TORNIDENTIFIER: IDENTIFICACIÓN Y CLASIFICACIÓN AUTOMÁTICA DE TORNILLOS CON REDES NEURONALES PROFUNDAS

Presentado para obtener el título de:

MAGISTER EN MATEMÁTICA APLICADA Y CIENCIAS DE LA COMPUTACIÓN

Luis Alejandro Garcia Espitia
Juan David Rojas Gacha

Dirección:
Egdar Jose Andrade Lotero
Edwin Santiago Alférez Baquero

Universidad del Rosario
Escuela de Ingeniería, Ciencia y Tecnología
Maestría en Matemática Aplicada y Ciencias de la Computación

Dedicatoria

Luis Alejandro,
Para mis padres ejemplo a seguir.

Resumen

La tarea de clasificación de tornillos hasta el momento es sólo ejecutada por humanos. De hecho, las fotos no son aceptadas como insumo para la clasificación de tornillos debido a que existe información que no se puede determinar con las imágenes, como el diámetro del tornillo y el paso de la rosca. Con el avance de los modelos del aprendizaje automático de máquina y la inclusión de la clasificación automática de imágenes digitales con arquitecturas de redes neuronales profundas, no se ha explorado la solución de esta tarea, en gran parte, porque el factor trascendental para su entrenamiento es un conjunto de datos apropiado que no existe para este problema. En el presente proyecto se construyó un conjunto de imágenes inédito con el cual se pretende entrenar redes neuronales profundas para la clasificación de los tornillos. Además, se entrenó un modelo de detección de objetos especializados para tornillos el cual funcionará juntamente con el modelo de clasificación para aparte de dar una clasificación se identifique en que parte de la imagen este el tornillo. Por último, los modelos fueron puestos en producción dentro de una interfaz en la cual el objetivo es subir una imagen con tornillos y que los modelos sean capaces de detectar donde están y clasificar sus características.

Índice

Índice de cuadros

Índice de figuras

1. Introducción

La detección y clasificación de objetos de una imagen de manera automática se ha vuelto un problema que se comienza a presentar en más y más en situaciones de la vida cotidiana. Los tornillos por ejemplo, son parte fundamental de muchas estructuras, electrodomésticos, muebles, entre otros. La mayoría de veces se necesita conocer las características más importantes del tornillo de manera rápida y sencilla. Una opción es ir a la ferretería o tornillería más cercana, pero si allí no se encuentra un vendedor especializado en su venta, la tarea puede ser más complicada aún si no fue posible llevar una muestra del tornillo. Es por esto que, se hace atractivo tener una herramienta de fácil acceso para las personas de este entorno con la cual les sea más sencillo la identificación del tornillo y sus características.

Los modelos de detección y clasificación automática que más se trabajan en la actualidad son los basados en redes neuronales, sobretodo redes neuronales convolucionales (CNN), las cuales se especializan en el tratamiento de imágenes. Pero una desventaja del uso de estos modelos es que necesitan de grandes conjuntos de datos para entrenarse, especialmente para el caso que trata éste trabajo, dado que en la literatura no existe un conjunto de datos de tornillos para clasificación. Es por ello que, en primer lugar se debe construir un conjunto de datos de imágenes de tornillos para clasificación y, posteriormente, se puede aplicar estos modelos de redes neuronales con todas las consideraciones a lugar para tener el mejor desempeño. Por otro lado, la fase de detección de tornillos se puede desarrollar con base en un conjunto de datos para la detección de tornillos [29], con el cual se puede entrenar un primer modelo de detección para luego utilizarlo en las imágenes creadas y poder entrenar un modelo de clasificación. Por último, es importante que el resultado más allá de ser un modelo entrenado, debe ser una herramienta de fácil uso, en la cual se pueda subir una imagen y la herramienta retorne los valores de las características de los tornillos.

2. Objetivos

2.1. Objetivo general

Desarrollar un sistema para la identificación y clasificación de roscas métricas paso normal y paso fino de diámetros entre 6 y 12 milímetros, y rosca americana unificada paso normal y paso fino de diámetros entre 1/4 y 1/2 pulgada mediante redes neuronales profundas.

2.2. Objetivos específicos

1. Construir y preprocesar un conjunto de datos de imágenes de roscas métricas paso normal y paso fino de diámetros entre 6mm y 12mm, y rosca americana unificada paso normal y paso fino de diámetros entre 1/4' y 1/2' presentes en situaciones cotidianas como en la mecánica automotriz y el ensamblaje de muebles.
2. Implementar un modelo de detección de objetos para identificar tornillos en una imagen.
3. Implementar arquitecturas de redes neuronales profundas para la clasificación automática de roscas de tornillos por su tipo, diámetro y paso.
4. Evaluar el desempeño del pipeline obtenido al incorporar los modelos de detección y clasificación.
5. Crear una interfaz gráfica que facilite la puesta en producción del pipeline para hacer pruebas de concepto del modelo de identificación y clasificación.

3. Poblema y Justificación

La presencia de los tornillos en el transcurso del día a día es imperceptible. Pero ellos son esenciales para que una variedad de objetos cumpla su función, por mencionar algunos: muebles, relojes, celulares, carros, televisores, computadores, etc. Esto motiva que haya negocios especializados en la producción y otros en la comercialización de los tornillos.
En los establecimientos especializados en su comercialización, conocidos como tornillerías o ferreterías es necesario que el cliente lleve una muestra, al menos una fracción del tornillo (que contenga al menos unos 4 hilos de la rosca), o que conozca la referencia. Esta última se establece con sus datos físicos: diámetro de la parte cilíndrica o roscada, tipo de rosca, largo sin incluir la cabeza, color, tipo de cabeza y la resistencia del material. Ahora bien, la necesidad usual en las tornillerías es encontrar la tuerca o un tornillo con características similares. Para poder responder a esta necesidad, sólo es necesario saber la referencia de la parte roscada que se establece mediante el tipo de rosca, el paso y el diámetro. Los tipos de rosca son variados dependiendo de las necesidades, en la Figura 1 se muestran los nombres de los tipos de roscas más usados.

Nombre de Roscas de uso frecuente	Denominación común	Otras denominaciones
American Petroleum Institute	API	
British Association	BA	
International Standards Organization	ISO	
Rosca CEI (para bicicletas y motocicletas)	C	
Rosca Edison (bombillas)	E	
Rosca de Filetes Redondos	Rd	
Rosca de Filetes Trapezoidales (o Trapecial)	Tr	
Rosca Dientes de Sierra	S	
Rosca para Tubos Blindados	PG	Pr
Rosca Whitworth de paso Normal	BSW	W
Rosca Whitworth de paso Fino	BSF	
Rosca Whitworth Cilindrica para Tubos	BSPP	
Rosca Whitworth Cónica para Tubos	BSPT	KR
Rosca Whitworth Gas (o Whitworth para tubos)	BSP	R
Rosca Métrica paso Normal	M	SI
Rosca Métrica paso Fino	M	SIF
Rosca Métrica para Aeronaútica	MJ	
Rosca Americana Unificada paso Grueso (o Basto)	UNC	NC, USS
Rosca Americana Unificada paso Fino	UNF	NF, SAE
Rosca Americana paso Extrafino	UNEF	NEF
Rosca Americana Cilindrica para Tubos	NPS	
Rosca Americana Cónica para Tubos	NPT	ASTP
Rosca Americana Paso Especial	UNS	NS
Rosca Americana Cilíndrica "dryseal" para tubos	NPSF	
Rosca Americana Cónica "dryseal" para tubos	NPTF	
Rosca Sharp Americana	V	VEE

Figura 1: Tabla con tipos de roscas
Fuente: https://commons.wikimedia.org/wiki/File:Cuadro_de_Roscas.png

Dentro de esta lista de tipos de rosca, los más solicitados en el comercio son los usados

en las industrias automotriz y de construcción cómo se lista a continuación:

- Rosca métrica paso normal y fino: cuyo diámetro varía desde 3 hasta 39 milímetros [2].
- Rosca americana unificada paso normal y fino: cuyo diámetro varía desde 1/8 hasta 2 pulgadas [3].

En esta variedad de diámetros comercialmente los más solicitados son de 6 a 12 milímetros para la rosca métrica y 1/4 a 1/2 pulgada para la rosca americana unificada, por lo que el presente estudio se va a focalizar en esos diámetros en los pasos normal y fino.

Al limitar el problema a las referencias mencionadas anteriormente se inducen 20 referencias diferentes. Además, se pueden presentar caso en que diámetros similares tienen tipos de rosca difíciles de diferenciar, inclusive para comerciantes especializados, como el caso de las referencias de rosca métrica 6MM P1.0 y la rosca americana unificada paso fino de 1/4 de diámetro que se muestran en la figura 2.

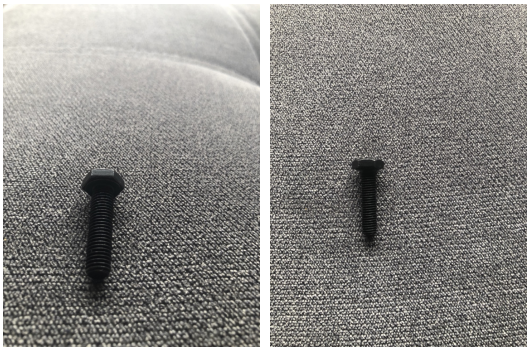

Figura 2: Tornillos a la izquierda 1/4 RF y a la derecha 6MM
Fuente: Autores

Otra situación en donde se puede generar confusión por la falta de información adicional acerca del tornillo que se busca, es la distancia entre el tornillo y la cámara. En la figura 3 se visualizan tornillos con rosca americana unificada rosca fina de 1/4 y 5/16, así su diámetro es 1/16 de pulgada de diferencia, pero al observar las imágenes es difícil diferenciarlos.

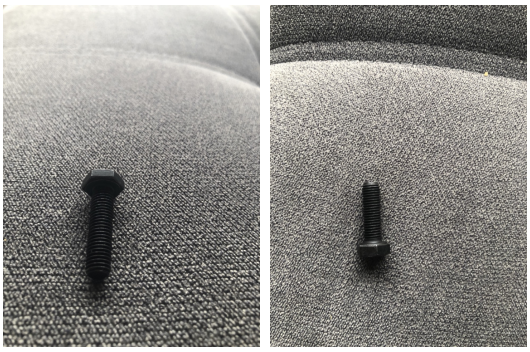

Figura 3: Tornillos a la izquierda 1/4 RF y a la derecha 5/16 RF
Fuente: Autores

Las dos situaciones anteriores muestran porque en el comercio no son aceptadas las fotos como insumo para la venta de tornillos.

Dado lo anterior, existe una barrera entre las fotos y la comercialización de los tornillos. Esto debido a que no se ha desarrollado una herramienta tecnológica que permita clasificar los tornillos a partir de las fotos, como tampoco se ha creado un conjunto de imágenes con tornillos roscados para entrenar un modelo de clasificación automática que aborde está tarea.

