Universidad Zaragoza
1542

Trabajo Fin de Máster

Estudio de la viabilidad de redes neuronales artificiales para la detección de defectos sobre la superficie de los salpicaderos de automóviles

Study of the feasibility of artificial neural networks for the detection of defects on the surface of car dashboards

Autor

José Antonio Núñez Carrión

Director

Jorge Santolaria Mazo

Máster en Ingeniería Industrial

ESCUELA DE INGENIERÍA Y ARQUITECTURA, 2021

Agradecimientos

Este trabajo merece un especial agradecimiento, a todas las personas que han hecho posible este trabajo y han aportado algo en mí durante este periodo de aprendizaje.
A los que han hecho posible la Cátedra CEFA, responsables de CEFA y los profesores Ángel Fernandez e Ignacio Martínez. A mi tutor, Juan José Pradas.
A mi director de trabajo, Jorge Santolaria, y sus compañeros de J3D Vision.
A mis compañeras de prácticas, por ser mis chóferes personales.
Pero sobre todo a mi familia, mis padres, José Antonio y Dolca y a mi hermana Joandra, por todo el apoyo en estos días.

Resumen

En una gran mayoría de los productos industriales, éstos se encuentran inevitablemente sujetos a una variabilidad de defectos debido a su diseño, condiciones del entorno o manipulación. Especialmente en los productos expuestos a la vista del consumidor la calidad es un aspecto especialmente crítico. Un defecto puede suponer el deshecho de la pieza con la consiguiente pérdida de recursos e impacto medioambiental.

Los sistemas de inspección de defectos modernos se basan en la generación de un barrido de iluminación para producir sobre la superficie del defecto el llamado 'efecto amplificación' como consecuencia de las sombras generadas en los alrededores del defecto aumentando el tamaño del reflejo en la superficie.

Sin embargo, las piezas de salpicadero y de revestimiento interior de automóviles presentan una serie de dificultades que dificultan su detección por este método. Presentan defectos con una alta variabilidad en sus características junto con una superficie mate oscura con huella impresa que llegan a resultar difícil su inspección incluso al ojo humano.

A modo de solucionar este problema, este trabajo presenta método de estudio de la viabilidad de la detección de defectos sobre la línea de producción de salpicaderos del modelo Opel Corsa que se produce en las instalaciones de Módulos de Ribera alta (MRA) en Figueruelas (España).

Este método propone un estudio previo en un banco de ensayos de las mejores configuraciones y condiciones para la implementación de un sistema de captura de defectos en la línea de producción. Mediante la captura masiva de imágenes de piezas en línea se ha creado un conjunto de 2160 imágenes consistentes en arrugas, gap, manchas y piezas sin defecto.

Como es habitual, en problemas de reconocimiento de imágenes, la falta de imágenes ocasiona problemas de sobreajuste y datos desbalanceados. Se han usado técnicas de data augmentation y redes GAN para la generación de datos y de regularización para corregir el sobreajuste. Usando la transferencia de aprendizaje se han entrenado cuatro modelos de redes convolucionales preentrenadas: AlexNet, GoogLeNet-ImageNet; GoogLeNet-Places365 y ResNet-50 obteniendo los resultados de precisión sobre la validación de 99,22%, 99,34%, 99,54% y 98, 54% respectivamente; y 41.58%,56.44%, 41.58%, 52.48% sobre el conjunto de test.

El modelo de mejor rendimiento ha sido usado como red de extracción para entrenamiento de detectores de objetos YOLOv2 y Faster R-CNN obteniendo un resultado máximo de mAP de 0,935.

Keywords: deep learning, transferencia de aprendizaje, redes GAN, data augmentation, mAP

Índice

1. Introducción

1.1. Motivación

El sector de los sistemas de inspección automatizados estuvo valorado en 17,6 mil millones de dólares en 2018. Se estima que para 2028 su valoración incremente un 160% hasta 45,8 mil millones de dólares. Las búsquedas de servicios relacionados con este sector se han incrementado un 34% de año a año y solo un 133% en el primer cuatrimestre de 2020. [1]

Entre sus principales aplicaciones industriales, el sector automovilístico es el que ha adoptado enormemente estos sistemas debido a que este sector requiere de un amplio abanico de procesos de alta calidad, repetibles y automatizados para el ensamblaje de las piezas. También en el sector de la alimentación ha ganado gran popularidad debido al procesamiento de un flujo elevado de productos. [1]

Estas industrias se caracterizan por presentar altos tiempos de ciclo durante largas jornadas de trabajo. A lo que el esfuerzo físico de los empleados y la exposición reiterada a tareas de concentración visual ocasionan la aparición de fatiga física y mental. La inspección humana aumenta el error de clasificación con el cansancio y la fatiga, y además tienen la tendencia de cambiar el criterio de detección con el tiempo dando lugar a una falta de consistencia en la clasificación de los datos. [1]

Para disminuir estos errores de tipo sistemático, muchas empresas se ven obligadas a buscar una solución mediante la contratación de más personal destinado a las tareas de inspección, asumiendo los costes fijos asociados y de riesgo laboral.

A diferencia de éstos, los sistemas de visión artificial presentan una precisión constante a lo largo del tiempo y son capaces de superar la predicción del cerebro humano (ResNet consiguió un error de 3,2% en 2015). Para conseguir esta eficiencia, las tareas de procesamiento de imágenes han pasado por varios hitos: clasificación; localización y detección de objetos; detección por segmentación.

A partir de las primeras aspiraciones de los científicos por emular el cerebro humano nacieron los primeros modelos simples como la neurona de McCulloch-Pitts y sus derivados. Las influencias en el campo de la neurología (Hebbs, 1949) y la biología se han conformado el modelo de perceptrón simple (F. Rosenblatt, 1958) y ADALINE (Wirow, Hoff, 1959) que demuestran su capacidad para representar funciones lógicas y despiertan admiración por su capacidad de reconocimiento de patrones.

Sin embargo, el libro 'Perceptrons' (Minsky, Papert, 1969) demuestra que las redes monocapa no son capaces de resolver problemas no lineales simples como XOR y supuso el abandono de muchos científicos en la investigación de las redes neuronales. Las redes de varias capas como el perceptrón multicapa (1965) no tenían esta limitación sin embargo los cálculos eran hechos a mano y el cálculo con muchas capas profundas era inviable.

En 1986 se abrió un nuevo panorama con el redescubrimiento del algoritmo de propagación hacia atrás o 'backpropagation' (Rumelhart, 1986) que soluciona la limitación del cálculo no lineal.

En 1989, Yan LeCun utilizó campos receptivos para el reconocimiento de códigos postales y años más tarde creó LeNet5 marcando la estructura de las redes neuronales convolucionales (CNN) que conocemos.

A partir de 2010, investigadores compiten en el 'Image Net Large Scale Visual Recognition Challenge' (ILSVRC) para crear la mejor red neuronal convolucional. No fue importante hasta que el ganador de 2012, 'AlexNet' (A. Krizhevsky, 2012) bajo la tasa de error de 25% a 17%, con más de 10.8 puntos por

debajo del siguiente competidor. Introdujo las funciones de activación ReLU que redujo el tiempo de entrenamiento sin la pérdida de precisión. [3]

1.2. Objetivos

Este trabajo tiene como propósito final el entrenamiento y validación de modelos basados en redes neuronales profundas para la clasificación de imágenes de un problema real.

Para llevar a cabo este propósito se discretizan en los siguientes objetivos simples:

- Revisión del estado del arte de los métodos de clasificación de imágenes, detección y segmentación de objetos.
- Estudio de las características y condiciones de los defectos de la pieza.
- Estudio y selección de las posibles configuraciones de captación de imágenes en linea de la pieza.
- Adquisición de un conjunto de imágenes para entrenamiento, validación y test.
- Estudio y selección de las mejores configuraciones de clasificación de defectos.
- Desarrollo de detección de objetos

1.3. Estructura de la memoria

Este trabajo se estructura de la siguiente manera:

- Capítulo 1. **Introducción**. Se explica la importancia, se describe el problema y los objetivos.

- Capítulo 2. **Fundamento teórico**. Se explican los fundamentos teóricos necesarios para entender este trabajo.

- Capítulo 3. **Estudio de la pieza**. Se realiza un estudio de conocimiento de la pieza para encontrar las mejores condiciones de capturar imágenes.

- Capítulo 4. **Estudio de la localización del sistema de captura**. Se estudian las diferentes configuraciones de captura en línea de producción.

- Capítulo 5. **Estudio de clasificación de imagen**.

- Capítulo 6. **Estudio de detección de objetos**.

1.4. Herramientas

Este trabajo incluye la adquisición de imágenes y la fase de estudio de Deep learning. Para la fase de adquisición de imágenes el sistema está constituido por una cámara industrial BFLY-PGE-20E4M-CS 1/1.8" con banda ancha Gigabit Ethernet basada en la interfaz de datos estándares GigE Vision para asegurar la transferencia de imágenes de alta resolución y el software de captura del fabricante 'Point Grey FlyCap2'. El sistema óptico consta de un equipo de lentes de distancia focal f = 16 mm, f = 50 mm, f = 100 mm. Para los estudios de deep learning, usamos Matlab 2021a con el framework de deep learning 'Matlab Deeplearning Toolbox' bajo la versión Cuda 8.0. Ejecutando un sistema operativo Microsoft Windows 10 Pro. El hardware utilizado tiene las propiedades de la Tabla 1.

Tabla 1. Configuración de hardware

Hardware	Parametro
CPU	Intel® Core™ i5-10300H @2.5GHz
RAM	DDR4 16 Gb
GPU	NVIDIA GeForce GTX 2060 6Gb

2. Fundamento teórico

2.1. Proceso de formación de la imagen

El proceso de formación de la imagen consiste en dos pasos, reflexión óptica y conversión fotoeléctrica.

2.1.1. Modelo de reflexión de Phong

El modelo de iluminación convierte las propiedades físicas como la reflectividad de la superficie en intensidad de luz reflejada. El modelo de reflexión de Phong (Ec. 1) es un modelo ampliamente usado para entender la física de la luz. Su estructura principal consiste en los términos de ambiente, términos difusos y términos especulares: [6]

$$R_p = I_p^{amb} + I_p^{dif} + I_p^{spe} = I_p^{amb} + \sum_{m \in lights} \left(k_d^{dif} (\vec{l}\,\vec{n})\, I_{m,d}^{l} + k_d^{spe} (\vec{r}\,\vec{v})\, I_{m,d}^{l} \right) \quad (1)$$

Las componentes del modelo de reflexión de Phong

1. Reflexión especular. Es la componente de luz reflejada procedente de una fuente que presenta el mismo ángulo de reflexión y de incidencia.
2. Reflexión difusa. Es la componente de la luz reflejada procedente de una fuente que se refleja en varias direcciones a consecuencia de la rugosidad de la superficie. La distribución de esta luz tiene forma de lóbulo según la ecuación de Lambert.
3. Luz ambiente. Es la componente reflejada de la luz externa del ambiente

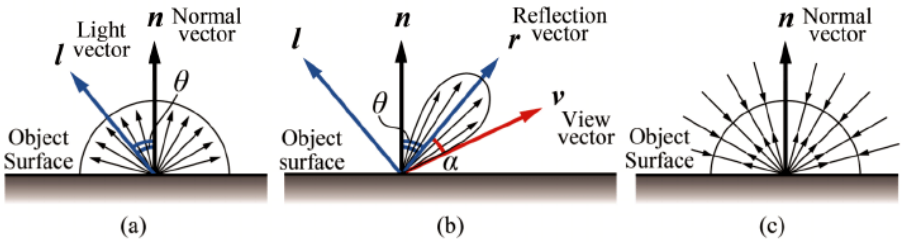


Figura 2. Componentes modelo de reflexión de Phong: (**a**) luz reflejada especular; (**b**) luz reflejada difusa; (**c**) luz ambiente.

2.1.2. Conversión fotoeléctrica

El proceso fotoeléctrico convierte la intensidad de la luz reflejada en imagen digital del objeto a través de una lente y un conversor analógico digital: [6]

$$I_p = \alpha \cdot R_p + \beta \quad (2)$$

donde α y β son parámetros constantes especificados por la cámara

Esta imagen digital contiene el valor de intensidad de los píxeles I_p, que se introducen en la capa de entrada de una red neuronal convolucional.

2.2. Redes neuronales convolucionales

Son un tipo de redes neuronales artificiales donde las neuronas corresponden a campos receptivos de una manera muy similar a las neuronas en la corteza visual primaria de un cerebro biológico. Debido a que su aplicación es realizada en matrices bidimensionales son muy efectivas para tareas de visión artificial. [5]

Las neuronas se activan cuando se reconocen ciertas características importantes para la clasificación de la imagen. Las primeras capas corresponden a características menores y más generales como esquinas, contornos, siluetas. Las siguientes corresponden son más específicas como ojos nariz boca, las siguientes son más específicas como rostros.

2.2.1. Estructura de redes convolucionales

La estructura está formada por una capa de entrada, etapas intermedias y capa totalmente conectada. Las capas intermedias están formadas por repeticiones de bloques de capas convolucionales, capas de activación y capas de agrupación. Estás tres capas forman el núcleo de bloques de redes neuronales convolucionales.

2.2.1.1. Capa de entrada

El número de neuronas de la capa de entrada corresponde al número de píxeles de la imagen y su valor es la intensidad del píxel. Su función es centrar y normalizar los datos para acelerar la convergencia.

2.2.1.2. Capas convolucionales

Las capas convolucionales definen un conjunto de filtros con pesos que son actualizados durante el entrenamiento. Estos filtros son matrices cuadradas que realizan la operación de convolución como el producto escalar del filtro por un tamaño igual en la imagen y su suma es el valor de la capa convolucionada. El filtro recorre toda la imagen calculando todos los valores de salida. [3]

2.2.1.3. Capa de activación

La capa de activación realiza una operación no lineal como la función ReLU lo que permite al modelo aproximarse a funciones no lineales que relacionan pixeles con contenidos semánticos de la red.

2.2.1.4. Capa de agrupación

Las capas de agrupación disminuyen la dimensionalidad de los datos conservando las características más importantes. Lo consiguen realizando operaciones básicas como obtener el valor máximo o media sobre los valores de una subregión de la matriz.

2.2.1.5. Capas totalmente conectadas

Son las capas encargadas de la clasificación, reciben las características de las capas anteriores y están conectadas de forma que las dependencias entre ellas producen determinados valores de activación. En las neuronas finales el valor de activación corresponde a la probabilidad de pertenencia de la imagen a una determinada clase.

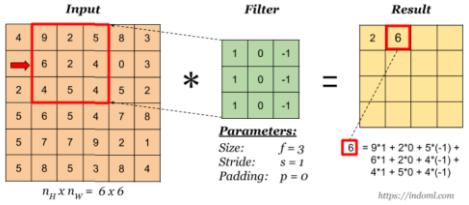


Figura 2.1. Capas convolucionales. Fuente: [4]

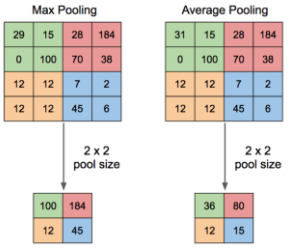


Figura 2.2. Capas de agrupación o pooling. Fuente: [4]

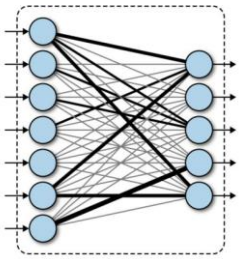


Figura 2.4. Capas totalmente conectadas. Fuente: [4]

2.3. Transferencia de aprendizaje

El diseño de la arquitectura de una red neuronal desde cero requiere alto conocimiento y experiencia en arquitectura de redes, además de una cantidad ingente de datos y mucho tiempo de computación. [7]

La transferencia de aprendizaje es el proceso mediante el cual el conocimiento sobre un tema se traspasa a otro sistema. En deep learning reutilizamos el conocimiento de extracción de características de redes preentrenadas por expertos para adaptarlas a la clasificación de nuestro problema.

Durante la transferencia de aprendizaje, las capas de clasificación de la red preentrenada son reemplazadas por nuevas capas de clasificación y se entrena el modelo con nuestros datos congelando las capas de extracción de la red anterior. De este modo se reducen drásticamente la cantidad de datos necesarios y el tiempo de entrenamiento.

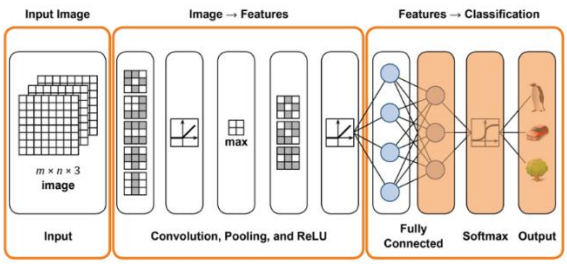


Figura 2.5. Transferencia de aprendizaje en red neuronal convolucional. Reemplaza las capas de clasificación. Fuente: [7]

2.3.1. Redes preentrenadas

Las redes preentrenadas han sido diseñadas y entrenadas por expertos con una colección de imágenes de millones de clases para clasificar miles de objetos tomando semanas de entrenamiento de la red. Para la selección de redes preentrenadas hay que tener en cuenta el coste de oportunidad entre: (i) la precisión de validación (ii) la velocidad de entrenamiento y (iii) el tamaño en disco. [7]

2.3.1.1. AlexNet

AlexNet puede clasificar más de 1000 clases diferentes, tiene 60 millones de parámetros y su estructura presenta cinco capas de convolución, tres capas de agrupamiento y dos capas totalmente conectadas y una capa Softmax. La dimensión de la capa de entrada es 227x227x3 y las primeras capas convierten la imagen con 96 kernels de 11x11x3. Para la reducción del sobreajuste utiliza dos técnicas: data augmentation y dropout. [3]

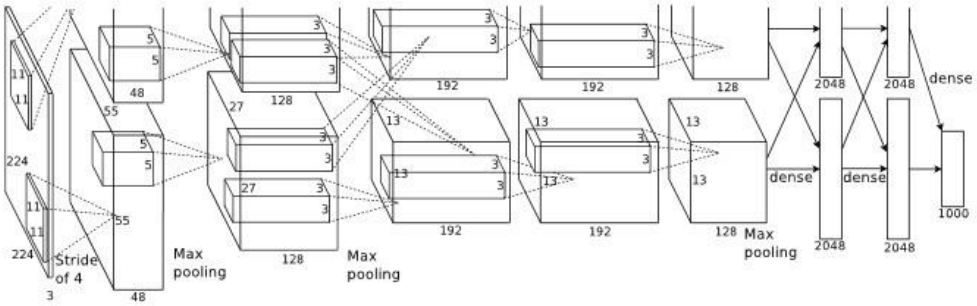


Figura 2.6. Arquitectura red AlexNet

2.3.1.2. GoogLeNet

Tiene una red de 22 capas con 5 millones de parámetros con un tamaño de filtro 1x1, 3x3 y 5x5 para extraer características a varias escalas junto con max pooling. [2]

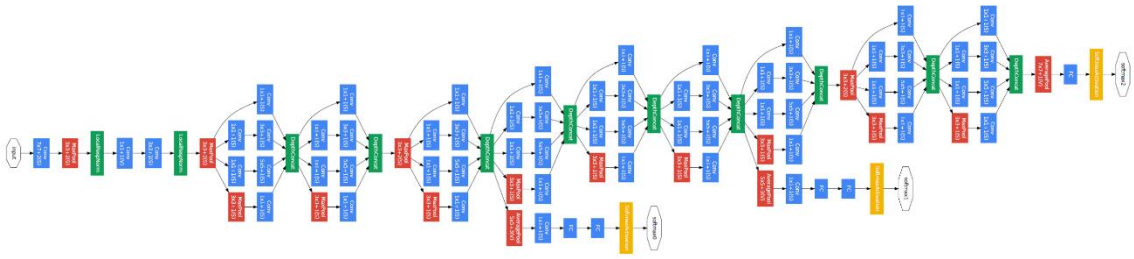


Figura 2.7. Arquitectura red GoogLeNet

2.3.1.3. ResNet-50

ResNet-50 pertenece a la familia de Redes Residuales y tiene 50 capas con 26 millones de parámetros. En las redes residuales aprendemos de los residuos que on sustracciones de características aprendidas por las capas de entrada. ResNet usa una función que conecta la entrada de una n-capa con una (n+x)-capa permitiendo añadir perturbaciones a las imágenes y evitando los problemas de vanishing explode gradient. [2] [3]

[Brain tumor classification in MRI image using convolutional neural network]

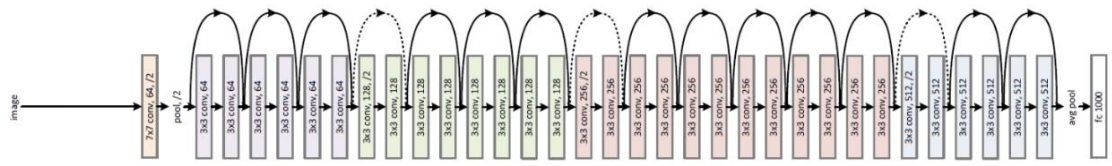


Figura 2.8. Arquitectura red ResNet-50

2.3.2. Conjuntos de preentrenamiento

Los conjuntos ImageNet y Places365 son conjuntos a larga escala, conjuntos de imágenes naturales, y cubren una amplia variedad de imágenes centradas en objetos (ImageNet) y escenarios (Places365).

2.3.2.1. ImageNet

Es una gran base de datos visual diseñada para el uso de la investigación. Contiene más de 20000 categorías con más de 14 millones de imágenes anotadas a mano Las imágenes están organizadas y etiquetadas de forma jerárquica. Desde 2010, existe una competición anual de software, ImageNet Large Scale Visual Recognition Challenge (ILSRVC) donde los programas compiten para clasificar y detectar objetos correctamente. [8]

2.3.2.2. Places365

El conjunto de datos Places está diseñado siguiendo los principios de la cognición visual humana. Su objetivo es construir un núcleo de conocimiento visual, que pueda usarse para entrenar sistemas artificiales con tareas de compresión visual de alto nivel. Su característica principal es su gran volumen de imágenes de escenas a gran escala. Tiene 1.8 millones de imágenes de 365 categorías, donde hay como máximo 5000 imágenes por categoría. [9]

Figura 2.9. Conjuntos de datos de preentrenamiento (**Izquierda**) ImageNet. (**Derecha**) Places365. Fuente: Google Images

2.4. Redes GAN

Una red 'Generative adversarial Network' (GAN) es un tipo de red de deep learning que puede generar datos con características similares a datos de entrada reales. Consiste en el entrenamiento de dos tipos de redes entrenadas simultáneamente para maximizar el rendimiento de ambas.

1. **Generador**.
   A partir de un vector de valores aleatorios como entrada, esta red genera datos con la misma estructura que los datos de entrenamiento con el propósito de engañar a la red discriminadora consiguiendo que clasifique sus imágenes como reales.
2. **Discriminador**.
   Esta red intenta clasificar las observaciones como 'real' o 'falsas' a partir de lotes de datos reales de entrenamiento y datos generados.

Idealmente estas estrategias resultan en un generador que genera datos convincentemente realistas y un discriminador ha aprendido fuertes representaciones de características del conjunto de entrenamiento. [11]

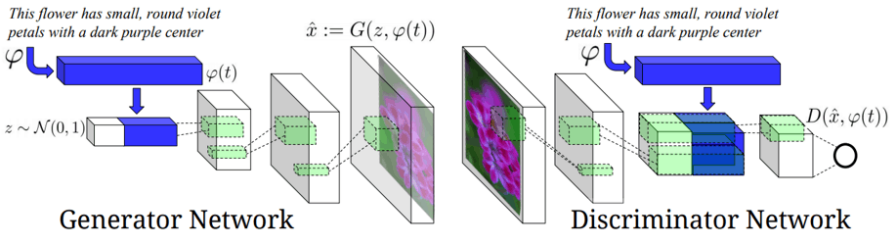


Figura 2.10. Estructura red GAN. Fuente: [12]

2.5. Detección de objetos

La detección de objetos es el proceso de localizar y clasificar objetos en una imagen. Los detectores que vamos a probar son Faster R-CNN y YOLOv2. Tanto los detectores Faster R CNN como YOLO tienen la ventaja de usar anchor boxes para mejorar la velocidad y la eficiencia de detección.

