



COMILLAS

UNIVERSIDAD PONTIFICIA

ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS
INDUSTRIALES

TRABAJO FIN DE GRADO

DISEÑO Y CARACTERIZACIÓN DE UN
GENERADOR SINTÉTICO DE PLANOS PARA EL
ENTRENAMIENTO DE UN MODELO DE DEEP
LEARNING

Autor: Galo Iglesias Aramburu

Directores: Álvaro Jesús López López y José Portela González

Madrid

Julio de 2024

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título
Diseño y caracterización de un generador sintético de planos para el entrenamiento de un
modelo de Deep Learning

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2023/24 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.



Fdo.: Galo Iglesias Aramburu

Fecha: 10/ 07/ 2024

Autorizada la entrega del proyecto

DIRECTORES DEL PROYECTO

**Álvaro Jesús
López López**

Firmado digitalmente por
Álvaro Jesús López López
Fecha: 2024.07.11 13:43:30
+02'00'

Fdo.: Álvaro Jesús López López

Fecha: 10/ 07/ 2024

**José
Portela
González**

Firmado digitalmente
por José Portela
González
Fecha: 2024.07.12
17:59:57 +02'00'

Fdo.: José Portela González

Fecha: 10/ 07/ 2024



COMILLAS
UNIVERSIDAD PONTIFICIA

ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS
INDUSTRIALES

TRABAJO FIN DE GRADO

DISEÑO Y CARACTERIZACIÓN DE UN
GENERADOR SINTÉTICO DE PLANOS PARA EL
ENTRENAMIENTO DE UN MODELO DE DEEP
LEARNING

Autor: Galo Iglesias Aramburu

Directores: Álvaro Jesús López López y José Portela González

Madrid

Julio de 2024

Agradecimientos

En primer lugar, me gustaría dar las gracias a mis tutores del TFG, Álvaro López y José Portela. Llevar a cabo este proyecto ha sido un reto y una oportunidad única para seguir aprendiendo de la ruta que está siguiendo el mundo de la ingeniería en el tema de la Inteligencia Artificial. Siempre estaré enormemente agradecido por su ayuda, por su constante motivación, aun cuando aparecían problemas que no parecían tener salida, por esas reuniones intempestivas a horas inusuales a causa del cambio horario. Este TFG me ha permitido aprender de dos grandes figuras, en lo profesional, en lo académico y, ante todo, en lo personal. Siempre serán dos personas a las que tomar como referencia, y que me han ayudado a crecer en muchos más aspectos de los que abarca este proyecto.

En segundo lugar, debo dar las gracias a la Cátedra de Industria Conectada. Desde que entré en el verano de 2do de carrera, la Cátedra, junto al IIT, me han proporcionado un ambiente de experimentación y aprendizaje continuo, rodeado de gente a la que admiro y respeto. Los viernes de Cervezas Conectadas tendrán siempre un hueco reservado en mi memoria.

Como no podía ser de otra forma, siempre estaré enormemente agradecido a mis padres, Berta y Enrique, y a mis hermanas, Marta e Irene. Son los que han tenido que aguantar los días (y noches) de dejar el ordenador trabajando en la cocina, con los ventiladores a pleno rendimiento. Siempre han sido un apoyo incondicional en todo lo que hago, y gracias a ellos, estoy donde estoy, teniendo que aguantar mis malos días, y mis frustraciones constantes.

Otro grupo de personas que me ha ayudado enormemente durante la realización de este proyecto es el de “Míchigan ICAI”. Durante el intercambio, he tenido el placer de hacer un grupo increíble de amigos que han sido un apoyo constante durante todo el año y que ha hecho que la experiencia en la Universidad de Michigan tenga un espacio reservado en mi corazón.

Por último, y no por ello, menos importantes, a mi grupo de amigos del colegio Virgen de Mirasierra, que siempre me ha ayudado en los momentos más difíciles y que, a pesar de la distancia y de nuestras rutinas, siempre nos tenemos los unos a los otros. Por muchas cosas que nos pasen, siempre tendremos las mil anécdotas e historias que hemos vivido (y las que nos quedan) y la misma conexión que cuando estudiábamos el bachillerato.

A todos los que han contribuido a la consecución de este TFG y a los que he tenido cerca a lo largo de toda mi experiencia universitaria, solo puedo decir:

HA SIDO UNA SUERTE Y AUTÉNTICO PLACER HABER TENIDO LA OPORTUNIDAD DE CONOCEROS: GRACIAS DE CORAZÓN

DISEÑO Y CARACTERIZACIÓN DE UN GENERADOR DE PLANOS SINTÉTICOS PARA EL ENTRENAMIENTO DE UN MODELO DE DEEP LEARNING

Autor: Iglesias Aramburu, Galo.

Directores: López López, Álvaro Jesús / Portela González, José.

Entidad Colaboradora: ICAI – Universidad Pontificia Comillas.

RESUMEN DEL PROYECTO

En los últimos años, el campo de la Inteligencia Artificial ha experimentado un crecimiento exponencial, siendo el siguiente escalón en la Industria 4.0 mediante su implementación en el mundo profesional. Este proyecto se centra en el diseño y caracterización de un generador de planos industriales sintéticos, destinado al entrenamiento de un modelo de *Deep Learning*, empleando la arquitectura YOLO, integrada en la red neuronal DarkNet. Se busca solucionar el problema de la escasez de planos de producción industrial reales para entrenar un modelo de *Machine Learning* mediante la generación de datos artificiales que repliquen de manera realista la variabilidad existente en los ejemplos que se encuentran en la vida real. Se propone mejorar el rendimiento y eficiencia de esta tecnología, con el fin de proporcionar una posible implementación de la IA en el sector de la manufactura. Los resultados recogidos demuestran que los datos sintéticos obtenidos a partir del generador diseñado mejoran la detección de tolerancias realizada por el modelo, lo que allana el camino para una potencial aplicación de la Inteligencia Artificial en el sector industrial, pese al obstáculo de no poder disponer de suficientes ejemplos reales para desarrollarla.

Palabras clave: Deep Learning, Datos sintéticos, YOLO, Planos industriales, Tolerancias, Industria 4.0, Computer Vision, Redes neuronales

1. Introducción

En el sector de la manufactura y la producción industrial, los planos que detallan las dimensiones y tolerancias admitidas en un producto actúan a modo de “manual de instrucciones”. Por ende, su correcta y precisa interpretación es esencial e imprescindible para el buen devenir de un proceso de fabricación.

Un análisis impreciso o incompleto de un plano y de las tolerancias que definen el rango admisible de las imperfecciones que puede presentar un producto puede desembocar en problemas, tanto a nivel industrial (traduciéndose en un decremento de la calidad del

producto, en una mayor tasa de defectos, en un mayor número de unidades desechadas, etc) como económico (un decremento de la calidad de producción puede derivar en pérdidas monetarias, en un deterioro de la imagen del fabricante, etc).

Actualmente, los planos son usualmente analizados por operarios, los cuales son encargados de comunicar a las etapas posteriores de la cadena de producción aquellos aspectos que deben ser considerados a la hora de producir el elemento que en los planos se describe.

Ante la importancia que esta etapa de análisis previo representa en el buen desarrollo del proceso de manufactura, la implementación de una tecnología basada en la Inteligencia Artificial puede ser especialmente beneficiosa. Automatizar esta etapa mediante la inclusión de una herramienta de IA lo suficientemente precisa puede ayudar a reducir el error en la detección de las tolerancias presentes, y asignar al personal inicialmente destinado a esta tarea mecánica y repetitiva a labores más productivas.

2. Definición del proyecto

Este trabajo propone el desarrollo de una red neuronal convolucional para automatizar el análisis de planos en entornos industriales. Esta tecnología queda enmarcada en el ámbito del *Deep Learning*, siendo empleada en este caso en una labor de *Computer Vision*, ciencia dedicada a desarrollar métodos de detección de objetos en imágenes.

Dado que una imagen es en realidad una matriz de píxeles, las redes neuronales convolucionales, las cuales aplican sucesivamente la convolución (una operación matricial) a través de sus diferentes capas a la matriz que se le introduce, pueden emplearse en el proceso de detección de objetos en imágenes [1].

El modelo que se busca desarrollar a lo largo de este proyecto sería, por tanto, una red neuronal convolucional, basada en la arquitectura YOLO (You Only Look Once). La arquitectura YOLO es de las arquitecturas más utilizadas en los procesos de detección de objetos. Se basa en la división de una imagen según una cuadrícula; en cada una de las regiones de dicha cuadrícula, la red determina si existe o no un objeto de interés. A partir del análisis individual de cada región, se realiza el proceso de detección, recuadrando aquellos objetos que se identifiquen [2].

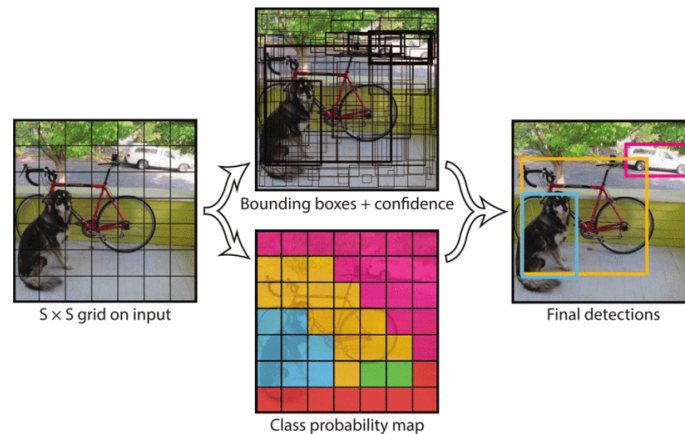


Ilustración 1: Funcionamiento de la arquitectura YOLO [2]

Si bien existen muchas otras arquitecturas y métodos para detectar objetos (como la tecnología Faster R-CNN, el método de la ventana deslizante, etc), la arquitectura YOLO es la que mejores resultados presenta a nivel de velocidad, precisión y generalización [3].

No obstante, el entrenamiento de un modelo como este en el contexto descrito presenta una gran limitación: la escasez de planos que pueden ser empleados para entrenar la red.

Con todo esto, este proyecto no solo busca entrenar un modelo de *Deep Learning* sino que trata de buscar soluciones ante la falta de ejemplos suficientes para entrenarlo. Pocas empresas están dispuestas a ceder sus planos de producción para entrenar un modelo como este; a esto se une el hecho de que muchos planos no cuentan con ejemplos de las tolerancias de interés, dificultando aún más el proceso de reunir un set de datos lo suficientemente completo para lograr obtener un modelo lo suficientemente preciso y eficiente.

Así, el objetivo fundamental del proyecto es **mejorar el rendimiento de un sistema de detección de tolerancias en planos de ingeniería basado en la tecnología YOLO.**

Para la consecución de este objetivo y con la intención de solucionar el problema que representaba la escasez de planos reales, se optó por diseñar y caracterizar una herramienta que generara datos artificiales que pudieran ser empleados en el entrenamiento de una CNN. Un generador de planos sintéticos lo suficientemente completo como para captar una parte significativa de la variabilidad que se puede encontrar en planos industriales reales puede contribuir a superar el obstáculo planteado

y lograr el objetivo propuesto, buscando desarrollar una tecnología lo suficientemente buena como para poder ser implementada en un entorno industrial real.

Este TFG busca alinearse con el estado del arte actual. El uso de datos sintéticos ya ha sido tratado en múltiples artículos. Su uso en el entrenamiento de modelos de *Computer Vision* destinados a reconocer símbolos en diagramas de tuberías, elementos en planos arquitectónicos [4], múltiples componentes de planos ingenieriles [5] o dibujos hechos a mano alzada [6], ha arrojado resultados positivos, convirtiéndose en una potencial solución ante la falta de datos.

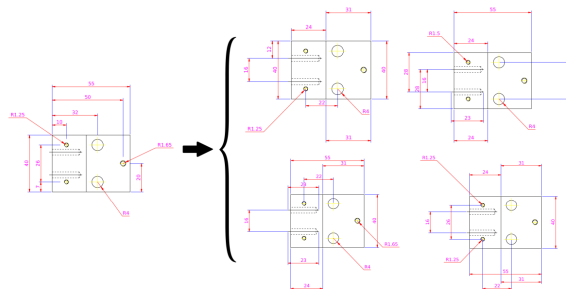


Ilustración 2: Datos sintéticos generados a partir de una figura real, modificando el dimensionado [5]

3. Descripción del modelo/sistema/herramienta

En este proyecto, se usarían/desarrollarían tres herramientas principales:

1. **Generador de planos sintéticos:** para este TFG, se partió de una versión prematura de un generador de planos sintéticos, desarrollada en un proyecto anterior. Esta herramienta fue programada en Python. Su funcionamiento es básico: a partir de planos fuente vacíos (es decir, sin ninguna tolerancia en ellos), se generan recortes de manera aleatoria; a estos recortes se les introduce tolerancias aleatoriamente, con diferentes tamaños, rotándolas, añadiendo letras y números que las acompañan, etc. En este trabajo, se modificaría el generador para mejorar el realismo de los datos generados de forma artificial. Algunas de estas mejoras incluyeron la adición de más ejemplos tanto de tolerancias como de planos fuente usados en la generación, la opción de incluir agrupaciones de dos o tres tolerancias, las cuales aparecen juntas en el plano, o la mejora de la rotación de las tolerancias insertadas.

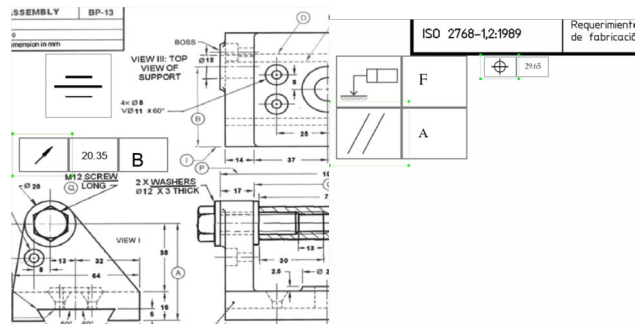


Ilustración 3: Ejemplos de imágenes sintéticas creadas por el generador desarrollado, introduciendo ejemplos de tolerancias de interés en recortes vacíos.

2. Modelo de *Deep Learning*/Red neuronal convolucional: la red neuronal que se entrena en este proyecto es DarkNet [7], una red neuronal *Open Source*, que emplea YOLOv4, ofreciendo tanto un proceso de entrenamiento sencillo de realizar como múltiples métricas de análisis, tales como la *Precision*, el *Recall*, o el *mean Average Precision* (mAP); este último es comúnmente utilizado para comparar diferentes modelos destinados a la detección de objetos, y es la métrica que permitiría determinar qué modelo actuaba mejor. La red debería reconocer seis tolerancias de interés, las cuales se aprecian en la Ilustración 4, algo que también se tendría en cuenta a la hora de generar el set de datos sintéticos. Se entrenarían tres modelos diferentes:

- i) Modelo entrenado con datos reales
- ii) Modelo entrenado con datos sintéticos
- iii) Modelo entrenado con una combinación de datos reales y sintéticos

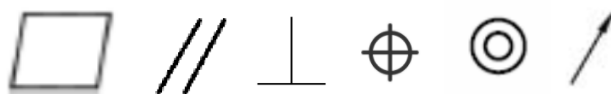


Ilustración 4: Tolerancias a reconocer por la red. De derecha a izquierda: tolerancia de planicidad, paralelismo, perpendicularidad, posición, concetricidad y excentricidad circular

Los datos reales se conseguirían a partir de un set de planos reales (con escasos ejemplos de las tolerancias de interés) y los datos sintéticos se obtendrían a partir del generador de datos sintéticos.

Una vez entrenados los modelos, se analizarían los resultados obtenidos con cada uno de ellos empleando un set de test (también de planos reales). A partir de este análisis, se observaría el efecto que el uso de datos sintéticos tiene en el

entrenamiento del modelo; además, se realizaría un estudio cualitativo para entender las deficiencias que los datos sintéticos mostraban en el proceso de detección para realimentar al generador, introduciendo las mejoras pertinentes.

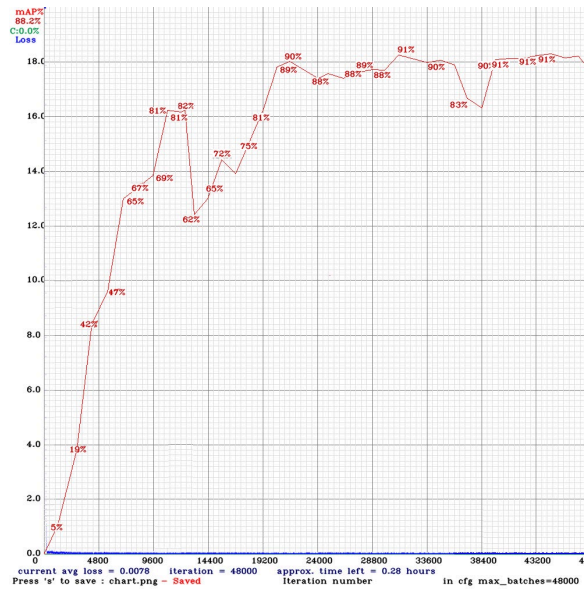


Ilustración 5: Ejemplo de una gráfica de entrenamiento de uno de los modelos

3. GUI: para mostrar cómo se podría implementar esta tecnología en el sector de la manufactura, se desarrollaría una GUI en Python que permitiera al usuario seleccionar uno o varios planos, sobre los cuales se realizaría el proceso de detección de tolerancias. Con esta GUI se usaría aquel modelo que, de los obtenidos, arrojará los mejores resultados en la sección de análisis.



Ilustración 6: GUI diseñada

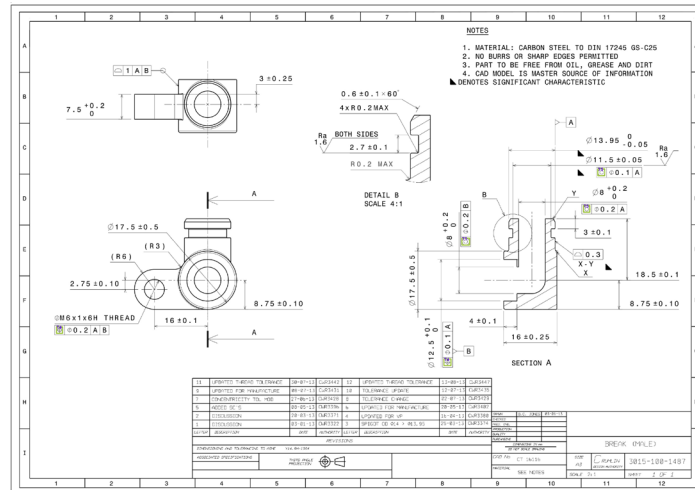


Ilustración 7: Ejemplo de un plano sobre el que se ha hecho la detección de tolerancias por medio de la GUI

4. Métodos de análisis de los modelos

Para analizar y comparar los modelos descritos, se recurrió a tres métricas fundamentales:

1. *Precision*: se trata de una métrica que mide qué fracción de las detecciones positivas representan realmente verdaderos positivos. Para comparar entre modelos, se obtuvo para un valor umbral de confianza de 0.25.

$$Precision = \frac{TP}{TP + FP}$$

2. *Recall*: esta métrica mide qué porcentaje de verdaderos positivos han sido detectados de manera correcta. Al igual que la *Precision*, se halló para un valor umbral de confianza de 0.25.

$$Recall = \frac{TP}{TP + FN}$$

3. *Average Precision (AP)* y *mean Average Precision (mAP)*: ambas son métricas que permiten comparar el rendimiento de diferentes modelos. Mientras que el AP se calcula por cada tolerancia (y, por tanto, permite entender qué modelo detecta mejor cada tolerancia), el mAP da una idea general de cómo rinde el modelo en conjunto considerando todas las clases. Estas métricas se obtienen siguiendo una serie de pasos:

- i) Obtención de la curva *Precision-Recall* para diferentes valores umbrales de confianza (los cuales determinan el valor mínimo de confianza, entre 0 y 1, que la red debe otorgar a una detección para considerarla válida) para cada una de las clases.

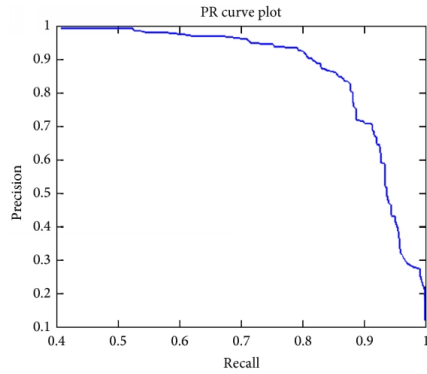


Ilustración 8: Ejemplo de una curva Precision-Recall [8]

- ii) Cálculo del AP de cada clase. El AP se define como el área debajo de la curva *Precision-Recall*.
- iii) Cálculo del mAP. Se obtiene haciendo la media de los APs de las diferentes clases.

5. Resultados

Después de introducir múltiples mejoras en el generador de planos sintéticos, se obtuvieron los resultados que figuran en la Tabla 1.

Como se aprecia, tanto el modelo entrenado con datos sintéticos (SYNTH) como el entrenado con datos híbridos (HYBRID) presentan un mejor mAP en comparación con el modelo entrenado con datos reales, el cual fue entrenado primero con 12000 iteraciones (RAW_Cropped12000), al igual que los dos mencionados anteriormente, y más tarde con 48000 iteraciones (RAW_Cropped48000).

Clase	Planicidad			Perpendicularidad			Paralelismo			Concentricidad			Posición			Circular			mAP
	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	
RAW_Cropped12000	0.43	0.57	43.03%	0.31	0.53	43.60%	0.37	0.28	28.33%	0.50	0.79	67.24%	0.54	0.75	60.35%	0.42	0.49	37.83%	46.73%
RAW_Cropped48000	0.76	0.85	81.40%	0.44	0.72	68.97%	0.58	0.72	72.30%	0.75	0.91	89.76%	0.73	0.88	82.45%	0.58	0.59	48.22%	73.85%
SYNTH	0.28	0.98	92.70%	0.36	0.75	68.83%	0.68	0.84	77.26%	0.84	0.91	91.31%	0.74	0.74	75.80%	0.61	0.95	84.95%	81.81%
HYBRID	0.74	0.93	91.28%	0.59	0.84	79.88%	0.81	0.88	84.59%	0.89	0.94	96.30%	0.77	0.82	81.98%	0.76	0.92	88.06%	87.02%

Tabla 1: Resultados obtenidos con los modelos estudiados

6. Conclusiones

Los resultados obtenidos permiten concluir que los datos creados de manera sintética por medio del generador mejoran el rendimiento de la red neuronal. Así, si los datos sintéticos son capaces de reflejar la variabilidad que existe en la realidad, se pueden obtener mejores resultados con un set de datos sintético, con mayor cantidad de ejemplos, que con un set de datos reales que presente pocos ejemplos de las clases de interés, incluso con un menor tiempo de entrenamiento. También se observó que el modelo entrenado con la combinación de datos reales y sintéticos (modelo HYBRID) es el que mejores resultados presenta; si bien los datos sintéticos funcionan bien por su cuenta, su combinación con los datos reales mejora la generalización del modelo.

7. Referencias

- [1] K. O'Shea y R. Nash, «An Introduction to Convolutional Neural Networks,» *ArXiv (Cornell University)*, 2015, doi: <https://doi.org/10.48550/arXiv.1511.08458>.
- [2] J. Redmon, S. Divvala, R. Girshick y A. Farhadi, «You Only Look Once: Unified, Real-Time Object Detection,» 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA (June 27-30, 2016), pp. 779-788, 2016, doi: <https://doi.org/10.1109/cvpr.2016.91>.
- [3] J. Kim, J. Y. Sung y S. Park, «Comparison of Faster-RCNN, YOLO, and SSD for Real-Time Vehicle Type Recognition,» *2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia)*, Seoul, South Korea (November 1-3, 2020), pp. 1-4, 2020, doi: <https://doi.org/10.1109/icce-asia49877.2020.9277040>.
- [4] M. Delalandre, E. Valveny, T. Pridmore y D. Karatzas, «Generation of synthetic documents for performance evaluation of symbol recognition & spotting systems,» *IJDAR* 13(3), pp. 187-207, 2010, doi: <https://doi.org/10.1007/s10032-010-0120-x>.
- [5] W. Zhang, Q. Chen, C. Koz, L. Xie, A. Regmi, S. Yamakawa, T. Furuhashi, K. Shimada y L. B. Kara, «Data Augmentation of Engineering Drawings for Data-Driven Component Segmentation,» *Proceedings of the ASME 2022 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Volume 3A: 48th Design Automation Conference (DAC)*, St. Louis, Missouri, USA (August 14-17, 2022), 2022, doi: <https://doi.org/10.1115/detc2022-91043>.
- [6] L. Fu y L. B. Kara, «From engineering diagrams to engineering models: Visual recognition and applications,» *Computer-Aided Design*, 43(3), pp. 278-292, 2011, doi: <https://doi.org/10.1016/j.cad.2010.12.011>.

- [7] A. Bochkovski, C. Y. Wang y H. Y. Liao, «YOLOv4: Optimal Speed and Accuracy of Object Detection,» *ArXiv (Cornell University)*, 2020, doi: <https://doi.org/10.48550/arXiv.2004.10934>.
- [8] Y. Liu, «The Confusing Metrics of AP and mAP for Object Detection,» Medium, 2018. [En línea]. Available: <https://yanfengliux.medium.com/the-confusing-metrics-of-ap-and-map-for-object-detection-3113ba0386ef>. [Último acceso: 8 de febrero de 2024].

DESIGN AND CHARACTERIZATION OF A SYNTHETIC PLANE GENERATOR FOR THE TRAINING OF A DEEP LEARNING MODEL

Author: Iglesias Aramburu, Galo.

Supervisors: López López, Álvaro Jesús / Portela González, José.

Collaborating Entity: ICAI – Universidad Pontificia Comillas.

ABSTRACT

In recent years, the field of Artificial Intelligence has experienced exponential growth, becoming the next step in Industry 4.0 through its implementation in the professional world. This project focuses on the design and characterization of a synthetic industrial drawing generator, intended for training a Deep Learning model using the YOLO architecture, integrated into the DarkNet neural network. The aim is to address the problem of the scarcity of real industrial production drawings for training a Machine Learning model by generating artificial data that realistically replicates the variability found in real-life examples. The goal is to improve the performance and efficiency of this technology to provide a potential implementation of AI in the manufacturing sector. The collected results demonstrate that the synthetic data obtained from the designed generator enhances the tolerance detection performed by the model when compared to the use of real data, paving the way for a potential application of Artificial Intelligence in the industrial sector, despite the obstacle of not having enough real examples to develop it.