El presente proyecto se ha realizado creando un conjunto de imágenes digitales y luego desarrollando una metodología de identificación y clasificación automática de diferentes tipos de tornillos, involucrando técnicas de visión por computador y modelos de redes neuronales artificiales profundas. De esta forma, se determina que el problema de identificación de tornillos se puede resolver mediante la metodología propuesta y sienta las bases para la ejecución de está tarea.

4. Marco Teórico y Estado del Arte

Los modelos de redes neuronales son modelos de aprendizaje automático de máquina, los cuales se basan en que un conjunto de valores de entrada logren estimar un valor real de salida o predecir una clase o categoría. En la actualidad existen muchas áreas del conocimiento donde se aplican estos modelos, tales cómo: predicción de series financieras, identificaciones de objetos en una imagen, clasificación automática, procesamiento del lenguaje natural para interpretar y manejar el lenguaje, entre otros [1].

Aunque estos modelos de redes neuronales simples tienen un buen rendimiento, no dan una ventaja significativa en la clasificación de imágenes dada la cantidad de variables de entrada, las cuales constituyen una gran cantidad de información que la red debe aprender [4] [5]. Por esta razón, se introdujeron capas convolucionales las cuales consisten en uno o más filtros que se deben pasar por la imagen para obtener las características más relevantes que ayuden a la clasificación. Una red que utilice una o más capas convolucionales se denomina red neuronal convolucional (CNN), un ejemplo de una CNN se puede ver en la figura 4. Al poner varias capas convolucionales con muchos filtros se llega a una arquitectura que se denomina como red convolucional profunda.

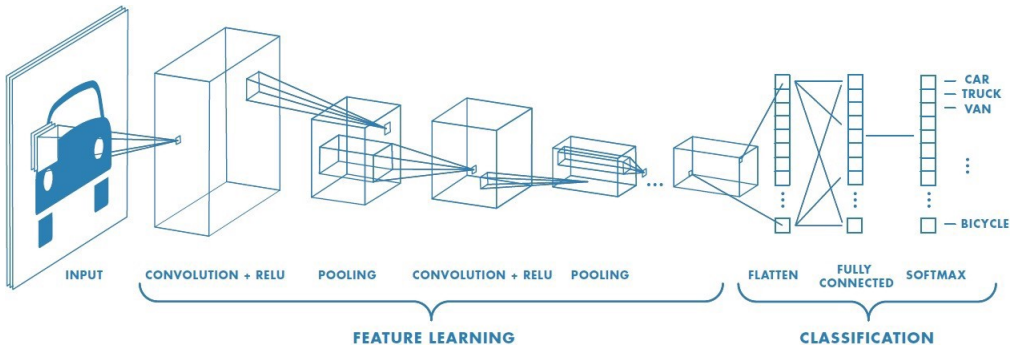

Figura 4: Red convolucional profunda.
Fuente: [6]

Sin embargo, la inclusión de esta nueva capa dentro de la arquitectura de la red no es suficiente para tener un buen desempeño, también se hace necesario tener grandes conjuntos de imágenes con un porcentaje de muestra significativo dentro de cada etiqueta a clasificar, [4]. Estos conjuntos generalmente están conformados con miles (o hasta millones) de imágenes; algunos ejemplos de ellos son: ImageNet-1k[16] y CIFAR-10/100 [14], entre otros.

Por otro lado, la cantidad de filtros que se tenga que incluir y el tamaño de estos son hiperparámetros de la red neuronal, cada una de estas configuraciones de capas y conexiones reciben el nombre de arquitectura. En la actualidad existen arquitecturas con pesos pre-

entrenados los cuales permiten no entrenar los parámetros de la red neuronal desde cero, sino que ayudan a transferir el conocimiento de una red a otra. A esta metodología se le conoce como *transfer learning* y se ha demostrado que disminuye los tiempos de entrenamiento sin reducir el desempeño [48]. Hay muchos tipos de arquitecturas pre-entrenadas, las más conocidas son: ConvNext, ResNet [27], GoogleNet [15], entre otras.

Una de las arquitecturas más utilizadas es la ResNet presentada en 2015 [27], su nombre se debe a una abreviación de Residual Net (Red Residual), y esto es por que utiliza conexiones entre salidas (residuales) y entradas de algunas capas. En [27] se demuestra que estas conexiones evitan el desvanecimiento en el gradiente y hacen que el proceso de aprendizaje sea mucho más rápido, además de que logra que las capas de filtros se entrenen con detalle. El fenómeno de desvanecimiento en el gradiente se debe a que en el proceso de entrenamiento de la red, en la propagación hacía atrás, la actualización de los pesos es muy pequeña en las primeras capas, lo cual retrasa el entrenamiento en estas primeras capas y lo hace mucho más lento. Un ejemplo de la arquitectura ResNet se puede ver en la figura 5. La capas convolucionales se representa por el tamaño del núcleo generalmente cuadrado, el número de núcleos y el salto de píxeles que tiene que dar en la imagen, entonces por cuando se denota una capa convolucional 3×3 64 /1 quiere decir una capa convolucional de 64 núcleos de tamaño 3×3 y salto de 1 en 1.

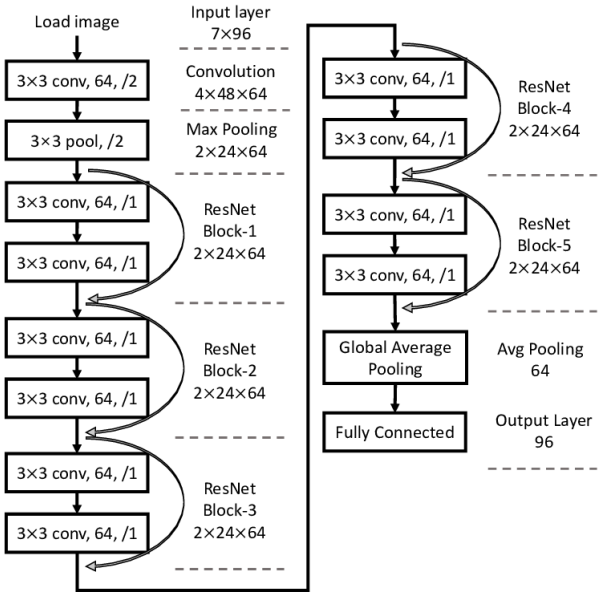

Figura 5: Ejemplo arquitectura ResNet 12.
Fuente: [39]

Asimismo, en [25] se presenta una nueva arquitectura de una red neuronal llamada transformers, la cual está basada en modelos de encoders y decoders, con la diferencia que este

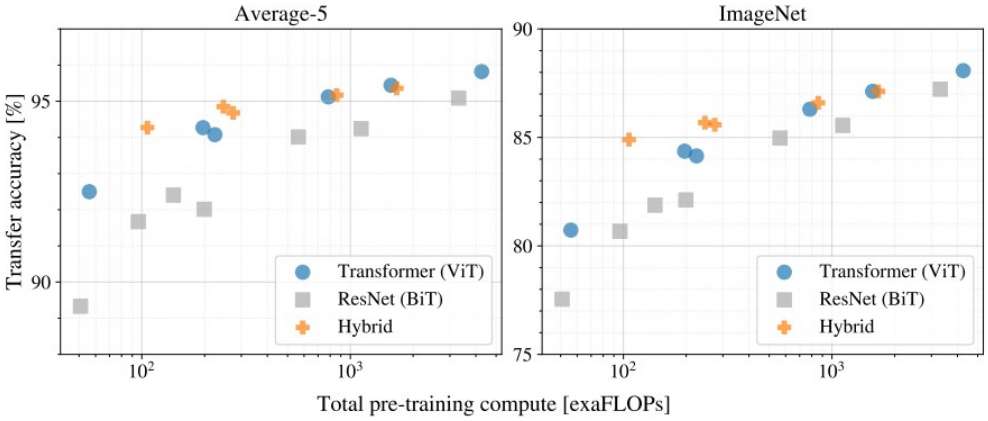

Figura 6: Desempeño de ViT frente a CNN.
Para los conjunto de imágenes ImageNet y Average-5, se observa como el ViT y la arquitectura híbrida logran tener una mejor precisión con la misma cantidad de FLOPS en comparación con la arquitectura ResNet. Fuente: [26]

utiliza en varias capas acumuladas de atención multi-cabeza (*multi-head attention*) tanto como en las entradas como en las salidas. Además, esta nueva arquitectura incorpora un *embedding* posicional, puesto que estos modelos de atención se usan en principio para información de tipo secuencial, entonces este *embedding* se usa para darle dar información dentro de la secuencia que después pasará por el modelo de atención (es decir, tendrá en cuenta que es una secuencia). Esta arquitectura produce generalmente un mejor desempeño en modelos de clasificación [25]. Por otro lado, al incorporar una metodología de *encoder* se logra transformar el espacio donde esta contenida la información de las imágenes el cual es generalmente de menor dimensión. En [26] se presenta una integración entre estos modelos de transformer y el tratamiento de imágenes para clasificación (ViT, Vision transformer), en donde, la metodología propuesta se basa en partir las imágenes de entrenamiento en recuadros de tamaño 16×16 píxeles y después utilizar capas de *Multi-Head attention* para mantener la estructura del transformer. También en [26], se presenta una estructura híbrida entre ViT y CNN, un ejemplo de esta estructura se conoce como ConvNexT [28]. Una desventaja de los modelos ViT, es que la cantidad de parámetros y esfuerzo computacional es mayor en comparación con la CNN, aunque en conjuntos de datos como imagenet-1k se han logrado mejores desempeños tanto con ViT como con los modelos híbridos [26], tal como se puede observar en la figura 6. Una de las arquitecturas de transformer más utilizadas y con un buen desempeño en el Imagenet 1k se presenta en [49] y se conoce como Swim.

