ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA DE AUTOMATIZACIÓN E INFORMÁTICA INDUSTRIAL

EN COLABORACIÓN CON PHOENIX CONTACT S.A.U.

CLASIFICACIÓN AUTOMÁTICA DE FRUTOS SEGÚN MADURACIÓN

D. AINSUA CIANCA, Pedro

D.N.I. 71677899-D

TUTOR: D. GONZÁLEZ DE LOS REYES, Rafael Corsino

Gijón, mayo de 2021

MEMORIA

Índice

1. Resumen

En el presente documento se aborda el problema del control de calidad en industria agrícola, clasificando los frutos que entran a planta según su maduración. El objetivo es automatizar el proceso sustituyendo la supervisión humana por un sistema de visión por computador basado en inteligencia artificial. Se propone una solución basada en redes neuronales convolucionales (CNN) y transfer learning que aprenda a clasificar imágenes de frutos según el grado de maduración que indique su aspecto visual. En cuanto a preprocesado de imágenes, se recurre a una transformación homográfica implementada en Open-CV para obtener una vista cenital del producto y eliminar objetos y fondos indeseados. Por la ausencia de datos etiquetados en este momento del proyecto, se recurre a un dataset de tomates (6800 imágenes) disponible en la red para desarrollar este estudio. Para potenciar dicho dataset se recurre al aumento de datos duplicando los ejemplos de entrenamiento (75 % del total) y se guarda para test el 25 % restante del dataset original. Se entrena a lo largo de 100 epochs utilizando batches de 32 imágenes, consiguiendo una tasa de acierto de clasificación en test que alcanza finalmente el 97 %. Dado este rendimiento del sistema, se califica la solución propuesta como exitosa, demostrando ser una opción válida y se abre la vía futura de entrenar el sistema con datos reales de industria, con objeto de desplegarlo en planta para satisfacer las necesidades de un cliente profesional.

2. Introducción

2.1. Motivación

Phoenix Contact S.A.U., corporación en la que el autor de este proyecto realiza sus prácticas, es un fabricante alemán de productos de electrotécnia y automatización que además, ofrece soluciones completas de automatización a sus clientes con desarrollos que aprovechan las posibilidades de sus propios dispositivos. El grupo empresarial se compone de 12 empresas y presenta sedes por todo el mundo, 50 filiales y alrededor de 30 representaciones.

En el año 2017, Phoenix ocupó el duodécimo puesto en el ranking de estimación de ventas mundiales en el mercado de PLCs, por detrás de gigantes del sector como Siemens, Rockwell, Schneider o Mitsubishi y compitiendo al nivel de grandes fabricantes como Panasonic, Hitachi, Bosch o ABB entre otros.

Phoenix España tiene sus oficinas centrales en el Parque Tecnológico de Asturias, en Llanera, sede en la que se desarrollan las prácticas de máster del autor, enmarcadas en el departamento de Industry Management and Automation (IMA). Durante dichas prácticas, Phoenix entra en un proyecto de industria 4.0 para una almazara en la que se procesa aceituna para la fabricación de aceite. Dicho proyecto se basa en la automatización del control de calidad de aceituna en cuanto a maduración, procedimiento que se venía haciendo hasta el punto mediante maquinaria gobernada por personal técnico.

El sector agrícola, seguramente por ser uno de los oficios más antiguos del ser humano, siempre ha tenido un carácter muy conservador y tradicional. A partir de los años 50 del siglo pasado, la agricultura de autoconsumo predominante en gran parte del Estado fue ampliando miras y renovando técnicas, herramientas e infraestructuras para ocupar un lugar en el mercado nacional. La agricultura extensiva, que ya existía en algunas zonas como el sur de la península, fue tomando fuerza y saltando fronteras con la exportación. Con el retorno de la democracia y la entrada en la UE, el campo accede a fondos y ayudas que asistieron las fuertes inversiones necesarias para la ya imprescindible modernización del sector. Hoy en día, España es una de las principales potencias agrícolas de Europa y ha de seguir apostando por el avance tecnológico para mantenerse como un referente.

Una tarea de producción agraria con gran potencial para automatizarse es el control de calidad. Además de valores nutricionales o sanitarios, el cliente final de producto fresco demanda una apariencia determinada, lo cual repercute en el precio que está dispuesto a pagar. Por ello, es necesario determinar calidades para separar la materia prima y diferenciar empaquetado, etiquetado, canales de venta, etc. Aquellos productos que no se venden frescos sino que se procesan, como los jugos, aceites, harinas o productos más elaborados y sus respectivos procesos, también se ven afectados por el estado de las distintas remesas de frutos. Maduración, suciedad, tamaño o conservación determinan precios de compra al agricultor y las necesidades de procesamiento posterior.