Para entrenar los detectores de objetos, usamos la técnica de transferencia de aprendizaje para aprovechar el conocimiento de extracción de características de las redes entrenadas como clasificadores de nuestro problema.

Durante el entrenamiento se optimiza la pérdida entre las bounding boxes predichas y las reals. La función de coste de estos modelos tiene en cuenta el error de localización, el error de confianza y el error de clasificación: [14]

$$K_1\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}\left[\left(x_i-\bar{x}_i\right)^2+\left(y_i-\bar{y}_i\right)^2\right]+K_1\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}\left[\left(\sqrt{w_i}-\sqrt{\bar{w}_i}\right)^2+\left(\sqrt{h_i}-\sqrt{\bar{h}_i}\right)^2\right]$$
$$+K_2\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}\left(C_i-\bar{C}_i\right)^2+K_3\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{noobj}\left(C_i-\bar{C}_i\right)^2+K_4\sum_{i=0}^{S^2}1_{i}^{obj}\sum_{c\in classes}\left(p_i-\bar{p}_i\right)^2 \quad (3)$$

2.5.1. Anchor Boxes

Los 'anchor boxes' son un conjunto de 'bounding boxes' predefinidos con unas determinadas dimensiones. Estas cajas están definidas para capturar la escapa y la ratio de aspecto de objetos específicos que se quieren detectar y son típicamente elegidos basados en el tamaño de los datos de entrenamiento. [13]

2.5.2. Faster RCNN

El detector Faster R-CNN añade una red de propuesta de regiones para generar propuestas de regiones directamente en la red en vez de usar un algoritmo externo. La red de proposición de regiones usa Anchor boxes. Generar posibles regiones en la red es más rápido y se ajusta más a los datos. [13]

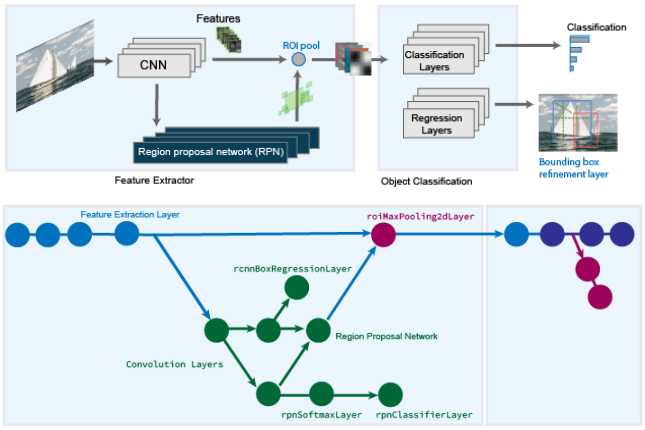


Figura 2.11. Arquitectura Faster R-CNN. Fuente: [13]

2.5.. YOLOv2

You-only-look-once (YOLO) v2 es un detector de objetos que usa una red de detección de una etapa por tanto es más rápido que otros detectores de dos etapas como Faster R-CNN.

El modelo de YOLOv2 ejecuta una CNN en una imagen de entrada para producir predicciones de red . El predictor de red decodifica las predicciones y genera bounding boxes. [13]

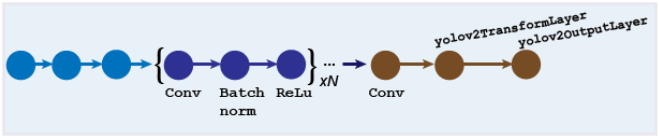


Figura 2.12. Arquitectura YOLOv2. Fuente: [13]

2.5.4. Metricas de detectores de objetos

Para la medición de la precisión en detectores de objetos, 'average precision' es una métrica muy popular. La definición de 'average precision' (AP) es el área bajo la curva de 'precision-recall'. 'Precision' es una métrica que mide el porcentaje de predicciones correctas y 'recall' es una métrica que mide qué tan bien encuentra los positivos (anomalías). Si ordenamos las predicciones según el nivel de confianza, el valor de 'recall' aumenta mientras que 'precision' presenta un patrón característico de zigzag. mAP es la media de valores de AP de cada clase. [15]

$$AP = \int_0^1 p(r)\,dr \tag{4}$$

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{5}$$

2.6. Optimización de modelo

El método de optimización de modelo consiste en el uso de modelos simples e iterar probando diferentes configuraciones de preprocesamiento de imágenes y opciones de entrenamiento. Una vez que se tiene una idea de que configuraciones van bien se puede pasas a redes más precisas. [7]

2.6.1. Balanceo de datos

El problema de desbalanceado de datos ocurre cuando hay una desproporción significativa entre el número de ejemplos de cada clase de un conjunto. Los algoritmos de Machine Learning funcionan mejor cuando el número de datos están equilibrados porque están diseñados para maximizar la precisión y reducir el error. Para solucionar este problema: [16]

1. **Cambio de métricas**.
   En estos casos 'Accuracy' no es un buen estimador y se recomienda el uso de la matriz de confusión, la precisión, el recall y f1-score.
2. **Sobremuestreo**.
   Consiste en igualar los datos de las clases añadiendo más datos de entrada de las clases minoristas. La creación de datos sintéticos para aumentar el tamaño de datos puede realizarse a mano o mediante redes GAN.
3. **Submuestreo**.
   Consiste en eliminar aleatoriamente datos de las clases en mayoría hasta igualar datos. Tiene el inconveniente de eliminar información relevante y puede ser contraproducente.
4. **Capas penalizadoras de clases**
   La capa de clasificación penaliza los resultados según el tamaño de datos de las clases.

2.6.2. Errores de predicción

Existen dos tipos de errores asociados a los modelos de predicción: **sesgo** y **varianza**. Estimar el sesgo y la varianza de una distribución te permite priorizar las técnicas de corrección siguientes.

1. **Sesgo o bias**.
   Error cometido entre la predicción medida de nuestro modelo y el valor real que tratamos de predecir. Un sesgo alto ignora las relaciones importantes entre las variables y presenta un rendimiento pobre sobre el conjunto de entrenamiento. [17]
2. **Varianza**.
   Variabilidad de predicción que presenta nuestro modelo. Una varianza alta se caracteriza por un sobreajuste del conjunto de entrenamiento y una mala generalización del conjunto de test y presenta una amplia diferencia de error entre el rendimiento de entre namiento y de validación suponiendo la misma distribución. [17]

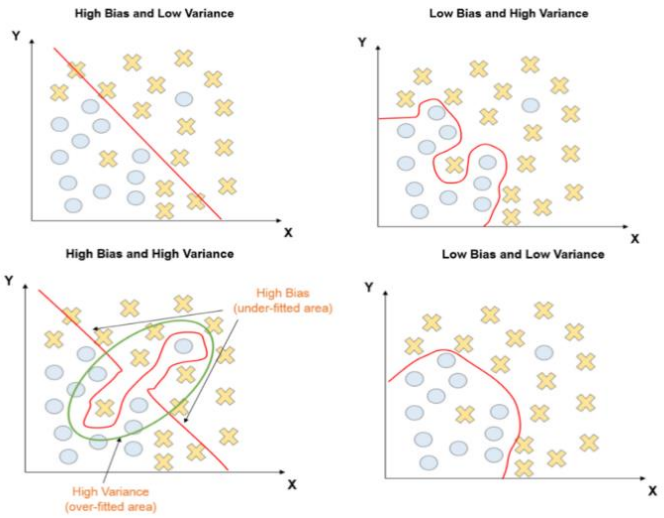


Figura 2.13. Bias Variance: (**a**) High bias and low variance; (**b**) Low bias and high variance; (**c**) High bias and high variance; (**d**) Low bias and low variance. Fuente: [18]

Para conseguir un sesgo bajo y varianza baja hay que tener en cuenta el coste entre ambos. Antiguamente no había muchas herramientas que aumentaban uno sin perjudicar al otro. Pero en la era moderna entrenar redes más grandes siempre reduce el bias sin afectar a la varianza aplicando regularización. Para reducir la varianza conseguir nuevos datos es la mejor manera. [18]

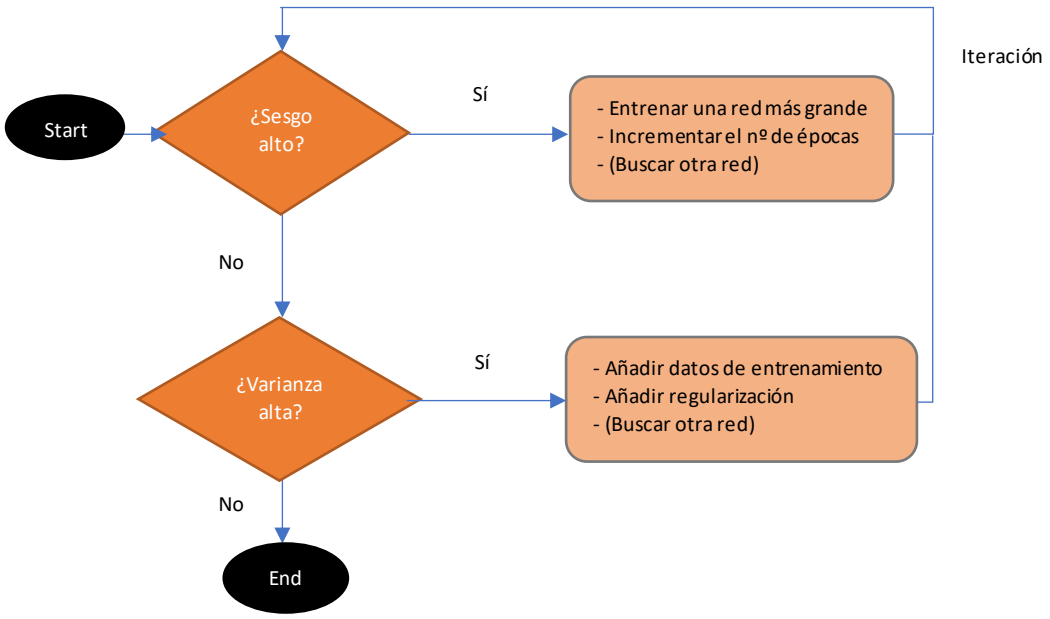


Figura 2.14. Proceso de corrección frente al conflicto bias-variance. Fuente: [18].

2.6.3. Reducción de sobreajuste

2.5.3.1. Data augmentation

Es una técnica que consiste en aplicar un conjunto de operaciones de transformación sobre las imágenes con el objetivo de aumentar el tamaño de los datos o mejorar la generalización del modelo. Las operaciones pueden ser transformaciones espaciales como rotaciones, volteos, desplazamientos deformaciones en la imagen como cizalladuras, reescalamientos, recortes, cambios en el color y ruido de la imagen. [19]

Está demostrado que da buenos resultados y se basa en que las transformaciones menores son interpretadas como imágenes diferentes según la red neuronal. Puede ser de dos tipos: [20]

1. **Offline data augmentation**
   Crea nuevas imágenes a partir de transformaciones y las almacena en el disco. Es deseable para aumentar el tamaño de los datos en conjuntos pequeños.
2. **Online data augmentation**
   Realiza operaciones de transformación aleatorias al principio de cada minibatch de entrenamiento. Mejora la capacidad de generalización del modelo sin aumentar su tamaño.

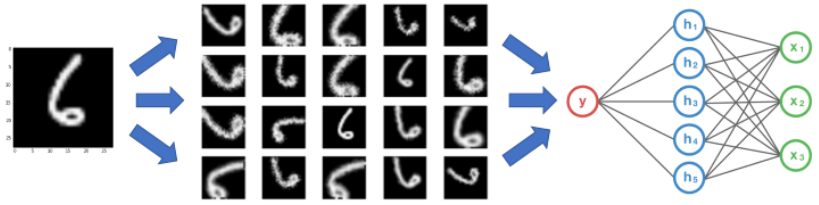


Figura 2.15. Proceso de aplicación de data augmentation. Fuente: Google Images

2.5.3.2. Regularización

La regularización es una herramienta de reducción del sobreajuste que penaliza los parámetros con valores grandes. Elimina datos espurios causando valores dentro del rango lineal de la función de activación tanh. La función pasa a ser relativamente lineal y reduce la no linealidad de la función de coste reduciendo el sobreajuste. La regularización L2 se controla con el valor λ de acuerdo con la fórmula. [18]

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \cdot \log\left(h_\theta(x^{(i)})\right)_k + \left(1 - y_k^{(i)}\right) \cdot \log\left(1 - \left(h_\theta(x^{(i)})\right)\right)_k\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(\theta_{ji}^{(l)}\right)^2 \quad (6)$$

2.5.3.3. Dropout

El dropout es una técnica de reducción del sobreajuste que asigna neuronas a valor nulo según un valor probabilístico. De esta forma se disminuye el número de parámetros en la función de coste y la probabilidad de sobreajuste.

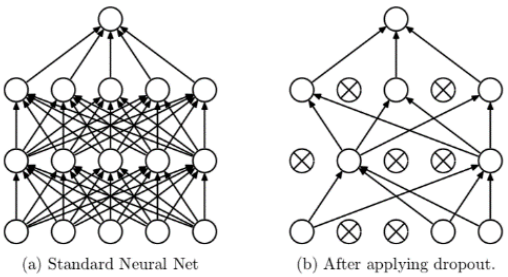


Figura 2.16. Aplicación de dropout: (**a**) Red neuronal estándar (**b**) Red después de dropout. Fuente: [18]

2.6. Opciones de entrenamiento

2.6.1. Hiperparámetros

Los hiperparámetros son los parámetros que optimizan la optimización del modelo. La influencia de los hiperparámetros en el resultado es subjetiva, según Andrew Ng, podemos afirmar que el orden de importancia es (1) learning rate (2) la caída de learning rate y el número de capas y (3) el parámetro del optimizador, número de capas ocultas y tamaño del minibatch. Mediante la selección de redes de estudio fijamos el número de capas y optimizamos el resto de las variables. [18]

2.6.1.1. Tasa de aprendizaje

Es el hiperparámetro que controla el paso de corrección del gradiente. Para valores pequeños el camino hasta el mínimo optimo será un proceso lento pero seguro. Para valores altos, sobrepasa el mínimo optimo y no converge. El valor de esta variable puede configurarse para cambiarse. La caída de la tasa de aprendizaje es el parámetro que disminuye su valor en un número de épocas definido, (periodo de caída).

2.6.1.2. Tamaño del minibatch

En mini batch gradient descent, el conjunto total de entrenamiento se divide en porciones del conjunto llamados minibatches. El número de datos que se utilizan para el cálculo del gradiente. Permite asegurarse que la red neuronal está ejecutándose eficientemente. Si es demasiado grande puede superar la capacidad de almacenamiento y reducir la velocidad drásticamente. Es generalizado su uso en valores de potencias de dos porque aceleran el cálculo [18].

2.6.1.3. Parámetro del optimizador

El descenso de gradiente estocástico es propuesto por Herbert y Sutton. En resumen, para el cálculo tomamos las derivadas de los pesos, *dW*, y la derivada del sesgo, *db* para cada época y los multiplicamos por la tasa de aprendizaje. [2]

$$W = W - \eta \times \mathrm{d}W \tag{7}$$
$$b = b - \eta \times \mathrm{d}b \tag{8}$$

2.6.4.3.1. Descenso del gradiente con momento

Mientras que el descenso del gradiente estocástico con momento β es la media móvil de los gradientes, aquí es la media móvil entre 0 y 1 cuando calculamos *dW* y *db*. [2]

$$V_{dW} = \beta \times V_{dW} + (1 - \beta) \times \mathrm{d}W \tag{9}$$
$$V_{db} = \beta \times V_{db} + (1 - \beta) \times \mathrm{d}b \tag{10}$$

2.6.4.3.2. RMSProp

Similarmente, Root Mean Squared Prop es una tasa de aprendizaje adaptiva presentada por Geoff Hinton. RMSprop toma la raíz de la media móvil de los gradientes. β es el hiperparámetro que controla la media ponderada exponencialmente. [2]

$$S_{dW} = \beta \times S_{dW} + (1 - \beta) \times \mathrm{d}W^2 \tag{11}$$
$$S_{db} = \beta \times S_{db} + (1 - \beta) \times \mathrm{d}b^2 \tag{12}$$

2.6.4.3.3. Adam

Combinando las características de la media ponderada de los gradientes anteriores y el cuadrado de la media ponderada de los gradientes anteriores implementamos la técnica Adam. Épsilon es un numero pequeño que previene la división por cero η es una tasa de aprendizaje con diferente rango de valores. [2]

$$W = W - \eta \times \left(V_{dw} / \sqrt{S_{dw} + \varepsilon} \right) \tag{13}$$

$$b = b - \eta \times \left(V_{db} / \sqrt{S_{db} + \varepsilon} \right) \tag{14}$$

2.6.4.4. Estrategias de optimización

En un hiperespacio de millones de parámetros es difícil conocer que parámetros funciona mejor para cada modelo. Antiguamente la búsqueda de mejores variables se realizaba mediante la búsqueda en rejilla, asigna valores fijos a las variables independientes y calcula el resultado de la variable dependiente. El muestreo aleatorio asignar valores aleatorios a las variables independientes en cada experimento. De este método, se barrer más espacio dentro del hiperespacio del modelo. La optimización bayesiana tiene en cuenta los resultados de experimentos anteriores para elaborar una región de probabilidad de valores óptimos acorde a la fórmula. [18]

$$\mathbb{P}\left(F_n(X) \middle| X_n \right) = \frac{e^{-\frac{1}{2} \cdot F_n^T \varepsilon_n^{-1} F_n}}{\sqrt{(2 \cdot \pi)^n \cdot |\varepsilon_n|}} \tag{15}$$

3. Estudio de la pieza

3.1. Introducción

Desde los años 70, el plástico llegó a las zonas más significativas del automóvil debido a su ligereza y aumento de resistencia y seguridad. En el revestimiento interior de los automóviles juega un papel muy importante debido a su facilidad de diseño de formas complicadas y estética interior.

En los acabados interiores, el uso de termoplásticos transformados por inyección es ampliamente utilizado debido a su buen acabado superficial a un coste razonable. Las partes estructurales que requieren de buena rigidez y comportamiento se conforman mediante termoestables reforzados con fibra. [21]

3.1.1. Descripción de la pieza

El panel de instrumentos o salpicadero es el dispositivo que se encuentra frente al conductor y semiconductor y alberga los instrumentos e indicadores que permiten el funcionamiento del vehículo.

En la planta de CEFA, se producen varios modelos de paneles de instrumentos. La pieza de estudio es un panel de instrumentos o salpicadero perteneciente al modelo OPEL Corsa (6ª Gen). El salpicadero es de tipo flexible, este tipo de salpicaderos están compuestos por varias capas: [22]

1- capa superior rugosa (huella) hecha típicamente de PVC plastificado
2- capa intermedia hecha de poliestireno o poliuretano expandido
3- capa base metálica, de plástico o aglomerado.

El alcance de este trabajo se centra en la inspección y detección de los defectos sobre la capa superior rugosa también denominada tela.

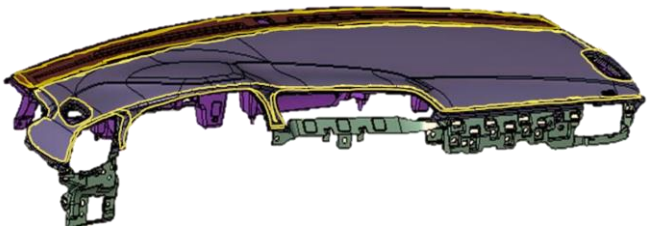


Figura 3.1.1. Panel de instrumentos Opel Corsa. Tela (contorno amarillo). **Fuente**: CEFA S.A.

3.1.2. Proceso de producción de la pieza

El proceso de producción de la tela parte de una lámina semielaborada de termoplástico. El material en forma de rollo es arrastrado mediante tetones a una máquina de termoconformado. El proceso de termoconformado le confiere las curvaturas finales de la pieza además del dibujo de la huella. Se suele se suele realizar tratamientos superficiales y adición de aditivos previa la incorporación de espuma.

La tela moldeada y la base de plástico se disponen en cunas macho y hembra. La cuna tiene las condiciones de temperatura y presión apropiadas para la reacción de expansión del isocianato y el poliol en poliuretano. La espuma se adapta a la forma de la tela moldeada y la correcta distribución de espuma se realiza por medio de una cámara de visión térmica.

La pieza es refrigerada por convección natural a su transporte por gravedad en unos ganchos. En una cuna, se troquela el excedente de pieza y la zona del airbag del salpicadero es debilitada por medio de una cortadora láser de precisión. La pieza de salpicadero es montada en la estructura general del panel de instrumentos. Y en los sucesivos pasos recibe la instalación del rsto componentes y dispositivo del airbag.

En su etapa final, la pieza se monta en un sistema de AGVs donde recibe montaje de los últimos componentes. Al final, recibe la inspección manual de la superficie. Las piezas OK son empaquetadas y enviadas al almacén. Mientras que las piezas detectadas como NOK son separadas en un puesto de retrabajo.

En la misma línea se producen piezas en ambos sentidos de conducción, izquierda y derecha, aunque esta última en menor proporción.

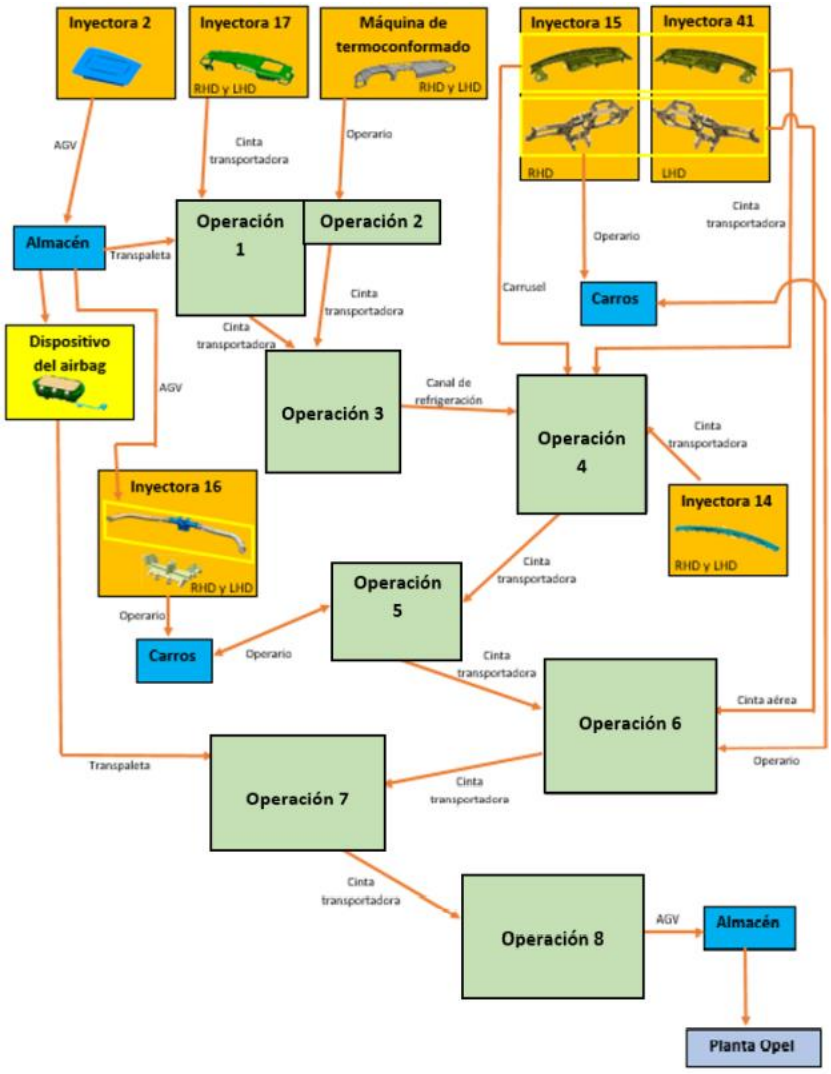


Figura 3.1.2. Proceso de producción del Opel Corsa. **Fuente**: 'Estudio de la viabilidad técnica y económica de la automatización del suministro logístico de productos en Módulos Ribera Alta SL'. Báguena Gutiérrez.