Por otra parte, existe una arquitectura convolucional muy utilizada presentada en [28] conocida como ConvNext, y aunque es una arquitectura muy nueva, para el ImageNet 1k demuestra tener un menor error en comparación con el transformer Swim [28]. El ConvNext

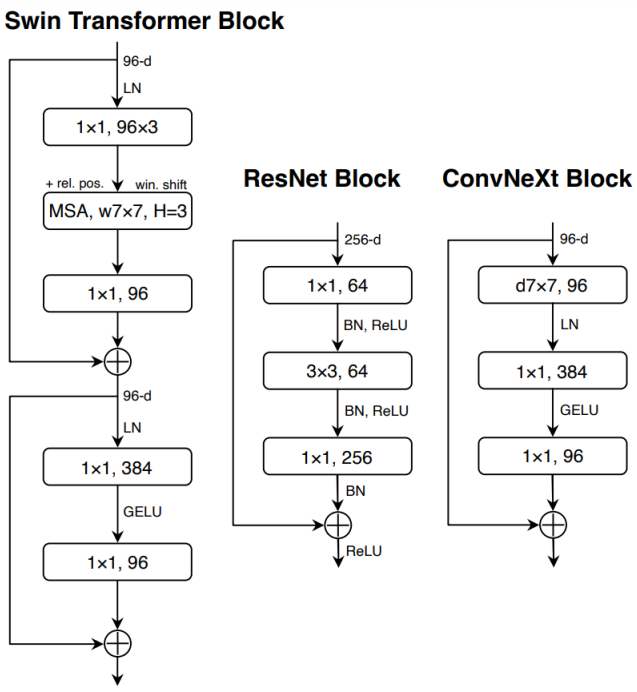

Figura 7: Comparación bloques ResNet, ConvNext y Swim.
Fuente: [40]

es una arquitectura que cuenta con la base de una arquitectura ResNet y se inspira en las arquitecturas de transformer para mejorar su rendimiento. Las mejoras con respecto a la arquitectura ResNet se basa en varios puntos, cómo se describe a continuación:

- **Metodología de entrenamiento:** toma el optimizador AdamW de los transformers.
- **Diseño Macro:** mejora el diseño de la arquitectura en la entrada de la imagen, similar a los recuadros de los transformers.
- **Tamaño de núcleo grandes:** al igual que en los transformers, se proponen que los núcleos de las capas convolucionales sean de por lo menos de 7×7.
- **Micro diseño:** se cambian las funciones de activación de ReLU a GERU, la cual es una variante Gaussiana más suave. Además, se reducen las funciones de normalización y se utilizan normalizaciones simples de los trasnfomers.

En la figura 7 se muestra una comparación entre los bloques de ResNet, ConvNext y Swim, allí se puede apreciar de manera visual las diferencias en cada uno de los modelos. La arquitectura más compleja y con más parámetros es la Swim, seguida de la ConvNext y por último la ResNet. Se observa también, como la arquitectura ConvNext parece una mezcla entre las dos otras arquitecturas es por eso que a veces se lo conoce como arquitectura híbrida.

4.1. Detección de Objetos

Como consecuencia de las redes CNN, se crearon nuevas arquitecturas en las que el objetivo aparte de clasificar el objeto que está la imagen también logra identificar las coordenadas donde se encuentra el objeto y posterior a ello lo clasifica. A esta tarea se le conoce como detección de objetos, un ejemplo de esto se puede ver en la figura 8. La dificultad para el desarrollo de la detección de objetos es el área de la imagen en la que el algoritmo se tiene que enfocar, puesto que de una imagen a otra puede cambiar ésta área.

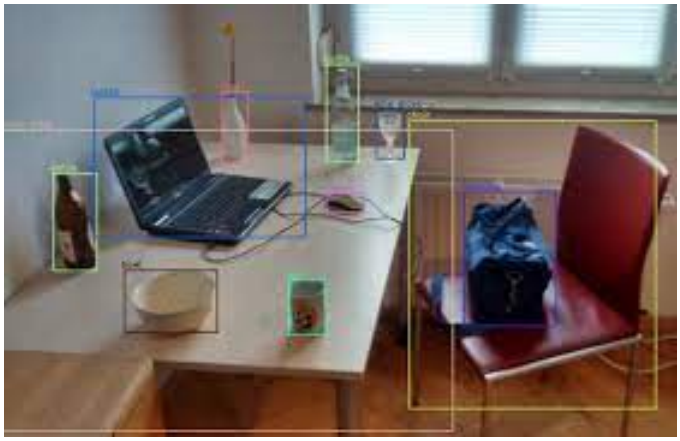

Figura 8: Detección de objetos.
Fuente: [10]

Para desenvolver está tarea se han creado diferentes conjuntos de imágenes entre los que se encuentran: *PASCAL VOC* [41], *ImageNet* [42], *MS COCO* [43], *Places* [44] y *Open Images* [45]. En la actualidad, se expone el rendimiento de un modelo de detección con base en el desempeño en el conjunto *MS COCO*, esto debido a la variedad de imágenes que posee y a los ambientes en los cuales están presentes los objetos. Para determinar el desempeño de un modelo de detección de objetos se define mAP_{COCO} [31], la cual resalta la precisión en la detección de objetos. Para definirla se exponen a continuación los conceptos necesarios: verdadero positivo, falso positivo y precisión promedio.

La salida estándar de un detector aplicado a una imagen de prueba I es un conjunto de tripletas: $\{(b_j, c_j, p_j)\}_j$, donde j es el número de objetos detectados, b_j representan las cuatro (4) coordenadas de la caja encontrada, c_j la categoría o label al que pertenece la caja y p_j la probabilidad asociada a la categoría. Una predicción (b, c, p) es Verdadera Positiva (TP sus siglas en inglés) si:

- La categoría generada c es igual a la categoría de la imagen c_g.
- El área de la intersección, dividida por el área de la unión (IoU sus siglas en inglés) entre la caja producida b y la caja real b_g debe ser mayor o igual que un umbral predefinido

ϵ. Un valor típico para ϵ es 0.5. El IoU es una adaptación del índice de Jaccard [18] para áreas. La fórmula del IoU para este caso es:

$$IoU(b, b_g) = \frac{Area(b \cap b_g)}{Area(b \cup b_g)}$$

En otro caso, la tripleta (b, c, p) es considerada como Falsa Positiva (FP). El nivel de confianza es usualmente comparado con algún umbral β para determinar cuando el label c es aceptado como la clase.

Para una clase de objetos c y una imagen de test I_i, considere $S_i = \{(b_{ij}, p_{ij})\}_{j=1}^{M}$ las predicciones hechas por el detector, ordenadas de mayor a menor por la probabilidad de predicción asociada a la clase (p_{ij}). Cada predicción (b_{ij}, p_{ij}) es o Verdadero Positivo, o Falso Positivo. Ahora bien, variando el umbral de la predicción β se pueden calcular [20]:

- $P(\beta)$ la precisión de las predicciones con p_{ij} mayor a β,
- $R(\beta)$ el recall de las predicciones con p_{ij} mayor a β.

Así pues variando β en el intervalo $[0, 1]$ se construye una curva en el plano real con coordenadas $(P(\beta), R(\beta))$, esta curva se puede asociar a una función $P(R)$ y se obtiene la precisión promedio [21] definida como:

$$AP = \int_0^1 P(R)dR.$$

Finalmente, se define *mAP: mean Average Precision* como la media sobre todas las categorías de las Precisiones Promedio (AP) de cada categoría, es decir:

$$mAP = \frac{\sum_{c \in \text{Categorías}} AP(c)}{\text{Número de Categorías}}$$

Por defecto, cada AP es calculada sobre el umbral defecto de 0.5, ahora bien como parte de la competencia COCO se define la métrica mAP_{COCO} como la media de las mAP de umbrales entre 0.5 y 0.95 con incremento de 0.05, así la definición formal es:

$$mAP_{COCO} = \frac{mAP_{\epsilon=0,5} + mAP_{\epsilon=0,55} + \cdots + mAP_{\epsilon=0,95}}{10}$$

4.1.1. Redes Neuronales para la detección de objetos

En [7] proponen RCNN (*Region-based Convolutional Neural Network)* una metodología para solucionar el problema de detección de objetos y mejorar el tiempo de procesamiento. Esta metodología consiste en proponer aproximadamente 2000 regiones que se pueden ir

fusionando y creciendo, luego éstas regiones pasan por una red convolucional profunda y el resultado es un vector de tamaño fijo que por último pasa por una capa completamente conectada para identificar el objeto.

Para mejorar la capacidad de detección, la velocidad de entrenamiento y la evaluación en el conjunto de prueba, se introdujo la arquitectura Fast RCNN [37]. En esta nueva arquitectura se aprende de manera conjunta la tarea de clasificación y la de regresión con las coordenadas de las cajas que contienen la clase asociada. Para esto se agrega una capa de agrupación de regiones de interés (RoI) entre la última capa convolucional y la primera completamente conectada, con el fin de extraer una característica de longitud fija para cada propuesta de región. Después de las capas completamente conectadas se produce una salida de dos capas hermanas: por un lado las probabilidades obtenidas a partir de un capa softmax para la clasificación del objeto encontrado en la región y por otro lado la regresión de la región. En comparación con su predecesora es tres (3) veces más rápida en entrenamiento y diez (10) veces más rápida en evaluación.

Aunque Fast RCNN acelera considerablemente el proceso de detección, aún depende de propuestas de regiones externas, lo cual significa una reducción en velocidad. Con el avance en el estudio de arquitecturas convolucionales y su capacidad de detectar objetos [47], habilidad que se debilita al ser sometidas a las capas fully connected, se introduce una capa CNN para realizar la tarea de proponer regiones. La arquitectura Faster RCNN [30] ofrece una eficiente y precisa Red de Propuestas Regionales (RPN sus siglas en inglés) para la generación automática de regiones, un ejemplo de está se puede ver en la figura 9.

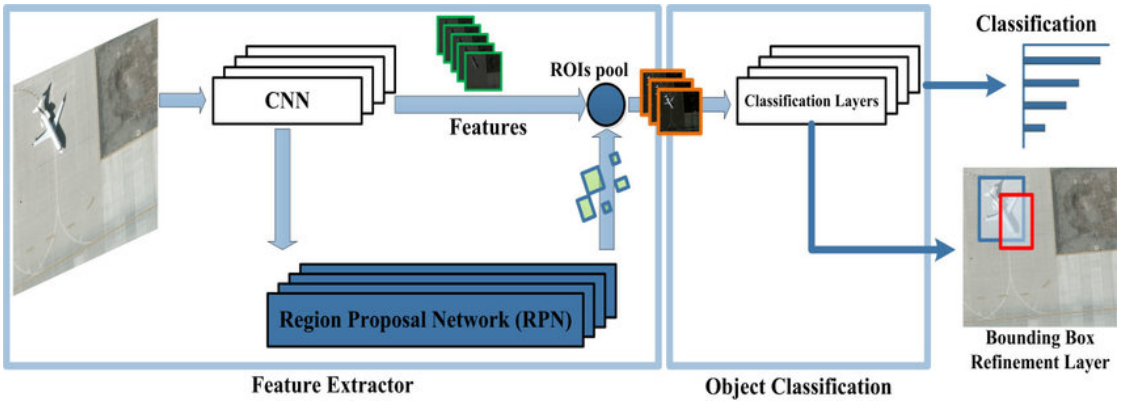

Figura 9: Arquitectura Faster RCNN.
Fuente: [38]

Esta red genera k cajas de referencia (conocidas como *anchors*) de diferentes radios y escalas en cada ubicación de la extracción de características de la capa convolucional intermedia. Las posiciones de estás cajas son independientes de la imagen, pero los vectores

de características extraídos de las cajas si dependen de la imagen. Así, a la salida de cada caja se le asigna un vector que alimenta dos capas fully connected, una para clasificación del objeto contenido y la otra para la regresión de la caja. Esto tiene como consecuencia que su función de pérdida tenga en cuenta el desempeño de la capa RPN y de la salida, así pues se calcula como la suma de cuatro (4) funciones de pérdida:

1. Binary cross entropy ([23]) entre la predicción hecha para las cajas generadas aleatoriamente por la capa RPN si contenía o no algún objeto clasificado.
2. Smooth L1 Loss ([22]) con beta 0.9 para la regresión de las cajas generadas aleatoriamente por la capa RPN y las regiones rectangulares en donde hay un objeto en común.
3. Cross entropy ([23]) entre el vector de probabilidades de que la caja generada en la última capa fully connected contenga el objeto asociado a cada label y los etiquetas reales.
4. Smooth L1 Loss ([22]) con beta 0.9 para la regresión de las cajas generadas en la última capa fully connected y las cajas reales con el mismo label asociado.