Keywords: Deep Learning, Synthetic data, YOLO, Industrial drawings, Tolerances, Industry 4.0, Computer Vision, Neural Networks

1. Introduction

In the manufacturing and industrial production sector, blueprints that detail a product's dimensions and allowable tolerances act as "instruction manuals." Therefore, their correct and precise interpretation is essential and indispensable for the smooth running of the manufacturing process.

An inaccurate or incomplete analysis of an industrial drawing and the tolerances that define the acceptable range of imperfections a product may have can lead to problems at both the industrial level (resulting in a decrease in product quality, a higher defect rate, more discarded units, etc.) and the economic level (a reduction in production quality can lead to financial losses, deterioration of the manufacturer's image, etc.).

Currently, blueprints are usually analyzed by operators, responsible for communicating to the subsequent stages of the production chain the aspects that need to be considered when producing the element described in the blueprints.

Given the importance of this preliminary analysis stage in the successful development of the manufacturing process, implementing a technology based on Artificial Intelligence can be especially beneficial. Automating this stage by incorporating a sufficiently precise AI tool can help reduce errors in the detection of present tolerances and reassign the personnel initially dedicated to this mechanical and repetitive task to more productive activities.

2. Project Definition

This work proposes the development of a convolutional neural network to automate the analysis of blueprints in industrial environments. This technology is framed within the field of Deep Learning, being used in this case for a Computer Vision task, a science dedicated to developing methods for object detection in images.

Since an image is essentially a matrix of pixels, convolutional neural networks, which successively apply convolution through their different layers to the input matrix, can be employed in the process of object detection in images [1]. The model aimed to be developed throughout this project would, therefore, be a convolutional neural network based on the YOLO (You Only Look Once) architecture. The YOLO architecture is one of the most commonly used architectures in object detection processes. It is based on dividing an image into a grid; in each of the regions of this grid, the network determines whether or not there is an object of interest. From the individual analysis of each region, the detection process is carried out, bounding the identified objects [2].

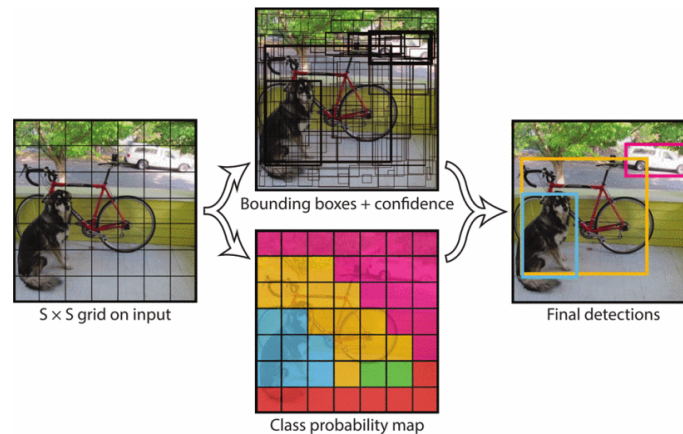


Figure 1: How the YOLO architecture works [2]

Although there are many other architectures and methods for object detection (such as Faster R-CNN technology, the sliding window method, etc.), the YOLO architecture presents the best results in terms of speed, precision, and generalization [3]. However, training a model like this in the described context presents a significant limitation: the scarcity of blueprints that can be used to train the network.

Therefore, this project not only aims to train a Deep Learning model but also seeks solutions to the lack of sufficient examples to train it. Few companies are willing to share their production blueprints to train a model like this; additionally, many blueprints do not include examples of the tolerances of interest, further complicating the process of gathering a sufficiently complete dataset to achieve a sufficiently accurate and efficient model.

Thus, the fundamental objective of the project is **to improve the performance of a tolerance detection system in engineering blueprints based on YOLO technology**. To achieve this objective and to address the problem posed by the scarcity of real blueprints, it was decided to design and characterize a tool that generates artificial data that could be used in the training of a CNN. A synthetic blueprint generator sufficiently complete to capture a significant part of the variability that can be found in real industrial blueprints can help overcome the posed obstacle and achieve the proposed objective, aiming to develop technology good enough to be implemented in a real industrial environment.

This final degree project seeks to align with the current state of the art. The use of synthetic data has already been addressed in multiple articles. Its use in training

Computer Vision models aimed at recognizing symbols in piping diagrams, elements in architectural plans [4], multiple components of engineering blueprints [5], or hand-drawn sketches [6], has yielded positive results, becoming a potential solution to the lack of data.

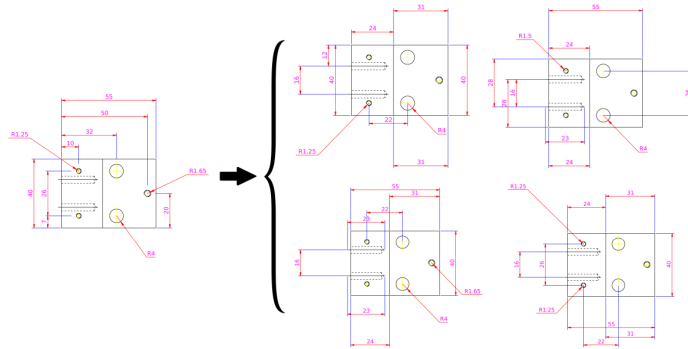


Figure 2: Synthetic data generated from a real figure, modifying the dimensions

3. Model/System/Tool Description

In this project, three main tools would be used/developed:

1. Synthetic Drawing Generator: For this final degree project, an early version of a synthetic blueprint generator, developed in a previous project, was used as a starting point. This tool was programmed in Python. Its functionality is basic: starting from empty source blueprints (i.e., without any tolerances), it randomly generates cutouts; tolerances are randomly introduced into these cutouts, with different sizes, rotations, adding accompanying letters and numbers, etc. In this work, the generator would be modified to improve the realism of the artificially generated data. Some of these improvements included adding more examples of both tolerances and source blueprints used in the generation, the option to include groups of two or three tolerances that appear together in the blueprint, and improving the rotation of the inserted tolerances.

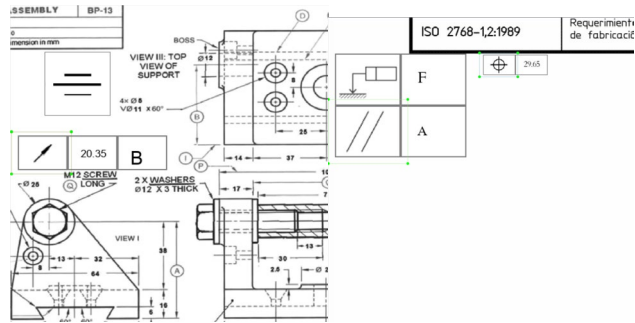


Figure 3: Examples of synthetic images created by the developed generator, introducing examples of tolerances of interest in empty cutouts.

2. Deep Learning Model/Convolutional Neural Network: The neural network trained in this project is DarkNet [7], an Open Source neural network that uses YOLOv4, offering both an easy-to-perform training process and multiple analysis metrics, such as Precision, Recall, or mean Average Precision (mAP); the latter is commonly used to compare different object detection models and is the metric that would allow determining which model performs better. The network should recognize six tolerances of interest, which are shown in Figure 4, something that would also be taken into account when generating the synthetic dataset.

Three different models would be trained:

- i) Model trained with real data
- ii) Model trained with synthetic data
- iii) Model trained with a combination of real and synthetic data

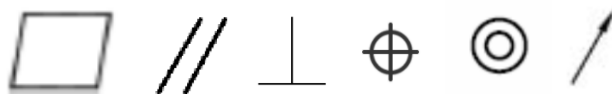


Figure 4: Tolerances to be recognized by the network. From right to left: flatness, parallelism, perpendicularity, position, concentricity, and circular runout.

The real data would be obtained from a set of real blueprints (with few examples of the tolerances of interest), and the synthetic data would be obtained from the synthetic data generator. Once the models are trained, the results obtained with each of them would be analyzed using a test set (also of real blueprints). From this analysis, the effect of using synthetic data in training the model would be observed; additionally, a qualitative study would be conducted to understand the deficiencies

that synthetic data showed in the detection process to feed back into the generator, introducing the relevant improvements.

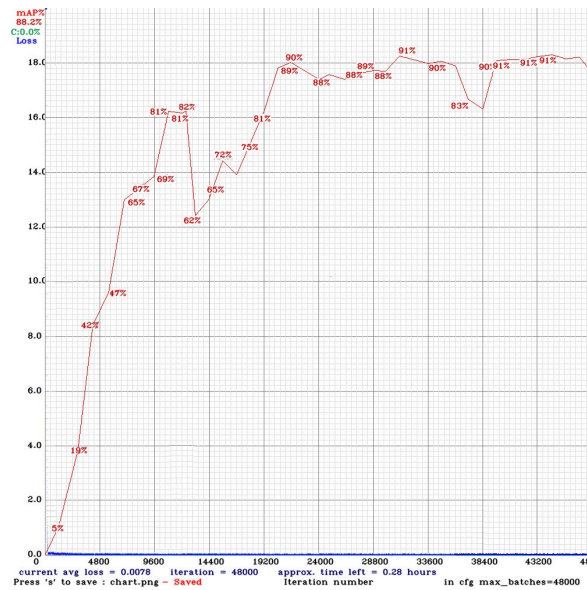


Figure 5: Example of a training graph of one of the models.

3. GUI: To demonstrate how this technology could be implemented in the manufacturing sector, a GUI would be developed in Python, allowing the user to select one or more blueprints on which the tolerance detection process would be performed. This GUI would use the model that, of the obtained ones, produced the best results in the analysis section.



Figure 6: Designed GUI.

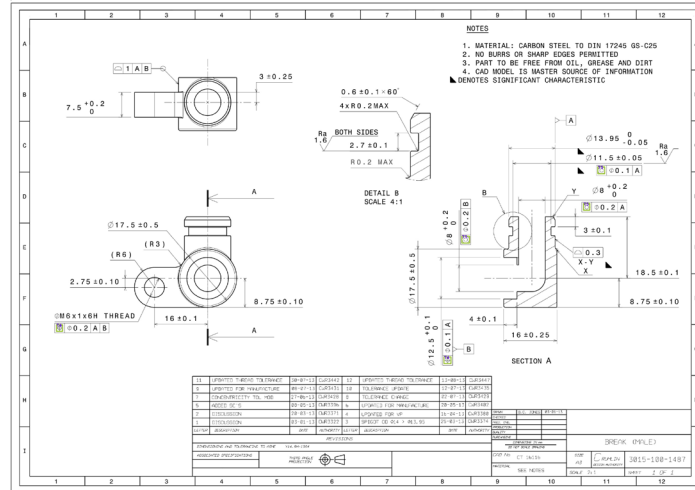


Figure 7: Example of a blueprint on which tolerances have been detected using the GUI.

4. Methods for analyzing models

To analyze and compare the described models, three fundamental metrics were used:

1. Precision: This metric measures the fraction of positive detections that are actually true positives. To compare between models, it was obtained for a confidence threshold value of 0.25.

$$Precision = \frac{TP}{TP + FP}$$

2. Recall: This metric measures the percentage of true positives that have been correctly detected. Like Precision, it was found for a confidence threshold of 0.25.

$$Recall = \frac{TP}{TP + FN}$$

3. Average Precision (AP) and Mean Average Precision (mAP): Both metrics allow for the comparison of different models' performance. While AP is calculated for each class (thus allowing to understand which model better detects each tolerance), mAP provides a general idea of how the model performs overall considering all classes. These metrics are obtained through the following steps:

- i) Obtaining the Precision-Recall curve for different confidence threshold values (which determine the minimum confidence level, between 0 and 1, that the network must assign to a detection to consider it valid) for each class.

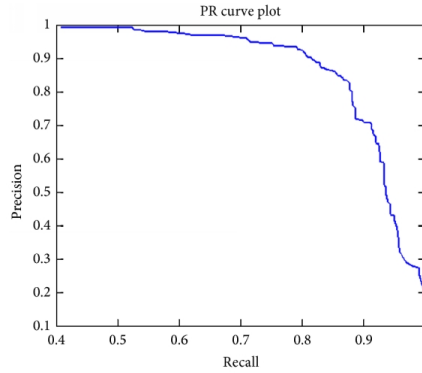


Figure 8: Example of a Precision-Recall Curve [8]

- ii) Calculation of the AP for each class. AP is defined as the area under the Precision-Recall curve.
- iii) Calculation of mAP. It is obtained by averaging the APs of the different classes.

5. Results

After introducing multiple improvements to the synthetic generator, the results shown in Table 1 were obtained. As can be seen, both the model trained with synthetic data (SYNTH) and the model trained with hybrid data (HYBRID) show better mAP compared to the model trained with real data, which was first trained with 12,000 iterations (RAW_Cropped12000), just like the two previously mentioned, and later with 48,000 iterations (RAW_Cropped48000).

Class	Flatness			Perpendicularity			Parallelism			Concentricity			Position			Circular runout			mAP
Model	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	
RAW_Cropped12000	0.43	0.57	43.03%	0.31	0.53	43.60%	0.37	0.28	28.33%	0.50	0.79	67.24%	0.54	0.75	60.35%	0.42	0.49	37.83%	46.73%
RAW_Cropped48000	0.76	0.85	81.40%	0.44	0.72	68.97%	0.58	0.72	72.30%	0.75	0.91	89.76%	0.73	0.88	82.45%	0.58	0.59	48.22%	73.85%
SYNTH	0.28	0.98	92.70%	0.36	0.75	68.83%	0.68	0.84	77.26%	0.84	0.91	91.31%	0.74	0.74	75.80%	0.61	0.95	84.95%	81.81%
HYBRID	0.74	0.93	91.28%	0.59	0.84	79.88%	0.81	0.88	84.59%	0.89	0.94	96.30%	0.77	0.82	81.98%	0.76	0.92	88.06%	87.02%

Table 1: Results obtained with the studied models.

6. Conclusions

The obtained results allow us to conclude that the data created synthetically through the generator improve the performance of the neural network. Thus, if synthetic data can reflect the variability that exists in reality, better results can be achieved with a synthetic dataset, with a greater number of examples, than with a real dataset that has few examples of the classes of interest, even with less training time. It was also observed that the model trained with a combination of real and synthetic data (HYBRID model) shows the best results; although synthetic data perform well on their own, their combination with real data improves the model's generalization.

7. References

- [1] K. O'Shea y R. Nash, «An Introduction to Convolutional Neural Networks,» *ArXiv (Cornell University)*, 2015, doi: <https://doi.org/10.48550/arXiv.1511.08458>.
- [2] J. Redmon, S. Divvala, R. Girshick y A. Farhadi, «You Only Look Once: Unified, Real-Time Object Detection,» 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA (June 27-30, 2016), pp. 779-788, 2016, doi: <https://doi.org/10.1109/cvpr.2016.9>.
- [3] J. Kim, J. Y. Sung y S. Park, «Comparison of Faster-RCNN, YOLO, and SSD for Real-Time Vehicle Type Recognition,» 2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia), Seoul, South Korea (November 1-3, 2020), pp. 1-4, 2020, doi: <https://doi.org/10.1109/icce-asia49877.2020.9277040>.
- [4] M. Delalandre, E. Valveny, T. Pridmore y D. Karatzas, «Generation of synthetic documents for performance evaluation of symbol recognition & spotting systems,» *IJDAR* 13(3), pp. 187-207, 2010, doi: <https://doi.org/10.1007/s10032-010-0120-x>.
- [5] W. Zhang, Q. Chen, C. Koz, L. Xie, A. Regmi, S. Yamakawa, T. Furuhashi, K. Shimada y L. B. Kara, «Data Augmentation of Engineering Drawings for Data-Driven Component Segmentation,» *Proceedings of the ASME 2022 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Volume 3A: 48th Design Automation Conference (DAC)*, St. Louis, Missouri, USA (August 14-17, 2022), 2022, doi: <https://doi.org/10.1115/detc2022-91043>.
- [6] L. Fu y L. B. Kara, «From engineering diagrams to engineering models: Visual recognition and applications,» *Computer-Aided Design*, 43(3), pp. 278-292, 2011, doi: <https://doi.org/10.1016/j.cad.2010.12.011>.
- [7] A. Bochkovskiy, C. Y. Wang y H. Y. Liao, «YOLOv4: Optimal Speed and Accuracy of Object Detection,» *ArXiv (Cornell University)*, 2020, doi: <https://doi.org/10.48550/arXiv.2004.10934>.

[8] Y. Liu, «The Confusing Metrics of AP and mAP for Object Detection,» Medium, 2018. [Online]. Available: <https://yanfengliux.medium.com/the-confusing-metrics-of-ap-and-map-for-object-detection-3113ba0386ef>. [Last access: 8 February 2024].

Índice de la memoria

Capítulo 1. Introducción	7
1.1 Motivación del proyecto.....	7
1.1.1 Ventajas en el ámbito industrial.....	8
1.1.2 Ventajas en el ámbito económico.....	9
1.2 Justificación del proyecto.....	9
1.2.1 Solución ante la falta de ejemplos.....	9
1.3 Objetivos del proyecto.....	10
1.4 Objetivos de desarrollo sostenible.....	12
1.4.1 Objetivo 9. Industria, innovación e infraestructura.....	12
1.4.2 Objetivo 8. Trabajo decente y crecimiento económico.....	12
1.5 Metodología.....	13
1.5.1 Definición del Proyecto.....	14
1.6 Planificación y estimación económica.....	15
Capítulo 2. Estado de la cuestión	16
2.1 Introducción.....	16
2.2 Deep Learning en la industria 4.0.....	16
2.3 Evolución del Deep Learning en el reconocimiento de símbolos.....	17
2.3.1 Redes neuronales convolucionales (O Convolutional Neural Networks, CNNs).....	17
2.3.2 Deep Learning para el reconocimiento de patrones complejos.....	24
2.3.3 Datos sintéticos en el entrenamiento de redes de Deep Learning.....	26
2.4 Futuras direcciones en este ámbito.....	30
Capítulo 3. Desarrollo del detector de tolerancias en planos ingenieriles	31
3.1 Análisis y estudio del problema.....	31
3.2 Selección de las clases.....	32
3.3 Implementación del detector de tolerancias en planos de ingeniería.....	35
3.3.1 Selección de la red a emplear: DarkNet.....	36
3.4 Desarrollo del generador sintético.....	41
3.4.1 Funcionamiento.....	42
3.4.2 Mejoras.....	46
3.4.3 Generación de imágenes.....	48

3.5	Obtención de los datos a emplear.....	48
3.5.1	<i>Sets de datos de entrenamiento</i>	49
3.6	Entrenamiento de los modelos	54
3.6.1	<i>Modelo I: Datos reales (M01_RAW_Cropped)</i>	56
3.6.2	<i>Modelo II: Datos sintéticos (M02_SYNTH)</i>	59
3.6.3	<i>Modelo III: Datos híbridos (M03_HYBRID)</i>	60
3.7	Análisis de resultados.....	61
3.7.1	<i>Resultados por versión del generador</i>	61
3.7.2	<i>Resultados por tipo de modelo</i>	66
3.7.3	<i>Elección del modelo</i>	67
3.8	Aplicación e implementación.....	68
3.8.1	<i>Detalles del desarrollo de la aplicación</i>	68
Capítulo 4. Conclusiones y trabajos futuros		77
4.1	Conclusiones	77
4.2	Trabajos futuros.....	78
4.2.1	<i>Mejora del generador sintético de planos</i>	78
4.2.2	<i>Solución al problema del recorte de tolerancias</i>	78
4.2.3	<i>Desarrollo de una webapp</i>	79
Capítulo 5. Bibliografía		80

Índice de figuras

Figura 1: Diagrama de flujo del proyecto.....	14
Figura 2: Representación visual de una capa convolucional [7]	18
Figura 3: Funcionamiento de la arquitectura YOLO [13]	19
Figura 4: Comparación de YOLO, Faster R-CNN y SSD [15].....	20
Figura 5: Estructura de YOLOv4 [17].....	20
Figura 6: Evaluación de YOLOv4 frente a otras arquitecturas [17].....	21
Figura 7: Estructura de R-CNN [20]	22
Figura 8: Estructura de Fast R-CNN [21].....	22
Figura 9: Estructura de Faster R-CNN [22]	23
Figura 10: Detección de una escalera de incendios en un plano arquitectónico [26].....	24
Figura 11: Detección de objetos en diagramas de tuberías e instrumentación [27]	25
Figura 12: Detección de vistas en un plano de ingeniería [29]	25
Figura 13: Estructura de una GAN [36]	27
Figura 14: Datos sintéticos generados a partir de uno real [38]	28
Figura 15: Resultados obtenidos con datos reales y sintéticos [38]	28
Figura 16: Ejemplos de variabilidad en un símbolo [39]	29
Figura 17: Restricciones en planos arquitectónicos. La puerta debe hallarse en la conexión entre dos habitaciones, la ventana debe dar al exterior y la cama debe estar alineada con la pared [40].....	30
Figura 18: Ejemplo de tolerancia geométrica y explicación de sus posibles componentes [41]	33
Figura 19: Ejemplos de símbolos empleados en tolerancias geométricas [41]	33
Figura 20: Símbolo de planicidad.....	34
Figura 21: Símbolo de paralelismo.....	34
Figura 22: Símbolo de perpendicularidad	34
Figura 23: Símbolo de posición.....	35
Figura 24: Símbolo de concetricidad.....	35
Figura 25: Símbolo de excentricidad circular	35

Figura 26: Función de pérdida CIoU [42]	36
Figura 27: Formato YOLO del txt que debe acompañar a la imagen (coordenadas, ancho y largo relativos a las dimensiones del plano).....	37
Figura 28: Concepto de IoU (Intersection over Union) [43].....	38
Figura 29: Ejemplos de TP, FP y FN en Computer Vision [44]	38
Figura 30: Ejemplo de una curva Precision-Recall [45].....	41
Figura 31: Ejemplo de plano sin tolerancias de interés en la carpeta Planos_Fuentes.....	42
Figura 32: Ejemplos de imágenes de las tolerancias de interés en la carpeta Tol_Recortadas	43
Figura 33: Ejemplos de imágenes de símbolos de ruido de la carpeta Tol_Recortadas_ruido	43
Figura 34: Ejemplos de imágenes generadas artificialmente en la carpeta Output_Planes.	43
Figura 35: Fichero txt generado junto a la imagen sintética de la carpeta Labels.....	44
Figura 36: Ejemplo de un fichero txt ya corregido de la carpeta Corrected_labels	44
Figura 37: Imágenes sintéticas con las tolerancias recuadradas a partir del txt generado... 45	
Figura 38: Ejemplos de imágenes sintéticas con agrupaciones de 2 y 3 tolerancias.....	48
Figura 39: Ejemplo de tolerancia recortada en el proceso de preparación del set de datos reales.....	50
Figura 40: Distribución de ejemplos por clase en el set de datos de imágenes reales.....	51
Figura 41: Distribución de ejemplos por clase en el set de datos de imágenes sintéticas ...	52
Figura 42: Distribución de ejemplos por clase en el set de datos híbrido	53
Figura 43: Distribución de ejemplos por clase en el set de datos de test	54
Figura 44: Gráfica del entrenamiento del Modelo I hasta las 12000 iteraciones	57
Figura 45: Gráfica del entrenamiento del Modelo I hasta las 48000 iteraciones	58
Figura 46: Gráfica de entrenamiento del Modelo II.....	59
Figura 47: Gráfica de entrenamiento del Modelo III.....	60
Figura 48: Menú de la GUI	69
Figura 49: Ejemplo de un plano que se encuentra en la carpeta PNG_Plane.....	70
Figura 50: Ejemplo de un recorte que se encuentra en la carpeta Cropped_planes	70
Figura 51: CSV que contiene los datos de los recortes de un plano.....	71

Figura 52: Ejemplo de recorte que se encuentra en la carpeta Output_yolo	71
Figura 53: Ejemplo de un plano que se encuentra en la carpeta Planos_Reconstruidos	72
Figura 54: Ejemplo de mensaje de error devuelto por la herramienta.....	73
Figura 55: Restricción en las extensiones a la hora de seleccionar archivos individuales..	74
Figura 56: Mensaje que salta cuando algún archivo de la carpeta no tiene una extensión válida	74
Figura 57: Selección de múltiples archivos de manera no masiva	75

Índice de tablas

Tabla 1: Resultados con datos reales	61
Tabla 2: Resultados iniciales	62
Tabla 3: Resultados tras las mejoras	64
Tabla 4: Resultados de los modelos SYNTH	66
Tabla 5: Resultados de los modelos HYBRID	67

Capítulo 1. INTRODUCCIÓN

En los últimos años, la Inteligencia Artificial ha visto un crecimiento exponencial en todos los sectores profesionales, ofreciendo ventajas increíblemente innovadoras en numerosos ámbitos. La digitalización de procesos que hasta hace unos años se realizaban de manera manual ha permitido mejorar la eficiencia y realización de los mismos. El mundo de la IA es un mundo que, a pesar de los avances que ha experimentado en estos años, está por explorar; si bien en los últimos meses se está identificando toda la IA con la IA generativa por el auge de los grandes modelos de lenguaje como Chat GPT, lo cierto es que sus posibilidades van más allá, y la IA discriminativa sigue indudablemente teniendo potencial para mejorar innumerables procesos.

Una de las aplicaciones donde la Inteligencia Artificial se ha mostrado más útil es en la detección de objetos y en *Computer Vision*. La idea de que una máquina sea capaz de identificar y clasificar objetos de manera autónoma, sin necesidad de ayuda humana, se presenta como increíblemente disruptiva, pues abre la puerta a infinitud de nuevas herramientas y ventajas.

Una vez entendido el contexto, este proyecto busca desarrollar y analizar posibles herramientas que permitan la integración de este tipo de tecnología en el análisis de planos ingenieriles, en concreto de mecanismos y demás piezas fabricadas en un entorno industrial, de manera más sencilla y con la mayor calidad posible.

1.1 MOTIVACIÓN DEL PROYECTO

El sector de manufactura industrial siempre ha estado marcado por el constante deseo de mejorar sus procesos de producción. En medio de la Industria 4.0, la llegada de nuevas tecnologías a este ámbito pretende abrir nuevos horizontes, buscando mejorar aún más los procesos de fabricación actuales.

Este proyecto nace de la intención de convertir los procesos de manufactura en procesos más eficientes a múltiples niveles.

Los planos ingenieriles representan el “manual de instrucciones” de un proceso de fabricación, conteniendo todas las dimensiones y tolerancias necesarias para la correcta manufactura de un producto. Así, su análisis requiere de precisión y completitud.