3.1. 3. Descripción de los defectos

Los tipos de defectos más importantes son:

1- **Raya**. Defecto superficial con forma de surco producidas por el contacto con un objeto con filo.
2- **Roces**. Defecto superficial que presenta rotura de la capa superior en forma de levantamiento irregular y discontinuo de la superficie de tela. Está producida por el contacto por fricción con otros objetos.
3- **Bollos**. Defectos que producen zonas elevadas sobre la superficie, generalmente asociada a variaciones en las condiciones térmicas. Se presenta como una forma abombada o como una distorsión dimensional. Las causas más comunes son la presión de los gases dentro de la parte en expansión o por esfuerzos residuales internos.
4- **Hundimientos.** Defectos que producen una zona inferior respecto a la superficie. Puede ocurrir que el rechupe aparezca una vez se ha extraído la pieza, esto es debido a que al expulsar el calor de la pieza se crea un estado tensional que se traduce en contracciones. Las incrustaciones son imperfecciones debido a la presencia de partículas en el molde. [23]
5- **Arrugas en las esquinas.** Imperfecciones que tiene la apariencia de una onda moldeada sobre la capa superior que se localizan en las esquinas de la pieza.
6- **Manchas.** Defectos presentan comportamientos visuales diferentes en ciertas regiones. Los defectos más conocidos son los de decoloración, cambios en el color original a menudo causada por sobrecalentamiento exposición a la luz o ataques químicos. El defecto de brillo se denomina a cuando la región pierde 'gloss' o grado de brillo. [24]
7- **Gap.** defecto dimensional para referirnos a las desuniformidades en las juntas entre piezas.

3.1.4. Necesidades de dataset

El conjunto de imágenes (data set) debe de satisfacer las siguientes necesidades:

- Representatividad de la población a detectar
- Diferenciación de los parámetros de defectos
- Número mínimo de imágenes de una misma clase

(a)

(b)

(c)

(d) (e)

(f) (g)

(h) (i)

Figura 3.1.3. Defectos de producción más comunes en la superficie del salpicadero: (**a**) rayas; (**b**) roces; (**c**) bollos; (**d**) hendiduras; (**e**) incrustaciones; (**f**) manchas; (**g**) defectos de brillos; (**h**) arrugas en las esquinas; (**i**) gap. **Fuente**: Propia.

3.1.5. Distribución de defectos

La aparición de defectos es ocasionada por la variación de las condiciones de operación de la pieza. Los defectos poseen una alta variabilidad de tamaño, forma y tipo. Según los datos de la figura 3.1.5., las esquinas son las zonas de la pieza con mayor probabilidad de aparición de defecto, siendo más grave la zona del conductor. En ambas zonas se localizan principalmente los defectos de arrugas, el cual es el defecto más probable por zona.

Los datos recogidos sobre la frecuencia de defectos corresponden al periodo entre 09/2020 - 02/2021.

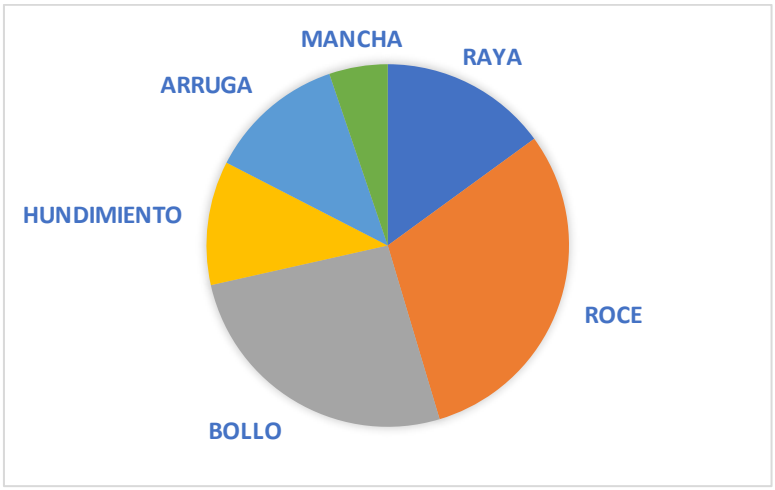


Figura 3.1.4. Distribución de defectos por tipo. Fuente: CEFA S.A.

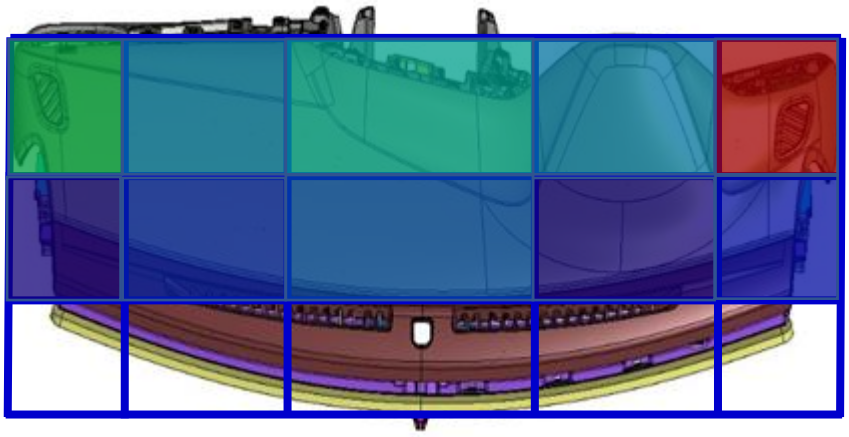


Figura 3.1.5. Mapa de distribución de defectos por zonas. Fuente: CEFA S.A.

3.2. Estudio de las condiciones de captación

Los defectos presentan una alta variabilidad de sus características. El objetivo de este estudio es maximizar la probabilidad de visualizar los defectos con un sistema de detección fijo.

3.2.1. Diseño de experimentos

Con el propósito de seleccionar las mejores condiciones del sistema de captura se plantea el diseño de un banco de experimentos que reproduzca las condiciones del sistema de captura en el entorno deseado.

El modelo teórico ideal del banco de ensayos consistiría en un sistema de infinitas cámaras y sistemas de iluminación ocupando todos los puntos del espacio analizando los resultados individualmente. En el modelo real aproximado escogemos ciertos puntos estratégicos de estudio que nos permiten generalizar el aprendizaje.

3.2.2. Subsistema de adquisición

El estudio de la posición de la cámara se divide en el estudio de 5 ángulos horizontales [22.5º 45º 67.5º 90º] y 4 ángulos verticales [0º 15º 30º 45º] dispuestas en forma de semiabanico. La distancia de la cámara es fija, 1050 mm. El modo de captura es secuencial con una tasa de 10 [frames/seg].

3.2.3. Subsistema de iluminación

La imagen deberá tener una iluminación tal que permita iluminar nítidamente el defecto sin saturar la imagen. Se debe ajustar la distancia del foco a la superficie del defecto junto con el obturador analógico y digital. Un nivel de exposición optimo es 1200-1350. El ángulo de incidencia de la luz debe ser el mismo ángulo de incidencia de la cámara.

3.2.4. Calibración de la cámara

La calibración se realiza con la pieza situada en la posición de referencia y el punto deseado en el plano de la imagen. Se ajusta el enfoque de imagen con la ayuda de un mapa de enfoque. El ajuste del nivel de exposición mediante el obturador analógico.

3.2.5. Descripción de las pruebas

Las piezas de estos experimentos son piezas detectados en planta como NOK no recuperadas. La pieza se arrastra sobre un carro sobre la línea continua a una velocidad aproximada de 3 [m/s]. Se realizan una serie de estudios con el objetivo de sacar conclusiones de cara a encontrar un rango de valores óptimos para el ángulo horizontal y vertical de la cámara.

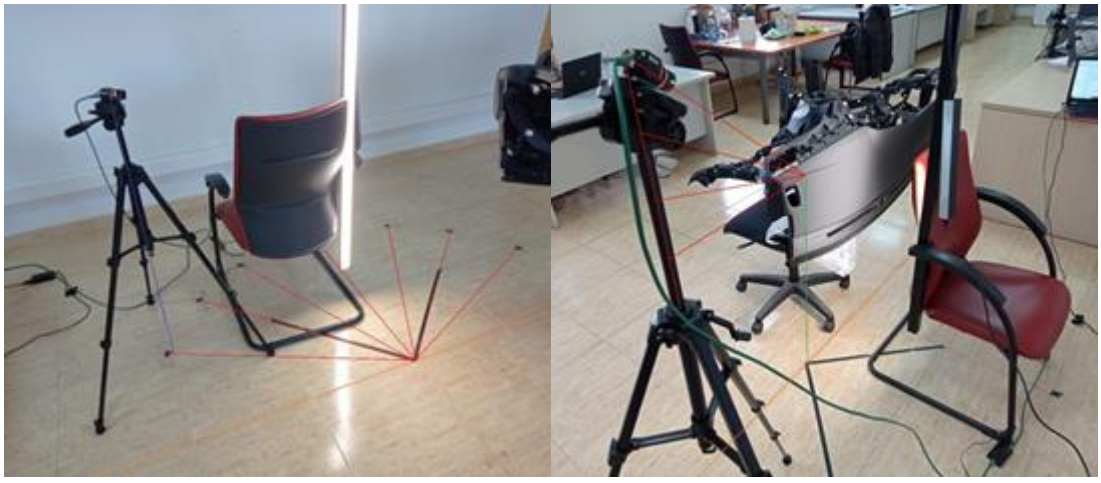


Figura 3.2.1. Configuración de banco de ensayos: (**a**) ángulos horizontales; (**b**) ángulos verticales. Fuente: Propia.

3.3. Resultados

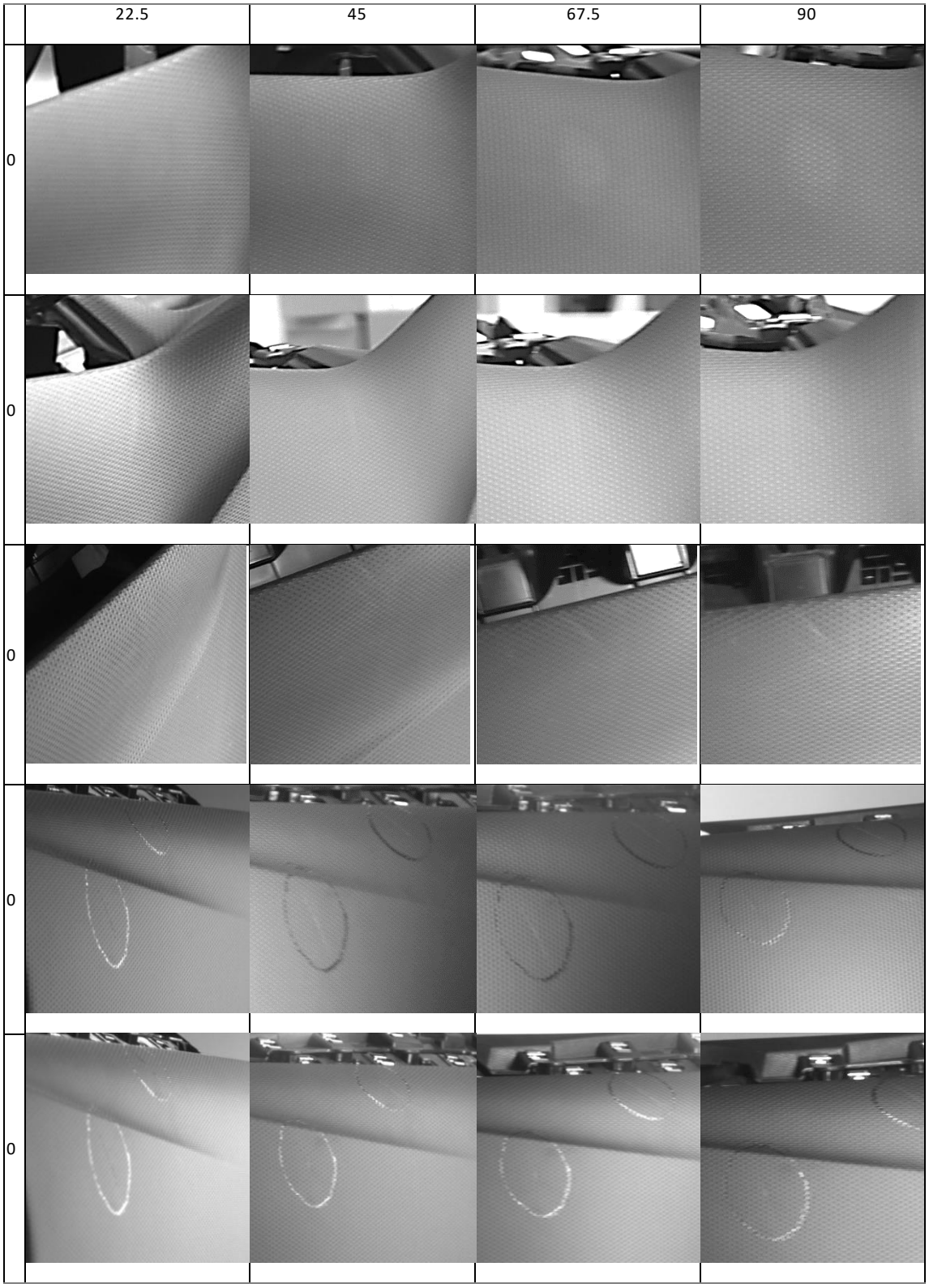


Figura 3.2.2. Imágenes de pruebas de defectos captadas en banco de ensayos: (**fila 1**) mancha; (**fila 2**) defecto de brillo; (**fila 3**) hendidura; (**fila 4**) raya oblicua sin difusor de luz; (**fila 5**) raya oblicua con difusor de luz. Fuente: Propia.

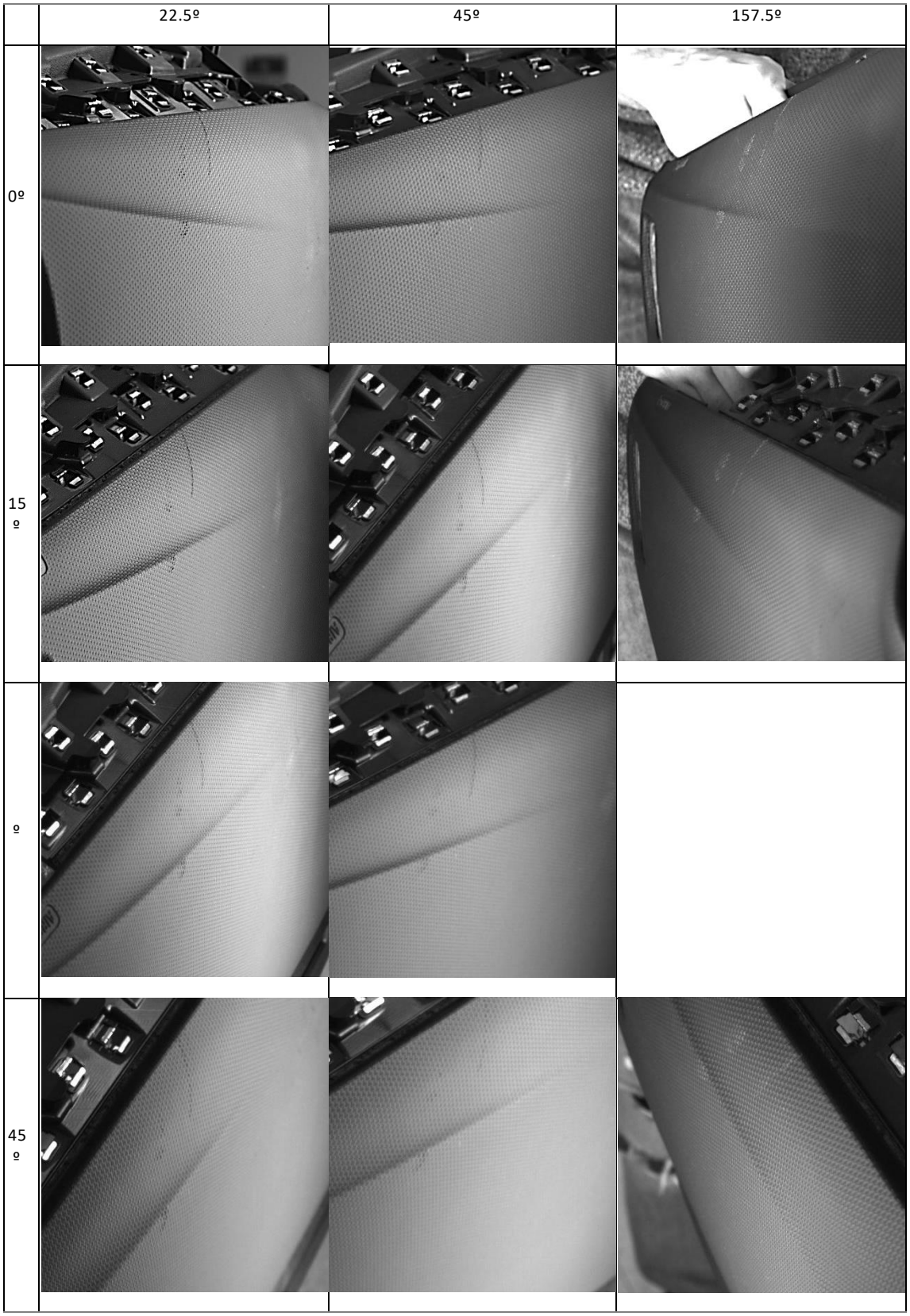


Figura 3.2.3. Imágenes de pruebas de defecto de raya vertical captada en banco de ensayos. Fuente: Propia.

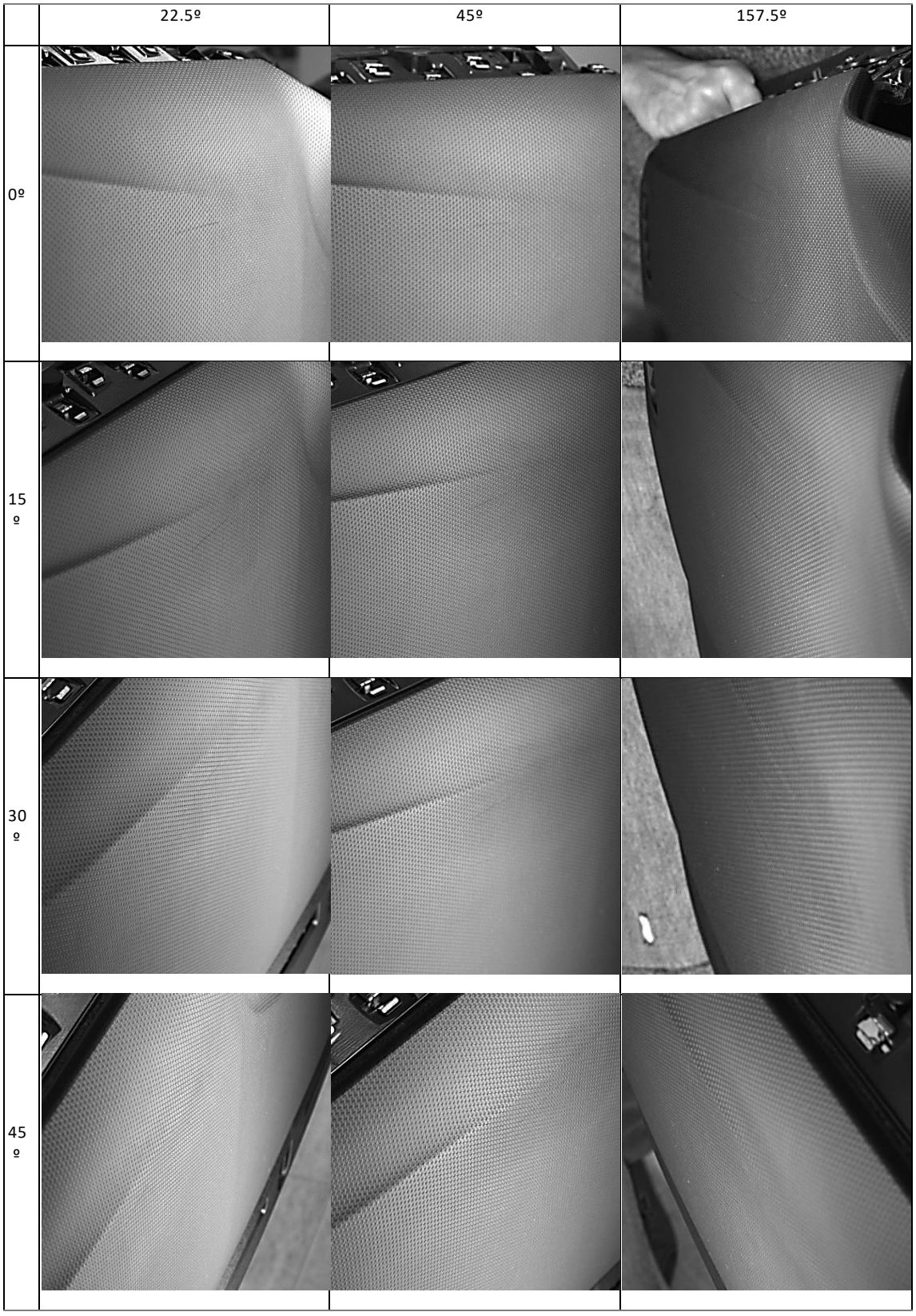


Figura 3.2.4. Imágenes de pruebas de defecto de raya horizontal captada en banco de ensayos. Fuente: Propia.