La complejidad del entrenamiento de Faster RCNN es una desventaja, además de la demora en las predicciones, lo que dificulta usarlo en aplicaciones de tiempo real y dispositivos móviles. Esto último motiva el desarrollo en paralelo de sistemas unificados o de un paso, que no se basan en la proposición de regiones como RCNN, Fast RCNN y Faster RCNN, sino que desarrollan estrategias para la detección unificada. Modelos que desarrollan esta metodología son DetectorNet (Szegedy et al. 2013), OverFeat Sermanet et al. (2014), YOLO (Redmon et al. 2016), YOLO v2 y YOLO9000 Redmon and Farhadi (2017), SSD (Liu et al. 2016) y CornerNet, Law and Deng (2018)

YOLO por sus siglas en inglés *You only look once* es el modelo más representativo de los sistemas unificados. Es usado en algunos automóviles para la conducción automática mediante la identificación (y segmentación) de objetos. YOLO se basa en partir la imagen en regiones y a estas les calcula el recuadro donde esta el objeto y la probabilidad de que sea un objeto, de esta manera todas los recuadros que se calculan están ponderados por una probabilidad [8]. Un ejemplo de la arquitectura de YOLO se puede ver en la figura 10.

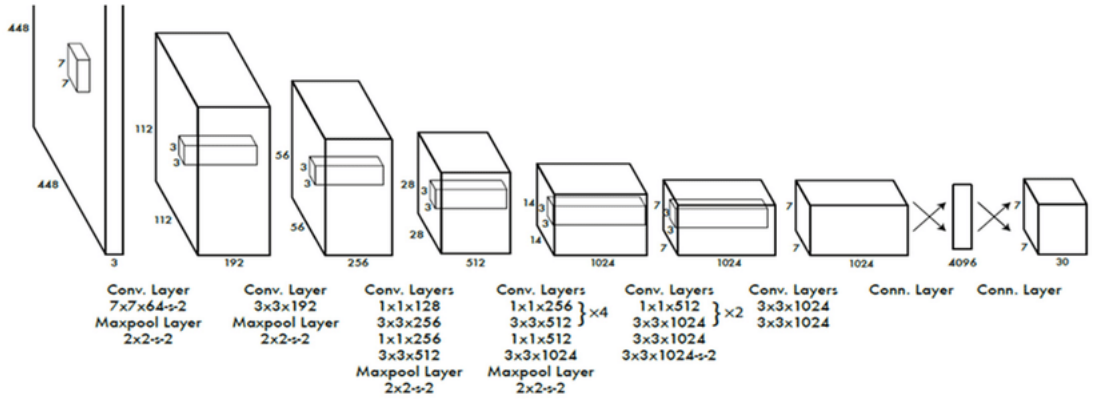

Figura 10: Arquitectura YOLO.
Fuente: [12]

5. Metodología

Las características de los conjuntos de datos usados para el entrenamiento de los modelos de detección de objetos y de clasificación de objetos son diferentes. El primero requiere para su entrenamiento una colección de imágenes en las cuales puede o no contener los objetos a identificar, su etiqueta y además de la ubicación exacta de cada uno mediante dos coordenadas del rectángulo que encierra cada objeto en la imagen (basta con la esquina inferior izquierda y la esquina superior derecha). Por otro lado, para el modelo de clasificación el conjunto de datos debe estar constituido por imágenes del objeto a clasificar junto con su etiqueta. Ahora bien, como se describe anteriormente en la tarea de detección se podría desarrollar la tarea de clasificación si se dispone de un conjunto de imágenes con las categorías objetivo. En vista de que no existe un conjunto de datos que cumpla los dos requerimientos descritos anteriormente, se ha entrenado en paralelo un modelo de detección con un conjunto de datos tomado de [29], mientras que para la tarea de clasificación los autores han creado el conjunto de imágenes. Una vez encontrados los modelos que satisfacen los requerimientos de calidad y precisión definidos, se unen los modelos de manera secuencial, para que en una primera instancia se identifiquen las regiones en las cuales están presentes los tornillos, y posteriormente sea determinada la referencia de cada tornillo detectado.

5.1. Conjunto de datos para detección

El conjunto de datos fue tomado de [29][1]. Contiene 348 imágenes en formato `.png` tamaño promedio de 3.6MB, resolución de 300ppp y dimensiones $1920px \times 1440px$, junto con un diccionario que relaciona cada imagen con las coordenadas de los rectángulos orientados en la dirección en la que aparece el objeto como se puede apreciar en la figura de la izquierda 11 y la etiqueta de cada objeto. En las imágenes aparecen 13 referencias diferentes de tornillos, entre ellas tornillos roscados (objetivo principal del proyecto), tornillos autoperforantes (usados para perforar madera o lamina) y tuercas. Las fotos vienen con fondo de madera amarilla clara y la foto tomada siempre desde la perspectiva superior. Un ejemplo de este conjunto de imágenes, junto con las regiones transformadas a como son usadas en el trabajo se puede ver en la figura de la derecha 11.

[1] `https://www.mvtec.com/company/research/datasets/mvtec-screws`

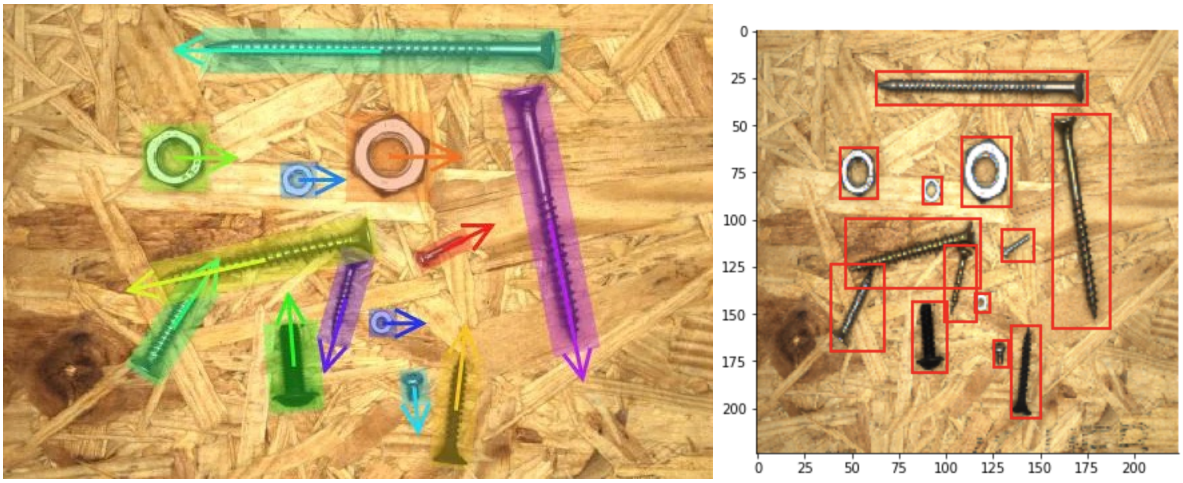

Figura 11: Imagen del conjunto de datos para detección.
a) Imagen del dataset con cajas originales, orientadas en dirección del tornillo. b) misma imagen reescalada a 256×256 y con cajas transformadas a formato de entrenamiento.
Fuente: a) Recuperado de `https://www.mvtec.com/company/research/datasets/mvtec-screws`, b) autores

5.2. Conjunto de datos para clasificación

Se ha construido desde cero por parte de los autores, un conjunto de datos de imágenes compuesto por 100 fotos de tornillos de cada referencia de rosca que va a hacer parte del estudio:

- Para las roscas métricas se han considerado las referencias: 6MM P1.0, 7MM P1.0, 8MM P1.0, 8MM P1.25, 10MM P1.0, 10MM P 1.25, 10MM P1.5, 12MM P1.25, 12MM P1.5 y 12MM P1.75.
- Para las roscas americanas unificadas se ha considerado las referencias: 1/4 RO, 1/4 RF, 5/16 RO, 5/16 RF, 3/8 RO, 3/8 RF, 7/16 RO, 7/16 RF, 1/2 RO y 1/2 RF.

La parte roscada de los tornillos usados para la creación de este conjunto de imágenes digitales tiene una longitud de una pulgada (25.4mm) y no poseen parte lisa. Las fotos fueron tomadas con fondos cotidianos, es decir, sobre alguna superficie (piso, tapete, mesa, sofá, silla, bolsa, ventana, etc) o sujeto en la mano y en diferente posiciones tales como inclinadas o laterales. Se incluyen algunas con vista superior para generalizar el modelo cuando sea entrenado, pero en pruebas previas que se hicieron para la viabilidad del proyecto son las que menor precisión tienen. Esto debido a que con este tipo de fotos no se puede extraer información suficiente del tipo de rosca, sólo posiblemente del diámetro y se tienen hasta 6 referencias –de las consideradas en el trabajo– de tornillos diferentes con el mismo diámetro (roscas métricas de 12MM y rosca americana de 1/2 pulgada).

De las 100 fotos de cada referencia fueron tomadas 50 de cada una con la cámara posterior de los celulares de los autores del proyecto iPhone X y Motorola G60. Las fotos tomadas con el iPhone X fueron procesadas en un computador MacBook Pro para ser convertidas de formato `.HEIC` a formato `.jpeg` conservando su metadata y considerando el tamaño máximo de conversión, luego de este proceso el tamaño promedio resultó en 200Kbs, la resolución obtenida fue de 72ppp y las dimensiones obtenidas fueron de $960px \times 1280px$. Las fotos tomadas con el celular Motorola se guardaron en formato `.jpg` directamente, con un peso promedio de 3100 Kbs, resolución de 108 ppp y dimensiones de $1800px \times 4000px$.

5.3. Software

Para la ejecución de los algoritmos que se han mencionado en el proyecto los autores han adquirido una membresía en la plataforma Paperspace en donde se han creado jupyter notebooks de Python con los frameworks pre instalados de *Deep Learning* PyTorch para la tarea de detección y FastAI, que es un framework de alto nivel construido sobre PyTorh, para la de clasificación junto con la librería Timm para la descarga de los modelos pre entrenados. Los notebooks han sido ejecutados con una tarjeta gráfica NVIDIA RTX A4000 con 16GB GPU, 8 CPU y 45 GB de RAM. Adicional del framework pre instalado, se ha hecho uso de librerías de la versión 0.3.0 del repositorio `https://github.com/pytorch/vision.git`.