La incorrecta o incompleta traducción de un plano y sus tolerancias puede derivar en graves consecuencias, tanto en el ámbito industrial como en el económico.

Así, el proyecto busca, ante todo, analizar la viabilidad de implementar una tecnología que pueda mejorar un proceso de producción en el contexto descrito. Esta tecnología puede servir de apoyo para los operarios de oficina técnica, automatizando la extracción de información de planos empleados en el sector industrial, algo que hasta la fecha no es posible con las técnicas de análisis clásicas.

Actualmente, dentro del fenómeno de la Industria 4.0, los fabricantes están empezando a introducir nuevas tecnologías en sus procesos de producción, buscando una mayor eficiencia y una reducción de costes. A día de hoy, los procesos de producción industrial cuentan con una fase manual de detección de la información presente en los planos que se utilizan en la fabricación de piezas; la automatización de esta etapa, sobre todo en industrias que fabrican cientos de elementos diferentes cada mes, puede traer enormes ahorros y ventajas.

1.1.1 VENTAJAS EN EL ÁMBITO INDUSTRIAL

En el ámbito industrial, un incorrecto análisis de un plano derivado del error humano puede traducirse en una menor calidad en los lotes fabricados. Además, no analizar debidamente las tolerancias puede derivar en que los productos salgan defectuosos, al no cumplir con los requisitos especificados en el plano.

Una tecnología debidamente desarrollada reduce el error en el análisis de las tolerancias, permitiendo una mayor precisión en la detección de estas, recopilando información que puede ser posteriormente transferida a las siguientes etapas de la cadena de producción.

1.1.2 VENTAJAS EN EL ÁMBITO ECONÓMICO

En el ámbito económico, y unido al ámbito industrial, la posibilidad de que se produzcan lotes defectuosos por no cumplir debidamente con las tolerancias especificadas puede traducirse en pérdidas monetarias, retrasos en las entregas y el deterioro de la imagen del fabricante.

El desarrollo de una tecnología, cuyos únicos costes sean los destinados a su concepción inicial, diseño y entrenamiento, puede ayudar a reducir costes y/o distribuir el capital humano de manera más eficiente, pudiendo emplear a los trabajadores inicialmente destinados a esta tarea a otras actividades más productivas.

1.2 JUSTIFICACIÓN DEL PROYECTO

La satisfacción de la motivación de este proyecto puede lograrse mediante la implementación de una tecnología basada en *Computer Vision*, perteneciente al campo del *Deep Learning*.

La detección de objetos mediante *Computer Vision* se puede llevar a cabo mediante el uso de redes convolucionales (también denominadas CNNs, por su nombre en inglés, *Convolutional Neural Networks*). Estas redes se basan en la aplicación sucesiva de la convolución, una operación matricial que permite modificar y alterar matrices. Dado que una imagen es, al fin y al cabo, una matriz numérica donde cada número implica un color en un determinado píxel, la aplicación de estas redes al tratamiento de imágenes es especialmente interesante.

El empleo de técnicas de *Deep Learning* en el sector industrial es un proceso que lleva varios años en desarrollo. Las redes neuronales aplicadas a la detección de objetos en planos industriales se presentan como un avance de potencial utilidad en el sector, con un único inconveniente, la necesidad de una gran cantidad de datos para su desarrollo, que este proyecto pretende solucionar.

1.2.1 SOLUCIÓN ANTE LA FALTA DE EJEMPLOS

El entrenamiento de una red en este contexto presenta una gran limitación: la falta de planos para su entrenamiento. Una red neuronal requiere de un set de datos considerablemente grande

para obtener resultados aceptables. Sin embargo, la obtención de un set de datos de este calibre se presenta complicado, pues no todas las empresas están dispuestas a ceder sus planos para el entrenamiento de la red, unido a que no todos los planos poseen tolerancias que puedan ser usadas en el entrenamiento del modelo.

Ante esta limitación, este proyecto busca analizar la viabilidad de usar imágenes sintéticas, obtenidas de manera artificial con un generador de planos, en el entrenamiento de la red neuronal, observando si es posible construir una red eficiente a partir de planos creados de forma sintética.

1.3 OBJETIVOS DEL PROYECTO

En base a lo expuesto anteriormente, el objetivo principal de este proyecto es **mejorar el rendimiento de un sistema de detección de símbolos en planos de ingeniería basado en la tecnología YOLO**.

Este objetivo es el centro alrededor del cual gira el proyecto. La idea básica y fundamental del trabajo es estudiar la posibilidad de mejorar una tecnología destinada a la detección de tolerancias en planos de piezas industriales mediante el uso de imágenes sintéticas. Así, se busca analizar la efectividad de emplear imágenes artificiales en la configuración de un sistema aplicado al caso que se considera.

De este objetivo, nacen cinco tareas principales o subobjetivos, que darán forma a la metodología que se seguirá a lo largo del proyecto y que se detallará en posteriores secciones.

1º Perfeccionamiento y puesta en marcha del generador de planos sintéticos

Para la realización de este Trabajo de Fin de Grado, se ha proporcionado una primera versión ya desarrollada de un generador de imágenes sintéticas.

Así, el primer paso es perfeccionar dicho generador para capturar, de la mejor manera posible, la variabilidad que un plano ingenieril puede presentar en el mundo real.

El análisis de los resultados obtenidos a partir de las imágenes artificiales servirá de retroalimentación, introduciendo las mejoras que se consideren necesarias para mejorar el funcionamiento del generador.

2º Creación de los sets de datos

El segundo paso será la creación de los sets de datos para el entrenamiento de los modelos. Se entrenarán diferentes modelos con sets de datos tanto reales como sintéticos e híbridos.

En primer lugar, se deberá generar el set de datos de imágenes reales, a partir de un conjunto de planos reales destinados a este proyecto.

En segundo lugar, se generará el set de datos de imágenes sintéticas a partir del generador de planos creado para este fin.

3º Entrenamiento de los modelos

Una vez obtenidos los sets de datos requeridos, se procederá al entrenamiento de los modelos a analizar. Se partirá de una red neuronal ya desarrollada, con unos pesos iniciales, aplicando *fine-tuning* para obtener nuevos pesos a partir de cada uno de los sets de datos.

En un principio, se entrenarán 3 modelos diferentes:

1. Modelo entrenado con el set de datos de imágenes reales.
2. Modelo entrenado con el set de datos de imágenes sintéticas.
3. Modelo entrenado con un set de datos híbrido, que incluya tanto las imágenes reales como las artificiales.

4º Análisis y comparación de los modelos resultantes

Después del entrenamiento de los diferentes modelos, se procederá a su análisis y comparación para poder extraer conclusiones y resultados.

En primer lugar, se estudiará la viabilidad de emplear imágenes sintéticas, observando los resultados tanto del modelo entrenado con estas como del modelo híbrido, para determinar si dichas imágenes pueden contribuir al entrenamiento de una red neuronal de este tipo.

En segundo lugar, se compararán los resultados de los tres modelos, buscando aquel que sea lo suficientemente fiable y preciso para una posible aplicación en una cadena de producción industrial.

5º Desarrollo de una aplicación para el análisis de planos

Una vez obtenido un modelo lo suficientemente bueno y preciso, se procederá al desarrollo de una aplicación que sirva como prototipo para una potencial aplicación destinada al análisis de planos en un proceso industrial real.

1.4 OBJETIVOS DE DESARROLLO SOSTENIBLE

El proyecto aborda cuestiones relacionadas con dos de los Objetivos de Desarrollo Sostenible, 17 objetivos propuestos por la ONU en 2015, destinados a fomentar la consecución de un futuro mejor y un desarrollo sostenible.

1.4.1 OBJETIVO 9. INDUSTRIA, INNOVACIÓN E INFRAESTRUCTURA

Este TFG persigue el desarrollo de una tecnología destinada a la mejora de los procesos de manufactura; concretamente, busca innovar en una de las fases de una línea de producción. Este proyecto pretende combinar la industria tradicional con nuevas tecnologías disruptivas, como las implementadas en el ámbito del *Deep Learning*, centrándose en la innovación alineada con el fenómeno de la Industria 4.0 que se lleva produciendo en las últimas décadas.

1.4.2 OBJETIVO 8. TRABAJO DECENTE Y CRECIMIENTO ECONÓMICO

Alineado con el ODS número 9, este TFG, al perseguir una mejora de una tecnología posiblemente aplicable a un proceso de producción industrial, puede ayudar a fomentar el crecimiento económico del sector. La fase de análisis de un plano requiere de mano de obra destinada únicamente a esta etapa, y no siempre es del todo eficiente. La implementación de una tecnología dedicada a reconocer tolerancias y símbolos en el plano que se va a emplear para producir una pieza puede ayudar a incrementar la eficiencia de todo el proceso industrial,

permitiendo una mejor distribución de la mano de obra y reduciendo la tasa de defectos o los retrasos debidos a una falta de precisión en esta etapa.

1.5 METODOLOGÍA

Las tareas previamente expuestas en la sección Objetivos del proyecto marcan claramente el camino que se seguirá en el desarrollo del proyecto. El proyecto tendrá un desarrollo cíclico, pues se utilizarán los resultados obtenidos con el generador para entender qué limitaciones presenta y qué tinte deben adquirir las mejoras a implementar en el mismo.

A partir de las cinco tareas mencionadas, el proyecto se realizará en tres etapas fundamentales: la obtención de los modelos (basada en las primeras tres tareas), su posterior análisis y estudio (cuarta tarea) y el desarrollo de una herramienta para su uso con planos industriales reales (quinta tarea).

1º etapa: Obtención de los modelos

En primer lugar, se modificará la versión existente del generador de planos para su mejora y mayor efectividad. El generador ha sido inicialmente programado en Python, por lo que se continuará con este lenguaje para los retoques que se le realicen. Se entenderá su funcionamiento, la lógica detrás de los ficheros iniciales y se añadirán aquellas mejoras que perfeccionen su rendimiento, permitiendo así producir imágenes sintéticas que se ajusten de manera fiel a la realidad. Existirá un proceso de realimentación a partir de los resultados obtenidos en la evaluación de los modelos, de manera que se logre mejorar el funcionamiento del generador a partir del estudio cualitativo de las deficiencias que se identifiquen.

El set de datos de imágenes reales se generará recortando planos ingenieriles reales, ajustando los recortes a las dimensiones requeridas por la red.

El set de datos de imágenes sintéticas se obtendrá con el generador ya mencionado.

Los modelos se basarán en una red neuronal ya existente. Se usará un set de datos específico para cada uno de ellos, destinando una parte al entrenamiento y otra parte a la validación.

2º etapa: Evaluación de los modelos

El estudio y comparación de los modelos se realizará con un set de datos destinado únicamente al test. Se empleará un conjunto adicional de planos reales para evaluar la precisión de los modelos y compararlos entre ellos, a través de métricas comúnmente utilizadas en el ámbito de *Computer Vision*. Como ya se ha mencionado, las etapas 1 y 2 se realizarán de manera cíclica, buscando mejorar el proceso de generación de imágenes sintéticas.

3º etapa: Desarrollo de una aplicación

Se desarrollará una aplicación, a modo de prototipo básico, que pase un plano por la red neuronal y detecte las tolerancias, devolviendo el mismo plano con las tolerancias recuadradas. El lenguaje que se empleará para esto será Python (aunque el lenguaje podrá ser sustituido por otro si se considera según se vaya progresando).

1.5.1 DEFINICIÓN DEL PROYECTO

Este proyecto se enmarca en el ámbito del *Deep Learning*. Como resultado, se escogió la estructura característica de este tipo de trabajos, adaptado a las necesidades específicas del proyecto. Las fases del proyecto se aprecian en la Figura 1.

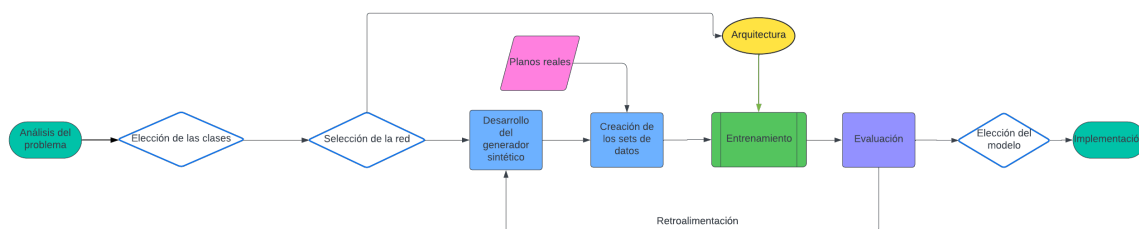


Figura 1: Diagrama de flujo del proyecto

1.6 PLANIFICACIÓN Y ESTIMACIÓN ECONÓMICA

El cronograma inicial que se pretenderá seguir para la consecución del proyecto se presenta a continuación.

HITOS	FECHAS
Creación del set de datos de imágenes reales	15 de mayo 2023-31 de mayo 2023
Retoque del generador sintético final*	31 de mayo 2023-1 de febrero 2024
Creación del set de datos de imágenes sintéticas*	31 de mayo 2023-1 de febrero 2024
Entrenamiento de los modelos*	31 de mayo 2023-1 de febrero 2024
Análisis y evaluación*	31 de mayo 2023-1 de febrero 2024
Desarrollo de la aplicación	1 de febrero 2024-1 de mayo 2024

*Estos cuatro hitos se enmarcan en las mismas fechas, dado que es un proceso cíclico, buscando realimentar el comportamiento del generador

Capítulo 2. ESTADO DE LA CUESTIÓN

2.1 INTRODUCCIÓN

La integración del *Deep Learning* (de ahora en adelante, DL) en el reconocimiento de imágenes y símbolos ha sido la base fundamental de los avances más recientes que se han producido en el sector del aprendizaje automático y, más particularmente, en *Computer Vision* [1]. Especialmente en ingeniería, donde la precisión es primordial, la detección automática y la interpretación de símbolos en planos de ingeniería utilizando métodos de DL presentan una oportunidad transformadora y disruptiva en el sector, pudiendo ser el siguiente gran paso en la Industria 4.0, dadas las ventajas que ofrecen en términos de precisión y facilidad de desarrollo (dado que únicamente requiere de una fase de entrenamiento para su obtención) [1]. Esta revisión examina críticamente el panorama actual de las aplicaciones de la detección automática en el reconocimiento de símbolos, centrándose en el uso de imágenes de entrenamiento reales frente a las generadas sintéticamente. El análisis del DL, desde sus fundamentos conceptuales hasta su aplicación en planos de ingeniería, está marcado por hitos significativos, desafíos y posibles direcciones futuras.

2.2 DEEP LEARNING EN LA INDUSTRIA 4.0

La aplicación del DL en el reconocimiento de símbolos no se limita a la investigación académica; tiene profundas implicaciones para la Industria 4.0. En las últimas décadas se ha comenzado la integración de nuevas tecnologías en procesos industriales, como el uso de CMMs o *Coordinate Measuring Machines*, indicando un cambio hacia sistemas más automatizados e inteligentes [2]. La automatización del análisis de planos de ingeniería utilizando DL se alinea con esta tendencia, ofreciendo mejoras potenciales en eficiencia, precisión y flexibilidad en los procesos de fabricación, en comparación a los métodos tradicionales [3].

2.3 EVOLUCIÓN DEL DEEP LEARNING EN EL RECONOCIMIENTO DE SÍMBOLOS

2.3.1 REDES NEURONALES CONVOLUCIONALES (O CONVOLUTIONAL NEURAL NETWORKS, CNNs)

El campo de *Computer Vision* incluye aquellos algoritmos que buscan e identifican patrones en una imagen [1]. A día de hoy, la detección de objetos mediante *Computer Vision* se lleva a cabo habitualmente mediante el uso de redes neuronales convolucionales (o CNNs por sus siglas en inglés); si bien nuevas tecnologías como el transformer de visión [4] están ganando fuerza en este ámbito, las CNNs siguen siendo relativamente eficientes a nivel de computación, especialmente en la fase de entrenamiento, gracias a la facilidad para ser entrenadas empleando GPUs [5] [6].

Las redes neuronales convolucionales se basan en la aplicación sucesiva de la convolución, una operación matricial que permite modificar matrices. Estas redes están compuestas por diferentes capas, cada una de las cuales contiene una matriz, denominada filtro. Cuando a la red se le introduce una matriz (o vector) de información, esta pasa por cada una de las capas, siendo modificada mediante la convolución a partir de los diferentes filtros. Los filtros, por tanto, pueden servir para extraer un determinado *output* de interés, a partir de la información de entrada. La clave de esta tecnología reside, por ende, en buscar qué números debe contener cada matriz para devolver la salida de interés. A cada uno de estos números se les denomina pesos, y para obtenerlos, se requiere de un proceso de entrenamiento, en el que la red aprende qué pesos son los que mejor funcionan [7] [8].

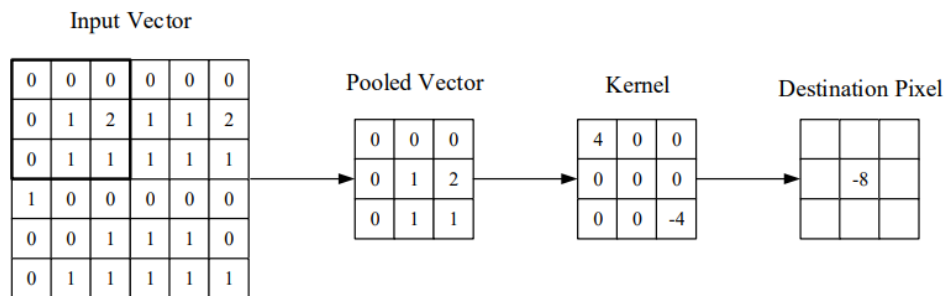


Figura 2: Representación visual de una capa convolucional [7]

Dado que una imagen es, al fin y al cabo, una matriz numérica, la aplicación de estas redes al tratamiento de imágenes se presenta como una oportunidad disruptiva.

Las CNNs, con su capacidad para extraer características de las imágenes mediante filtros convolucionales, se han convertido en la columna vertebral de muchos sistemas de reconocimiento de imágenes, mediante diferentes procesos y técnicas que se abordarán más adelante. En el contexto del reconocimiento de símbolos en planos de ingeniería, la capacidad de las CNNs para reconocer patrones y formas puede resultar excepcionalmente beneficiosa.

2.3.1.1 Ejemplos de Redes Neuronales

El panorama de la detección automática es rico en diversas arquitecturas y algoritmos, cada uno con sus puntos fuertes y débiles. Algunas de las redes más famosas en el campo de *Computer Vision* son LeNet-5 [9], la primera red neuronal plenamente entrenable, la cual estaba orientada al reconocimiento de dígitos y caracteres escritos, AlexNet [10], la cual marcó un gran avance reduciendo enormemente la tasa de error obtenida en el *ImageNet Large Scale Visual Recognition Challenge* de 2010, o VGG-16 [11], revolucionaria en la época por contar con hasta 16 capas convolucionales. Otro tipo de arquitectura muy empleada en el mundo de la detección de imágenes es la usada en las ResNet o *Residual Networks* [12], la cual permite que la red, al contar con las llamadas conexiones residuales, pueda contener un mayor número de capas y reducir aún más el error obtenido, solucionando el problema del desvanecimiento y la explosión del gradiente que se observa en otras arquitecturas.

2.3.1.2 YOLO (You Only Look Once)

Una red destinada a la localización y clasificación de objetos no es eficiente si no emplea una estructura apropiada para descomponer una imagen y localizar los objetos que en ella se encuentran. A día de hoy, la arquitectura YOLO, desarrollada por Redmon et al en su publicación “You Only Look Once: Unified, Real-Time Object Detection” [13], es la más famosa y la más utilizada en este tipo de redes.

A diferencia de otro tipo de algoritmos empleados en la detección de objetos, como el método de la ventana deslizante, la arquitectura YOLO realiza la detección mediante un proceso en el que la imagen se divide según una cuadrícula, detectando objetos en cada una de las sub-imágenes de manera independiente. A partir de esta detección, se crea un mapa de probabilidades que permite a la red recuadrar los objetos detectados.

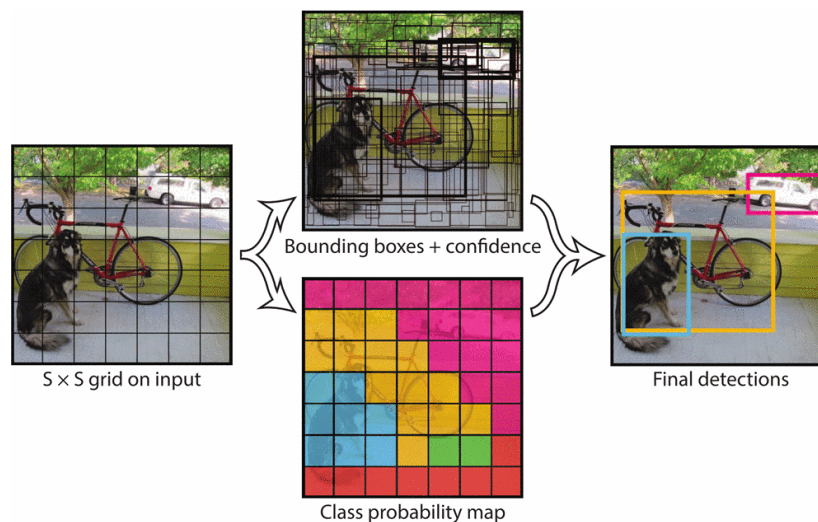


Figura 3: Funcionamiento de la arquitectura YOLO [13]

La arquitectura YOLO destaca por una gran rapidez, precisión y generalización, algo que queda corroborado en diversos estudios que analizan las diferentes versiones de YOLO hasta la fecha (v1-v7); en ellos, quedan constatados los buenos resultados que ofrece tanto en *accuracy* como en velocidad de inferencia, dejando clara su potencial aplicación en la detección de objetos en tiempo real [14]. Otros estudios comparativos llegan a la misma conclusión, mostrando la superioridad de YOLO ante otras arquitecturas como Faster R-CNN o SSD [15] [16].

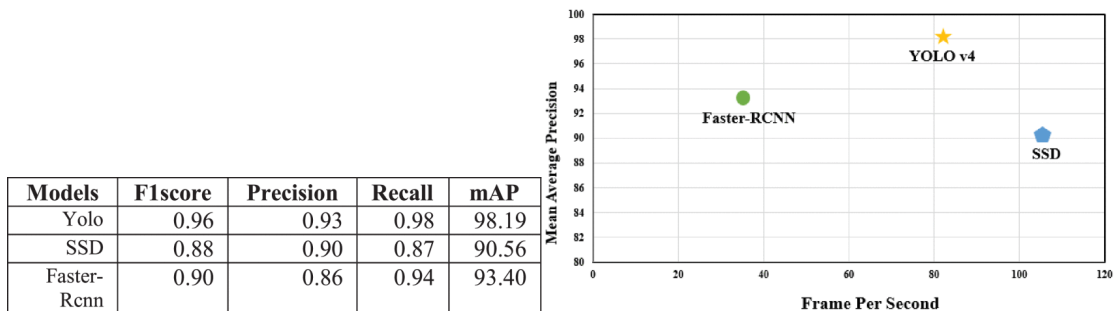


Figura 4: Comparación de YOLO, Faster R-CNN y SSD [15]

2.3.1.2.1 YOLOv4

En este proyecto se empleó una red neuronal con la versión YOLOv4. Esta arquitectura mejora las anteriores versiones en términos de *accuracy*, eficiencia y velocidad, centrándose en resolver las limitaciones de las versiones anteriores en cuanto a la detección de objetos pequeños [17].

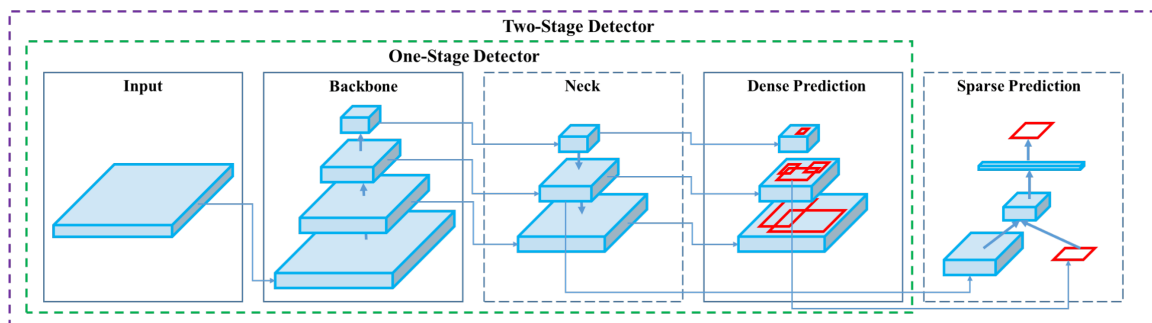


Figura 5: Estructura de YOLOv4 [17]

YOLOv4 hace uso de técnicas *bag of freebies* para mejorar la *accuracy* del modelo sin incrementar el coste de inferencia. El típico ejemplo de este tipo de técnicas es *Data augmentation*, técnica que incrementa la cantidad de datos empleados por la red mediante la modificación de los datos ya disponibles, incrementando la variabilidad recogida por los datos empleados en el entrenamiento y buscando una mayor generalización del modelo.

Adicionalmente, se trata de una versión de YOLO que puede ser usada y entrenada en una sola GPU convencional, lo que facilita su uso.

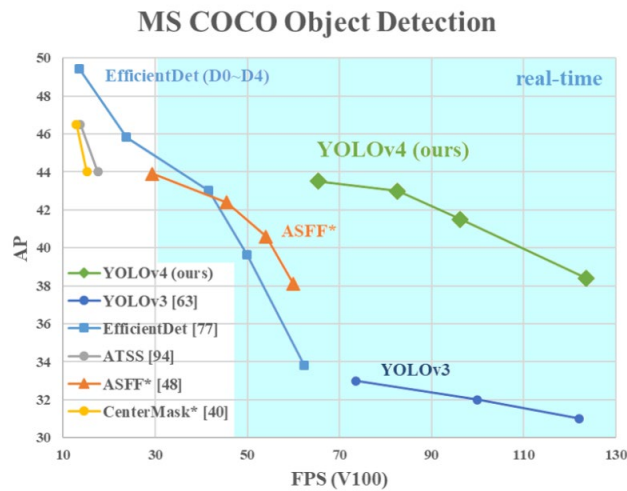


Figura 6: Evaluación de YOLOv4 frente a otras arquitecturas [17]

La red neuronal comúnmente empleada con YOLOv4, bajo el nombre de DarkNet, es la que se emplearía en la realización de este proyecto [17].