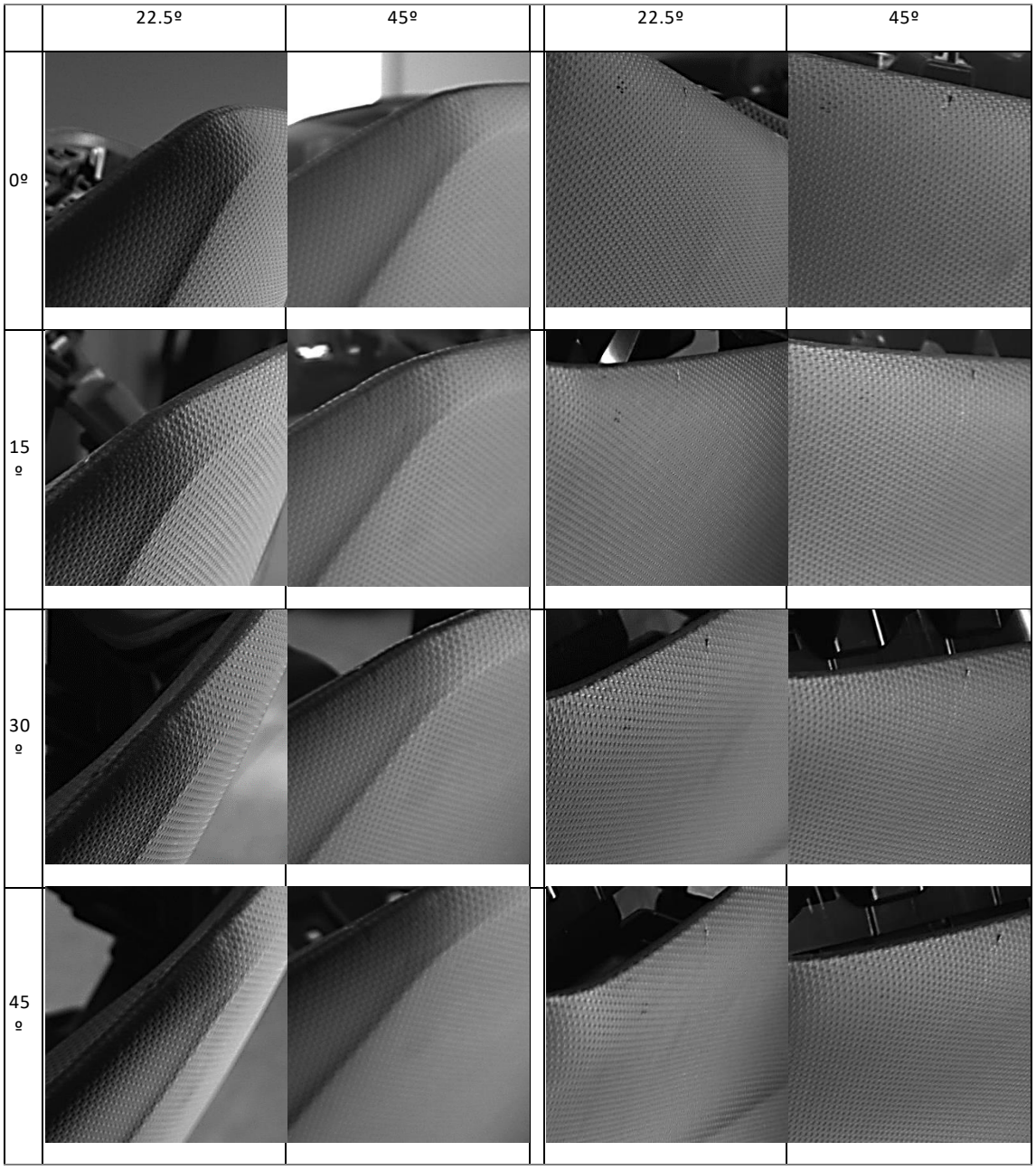


Figura 3.2.5. Imágenes de pruebas de defectos de (**col. 1 y 2**) rechupe; (**col. 3 y 4**) roce en banco de ensayos. Fuente: Propia.

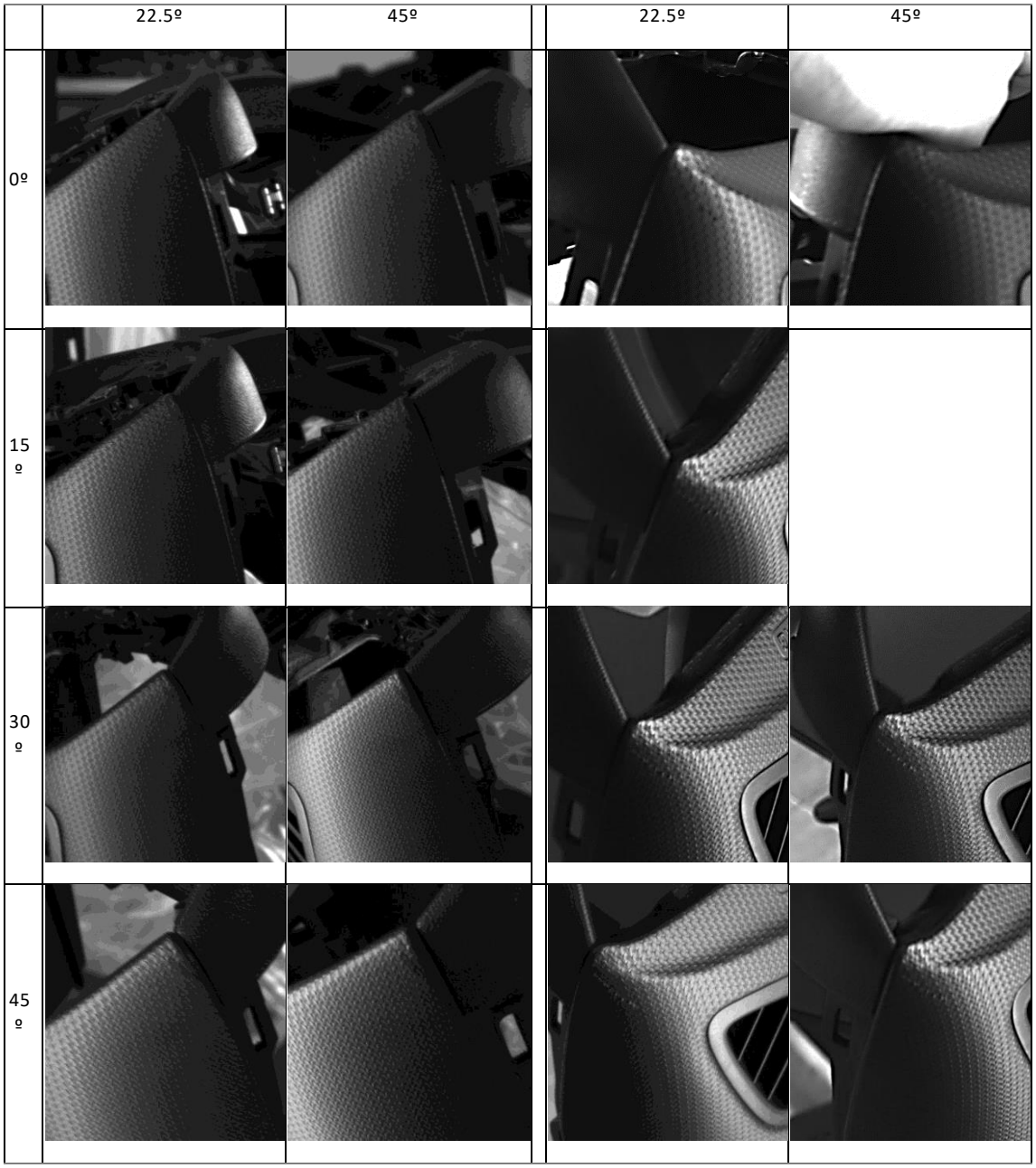


Figura 3.2.6. Imágenes de pruebas defectos de (**col. 1 y 2**) arruga en esquina; (**col. 3 y 4**) roce en esquina captadas en banco de ensayos. Fuente: Propia.

3.4. Conclusiones

La luz difusa permite una iluminación más uniforme (**Figura 3.2.2.5.**) que es buena para detectar ciertos tipos de defectos, aunque no se ha considerado necesario su uso porque la diferencia es no es sustancial y requiere de posiciones muy cortas del foco o luminarias mayor potencia.

Para una determinada posición de la pieza, el defecto presenta un punto ciego donde no crea sombra (**Figura 3.3.1**). Esta posición depende de la geometría y superficie del defecto. Para evitar la ausencia de defectos visibles dentro del plano imagen es recomendable aumentar el ángulo de incidencia.

A medida que aumenta el ángulo de incidencia, disminuye la reflexión de luz difusa y la luz reflejada es la componente especular. Como observación general, este efecto permite diferenciar nítidamente la presencia de defectos muy pequeños en superficies planas. Sin embargo, es contraproducente en algunos casos, ya que depende del tipo de defecto y su geometría. Cuanto más inclinado es el ángulo con la pieza disminuye la visibilidad de las dimensiones perpendiculares al haz de visión (**Figura 3.3.2.**) y ciertos tipos de defectos como las decoloraciones son peor detectadas (**Figura 3.2.3.1**).

Teniendo en cuenta los datos anteriores, los ángulos de Cámara de (45º, 30º) se ha demostrado que presenta buenos resultados generales.

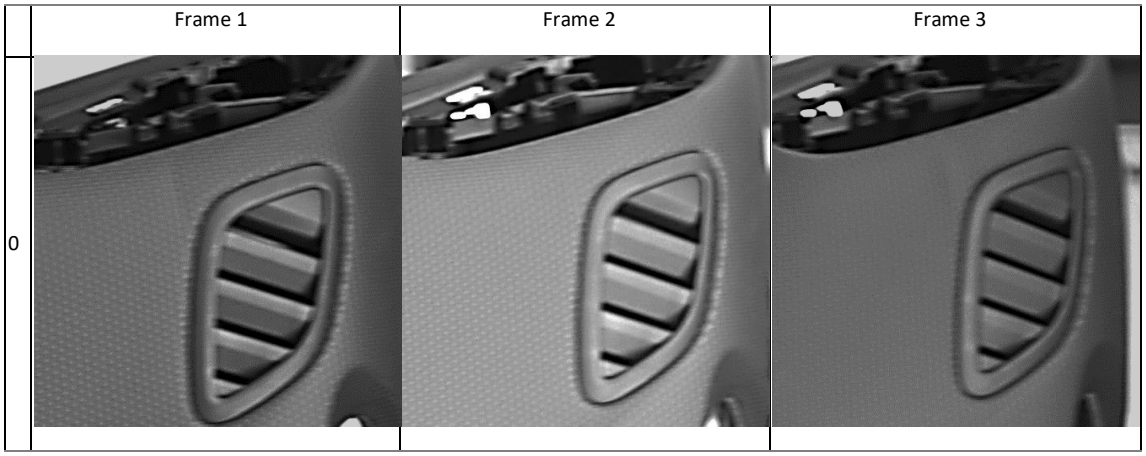


Figura 3.3.1. Efecto de punto ciego

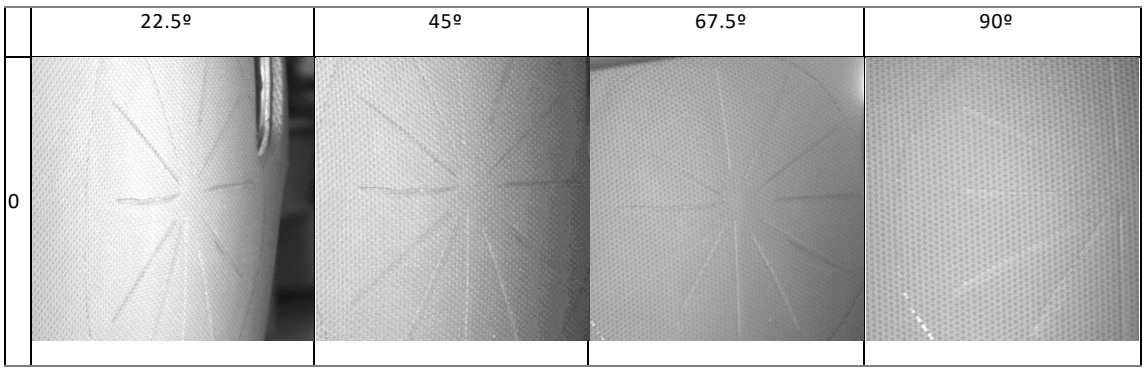


Figura 3.3.2. Efecto del reflejo especular.

4. Estudio de la localización del sistema de captura

4.1. Introducción

La localización del sistema de captura es un espacio de la línea dentro de la etapa de producto acabado de la pieza donde se plantea la captura masiva de imágenes de defectos válidas. Las mejores condiciones de captura no siempre son posible transferirlas al sistema real debido a las restricciones del entorno.

El propósito de este apartado es seleccionar la configuración del sistema de captación que permitan capturar la mayor cantidad de imágenes válidas del mismo tipo de defecto. Para ello vamos a tener en cuenta:

- Condiciones de captación de defectos
- Distribución de defectos por región
- Distribución de defectos por tipo

En la planta, la línea de producción presenta un espacio situado entre la finalización de componentes que cumple con los requisitos necesarios donde las tres configuraciones posibles son:

(A) **Suelo**. Configuración de captura de pruebas inicial. No permite ángulos verticales ni ángulos horizontales mayores de 45º.

(B) **Estantería**. Restricciones de movimiento para el trípode y ángulos delimitados por la posición del foco (22.5º, 45º). La luz incide sobre la zona central.

(C) **Entrada a la cabina**. Escasas restricciones de movimiento para el trípode, permite varias configuraciones. Posibilidad de recibir vibraciones de puerta. La luz incide sobre la esquina.

La principal diferencia entre B y C es la posición del foco de luz. El foco de luz tiene un espacio limitado y está obligado a adoptar una posición en cada configuración para iluminar correctamente la pieza.

Figura 4.1. Selección de diferentes posiciones de cámara: (**A**) suelo; (**B**) estantería; **C**) puerta de cabina. Fuente: Propia.

4.2. Resultados

Resultados configuración A (45º, 0º):

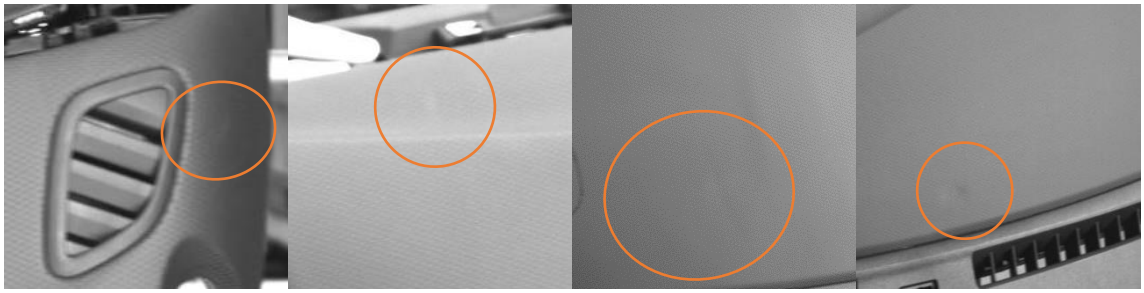


Figura 4.2.1. Imágenes de defectos capturados desde la configuración A: (**a**) roce; (**b**) mancha; (**c**) hendidura-1; (**d**) hendidura-2. Fuente: Propia.

Resultados configuración B (22.5º, 45º):

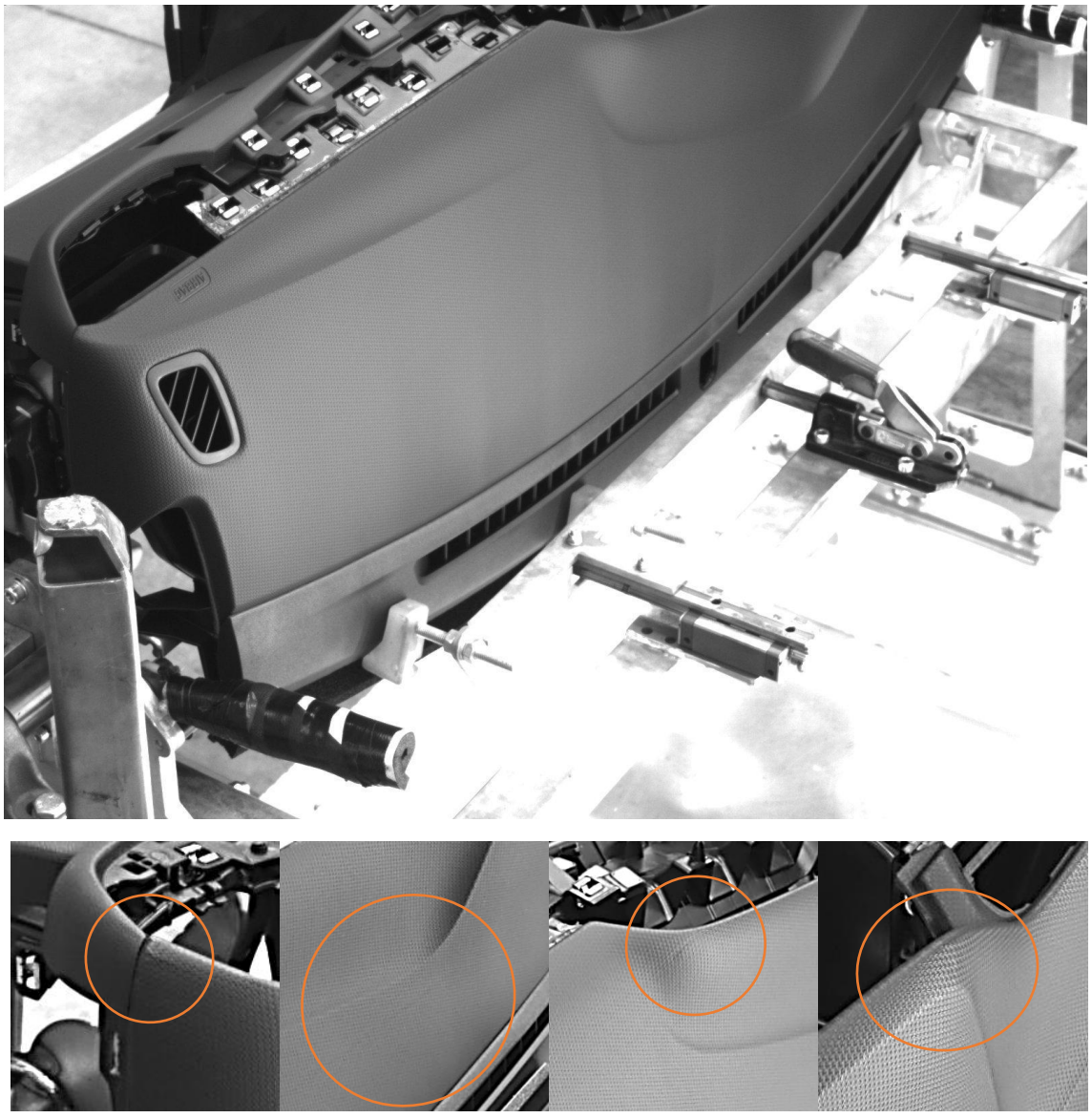


Figura 4.2.2. Imágenes de defectos capturados desde la configuración B: (**a**) arruga; (**b**) raya-1; (**c**) raya-2; (**d**) bollo. Fuente Propia.

Resultados configuración C (30º, 45º):

Figura 4.2.3. Imágenes de pieza y defectos: (**a**) arruga; (**b**) raya-1; (**c**) raya-2; (**d**) gap; capturados con la configuración C. Fuente: Propia.

4.3. Conclusiones

La escasa versatilidad de la configuración A le somete a caer en pobres resultados de visualización. La configuración B, es una posición ideal para capturar defectos en la zona central, aunque no recibe iluminación en las esquinas, es una buena opción para defectos de rayas y roces. La configuración C, es una posición ideal para capturar defectos en la zona de la esquina. Además, la libertad de movimiento permite ajustar el ángulo para optimizar la detección sobre un tipo de defecto específico.

Teniendo en cuenta la información de distribución de defectos por zonas nos vamos a centrar en la captura de arrugas en las esquinas con la configuración C.

4.4. Ajuste de la configuración

Seleccionada la configuración, es necesario ajustar la imagen a las condiciones deseadas. La lente es el dispositivo encargado de adecuar los rayos de luz a la imagen que recibe el sensor fotográfico. El principal parámetro de la lente es la distancia focal, que actúa como un zoom sobre la imagen.

Ajustamos la imagen de forma que la luz permita visualizar el defecto deseado nítidamente. La lente de f = 50 mm permite el rango de enfoque dentro del rango óptimo de captura (Fig. 12)

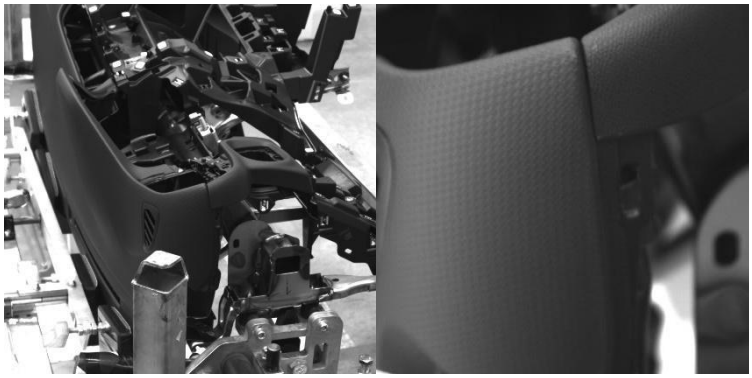


Figura 4.4. imágenes de selección de lente capturados con la configuración C: (**a**) f = 18 mm; (**b**) f = 50 mm; Fuente: Propia.

4.5. Adquisición de imágenes

El proceso de adquisición de imágenes ha consistido en la captura de secuencias de las imágenes de las piezas tanto de conducción izquierdas como derechas en las opciones especificadas en el apartado anterior para la creación del conjunto de datos.

La captura se ha realizado durante intervalos regulares diarios. Las imágenes presentan ligeras diferencias de posición de cámara y en algunos casos hay grandes variaciones de iluminación al no estar controlada la iluminación externa. El etiquetado de imágenes se ha realizado según el criterio de los empleados de la planta y posteriormente se ha procesado para la creación del conjunto de datos.

La captura se ha configurado con una tasa de captura de 5 frames/seg para una velocidad de pieza aproximada de 1.6 m/s. Un ajuste del nivel de exposición EV = 1200 - 1350. El tamaño de imágenes = 1600 x 1200 pxls.

5. Estudio de clasificación de imágenes

5.1. Introducción

El objetivo de este estudio es seleccionar el mejor modelo de clasificación de las imágenes con defectos. Para ello vamos a probar diferentes redes con diferentes opciones de entrenamiento y escoger aquellas que den un mejor rendimiento en la validación y testeo.

5.1.2. Conjunto de imágenes

Para la creación del conjunto de imágenes se han seleccionado aquellos tipos de defectos que presentan un número suficiente de imágenes válidas.

1. Arruga
Dentro de la categoría arruga se incluyen aquellas imperfecciones localizadas en la esquina superior. Con forma de uña, bulto, pliegues de la tela, ondas moldeadas, etc. Además de los defectos con características semejantes, pero no consideradas críticas por sus dimensiones menores (**Fig 4.1.c**.).

2. Gap
Esta categoría incluye el defecto de gap conocido como las desviaciones en la alineación entre contornos de las piezas. El defecto de gap no es considerado un defecto superficial, por ende, no se tienen datos. Sin embargo, se ha observado una alta probabilidad de ocurrencia de este defecto.

3. Suciedad
En esta categoría se han incluido las anormalidades de tonalidades y brillos en la superficie. Ocasionadas por manchas de grasa, polvo, líquidos…

4. OK
Clase con ausencia de defectos. Se dispone de un gran volumen de imágenes de piezas correctas. Se han escogido piezas contiguas a las defectuosas para homogeneizar las condiciones.

El **conjunto de entrenamiento** es el conjunto utilizado para entrenar el modelo. El **conjunto de validación** es el conjunto para evaluar la capacidad de generalización de la población y comprobación del correcto funcionamiento de la red. El conjunto de entrenamiento y validación han sido divididos en una proporción 0.8/0.2. El **conjunto de test** es un conjunto con la finalidad de medir el rendimiento del modelo con imágenes de defectos nunca antes vistas durante el entrenamiento ni validación.

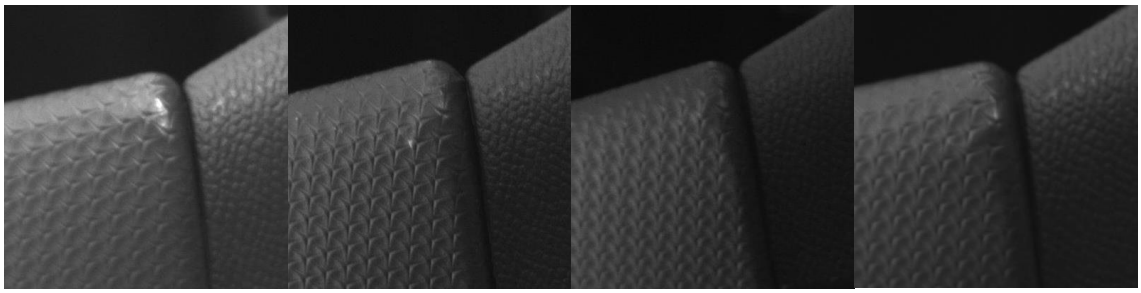


Figura 5.1. Diferentes tipos de formas de defectos de arruga usados en el conjunto de datos: (**a**) bulto crítico; (**b**) pliegue crítico; (**c**) pliegue no crítico; (**d**) uña no critica. Fuente: Propia.