5.4. Modelos y su entrenamiento

El desarrollo del proyecto se divide en dos tareas principales de visión por computador: detección de objetos (roscas de tornillos) y clasificación (referencia precisa de cada rosca). Para ambas tareas se va a utilizar la técnica de transferencia de conocimiento (*transfer learning*) para redes neuronales, con el objetivo de reducir el tiempo de entrenamiento.

5.4.1. Modelo de detección de objetos

Este modelo recibe una imagen que contiene una o más roscas de tornillos y como salida se obtendrá un conjunto de cuádruplas de números reales (x_1, y_1, x_2, y_2) los cuales indican la posición en píxeles de las esquinas superior izquierda (x_1, y_1) e inferior derecha (x_2, y_2) del rectángulo que bordea cada objeto deseado en la imagen dada.

Para su entrenamiento se establece una secuencia de transformaciones a las imágenes, por defecto las siguientes son efectuadas al cargar la imagen:

1. Transformar a escala de grises
2. Reescalar las dimensiones a $256px \times 256px$, junto con las coordenadas de los rectángulos.

3. Eliminar las 13 etiquetas de los objetos presentes en las imágenes y sólo establecer una. Así en cada imagen de entrenamiento los objetos presentes serán marcados como tornillo (1) o no tornillo (0).

Posteriormente, son ejecutadas las siguientes técnicas de data augmentation que se aplican con una probabilidad de 0.5 por defecto (caso sea diferente se menciona) en el orden dispuesto a continuación:

1. Desenfoque Gaussiano con valores (1,3).
2. Cambio de brillo y contraste aleatorio con cambios máximo de 0.4 en cada uno y una probabilidad de que sea aplicado de 0.7.
3. Eliminación aleatoria de píxeles.
4. Corte aleatorio de la imagen sin cortar ninguno de los rectángulos incluidos en la imagen e interpolación cúbica para extrapolar la imagen recortada.
5. Giro vertical y horizontal de la imagen.

Luego de componer la transformación básica con las transformaciones aleatorias se obtienen imágenes como las mostradas en la figura 12.

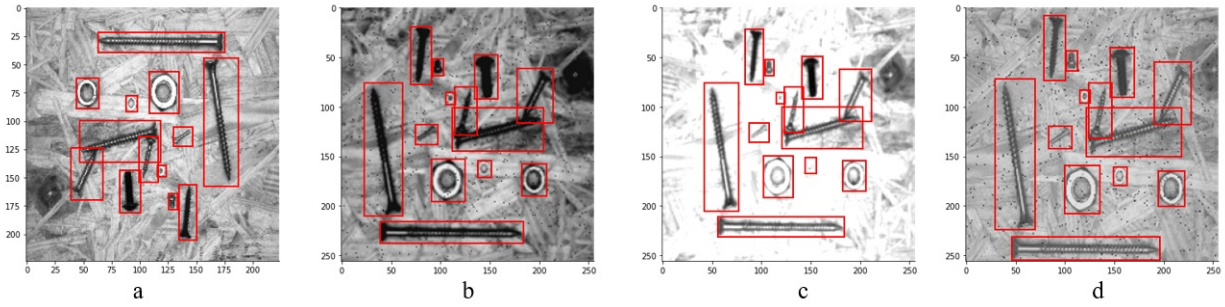

Figura 12: Data augmentation en el conjunto de imágenes para detección.
Imagen a) relacionada a la imagen original solo reescalada y en escala de grises, b), c) y d) imágenes transformadas usando las técnicas de data augmentation.
Fuente: Autores

Para el entrenamiento son usadas las arquitecturas:

1. Faster RCNN [30]
    a) Backbone ResNet50
    b) Backbone MobilNet v2
2. Faster RCNN v2 [46]
    a) Backbone MobilNet v2

Cada arquitectura fue entrenada durante 100 épocas, el optimizador de entrenamiento usado fue el Gradiente Descendiente Estocástico (`torch.optim.SGD`) con los siguientes hiperparámetros: tasa de aprendizaje (`lr`) de 0.001, momentum de 0.9 y decaimiento de peso de 0.001; adicionalmente fue configurado un scheduler StepLR con un gamma de 0.5 y actualización de los pesos cada 10 épocas. El gráfico del learning rate vs el número de epoch usado en cada una de las redes se puede observar en la figura 13.

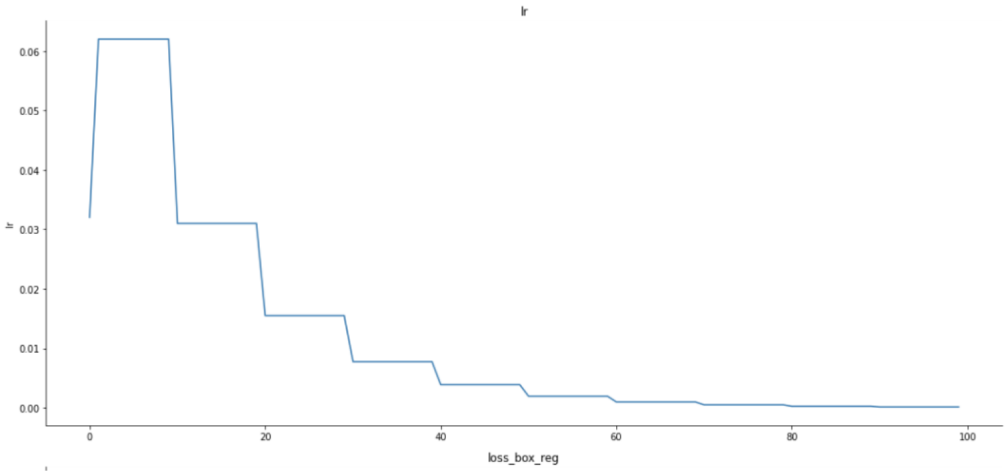

Figura 13: Learning Rate per epoch
Fuente: Autores

5.4.2. Modelo de clasificación

La entrada de este modelo viene dado por los recortes de los rectángulos que se identificaron en el modelo anterior, en donde se espera que sea una imagen limpia del tornillo. La salida del modelo será una etiqueta de las usadas para el estudio: 6MM P1.0, 7MM P1.0, 8MM P1.0, 8MM P1.25, 10MM P1.0, 10MM P 1.25, 10MM P1.5, 12MM P1.25, 12MM P1.5, 12MM P1.75, 1/4 RO, 1/4 RF, 5/16 RO, 5/16 RF, 3/8 RO, 3/8 RF, 7/16 RO, 7/16 RF, 1/2 RO y 1/2 RF

Dado que se van a utilizar arquitecturas pre entrenadas de redes neuronales, es necesario asegurarse que el tamaño de las imágenes de entrada sea igual con el que la red se pre entreno. Para el caso de las arquitecturas ResNet, ConvNext y Swim, las cuales son las redes que se probaron, el tamaño de entrada es de $244px \times 224px$. Después de esto se procedió a hacer diferentes transformaciones de las imágenes con el objetivo de aumentar el tamaño del conjunto de entrenamiento (data augmentation). Además que es una buena técnica para evitar el sobre ajuste de estos modelos. Las transformaciones que se utilizaron conjuntamente fueron

- Giro vertical y horizontal de la imagen.

- Giros aleatorios de hasta 45 grados de inclinación con probabilidad (0.7)
- Cambios en el brillo y el contraste con probabilidad (0.9)
- Transformaciones matemáticas afines
- Cortes aleatorios con probabilidad (0.7)

Para el entrenamiento de las redes se utilizó la metodología de congelar el cuerpo de la red durante cierta cantidad de épocas y solo entrenar la última parte relacionada con la clasificación conocida como *head*. Esto a razón de que los pesos de del cuerpo viene con valores que ya fueron pre entrenados, mientras que la cabeza no. Al entrenar conjuntamente el cuerpo con la cabeza conjuntamente puede generar una demora en el tiempo de entrenamiento, este procedimiento usualmente se considera dentro del *fine-tuning*.

Red Neuronal	**Épocas con el Cuerpo congelado**	**Épocas de entrenamiento de toda la red**
ResNet	20	70
ConNext	15	50
Swim	15	50

Cuadro 1: Cantidad de épocas de entrenamiento del modelo de clasificación

En la tabla 1, se observa la cantidad de épocas de entrenamiento que se utilizó para cada modelo. La diferencia entre la cantidad épocas con las que se entrenó cada red es debido a que la arquitectura ResNet es mucho más rápida de entrenar en comparación con la arquitectura Swin y ConvNext, por lo que se utilizaron más épocas para la arquitectura ResNet.

Asimismo, el framework de fastai utiliza el optimizador de Adam por defecto, una tasa de aprendizaje adaptativa para que la convergencia del modelo sea mucho más rápida(Metodología *warm up*) y una función de perdida de entropía cruzada. Por último, el conjunto de datos se dividió en conjunto de entrenamiento con un 70 % (1400 imágenes), y conjunto de validación con 30 %(600 imágenes) y se entrenó en batchs de tamaño de 64 imágenes.

5.4.3. Evaluación

Para el modelo de detección se elegirá el modelo con mayor métrica de desafío principal del dataset COCO [31] en el subconjunto de datos destinado para prueba. Esta es calculada como el *mean Average Precision* (mAP) considerando cajas de todos los tamaños, iterando los valores del umbral del índice de Jaccard ([18]) entre 0.5 y 0.95 (con pasos de 0.05) y considerando un máximo de objetos detectados de 100 (las fotos usadas para entrenamiento tienen a lo más 15 objetos por foto, por esto no es necesario limitar más objetos). Una vez entrenado y escogido el mejor modelo se procede a evaluarlo en el conjunto de imágenes

creado para la tarea de clasificación con el criterio que con un umbral de score de 0.95 identifique al menos en el 90 % de las fotos la localización del tornillo correctamente (está última prueba se hace de manera manual sobre un subconjunto aleatorio de 200 imágenes).

Para el problema de clasificación se va a escoger el modelo que tenga el mejor desempeño con base en las métricas de área bajo la curva ROC ([19]) y *accuracy* ([50]). El cálculo de estas métricas se va a hacer por cada referencia de tornillo, así como de manera global.

5.4.4. Interfaz

Una vez los modelos de detección y clasificación han sido entrenados, con buenos desempeños en los dos modelos se procede a crear una interfaz gráfica, gracias a que el programa que se utilizó para entrenar los modelos fue Python, se eligió la librería Dash para crear una interfaz amigable con el usuario la cual tendrá como objetivo que se pueda cargar una imagen desde la maquina donde se este ejecutando y en tiempo real sea capaz de detectar el rectángulo de la imagen donde se encuentra el tornillo y clasificarlo en alguna de las categorías con las que fue entrenadas el modelo, en caso de que el modelo de detección no detecte algún tornillo entonces se pasará al modelo de clasificación de toda la imagen. Al igual si para el modelo de clasificación no se tiene un resultado certero entonces la salida será que con esa imagen no es posible llegar a decir cuales son las características del tornillo.