2.3.1.3 Otras arquitecturas empleadas en la detección de objetos

2.3.1.3.1 Ventana deslizante (Sliding Window)

El método de la ventana deslizante, comúnmente conocido en el ámbito de *Computer Vision*, se basa en el uso de una caja rectangular que se va moviendo a lo largo de la imagen; de esta manera, la red neuronal analiza en cada zona recuadrada la existencia o no existencia de un objeto de interés [18].

Se trata de un método sencillo pero que requiere un gran esfuerzo computacional.

La ventana deslizante fue uno de los primeros métodos propuestos para la localización y detección de objetos, siendo inicialmente implementado en una red destinada a detectar peatones, lo que exigía tener que reconocer a varias personas en una única fotografía [19].

2.3.1.3.2 Region proposals

Otro método para la localización y clasificación de objetos es el de *Region proposals*. Cuando se emplea el método de la ventana deslizante, una cuadrícula se va moviendo a lo largo de una

imagen, analizándola por completo. No obstante, puede darse que haya un gran número de zonas “vacías” y que, por tanto, no tiene sentido analizar. Este método, por ende, busca descartar aquellas regiones que no contienen objetos. Mediante un algoritmo de segmentación, se proponen una serie de regiones de estudio de la imagen, mediante el análisis de bordes, cambios de color, etc. Una vez descartadas las regiones “vacías”, la red trata de localizar el objeto en las regiones restantes [20]. A partir de este método han surgido diversas arquitecturas:

- R-CNN [20]: se clasifican las regiones propuestas una a una.

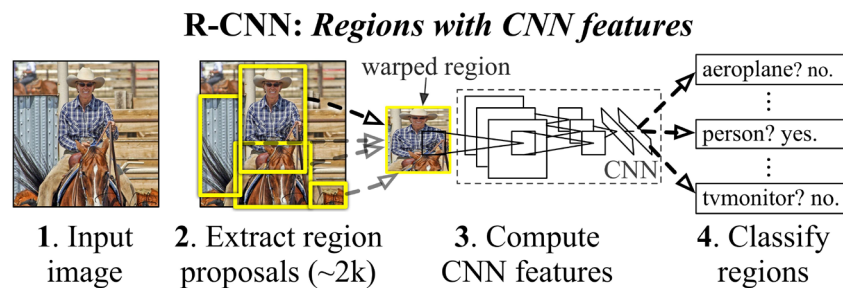


Figura 7: Estructura de R-CNN [20]

- Fast R-CNN [21]: inicialmente se aplica a la imagen una serie de operaciones convolucionales que genera un mapa de características; una vez hecho esto, la red neuronal aplica una técnica, denominada *RoIPooling*, para enfocarse únicamente en las áreas de interés. Esto permite compartir cálculo y memoria entre regiones, incrementando la velocidad con respecto a R-CNN.

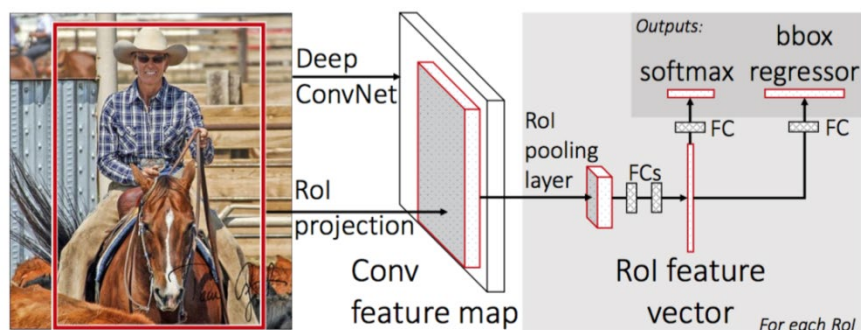


Figura 8: Estructura de Fast R-CNN [21]

- Faster R-CNN [22]: Faster R-CNN añade una RPN (*Regional Proposal Network*), una red neuronal que, integrada con la red de detección, permite la propuesta de regiones de manera prácticamente instantánea, eliminando la necesidad de un algoritmo externo que proponga las regiones, lo cual acelera enormemente el proceso de detección. Así, Faster R-CNN nace de la fusión de la RPN con una red Fast R-CNN.

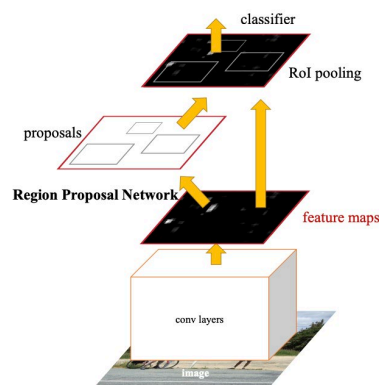


Figura 9: Estructura de Faster R-CNN [22]

El método de Region proposals y, en concreto, la estructura de Faster R-CNN ofrece grandes resultados en términos de velocidad y *accuracy*. No obstante, la comparación entre el método YOLO y Faster R-CNN muestra que el primero se presenta como más preciso, sin perder velocidad de procesamiento respecto al segundo [15] [16] [23]. Otra gran diferencia entre Faster R-CNN y YOLO radica en su estructura. Mientras la primera requiere del entrenamiento tanto de la RPN como de la red Fast R-CNN, el YOLO solo cuenta con una única estructura, lo que simplifica la tarea de obtención de los pesos adecuados para su posterior uso [24].

2.3.1.3.3 Otros métodos

El mundo del *Computer Vision* y de la localización y detección de objetos se ha convertido en un campo ampliamente estudiado. Son muchas las publicaciones que proponen nuevas arquitecturas dedicadas al reconocimiento de objetos en imágenes.

Por ejemplo, se han propuesto estructuras alternativas como las redes neuronales sin peso (o WNNs, por sus siglas en inglés, *Weightless Neural Networks*). Se trata de redes neuronales que no utilizan pesos para almacenar información. En cambio, usan patrones de activación o

memorias basadas en reglas, empleando técnicas de discretización de los datos, convirtiéndolo a lenguaje binario para su almacenamiento en bits [25].

2.3.2 DEEP LEARNING PARA EL RECONOCIMIENTO DE PATRONES COMPLEJOS

Los planos de ingeniería son intrínsecamente complejos y presentan patrones y símbolos (como los que describen algún tipo de tolerancia geométrica) que son difíciles de descifrar. Se ha demostrado que los modelos de DL, en particular las versiones avanzadas de las CNNs, destacan en este tipo de tareas de reconocimiento de patrones complejos.

Numerosos artículos analizan la efectividad de arquitecturas como YOLO o Faster R-CNN en la detección de elementos complejos y, en particular, de elementos en planos ingenieriles o arquitectónicos.

Uno de estos artículos estudia el comportamiento de una red neuronal Faster R-CNN aplicada a la detección de objetos en planos arquitectónicos, destinada al análisis de la seguridad frente a incendios de un edificio, mostrando la adaptabilidad que las técnicas de DL presentan a planos complejos [26].

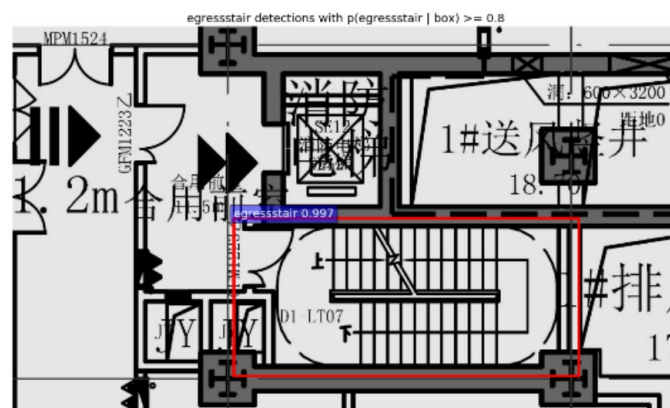


Figura 10: Detección de una escalera de incendios en un plano arquitectónico [26]

Otras publicaciones recogen un análisis de arquitecturas como YOLO y Faster R-CNN en el proceso de detección de objetos en diagramas de tuberías e instrumentación, centrándose en la detección de bombas, válvulas y muchos otros elementos. Ambas arquitecturas arrojan resultados prometedores en la detección de símbolos tanto grandes como pequeños, siendo

YOLO la que mejor resultados muestra en este estudio; no obstante, ambas estructuras presentan ciertos problemas a la hora de clasificar objetos de formas o tamaños inusuales [27] [28].

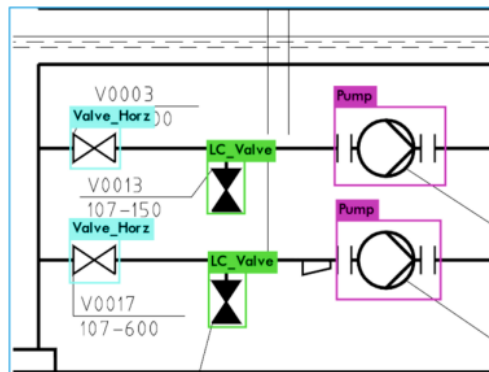


Figura 11: Detección de objetos en diagramas de tuberías e instrumentación [27]

Yendo más allá, otros investigadores no solo analizan la efectividad del YOLO en su publicación, sino que también buscan en el DL una manera de afrontar los retos e inconsistencias derivadas de una interpretación manual de los planos ingenieriles. La falta de estandarización en aspectos como el formato o la disposición de los símbolos en un plano de ingeniería puede derivar en un mayor error humano en su análisis, derivando en una mayor ineficiencia e imprecisión en su interpretación. Así, aplican las redes neuronales al análisis de planos ingenieriles, detectando vistas (85% *accuracy*), anotaciones (70% *accuracy*), o textos y símbolos (80% *accuracy*) [29].

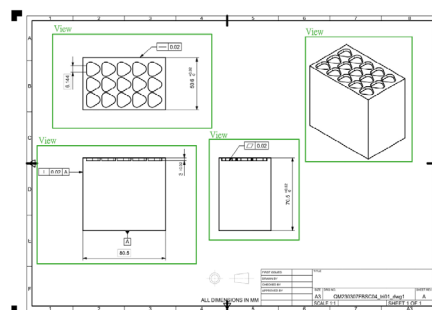


Figura 12: Detección de vistas en un plano de ingeniería [29]

Varios también mencionan la necesidad de continuar investigando ante el reto que supone la detección de objetos en planos ingenieriles complejos debido a la gran variedad de símbolos que pueden hallarse en estos, las diferencias de tamaño o la presencia de texto [30].

El estudio del uso del DL en planos ingenieriles también propone nuevos enfoques en este campo, como la combinación de modelos topológicos con redes neuronales en la digitalización de planos de construcción con el objetivo de transferir información a un modelo del producto [31], la búsqueda de no solo los símbolos sino también su conexión entre ellos [32], o su implementación en ámbitos como la ingeniería estructural [33].

Más allá del DL, otros algoritmos de aprendizaje automático también desempeñan un papel importante en la clasificación de símbolos. Numerosos artículos realizan análisis de varios algoritmos de aprendizaje automático en la clasificación de símbolos en planos de ingeniería, comparando las redes neuronales con otros métodos como el *Support Vector Machine* (SVM) o el *Random Forests* (RF), obteniendo resultados muy similares entre las tres tecnologías [34]. Tales análisis comparativos ayudan a comprender las limitaciones y ventajas de cada enfoque, guiando la selección del algoritmo más adecuado o una combinación de algoritmos para aplicaciones específicas en el análisis de planos de ingeniería.

2.3.3 DATOS SINTÉTICOS EN EL ENTRENAMIENTO DE REDES DE DEEP LEARNING

2.3.3.1 Cómo abordar la escasez de datos con imágenes sintéticas

Uno de los retos más importantes a la hora de entrenar modelos de DL para tareas específicas, como el reconocimiento de símbolos en planos de ingeniería, es la escasez de datos reales anotados. Los planos ingenieriles empleados en un proceso de producción son difíciles de conseguir; por cuestiones de propiedad intelectual, la donación de planos para el entrenamiento de una red neuronal es escasa, a lo que se une el hecho de que muchos planos carecen de los símbolos de interés para el entrenamiento.

Ante esta problemática, se han buscado alternativas, entre las que destaca la generación de planos o imágenes sintéticas. Mediante la creación de imágenes de manera artificial, el problema de la escasez de ejemplos para el entrenamiento de una red queda resuelto, siempre y

cuando estas imágenes se asemejen significativamente a las que se podrían encontrar en el mundo real.

En las últimas décadas, la búsqueda de métodos para solventar el problema de la falta de datos para el entrenamiento de modelos de DL ha derivado en el desarrollo de nuevas tecnologías destinadas a generar datos sintéticos realistas con los que entrenar una red neuronal.

Por ejemplo, algunos investigadores proponen un método semi-supervisado de aumento de un set de datos etiquetados mediante técnicas de transducción gráfica, logrando resultados exitosos en términos de mejora de los resultados obtenidos en determinadas arquitecturas de redes neuronales [35].

Otro ejemplo son las redes neuronales adversariales (GAN). En su publicación, varios investigadores buscan emplear este tipo de redes, como las DCGAN (*Deep Convolutional GAN*) o las LSGAN (*Least Square GAN*), para generar datos sintéticos, con el objetivo de afrontar el problema de la falta de ejemplos para el entrenamiento de una red destinada a reconocer símbolos en planos y diagramas unifilares. La estructura de las GAN está conformada por un generador, encargado de crear las imágenes sintéticas, y un discriminador, el cual determina si una imagen es real o falsa en comparación con una imagen real; el objetivo del generador es “engañar” al discriminador, buscando generar imágenes lo suficientemente realistas para que sean clasificadas como reales por el discriminador. El uso de estas redes GAN les permitió aumentar el dataset de entrenamiento de la red, mejorando la *accuracy* de un 89% (set de datos original) a un 95% (set de datos ampliado) [36].

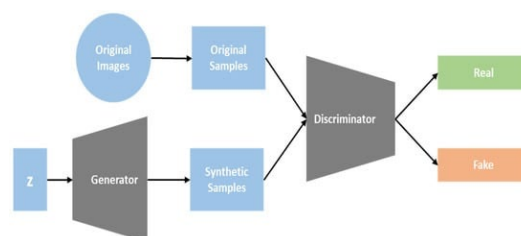


Figura 13: Estructura de una GAN [36]

Las GAN se presentan como método para combatir el desbalance de clases y la falta de planos reales y su aplicación en el entrenamiento de modelos de DL se trata en más de un artículo, arrojando buenos resultados [37].

Otro estudio muestra los resultados derivados de entrenar un modelo destinado a la detección de elementos de un plano (como líneas de contorno, medidas, líneas de dimensionado, etc) con un set de datos sintéticos, empleando un método restrictivo. A partir de una imagen real, un generador de datos sintéticos crea nuevas imágenes modificando el dimensionado del plano, restringiendo las nuevas dimensiones de manera que no se superpongan unas con otras (C1) y estas se hallen fuera del dibujo de la pieza (C2). Los resultados obtenidos de la comparación de múltiples modelos (uno entrenado con datos reales, otro entrenado con datos sintéticos sin restringir, otro entrenado con datos sintéticos restringidos con la C1, otro entrenado con datos sintéticos restringidos con la C2 y un último entrenado con datos sintéticos restringidos con la C1 y la C2 al mismo tiempo) permite concluir que el uso de datos sintéticos contribuye a mejorar la eficiencia y la generalización de un modelo cuando los datos reales son escasos [38].

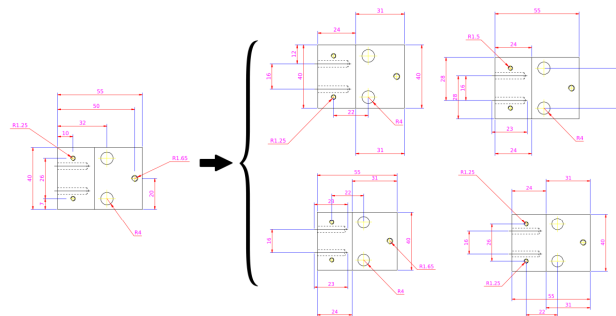


Figura 14: Datos sintéticos generados a partir de uno real [38]

Validation Accuracy %	RF	DT	MLP
No synthesis	58.19	56.50	45.74
Unconstrained	66.56	64.12	58.03
C1 only	82.12	81.31	70.65
C2 only	80.27	77.84	67.49
C1+C2(fully constrained)	87.52	86.29	76.84

Figura 15: Resultados obtenidos con datos reales y sintéticos [38]

2.3.3.2 Acortar la distancia entre los datos sintéticos y los reales

Los planos de ingeniería se caracterizan por su gran variabilidad, complejidad y, a menudo, oclusiones. Estos factores plantean importantes retos a los sistemas de reconocimiento automático.

En esta línea, algunos investigadores remarcaron en su publicación la necesidad de captar la variabilidad de los datos en un proyecto basado en la digitalización de diagramas esbozados a mano alzada mediante el uso de una red neuronal. Dado que los símbolos pueden ser dibujados de múltiples maneras por una persona, la generación de datos sintéticos, mediante la distorsión aleatoria de los símbolos de interés y su inclusión en un plano fuente, permitió que la red neuronal empleada entrenara con datos más variados, mejorando la generalización del modelo, que llegó a alcanzar una *accuracy* del 97.7% [39].



Figura 16: Ejemplos de variabilidad en un símbolo [39]

La eficacia de los datos sintéticos en el entrenamiento de modelos de este tipo depende de lo bien que estas imágenes puedan reproducir los escenarios del mundo real que los modelos acabarán encontrando. En la investigación para garantizar la fidelidad de los datos sintéticos, algunos autores exploraron en su trabajo métodos para hacer que las imágenes sintéticas fueran más realistas y representativas, estableciendo una serie de restricciones para mejorar el realismo de las imágenes generadas de manera artificial. Este método, que restringía aspectos como la rotación, la posición, o el tamaño de los símbolos, fue aplicado a planos arquitectónicos y electrónicos, obteniendo imágenes coherentes y realistas, adaptándose a los estándares y lógica de este tipo de planos [40].

La clave es capturar la variabilidad y la complejidad inherentes a los planos de ingeniería reales de manera realista, permitiendo así que los modelos entrenados generalicen bien a los datos del

mundo real. En especial en el sector industrial, donde los símbolos de un plano ingenieril pueden estar rotados, tener diferentes tamaños, estar acompañados de letras, números, etc, captar la variabilidad real en los datos sintéticos es vital para lograr resultados prometedores.

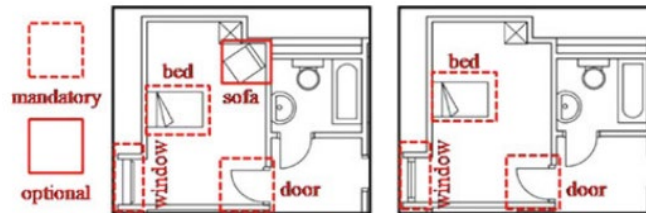


Figura 17: Restricciones en planos arquitectónicos. La puerta debe hallarse en la conexión entre dos habitaciones, la ventana debe dar al exterior y la cama debe estar alineada con la pared [40].

2.4 FUTURAS DIRECCIONES EN ESTE ÁMBITO

En conclusión, la aplicación del DL a la detección de símbolos en planos de ingeniería es un campo en rápida evolución con un inmenso potencial. La comparación entre modelos de entrenamiento que utilizan imágenes reales y sintéticas emerge como un área fundamental de investigación, abordando retos clave como la escasez de datos y la generalización de modelos. La investigación futura debería centrarse en perfeccionar aún más estas tecnologías, garantizando su aplicabilidad en escenarios variados y complejos típicos de los planos de ingeniería. Además, alinear estos avances con las necesidades cambiantes de la fabricación inteligente y la Industria 4.0 será crucial para su integración con éxito en las aplicaciones industriales.

Capítulo 3. DESARROLLO DEL DETECTOR DE TOLERANCIAS EN PLANOS INGENIERILES

3.1 ANÁLISIS Y ESTUDIO DEL PROBLEMA

A la hora de desarrollar una tecnología destinada al reconocimiento de tolerancias en un plano, se presenta un gran problema: la falta de datos.

El número de planos disponibles para el desarrollo de una red neuronal de este calibre es limitado. Pocas empresas están dispuestas a ceder sus planos para esta labor y, además, no todos los planos disponibles presentan las tolerancias de interés. Así, el entrenamiento de una red destinada al análisis de planos mediante el uso exclusivo de datos reales es prácticamente inviable.

Ante este problema, en este proyecto se plantea una solución alternativa. Ante la falta de datos reales necesarios para entrenar una red neuronal de *Computer Vision*, se plantea la posibilidad de entrenar esta tecnología con datos generados de manera artificial. Mediante un generador eficiente y completo, capaz de captar la variabilidad que presentan los planos industriales reales, que introduzca tolerancias de manera artificial en planos reales vacíos, la solución al problema que se aborda pasa por analizar la viabilidad de entrenar una red convolucional con planos que han sido elaborados de manera sintética, planos que no existen en la realidad.

Este proyecto busca alinearse con el estado del arte existente y trata de buscar una mejora en el entrenamiento de un modelo de *Deep Learning* mediante el uso de datos sintéticos. En concreto, busca comparar tres modelos diferentes, con la misma arquitectura, pero entrenada con set de datos distintos:

1. Modelo 1: red entrenada con datos reales
2. Modelo 2: red entrenada con datos sintéticos

3. Modelo 3: red entrenada con un set de datos híbrido, conformado por tanto datos reales como datos híbridos.

Mediante el entrenamiento y la evaluación de estos tres modelos, se buscará responder a tres preguntas fundamentales:

1. ¿Los datos sintéticos pueden entrenar por sí solos una red neuronal de *Computer Vision*? De no ser así, ¿pueden combinarse con planos reales para mejorar los resultados obtenidos?
2. ¿Qué aspectos se pueden mejorar para lograr el generador sintético más efectivo posible?
3. ¿Hasta qué punto el generador de datos sintéticos puede captar la variabilidad del mundo real?

3.2 SELECCIÓN DE LAS CLASES

En el sector industrial, se hace uso de una serie de tolerancias, denominadas tolerancias geométricas (*General Dimensioning and tolerancing, GD&T* en inglés), que ayudan a marcar el nivel de defectos aceptable que puede presentar la pieza que se fabrica.

Estas tolerancias constan de una serie de símbolos, pudiendo usar referencias a un cierto *datum* (ver Figura 18). Existe una gran variedad de símbolos que pueden ser empleados como tolerancia geométrica, como se puede observar en la Figura 19.

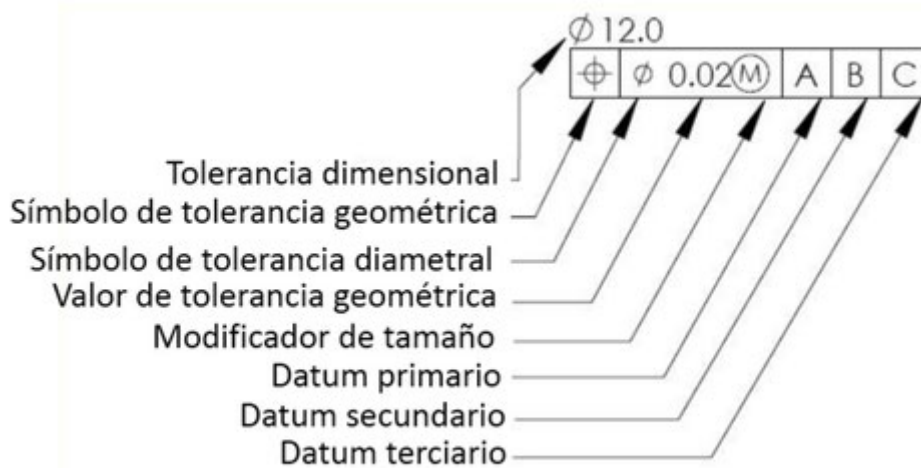


Figura 18: Ejemplo de tolerancia geométrica y explicación de sus posibles componentes [41]

Símbolos y características de la tolerancia

Elementos y tipo de tolerancia		Características geométricas	Símbolo
Elementos aislados	Forma	Rectitud	
		Planicidad	
		Redondez	
		Cilindricidad	
Elementos aislados o asociados		Perfil de una línea	
		Perfil de una superficie	
Elementos asociados	Orientación	Perpendicularidad	
		Angularidad	
		Paralelismo	
	Localización	Simetría	
		Posición	
		Coaxialidad y Concentricidad	
	Cabeceo	Oscilación circular	
		Oscilación total	

Figura 19: Ejemplos de símbolos empleados en tolerancias geométricas [41]

De cara al desarrollo de este proyecto, se eligieron seis símbolos en concreto, los cuales debían ser detectados por la red; el resto serían tolerancias de ruido, que la red debía ignorar. Las seis tolerancias que debía detectar el modelo son:

1. Planicidad: establece una tolerancia para la planicidad de una superficie, indicando dos planos que sirven como límite superior e inferior de la superficie afectada respecto del plano original.



Figura 20: Símbolo de planicidad

2. Paralelismo: al igual que la tolerancia de planicidad, la tolerancia de paralelismo impone, a partir de la recta que sirve como referencia, dos límites que la recta afectada no puede cruzar, estableciendo un área en la cual dicha recta debe quedar encerrada.



Figura 21: Símbolo de paralelismo

3. Perpendicularidad: como en el caso de las dos tolerancias anteriores, esta tolerancia determina una zona de tolerancia, permitiendo que una superficie que debe ser perpendicular a otra pueda tener cierto margen, pudiendo tener ligeras desviaciones respecto al ángulo de 90 grados.



Figura 22: Símbolo de perpendicularidad

4. Posición: esta tolerancia es usada con elementos de una pieza que tienen una localización muy específica respecto a otro elemento que sirve de referencia. Esta

tolerancia establece una zona alrededor del *datum*, en la cual el elemento se puede posicionar sin que se considere defectuoso.



Figura 23: Símbolo de posición

5. Concentricidad: esta tolerancia establece una zona para la localización del eje de un elemento que debe ser concéntrico a otro.



Figura 24: Símbolo de concentricidad

6. Circular: esta tolerancia se emplea en piezas de revolución. Marca una zona de tolerancia de la pieza; dado que la superficie de revolución puede tener imperfecciones, se marca un límite inferior y superior para la distancia de dicha superficie al eje de revolución que se marca como referencia.



