

# **Memoria técnica: TF02**

Clasificación de piezas mediante Deep Learning para  
inspección industrial



**Alumno:** Martín Solano Martínez  
**Asignatura:** Análisis de Imagen y Visión Artificial

7 de enero de 2026

# Índice

<b>1. Introducción y objetivos</b>	<b>3</b>
1.1. Requisitos del sistema . . . . .	3
<b>2. Dimensionamiento del sistema de captura</b>	<b>4</b>
2.1. Cálculo de resolución y campo de visión . . . . .	4
2.2. Validación del tamaño máximo y rango dinámico . . . . .	4
2.3. Selección y especificaciones del sensor . . . . .	5
2.3.1. Justificación tecnológica: Global Shutter . . . . .	5
2.3.2. Modelo seleccionado: Sony IMX547 . . . . .	6
2.3.3. Cálculo de la distancia focal . . . . .	6
2.3.4. Selección de la lente comercial y verificación . . . . .	7
<b>3. Desarrollo de la solución software</b>	<b>7</b>
3.1. Arquitectura y eficiencia: MobileNetV2 . . . . .	7
3.2. Preprocesamiento del dataset: Extracción de ROIs . . . . .	8
3.3. Entrenamiento del modelo y estrategia de aprendizaje . . . . .	11
3.3.1. Infraestructura computacional: Google Colab . . . . .	11
3.3.2. Estrategia de Transfer Learning . . . . .	11
3.3.3. Ingesta de datos y generadores . . . . .	13
<b>4. Resultados y validación experimental</b>	<b>13</b>
4.1. Métricas de entrenamiento . . . . .	13
4.2. Matriz de confusión y métricas . . . . .	14
4.2.1. Interpretación de la fiabilidad por clase . . . . .	15
4.3. Conclusiones y validación del sistema . . . . .	15
4.4. Auditoría visual de errores . . . . .	16
<b>5. Validación de eficiencia computacional</b>	<b>17</b>
5.1. Cálculo de la ventana de tiempo . . . . .	18
5.2. Prueba de estrés (Benchmark) . . . . .	18
5.2.1. Análisis de viabilidad temporal . . . . .	18
<b>6. Conclusiones y líneas futuras</b>	<b>19</b>
6.1. Líneas futuras de mejora . . . . .	19

# 1. Introducción y objetivos

La visión artificial se ha convertido en un pilar fundamental de la industria, permitiendo la automatización de tareas de control de calidad con una precisión y velocidad superiores a la inspección humana.

Este trabajo tiene como objetivo el desarrollo integral de un sistema de visión artificial para una factoría de ensamblado de muebles. El sistema debe ser capaz de clasificar pequeño material mecánico (tornillos, tuercas, etc.) que circula por una cinta transportadora a alta velocidad.

## 1.1. Requisitos del sistema

Los requisitos operativos definidos por el ingeniero de calidad son:

- **Velocidad de cinta:** 1 m/s (1000 mm/s).
- **Distancia de trabajo:** Entre 500 mm y 700 mm.
- **Resolución mínima:** La tuerca más pequeña (6 mm de diámetro) debe representarse con al menos 135 píxeles.
- **Clasificación:** Detección robusta de múltiples clases de pequeño material mecánico.



Figura 1: Ejemplo de una imagen de nuestro dataset con sus respectivas etiquetas para las diferentes clases de pequeño material mecánico.

## 2. Dimensionamiento del sistema de captura

Para garantizar la viabilidad del software, primero es necesario dimensionar el hardware (cámara y óptica) para que cumpla con las restricciones físicas.

### 2.1. Cálculo de resolución y campo de visión

El requisito crítico establece que un objeto de  $L = 6$  mm debe ocupar  $R_{px} = 135$  píxeles. Esto define la densidad de píxeles necesaria:

$$\text{Densidad} = \frac{135 \text{ px}}{6 \text{ mm}} = 22,5 \text{ px/mm} \quad (1)$$

Considerando una cámara industrial estándar con un sensor de 5 Megapíxeles (aprox.  $2448 \times 2048$  píxeles), podemos calcular el Campo de Visión (FOV) máximo que podemos cubrir manteniendo esa densidad:

$$\text{FOV}_{\text{horizontal}} = \frac{2448 \text{ px}}{22,5 \text{ px/mm}} \approx 108,8 \text{ mm} \quad (2)$$

### 2.2. Validación del tamaño máximo y rango dinámico

Este valor de  $\text{FOV}_{\text{horizontal}} \approx 108,8$  mm establece el límite físico superior para las piezas: ningún objeto puede exceder esta longitud para ser capturado íntegramente en una sola toma. Aunque las especificaciones del proyecto no detallan las dimensiones máximas del inventario (solo la mínima de 6 mm), la inspección visual preliminar del conjunto de datos sugiere que incluso los tornillos más largos son significativamente menores a este límite.

**Validación computacional del peor caso:** Para corroborar esta observación visual de manera rigurosa y matemática, se desarrolló un algoritmo de análisis sobre la totalidad del conjunto de datos. El objetivo fue encontrar el “peor caso”: la imagen con la mayor disparidad de tamaños entre piezas, calculando el ratio entre la diagonal del objeto más grande ( $L_{\text{máx}}$ ) y la del más pequeño ( $L_{\text{mín}}$ ) en cada una de las **384 entradas únicas** procesadas.

$$\text{Ratio} = \frac{L_{\text{máx}}}{L_{\text{mín}}} \quad (3)$$

El análisis automático identificó el escenario más crítico en el conjunto de entrenamiento:

- **Imagen crítica:** screws\_005.png (Set de entrenamiento).
- **Ratio máximo ( $R_{\text{máx}}$ ): 14.27.**

A continuación se muestra la visualización generada para este caso extremo:

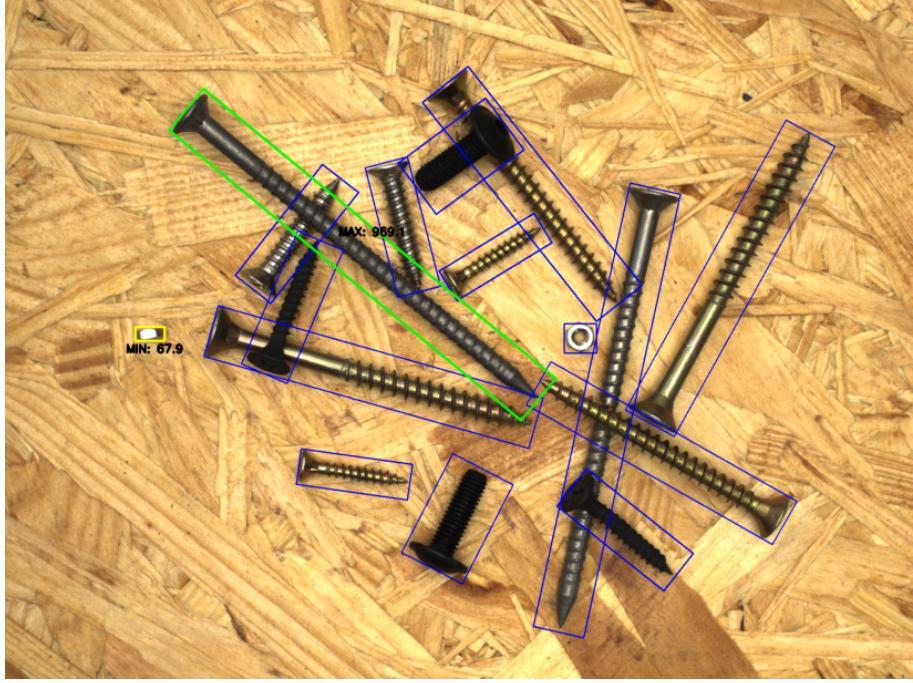


Figura 2: Validación del rango dinámico de tamaños. La imagen muestra el caso más extremo del dataset, con un ratio de tamaño de 1:14.27 entre la pieza menor (amarillo) y la mayor (verde).

**Verificación final de viabilidad:** Utilizando este ratio empírico, podemos estimar la longitud máxima física real que el sistema deberá encuadrar. Asumiendo que la pieza menor detectada corresponde al requisito de calidad de 6 mm:

$$L_{\text{máx},\text{est}} = L_{\text{mín},\text{req}} \times R_{\text{máx}} = 6 \text{ mm} \times 14,27 \approx \mathbf{85,62} \text{ mm} \quad (4)$$

Finalmente, comparamos esta longitud estimada con el FOV horizontal disponible en la cámara seleccionada (108,8 mm):

$$85,62 \text{ mm (Pieza más grande estimada)} < 108,8 \text{ mm (FOV Disponible)} \quad (5)$$

Esto nos demuestra que el dimensionamiento propuesto es válido y robusto. El sistema ofrece un margen de seguridad de aproximadamente 23 mm (21 %) incluso para las piezas más grandes del inventario, garantizando simultáneamente la resolución requerida de 22,5 px/mm para las piezas más pequeñas.

## 2.3. Selección y especificaciones del sensor

Para la correcta digitalización de la escena, la elección del hardware no se ha basado únicamente en la resolución espacial necesaria, sino también en la naturaleza dinámica del proceso de inspección, caracterizado por una cinta transportadora en movimiento continuo a 1 m/s.

### 2.3.1. Justificación tecnológica: Global Shutter

Un requisito crítico para esta aplicación es el tipo de obturación del sensor. Se ha seleccionado una cámara industrial con sensor **CMOS de obturador global (Global Shutter)**.

Dado que las piezas se desplazan a una velocidad considerable (1 m/s), el uso de un sensor convencional de *Rolling Shutter* (barrido progresivo) produciría distorsiones geométricas en la imagen capturada, inclinando visualmente las piezas y falseando las mediciones necesarias para los algoritmos de clasificación. Por el contrario, la tecnología *Global Shutter* expone todos los píxeles de la matriz simultáneamente, *congelando* el movimiento de forma precisa y garantizando la fidelidad geométrica del objeto.

### 2.3.2. Modelo seleccionado: Sony IMX547

Basándonos en estos requisitos, se ha optado por un hardware basado en el sensor **Sony IMX547**. Este modelo representa el estándar actual en la industria de visión artificial por su eficiencia y su arquitectura de obturador global.

Las especificaciones clave consideradas para los cálculos ópticos posteriores son:

- **Formato del sensor:** 1/1.8" (Formato óptico estándar).
- **Dimensiones activas:** Ancho ( $h$ )  $\approx$  6,71 mm.
- **Resolución:** 5 Megapíxeles ( $2448 \times 2048$  px), cumpliendo con la densidad de píxeles requerida.
- **Tamaño de píxel:**  $2,74\mu m$ , ofreciendo un buen equilibrio entre sensibilidad a la luz y resolución.

### 2.3.3. Cálculo de la distancia focal

Para determinar la óptica adecuada, utilizamos el modelo de cámara estenopeica (*Pinhole Camera Model*), basándonos en la relación de semejanza de triángulos entre el plano del sensor y el plano del objeto.