5.1.3. Preprocesamiento de imágenes

Para aplicar la técnica de transferencia de aprendizaje con redes preentrenadas, ha sido necesario (1) redimensionar el tamaño de las imágenes al tamaño de la capa de entrada de la red y (2) convertir los canales de imagen de grises a RGB.

5.1.4. Data augmentation

Se ha empleado dos tipos de data augmentation: (1) balanceo de datos (2) transformaciones aleatorias de imagen para mejorar la generalización. Para balancear los datos, el conjunto de entrenamiento la adición de volteos horizontales, verticales y ambos sobre las clases minoristas. En los datos de validación y testeo únicamente volteos horizontales para mantener la condición de representatividad del sistema de detección real.

Las operaciones de transformación de imagen online se asignan valores aleatorios dentro de un rango (**Tabla 5.1.3.**) durante cada minibatch de entrenamiento.

Tabla 5.1.1. Conjunto de datos base

	Arrugas	Gap	Ok	Suciedad	Total
Training set	220	246	942	276	1684
Validation set	54	61	236	69	420
Test set	12	14	19	15	60
Total	286	321	1197	360	2160

Tabla 5.1.2. Conjunto de datos sobremuestreado con data augmentation

	Arrugas	Gap	Ok	Suciedad	Total
Training set	878	984	942	1104	3908
Validation set	108	122	236	138	604
Test set	24	28	19	30	101
Total	1010	1134	1197	1272	4613

Tabla 5.1.3. Operaciones de transformación de imagen online

Transformación online	min	max
Traslación X	-300	300
Traslación Y	-100	100
Re-escalamiento	0,75	1,25

5.1.5. Optimización de hiperparámetros

Para la optimización de hiperparámetros se ha realizado una optimización bayesiana de los valores con el conjunto de datos base (**Tabla 5.1.1.**). Ajuste rango cada 15 experimentos y un uso de escalas apropiadas para la tasa de aprendizaje, regularización L2 y el tamaño del minibatch. No se ha considerado necesario añadir caída de learning rate.

Tabla 5.1.4. Valores de hiperparámetros de modelos optimizados

Red preentrenada	Dataset Preentren.	Numero de épocas	Learning rate	Tamaño Minibatch	Momento SDGM
AlexNet	ImageNet	30	0.000823	64	0.8967
GoogleNet	ImageNet	30	0,00355	16	0.7443
GoogleNet	Places365	30	0,000238	16	0.9793
ResNet-50	ImageNet	30	0,00145	8	0.8379

Tabla 5.1.5. Valores de hiperparámetros de modelos optimizados con regularización L2

Red preentrenada	Dataset Preentren.	Numero de épocas	Learning rate	Tamaño Minibatch	Momento SDGM	L2Reg (10^x)
AlexNet	ImageNet	30	0,000733	32	0.6742	-4.5581
GoogleNet	ImageNet	30	0,002177	16	0.9	-6.752
GoogleNet	Places365	30	0,00072	32	0.9768	-7.999
ResNet-50	ImageNet	15	0,00555	8	0,6576	-6.390

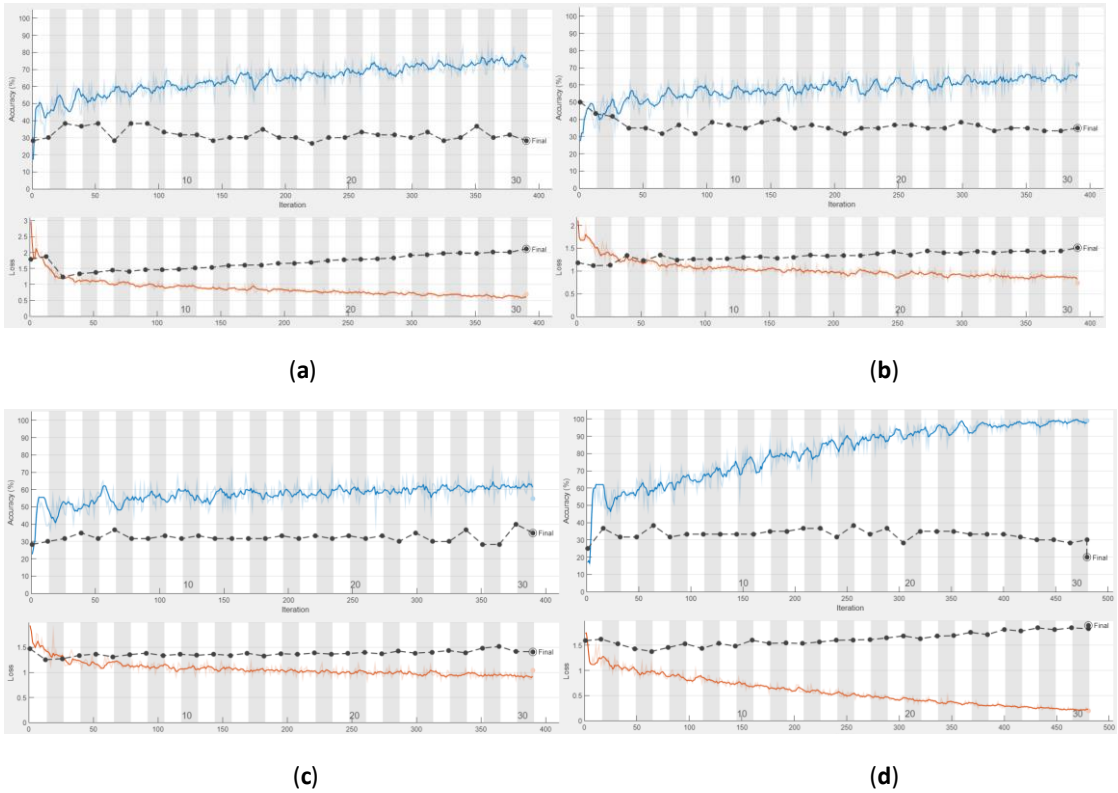
5.2. Resultados

5.2.1. Entrenamiento inicial

El primer paso consiste en entrenar rápidamente los modelos y observar su comportamiento para decidir el siguiente paso de estudio. Para este entrenamiento inicial se usa el conjunto de datos base (**Tabla 5.1.1.**) y las opciones de entrenamiento de la (**Tabla 5.2.1.**).

Tabla 5.2.1. Opciones de entrenamiento iniciales

Numero de épocas	Learning rate	Caída learning rate	Tamaño Minibatch	Momento SDGM	L2Reg
30	0.0001	0%	128	0.9	0



(**a**) (**b**)

(**c**) (**d**)

Figura 5.2.1. Graficas de entrenamiento de modelos: (**a**) AlexNet; (**b**) GoogLeNet-ImageNet; (**c**) GoogLeNet-Places365; (**d**) ResNet-50. Linea azul: Training accuracy suavizada; Linea roja: Training Loss suavizada; Linea negra discontinua: Validation Loss/Validation accuracy. Fuente: Propia.

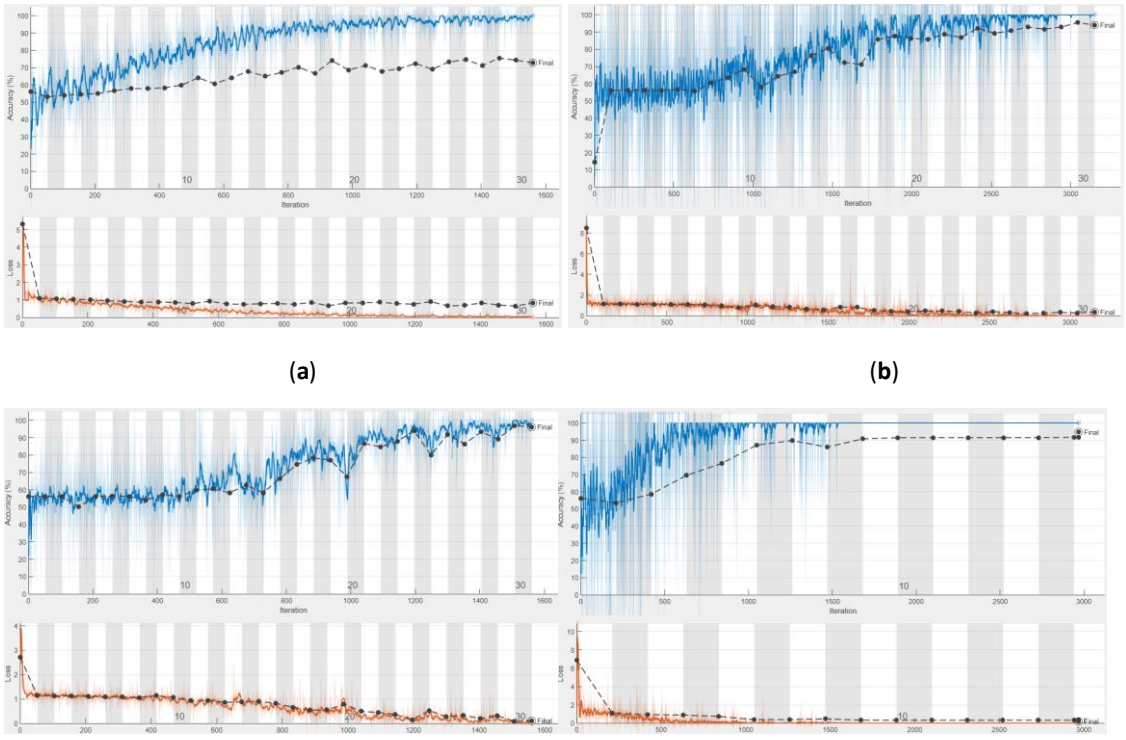
En esta situación el Validation Loss aumenta y Training Loss disminuye en todos los modelos. Los modelos sufren un problema común en Machine Learning conocido como overfitting. Los modelos se están centrando en el ruido de las imágenes de entrenamiento y está extrayendo las características en base a ello. Esto le ayuda a mejorar el rendimiento en el entrenamiento a costa de perjudicar su capacidad de generalización.

Para solucionar el overfitting existen técnicas como el uso regularización, dropout o añadir más datos. La adición de datos es la técnica más utilizada y puede hacerse mediante data augmentation o creación

de los datos artificiales con redes neuronales GAN. En este trabajo se estudian la regularización y la adición de datos con data augmentation y redes GAN.

4.2.2. Regularización

Conjunto de datos base (**Tabla 5.1.1.**) con los valores de hiperparámetros optimizados con regularización L2 (**Tabla 5.2.4.**)

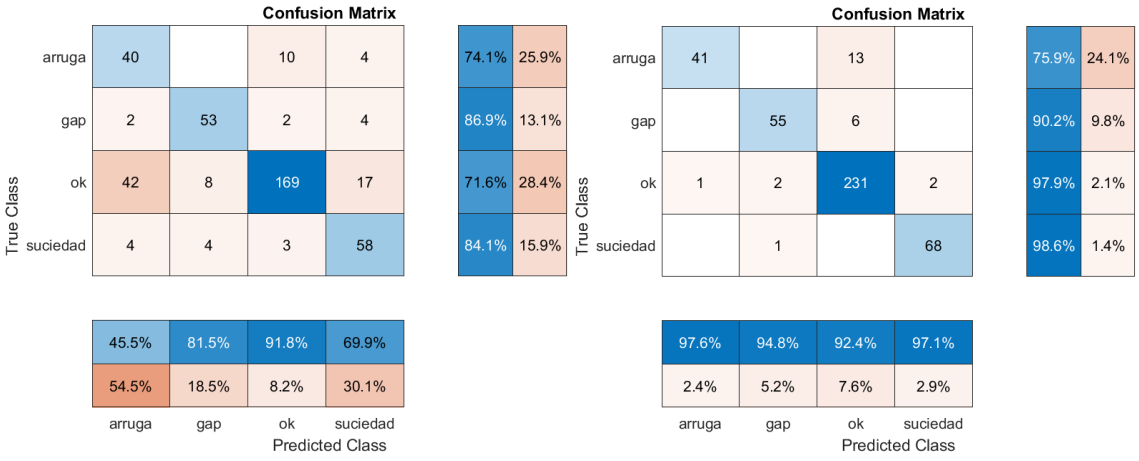


(**a**) (**b**)

(**c**) (**d**)

Figura 4.2.2. Graficas de entrenamiento de modelos: (**a**) AlexNet; (**b**) GoogLeNet-ImageNet; (**c**) GoogLeNet-Places365; (**d**) ResNet-50. Linea azul: Training accuracy suavizada; Linea roja: Training Loss suavizada; Linea negra semicontinua: Validation Loss/Validation accuracy. Fuente: Propia.

- Resultados sobre conjunto de validación



(**a**) (**b**)

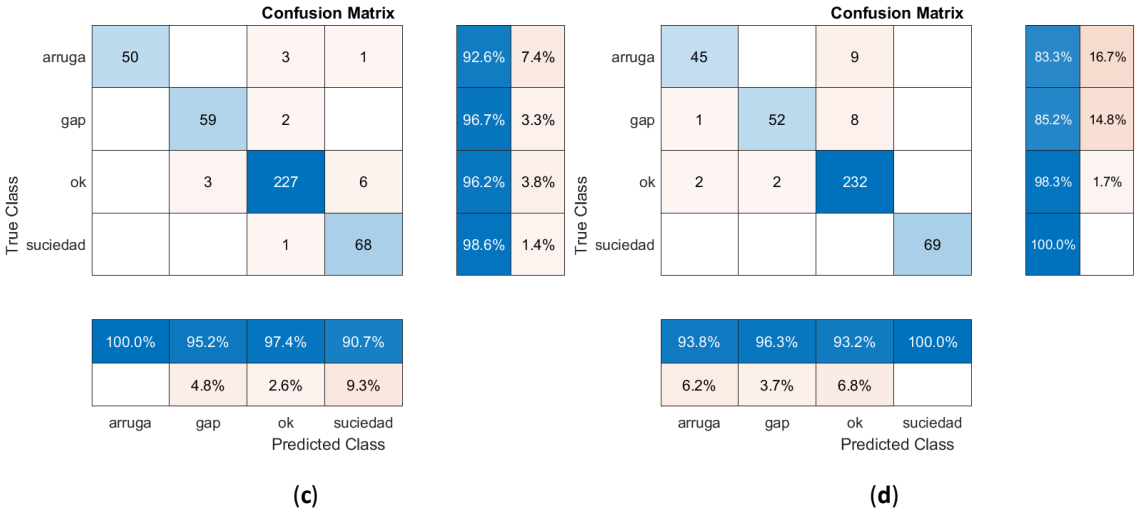
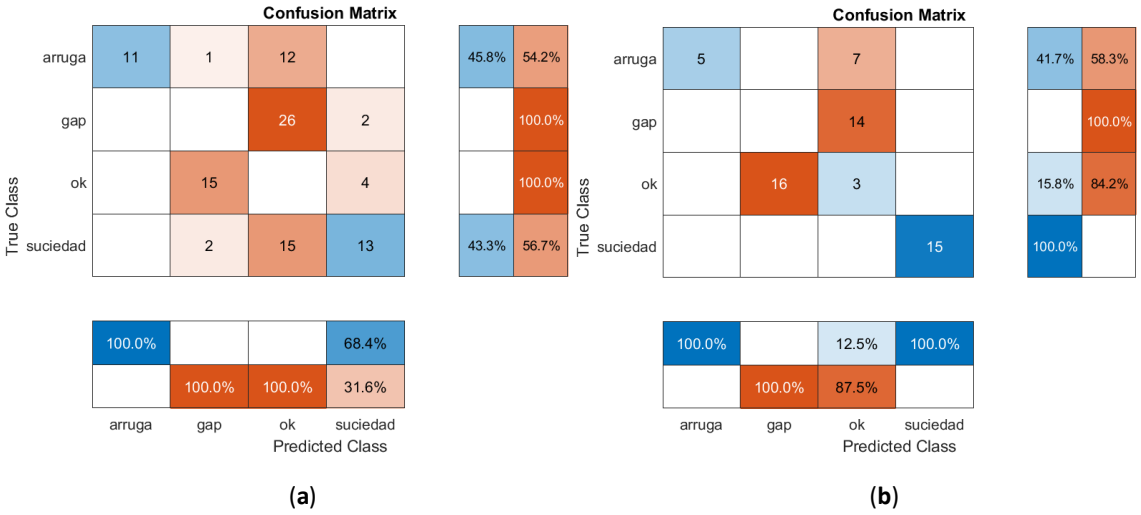
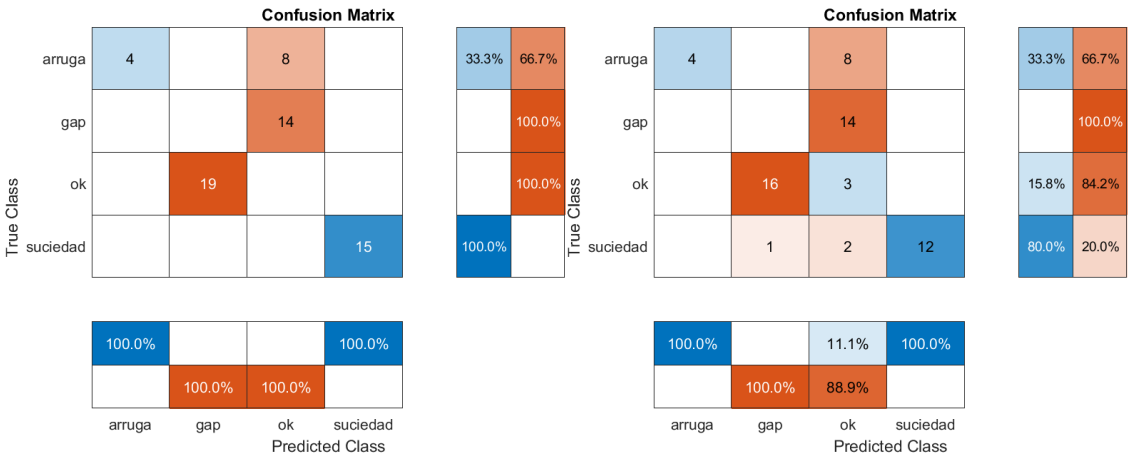


Figura 5.2.3. Resultados de matriz de confusión sobre el conjunto de validación para los modelos: (**a**) AlexNet; (**b**) GoogLeNet-ImageNet; (**c**) GoogLeNet-Places365; (**d**) ResNet-50. Fuente: Propia.

- Resultados sobre conjunto de test





(c) (d)

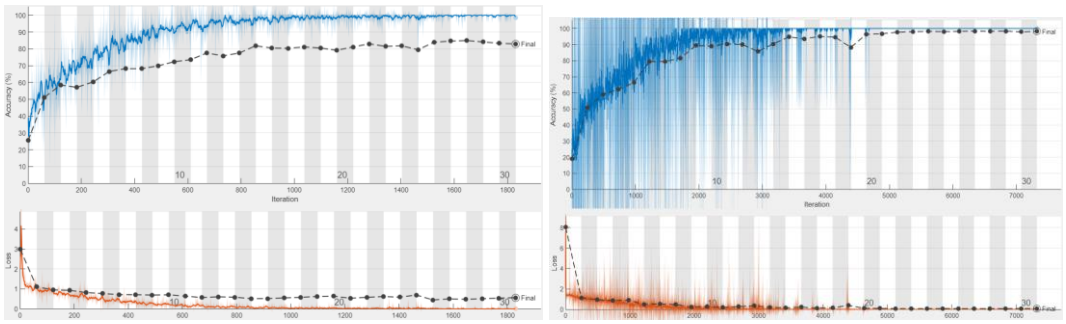
Figura 5.2.4. Resultados de matriz de confusión sobre el conjunto de test para los modelos: (**a**) AlexNet; (**b**) GoogLeNet-ImageNet; (**c**) GoogLeNet-Places365; (**d**) ResNet-50. Fuente: Propia.

Tabla 5.2.1. Resultados de validation accuracy, test accuracy con regularización L2.

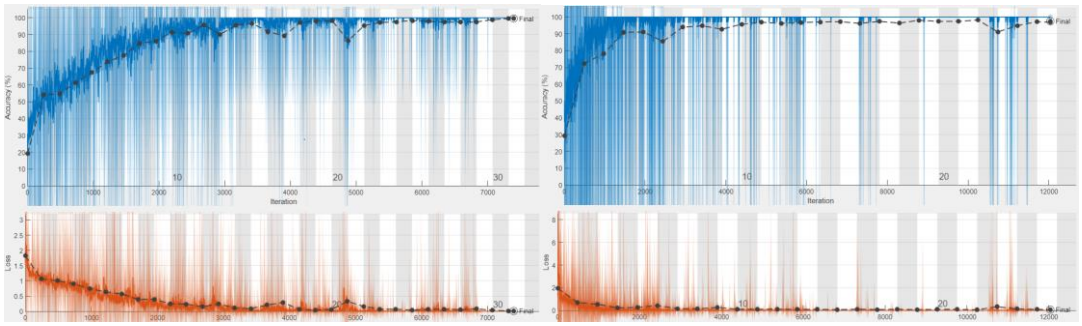
	Dataset preentr.	**Épocas**	**Tiempo entr.**	**Tamaño (MB)**	**Parámetros (Millions)**	**Validation Accuracy**	**Test Accuracy**
AlexNet	ImageNet	30	9' 4''	227	61	76,19%	23,76%
GoogleNet	ImageNet	30	14' 49''	27	7	94,05%	38%
GoogleNet	Places365	30	11' 46''	27	7	96,19%	25%
ResNet-50	ImageNet	15	16' 39''	96	25,6	98,84%	23,33%

5.2.3. Datos balanceados

Conjunto de datos sobremuestreado con data augmentation (**Tabla 4.1.2.**) con los valores de hiperparámatros optimizados (**Tabla 4.2.4.**)



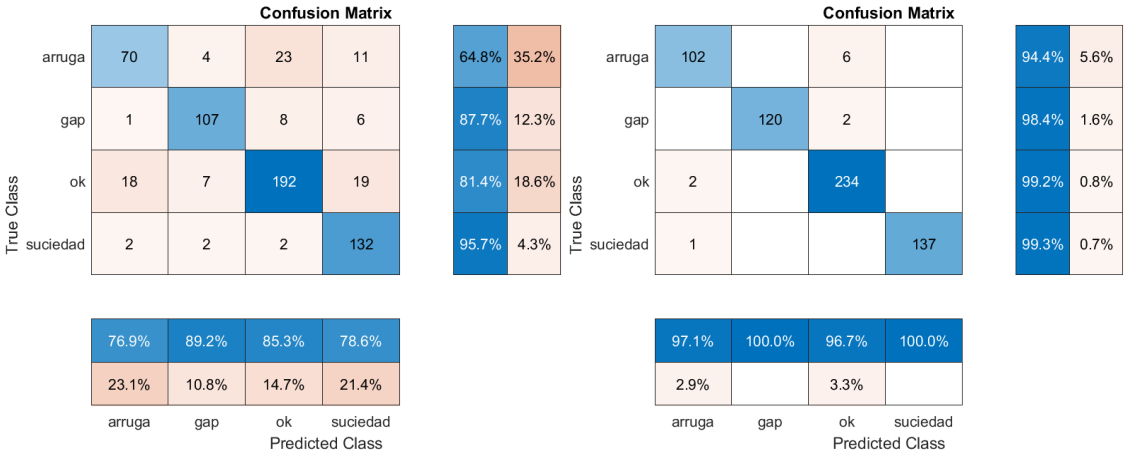
(a) (b)



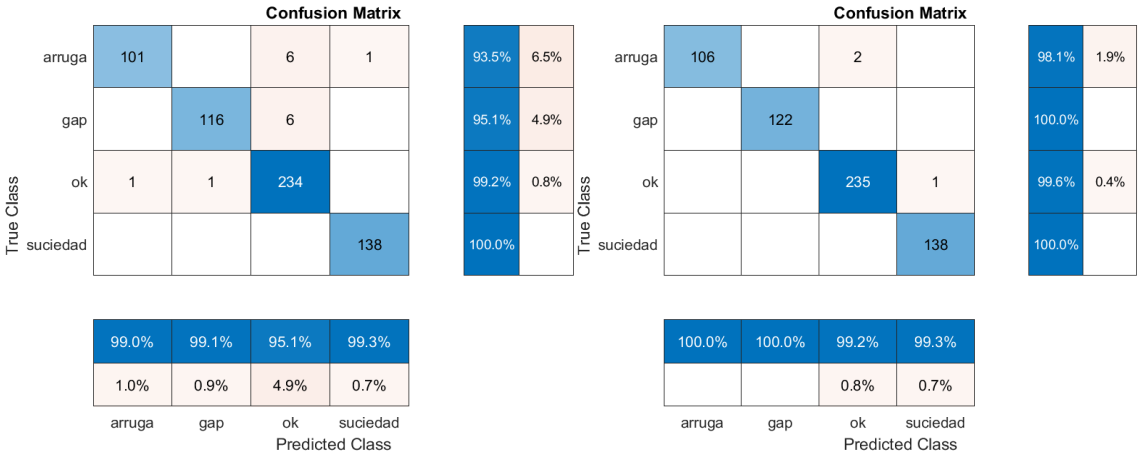
(b) **(d)**

Figura 5.2.3. Graficas de entrenamiento de modelos: (**a**) AlexNet; (**b**) GoogLeNet-ImageNet; (**c**) GoogLeNet-Places365; (**d**) ResNet-50. Linea azul: Training accuracy suavizada; Linea roja: Training Loss suavizada; Linea negra semicontinua: Validation Loss/Validation accuracy. Fuente: Propia.