Figura 25: Símbolo de excentricidad circular

3.3 IMPLEMENTACIÓN DEL DETECTOR DE TOLERANCIAS EN PLANOS DE INGENIERÍA

La tecnología a emplear en este proyecto es, como se ha mencionado en apartados anteriores, una red neuronal destinada a la detección de objetos. En lugar de diseñarla de cero, se decidió partir de una arquitectura preexistente, a la cual se le realizaría un proceso de *fine-tuning* mediante su entrenamiento. La diferencia entre los diferentes modelos sería, por tanto, únicamente los datos empleados para su entrenamiento.

3.3.1 SELECCIÓN DE LA RED A EMPLEAR: DARKNET

La red neuronal que se empleó fue DarkNet, una red neuronal *OpenSource* que emplea el algoritmo YOLO en el proceso de detectar objetos [17].

Esta arquitectura, que emplea la versión 4 de YOLO, permite entrenar la red y analizar su efectividad fácilmente. El manejo de la red, tanto en la fase de entrenamiento como en la fase de test, se llevaría a cabo mediante Google Colab, entorno de programación desarrollado por Google que permite la conexión a una GPU de manera sencilla. Para ello, los archivos y los documentos necesarios para el manejo de DarkNet serían descargados desde un repositorio de GitHub directamente a Google Drive, lo que permitiría una conexión sencilla entre los ficheros ejecutados en Google Colab y la red.

Esta red neuronal cuenta con 161 capas convolucionales y emplea la función de pérdida CIOU durante el entrenamiento. La ecuación de dicha función de pérdida aparece en la Figura 26 y es explicada y definida por su creador en la publicación que describe su fórmula [42]. La ecuación parte del término $1 - IoU$, que proporciona una idea del buen o mal solape que se da entre la caja detectada y la caja real que contiene al objeto. A menor IoU (y, por lo tanto, menos solape), mayor es la pérdida. El término $\frac{\rho^2(b, b^{gt})}{c^2}$ penaliza la distancia entre los centros de la caja predicha y la caja real; a mayor distancia entre centros, mayor penalización. Por último, el término αv penaliza las diferencias en la relación de aspecto o la relación entre el ancho y largo de las cajas.

$$\mathcal{L}_{CIOU} = 1 - IoU + \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2} + \alpha v.$$

Figura 26: Función de pérdida CIOU [42]

La red adapta los pesos para minimizar la pérdida en la detección de objetos del set de datos de entrenamiento.

DarkNet utiliza imágenes de 512x512 píxeles; además, al emplear YOLO, las imágenes debían estar acompañadas de un archivo txt que contuviese la información de las tolerancias presentes

en una determinada imagen, incluyendo el tipo de tolerancia y la posición de esta dentro de la imagen. Estos requisitos se tuvieron en cuenta más adelante, tanto en el desarrollo del generador sintético como en la preparación de los datos a emplear.

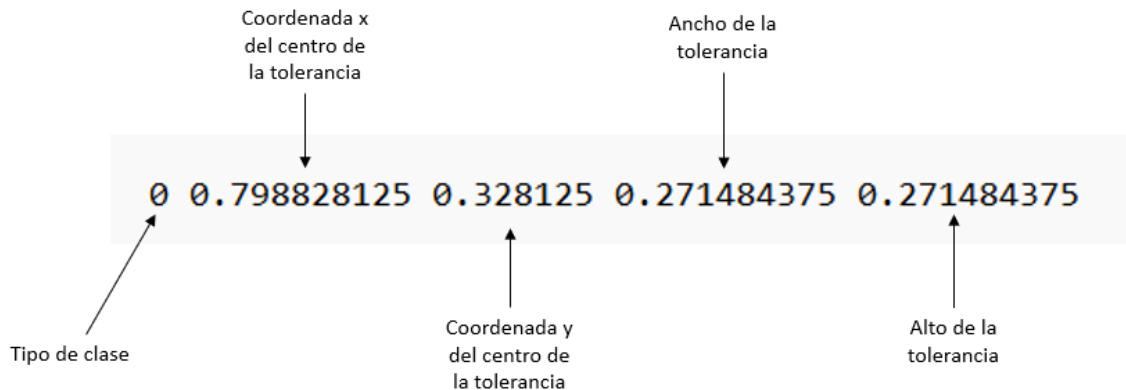


Figura 27: Formato YOLO del txt que debe acompañar a la imagen (coordenadas, ancho y largo relativos a las dimensiones del plano)

3.3.1.1 mAP y métricas de evaluación

DarkNet ofrece al usuario una serie de métricas y herramientas que permiten analizar el rendimiento y efectividad del modelo, ya sea con un set de datos de validación o con un set de datos de test.

Todas las métricas se basan en el concepto del IoU (*Intersection over Union*). El IoU se calcula a partir del área de la detección realizada por la red y el área real que contiene al objeto que se halla en la imagen. Así, el IoU es el cociente entre la intersección entre ambas áreas y la suma de dichas áreas (ver Figura 28).


$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


Figura 28: Concepto de IoU (Intersection over Union) [43]

A la hora de evaluar un modelo de *Computer Vision*, se determina un valor umbral de IoU, que será el que permita clasificar cada una de las detecciones (o no detecciones) como verdadero positivo (TP), falso positivo (FP), verdadero negativo (TN) o falso negativo (FN).

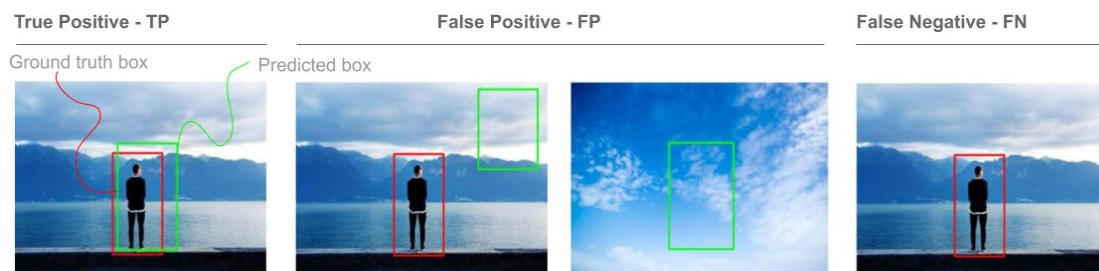


Figura 29: Ejemplos de TP, FP y FN en *Computer Vision* [44]

Para todas las métricas que calcula DarkNet, el valor umbral del IoU es de 0.5.

También se hará uso de los umbrales de confianza. Cuando el modelo hace una detección, lo hace con cierto grado de confianza, asociándole a cada detección un valor de 0 a 1 según la probabilidad que el modelo otorga a que la detección sea correcta. Variando el grado de

confianza mínimo que debe tener una detección para ser considerada válida, se pueden obtener diferentes valores de las métricas mostradas a continuación.

3.3.1.1.1 Precision y Recall

Precision

DarkNet permite calcular la precisión que la red (con los pesos correspondientes) ofrece para un set de datos de validación o de test.

Esta métrica busca determinar qué porcentaje de los casos que el modelo considera positivos son realmente positivos. Se trata del cociente entre el número de verdaderos positivos y la suma del número de verdaderos positivos y falsos positivos.

$$Precision = \frac{TP}{TP + FP}$$

DarkNet ofrece esta métrica para cada una de las clases estudiadas (en nuestro caso, las 6 tolerancias de interés).

Recall (o recuperación)

DarkNet permite calcular el *recall* (o recuperación) que la red (con los pesos correspondientes) ofrece para un set de datos de validación o de test.

Esta métrica busca determinar qué porcentaje de los positivos que debían ser reconocidos por la red han sido realmente identificados. Se trata del cociente entre el número de verdaderos positivos y la suma del número de verdaderos positivos y falsos negativos.

$$Recall = \frac{TP}{TP + FN}$$

DarkNet ofrece esta métrica para cada una de las clases estudiadas (en nuestro caso, las 6 tolerancias de interés).

Elección del umbral de confianza

Las métricas de *Precision* y *Recall* son de las más empleadas en el mundo del *Machine Learning*. Como no podía ser de otra manera, también son relevantes en la evaluación de los modelos obtenidos en este caso. Por defecto, DarkNet ofrece los valores de *Precision* y *Recall* para un umbral de IoU de 0.5 y un umbral de confianza de 0.25. Si bien se planteó incrementar el umbral de confianza, se optó por dejarlo como estaba. Normalmente, un mayor umbral de confianza se traduce en un mejor nivel de *Precision* y un peor nivel de *Recall*, ocurriendo lo contrario cuando el umbral de confianza disminuye. En este caso, dado que se busca desarrollar una herramienta que ayude al operario a detectar todas las tolerancias de un plano, es preferible un alto nivel de *Recall* a un alto nivel de *Precision*. Es decir, se da prioridad a que la red neuronal detecte todas las tolerancias del plano, a pesar de que se corra el riesgo de que realice detecciones incorrectas en forma de falsos positivos. Se consideró que es más fácil para el operario descartar las detecciones erróneas que buscar tolerancias que el modelo no haya identificado.

3.3.1.1.2 mAP (mean Average Precision)

DarkNet también permite calcular el $mAP@0.50$ (o mean Average Precision para un IoU de 0.50). El mAP es una métrica que permite comparar diferentes modelos de *Computer Vision* ante un set de datos de validación o de test. El cálculo del mAP se realiza de la siguiente manera:

1. Se obtiene la curva *Precision-Recall* hallada para cada una de las clases a partir del set de validación/test (ver Figura 30). Esta se obtiene para diferentes umbrales de confianza, pero manteniendo el valor umbral del IoU en 0.50.

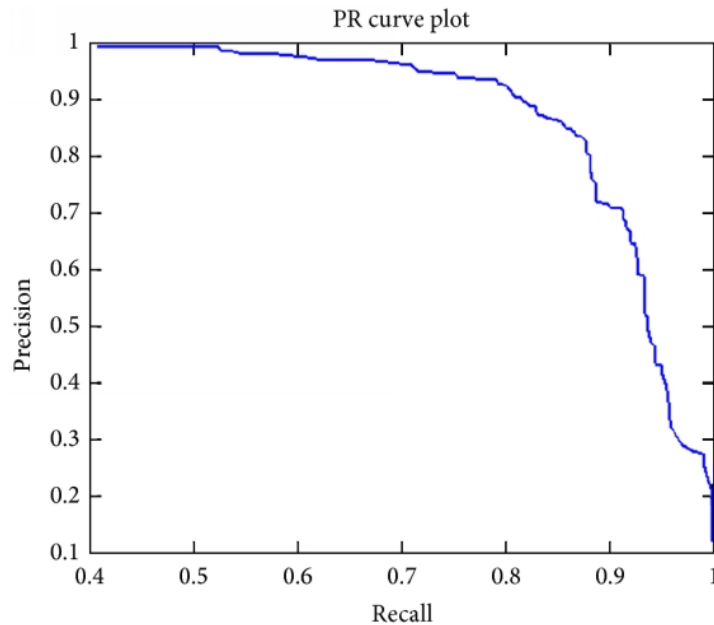


Figura 30: Ejemplo de una curva Precision-Recall [45]

2. Se calcula el *Average Precision* (AP) para cada una de las clases. El *Average Precision* se define como el área bajo la curva *Precision-Recall*.
3. Una vez obtenida la *Average Precision* para cada una de las clases, se realiza la media de todas estas para obtener el mAP.

Dado que el número obtenido depende de las curvas *Precision-Recall* de todas las clases, se trata de una métrica muy utilizada para comparar modelos.

3.4 DESARROLLO DEL GENERADOR SINTÉTICO

La creación de imágenes sintéticas se llevaría a cabo por medio de un generador sintético de planos. Este generador se programó en Python, por medio de un editor de código: Visual Studio Code.

Se partió de una versión prematura del generador, previamente diseñada por otros alumnos en un proyecto anterior. A partir de esta versión, se entendería su funcionamiento y se estudiarían

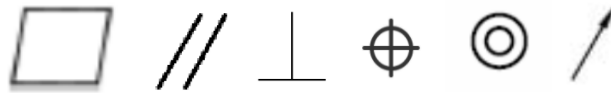


Figura 32: Ejemplos de imágenes de las tolerancias de interés en la carpeta
Tol_Recortadas

- Tol_Recortadas_ruido: esta carpeta recoge ejemplos de tolerancias que la red no debe reconocer. Son tolerancias que sirven de ruido y que buscan “confundir” a la red para mejorar la detección de las seis clases de interés.



Figura 33: Ejemplos de imágenes de símbolos de ruido de la carpeta
Tol_Recortadas_ruido

- Output_Planes: en esta carpeta se guardan las imágenes sintéticas generadas por el fichero.

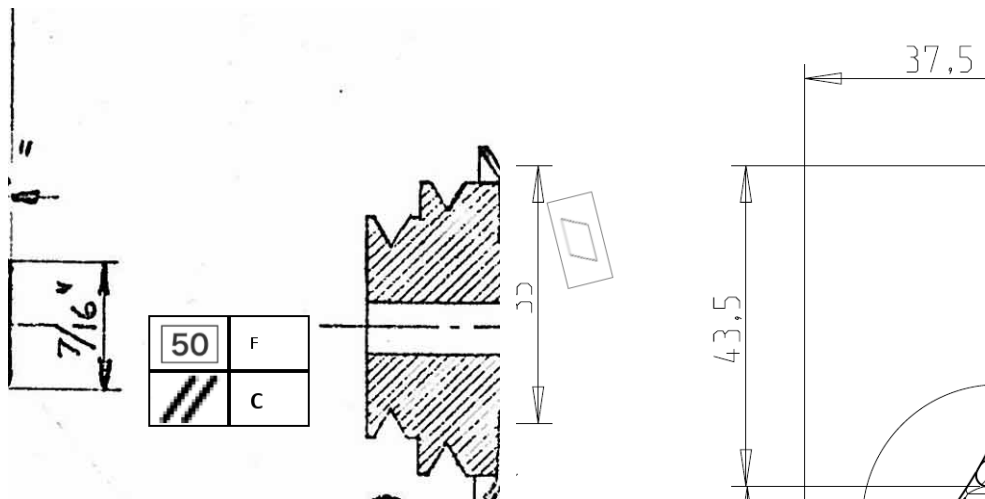


Figura 34: Ejemplos de imágenes generadas artificialmente en la carpeta
Output_Planes

- Labels: guarda los ficheros txt de los planos generados. Asocia a cada clase un número del 1 al 6 y las coordenadas en absoluto.

```
1,51.0,147.5,41.6,27.3  
2,267,230,147,134
```

Figura 35: Fichero txt generado junto a la imagen sintética de la carpeta Labels

- Corrected_labels: en esta carpeta, se guardan los txt corregidos, corrigiendo el número asociado a cada clase (en vez de estar asociadas a un número del 1 al 6, pasan a estar asociadas a un número del 0 al 5, restando 1) y las coordenadas, pasándolas a coordenadas relativas dividiendo por las dimensiones totales de la imagen. Esta corrección es necesaria para la lectura correcta de los txt por la red.

```
0 0.099609375 0.2880859375 0.08125 0.0533203125  
1 0.521484375 0.44921875 0.287109375 0.26171875
```

Figura 36: Ejemplo de un fichero txt ya corregido de la carpeta Corrected_labels

Una vez descritas las carpetas empleadas por el generador durante su ejecución, se procede a describir su funcionamiento.

En primer lugar, el fichero escoge al azar un plano de la carpeta Planos_Fuente. De dicho plano obtiene un recorte de 512x512 (dimensiones aceptadas por la red).

En segundo lugar, el fichero decide aleatoriamente si el plano contendrá cero, una o dos tolerancias (asignando diferentes probabilidades a cada uno de los casos). De tener que insertar tolerancias, tanto si son de interés como si son de ruido, el fichero seleccionará una imagen de un símbolo al azar de la carpeta correspondiente y tomará una serie de decisiones, también de manera aleatoria, relativas a:

1. Tamaño: el generador creará imágenes con tolerancias de diferentes tamaños para lograr generar datos más completos.
2. Rotación: el generador podrá rotar las tolerancias para contar con ejemplos en los que las tolerancias no estén dispuestas en horizontal.

- Inclusión de letras/texto: las tolerancias pueden ser generadas con un número, una letra, o ambos, acompañándolas, buscando que las tolerancias se asemejen lo máximo posible a las tolerancias halladas en planos reales (ver Figura 37).

Por último, el fichero, por medio de una función que busca posiciones libres dentro del recorte (que además estén próximas a píxeles negros para tratar de hacer que las tolerancias se hallen cerca de las figuras que se encuentren en la imagen, como ocurre con las tolerancias en planos industriales reales), el fichero de Python insertará la tolerancia en el recorte hecho al plano fuente seleccionado, obteniendo así la imagen sintética final, que guardará en la carpeta Output_Planes. Además, guardará el correspondiente txt asociado a la imagen en la carpeta Labels en el caso de que la imagen contenga tolerancias de interés. Una vez se hayan generado todas las imágenes, se usará otro fichero para corregir los txt; los txt corregidos se guardarán en la carpeta Corrected_labels.

Los datos sintéticos ya estarán listos para ser usados en el entrenamiento de la red.

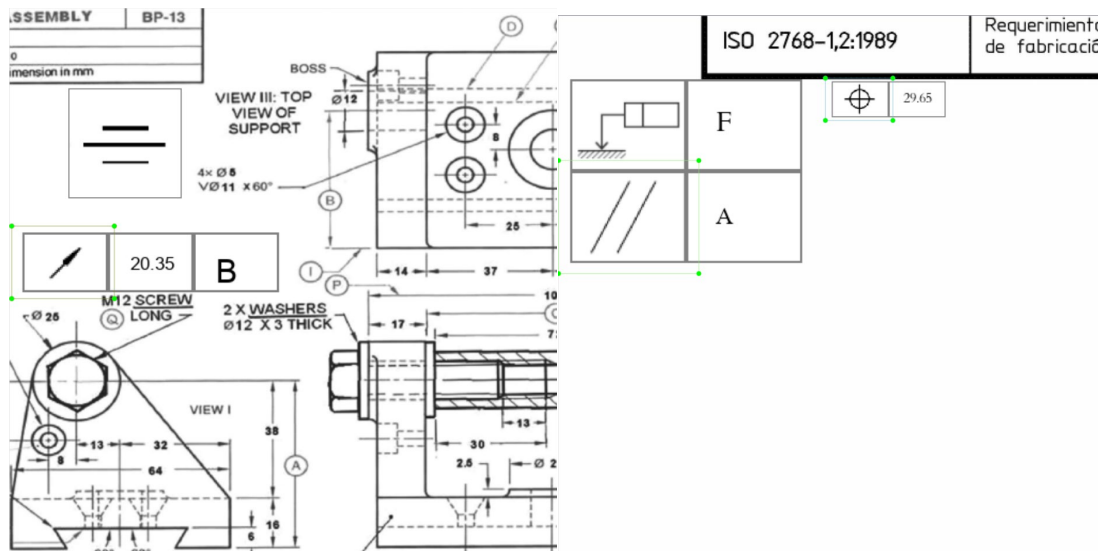


Figura 37: Imágenes sintéticas con las tolerancias recuadradas a partir del txt generado

3.4.2 MEJORAS

La versión que se proporcionó del generador era una versión funcional, pero que requirió de una serie de modificaciones para mejorar la variabilidad que era capaz de captar, buscando maximizar la completitud del conjunto de imágenes generadas.

3.4.2.1 Ampliación de los inputs

Si bien el generador era funcional y generaba imágenes de manera correcta, el número de datos que empleaba, tanto del conjunto de planos fuente como del conjunto de imágenes de las tolerancias, era muy reducido. Inicialmente, el generador solo contaba con 183 planos fuente y 4 ejemplos de cada tolerancia.

Esa cantidad de *inputs* no era suficiente para generar un conjunto de datos lo suficientemente variable para entrenar adecuadamente la red, pues podía dar lugar a problemas de generalización y, por ende, a una mala detección con planos reales.

Así, se decidió ampliar el número de datos de partida, pasando a tener 566 planos fuente y 17 ejemplos de cada tipo de tolerancia de interés.

También se decidió ampliar el set de tolerancias de ruido, añadiendo más elementos como números o variaciones de tolerancias que el modelo no debía detectar.

3.4.2.2 Ampliación del ángulo de rotación de las tolerancias

Como ya se comentó más arriba, el generador decide aleatoriamente si una tolerancia es rotada o no.

En la versión inicial del generador, la rotación de la tolerancia estaba limitada entre -45 y 45 grados. No obstante, se observó que en muchas ocasiones las tolerancias aparecen dispuestas en vertical.

Para conseguir un conjunto de datos con mayor variabilidad, se decidió ampliar el rango del ángulo de rotación de las tolerancias a -90 y 90 grados, pudiendo generar imágenes con

tolerancias dispuestas en un mayor arco de ángulos, logrando captar un mayor número de situaciones que se podrían dar en la vida real.

3.4.2.3 Incremento del rango de tamaño

Para regular el tamaño de la tolerancia, el generador elige aleatoriamente un *target ratio*, el cual establece la relación que debe tener el área de la tolerancia y el área de la imagen (todos positivos y menores a 1). Inicialmente, el rango establecido para este *target ratio* estaba entre 0.01 y 0.05. No obstante, a partir de un análisis previo de planos reales, en concreto de recortes de 512x512 de estos, se concluyó que este rango se podía incrementar, estableciéndolo entre 0.01 y 0.1.

De nuevo esta medida buscaba incrementar la variabilidad en el conjunto de imágenes generado.

3.4.2.4 Inclusión de agrupaciones de tolerancias

En un análisis inicial de las tolerancias presentes en planos industriales reales, se observó que, en múltiples ocasiones, las tolerancias aparecen en agrupaciones de dos o incluso de tres.

Por ende, se modificó el fichero del generador para que también pudiese generar este tipo de agrupaciones. De esta manera, antes de insertar una tolerancia en la imagen, se elegiría de manera aleatoria si se haría en forma de una tolerancia individual, o de una agrupación de tolerancias, asignando diferentes probabilidades a cada uno de los casos.

Esta mejora se hizo pensando en un entrenamiento más completo de la red, queriendo asegurar una correcta detección de las tolerancias, aun cuando estas se hallan muy próximas entre ellas.

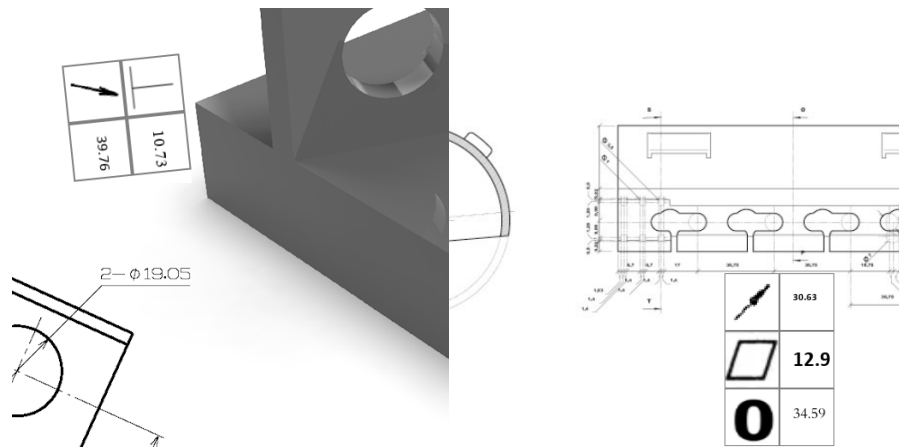


Figura 38: Ejemplos de imágenes sintéticas con agrupaciones de 2 y 3 tolerancias

3.4.3 GENERACIÓN DE IMÁGENES

Habiendo puesto el generador a punto, se comenzó la generación de imágenes sintéticas. Se generaron un total de 60000 imágenes, que serían utilizadas en el entrenamiento de los modelos para estudiar la viabilidad de entrenar una red neuronal con datos artificiales.

La generación de una imagen podía oscilar entre los 5 segundos y un minuto, tardando 15 segundos de media aproximadamente. Dado que la generación de 60000 imágenes podía ser un proceso arduo y lento, se optó por ejecutar el programa en 10 instancias en una máquina con mayor capacidad computacional, algo que rebajaba enormemente el tiempo de generación de las imágenes requeridas.

3.5 OBTENCIÓN DE LOS DATOS A EMPLEAR

Una vez se terminó de poner a punto el generador de planos sintéticos, se inició la tarea de preparar los sets de datos para el entrenamiento y el testeo de los diferentes modelos a analizar. Como ya se mencionó, todos los modelos usaron DarkNet, teniendo la misma estructura y número de capas; la única diferencia entre estos residía en el conjunto de datos empleados en su entrenamiento.

Se procederá a analizar cada uno de los conjuntos de datos empleados, tanto los usados en el entrenamiento como los empleados en el testeo.

3.5.1 SETS DE DATOS DE ENTRENAMIENTO

Como ya se mencionó en apartados anteriores, se entrenaron tres tipos de modelos:

1. Modelo entrenado con datos reales
2. Modelo entrenado con datos sintéticos
3. Modelo entrenado con un set de datos híbrido

De esta forma, se debieron preparar tres sets de datos distintos para el entrenamiento de cada uno de los modelos.

3.5.1.1 Set de datos reales

Para el primer set de datos, se dispuso de un conjunto de 376 planos reales. Dichos planos eran muy variados, tanto en las dimensiones como en el contenido de tolerancias de interés o la calidad del archivo.

Dado que la red únicamente admitía imágenes de 512x512, los planos debieron ser sometidos a un proceso de recorte, obteniendo imágenes más pequeñas de cada uno de los planos.

Para ello, se programó una función en Python que recortaba los planos en imágenes de la dimensión deseada y generaba los txt pertinentes.

No obstante, a la hora de realizar los recortes, se observó un problema: el proceso de dividir el plano en imágenes más pequeñas podía cortar una tolerancia a la mitad, lo que derivaba en que no se generaba una línea de txt para dicha tolerancia, perdiendo algunos de los ya escasos ejemplos para entrenar la red (ver Figura 39)

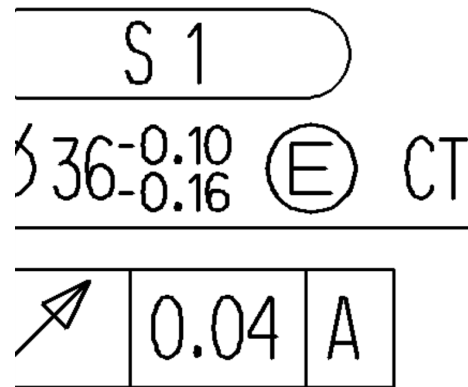


Figura 39: Ejemplo de tolerancia recortada en el proceso de preparación del set de datos reales

Para solucionar este problema, se tomaron dos medidas:

1. Incluir cierto solape: los planos fueron recortados con cierto solape; de esta manera, se buscaba minimizar la pérdida de ejemplos al recortar los planos, pues si una tolerancia quedaba cortada en un plano, podía recuperarse en el recorte contiguo.
2. Conservar las tolerancias ligeramente recortadas: el recorte de planos se programó de tal manera que, si una tolerancia quedaba recortada ligeramente (es decir, seguía siendo perfectamente reconocible dentro del recorte), este ejemplo se conservaba, generándose la línea para dicha tolerancia en el txt.