La fórmula fundamental relaciona la distancia focal ( $f$ ), el tamaño del sensor ( $h$ ), la distancia de trabajo ( $WD$ ) y el campo de visión ( $FOV$ ):

$$\frac{f}{h} = \frac{WD}{FOV} \Rightarrow f = \frac{h \times WD}{FOV} \quad (6)$$

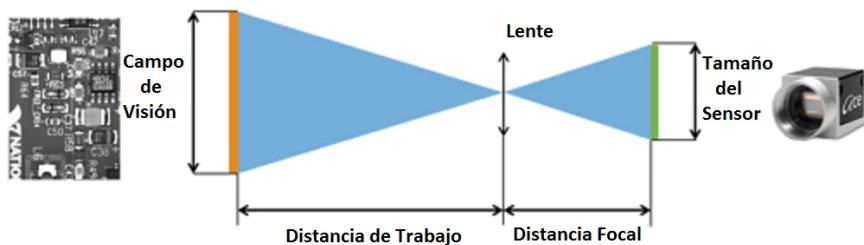


Figura 3: Geometría de la relación de semejanza de triángulos entre el plano del sensor y el plano del objeto.

Considerando el ancho del sensor  $h = 6,71$  mm, el  $FOV$  requerido de 108,8 mm calculado anteriormente, y fijando una distancia de trabajo nominal de 600 mm (centro del rango disponible 500–700 mm):

$$f = \frac{6,71 \text{ mm} \times 600 \text{ mm}}{108,8 \text{ mm}} \approx 37,0 \text{ mm} \quad (7)$$

#### 2.3.4. Selección de la lente comercial y verificación

El valor teórico de 37,0 mm nos sitúa entre dos estándares comerciales: 35 mm y 50 mm.

- **Opción 50 mm:** Requeriría una distancia de trabajo de  $WD \approx 810$  mm, excediendo el límite físico de la fábrica (700 mm).
- **Opción 35 mm:** Es la opción más cercana por defecto.

Verificamos la nueva distancia de trabajo ( $WD_{real}$ ) utilizando la lente comercial de **35 mm**:

$$WD_{real} = \frac{f_{35mm} \times FOV}{h} = \frac{35 \text{ mm} \times 108,8 \text{ mm}}{6,71 \text{ mm}} \approx 567,4 \text{ mm} \quad (8)$$

Al usar una lente comercial de 35 mm, la cámara debe colocarse a **567 mm** de la cinta. Dado que este valor está perfectamente centrado dentro del rango permitido ( $500 \leq 567 \leq 700$ ), la selección es **válida técnica y físicamente**.

**Presupuesto del equipo:**

- **Cámara:** Basler ace 2 a2A2448-75uc (5MP, Global Shutter, Sensor Sony IMX547).
  - Precio de mercado:  $\approx 650\text{€}$ .
  - *Ver ficha técnica (Basler AG)*
- **Óptica:** Lente C-Mount **35 mm** Serie Computar MPW3.
  - **Nota técnica:** Se selecciona una lente de formato **2/3"** y resolución **6MP**. Al ser especificaciones superiores a las del sensor (1/1.8" y 5MP), se garantiza el uso del centro óptico de la lente (minimizar distorsión) y una resolución suficiente para el tamaño de píxel de  $2,74\mu\text{m}$ .
  - Precio estimado:  $\approx 240\text{€}$ .
  - *Ver ficha técnica (Computar Optics)*
- **Total hardware de captura:**  $\approx 890\text{€}$  (sin incluir gastos de envío, iluminación, ni PC).

### 3. Desarrollo de la solución software

Para la clasificación de las piezas, se ha optado por técnicas de *Deep Learning* utilizando Transferencia de Aprendizaje (*Transfer Learning*).

#### 3.1. Arquitectura y eficiencia: MobileNetV2

Para garantizar la inferencia en tiempo real sobre la cinta (1 m/s), se ha seleccionado **MobileNetV2**. Esta arquitectura reduce los parámetros a  $\approx 3,5$  millones (un 86 % menos que ResNet50) gracias a las **convoluciones separables en profundidad** (*Depthwise Separable Convolutions*).

Esta técnica factoriza la costosa convolución estándar en dos operaciones secuenciales más eficientes:

1. **Depthwise** ( $3 \times 3$ ): Filtrado espacial independiente por cada canal de entrada.
2. **Pointwise** ( $1 \times 1$ ): Combinación lineal de los canales para generar nuevas características.

Esta descomposición reduce las operaciones de punto flotante (FLOPs) en un factor de **8** a **9** veces, regido por la siguiente relación (donde  $N$  son los canales y  $D_K$  el tamaño del kernel):

$$\text{Reducción de coste} \approx \frac{1}{N} + \frac{1}{D_K^2} \quad (9)$$

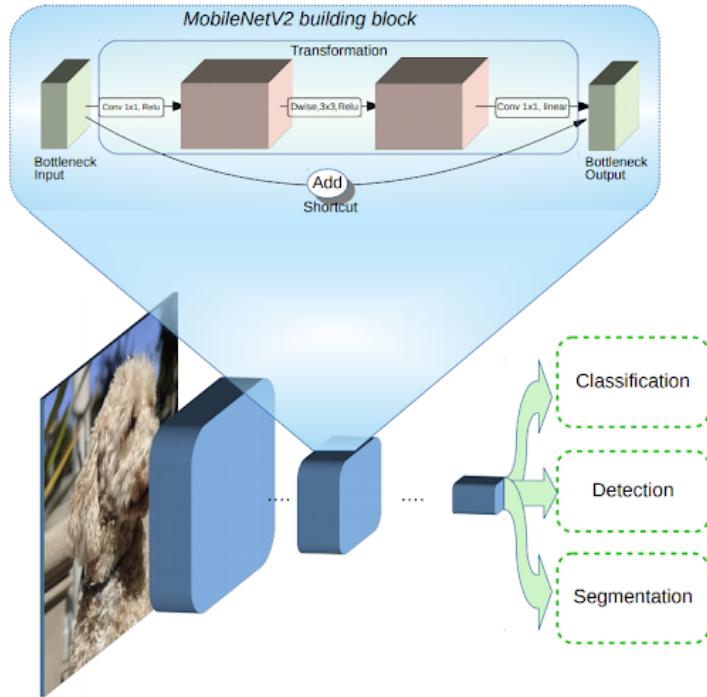


Figura 4: Descripción general de la arquitectura MobileNetV2. Los bloques azules representan bloques de construcción convolucionales compuestos, como se muestra arriba.