- Resultados sobre el conjunto de validación



(a) **(b)**



(c) **(d)**

Figura 5.2.3. Resultados de matriz de confusión sobre el conjunto de validación para los modelos: (**a**) AlexNet; (**b**) GoogLeNet-ImageNet; (**c**) GoogLeNet-Places365; (**d**) ResNet-50. Fuente: Propia.

- Resultados sobre el conjunto de validación

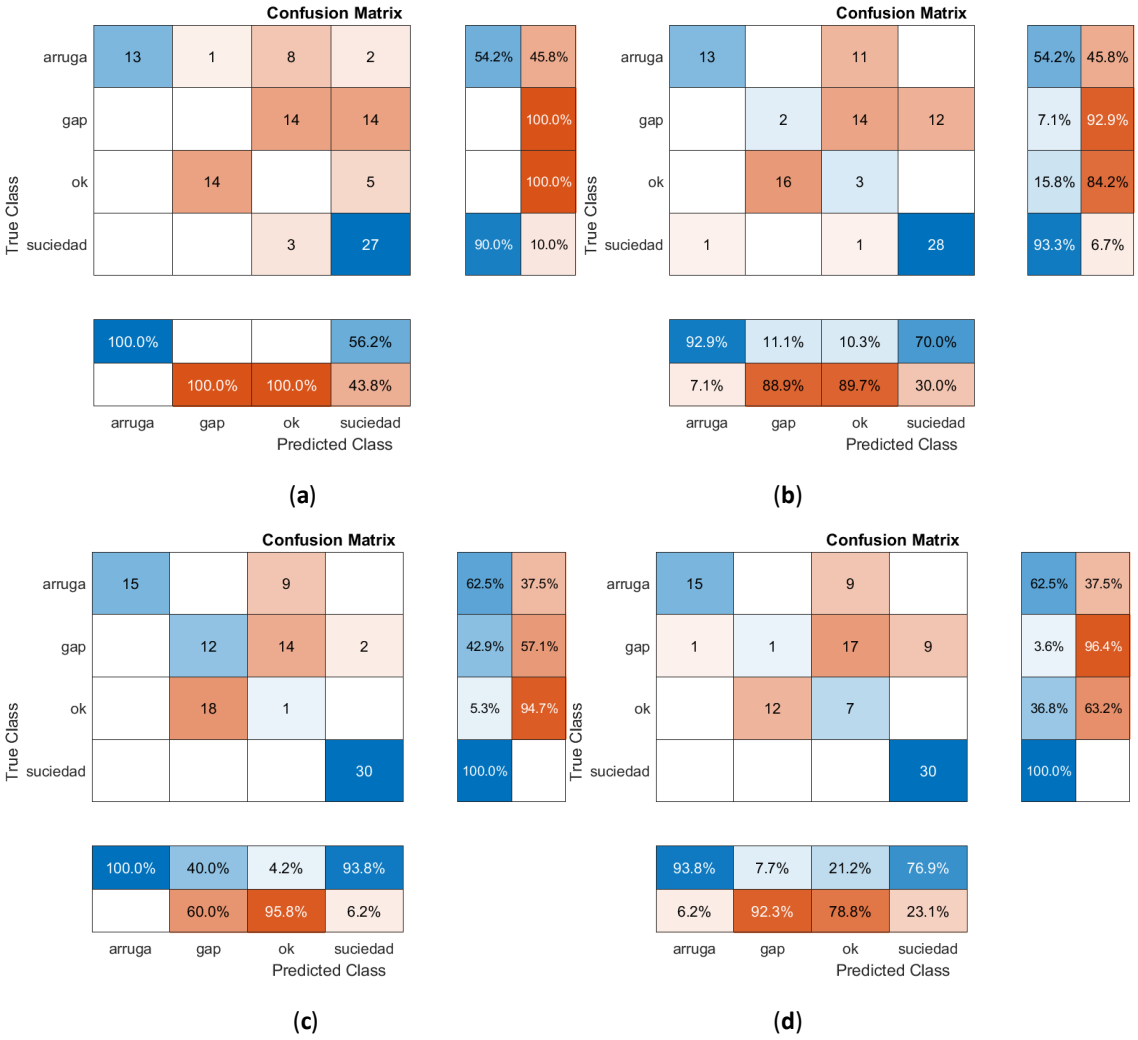


Figura 5.2.3. Resultados de matriz de confusión sobre el conjunto de validación para los modelos: (**a**) AlexNet; (**b**) GoogLeNet-ImageNet; (**c**) GoogLeNet-Places365; (**d**) ResNet-50. Fuente: Propia.

Tabla 4.2.2. Resultados de validation accuracy, test accuracy con balanceo de datos.

	Dataset preentr.	**Épocas**	**Tiempo entr.**	**Tamaño (MB)**	**Parámetros (Millones)**	**Validation Accuracy**	**Test Accuracy**
AlexNet	ImageNet	30	16'04''	227	61	82,95%	25%
GoogleNet	ImageNet	30	34'30''	27	7	98,18%	46%
GoogleNet	**Places365**	**31**	**32'13''**	**27**	**7**	**99,50%**	**57%**
ResNet-50	ImageNet	25	68'17''	96	25,6	97,52%	52,48%

5.2.4. Transformaciones de imágenes

Conjunto de datos aumentado con data augmentation (**Tabla 5.1.2.**) con los valores de hiperparámatros optimizados (**Tabla 5.2.4.**) y las operaciones de transformación de imagen online (**Tabla 5.1.3.**)

- Resultados sobre conjunto de validación

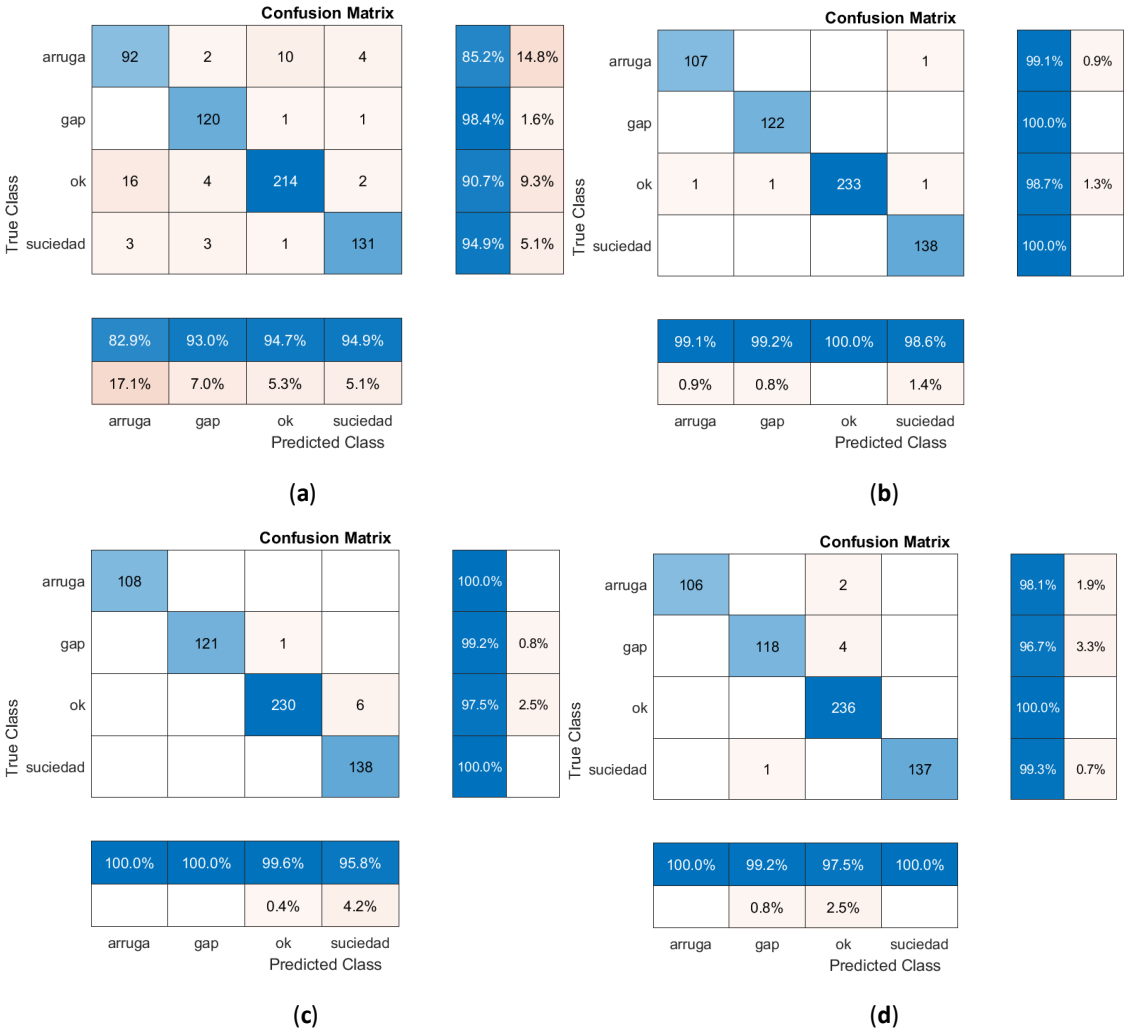
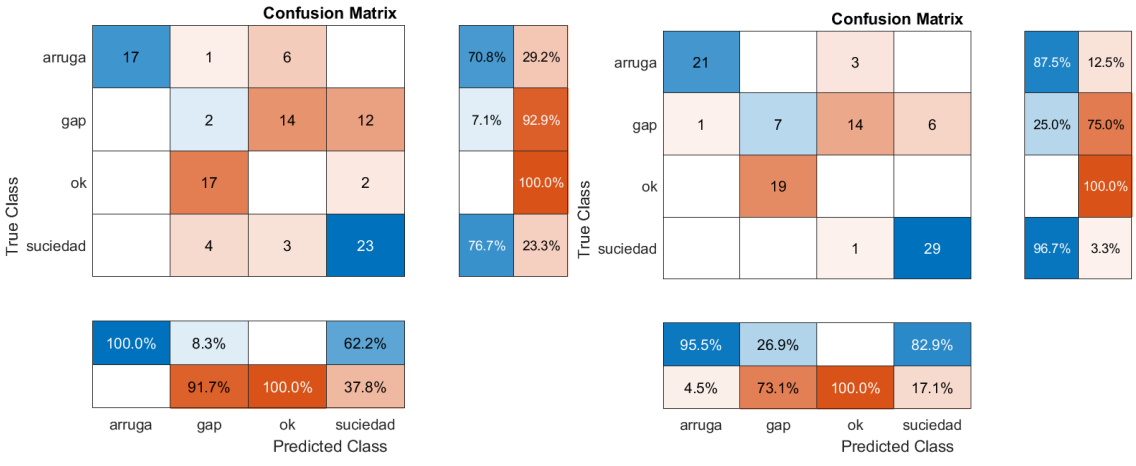


Figura 5.2.3. Resultados de matriz de confusión sobre el conjunto de validación para los modelos: (**a**) AlexNet; (**b**) GoogLeNet-ImageNet; (**c**) GoogLeNet-Places365; (**d**) ResNet-50. Fuente: Propia.

- Resultados sobre conjunto de test



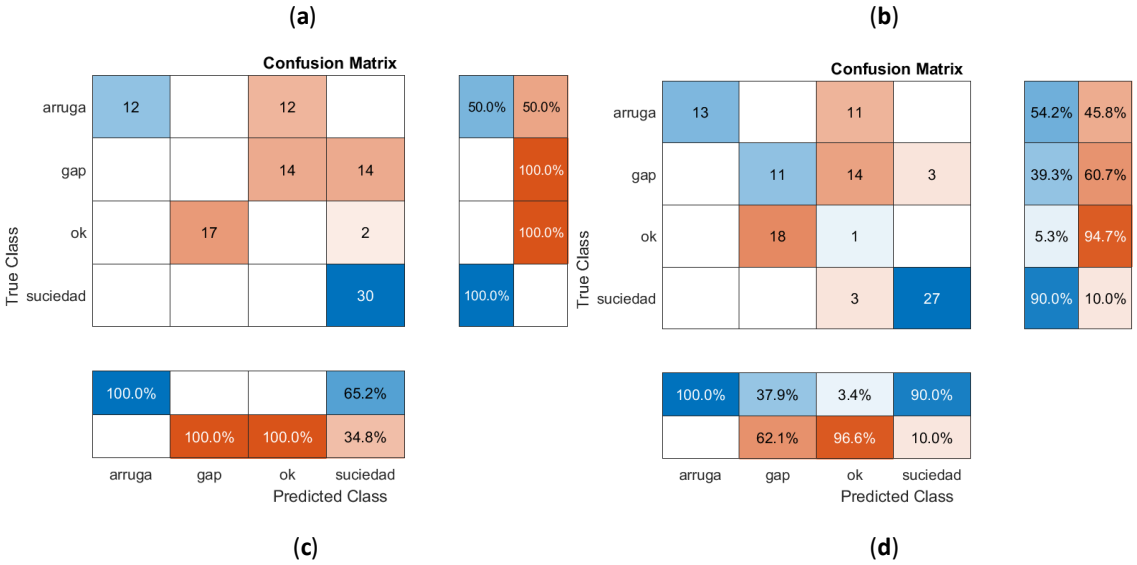


Figura 5.2.4. Resultados de matriz de confusión sobre el conjunto de test para los modelos: (**a**) AlexNet; (**b**) GoogLeNet-ImageNet; (**c**) GoogLeNet-Places365; (**d**) ResNet-50. Fuente: Propia.

Tabla 5.2.3. Resultados de validation accuracy, test accuracy con operaciones de transformación.

Red preentr.	Dataset preentr.	Épocas	Tiempo entr.	Tamaño (MB)	Parámetros (Millones)	Validation accuracy	Test accuracy
AlexNet	ImageNet	50	47’ 31’’	227	61	92,22%	41,58%
GoogleNet	**ImageNet**	**30**	**40’ 07”**	**27**	**7**	**99,34%**	**56,44%**
GoogleNet	Places365	35	44’ 53’’	27	7	98,84%	41,58%
ResNet-50	ImageNet	26	80’ 05’’	96	25,6	98,84%	51,49%

5.3. Conclusiones

El método de cálculo de la red, “minibatch gradient descent”, calcula los gradientes entre cada minibatches y baraja aleatoriamente las imágenes entre épocas para mejorar la generalización. Es por ello por lo que los resultados tienen una alta variabilidad en los resultados y las gráficas muestran ruido.

Para el conjunto de test se han elegido imágenes de piezas que no han sido vistas durante el entrenamiento ni validación. Para aumentar la dificultad, estas imágenes incluyen características difíciles de diferenciar con otras clases.

En la industria es más importante evitar los falsos negativos que los falsos positivos. Así, por tanto, priorizamos el modelo que permite un alto rendimiento en los términos de precisión sobre los defectos. La mayoría de los modelos han tenido altos resultados sobre el conjunto de validación. Entre los dos modelos que han presentado mayores resultados sobre el conjunto de test: GoogLeNet-ImageNet ofrece una tasa de acierto sobre defectos de arrugas de 87,5% y gap de 22,5% (**Fig. 5.2.4.b.**). Mientras que GoogLeNet-365 ofrece una tasa de acierto sobre los defectos de arrugas de 62,5% y gap de 42,9% (**Fig. 5.2.3.c.**)

Para la selección del modelo, escogemos GoogLeNet-ImageNet porque responde mejor a los defectos de arruga los cuales presentan mucha variabilidad en sus características de forma y posición y son más difíciles de detectar.

5.4. Generative Adversarial Network

5.4.1. Introducción

Para el entrenamiento de una red GAN usamos el conjunto de datos aumentado (**Tabla 5.1.2.**) de una clase de defecto. Las imágenes son transformadas a 64x64x3 para reducir el coste computacional y 500 épocas de entrenamiento. Los parámetros de entrenamiento son [11]

Para la red generadora se crea una red que convierte un vector de ruido de tamaño 100 en una imagen de 64x64x3. La capa de entrada transforma el vector de ruido a 7x7x128 y le sucede una etapa de capas de convolución y capas de activación ReLU, de tamaño 5x5xN con N, el número de filtros decreciente y cuya última capa de activación es tanh.

Para la red discriminadora se crea una red que acepta imágenes de 64x64x3 y devuelve una puntuación de clasificación. Usando una serie de capas de convolución con capas de activación ReLU. Las capas de convolución son de tamaño 5x5xN con N el número de filtros creciente. La capa final es 4x4x1. Para controlar el aprendizaje del discriminador dropout de 0.5.

5.4.2. Resultado

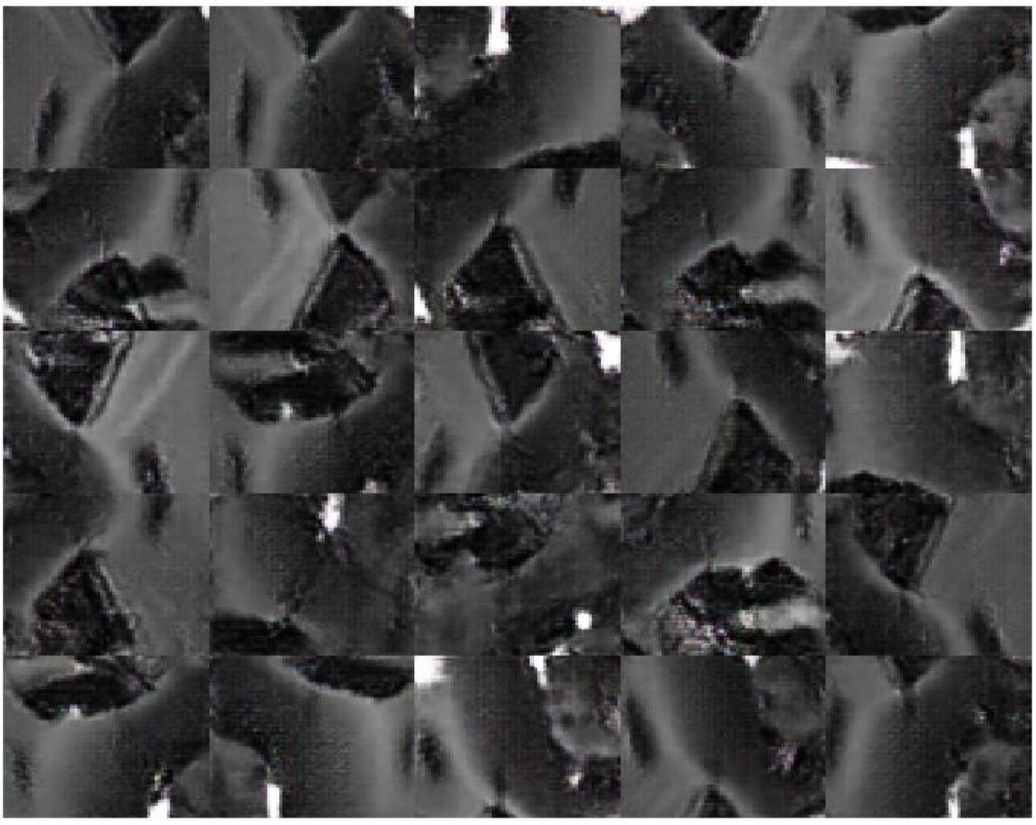


Figura 5.4.2. Imágenes de defectos sintéticos originados con red GAN

5.4.3. Conclusión

La generación de datos sintéticos a partir de redes GAN, es una práctica ampliamente utilizada por sus excelentes resultados para el entrenamiento de redes. En las imágenes sintéticas válidas, se aprecia el defecto como si se estuvieran viendo de lejos. En nuestro caso, no se ha considerado necesario el uso de las imágenes para entrenamiento.

5.5. Análisis de clasificación

5.5.1. Introducción

Las herramientas de visualización nos permiten entender el comportamiento de las redes por medio de la resaltación de las características que influyen en la clasificación. Para comprobar la correcta detección de la red visualizamos los resultados de occlusion sensitivity y LIME de las diferentes formas de los defectos de la clase de arruga. [25]

5.5.2. Resultados

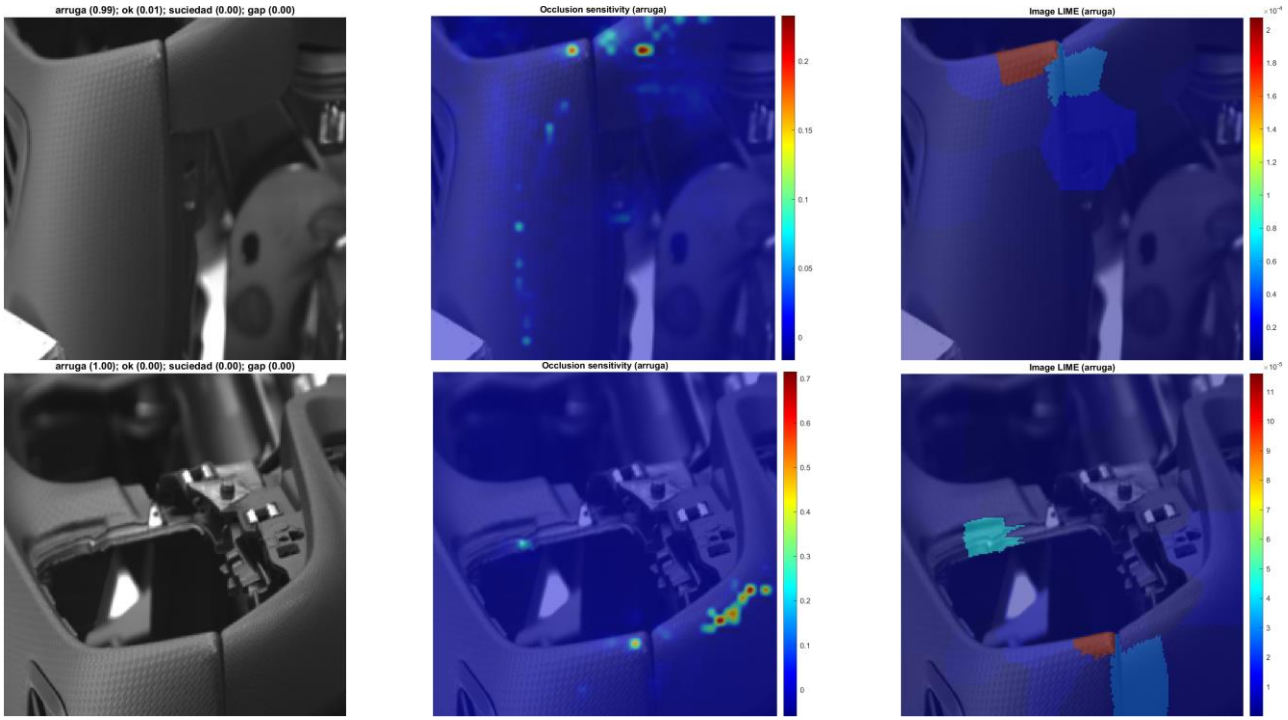


Figura 5.5.1. Resultados de visualización de arruga de bulto crítico con: (**a**) imagen; (**b**) occlusion sensitivity; (**c**) LIME.