Como paso adicional, si en la imagen se detecta más de un tornillo, cada trozo de imagen identificado se pasará por el modelo de clasificación y se dará una predicción a cada caja y esto también se verá reflejado en la salida.

6. Resultados y Discusión

6.1. Resultados

6.1.1. Modelo de detección

En el entrenamiento del modelo de detección el éxito radica en las técnicas de data augmentation que se realizan al conjunto de entrenamiento. En la figura 14 se puede observar el comparativo de la métrica mAP_{COCO} aplicado al subconjunto de datos usado para test, y calculado para las tres arquitecturas consideradas:

1. Faster RCNN [30]
    a) Backbone ResNet50
    b) Backbone MobilNet v2
2. Faster RCNN v2 [46]
    a) Backbone MobilNet v2

Este comparativo se hace a lo largo de las 100 épocas en las que se entrenaron los modelos y usando la configuración de learning rate expuesta en la metodología.

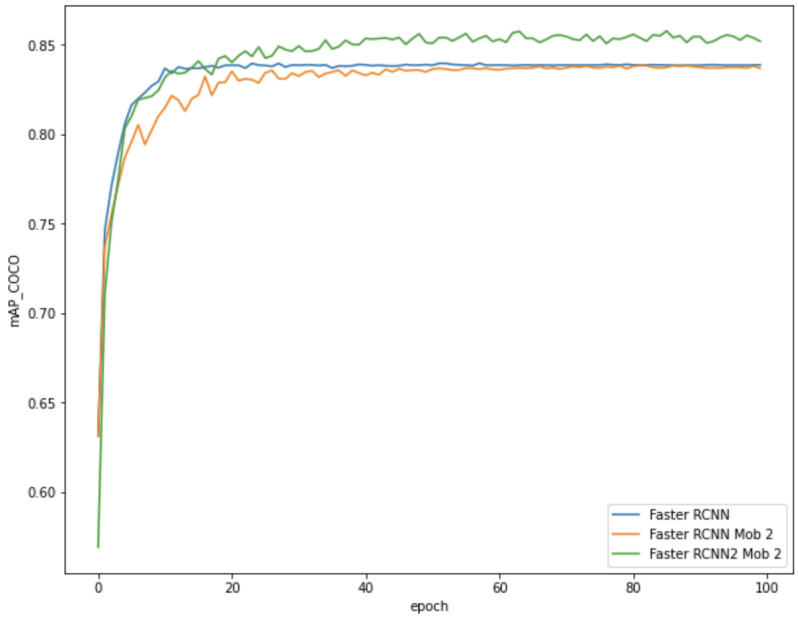

Figura 14: mAP_{COCO} de los modelos de detección entrenados.
Fuente: Autores

En la tabla 2 se muestra el comparativo de las métricas mAP_{COCO} entre los tres modelos en la época 100 de entrenamiento sobre el mismo subconjunto test. Las métricas mAP^{small}_{COCO},

mAP^{medium}_{COCO} y mAP^{large}_{COCO} son calculadas como la métrica mAP_{COCO} pero restrictas a regiones pequeñas, medianas y grandes respectivamente. Una región es considerada como pequeña si su área es menor o igual a $32px^2$, como mediana si su área está entre $32px^2$ y $96px^2$, y como grande si es mayor o igual a $96px^2$.

Modelo + Backbone	mAP_{COCO}	mAP^{small}_{COCO}	mAP^{medium}_{COCO}	mAP^{large}_{COCO}
Faster RCNN + ResNet50	0.8387	0.8047	0.8711	0.8846
Faster RCNN + MobilNet v2	0.8368	0.804	0.8689	0.8845
Faster RCNN v2 + MobilNet v2	**0.8519**	**0.8227**	**0.8831**	**0.9208**

Cuadro 2: Comparativo de las métricas de detección para los modelos usados en la tarea de detección.
El valor de cada métrica es calculada sobre el modelo entrenado en la época 100 y sobre el subconjunto de imágenes usadas para test a lo largo del entrenamiento.

Cada uno de los modelos tuvo un buen desempeño y se puede notar como se estabilizan luego de la época 50. Con base en el desempeño superlativo del modelo Faster RCNN con backbone MobileNet v2 y además de aprobar los criterios de aceptación en el conjunto de clasificación (predecir las cajas de al menos el 90 % de cada categoría del conjunto de clasificación), se elige este para desempeñar la tarea de detección. En la figura 15 se puede observar las predicciones que estableció el modelo para la imagen del conjunto de entrenamiento que ha sido discutida a lo largo del documento, así como de dos fotos del conjunto de entrenamiento del modelo de clasificación.

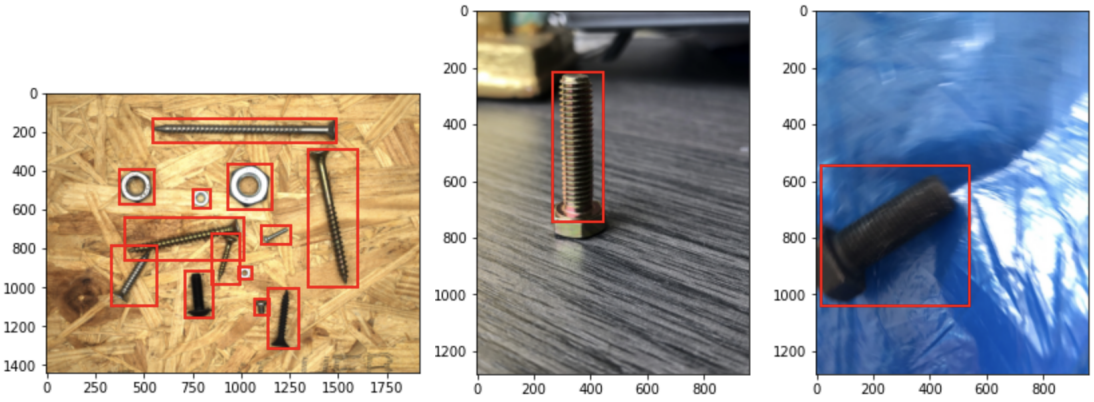

Figura 15: Predicciones del modelo de detección.
Fuente: Autores

Aunque el tiempo de entrenamiento del modelo elegido dobla el de sus competidores, se ha escogido por tener el mayor valor de la métrica mAP_{COCO} en el conjunto de entrenamiento, así como cumplir el criterio establecido por los autores en el conjunto de datos creado para la clasificación.

6.1.2. Modelo de clasificación

Una vez se hizo la detección de objetos para el conjunto creado por los autores con el modelo de detección entrenado, se procede a entrenar las redes ResNet, ConvNext y Swim. El entrenamiento se hace con la metodología de *Fine-Tuning* es por ello que en la figura 16 se ve un salto en la función de perdida para cada red puesto que en esa época es donde se descongeló el cuerpo y la actualización de pesos se hizo para toda la red.

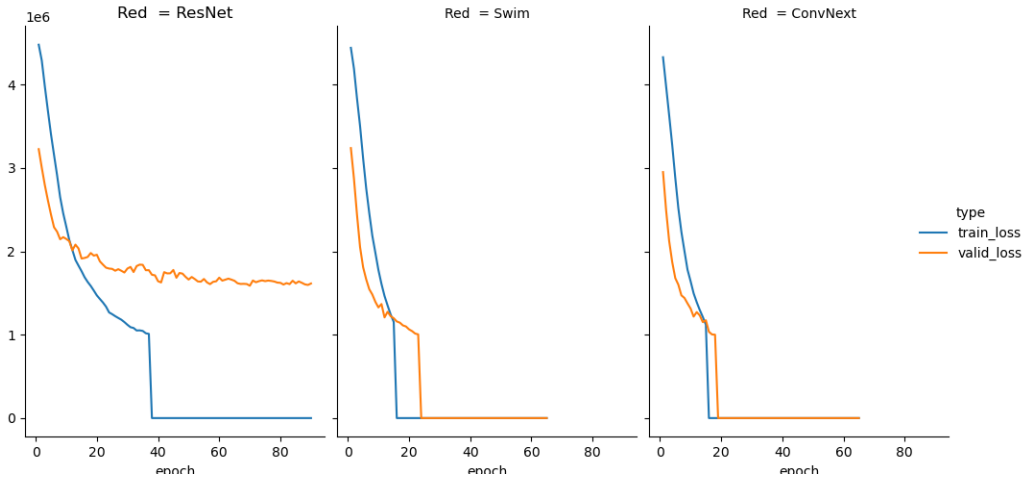

Figura 16: Perdida para el grupo de entrenamiento vs el de validación para cada Red.
Fuente: Autores

Por otro lado se ve que la arquitectura ResNet aunque tuvo un salto en la función de perdida para el entrenamiento no lo tuvo para el conjunto de validación. Para las arquitecturas Swim y ConvNext el salto si se ve tanto como en el conjunto de entrenamiento como el de validación, aunque este segundo con cierta demora. Siguiendo en línea en la figura 17 se ve el porcentaje de precisión de la red para el conjunto de validación, como era de esperarse, la arquitectura ResNet tiene un desempeño muy por debajo de las otras dos, mientras que las arquitecturas Swim y ConvNext tiene valores y comportamientos muy similares, la arquitectura con mayor precisión es la Swim.

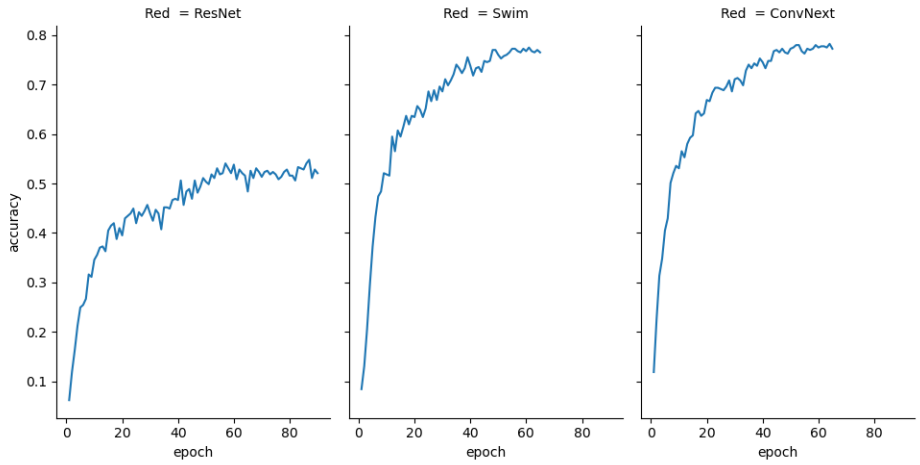

Figura 17: Precisión de entrenamiento en el grupo de validación para cada Red.
Fuente: Autores

El comportamiento del área bajo la curva ROC(Figura 18) refleja lo que ya se había mostrado en otras gráficas, la arquitectura ResNet tiene un bajo desempeño respecto a las otras dos y la arquitectura Swim y ConvNext tienen comportamientos y valores muy similares.