### 3.2. Preprocesamiento del dataset: Extracción de ROIs

El dataset original contiene imágenes de alta resolución con múltiples piezas. Para generar el conjunto de entrenamiento, se ha implementado en una celda de nuestro notebook un script de segmentación que extrae cada pieza individualmente basándose en las anotaciones proporcionadas.

El algoritmo implementado sigue la lógica de conversión de cajas rotadas a cajas envolventes rectas (*Axis-Aligned Bounding Boxes*), garantizando que la pieza completa sea capturada sin modificar la geometría de los píxeles originales.

1. **Decodificación de anotaciones (JSON):** Se procesan los archivos de metadatos para extraer los parámetros de la *Oriented Bounding Box* (OBB) de cada objeto: centro ( $row, col$ ), dimensiones ( $w, h$ ) y ángulo de rotación ( $\phi$ ).

2. **Cálculo de vértices (cv2.boxPoints):** Se transforman los parámetros de la caja rotada a coordenadas cartesianas para obtener la posición exacta ( $x, y$ ) de los cuatro vértices que delimitan el tornillo inclinado en la imagen.
3. **Generación de la envolvente recta (cv2.boundingRect):** Dado que las redes neuronales operan sobre tensores rectangulares, se calcula el rectángulo recto mínimo capaz de encerrar los cuatro vértices de la pieza.  
**Justificación:** Este enfoque genera un recorte que incluye la pieza completa y el fondo necesario para cubrir su inclinación. Al evitar la rotación artificial de la imagen, se consiguen dos ventajas clave:
  - Se evitan los artefactos de interpolación que degradan la calidad de la imagen.
  - Se preserva la variabilidad natural de rotación y escala, obligando al modelo a aprender características invariantes a la rotación.
4. **Validación de fronteras:** Se aplican algoritmos de recorte seguro para garantizar que las coordenadas ( $x, y, w, h$ ) no excedan las dimensiones físicas del sensor ( $1920 \times 1080$ ), previniendo errores de desbordamiento de memoria o recortes vacíos en los bordes de la imagen.
5. **Estructuración jerárquica del dataset:** Los recortes válidos se almacenan siguiendo una estructura de directorios estandarizada y dividida en subconjuntos funcionales (*Train*, *Validation*, *Test*). Dentro de cada partición, las imágenes se organizan en carpetas por clase semántica (`processed/train/type_XXX/`).

En la página siguiente se presentan diversas muestras del dataset procesado, lo que permite comprobar el correcto funcionamiento del script y ejemplificar la generación de la envolvente recta mencionada anteriormente.

