueden ser divididos en varios subgrupos, identificados como test_número. En nuestro caso entrenamos los clasificadores con el 75% de imágenes, es decir 1878 imágenes, dividiendo las 626 imágenes restantes en 4 test de 154 imágenes cada uno.

a. SVM:

Test 1:

	precision	recall	f1-score	support
0	0.88	0.88	0.88	77
1	0.88	0.88	0.88	77
accuracy			0.88	154
macro avg	0.88	0.88	0.88	154
weighted avg	0.88	0.88	0.88	154
Matriz de confusión				
68		9		
9		68		
SVM predijo correctamente 136 de 154				

Test 2:

	precision	recall	f1-score	support
0	0.90	0.95	0.92	77
1	0.95	0.90	0.92	77
accuracy			0.92	154
macro avg	0.92	0.92	0.92	154
weighted avg	0.92	0.92	0.92	154
Matriz de confusión				
73		4		
8		69		
SVM predijo correctamente 142 de 154				

Test 3:

	precision	recall	f1-score	support
0	0.96	0.95	0.95	77
1	0.95	0.96	0.95	77
accuracy			0.95	154
macro avg	0.95	0.95	0.95	154
weighted avg	0.95	0.95	0.95	154
Matriz de confusión				
73		4		
3		74		
SVM predijo correctamente 147 de 154				

Test 4:

	precision	recall	f1-score	support
0	0.94	0.86	0.90	77
1	0.87	0.95	0.91	77
accuracy			0.90	154
macro avg	0.91	0.90	0.90	154
weighted avg	0.91	0.90	0.90	154
Matriz de confusión				
66		11		
4		73		
SVM predijo correctamente 139 de 154				

b. Cascade:

Test 1:

	precision	recall	f1-score	support
0	0.59	0.70	0.64	77
1	0.63	0.51	0.56	77
accuracy			0.60	154
macro avg	0.61	0.60	0.60	154
weighted avg	0.61	0.60	0.60	154
Matriz de confusión				
54		23		
38		39		
Cascade predijo correctamente 93 de 154				

Test 2:

	precision	recall	f1-score	support
0	0.53	0.65	0.58	77
1	0.55	0.43	0.48	77
accuracy			0.54	154
macro avg	0.54	0.54	0.53	154
weighted avg	0.54	0.54	0.53	154
Matriz de confusión				
50		27		
44		33		
Cascade predijo correctamente 83 de 154				

Test 3:

	precision	recall	f1-score	support
0	0.51	0.53	0.52	77
1	0.51	0.48	0.49	77
accuracy			0.51	154
macro avg	0.51	0.51	0.51	154
weighted avg	0.51	0.51	0.51	154
Matriz de confusión				
41		36		
40		37		
Cascade predijo correctamente 78 de 154				

Test 4:

	precision	recall	f1-score	support
0	0.44	0.40	0.42	77
1	0.45	0.49	0.47	77
accuracy			0.45	154
macro avg	0.45	0.45	0.45	154
weighted avg	0.45	0.45	0.45	154
Matriz de confusión				
31		46		
39		38		
Cascade predijo correctamente 69 de 154				

c. MLP:

Test 1:

	precision	recall	f1-score	support
0	0.79	0.77	0.78	77
1	0.77	0.79	0.78	77
accuracy			0.78	154
macro avg	0.78	0.78	0.78	154
weighted avg	0.78	0.78	0.78	154
Matriz de confusión				
59		18		
16		61		
MLP predijo correctamente 120 de 154				

Test 2:

	precision	recall	f1-score	support
0	0.90	0.92	0.91	77
1	0.92	0.90	0.91	77
accuracy			0.91	154
macro avg	0.91	0.91	0.91	154
weighted avg	0.91	0.91	0.91	154
Matriz de confusión				
71		6		
8		69		
Cascade predijo correctamente 83 de 154				