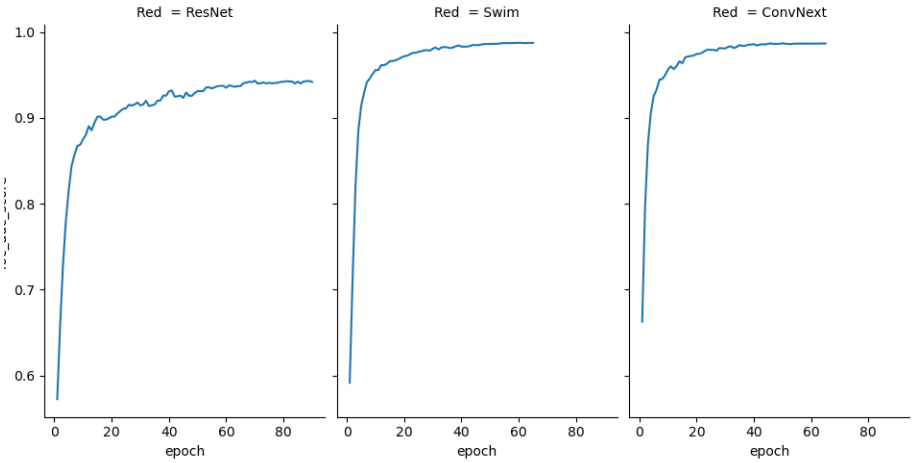

Figura 18: Área bajo la curva para el grupo de validación para cada Red.
Fuente: Autores

La diferencia entre la arquitectura Swim y ConvNext finalmente llega por su tiempo de procesamiento, mientras que el tiempo total empleado para entrenar la red Swim fue de 44 minutos 43 segundos, para la red ConvNext fue de 116 minutos 2 segundos. En la tabla 3 se puede observar el resumen de los entrenamientos para cada red.

Red Neuronal	Mejor Presición	AUC	Época	Tiempo
ResNet	54.84 %	0.9423	87	44 min 44s
ConNext	78.27 %	0.9867	64	116 min 2s
Swim	**77.53 %**	**0.9874**	**61**	**44 min 43 s**

Cuadro 3: Resumen desempeño de los modelos en el modelo de clasificación

6.1.3. Interfaz

Después de encontrar los modelos con el mejor ajuste, se pasaron a producción y se montaron en una interfaz de Dash. A continuación una pantalla de ella

Figura 19: Interfaz con los modelos de detección y clasificación en producción
Fuente: Autores

6.2. Discusión

Los modelos de detección son capaces de identificar la posición de un objeto en una imagen, así como de determinar la categoría a la que pertenece. Sin embargo, como se menciona en la sección 5 no existe un conjunto de entrenamiento que cumpla con los requerimientos necesarios para llevar a cabo la tarea de detección y cuyos objetos sean tornillos con las referencias establecidas en el objetivo general (sección 2.1). Por está razón, se dividen las tareas en identificación (sección 6.1.1) y clasificación (sección 6.1.2), usando para la primera el conjunto de datos tomado de [29], mientras que para la segunda se crea un conjunto de imágenes.

El modelo de detección es capaz de identificar tornillos que no hacían parte de su conjunto de imágenes de entrenamiento, esto hace que sea capaz de identificar los tornillos de las imágenes creadas por los autores. A su vez, el modelo de clasificación tiene una precisión de 80 % en las imágenes usadas para validación muy superior al 5 % que se obtiene de la clasificación aleatoria. Al unir los dos modelos la precisión obtenida es de 75 % sobre el conjunto de validación usado para el modelo de clasificación.

El conjunto de entrenamiento del modelo de detección no tiene relación directa con las imágenes creadas para el modelo de clasificación, de hecho como se observo en la descripción del mismo, dentro de las imágenes se incluyen tipos de tornillos con roscas diferentes, otros tipos de cabeza, composición entre parte lisa y parte roscada diferente, además de incluir tuercas. Por otro lado, el fondo es el mismo para todas: madera clara. Esto conlleva a que las imágenes deban ser sometidas a diferentes preprocesamientos en el entrenamiento de los modelos usados con el fin de evitar un sobreajuste y que al finalizar su entrenamiento sea capaz de identificar tornillos en ambientes diferentes.

Las diferentes arquitecturas usadas en el proceso de entrenamiento del modelo de detección obtenían resultados similares tanto para las métricas de perdida así como en las métricas de evaluación en las imágenes de validación. Sin embargo, cuando el modelo era usado para detectar objetos en el conjunto de datos creado por los autores, no se tenían resultados favorables y con probabilidad de presencia de objeto debajo de 0.5, este proceso de prueba error conllevo a la inclusión de las etapas de data augmentation en el entrenamiento. Cada una de éstas fue probada con la arquitectura básica Faster RCNN y se iban incluyendo transformaciones a las imágenes si se detectaba un fallo en las imágenes construidas. Un ejemplo de lo mencionado anteriormente sucedió cuando se detectaron fotos en los que la figura se veía distorsionada. Con el fin de simular este efecto y el modelo sea capaz de determinar la figura del tornillo inclusive en imágenes que no tienen el enfoque apropiado fue incluido el desenfoque Gaussiano [24]. Una modificación importante para mejorar el desempeño del modelo en las imágenes creadas, fue la no distinción entre tipos de tornillos (es decir, eliminar los 13 etiquetas y sólo considerar presencia o no del objeto).

En cuanto a las arquitecturas en la mayoría de los escenarios y configuraciones entrenadas se obtuvo un mAP_{COCO} (ver sección 4.1) superior al 80 % y no se logró superar el máximo de 82.5 %. Sin embargo, en todos los modelos y configuraciones entrenadas el conjunto de validación del modelo de detección tenía muy buenas predicciones, mientras que cuando se evaluaba en el conjunto usado para la clasificación no se tenían predicciones buenas y con una probabilidad menor a 0.5 de que el objeto identificado fuera un tornillo. Esto último fue el factor decisivo a la hora de elegir los modelos, es decir, por un lado que el modelo fuera capaz de predecir sobre el conjunto de entrenamiento, pero que a su vez, el modelo

de detección fuera capaz de generalizar y detectar tornillos en las imágenes creadas para la clasificación.

Por otro lado, al incluir las formas de los tornillos que se encontraban en el conjunto de datos extraído para detección, el modelo es capaz de detectar tornillos de diferentes largos y con cabezas diferentes. Esto permite que la variedad de tornillos que se pueda detectar sea mayor.

Se encuentra en [33] una adaptación del modelo Faster RCNN con el backbone MobileNet v2 con diferentes configuraciones de anchura y que posee un mAP_{COCO} de 92.4 %, sosteniendo su buen desempeño en la extracción de features de alta calidad. Caso similar ocurre en [34] en donde se compara el modelo Faster RCNN con backbone por default ResNet 50 contra el SSD con backbone MobileNet v2 en donde se compara la velocidad de predicción dando 15ms vs 10ms y teniendo un mAP_{COCO} en el conjunto de datos $COCO$ de 28 vs 22. Estos trabajos y el hecho de que los modelos YOLO tengan un mal desempeño para identificar cajas pequeñas, motivó que se desarrollará el entrenamiento usando la arquitectura Faster RCNN. Se sabe que las ventajas del YOLO es su velocidad de predicción lo que permite ser usado en tareas como identificación en video, esto compensado con un bajo desempeño en cajas pequeñas como se puede evidenciar en trabajos recientes en donde se comparan las dos arquitecturas ([35], [36]). Ahora bien, en este trabajo se prioriza la precisión sobre el tiempo de predicción, pues el conjunto de entrenamiento no está compuesto por imágenes que serán usadas en producción, como se especificó en la etapa de entrenamiento, además se espera en un trabajo futuro escalar este desarrollo para que cualquier persona pueda subir imágenes sin importar su tamaño y sea capaz de detectarlo y clasificar su rosca.

Con relación al modelo de clasificación, las arquitecturas que se probaron para este se escogieron debido a sus desempeños en la clasificación del conjunto ImageNet 1k. Se comenzó con una arquitectura ResNet la cual aunque no tiene el mejor desempeño respecto al accuracy, pero tiene una arquitectura liviana. Para el caso de clasificación con las imágenes de los autores completas, el desempeño de este tipo de Red fue muy bueno, llegando casi a igualar a arquitecturas como ConvNext o Swim. Sin embargo, cuando se entrenó el modelo ResNet con las imágenes con los tornillos recortados su desempeño bajó considerablemente por lo que se descartó. La segunda opción considerada fue la arquitectura ConvNext, que desde el conjunto de entrenamiento con las imágenes completas mostraba un muy buen porcentaje de accuracy, y aunque el desempeño de este modelo disminuyó cuando se probó en imágenes cortadas, el porcentaje de accuracy siguió por encima del 70 % y su correspondiente métrica de área bajo la curva ROC de 0.98. Aunque ya se tenía un buen resultado con arquitecturas CNN, se decidió probar una arquitectura tipo transformer de tipo Swim , la cual es una arquitectura liviana dentro de los transformers con excelente desempeño. Esta arquitectura

demostró ser la mejor tanto en el entrenamiento con imágenes completas como recortadas, mostrando los mejores valores de accuracy y área bajo la curva ROC. Al tener un tamaño en disco y un tiempo de ejecución menor que la arquitectura ConvNext, en este trabajo se decidió finalmente utilizar la arquitectura Swim.

Es importante que los modelos desarrollados, sean fáciles de ejecutar para personas que no estén en el entorno académico. Más aún, cuando el objetivo es ayudar a suplir tareas cotidianas para ciertos nichos de la población, y que no se necesiten máquinas potentes con muchas características especiales para que se pueda ejecutar los modelos. Es por esto que la interfaz de usuario tiene que tener una buena experiencia y un buen diseño para que pueda ser utilizar satisfactoriamente por el usuario final. Cabe destacar que si el usuario desea conocer las características de 5 tornillos no es necesario subir 5 imágenes diferentes si no una sola imagen con los 5 tornillos, ya que se han desarrollado modelos de detección que son capaces de dar un resultado para cada uno de ellos.

7. Conclusiones

En este trabajo se ha logrado desarrollar una tarea de identificación y clasificación de tornillos con las roscas consideradas utilizando técnicas de aprendizaje profundo de máquina. Para la tarea de clasificación se ha creado desde cero un conjunto de datos de imágenes inédito de tornillos con roscas métricas paso normal y paso fino de diámetros entre 6mm y 12mm, y rosca americana unificada paso normal y paso fino de diámetros entre 1/4' y 1/2' ambientados en situaciones cotidianas. Generalizar la propuesta desarrollada tenía un reto adicional, pues las imágenes usadas para entrenar el modelo de detección tenían fondos similares y no consideraban el conjunto de datos particular que fue creado para la tarea de clasificación.