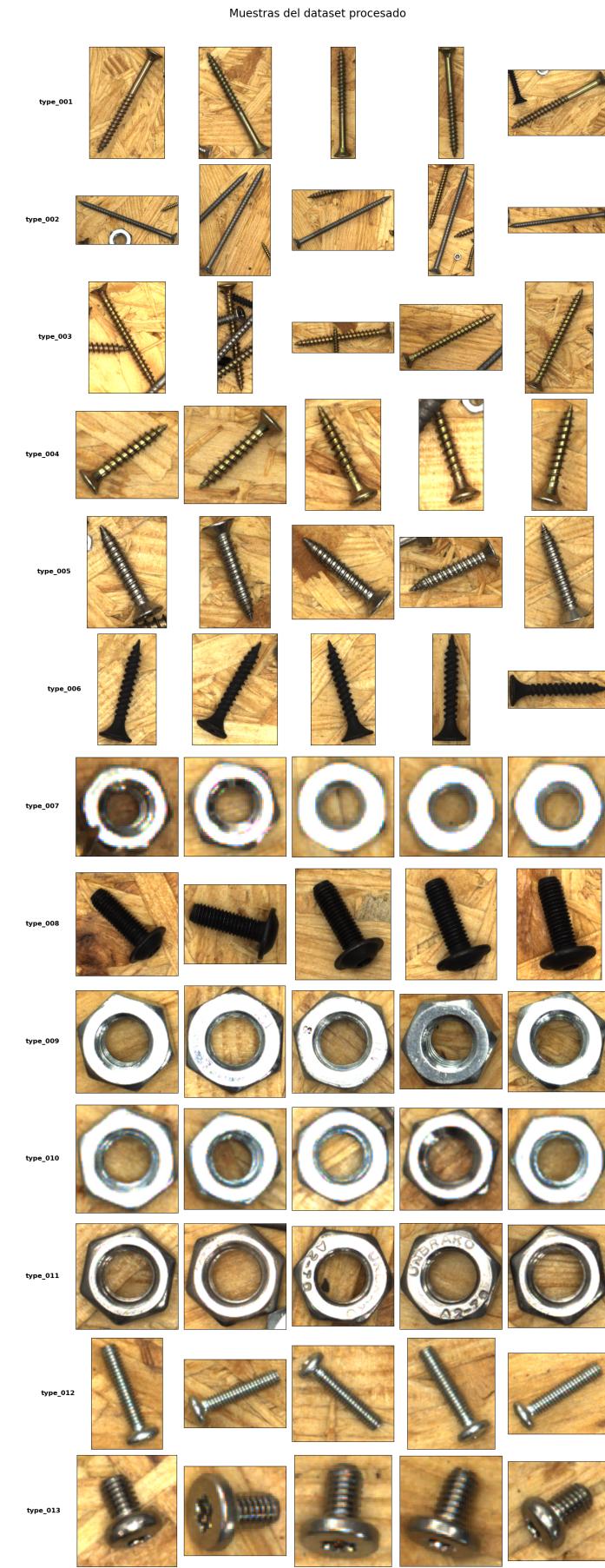


Figura 5: Muestras de tipos de clases del dataset procesado

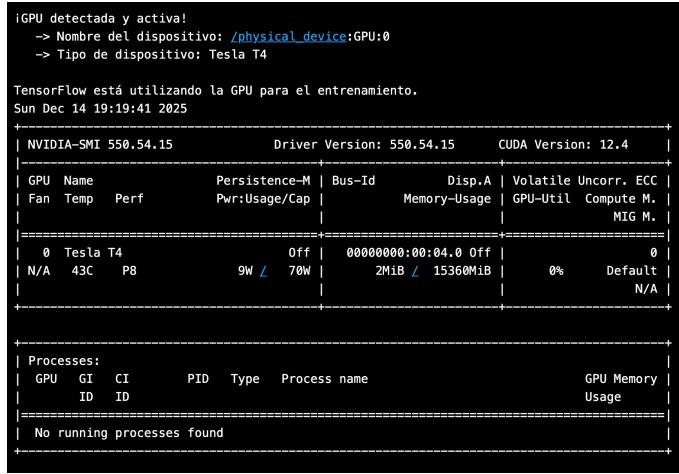
### 3.3. Entrenamiento del modelo y estrategia de aprendizaje

Una vez finalizado y verificado el preprocessamiento de las imágenes, procedemos a la fase crítica de entrenamiento del modelo de clasificación.

#### 3.3.1. Infraestructura computacional: Google Colab

Debido a la alta carga computacional inherente al entrenamiento de redes neuronales profundas (*Deep Learning*), se ha optado por externalizar el cómputo al entorno de **Google Colab**. Esta plataforma en la nube proporciona acceso gratuito a hardware de aceleración especializado, concretamente una **GPU NVIDIA T4**.

El uso de esta GPU es determinante para la viabilidad del proyecto, ya que su arquitectura paralela acelera drásticamente el cálculo de operaciones matriciales, gradientes y el algoritmo de propagación hacia atrás (*backpropagation*) en comparación con una CPU estándar.



```
iGPU detectada y activa!
-> Nombre del dispositivo: /physical_device:GPU:0
-> Tipo de dispositivo: Tesla T4

TensorFlow está utilizando la GPU para el entrenamiento.
Sun Dec 14 19:19:41 2025

+-----+
| NVIDIA-SMI 550.54.15      Driver Version: 550.54.15     CUDA Version: 12.4 |
|                               |                               |                               | |
| GPU Name                  Persistence-M | Bus-Id          Disp.A  | Volatile Uncorr. ECC |
| Fan  Temp     Perf          Pwr:Usage/Cap | Memory-Usage | GPU-Util  Compute M. |
|                               |                               |                               | MIG M.               |
+-----+
| 0  Tesla T4                Off        00000000:00:04.0 Off   0%      Default    0 |
| N/A  43C   P8              9W    70W | 2MiB  15360MiB |       0%  Default    N/A |
+-----+
| Processes:                   GPU Memory |
| GPU  GI CI      PID  Type  Process name        Usage  |
| ID   ID          |                         |
+-----+
| No running processes found |
+-----+
```

Figura 6: GPU usada para el entrenamiento del modelo.

#### 3.3.2. Estrategia de Transfer Learning

La metodología central para el entrenamiento es el **Transfer Learning** (aprendizaje por transferencia). Esta técnica consiste en reutilizar el cuerpo (las capas convolucionales extractoras de características) de un modelo pre-entrenado en un dataset masivo (ImageNet) y adaptarlo para nuestra tarea específica.

La justificación de este enfoque es doble:

- **Eficiencia de datos:** Permite obtener un alto rendimiento incluso con un dataset de tamaño moderado.
- **Reducción de tiempo:** Las capas iniciales de MobileNetV2 ya saben reconocer características visuales universales (bordes, texturas, formas geométricas), por lo que no es necesario entrenarlas desde cero.

Se ha implementado una estrategia de construcción del modelo en tres etapas:

1. **Modelo base inmutable:** Se instancia la arquitectura MobileNetV2 pre-entrenada excluyendo su capa superior original (*include\_top=False*). Se configuran todas sus capas como **congeladas** ('trainable=False'), de modo que

sus pesos permanezcan inalterados, preservando el conocimiento adquirido en ImageNet.

2. **Nueva cabeza de clasificación:** Se diseña y acopla una nueva estructura de capas finales específica para las 13 clases de piezas mecánicas:

- **GlobalAveragePooling2D:** Reduce los mapas de características espaciales a un vector de características unidimensional.
- **Dense + Dropout:** Capas densas intermedias para la combinación de características de alto nivel, con regularización *Dropout* para prevenir el sobreajuste.
- **Capa de salida (Softmax):** Una capa densa final con 13 neuronas y función de activación *Softmax*, que proporciona la distribución de probabilidad para cada tipo de tornillo.