Una vez ajustados los parámetros tanto del solape como del margen que permitía conservar tolerancias ligeramente recortadas, se obtuvo un set de datos de 22697 recortes, que presentaba la distribución de ejemplos por cada clase que se aprecia en la Figura 40.

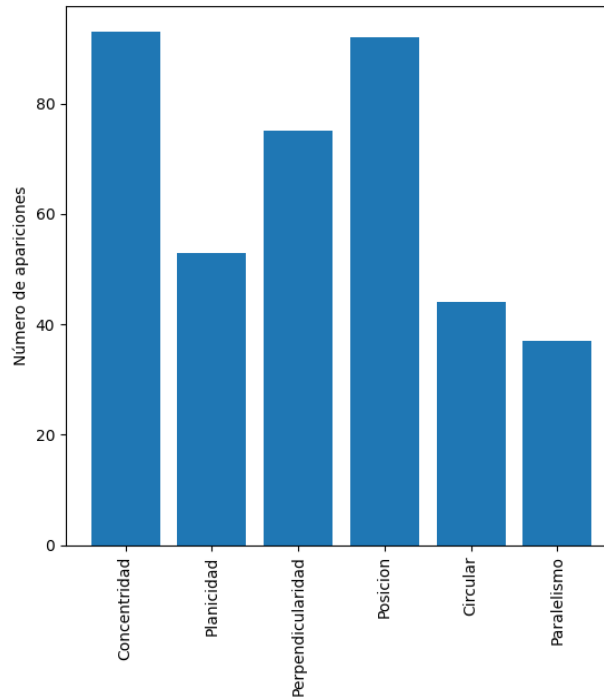


Figura 40: Distribución de ejemplos por clase en el set de datos de imágenes reales

Analizando la distribución, observamos que los ejemplos son muy escasos. Ninguna clase llega a los 100 ejemplos; en concreto, las clases Circular y Paralelismo ni siquiera sobrepasan los 50 ejemplos. Esta distribución permite entender la gran escasez de datos que se produce al tratar de usar un set de imágenes reales.

3.5.1.2 Set de datos sintéticos

El set de datos sintéticos fue creado a partir del generador de planos desarrollado en Python. Se generaron un total de 22697 recortes, con los archivos txt correspondientes. Se decidió crear un set de datos con el mismo número de recortes que el set de datos reales, con el objetivo de comparar los resultados obtenidos con ambos.

El set de datos generado presentaba la distribución de clases que se aprecia en la Figura 41.

Gracias al generador de imágenes, se contaba con una cantidad mucho mayor de ejemplos que la proporcionada por el set de datos real. Además, el desbalance de clases en cuanto al número

de ejemplos proporcionados de cada una desaparece, teniendo más de 1500 ejemplos para cada una.

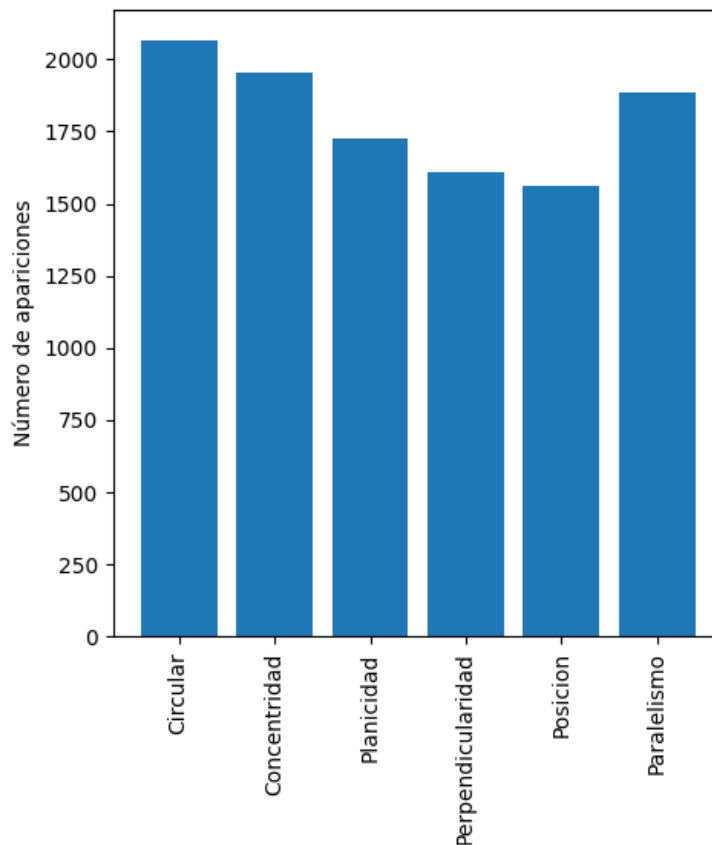


Figura 41: Distribución de ejemplos por clase en el set de datos de imágenes sintéticas

3.5.1.3 Set de datos híbrido

El set de datos híbrido se creó juntando los dos sets de datos anteriores. Así, se contaba con 22697 recortes reales y 22697 recortes sintéticos. La distribución de ejemplos por cada clase se presenta en la Figura 42.

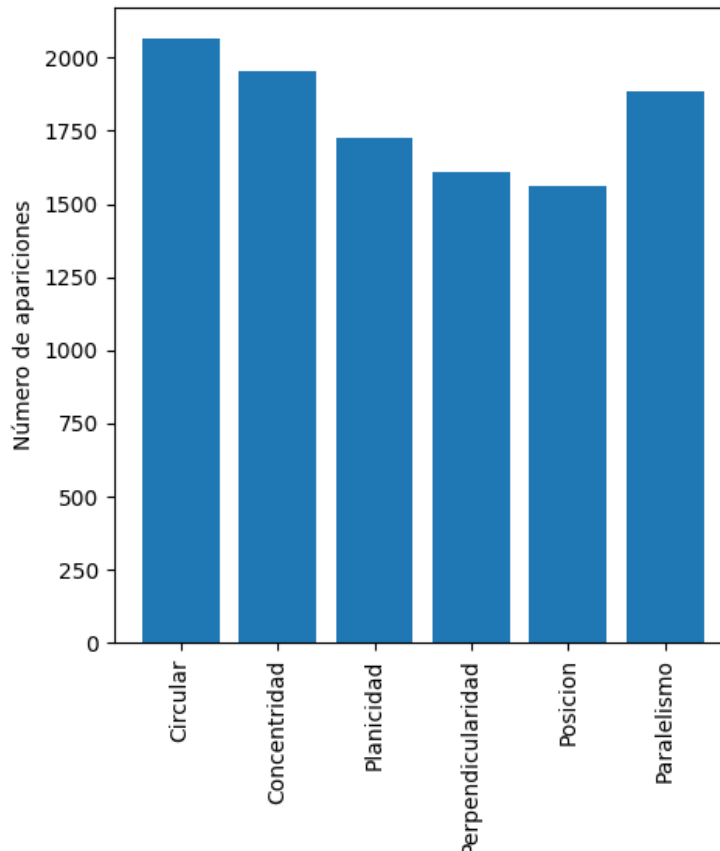


Figura 42: Distribución de ejemplos por clase en el set de datos híbrido

Dado que el número de ejemplos que aporta el set de datos reales es muy inferior al proporcionado por el set de datos sintético, la gráfica obtenida es muy parecida a la generada para el set de datos generado de manera artificial.

3.5.1.4 Set de datos de test

Con el fin de comparar los modelos obtenidos con los diferentes sets de datos, se contó con un set de datos específicamente destinado al test de estos.

Para este set de datos, se reservaron 228 planos reales. Dado que la red solo admite imágenes de 512x512, se recortaron los planos, obteniéndose recortes de esas dimensiones, empleando la misma herramienta que se empleó en el recorte de los planos que conformaban el set de datos reales que se iba a emplear para entrenar uno de los modelos. La distribución del número de ejemplos por clase se aprecia en la Figura 43Figura 42.

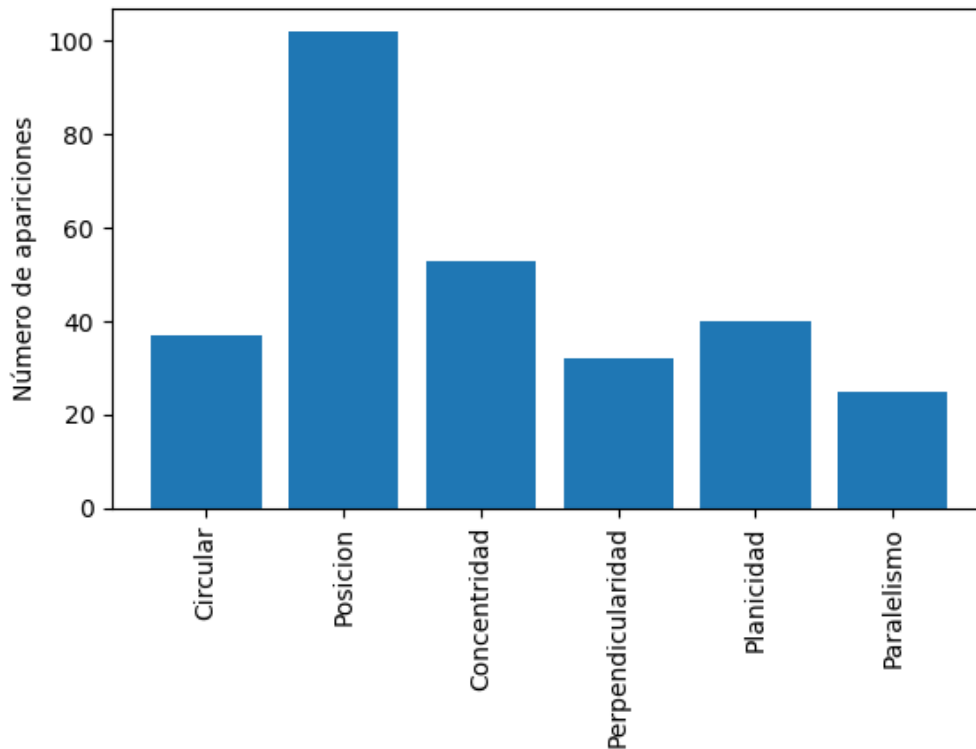


Figura 43: Distribución de ejemplos por clase en el set de datos de test

Este set de datos nos iba a permitir observar cómo de buena era la generalización del modelo, y nos iba a permitir determinar qué modelo arrojaba mejores resultados, pudiendo entender la mejora que los planos sintéticos podían ofrecer a la hora de entrenar un modelo que detectase tolerancias de manera precisa.

3.6 ENTRENAMIENTO DE LOS MODELOS

Como se ha comentado anteriormente, se partió de una red ya configurada, llamada DarkNet. La estructura no se modificó, y todos los modelos entrenados contaban con la misma arquitectura y las mismas capas convolucionales. La única diferencia que existía entre modelos residía en el set de datos empleado en su entrenamiento. Los tres modelos se definen de la siguiente manera:

- M01_RAW_Cropped: se trata del modelo entrenado con el set de datos conformado por los recortes de planos reales.
- M02_SYNT: se trata del modelo entrenado con el set de datos que recoge todos los recortes generados de manera sintética.
- M03_HYBRID: se trata del modelo entrenado con el set de datos híbrido, incluyendo el set de datos reales y el sintético.

El entrenamiento de cada uno de los modelos se realizó mediante un proceso de *fine-tuning*. Se partió de los pesos “yolov4.conv.137”; se trata de unos pesos pre-entrenados con el set de datos “MS COCO” (un set de datos que recoge una gran variedad de objetos como bicicletas, personas, coches, etc) hasta cierto punto (solo las primeras 137 capas convolucionales). A partir de estos pesos, y empleando los sets de datos descritos anteriormente, la red se entrenaría para detectar las tolerancias de interés, modificando los pesos de los que se partía.

El desarrollador de DarkNet, a nivel de sugerencia, recomendaba entrenar la red neuronal con un número de iteraciones equivalente a 2000 por el número de clases. Dado que se buscaba detectar 6 tolerancias de interés, se decidió que los modelos se entrenarían hasta las 12000 iteraciones y, más tarde, se valoraría ampliar el número de iteraciones en caso de que fuera necesario.

Durante el entrenamiento, se realizaba una validación del modelo cada cierto número de iteraciones (cabe mencionar que, de cada set de datos, el 85% de las imágenes estaban destinadas al entrenamiento y el 15% a la validación). Con las imágenes dirigidas a validar el entrenamiento del modelo, se calculaba el mAP, de manera que se podía supervisar cómo avanzaba el entrenamiento de la red.

Por último, cabe resaltar que la red guardaba ciertos pesos relevantes:

- Los pesos obtenidos cada 1000 iteraciones: yolo-obj_1000.weights, yolo-obj_2000.weights, etc. Esto permitía que, en caso de sobre-entrenamiento, se pudiera volver a un estado anterior del entrenamiento, antes de que se comenzara a dar el sobreajuste del modelo.

- Los pesos que mejor mAP daban en la validación: `yolo-obj_best.weights`.
- Los últimos pesos que se obtenían durante el entrenamiento y que se actualizaban cada 100 iteraciones: `yolo-obj_last.weights`. En caso de que se parara el entrenamiento de la red, estos pesos permitían reanudar el entrenamiento desde el punto donde se dejó de entrenar.

A continuación, se presentan las gráficas obtenidas durante el entrenamiento de cada uno de los tres modelos: el entrenado con datos reales, el entrenado con los datos sintéticos y el entrenado con el set de datos híbrido. Dichas curvas muestran, por un lado, la curva de pérdidas obtenida durante el entrenamiento, así como la evolución del mAP obtenido con el set de validación a lo largo de todo el proceso de *fine-tuning*.

3.6.1 MODELO I: DATOS REALES (M01_RAW_CROPPED)

Inicialmente, el modelo I se entrenó hasta las 12000 iteraciones, obteniendo la gráfica que se observa en la Figura 44. No obstante, se decidió prolongar su entrenamiento hasta las 48000 iteraciones dado que el mAP obtenido con el set de validación era todavía mejorable; la gráfica del entrenamiento de la red hasta las 48000 iteraciones se aprecia en la Figura 45.

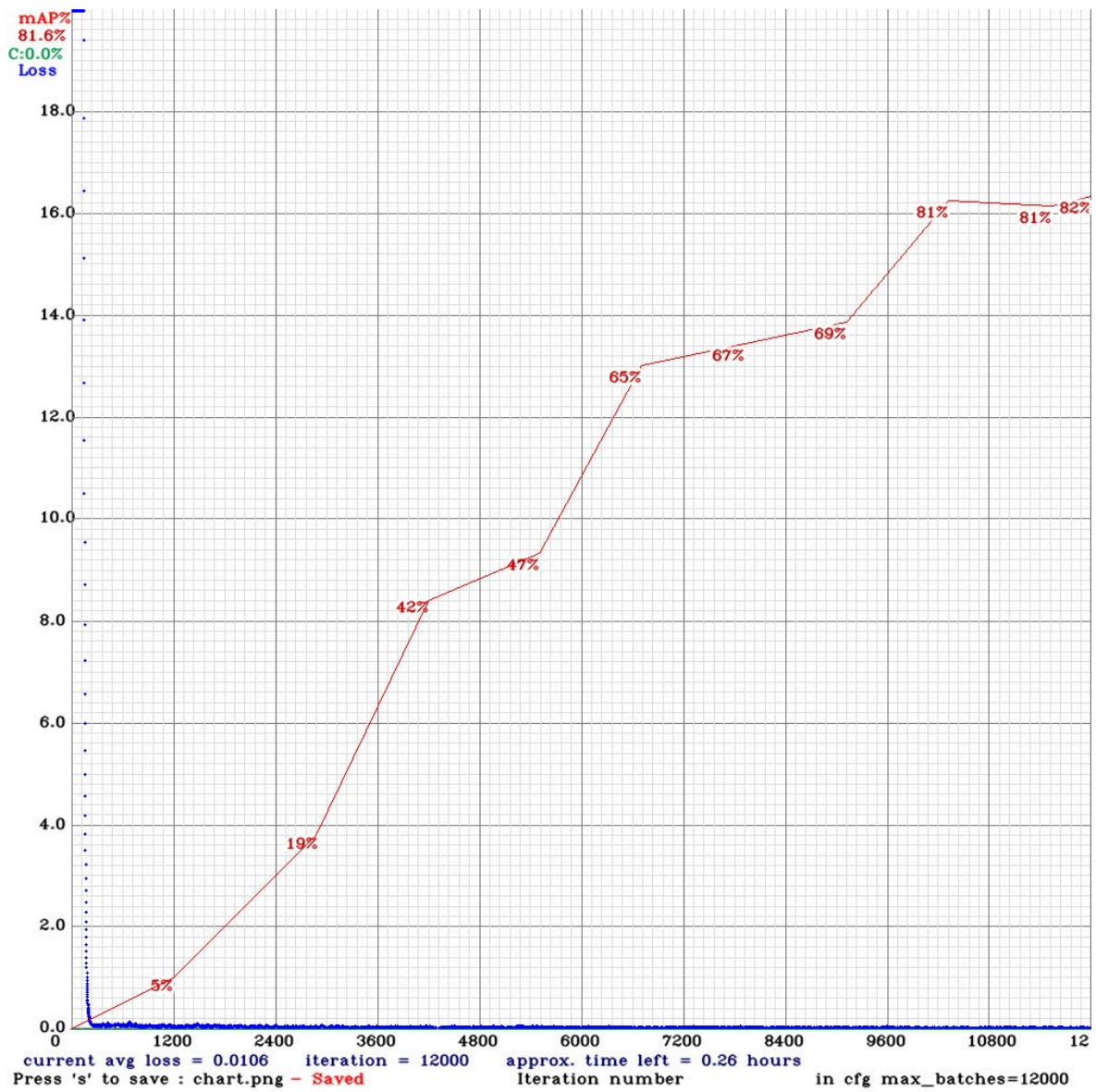


Figura 44: Gráfica del entrenamiento del Modelo I hasta las 12000 iteraciones

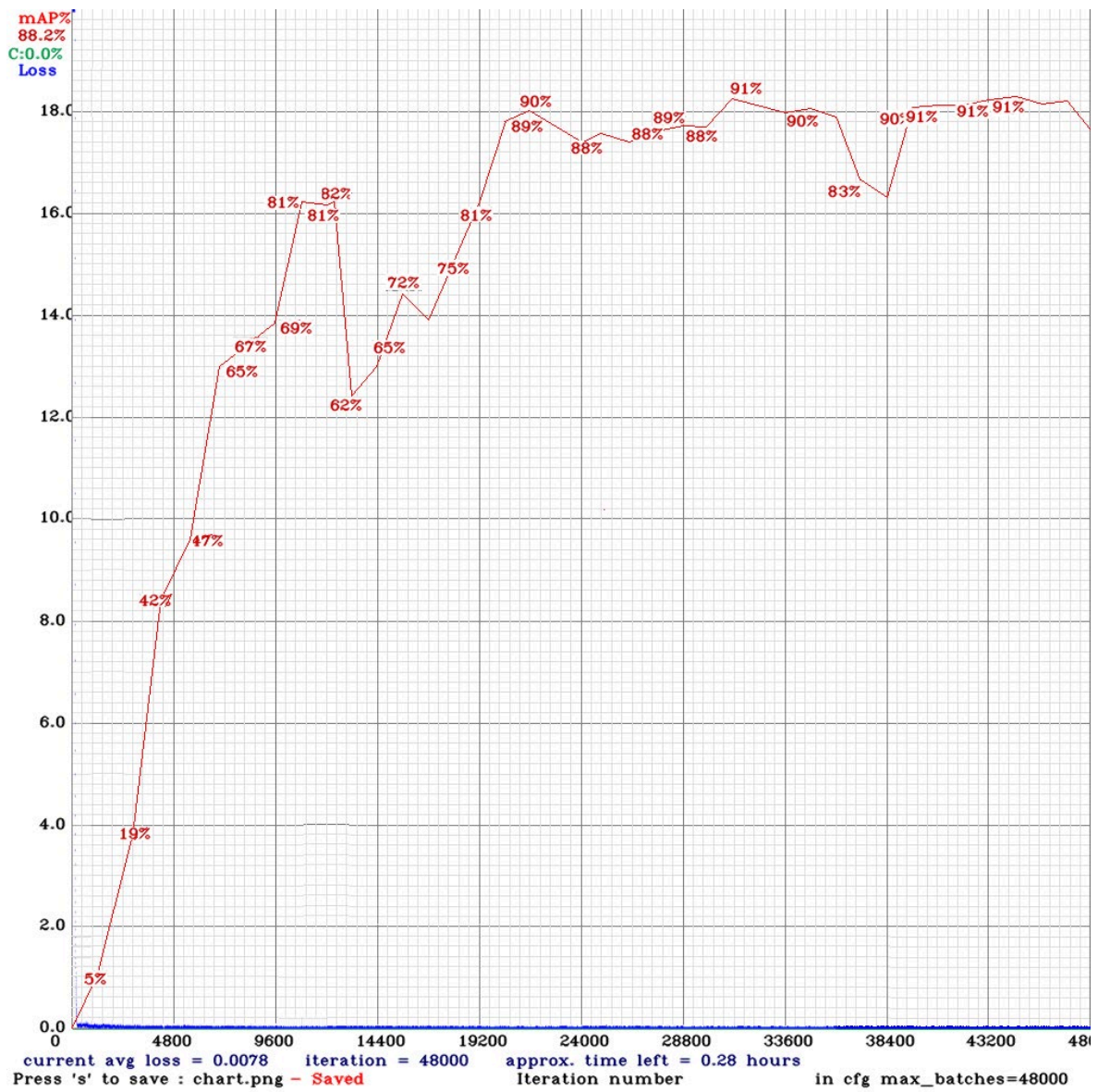


Figura 45: Gráfica del entrenamiento del Modelo I hasta las 48000 iteraciones

Con el objeto de diferenciar los resultados obtenidos con el modelo entrenado hasta las 12000 y las 48000 iteraciones, el modelo I tendrá dos variantes: RAW_Cropped12000 y RAW_Cropped48000.

3.6.2 MODELO II: DATOS SINTÉTICOS (M02_SYNTH)

Este modelo, entrenado y validado con datos sintéticos, alcanzó un mAP del 100% en la validación durante las 12000 iteraciones, por lo que no se prolongó su entrenamiento.

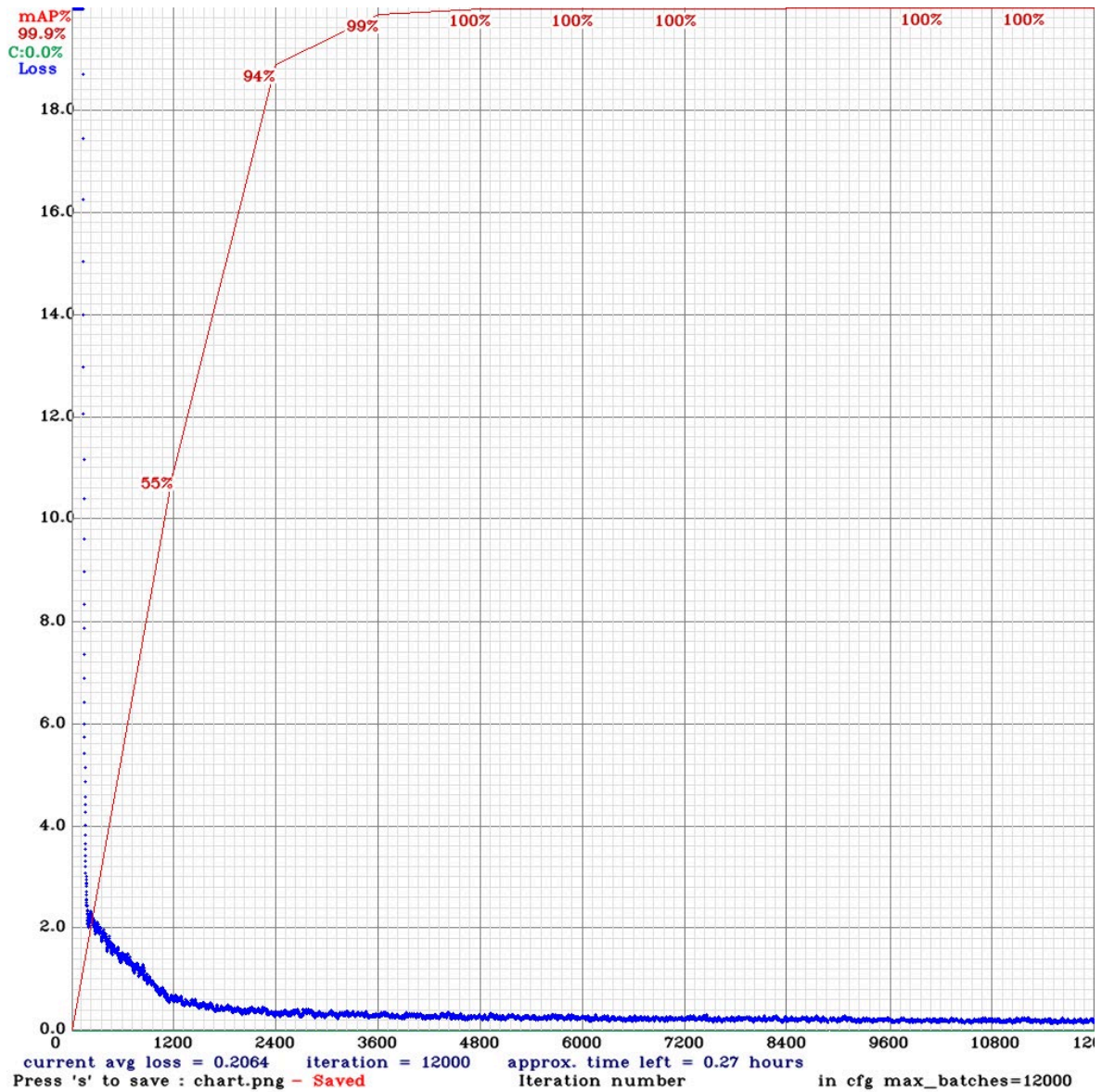


Figura 46: Gráfica de entrenamiento del Modelo II

3.6.3 MODELO III: DATOS HÍBRIDOS (M03_HYBRID)

Este modelo, entrenado y validado con datos híbridos, alcanzó un mAP del 100% en la validación durante las 12000 iteraciones, por lo que no se prolongó su entrenamiento.