Se ha construido un pipeline de entrenamiento para la tarea de detección y clasificación de tornillos que permite adaptar a nuevas arquitecturas y conjuntos de datos de entrenamiento. Adicionalmente, el actual modelo de detección es capaz de detectar tornillos en imágenes digitales con diferentes tipos de cabeza, rosca y largo. Esto permite en el corto plazo la creación de un conjunto de datos similar al usado para la detección pero incluyendo nuevos tipos de rosca y ambientados en escenarios cotidianos. La tarea anterior se desarrollaría mucho más rápido que hacerla de cero.

Se logró construir una interfaz amigable con el usuario, en la cual, puede subir fácilmente una foto de los tornillos de los cuales desea conocer su tamaño y tipo de rosca, teniendo en cuenta que en la imagen de salida de la interfaz identifica la parte de la imagen donde están los tornillos y su clasificación.

7.1. Trabajo Futuro

- Usar el modelo de detección para generar las coordenadas de los rectángulos que rodean los tornillos en el conjunto de imágenes creado, posteriormente agregar de manera manual las coordenadas de las imágenes que no logra identificar el modelo (menos del 10 %) y finalmente re entrenar el modelo de detección tomando como conjunto de entrenamiento las fotos usadas para la tarea de detección en el presente trabajo, junto con las imágenes creadas y sus etiquetas.

- Testear el modelo de clasificación con tornillos de longitud diferente a 25mm para identificar hasta que largo es capaz de reconocer la rosca. Una vez determinado el valor máximo de largo, se debe testear una estrategia de predicción dividiendo el rectángulo generado de diferentes maneras y generar predicciones en cada sub rectángulo, esto motivado al hecho de que al dividir un tornillo de 50mm de largo en dos cada parte

queda aproximadamente tendría un largo de 25mm y así el modelo sería capaz de detectar con alta precisión la referencia.

- Incluir imágenes de tornillos con roscas con diámetro menor a 1/4 y 6MM al conjunto creado, así se incluirían diámetros como 1/8, 5/32, 3/16 y sus respectivos en milímetros. Posteriormente, con la metodología definida en la primera propuesta etiquetar las imágenes con las nuevas referencias y re entrenar los modelos.

Referencias

[1] S. Russel and P. Norvig, «Artificial Intelligence: A Modern Approach», *Pearson Education Limited*, Third Edition, 2016.

[2] Millán Gómez, Simón (2006). Procedimientos de Mecanizado. Madrid: Editorial Paraninfo, 19/05/2022.

[3] Oberg, Erik, 1881-; McCauley, Christopher J. (2012). Machinery's handbook : a reference book for the mechanical engineer, designer, manufacturing engineer, draftsman, toolmaker, and machinist (29th ed edición). Industrial Press

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, «ImageNet Classification with Deep Convolutional Neural Networks», Comunications of the ACM, Vol 60 No 6 págs 84 - 90, 2017

[5] Dan Ciresan, Ueli Meier and Jurguen Schmidhuber, «Multi-Column Deep Neural Networks for Image Classification »

[6] A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way, https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, 19/05/2022

[7] Ross Girshick, Jeff Donahue, Trevor Darrell and Jitendra Malik , «Rich Feature hierarchies for accurate object the tension and semantic segmentation» CVPR. 2014.

[8] Albert Soto, «YOLO object detector for onboard driving images» Escola dÉnginyeria Universidad Autónoma de Barcelona.

[9] LeCun, Y., Huang, F., Bottou, L. , «Learning methods for generic object recognition with invariance to pose and lighting» In Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004, CVPR 2004. Volume 2 (2004). IEEE, II–97.

[10] Ling Guan; Yifeng He; Sun-Yuan Kung (1 March 2012). Multimedia Image and Video Processing. CRC Press. pp. 331–. ISBN 978-1-4398-3087-1.

[11] Griffin, G., Holub, A., Perona, P, «Caltech-256 object category dataset. » Technical Report 7694, California Institute of Technology, 2007.

[12] YOLO architecture, https://www.researchgate.net/figure/YOLO-architecture-YOLO-architecture-is-inspired-by-GooLeNet-model-for-image_fig2_329038564, 20/05/2022.

[13] Fei-Fei, L., Fergus, R., Perona, P., «Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories», Comput. Vision Image Understanding 106, 1 (2007), 59–70.

[14] Krizhevsky, A, «Learning multiple layers of features from tiny images». Master's thesis, Department of Computer Science, University of Toronto, 2009

[15] Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V., Rabinovich A., «Going deeper with convolutions» 2014

[16] J. Deng, W. Dong, R. Socher, L.- J. Li, Kai Li and Li Fei-Fei «ImageNet: A large-scale hierarchical image database». 2009. Conference on computer vision

[17] Hyndman, Rob J.; Koehler, Anne B. (2006). .ªnother look at measures of forecast accuracy". International Journal of Forecasting.

[18] Jaccard index, https://www.researchgate.net/publication/239604848_The_Probabilistic_Basis_of_Jacca 23/05/2022.

[19] Tom Fawcett, An introduction to ROC analysis, Institute for the Study of Learning and Expertise, (2005) https://people.inf.elte.hu/kiss/11dwhdm/roc.pdf23/05/2022.

[20] Powers David, Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation 2007, https://web.archive.org/web/20191114213255/https://www.flinders.edu.au/science_engineering/fms/S CSEM/publications/tech_repsresearch_artfcts/TRRA_2007.pdf 19/11/2022.

[21] Information Retrieval, https://en.wikipedia.org/w/index.php?title=Information_retrievaldirection=ne 19/11/2022

[22] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2009). The Elements of Statistical Learning.

[23] Cross Entropy, The Mathematics of Information Coding, Extraction and Distribution, by George Cybenko, Dianne P. O'Leary, Jorma Rissanen, 1999.

[24] Shapiro, L. G. & Stockman, G. C: Çomputer Vision", page 137, 150. Prentice Hall, 2001.

[25] Vaswani A., Shazeer N., Parmar N., Uskoreit J., Jones L., Gomez A., Lukasz K., Polosukhin I. «Attention Is All You Need » 31st Conference on Neural Information Processing Systems. 2017. Long Beach, CA, USA.

[26] Dosovitskiy A., Beyer L., Kolesnikov A., Weissenborn D., Zhai X, Unterthiner T., Dehghani M., Minderer M., Heigold G., Gelly S., Uszkoreit J., Houlsby N. « Image is Worth 16 × 16 Words: Transformers for Image Recognition at Scale» . 2021. Google Research, Brain team.

[27] He K., Zhang X., Ren S., Sun J. « Deep Residual Learning fro image Recognition » . 2015. Microsoft Research.

[28] Lui Z., Mao H., Wu C.-Y., Feichtenhofer C., Darrell T., Xie S. «A ConvNet for the 2020s ». 2022. Facebook AI Research

[29] Ulrich M., Follmann P., Neudeck J. «A comparison of shape-based matching with deep-learning-based object detection» Technisches Messen. 2019. DOI 10.1515/teme-2019-0076.

[30] Ren, Shaoqing He, Kaiming Girshick, Ross Sun, Jian. «Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks» IEEE Transactions on Pattern Analysis and Machine Intelligence. 2015. DOI 10.1109/TPAMI.2016.2577031.

[31] Detection Evaluation, https://cocodataset.org/home, 19/11/2022.

[32] Sandler, Mark and Howard, Andrew and Zhu, Menglong and Zhmoginov, Andrey and Chen, Liang-Chieh. «MobileNetV2: Inverted Residuals and Linear Bottlenecks», Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2018.

[33] Mudumbi, Terry Bian, Naizheng Zhang, Yiyi Hazoume, Florian. «An Approach Combined the Faster RCNN and Mobilenet for Logo Detection», Journal of Physics: Conference Series. 2019. 10.1088/1742-6596/1284/1/012072.

[34] Srivastava, Animesh and Dalvi, Anuj and Britto, Cyrus and Rai, Harshit and Shelke, Kavita «Explicit Content Detection using Faster R-CNN and SSD MobileNet v2.» 2020. Int. Res. J. Eng. Technol, 7, 5572–5577.

[35] Tan L, Huangfu T, Wu L, Chen W. «Comparison of YOLO v3, Faster R-CNN, and SSD for Real-Time Pill Identification» Research Square. 2021. DOI: 10.21203/rs.3.rs-668895/v1.

[36] Ahmed, Khaled R. «Smart Pothole Detection Using Deep Learning Based on Dilated Convolution» Sensors. 2021. DOI: 10.3390/s21248406.

[37] Girshick, Ross. «Fast R-CNN». Proceedings of the IEEE international conference on computer vision. 2015. p. 1440-1448.

[38] Karim, Shahid Zhang, Ye Yin, Shoulin Bibi, Irfana Brohi, Ali. «A brief review and challenges of object detection in optical remote sensing imagery. Multiagent and Grid Systems» 2020. DOI: 10.3233/MGS-200330.

[39] Short-Term Load Forecasting based on ResNet and LSTM - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/The-structure-of-ResNet-12_fig1_329954455 [accessed 23 Nov, 2022]

[40] Recuperado de https://miro.medium.com/max/778/0*LCoMNVdQjy1pzHXi

[41] Everingham, M., Eslami, S., Gool, L. V., Williams, C., Winn, J., Zisserman, A. «The pascal visual object classes challenge: A retrospective » IJCV. 2015.

[42] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. «ImageNet large scale visual recognition challenge» IJCV. 2015.

[43] Lin, T., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, L. «Microsoft COCO: Common objects in context» In ECCV. 2015.

[44] Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., Torralba, A. «Places: A 10 million image database for scene recognition» IEEE Transactions on Pattern Analysis and Machine Intelligence. 2017.

[45] Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., PontTuset, J., et al. «The open images dataset v4: Unified image classi- fication, object detection, and visual relationship detection at scale» arXiv:1811.00982. 2018.

[46] Li, Y., Xie, S., Chen, X., Dollar, P., He, K., Girshick, R. «Benchmarking detection transfer learning with vision transformers» arXiv preprint arXiv:2111.11429. 2021.

[47] Liu, L., Ouyang, W., Wang, X. et al. «Deep Learning for Generic Object Detection: A Survey» Int J Comput Vis. 2020. https://doi.org/10.1007/s11263-019-01247-4

[48] Chuanqi T., Fuchun S., Tao K.,Wenchang Z., Chao Y. Chunfang L.« A Survey on Deep Transfer Learning» The 27th International Conference on Artificial Neural Networks. 2018

[49] Ze L., Yutong L., Yue C., Han H., Yixuan W., Zheng Z., Stephen L. Baining G.«Swin Transformer: Hierarchical Vision Transformer using Shifted Windows» 2021

[50]