<code>block_16_project (Conv2D)</code>	<code>(None, 7, 7, 320)</code>	<code>307,200</code>	<code>block_16_depthwi...</code>
<code>block_16_project_BN (BatchNormalizatio...</code>	<code>(None, 7, 7, 320)</code>	<code>1,280</code>	<code>block_16_project...</code>
<code>Conv_1 (Conv2D)</code>	<code>(None, 7, 7, 1280)</code>	<code>409,600</code>	<code>block_16_project...</code>
<code>Conv_1_bn (BatchNormalizatio...</code>	<code>(None, 7, 7, 1280)</code>	<code>5,120</code>	<code>Conv_1[0][0]</code>
<code>out_relu (ReLU)</code>	<code>(None, 7, 7, 1280)</code>	<code>0</code>	<code>Conv_1_bn[0][0]</code>
<code>global_average_poo... (GlobalAveragePool...</code>	<code>(None, 1280)</code>	<code>0</code>	<code>out_relu[0][0]</code>
<code>dense (Dense)</code>	<code>(None, 512)</code>	<code>655,872</code>	<code>global_average_p...</code>
<code>dropout (Dropout)</code>	<code>(None, 512)</code>	<code>0</code>	<code>dense[0][0]</code>
<code>dense_1 (Dense)</code>	<code>(None, 13)</code>	<code>6,669</code>	<code>dropout[0][0]</code>

`Total params: 2,920,525 (11.14 MB)`

`Trainable params: 662,541 (2.53 MB)`

`Non-trainable params: 2,257,984 (8.61 MB)`

Figura 7: Últimas capas de nuestro modelo de clasificación. Se puede apreciar que el número de parámetros entrenables es bastante reducido en comparación con los no entrenables.

3. **Compilación:** Se configura el proceso de aprendizaje utilizando el **optimizador Adam** con una tasa de aprendizaje conservadora ( $\eta = 1e^{-4}$ ) y la función de pérdida *Categorical Crossentropy*, idónea para problemas de clasificación multiclas.

### 3.3.3. Ingesta de datos y generadores

Para garantizar un entrenamiento eficiente y evitar el desbordamiento de la memoria RAM, no se cargan todas las imágenes simultáneamente. En su lugar, se utilizan generadores de flujo de datos ('ImageDataGenerator' de Keras/TensorFlow) que realizan tres operaciones críticas en tiempo real (al vuelo):

1. **Preprocesamiento y normalización:** Adaptación de los valores de píxel al rango  $[-1, 1]$ , formato nativo esperado por los pesos pre-entrenados de MobileNetV2.
2. **Redimensionamiento:** Estandarización de todas las entradas a tensores de dimensión fija  $224 \times 224$  píxeles, independientemente del tamaño del recorte original.
3. **Aumento de datos:** Aplicación de transformaciones aleatorias (rotaciones, desplazamientos, zoom) a las imágenes del conjunto de entrenamiento. Esta técnica simula variaciones físicas en la cinta transportadora y fuerza al modelo a aprender características estructurales robustas en lugar de memorizar las imágenes exactas, mitigando el *overfitting*.

## 4. Resultados y validación experimental

### 4.1. Métricas de entrenamiento

El entrenamiento se llevó a cabo durante 50 épocas, empleando la técnica de detención temprana (Early Stopping, 10 épocas de paciencia) y guardado del mejor modelo (Model Checkpointing) para asegurar la máxima generalización posible según la métrica del conjunto de validación.

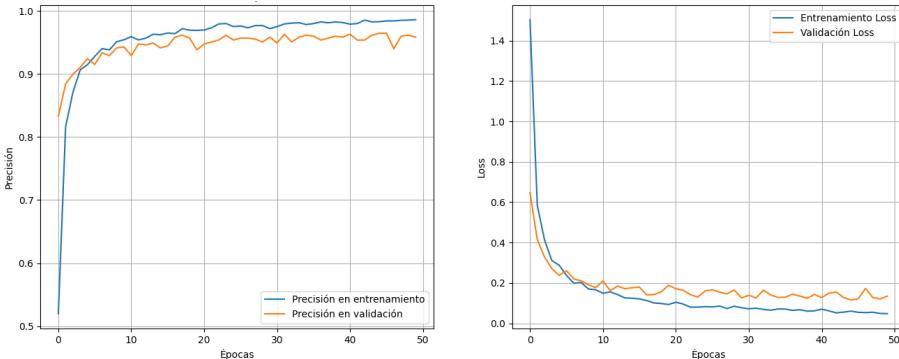


Figura 8: Curvas de precisión (izquierda) y pérdida (derecha) durante el entrenamiento. Se observa una convergencia rápida sin signos evidentes de overfitting.

Las gráficas muestran la evolución de la precisión y la función de pérdida a lo largo de las 50 épocas de entrenamiento. A continuación, se interpreta el comportamiento del modelo:

**1. Evolución de la precisión.** Se observa una **convergencia extremadamente rápida** en las primeras épocas. Este ascenso pronunciado es una prueba directa de la eficacia del **Transfer Learning**: el modelo MobileNetV2, al estar pre-entrenado en un dataset masivo, solo necesita un ajuste fino para adaptarse a la clasificación de las piezas mecánicas. El modelo alcanza valores de precisión finales superiores al **99 %** en entrenamiento y se estabiliza de forma robusta en torno al **95-96 %** en validación.

**2. Evolución de la función de pérdida.** Ambas curvas descienden rápidamente, confirmando que la red está minimizando el error.

- La pérdida de validación se aplana después de la época 15-20, mientras que la de entrenamiento continúa descendiendo. Esta **separación** (el *gap*) indica un **ligero sobreajuste** (*overfitting*), lo cual es esperable en redes profundas.
- El comportamiento estable de la curva de validación confirma que los mecanismos de regularización (**Dropout** y **Early Stopping**) funcionaron correctamente, deteniendo la degeneración del modelo y seleccionando el punto óptimo de generalización.

## 4.2. Matriz de confusión y métricas

El sistema ha sido evaluado sobre el conjunto de test aislado, alcanzando una precisión global elevada. A continuación se presentan las evidencias gráficas del rendimiento y el análisis por clases.