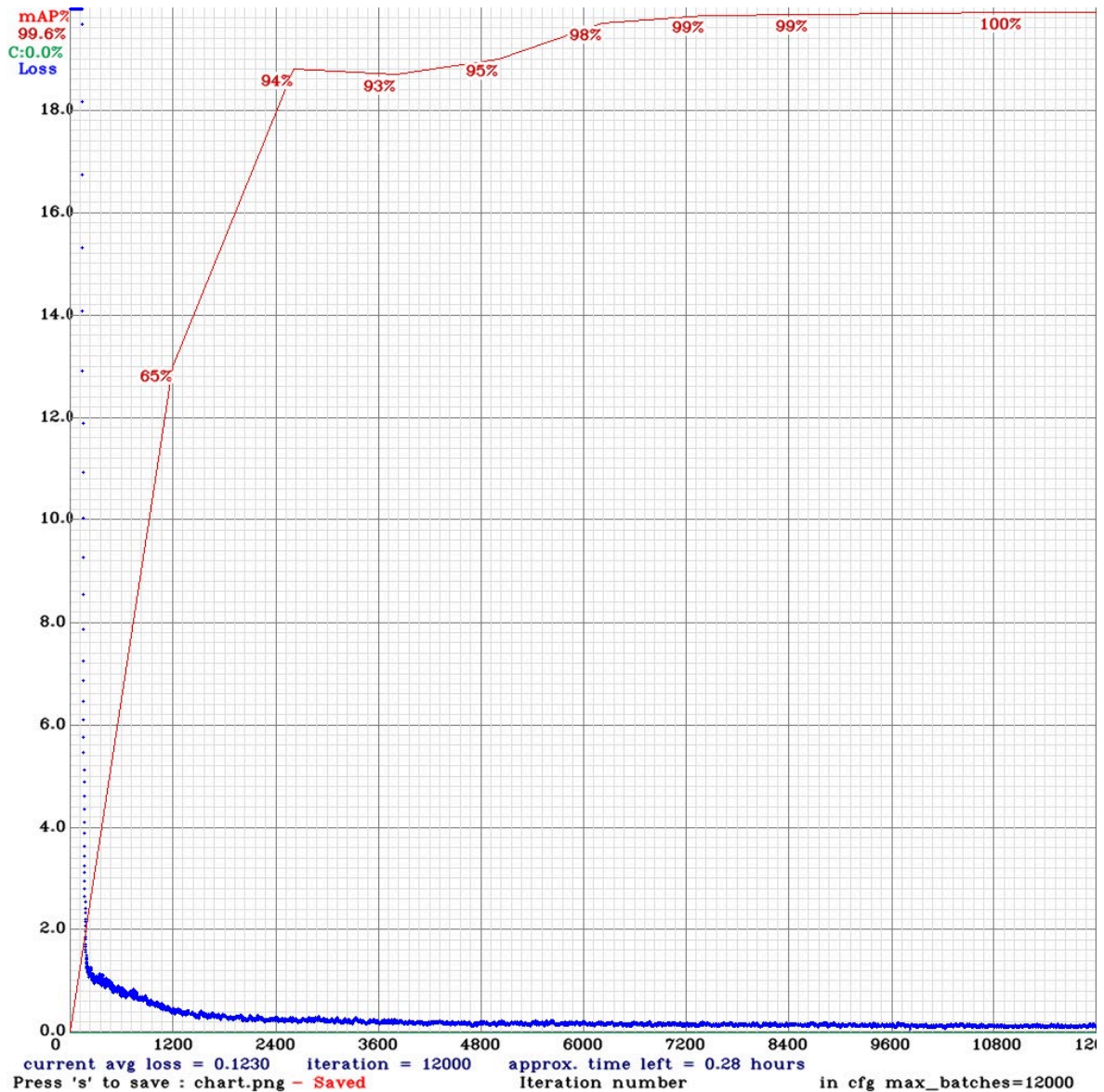


Figura 47: Gráfica de entrenamiento del Modelo III

3.7 ANÁLISIS DE RESULTADOS

Los resultados que se presentan a continuación han sido obtenidos a partir del set de datos de test, que se ha empleado en los tres modelos para poder compararlos de manera adecuada. Se ofrecen los resultados de cada modelo con los pesos que mejor comportamiento otorgaron a este según las métricas obtenidas en este apartado.

3.7.1 RESULTADOS POR VERSIÓN DEL GENERADOR

3.7.1.1 Resultados con datos reales

Clase	Planicidad			Perpendicularidad			Paralelismo			Concentricidad			Posición			Circular			mAP
Modelo	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	
RAW_Cropped12000	0.43	0.57	43.03%	0.31	0.53	43.60%	0.37	0.28	28.33%	0.50	0.79	67.24%	0.54	0.75	60.35%	0.42	0.49	37.83%	46.73%
RAW_Cropped48000	0.76	0.85	81.40%	0.44	0.72	68.97%	0.58	0.72	72.30%	0.75	0.91	89.76%	0.73	0.88	82.45%	0.58	0.59	48.22%	73.85%

Tabla 1: Resultados con datos reales

Antes de analizar los resultados obtenidos mediante el uso de datos artificiales, se procede a un análisis breve de los resultados obtenidos con los dos modelos entrenados con planos reales. Como ya se ha mencionado, se elaboraron dos modelos: uno entrenado con 12000 iteraciones y otro entrenado con 48000 iteraciones. Como se aprecia en la Tabla 1, el modelo entrenado con más iteraciones es el que mejor resultados arroja en todas las métricas. Esto se muestra lógico ya que ha sido entrenado con un mayor número de iteraciones. La mejora que ofrece un entrenamiento más largo es notable, incrementando el mAP general en casi un 30%.

3.7.1.2 Resultados iniciales

En las primeras etapas del proyecto, se creó un set de datos sintéticos a partir de la versión inicial del generador, esto es, sin apenas cambios con respecto a la versión prematura de la que se partía. Los resultados figuran en la Tabla 2. Se ha empleado un código de color para ordenar los diferentes modelos según su valor en cada una de las métricas estudiadas. De peor a mejor: Rojo, Naranja, Amarillo, Verde.

Clase	Planicidad			Perpendicularidad			Paralelismo			Concentricidad			Posición			Circular			mAP
	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	
RAW_Cropped12000	0.43	0.57	43.03%	0.31	0.53	43.60%	0.37	0.28	28.33%	0.50	0.79	67.24%	0.54	0.75	60.35%	0.42	0.49	37.83%	46.73%
RAW_Cropped48000	0.76	0.85	81.40%	0.44	0.72	68.97%	0.58	0.72	72.30%	0.75	0.91	89.76%	0.73	0.88	82.45%	0.58	0.59	48.22%	73.85%
SYNTH	0.87	0.85	89.83%	0.52	0.47	36.89%	0.36	0.64	53.75%	0.80	0.68	69.40%	0.81	0.34	35.37%	0.31	0.43	38.51%	53.96%
HYBRID	0.84	0.95	95.46%	0.76	0.78	81.23%	0.49	0.84	80.42%	0.78	0.94	94.04%	0.81	0.70	78.08%	0.63	0.73	76.88%	84.35%

Tabla 2: Resultados iniciales

Con este primer set de datos generados de manera artificial, se aprecia cierta evolución en los resultados obtenidos para cada clase:

- Planicidad: El modelo SYNTH presenta mejores resultados que ambos modelos RAW_Cropped, con una *Precision* de 87%, la más alta de los modelos estudiados, un *Recall* de 85% y un AP de 89.83%. El modelo HYBRID presenta una *Precision* algo inferior al modelo SYNTH (84%), pero es el modelo con mejor *Recall* (95%) y AP (95.46%) de los cuatro considerados.
- Perpendicularidad: El modelo SYNTH presenta mejor *Precision* que los dos modelos RAW_Cropped (52%), registrando el peor *Recall* (47%) y peor AP (36.89%) de los cuatro modelos. El modelo HYBRID es el que mejor resultados presenta en todas las métricas: *Precision* (76%), *Recall* (78%) y AP (81.23%).
- Paralelismo: El modelo SYNTH registra la peor *Precision* de los cuatro modelos (36%), y un *Recall* (64%) y un AP (53.75%) inferiores al de RAW_Cropped48000 y superiores al de RAW_Cropped12000. El modelo HYBRID presenta una *Precision* (49%) inferior

a la de RAW_Cropped48000 y superior a la de RAW_Cropped12000, y el mejor *Recall* (84%) y AP (80.42%) de los cuatro modelos.

- **Concentricidad:** El modelo SYNTH presenta la mejor *Precision* (80%), pero el peor *Recall* (68%) y un AP entre el obtenido por RAW_Cropped48000 y RAW_Cropped12000 (69.40%). El modelo HYBRID tiene la segunda *Precision* más alta (78%), cercano al SYNTH, y los mejores resultados para *Recall* (94%) y AP (94.04%).
- **Posición:** El modelo SYNTH presenta la mejor *Precision* (81%) pero los peores resultados de *Recall* (34%) y AP (35.37%). El modelo HYBRID presenta la mejor *Precision* (81%), igual a la del modelo SYNTH, el tercer mejor *Recall* (70%) y el segundo mejor AP (78.08%).
- **Circular:** El modelo SYNTH presenta los peores resultados de *Precision* (31%) y *Recall* (43%), y el segundo peor AP (38.51%). El modelo HYBRID es el que mejor resultados presenta para las tres métricas, *Precision* (63%), *Recall* (73%) y AP (76.88%).
- **mAP:** a partir de todos los valores de AP obtenidos, se calculó el mAP como la media de dichos valores. El modelo HYBRID presenta el mAP más alto (84.35%), seguido del RAW_Cropped48000 (73.85%). El SYNTH y el RAW_Cropped12000 presentan un mAP muy bajo en comparación a los otros dos (53.96% y 43.73% respectivamente).

Con esta primera versión del generador, el modelo HYBRID se presenta como la mejor alternativa, siendo el que mejor resultados presenta en el mAP general. El modelo entrenado con datos reales también presenta resultados aceptables, pero con un entrenamiento cuatro veces más largo. Los datos sintéticos de esta versión apenas superan los resultados obtenidos con el modelo de datos reales entrenado con 12000 iteraciones, teniendo amplio margen de mejora. Estos resultados muestran que la combinación de datos sintéticos y datos reales es la que mejor entrena al modelo.

3.7.1.3 Resultados tras las mejoras

A partir de los resultados iniciales, y mediante un análisis cualitativo de las detecciones hechas por cada modelo, se implementaron una serie de mejoras en el generador sintético, las cuales son descritas en el apartado de Mejoras dentro de la sección Desarrollo del generador sintético.

Con la nueva versión del generador sintético, se creó un nuevo set de datos artificiales, y se procedió a entrenar de nuevo cada uno de los tres modelos: RAW_Cropped, SYNTH y HYBRID. Los resultados se recogen en la Tabla 3.

Clase	Planicidad			Perpendicularidad			Paralelismo			Concentricidad			Posición			Circular			mAP
	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	
RAW_Cropped12000	0.43	0.57	43.03%	0.31	0.53	43.60%	0.37	0.28	28.33%	0.50	0.79	67.24%	0.54	0.75	60.35%	0.42	0.49	37.83%	46.73%
RAW_Cropped48000	0.76	0.85	81.40%	0.44	0.72	68.97%	0.58	0.72	72.30%	0.75	0.91	89.76%	0.73	0.88	82.45%	0.58	0.59	48.22%	73.85%
SYNTH	0.28	0.98	92.70%	0.36	0.75	68.83%	0.68	0.84	77.26%	0.84	0.91	91.31%	0.74	0.74	75.80%	0.61	0.95	84.95%	81.81%
HYBRID	0.74	0.93	91.28%	0.59	0.84	79.88%	0.81	0.88	84.59%	0.89	0.94	96.30%	0.77	0.82	81.98%	0.76	0.92	88.06%	87.02%

Tabla 3: Resultados tras las mejoras

Con este nuevo set de datos, los cuales incluyen las mejoras incluidas en el generador sintético, se obtuvieron los siguientes resultados para cada clase:

- Planicidad: El modelo SYNTH registra la peor *Precision* de los cuatro modelos (0.28), si bien presenta los mejores resultados a nivel de *Recall* (0.98) y AP (92.70%). El modelo HYBRID presenta la segunda mejor *Precision* (0.74), por detrás del modelo RAW_Cropped48000, el segundo mejor *Recall* (0.93), y el segundo mejor AP (91.28%), ambos por detrás del modelo SYNTH; en las tres métricas, el modelo HYBRID está muy próximo al que mejores valores presenta.
- Perpendicularidad: El modelo SYNTH se sitúa por encima del modelo RAW_Cropped12000 y por debajo del modelo RAW_Cropped48000 tanto en *Precision* (0.36) como AP (68.83%), y por encima de ambos en términos de *Recall* (0.75). Por otro lado, el modelo HYBRID es el que mejor resultados presenta en todas las categorías: *Precision* (0.59), *Recall* (0.84) y AP (79.88%).
- Paralelismo: El modelo SYNTH supera a los dos modelos RAW_Cropped en *Precision* (0.68), *Recall* (0.84) y AP (77.26%). De nuevo, el modelo HYBRID es el que mejor resultados arroja en las tres métricas estudiadas para esta clase: *Precision* (0.81), *Recall* (0.88) y AP (84.59%).

- **Concentricidad:** El modelo SYNTH supera a los dos modelos RAW_Cropped en *Precision* (0.84), *Recall* (0.91, igualando el *Recall* del modelo RAW_Cropped48000) y AP (91.31%). El modelo HYBRID vuelve a ser el que mejor valor ofrece en las tres métricas de esta clase: *Precision* (0.89), *Recall* (0.94) y AP (96.30%).
- **Posición:** El modelo SYNTH arroja mejor valor de *Precision* que los dos modelos RAW_Cropped (0.74), si bien registra el peor *Recall* de los cuatro modelos (0.74) y un AP superior al obtenido con el modelo RAW_Cropped12000 e inferior al que presenta el modelo RAW_Cropped48000 (75.80%). El modelo HYBRID muestra la mejor *Precision* (0.77), el segundo mejor *Recall* (0.82) y el segundo mejor AP (81.98%), únicamente por detrás del modelo RAW_Cropped48000.
- **Circular:** El modelo SYNTH mejora los resultados de los dos modelos RAW_Cropped en *Precision* (0.61), *Recall* (0.95), siendo el *Recall* más alto de los cuatro modelos, y AP (84.95%). El modelo HYBRID ofrece los valores más altos de *Precision* (0.76) y AP (88.06%), así como el segundo valor más alto de *Recall* (0.92), únicamente superado por el *Recall* del modelo SYNTH.
- **mAP:** a partir de todos los valores de AP obtenidos, se calculó el mAP como la media de dichos valores. El modelo HYBRID presenta el mAP más alto (87.02%), seguido del modelo SYNTH (81.81%), el modelo RAW_Cropped48000 (73.85%) y el modelo RAW_Cropped12000 (46.73%).

Tras la introducción de las mejoras en el generador de planos sintéticos, se observa una gran evolución de los resultados. El modelo SYNTH aumenta su mAP en casi un 30%, superando al modelo RAW_Cropped48000, algo que no ocurría con la primera versión del generador. Las mejoras realizadas al generador permiten obtener mejores resultados con los datos generados de manera artificial que con los datos reales, incluso con un entrenamiento más corto. De nuevo, la combinación de datos reales y sintéticos vuelve a ser el que mejor resultados ofrece; el modelo HYBRID presenta las mejores métricas respecto a los cuatro modelos, si bien la distancia con el modelo SYNTH se reduce considerablemente respecto a la versión inicial.

3.7.2 RESULTADOS POR TIPO DE MODELO

Con el objetivo de poder analizar la efectividad de los cambios introducidos en el generador a lo largo del proyecto respecto de la versión inicial de este de la que se partió, también se discuten y comparan los resultados de las distintas versiones de los modelos SYNTH y HYBRID.

3.7.2.1 Modelo sintético (SYNTH)

Se presentan a continuación los resultados que presenta el modelo SYNTH, es decir, el modelo entrenado a partir únicamente de datos sintéticos, con los dos sets de datos empleados a partir de las dos versiones del generador de planos sintéticos: la versión inicial y la que incluía las mejoras propuestas. Los resultados figuran en la Tabla 4.

Clase	Planicidad			Perpendicularidad			Paralelismo			Concentricidad			Posición			Circular			mAP
Modelo	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	
SYNTH1	0.87	0.85	89.83%	0.52	0.47	36.89%	0.36	0.64	53.75%	0.80	0.68	69.40%	0.81	0.34	35.37%	0.31	0.43	38.51%	53.96%
SYNTH2	0.28	0.98	92.70%	0.36	0.75	68.83%	0.68	0.84	77.26%	0.84	0.91	91.31%	0.74	0.74	75.80%	0.61	0.95	84.95%	81.81%

Tabla 4: Resultados de los modelos SYNTH

De la Tabla 4 se procede a realizar una comparación de ambas versiones del modelo SYNTH. A simple vista, se puede observar que el modelo SYNTH2 muestra un mejor rendimiento que el modelo SYNTH1. Salvo tres métricas concretas (*Precision* de la clase Planicidad, *Precision* de la clase Perpendicularidad, y *Precision* de la clase Posición), el modelo SYNTH2 ofrece valores más altos en cada una de las categorías estudiadas, traduciéndose en una diferencia de casi el 30% entre los valores del mAP de ambos modelos (53.96% vs 81.81%).

3.7.2.2 Modelo híbrido (HYBRID)

Se presentan a continuación los resultados que presenta el modelo HYBRID, es decir, el modelo entrenado a partir de datos tanto reales como sintéticos, con los dos sets de datos sintéticos empleados a partir de las dos versiones del generador de planos sintéticos: la versión inicial y la que incluía las mejoras propuestas. Los resultados figuran en la Tabla 5.

Clase	Planicidad			Perpendicularidad			Paralelismo			Concentricidad			Posición			Circular			mAP
Modelo	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	Precision	Recall	AP	
HYBRID1	0.84	0.95	95.46%	0.76	0.78	81.23%	0.49	0.84	80.42%	0.78	0.94	94.04%	0.81	0.70	78.08%	0.63	0.73	76.88%	84.35%
HYBRID2	0.74	0.93	91.28%	0.59	0.84	79.88%	0.81	0.88	84.59%	0.89	0.94	96.30%	0.77	0.82	81.98%	0.76	0.92	88.06%	87.02%

Tabla 5: Resultados de los modelos HYBRID

De la Tabla 5 se procede a realizar una comparación de ambas versiones del modelo HYBRID. Si bien el modelo HYBRID1 presenta un mejor rendimiento frente a la clase Planicidad (con un valor mayor en todas las métricas relativas a esta clase) y en otras categorías concretas como la *Precision* y el AP de Perpendicularidad, y la *Precision* de la clase Posición, el modelo HYBRID2 ofrece mejores resultados a nivel general. El mAP del modelo HYBRID1 es de 84.35%, mientras que el del modelo HYBRID2 es de 87.02%, con una diferencia de apenas 3%.

3.7.3 ELECCIÓN DEL MODELO

El análisis de los resultados obtenidos permite determinar qué modelo es el más completo y podría funcionar mejor en una posible aplicación industrial. Con ambas versiones del generador sintético de planos, se aprecia que el modelo híbrido, entrenado tanto con planos reales como sintéticos, es el que mejor funciona. Los datos sintéticos ayudan a completar el conjunto de datos de entrenamiento, eliminando el problema de la escasez de ejemplos; por otro lado, los datos reales ayudan a mejorar la generalización del modelo, incrementando la variabilidad del conjunto de datos empleado.

En base al análisis realizado, el mejor modelo es el modelo HYBRID obtenido con la segunda versión del generador. Es el modelo que mejor mAP general presenta y el que, a su vez, presenta mejores métricas en cada una de las clases individuales.

3.8 APLICACIÓN E IMPLEMENTACIÓN

De cara a probar el modelo elegido y su posible uso en un entorno industrial, se decidió desarrollar una Interfaz Gráfica de Usuario o GUI (*Graphical User Interface*) que permitiera el uso de la red neuronal en el proceso de detección de tolerancias geométricas en planos introducidos por un usuario. La creación de esta herramienta se realizaría por medio de Python, lenguaje que ofrece una forma sencilla de hacer uso de una arquitectura de *Deep Learning* como la empleada en este proyecto.

3.8.1 DETALLES DEL DESARROLLO DE LA APLICACIÓN

Para el desarrollo de esta GUI, se partió de una versión anterior realizada por otros alumnos en etapas anteriores del proyecto. Si bien se realizaron múltiples cambios, se reciclaron gran parte de las funciones presentes en esta versión ya existente, pues no requerían ningún tipo de modificación.

Esta herramienta se basó en dos librerías fundamentales:

- Tkinter: esta librería permite la creación sencilla de interfaces gráficas de usuario para aplicaciones sencillas.
- CV2: aparte de ser una librería para la gestión de imágenes, también permite el uso y manejo de un modelo de *Deep Learning* destinado a Computer Vision (como el empleado en este caso).

3.8.1.1 Funcionamiento

La aplicación desarrollada en Python permite al usuario seleccionar aquellos archivos sobre los que desea realizar el proceso de detección de tolerancias. El menú inicial de la herramienta cuenta con cuatro opciones diferentes:

1. Seleccionar archivo: se trata de la opción que debe marcar el usuario si desea introducir un único plano o archivo.

2. Seleccionar múltiples archivos: permite al usuario seleccionar varios archivos a su gusto, que se procesan uno tras otro.
3. Seleccionar carpeta: permite al usuario seleccionar una carpeta entera y procesar todos los documentos que en ella se encuentren.
4. Cerrar: cierra la herramienta si no se desea hacer ninguna operación adicional.



Figura 48: Menú de la GUI

Cuando el usuario introduce uno o varios planos, estos se recortan en imágenes de 512x512 píxeles (tamaño para el cual está preparada la red), los cuales son procesados por la red neuronal para identificar las tolerancias que en ellos se encuentran. Más tarde, los recortes ya procesados vuelven a ser reconstruidos para devolver el plano completo, con las tolerancias ya identificadas.

La herramienta utiliza en este proceso cuatro carpetas:

- PNG_Plane: esta carpeta contiene los planos introducidos por el usuario, los cuales deben ser procesados por la red neuronal.

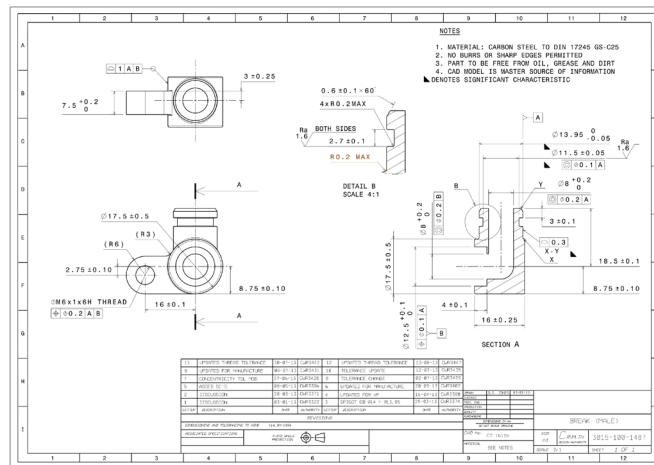


Figura 49: Ejemplo de un plano que se encuentra en la carpeta PNG_Plane

- Cropped_planes: guarda los recortes de los planos que introduce el usuario.

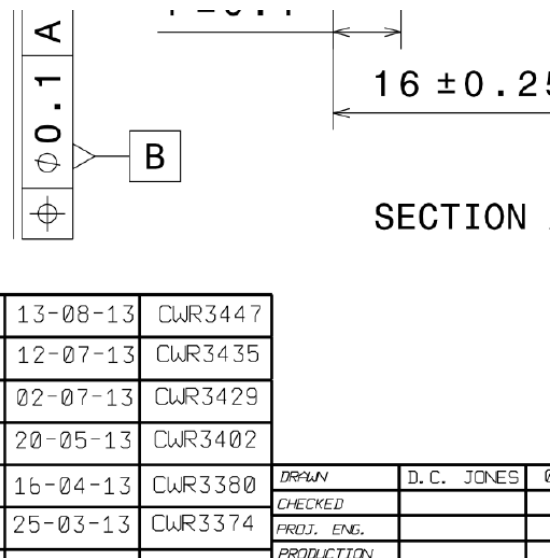


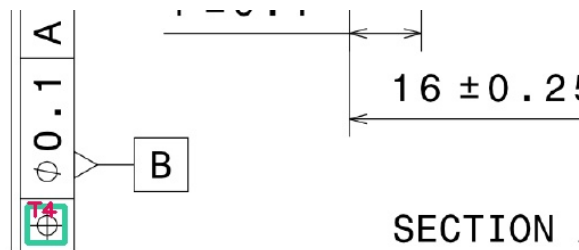
Figura 50: Ejemplo de un recorte que se encuentra en la carpeta Cropped_planes

- Cropped_planes_txt: guarda un CSV que recoge la posición de cada recorte en el plano original, para poder reconstruirlo una vez la detección de tolerancias ha sido realizada por la red.

Name	Coord_x	Coord_y	Shape_x	Shape_y	Total_size_x	Total_size_y
1 plane_5_crop_1.png	0	0	512	512	3307	2339
2 plane_5_crop_2.png	0	492	512	512	3307	2339
3 plane_5_crop_3.png	0	1004	512	512	3307	2339
4 plane_5_crop_4.png	0	1516	512	512	3307	2339
5 plane_5_crop_5.png	0	2028	512	311	3307	2339
6 plane_5_crop_6.png	492	0	512	512	3307	2339
7 plane_5_crop_7.png	492	492	512	512	3307	2339
8 plane_5_crop_8.png	492	1004	512	512	3307	2339
9 plane_5_crop_9.png	492	1516	512	512	3307	2339
10 plane_5_crop_10.png	492	2028	512	311	3307	2339
11 plane_5_crop_11.png	1004	0	512	512	3307	2339
12 plane_5_crop_12.png	1004	492	512	512	3307	2339
13 plane_5_crop_13.png	1004	1004	512	512	3307	2339
14 plane_5_crop_14.png	1004	1516	512	512	3307	2339
15 plane_5_crop_15.png	1004	2028	512	311	3307	2339
16 plane_5_crop_16.png	1516	0	512	512	3307	2339
17 plane_5_crop_17.png	1516	492	512	512	3307	2339
18 plane_5_crop_18.png	1516	1004	512	512	3307	2339
19 plane_5_crop_19.png	1516	1516	512	512	3307	2339

Figura 51: CSV que contiene los datos de los recortes de un plano

- Output_yolo: contiene los recortes que han sido procesados por la red, con las cajas que contienen las tolerancias ya añadidas.