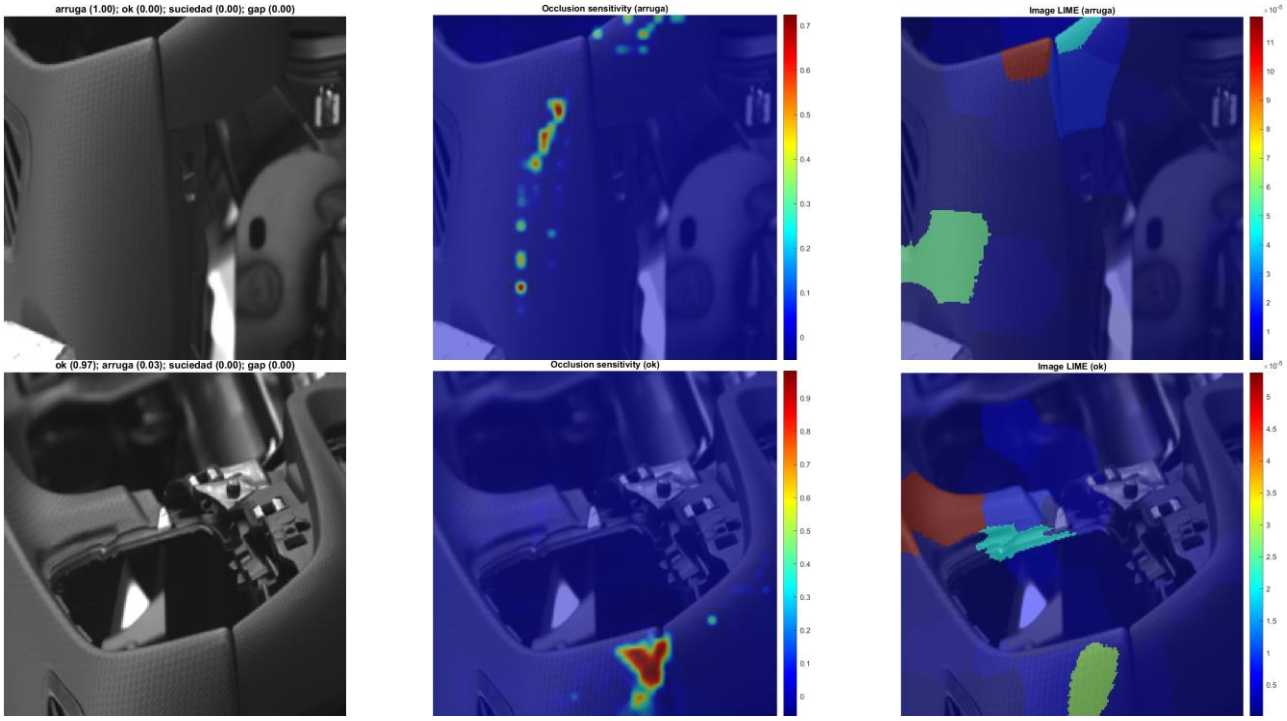


Figura 5.5.2. Resultados de visualización de arruga de bulto no crítico con: (**a**) imagen; (**b**) occlusion sensitivity; (**c**) LIME.

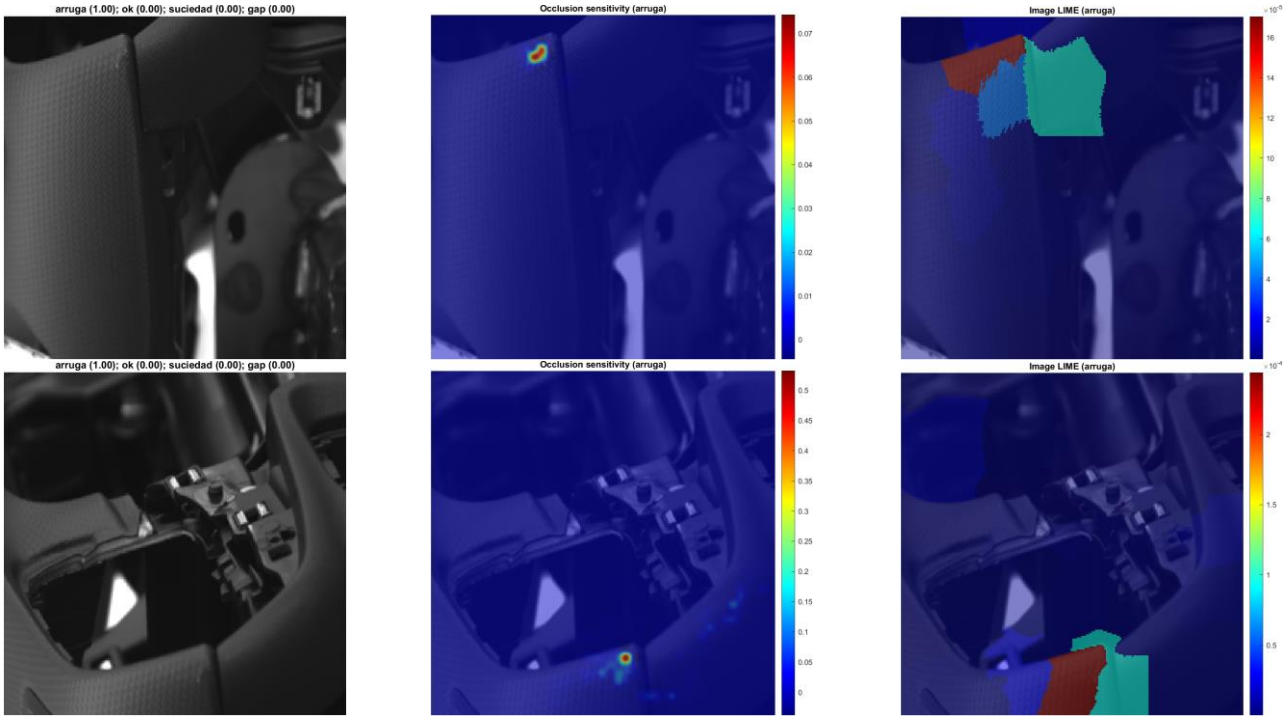


Figura 5.5.3. Resultados de visualización de arruga de pliegue crítico con: (**a**) imagen; (**b**) occlusion sensitivity; (**c**) LIME.

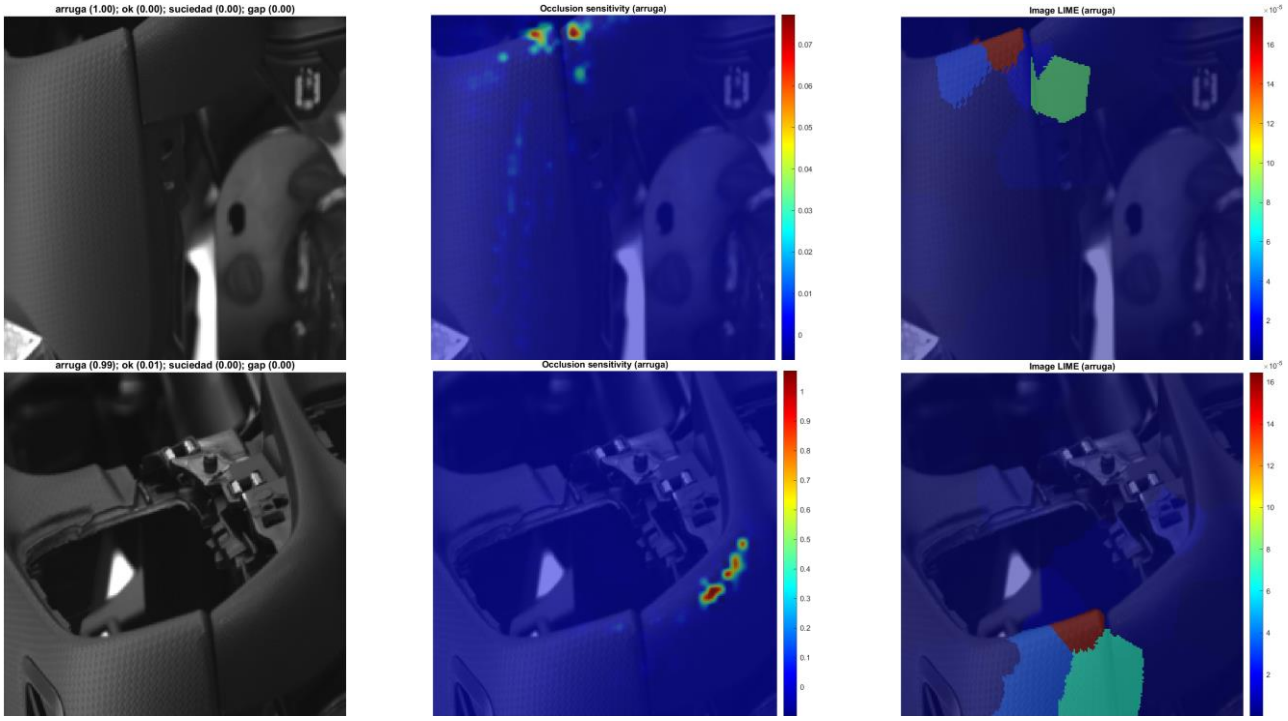


Figura 5.5.4. Resultados de visualización de arruga de pliegue no crítico con: (**a**) imagen; (**b**) occlusion sensitivity; (**c**) LIME.

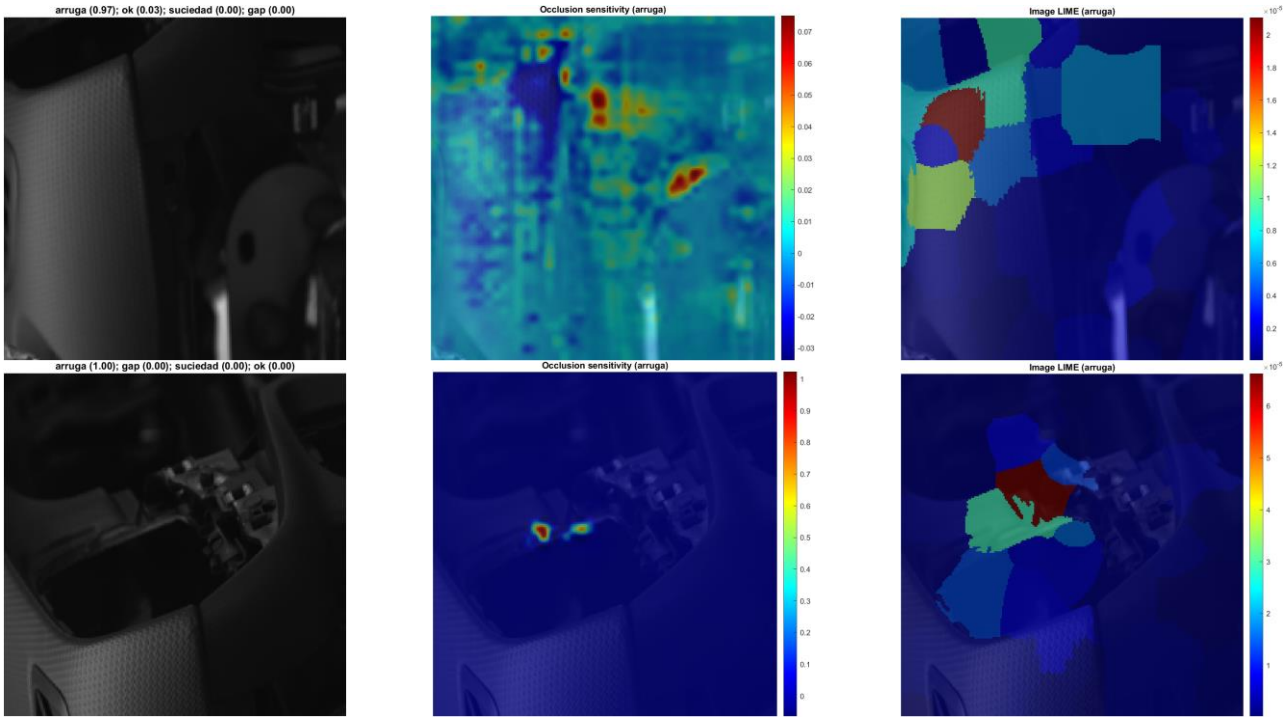


Figura 5.5.5. Resultados de visualización de arruga de pliegue crítico con falta de iluminación: (**a**) imagen; (**b**) occlusion sensitivity; (**c**) LIME.

5.5.3. Conclusiones

Los defectos de arrugas con forma de bultos tienen la dificultad de que requiere de escoger el ángulo adecuado para visualizar la distorsión de forma. Como consecuencia en la Figura 4.5.2., el bulto no crítico es detectado en la entrada de la pieza en el plano imagen, pero en la salida la pieza es detectada como OK.

Se ha comprobado que el detector es robusto ante las vibraciones externas. Sin embargo, la variación de iluminación externa afecta drásticamente a la detección de la red.

6. Estudio de detección de objetos

6.1. Introducción

Para entrenar los detectores de objetos requiere ajustar una serie de parámetros que parametrizan las redes YOLOv2 y Faster RCNN.

1- Numero de anchor boxes
2- Tamaño de la imagen
3- Capa de extracción de características

6.1.1. Ground truth data

Se ha creado el ground truth data a partir del conjunto de datos base con la aplicación de Matlab Image Labeler. En el etiquetado de las imágenes se han incluido clases manchas más específicos como defectos de brillo.

6.1.2. Número de anchor boxes

Aumentar el número de anchor box puede mejorar la medida de mean IoU. Sin embargo, aumentar el número puede incrementar el coste computacional y llegar a overfitting. Se obtienen una gran mejoría de mean IoU con el uso de 1-5 anchor boxes. A partir de 5 obtiene tan solo una mejoría marginal de mean IoU (**Fig. 6.1.1.**) a costa de perjudicar la precisión de la red.

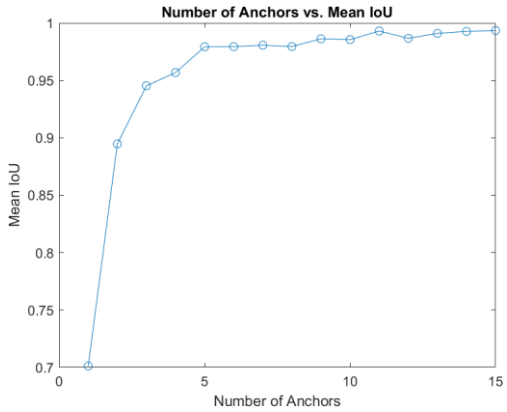


Figura 6.1.1. Número de anchor boxes frente mean IoU.

6.1. 3. Capa de extracción de características

La elección de la capa de extracción requiere de análisis empírico. La capa de extracción de características tiene una función distinta en la arquitectura del detector. En la red Faster RCNN alimenta la capa de RPN y ROI Pooling layer es incluida después de la capa de características. En la red YOLOv2 alimenta a la etapa de convolución de la red YOLO.

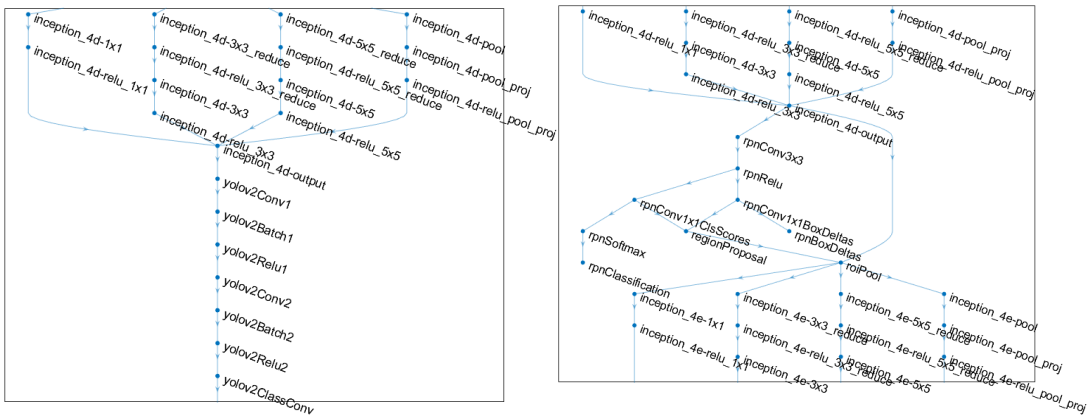


Figura 4.4.2. Arquitectura de la red de extracción: (**Izquierda**) YOLOv2; (**Derecha**) Faster RCNN.

6.1.4. Data augmentation

En este caso las operaciones de data augmentation son las mismas que en el apartado de clasificación con la diferencia de que no aumentamos el dataset. Los volteos en X e Y son variables aleatorias calculadas al principio de cada minibach.

6.1.5. Opciones de entrenamiento

Los valores se han optimizado mediante rejilla y se ha usado el optimizador Adam para mejorar la rapidez y el comportamiento de la red ante los 'puntos sillín'. El rango de solapamiento positivo y negativo para entrenar la red Faster R-CNN es [0.6 1] y [0 0.3].

Tabla 12. Opciones de entrenamiento

Detector de objetos	Red extractora	Capa de extracción	Anchor boxes	Épocas	Learning rate	Caída learning rate	Tamaño Minibatch	Squared Gradient	Gradient decay
YOLOv2	GoogleNet_ ImageNet	inception_ 4d-output	5	120	0.001	10%/75 epocas	16	0.99	0.9
Faster RCNN	GoogleNet_ ImageNet	inception_ 4d-output	3	75	0.0008	10%/ 20 épocas	2	0.9	0.95

6.2. Resultados

Para cada clase se obtiene la curva de precision-recall con el conjunto de validación:

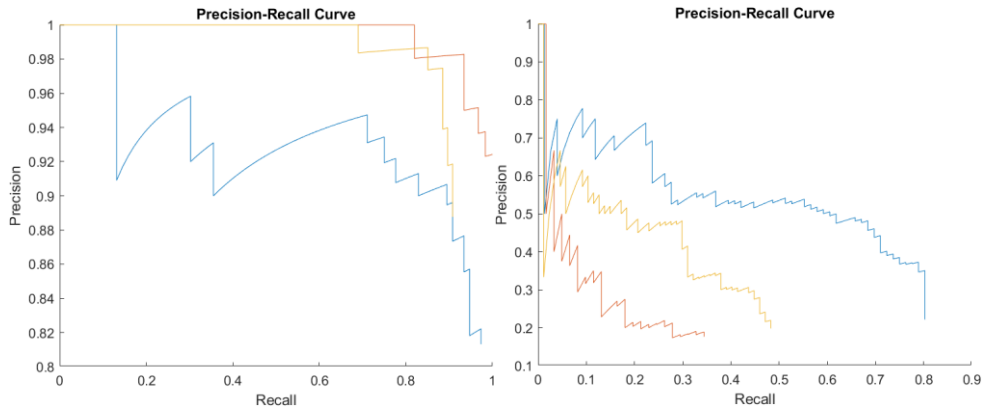


Figura 12. Resultados de curvas de precision-recall: (**Izquierda**) YOLOv2; (**Derecha**) Faster RCNN. Azul: arruga; rojo: gap; amarillo: suciedad.

Tabla 12. Resultados de average-precision para cada defecto y medium average precisión de cada red.

Detector	Tiempo de entr.	Tiempo detecc. (s)	arruga	gap	suciedad	mAP
YOLOv2	2h	0.9772	0.908	0.994	0.9032	0.935
Faster R-CNN	6 dias	0.6063	0.3683	0.118	0.164	0.216

6.2. Resultados de detección con YOLOv2

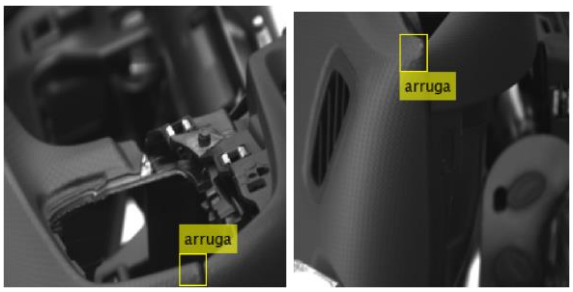


Figura 6.2.1. Resultados de predicciones de defectos de arrugas con detector YOLOv2

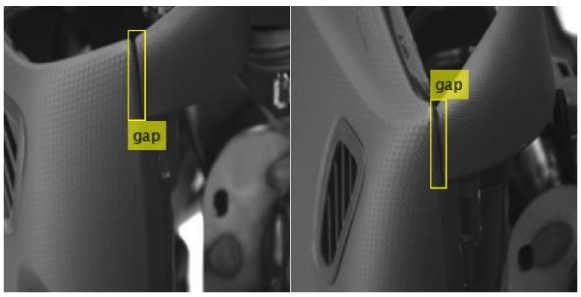


Figura 6.2.2. Resultados de predicciones de defectos de gap con detector YOLOv2

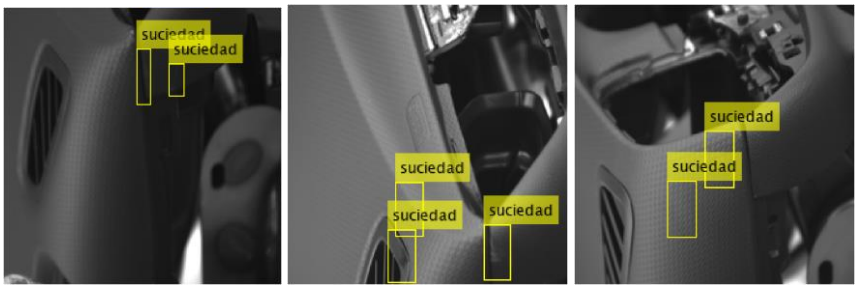


Figura 6.2.3. Resultados de predicciones de defectos de manchas con detector YOLOv2: (**a**) suciedades; (**b**) suciedad y liquido; (**c**) suciedad y brillo.

6.3. Conclusiones

Los 'bounding boxes' de los resultados tienen tamaños similares porque se han estimado los anchor boxes basados en los 'bounding boxes' del conjunto de entrenamiento para reducir el coste computacional y acelerar el entrenamiento. En caso contrario tendría que probar diferentes variaciones con diferentes tamaños de boxes.

El detector YOLOv2 generalmente presenta una precisión menor que el Faster RCNN en la detección de objetos con características complicadas, sin embargo, el mean average precisión ha sido superior para esta red. Las diferencias de precisión entre ambas redes pueden ser ocasionadas por la dificultad de entrenamiento de la red Faster R-CNN. Al ser una red tan grande agota los recursos de memoria y el entrenamiento ha sido realizado en condiciones de ahorro de memoria. Una GPU con mayor memoria permitiría un entrenamiento apropiado.