Reporte de Clasificación				
	precision	recall	f1-score	support
type_001	0.94	1.00	0.97	47
type_002	1.00	0.98	0.99	48
type_003	1.00	0.96	0.98	49
type_004	0.92	1.00	0.96	48
type_005	0.96	0.93	0.95	59
type_006	1.00	0.95	0.98	63
type_007	0.92	1.00	0.96	55
type_008	1.00	1.00	1.00	55
type_009	1.00	1.00	1.00	45
type_010	1.00	0.90	0.95	51
type_011	1.00	1.00	1.00	51
type_012	0.98	0.98	0.98	45
type_013	0.98	1.00	0.99	45
accuracy			0.98	661
macro avg	0.98	0.98	0.98	661
weighted avg	0.98	0.98	0.98	661

Figura 9: Reporte de clasificación detallado por clase (Precision, Recall y F1-Score).

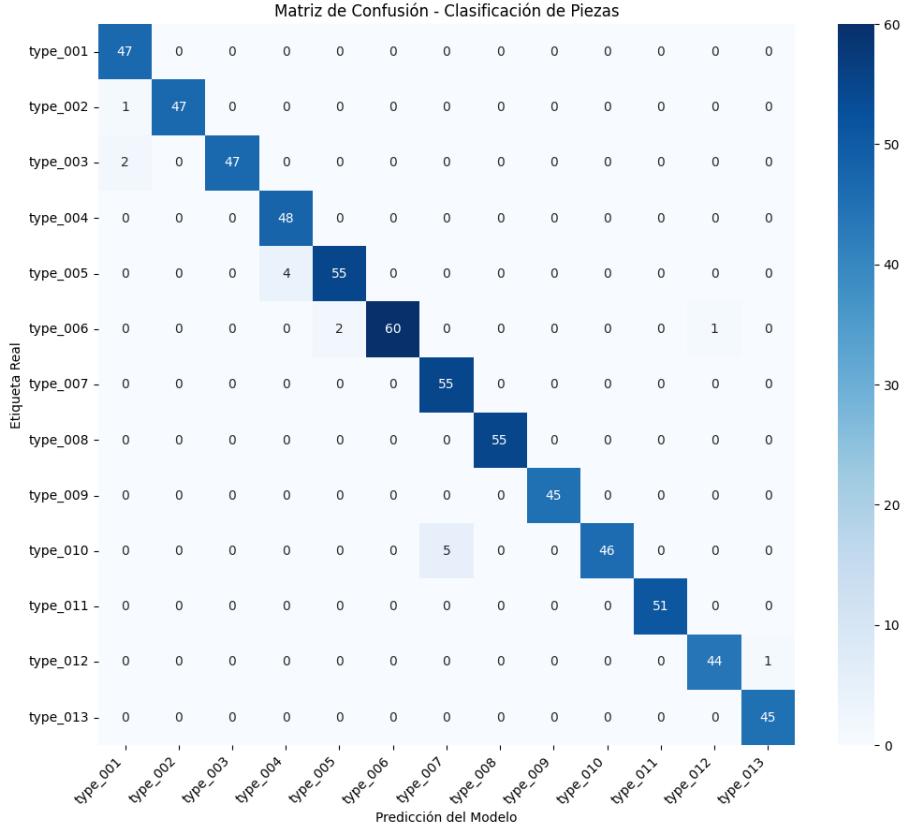


Figura 10: Matriz de confusión normalizada sobre el conjunto de test.

#### 4.2.1. Interpretación de la fiabilidad por clase

El análisis de la matriz de confusión (Figura 10) y el reporte de métricas revela un comportamiento desigual pero controlado entre las distintas categorías:

- **Identificación perfecta:** Las clases `type_008`, `type_009` y `type_011` obtuvieron métricas perfectas (Precisión y Recall del 100 %). Esto indica que estas piezas poseen características morfológicas muy distintivas (como longitud, cabeza o color) que la red ha aprendido a aislar sin error.
- **Conflictos geométricos (clases críticas):** Se ha identificado un patrón de error específico en la clase `type_010`, que presenta el rendimiento más bajo con un recall del 90 %. La matriz revela que el modelo clasifica erróneamente estas piezas como `type_007` en el 10 % de los casos. Esto sugiere una alta similitud visual entre ambos tipos de material mecánico.

### 4.3. Conclusiones y validación del sistema

El desarrollo y evaluación del sistema de visión artificial ha arrojado resultados altamente satisfactorios, permitiendo validar el prototipo para su uso industrial. Las conclusiones principales extraídas son:

1. **Rendimiento global y metodología.** El modelo ha alcanzado una exactitud (*accuracy*) del 98 % en test con un **F1-Score promedio de 0.98**.

**2. Idoneidad para el entorno industrial.** Considerando los requisitos de despliegue en una cinta transportadora de ensamblado:

- **Fiabilidad operativa:** El sistema minimiza los falsos positivos, lo cual es crítico para evitar paradas innecesarias en la línea de montaje o ensamblajes defectuosos.
- **Eficiencia computacional extrema:** La combinación de la arquitectura ligera MobileNetV2 con técnicas de compilación estática de grafos (`@tf.function`) ha permitido reducir la latencia a niveles de milisegundos, garantizando un funcionamiento en tiempo real holgado incluso en hardware convencional. (**Este aspecto será verificado empíricamente más adelante en la sección de validación de eficiencia**).

En conclusión, el prototipo cumple con los estándares técnicos y de calidad exigidos, presentándose como una solución viable y eficiente para la automatización del control de calidad en planta.

#### 4.4. Auditoría visual de errores

Se ha realizado un análisis visual profundo de los casos fallidos, revelando dos fenómenos interesantes:

1. **Errores de similitud visual:** Conflictos entre *type\_010* y *type\_007* (tuercas hexagonales). La pérdida de resolución a *224px* difumina detalles de la rosca. También ocurre este fenómeno entre otros materiales, como por ejemplo los tornillos *type\_04* y *type\_005*.
2. **Errores en el etiquetado:** En varios casos, el modelo clasificó correctamente la pieza basándose en su morfología visual, contradiciendo una etiqueta original errónea en el dataset. Esto demuestra que el modelo ha aprendido a generalizar mejor que el propio etiquetado manual. Esto lo vemos en la Figura 11 en los supuestos errores (realmente son aciertos) de las dos últimas filas.