13-08-13	CWR3447		
12-07-13	CWR3435		
02-07-13	CWR3429		
20-05-13	CWR3402		
16-04-13	CWR3380	DRAWN	D. C. JONES
25-03-13	CWR3374	CHECKED	
		PROJ. ENG.	
		PRODUCTION	

Figura 52: Ejemplo de recorte que se encuentra en la carpeta Output_yolo

- Planos_Reconstruidos: contiene los planos reconstruidos y ya procesados; en definitiva, el plano con la detección ya realizada.

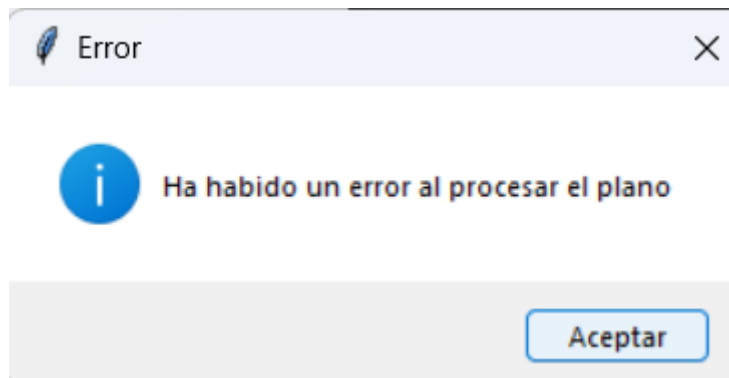


Figura 54: Ejemplo de mensaje de error devuelto por la herramienta

3.8.1.2.2 Selección de archivos PDF

Una de las primeras mejoras que se introdujeron fue la posibilidad de introducir archivos PDF. La versión inicial únicamente admitía imágenes como archivo válido. Dado que es habitual que este tipo de planos se encuentre en formato PDF, se adaptó la herramienta para permitir introducir archivos en este formato y ahorrar tiempo al usuario que emplee la herramienta. En caso de introducirse un archivo PDF, la herramienta lo convierte a un archivo de tipo imagen que puede ser procesado por la red neuronal.

3.8.1.2.3 Restricción de archivos

La versión inicial de la herramienta no restringía el tipo de archivos. De esta manera, el usuario podía introducir cualquier tipo de archivo, independientemente de su extensión. Dado que la herramienta únicamente maneja adecuadamente archivos .pdf, .jpeg, .jpg y .png, seleccionar un archivo que no tenga estas extensiones puede derivar en que la herramienta no funcione de manera correcta. Así, se ha reprogramado la herramienta para que “obligue” al usuario a introducir los archivos en el formato deseado.

En el caso de seleccionar archivos de manera individual, la restricción se realiza en el momento de la selección, de manera que no deja introducir archivos con extensiones diferentes a las mencionadas anteriormente (ver Figura 55).

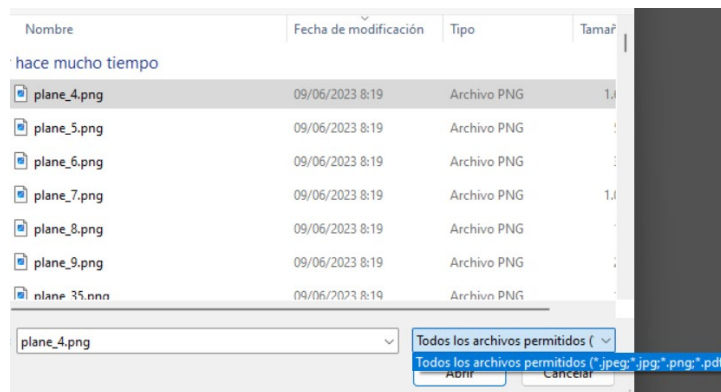


Figura 55: Restricción en las extensiones a la hora de seleccionar archivos individuales

En el caso de seleccionar una carpeta entera, la herramienta se asegura de que todos los archivos que en ella se encuentran tengan una extensión admitida; de no ser así, salta un mensaje al usuario indicándole que debe corregir las extensiones y le devuelve al menú inicial (ver Figura 56).

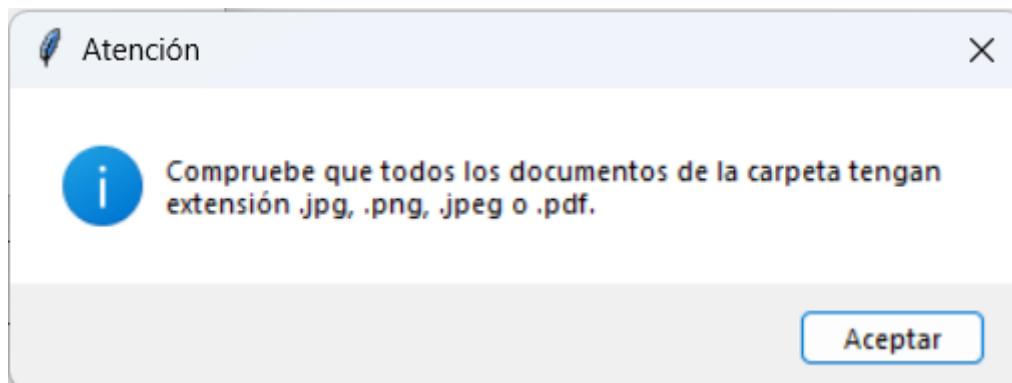


Figura 56: Mensaje que salta cuando algún archivo de la carpeta no tiene una extensión válida

3.8.1.2.4 Selección de múltiples archivos

Inicialmente, la GUI únicamente contaba con dos opciones de selección de archivos: introducir un único archivo o introducir una carpeta entera. Si bien ambas opciones resultan muy útiles de cara a procesar planos con la aplicación, dificultaba el uso de esta en dos situaciones:

1. Selección de varios planos que se encuentran en carpetas diferentes.
2. Selección de únicamente algunos archivos de una carpeta, en lugar de una carpeta entera.

Con el objetivo de mejorar la experiencia de usuario, se decidió añadir la opción de seleccionar varios archivos de manera manual, de forma que el usuario pueda elegir qué archivos específicos quiere procesar, pinchando sobre ellos en una ventana emergente (ver Figura 57).

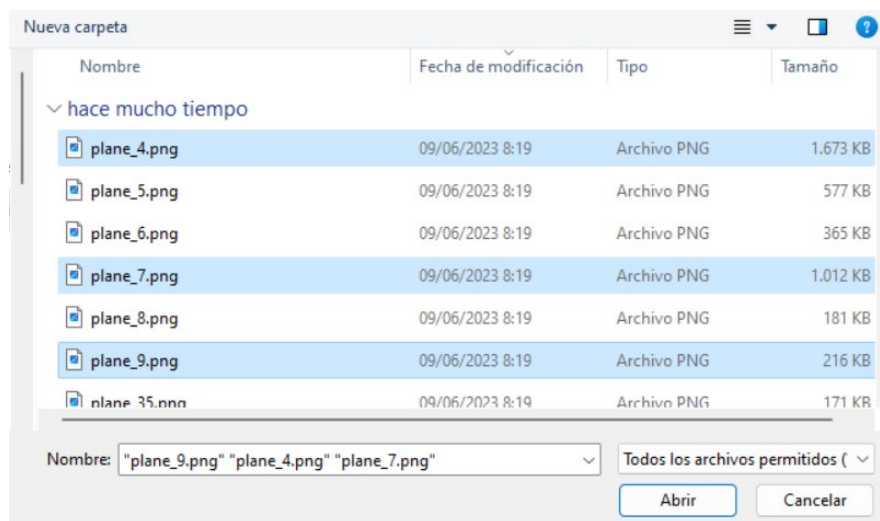


Figura 57: Selección de múltiples archivos de manera no masiva

3.8.1.2.5 Lógica de los mensajes de la herramienta

La última mejora que se incluyó en la herramienta de procesamiento de planos fue la adición de estructuras condicionales que proporcionarían un orden lógico en la aparición de los mensajes que la GUI manda al usuario.

Para describir esta mejora se procede a detallar un ejemplo de lo que ocurriría con la versión anterior del software.

Cuando se inicia el proceso de selección de un plano, aparece una ventana emergente de los archivos del ordenador, de manera que el usuario puede elegir qué archivo quiere procesar. En la versión inicial, si el usuario cerraba la pestaña sin elegir ningún archivo, se mostraba el mensaje “El plano se ha procesado correctamente”; así, el mensaje aparecía

independientemente de si se seleccionaba o no un archivo o de si el plano se procesaba bien o mal.

Con el objetivo de mejorar el funcionamiento de la GUI, se procedió a incluir una serie de estructuras para aportar lógica a los mensajes que el usuario recibe de la herramienta; de esta manera, se establecieron las condiciones necesarias para que cierto mensaje salte, asegurando que este sea recibido por el usuario en la situación correcta.

Capítulo 4. CONCLUSIONES Y TRABAJOS FUTUROS

4.1 CONCLUSIONES

Como se estipuló en la sección Objetivos del proyecto, el objetivo fundamental de este proyecto era **mejorar el rendimiento de un sistema de detección de símbolos en planos de ingeniería basado en la tecnología YOLO**. Tras la finalización del trabajo, se da este objetivo como conseguido.

La realización de este trabajo buscaba resolver el problema de la escasez de datos reales para entrenar una red neuronal mediante la creación de datos sintéticos.

Así, se entrenó la misma red neuronal con tres sets de datos diferentes: uno real, con una cantidad reducida de ejemplos de las tolerancias a identificar por la red; uno sintético, con una mayor cantidad de ejemplos, aunque obtenidos de manera artificial; y uno híbrido, combinación de los dos anteriores.

Tras terminar el trabajo, se pudo observar que los modelos entrenados con el set de datos sintético y el set de datos híbrido arrojan mejores resultados a la hora de identificar las tolerancias de interés que los modelos entrenados con el set de datos reales, incluso con un tiempo de entrenamiento cuatro veces más bajo.

La conclusión a la que se llega es clara: tanto por sí solos como en combinación con los datos reales, los datos sintéticos, adecuadamente generados, mejoran el sistema de detección de tolerancias en planos ingenieriles, representando una solución sólida frente a la falta de ejemplos reales para el entrenamiento del modelo.

Yendo más allá, los resultados obtenidos permiten concluir que la combinación del set de datos reales y el set de datos sintético es el que mejor rendimiento ofrece. Así, todo parece indicar que la contribución de los datos sintéticos al entrenamiento del modelo, al menos en este

contexto, es mejor cuando complementa a los datos reales que cuando los sustituye como conjunto de datos de entrenamiento.

4.2 TRABAJOS FUTUROS

A partir de este proyecto y su realización, se describen potenciales trabajos futuros, encaminados a seguir mejorando la aplicación de la tecnología de *Deep Learning* a la detección de símbolos en planos industriales.

4.2.1 MEJORA DEL GENERADOR SINTÉTICO DE PLANOS

Si bien se han realizado numerosas mejoras al generador de planos artificiales, el mAP más alto que se ha conseguido a lo largo de este proyecto, logrado con el modelo que empleaba tanto los datos reales como los datos sintéticos obtenidos con la versión mejorada del generador, ha sido 87.02%.

Existe amplio margen de mejora en la detección de las clases de interés, registrando algunos valores de *Precision*, *Recall* y *AP* muy inferiores a los deseados.

Para una potencial inclusión de este tipo de tecnologías en el mundo de la manufactura y producción industrial, se necesita un modelo que pueda llegar a los niveles de precisión humana. Ante la escasez de datos reales, conseguir este objetivo pasa por continuar mejorando el generador de planos sintéticos, de manera que este sea capaz de crear planos más realistas, los cuales capturen de manera más completa la variabilidad que se observa en los planos ingenieriles reales.

4.2.2 SOLUCIÓN AL PROBLEMA DEL RECORTE DE TOLERANCIAS

Desde el inicio del proyecto se ha tenido que lidiar con el problema de las tolerancias recortadas. A la hora de entrenar y usar la red neuronal, los planos reales debían ser transformados en recortes de 512x512 píxeles; esto hacía que las tolerancias que se hallaban en el límite entre dos recortes quedaran cortadas, dificultando tanto su uso como ejemplo en el entrenamiento como su uso a la hora de evaluar los resultados del modelo. Para lidiar con este problema, se

añadió cierto solape entre los recortes, ajustando este hiper parámetro al valor que menos tolerancias cortara. El desarrollo de una herramienta que ajustara el valor de ese hiper parámetro de manera dinámica, de forma que no se diese ningún caso en el que una tolerancia quede cortada a la mitad, podría contribuir a mejorar tanto el entrenamiento como la evaluación de los modelos estudiados.

4.2.3 DESARROLLO DE UNA WEBAPP

Si bien una herramienta muy sencilla se ha desarrollado a modo de demo para este proyecto, esta puede ser perfeccionada y transformada en una webapp que pueda ser potencialmente empleada por un fabricante en la etapa inicial de sus procesos de producción. Una vez desarrollada la tecnología suficientemente precisa para poder sustituir a un humano en la tarea de identificar tolerancias y otros símbolos, convertir la herramienta en un software intuitivo y manejable por un fabricante es el siguiente paso en el proceso de aplicar las técnicas del *Deep Learning* a la detección de objetos en el sector industrial.

Capítulo 5. BIBLIOGRAFÍA

- [1] N. O'Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. Velasco-Hernandez, L. Krpalkova, D. Riordan y J. Walsh, «Deep Learning vs. Traditional Computer Vision,» *ArXiv (Cornell University)*, 2019, doi: <https://doi.org/10.48550/arXiv.1910.13796>.
- [2] F. Ferreira y H. Guerra, «The coordinate measuring machines, essential tools for quality control of dimensional and geometrical specifications of technical components, in the context of the industry 4.0,» *Journal of Physics: Conference 1044, 012065*, Rio de Janeiro, Brazil (July 31-August 3 2017), 2018, doi: <https://doi.org/10.1088/1742-6596/1044/1/012065>.
- [3] R. Malhan y S. K. Gupta, «The Role of Deep Learning in Manufacturing Applications: Challenges and Opportunities,» *Journal of Computing and Information Science in Engineering*, 23(6), 2023, doi: <https://doi.org/10.1115/1.4062939>.
- [4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit y N. Houlsby, «An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale,» *ArXiv (Cornell University)*, 2021, doi: <https://doi.org/10.48550/arXiv.2010.11929>.
- [5] M. Krichen, «Convolutional Neural Networks: A Survey,» *Computers 2023*, 12(8), 151, 2023, doi: <https://doi.org/10.3390/computers12080151>.
- [6] D. Strigl, K. Kofler y S. Podlipnig, «Performance and Scalability of GPU-Based Convolutional Neural Networks,» *2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*, Pisa, Italy (February 17-19, 2010), pp. 317-324, 2010, doi: <https://doi.org/10.1109/PDP.2010.43>.

- [7] K. O'Shea y R. Nash, «An Introduction to Convolutional Neural Networks,» *ArXiv (Cornell University)*, 2015, doi: <https://doi.org/10.48550/arXiv.1511.08458>.
- [8] Y. Lecun, Y. Bengio y G. Hinton, «Deep Learning,» *Nature*, 521(7553), pp. 436-444, 2015, doi: <https://doi.org/10.1038/nature14539>.
- [9] Y. Lecun, L. Bottou, Y. Bengio y P. Haffner, «Gradient-based learning applied to document recognition,» *Proceedings of the IEEE*, 86(11), pp. 2278-2324, 1998, doi: <https://doi.org/10.1109/5.726791>.
- [10] A. Krizhevsky, I. Sutskever y G. E. Hinton, «ImageNet Classification with Deep Convolutional Neural Networks,» *Communications of the ACM*, 60(6), pp. 84-90, 2012, doi: <https://doi.org/10.1145/3065386>.
- [11] K. Simonyan y A. Zisserman, «Very deep convolutional networks for large-scale image recognition,» *ArXiv (Cornell University)*, 2015, doi: <https://doi.org/10.48550/arXiv.1409.1556>.
- [12] K. He, X. Zhang, S. Ren y J. Sun, «Deep Residual Learning for Image Recognition,» *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770-778, Las Vegas, NV, USA (June 27-30 2016), pp. 770-778, 2016, doi: <https://doi.org/10.1109/cvpr.2016.90>.
- [13] J. Redmon, S. Divvala, R. Girshick y A. Farhadi, «You Only Look Once: Unified, Real-Time Object Detection,» *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA (June 27-30, 2016), pp. 779-788, 2016, doi: <https://doi.org/10.1109/cvpr.2016.91>.
- [14] A. Nazir y M. A. Wani, «You Only Look Once - Object Detection Models: A Review,» *2023 10th International Conference on Computing for Sustainable Global Development*

- (*INDIACom*), New Delhi, India (March 15-17, 2023), pp. 1088-1095, 2023, URL: <https://ieeexplore.ieee.org/document/10112271>, [Último acceso: 10 de febrero de 2024].
- [15] J. Kim, J. Y. Sung y S. Park, «Comparison of Faster-RCNN, YOLO, and SSD for Real-Time Vehicle Type Recognition,» *2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia)*, Seoul, South Korea (November 1-3, 2020), pp. 1-4, 2020, doi: <https://doi.org/10.1109/icce-asia49877.2020.9277040>.
- [16] R. Savas y J. Hinckeldeyn, «Critical Evaluation of LOCO dataset with Machine Learning,» *ArXiv (Cornell University)*, 2022, doi: <https://doi.org/10.48550/arxiv.2209.13499>.
- [17] A. Bochkovskiy, C. Y. Wang y H. Y. Liao, «YOLOv4: Optimal Speed and Accuracy of Object Detection,» *ArXiv (Cornell University)*, 2020, doi: <https://doi.org/10.48550/arXiv.2004.10934>.
- [18] Y. Jiang, «ECE Senior Capstone Project 2021 Tech Notes Object Detection,» 2021, URL: https://sites.tufts.edu/eeseniordesignhandbook/files/2021/05/Jiang_ObjectDetection.pdf, [Último acceso: 11 de febrero de 2024].
- [19] N. Dalal y B. Triggs, «Histograms of Oriented Gradients for Human Detection,» *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, San Diego, CA, USA (June 20-26, 2005), pp. 886-893 vol. 1, 2005, doi: <https://doi.org/10.1109/cvpr.2005.177>.
- [20] R. Girshick, J. Donahue, T. Darrell y J. Malik, «Rich feature hierarchies for accurate object detection and semantic segmentation Tech report (v5),» *ArXiv (Cornell University)*, 2013, doi: <https://doi.org/10.48550/arXiv.1311.2524>.
- [21] R. Girshick, «Fast R-CNN,» *ArXiv (Cornell University)*, 2015, doi: <https://doi.org/10.48550/arXiv.1504.08083>.

- [22] S. Ren, K. He, R. Girshick y J. Sun, «Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,» *ArXiv (Cornell University)*, 2015, doi: <https://doi.org/10.48550/arxiv.1506.01497>.
- [23] J. Du, «Understanding of Object Detection Based on CNN Family and YOLO,» *Journal of Physics: Conference Series*, 1004, 012029, Hong Kong, China (February 23-25, 2018), 2018, doi: <https://doi.org/10.1088/1742-6596/1004/1/012029>.
- [24] R. Cheng, «A survey: Comparison between Convolutional Neural Network and YOLO in image identification,» *Journal of Physics: Conference Series*, 1453, 012139, Xi'an, China (October 25-27, 2019), 2020, doi: <https://doi.org/10.1088/1742-6596/1453/1/012139>.
- [25] Z. Susskind, A. Arora, S. Miranda, L. Armando, R. Fontella, L. Santiago, D. Leonel, P. Machado, F. M. Galvao, M. Breternitz y L. Z. John, «Weightless Neural Networks for Efficient Edge Inference,» *ArXiv (Cornell University)*, 2022, doi: <https://doi.org/10.48550/arxiv.2203.01479>.
- [26] M. C. Chiou, W. Liu y K. Wong, «Machine Learning Tool Development in Fire Safety Design Review,» *DEStech Transactions on Computer Science and Engineering*, ammms., 2019, doi: <https://doi.org/10.12783/dtce/ammms2018/27246>.
- [27] H. Bhanbhro, Y. K. Hooi y Z. Hassan, «Modern Approaches towards Object Detection of Complex Engineering Drawings,» *2022 International Conference on Digital Transformation and Intelligence (ICDI)*, Kuching, Sarawak, Malasia (December 1-2, 2022), pp. 01-06, 2022, doi: <https://doi.org/10.1109/icdi57181.2022.10007400>.
- [28] J. K. Nurminen, K. Rainio, J. K. Numminen, T. Syrjänen, N. Paganus y K. Honkoila, «Object Detection in Design Diagrams with Machine Learning,» *Progress in Computer Recognition Systems. CORES 2019. Advances in Intelligent Systems and Computing*, vol 977, Polanica Zdroj, Poland (May 20-22, 2019), 2019, doi: https://doi.org/10.1007/978-3-030-19738-4_4.

- [29] Y. H. Lin, Y. H. Ting, Y. C. Huang, K. L. Cheng y W. R. Jong, «Integration of Deep Learning for Automatic Recognition of 2D Engineering Drawings,» *Machines*, 11(8), 802, 2023, doi: <https://doi.org/10.3390/machines11080802>.
- [30] C. F. Moreno-García, E. Elyan y C. Jayne, «New trends on digitisation of complex engineering drawings,» *Neural Computing and Applications*, 31(6), pp. 1695-1712, 2019, doi: <https://doi.org/10.1007/s00521-018-3583-1>.
- [31] V. Berkahn y S. Tilleke, «Merging neural networks and topological models to re-engineer construction drawings,» *Advances in Engineering Software*, 39(10), pp. 812-820, 2008, doi: <https://doi.org/10.1016/j.advengsoft.2007.05.006>.
- [32] S. Mani, M. A. Haddad, D. Constantini, W. Douhard, Q. Li y L. Poirier, «Automatic Digitization of Engineering Diagrams using Deep Learning and Graph Search,» *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 673-679, Seattle, WA, USA (June 14-19, 2020), pp. 673-679, 2020, doi: <https://doi.org/10.1109/cvprw50498.2020.00096>.
- [33] P. Haneena Jasmine y S. Arun, «Machine learning applications in structural engineering - a review,» *IOP Conference Series: Materials Science and Engineering*, 1114(1), 012012, Kerala, India (December 17-19, 2020), 2021, doi: <https://doi.org/10.1088/1757-899x/1114/1/012012>.
- [34] E. Elyan, C. F. Moreno-García y C. Jayne, «Symbols Classification in Engineering Drawings,» *2018 International Joint Conference on Neural Networks (IJCNN)*, Rio de Janeiro, Brazil (July 8-13, 2018), pp. 01-08, 2018, doi: <https://doi.org/10.1109/IJCNN.2018.8489087>.
- [35] I. Elezi, A. Torcinovich, S. Vascon y M. Pelillo, «Transductive Label Augmentation for Improved Deep Network Learning,» *ArXiv (Cornell University)*, 2018, <https://doi.org/10.1109/icpr.2018.8545524>.

- [36] H. Bhanbhro, Y. K. Hooi, W. Kusakunniran y Z. Hassan Amur, «A Symbol Recognition System for Single-Line Diagrams Developed Using a Deep-Learning Approach,» *Applied Sciences*, 13(15), 8816, 2023, doi: <https://doi.org/10.3390/app13158816>.
- [37] E. Elyan, L. Jamieson y A. Ali-Gombe, «Deep learning for symbols detection and classification in engineering drawings,» *Neural Networks*, 129, pp. 91-102, 2020, doi: <https://doi.org/10.1016/j.neunet.2020.05.025>.
- [38] W. Zhang, Q. Chen, C. Koz, L. Xie, A. Regmi, S. Yamakawa, T. Furuhata, K. Shimada y L. B. Kara, «Data Augmentation of Engineering Drawings for Data-Driven Component Segmentation,» *Proceedings of the ASME 2022 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Volume 3A: 48th Design Automation Conference (DAC)*, St. Louis, Missouri, USA (August 14-17, 2022), 2022, doi: <https://doi.org/10.1115/detc2022-91043>.
- [39] L. Fu y L. B. Kara, «From engineering diagrams to engineering models: Visual recognition and applications,» *Computer-Aided Design*, 43(3), pp. 278-292, 2011, doi: <https://doi.org/10.1016/j.cad.2010.12.011>.
- [40] M. Delalandre, E. Valveny, T. Pridmore y D. Karatzas, «Generation of synthetic documents for performance evaluation of symbol recognition & spotting systems,» *IJDAR* 13(3), pp. 187-207, 2010, doi: <https://doi.org/10.1007/s10032-010-0120-x>.
- [41] «Prozion GD&T,» PROZION Metrology & Training., [En línea]. Available: <https://www.prozion.com.mx/gd-t/>. [Último acceso: 7 de febrero de 2024].
- [42] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye y D. Ren, «Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression,» *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07), New York, New York, USA (February 7-12, 2020), 12993-13000, 2020, doi: <https://doi.org/10.1609/aaai.v34i07.6999>.

- [43] A. Rosebrock, «PyImageSearch,» 2016. [En línea]. Available: <https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>. [Último acceso: 7 de febrero de 2024].
- [44] G. V. Babu, «Medium,» 2020. [En línea]. Available: <https://vignesh943628.medium.com/metrics-on-object-detection-b9fe3f1bac59>. [Último acceso: 9 de febrero de 2024].
- [45] Y. Liu, «The Confusing Metrics of AP and mAP for Object Detection,» Medium, 2018. [En línea]. Available: <https://yanfengliux.medium.com/the-confusing-metrics-of-ap-and-map-for-object-detection-3113ba0386ef>. [Último acceso: 8 de febrero de 2024].
- [46] J. Muller, A. Fregin y K. Dietmayer, «Disparity Sliding Window: Object Proposals from Disparity Images,» *ArXiv (Cornell University)*, 2018, doi: <https://doi.org/10.1109/iros.2018.8593390>.
- [47] P. M. Roth, «On-line conservative learning,» *ResearchGate*, 2011, URL: https://www.researchgate.net/publication/266215670_On-line_Conservative_Learning, [Último acceso: 15 de febrero de 2024].