En la figura puede demostrarse la eficacia de este sistema para la detección de manchas de diferentes características: manchas de suciedad, líquidos y defectos de brillos (Fig. 6.2.3.).

VII. Conclusiones

Este trabajo propone un estudio que juega un papel importante dentro de la inspección de defectos en piezas de termoplásticos. Las ideas adquiridas para la captación de los defectos en estas piezas pueden ser transferidas a piezas de características similares.

Para lograr resultados precisos y sofisticados las redes neuronales requieren de una cantidad ingente de datos para entrenar. Pero con la técnica de data augmentation para aumentar el tamaño de nuestros datos hemos demostrado que se pueden crear modelos de clasificación con precisiones superiores al 98%. Además, con un software no especializado en deep learning hemos obtenido grandes resultados gracias a un ajuste apropiado de las opciones de entrenamiento. Para la mejora de los modelos un ajuste de hiperparámetros y una mejora del preprocesamiento pueden ser llevados a cabo.

Se ha demostrado la robustez del modelo ante variaciones en el enfoque o perturbaciones en la imagen a excepción de baja luminosidad. Sin embargo, no ha sido posible demostrar su eficacia con defectos que no presentan características visibles en la imagen por lo que un sistema de cámaras extendido es requerido.

Los tres tipos de defectos típicos de la pieza que hemos usado para la detección se caracterizan por manifestar cambios de sombras y luces (arrugas); regiones de diferentes tonalidades y brillos en varias posiciones (manchas); y diferencias de tolerancias (gap); a raíz de esta información podemos asegurar que es posible detectar el resto de los defectos por medio de redes neuronales contando con un buen sistema de captura.

En general, se valida el uso de redes neuronales para los sistemas de inspección, que es aplicable a multitud de tipologías de piezas y problemas de inspección complejos.

Este trabajo incluye las fases de clasificación de imágenes y detección de objetos, no obstante, en el futuro este método puede ser extendido a fases de detección por segmentación y análisis de los parámetros de los defectos como el tamaño y la forma.

VIII. Bibliografía

[1] An Introduction to Machine Vision Systems. Thomasnet.com

[2] Ali Khan, H.; Jue, W.; Mushtaq, M. Brain tumor classification in MRI image using convolutional neural network. **2020**.

[3] Moreno Diaz, L. Análisis comparativo de arquitecturas de redes neuronales para la clasificación de las imágenes. **2020**.

[4] Tammina, S. Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images; **2019**.

[5] Redes Neuronales Convolucionales. Consultores estratégicos en Ciencia de Datos. Juanbarrios.com

[6] Zhou, Q; Chen, R; Huang, B; An Automatic Surface Defect Inspection System for Automobiles Using Machine Vision Methods. 2019

[7] Get started with transfer learning. Mathworks.com

[8] ImageNet. Wikipedia.org.

[9] Collado, P. Reconocimiento de escenas mediante integración multiescala de redes convolucionales. 2021.

[10] Gubta, S. Natural Scene Recognition Using Deep Learning. Towardsdatascience.com

[11] Train Generative Adversarial Network (GAN). Mathworks.com.

[12] Reed, S; Akata, Z. Generative Adversarial Text to Image Synthesis. 2016.

[13] Getting Started with Object Detection Using Deep Learning. Mathworks.com.

[14] trainYOLOv2ObjectDetector. Mathworks.com.

[15] Hui, J. mAP (mean Average Precision) for Object Detection. Medium.com. 2018.

[16] 10 Techniques to deal with Imbalanced Classes in Machine Learning. 2020. Analyticsvidhya.com

[17] Singh, S. Understanding the Bias-Variance Tradeoff. **2018**.

[18] Ng, A. Programa especializado de aprendizaje profundo. Coursera.org.

[19] Ghandi, A. Data Augmentation | How to use Deep Learning when you have Limited Data — Part 2. Nanonets.com.

[20] Pingel, J. Data Augmentation for Image Classification Applications Using Deep Learning. **2019**.

[21] Ramos Rivero, V. L. Evolución del uso de los materiales plásticos en la industria automotriz. **2018**.

[22] Cómo reparar salpicadero. Club.autodoc.es.

[23] Lopez Serrano, A. MOLDE DE INYECCIÓN. 2018. Universidad del país vasco.

[24] Problemas de desempeño de productos plásticos. Plastics Technology Mexico.

[25] Explore Network Predictions Using Deep Learning Visualization Techniques. Mathworks.com.

IX. Anexos

ANEXO I. Codigo de Matlab. Clasificación de imágenes.

```
clear
clc

%% Hiperparámetros

net = resnet50;
InitialLearnRate = 10^(-2.8385);
MaxEpochs = 30;
miniBatchSize = 8;
momentum = 0.8379;

%% Importing Dataset

imdsTrain = imageDatastore('C:\Users\José Antonio\Documents\MATLAB\datasets\[4] data set
augmented - training', ...
    "IncludeSubfolders",true, ...
    "LabelSource","foldernames");
imdsTrain.ReadSize = miniBatchSize;
imdsVal = imageDatastore('C:\Users\José Antonio\Documents\MATLAB\datasets\[5] data set
augmented - validation', ...
    "IncludeSubfolders",true, ...
    "LabelSource","foldernames");
imdsTest = imageDatastore("C:\Users\José Antonio\Documents\MATLAB\datasets\[6] data set
augmented - test", ...
    "IncludeSubfolders",true, ...
    "LabelSource","foldernames");

%% Creating the modified net

inputSize = net.Layers(1).InputSize;
numClasses = numel(categories(imdsTrain.Labels));
lgraph = pretrainedNetModifier(net, numClasses);
analyzeNetwork(lgraph)

%% Configuring Online Data augmentation
pixelRangeX = [-300 300];
pixelRangeY = [-50 50];
scaleRange = [0.75 1.25];
imageAugmenter = imageDataAugmenter( ...
    'RandXTranslation',pixelRangeX, ...
    'RandYTranslation',pixelRangeY, ...
    'RandScale',scaleRange);
%     'RandXReflection',true, ...'DataAugmentation',imageAugmenter

augimdsTrain =
augmentedImageDatastore(inputSize(1:2),imdsTrain,"ColorPreprocessing","gray2rgb");
augimdsVal =
augmentedImageDatastore(inputSize(1:2),imdsVal,"ColorPreprocessing","gray2rgb");
augimdsTest =
augmentedImageDatastore(inputSize(1:2),imdsTest,"ColorPreprocessing","gray2rgb");


%% Configuring Training options

valFrequency = floor(numel(augimdsTrain.Files)/miniBatchSize);
options = trainingOptions('sgdm', ...
    'Momentum', momentum,...
    'MiniBatchSize',miniBatchSize, ...
    'MaxEpochs',MaxEpochs, ...
    'InitialLearnRate',InitialLearnRate, ...
    'ValidationData',augimdsVal, ...
    'ValidationFrequency',valFrequency, ...
    'Verbose',true, ...
    'Plots','training-progress', ...
    'Shuffle', 'every-epoch');

%% Training the net

[net, info] = trainNetwork(augimdsTrain,lgraph,options);

%% Calculating Metrics
%
```

```
[testPreds, scrs] = classify(net,augimdsVal);
testActual = imdsVal.Labels;
numCorrect = nnz(testPreds == testActual);
fracCorrect = numCorrect/numel(testPreds);

figure
cm = confusionchart(imdsVal.Labels,testPreds);
cm.ColumnSummary = 'column-normalized';
cm.RowSummary = 'row-normalized';
cm.Title = 'Confusion Matrix';
%%


%% Auxiliary functions

function lgraph = pretrainedNetModifier(net,numClasses)
net.Layers(1)


if isa(net,'SeriesNetwork')
  lgraph = layerGraph(net.Layers);
else
  lgraph = layerGraph(net);
end

[learnableLayer,classLayer] = findLayersToReplace(lgraph);

if isa(learnableLayer,'nnet.cnn.layer.FullyConnectedLayer')
    newLearnableLayer = fullyConnectedLayer(numClasses, ...
        'Name','new_fc', ...
        'WeightLearnRateFactor',20, ...
        'BiasLearnRateFactor',20);


elseif isa(learnableLayer,'nnet.cnn.layer.Convolution2DLayer')
    newLearnableLayer = convolution2dLayer(1,numClasses, ...
        'Name','new_conv', ...
        'WeightLearnRateFactor',20, ...
        'BiasLearnRateFactor',20);
end

lgraph = replaceLayer(lgraph,learnableLayer.Name,newLearnableLayer);

newClassLayer = classificationLayer('Name','new_classoutput');
lgraph = replaceLayer(lgraph,classLayer.Name,newClassLayer);


layers = lgraph.Layers;
connections = lgraph.Connections;

layers(1:10) = freezeWeights(layers(1:10));
lgraph = createLgraphUsingConnections(layers,connections);
end
```

ANEXO II. Codigo de Matlab. Detección de objetos con YOLOv2.

```
clear
clc

load("googlenetaugmented.mat",'net');

%% Importing data

% sort data
rng(0);
data = load('gTruth2.mat');
gTruth = data.gTruth2;
% Add the full path to the local vehicle data folder.
gTruth.imageFilename = fullfile(gTruth.imageFilename);


%% Creating datasets

shuffledIndices = randperm(height(gTruth));
idx = floor(0.8 * length(shuffledIndices));
trainingDataTbl = gTruth(shuffledIndices(1:idx), :);
testDataTbl = gTruth(shuffledIndices(idx+1:end), :);

imdsTrain = imageDatastore(trainingDataTbl.imageFilename);
imdsTest = imageDatastore(testDataTbl.imageFilename);

bldsTrain = boxLabelDatastore(trainingDataTbl(:, 2:end));
bldsTest = boxLabelDatastore(testDataTbl(:, 2:end));

trainingData = combine(imdsTrain, bldsTrain);
testData = combine(imdsTest, bldsTest);


%% Create YOLOv2 network
% Input size for detector.
imageInputSize = [224 224 3];
% define classes
numClasses = width(gTruth)-1;
% estimate anchor boxes

%%
preprocessedTrainingData =
transform(trainingData,@(data)preprocessData(data,imageInputSize));
preprocessedTestData = transform(testData,@(data)preprocessData(data,imageInputSize));

% augment data
augmentedTrainingData = transform(preprocessedTrainingData,@augmentData);

%% Define YOLO v2 ObjectDetector

numAnchors = 5;
trainingDataForEstimation =
transform(trainingData,@(data)preprocessData(data,imageInputSize));
[anchorBoxes, meanIoU] = estimateAnchorBoxes(trainingDataForEstimation, numAnchors);
%
%%
% define feature extraction network
featureExtractionNetwork = net;
featureLayer = 'inception_4d-output';
lgraph =
yolov2Layers(imageInputSize,numClasses,anchorBoxes,featureExtractionNetwork,featureLayer
);

% train YOLOv2 object detector
options = trainingOptions('adam', ...
        'MiniBatchSize', 16, ....
        'InitialLearnRate',0.001, ...
        'LearnRateSchedule','piecewise', ...
        'LearnRateDropFactor', 0.1, ...
        'LearnRateDropPeriod', 75, ...
        'MaxEpochs',170,...
        'ValidationData', preprocessedTestData, ...
        'Verbose',true, ...
        'Plots','training-progress',...
        'Shuffle','once');

%% Train net
% Train the YOLO v2 detector
```

```
[detector,info] = trainYOLOv2ObjectDetector(augmentedTrainingData,lgraph,options);

%% Clasificacion

data = read(preprocessedTestData);

%%
% Get the image.
tic;
tstart = tic;
I = data{1,1};

[bboxes,scores,labels] = detect(detector,I);
idx = scores > 0.7;
bboxesTop = bboxes(idx,:);
labelsTop = labels(idx,:);

% Display the detections on image.
I = insertObjectAnnotation(I,'rectangle',bboxesTop, cellstr(labelsTop));

figure
imshow(I)
telapsed = toc(tstart);


%% Evaluacion del detector
results = detect(detector,preprocessedTestData,'MiniBatchSize',16);

[aP,recall,precision] = evaluateDetectionPrecision(results, preprocessedTestData);


figure
hold on
for k = 1:numel(recall)
plot(recall{k}, precision{k})
end
hold off
xlabel("Recall")
ylabel("Precision")
title("Precision-Recall Curve")
% % legend(preprocessedTestData.Properties.VariableNames(2:end))

%%

function data = augmentData(A)
data = cell(size(A));
for ii = 1:size(A,1)
    I = A{ii,1};
    bboxes = A{ii,2};
    labels = A{ii,3};
    sz = size(I);

    % Randomly flip image.
    tform = randomAffine2d('XReflection',true,...
    'YReflection',true,...
    'XTranslation',[-300 300],...
    'YTranslation', [-50 50],...
    'Scale',[0.75 1.25]);
    rout = affineOutputView(sz,tform,'BoundsStyle','centerOutput');
    I = imwarp(I,tform,'OutputView',rout);

    % Apply same transform to boxes.
    [bboxes,indices] = bboxwarp(bboxes,tform,rout,'OverlapThreshold',0.25);
    labels = labels(indices);

    % Return original data only when all boxes are removed by warping.
    if isempty(indices)
        data(ii,:) = A(ii,:);
    else
        data(ii,:) = {I, bboxes, labels};
    end
end
end

function data = preprocessData(data, targetSize)
% Resize the images and scale the pixels to between 0 and 1. Also scale the
% corresponding bounding boxes.

for ii = 1:size(data,1)
```

```
    I = data{ii,1};
    imgSize = size(I);

    % Convert an input image with single channel to 3 channels.
    if numel(imgSize) < 3
        I = repmat(I,1,1,3);
    end
    bboxes = data{ii,2};

    I = im2single(imresize(I,targetSize(1:2)));
    scale = targetSize(1:2)./imgSize(1:2);
    bboxes = bboxresize(bboxes,scale);

    data(ii, 1:2) = {I, bboxes};
end
end
```

```
clear
clc

%% Importing data

% sort data
rng(0);
data = load('gTruth2.mat');
gTruth = data.gTruth2;
% Add the full path to the local vehicle data folder.
gTruth.imageFilename = fullfile(gTruth.imageFilename);

%%
load('googlenetaugmented','net');

InitialLearnRate = 0.0008;
LRDropFactor = 0.9;
LRDropPeriod = 10;
SquaredGradientDecayFactor = 0.9;
GradientDecayFactor = 0.95;
MaxEpochs = 20;
numAnchors = 3;
miniBatchSize = 2;
ValidationFrequency = 250;
ExecutionEnvironment = 'auto';

imageInputSize = net.Layers(1,1).InputSize(1:2);

%%
rng(0);
shuffledIndices = randperm(height(gTruth));
idx = floor(0.8 * length(shuffledIndices));
trainingDataTbl = gTruth(shuffledIndices(1:idx), :);
testDataTbl = gTruth(shuffledIndices(idx+1:end), :);

imdsTrain = imageDatastore(trainingDataTbl.imageFilename);
imdsTest = imageDatastore(testDataTbl.imageFilename);

bldsTrain = boxLabelDatastore(trainingDataTbl(:, 2:end));
bldsTest = boxLabelDatastore(testDataTbl(:, 2:end));

trainingData = combine(imdsTrain, bldsTrain);
testData = combine(imdsTest, bldsTest);

%% Preprocessing data

preprocessedTrainingData =
transform(trainingData,@(data)preprocessData(data,imageInputSize));
preprocessedTestData = transform(testData,@(data)preprocessData(data,imageInputSize));

% augment data
augmentedTrainingData = transform(preprocessedTrainingData,@augmentData);

%% Training options

% train Faster RCNN object detector
options = trainingOptions('adam', ...
        'SquaredGradientDecayFactor', SquaredGradientDecayFactor, ...
        'GradientDecayFactor', GradientDecayFactor, ...
        'CheckpointPath', pwd, ...
        'MiniBatchSize', miniBatchSize, ....
        'InitialLearnRate',InitialLearnRate, ...
        'LearnRateSchedule','piecewise', ...
        'LearnRateDropFactor', LRDropFactor, ...
        'LearnRateDropPeriod', LRDropPeriod, ...
        'MaxEpochs',MaxEpochs,...
        'ValidationData', preprocessedTestData, ...
        'ValidationFrequency', ValidationFrequency, ...
        'Verbose',true, ...
        'Plots','training-progress',...
        'Shuffle','every-epoch', ...
        'ExecutionEnvironment', ExecutionEnvironment);
```

```
%% Create a Fast RCNN

lgraph = layerGraph(net);
%
% % Specify the number of classes the network should classify.
numClasses = width(gTruth)-1;
numClassesPlusBackground = numClasses + 1;

% Define the number of outputs of the fully connected layer.
numOutputs = 4 * numClasses;

% Create the box regression layers.
boxRegressionLayers = [
    fullyConnectedLayer(numOutputs,'Name','rcnnBoxFC')
    rcnnBoxRegressionLayer('Name','rcnnBoxDeltas')
    ];

% Add the layers to the network.
lgraph = addLayers(lgraph, boxRegressionLayers);

% Connect the regression layers to the layer named 'avg_pool'.
lgraph = connectLayers(lgraph,'pool5-drop_7x7_s1','rcnnBoxFC');

% Select a feature extraction layer.
featureExtractionLayer = 'inception_4d-output';

% Disconnect the layers attached to the selected feature extraction layer.
lgraph = disconnectLayers(lgraph, featureExtractionLayer,'inception_4e-1x1');
lgraph = disconnectLayers(lgraph, featureExtractionLayer,'inception_4e-3x3_reduce');
lgraph = disconnectLayers(lgraph, featureExtractionLayer,'inception_4e-5x5_reduce');
lgraph = disconnectLayers(lgraph, featureExtractionLayer,'inception_4e-pool');

% Add ROI max pooling layer.
outputSize = [14 14];
roiPool = roiMaxPooling2dLayer(outputSize,'Name','roiPool');
lgraph = addLayers(lgraph, roiPool);

% Connect feature extraction layer to ROI max pooling layer.
lgraph = connectLayers(lgraph, featureExtractionLayer,'roiPool/in');

% Connect the output of ROI max pool to the disconnected layers from above.
lgraph = connectLayers(lgraph, 'roiPool','inception_4e-1x1');
lgraph = connectLayers(lgraph, 'roiPool','inception_4e-3x3_reduce');
lgraph = connectLayers(lgraph, 'roiPool','inception_4e-5x5_reduce');
lgraph = connectLayers(lgraph, 'roiPool','inception_4e-pool');

%% Add region proporsal network (RPN)

% Define anchor boxes.
anchorBoxes = estimateAnchorBoxes(preprocessedTrainingData,numAnchors);

% Create the region proposal layer.
proposalLayer = regionProposalLayer(anchorBoxes,'Name','regionProposal');

lgraph = addLayers(lgraph, proposalLayer);

% Number of feature maps in coming out of the feature extraction layer.
numFilters = 1024;

rpnLayers = [
    convolution2dLayer(3, numFilters,'padding',[1 1],'Name','rpnConv3x3')
    reluLayer('Name','rpnRelu')
    ];

lgraph = addLayers(lgraph, rpnLayers);

% Connect to RPN to feature extraction layer.
lgraph = connectLayers(lgraph, featureExtractionLayer, 'rpnConv3x3');

% Add RPN classification layers.
rpnClsLayers = [
    convolution2dLayer(1, numAnchors*2,'Name', 'rpnConv1x1ClsScores')
    rpnSoftmaxLayer('Name', 'rpnSoftmax')
    rpnClassificationLayer('Name','rpnClassification')
    ];
lgraph = addLayers(lgraph, rpnClsLayers);
```

```
% Connect the classification layers to the RPN network.
lgraph = connectLayers(lgraph, 'rpnRelu', 'rpnConv1x1ClsScores');

% Add RPN regression layers.
rpnRegLayers = [
    convolution2dLayer(1, numAnchors*4, 'Name', 'rpnConv1x1BoxDeltas')
    rcnnBoxRegressionLayer('Name', 'rpnBoxDeltas');
    ];

lgraph = addLayers(lgraph, rpnRegLayers);

% Connect the regression layers to the RPN network.
lgraph = connectLayers(lgraph, 'rpnRelu', 'rpnConv1x1BoxDeltas');

% Connect region proposal network.
lgraph = connectLayers(lgraph, 'rpnConv1x1ClsScores', 'regionProposal/scores');
lgraph = connectLayers(lgraph, 'rpnConv1x1BoxDeltas', 'regionProposal/boxDeltas');

% Connect region proposal layer to roi pooling.
lgraph = connectLayers(lgraph, 'regionProposal', 'roiPool/roi');

% Show the network after adding the RPN layers.
figure
plot(lgraph)
ylim([30 42])

%% Training a Faster RCNN

% Train the faster CNN detector

[trainedDetector,info] =
trainFasterRCNNObjectDetector(augmentedTrainingData,lgraph,options,...
    'NegativeOverlapRange', [0 0.3], 'PositiveOverlapRange',[0.6 1], ...
    'NumRegionsToSample', 16, ...
    'FreezeBatchNormalization', true);

%%
data = read(preprocessedTrainingData);

%%
I = data{1,1};

[bboxes,scores,labels] = detect(trainedDetector,I,'MiniBatchSize',miniBatchSize);
idx = scores > 0.5;
bboxesTop = bboxes(idx,:);
labelsTop = labels(idx,:);

% Display the detections on image.
I = insertObjectAnnotation(I,'rectangle',bboxesTop, cellstr(labelsTop));

figure
imshow(I)

%% Evaluacion del detector
results = detect(trainedDetector,preprocessedTestData,'MiniBatchSize',2,
'ExecutionEnvironment', 'gpu');

[aP,recall,precision] = evaluateDetectionPrecision(results, preprocessedTestData);

figure
hold on
for k = 1:numel(recall)
plot(recall{k}, precision{k})
end
hold off
xlabel("Recall")
ylabel("Precision")
title("Precision-Recall Curve")



%% Auxiliary functions

function data = augmentData(A)
% Apply random horizontal flipping, and random X/Y scaling. Boxes that get
% scaled outside the bounds are clipped if the overlap is above 0.25. Also,
% jitter image color.
```

```
data = cell(size(A));
for ii = 1:size(A,1)
    I = A{ii,1};
    bboxes = A{ii,2};
    labels = A{ii,3};
    sz = size(I);

%     if numel(sz) == 3 && sz(3) == 3
%         I = jitterColorHSV(I,...
%             'Contrast',0.0,...
%             'Hue',0.1,...
%             'Saturation',0.2,...
%             'Brightness',0.2);
%     end

    % Randomly flip image.
    tform = randomAffine2d('XReflection',true,...
    'YReflection',true,...
    'XTranslation',[-300 300],...
    'YTranslation', [-50 50],...
    'Scale',[0.75 1.25]);
    rout = affineOutputView(sz,tform,'BoundsStyle','centerOutput');
    I = imwarp(I,tform,'OutputView',rout);

    % Apply same transform to boxes.
    [bboxes,indices] = bboxwarp(bboxes,tform,rout,'OverlapThreshold',0.25);
    labels = labels(indices);

    % Return original data only when all boxes are removed by warping.
    if isempty(indices)
        data(ii,:) = A(ii,:);
    else
        data(ii,:) = {I, bboxes, labels};
    end
end
end

function data = preprocessData(data, targetSize)
% Resize the images and scale the pixels to between 0 and 1. Also scale the
% corresponding bounding boxes.

for ii = 1:size(data,1)
    I = data{ii,1};
    imgSize = size(I);

    % Convert an input image with single channel to 3 channels.
    if numel(imgSize) < 3
        I = repmat(I,1,1,3);
    end
    bboxes = data{ii,2};

    I = im2single(imresize(I,targetSize(1:2)));
    scale = targetSize(1:2)./imgSize(1:2);
    bboxes = bboxresize(bboxes,scale);

    data(ii, 1:2) = {I, bboxes};
end
end
```