Figura 11: Comparativa visual: A la izquierda el error (input), a la derecha el ejemplo típico de la clase predicha. Aquí estamos mostrando solo algunos de los errores.

## 5. Validación de eficiencia computacional

Un requisito crítico es la compatibilidad con la velocidad de la cinta ( $v = 1 \text{ m/s}$ ).

## 5.1. Cálculo de la ventana de tiempo

Para maximizar el tiempo disponible de inspección, se propone instalar la cámara con el lado largo del sensor alineado con la cinta.

- **Ancho Sensor 5MP:**  $\approx 2448$  píxeles.
- **Densidad óptica:**  $22,5$  px/mm.
- **FOV Longitudinal:**  $2448/22,5 \approx 108$  mm.

El tiempo que la pieza permanece en la imagen ( $T_{window}$ ) es:

$$T_{window} = \frac{\text{FOV}}{v} = \frac{108 \text{ mm}}{1000 \text{ mm/s}} = 0,108 \text{ s} = \mathbf{108 \text{ ms}} \quad (10)$$

## 5.2. Prueba de estrés (Benchmark)

Se realizaron 1000 inferencias consecutivas con imágenes reales en el entorno de pruebas, aplicando optimización mediante grafos estáticos de ejecución para simular un entorno de producción real.

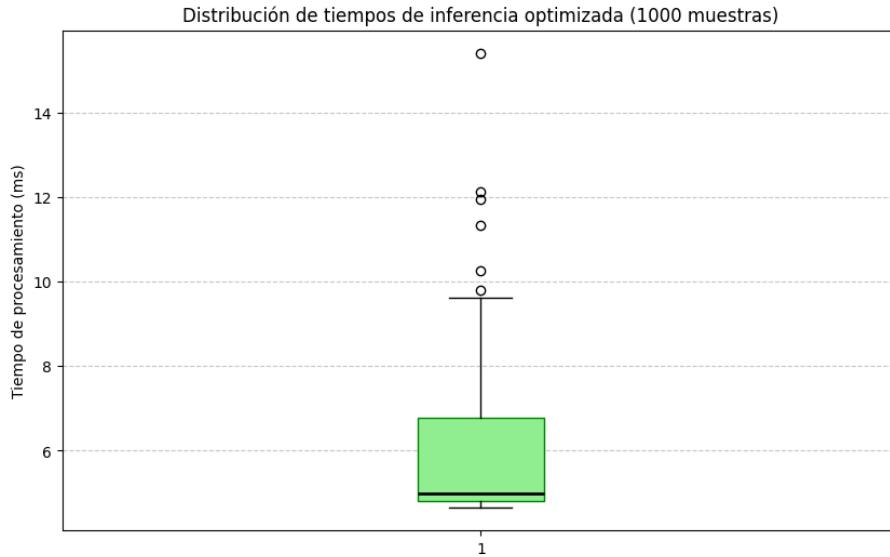


Figura 12: Distribución de tiempos de inferencia optimizada (Box-Plot).

**Resultado:** El tiempo promedio de inferencia obtenido fue de **5.77 ms**.

### 5.2.1. Análisis de viabilidad temporal

Comparando la latencia del modelo con la ventana física disponible:

$$t_{inferencia}(5,77 \text{ ms}) \ll T_{window}(108 \text{ ms}) \quad (11)$$

El sistema no solo cumple con el requisito, sino que ofrece un **margen de seguridad superior a 100 ms**. Esto implica que el algoritmo es capaz de procesar la imagen aproximadamente **18 veces** mientras la pieza cruza el campo de visión.

A diferencia de implementaciones interpretadas estándar, la optimización realizada ha estabilizado la varianza (ver Figura 12), eliminando los picos de latencia y garantizando un comportamiento determinista apto para la industria.

## 6. Conclusiones y líneas futuras

El trabajo realizado valida la viabilidad técnica de implementar redes neuronales convolucionales ligeras para la inspección visual de alta velocidad. La arquitectura MobileNetV2, optimizada mediante grafos de ejecución, ha demostrado ser capaz de operar a velocidades equivalentes a **173 FPS**, superando holgadamente las necesidades de una línea de 1 m/s.

### 6.1. Líneas futuras de mejora

Dado que el problema de la latencia ha sido resuelto mediante software, las líneas futuras se centran en la reducción de costes de hardware y la escalabilidad funcional:

1. **Implementación en Edge Computing (IoT):** Dada la bajísima latencia obtenida (5.77 ms), el sistema está sobredimensionado para un PC industrial estándar. Se propone portar el modelo a hardware de bajo coste y bajo consumo (como *Raspberry Pi 4* o *NVIDIA Jetson Nano*). Aunque el tiempo de inferencia aumentase ligeramente, seguiría estando dentro del margen de los 108 ms, permitiendo reducir drásticamente el coste por estación de inspección.
2. **Optimización del rendimiento y robustez:** El gran margen de tiempo disponible (sobran más de 100 ms por ciclo de inspección) abre dos vías estratégicas para la planta:
  - **Escalabilidad de producción:** Sería posible aumentar significativamente la velocidad de la cinta transportadora por encima de 1 m/s sin comprometer la capacidad de procesamiento del software.
  - **Inspección multi-vista para máxima precisión:** Se podría reinvertir ese tiempo sobrante en procesar múltiples capturas de la misma pieza desde distintos ángulos (utilizando sistemas de espejos o disparos secuenciales). Esto permitiría implementar un sistema de votación o validación cruzada para elevar el *accuracy* operativo incluso por encima del 98 % actual, eliminando ambigüedades por rotaciones desfavorables u occlusiones parciales.