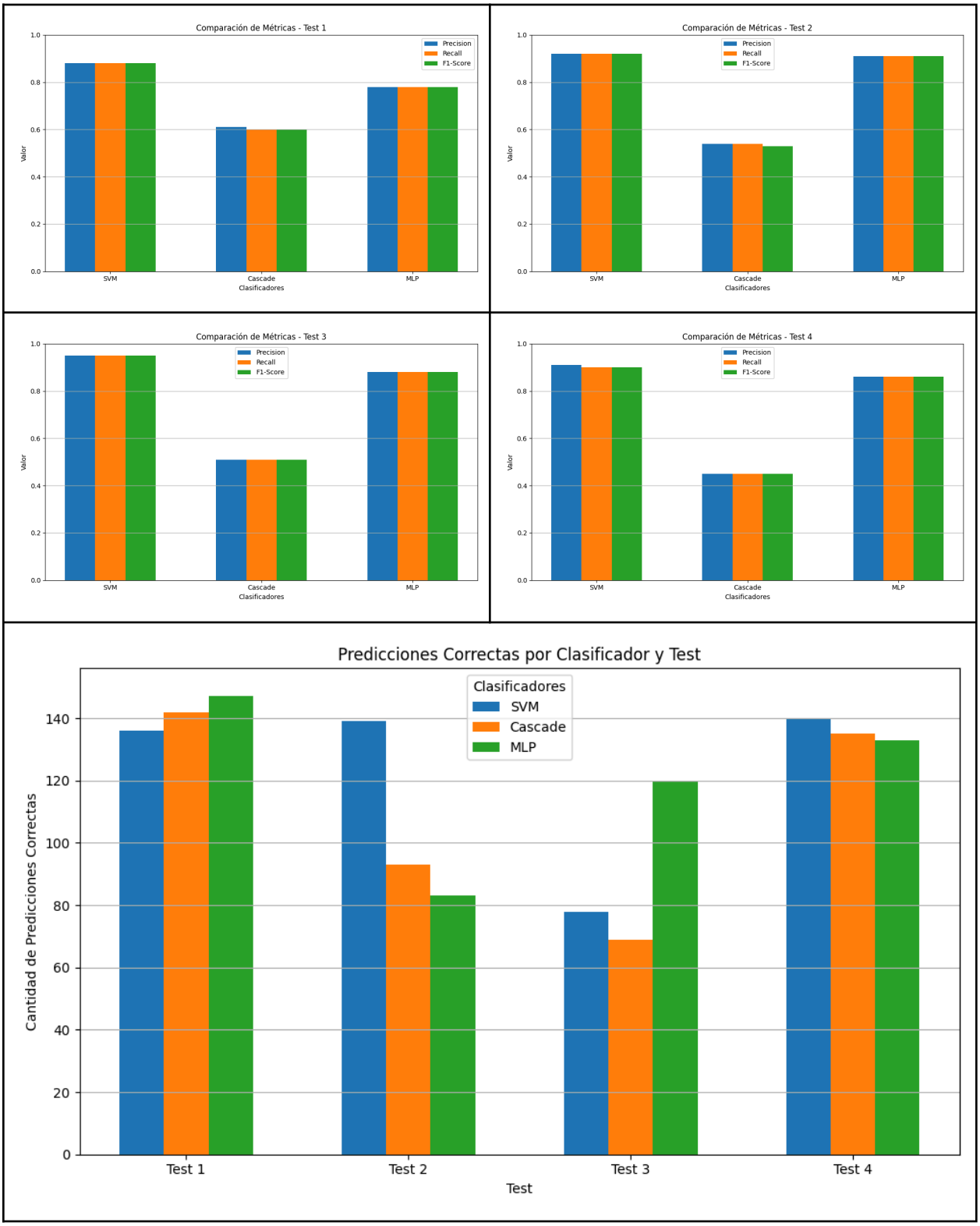
Test 3:

	precision	recall	f1-score	support
0	0.87	0.88	0.88	77
1	0.88	0.87	0.88	77
accuracy			0.88	154
macro avg	0.88	0.88	0.88	154
weighted avg	0.88	0.88	0.88	154
Matriz de confusión				
68		9		
10		67		
Cascade predijo correctamente 78 de 154				

Test 4:

	precision	recall	f1-score	support
0	0.88	0.84	0.86	77
1	0.85	0.88	0.86	77
accuracy			0.86	154
macro avg	0.86	0.86	0.86	154
weighted avg	0.86	0.86	0.86	154
Matriz de confusión				
65		12		
9		68		
Cascade predijo correctamente 69 de 154				

d. Comparaciónn mediante tablas:



5. Conclusiones:

Aunque los resultados teóricos mostraron que el SVM era el clasificador más preciso y robusto, los resultados prácticos demuestran una realidad diferente debido al comportamiento observado durante su implementación en un entorno más realista.

MLP resulta ser el clasificador más preciso en la práctica, logrando identificar defectos con un alto nivel de precisión. A pesar de los datos teóricos que sugerían un rendimiento ligeramente inferior al SVM, el MLP sobresale al manejar mejor las imágenes reales y minimizar los falsos positivos y negativos.

En el caso de Haarcascade los resultados teóricos mostraron un desempeño bajo,pero este se desempeñó mejor en la práctica en comparación con el SVM. Tiene errores en la detección bastante evidentes, nuestra teoría es que kernel lineal funciona bien cuando los datos son linealmente separables en el espacio de características, sin embargo, en problemas más complejos como la detección de defectos en madera (donde las características son más intrincadas), un kernel lineal puede no ser suficiente.

Video con los resultados: video_presentacion.mp4

6. Bibliografía:

- Codigos da práctica 4
- Codigos da práctica 5
- https://www.kaggle.com/datasets/pzq1995/wood-defect-20240828-max-pro
- https://omes-va.com/como-crear-tu-propio-detector-de-objetos-con-haar-cascade-python-y-opencv/