Dicha tarea, se ha realizado comúnmente a mano o al menos, mediante maquinaria gobernada por un profesional al cargo de las decisiones. Las tecnologías enmarcadas bajo el concepto de "Industria 4.0", pueden ser de alto aprovechamiento para automaticación de procesos como este, pasando de ejecuciones manuales o mecanizadas bajo gobierno o

vigilancia humana, a sistemas autónomos que sean capaces de trabajar por si mismos con altas tasas de rendimiento.

El concepto de Industria 4.0 se refiere a la evolución de la misma, principalmente mediante técnicas de conectividad, automatización e ingeniería de datos para representación, análisis y aprendizaje automático. Dichos campos de estudio pese a ser distintos, están netamente relacionados. La conectividad (IIoT) permite recolectar el dato de forma automática y sin acceso físico a los dispositivos, también permite enviar órdenes de automatización a los mismos. La automatización puede alimentarse con las conclusiones que arroja inteligencia artificial (IA) para predecir lo que va a ocurrir y todo ello, puede mejorarse representando los datos obtenidos, viendo las relaciones, la estructura de los datos como se hace por ejemplo con t-Distributed Stochastic Neighbor Embedding (tSNE) para reducción de dimensionalidad y persiguiendo su optimización.

Los beneficios de dichas técnicas son especialmente notables en aquellas tareas muy arcaicas, que se basan en la presencia de uno o varios profesionales observando y tomando decisiones basadas en la experiencia y el conocimiento del producto y el proceso, contando inevitablemente con el error humano y los límites de su velocidad de reacción. La automatización de tales tareas no supone desprenderse de ese personal, todo lo contrario. El profesional experimentado es vital para desarrollar los proyectos y también para controlar toda su vida útil, evolutivos, correctivos y métricas de rendimiento, pudiendo desprenderse de sus antiguas tareas y realizando otras más relevantes y con mayor productividad, donde la intervención humana sea más necesaria. A continuación se profundiza en el problema concreto que se desea resolver y se indica la arquitectura básica del sistema desarrollado.

2.2. Problema

En temporada de recogida de aceituna (de octubre a enero), los agricultores realizan jornadas de recolección en la mañana y de tarde llevan los frutos a la almazara para venderlos al fabricante de aceite. Dichos frutos entran por cintas de transportadores a planta para ser limpiados y almacenados. El rendimiento graso de las aceitunas varía notablemente según el grado de maduración de las mismas, por lo que se almacenan en silos separados frutos con grados de maduración distintos para optimizar el proceso. Además, el precio a pagar al agricultor también varía, siendo más alto en caso de aceitunas de altos rendimientos grasos (muy maduras, color negro) y menor el caso opuesto (poco maduras, color verde). Como se puede apreciar en la figura 1 los estados de maduración se reflejan visualmente como variaciones de coloración paulatinas del verde al negro pasando por tonos marrones más o menos extendidos por la oliva.

Figura 1: Diferencias visuales de maduración de oliva. Cañas en [1]

El cliente desea categorizar el producto entrante en 3 grupos: frutos muy maduros (predomina el negro), frutos muy poco maduros (predomina el verde) y por último, grados intermedios (marrones parciales o totales poco oscuros). Se entiende que tales características son suficientemente notables como para que un sistema de visión artificial con IA pueda diferenciarlas, lo que motiva el estudio de viabilidad que abarca el presente TFM de cara a confirmar dichas premisas. Para ello, se propone una solución con objeto de averiguar si cumple con las necesidades del cliente. El principal objetivo es evitar el almacenamiento en el mismo silo de producto con grados de maduración totalmente extremos, ya que se trata de un suceso indeseado que se da ocasionalmente con el procedimiento actual. Se pretende también, eliminar el error humano de fatiga así como objetivar la clasificación, ventajas de un sistema informático frente a la intervención humana.

Hay ciertos requisitos obligados por el cliente que el estudio de viabilidad ha de respetar, de cara a ser lo más aprovechable posible más adelante en el proyecto real. El cliente fija el modelo de cámara y su ubicación, fija el hardware informático y sistema operativo sobre el que ha de correr el software: PC industrial Windows sin GPU y también, restringe el uso de lenguajes de programación a aquellos de uso gratuito tanto de licencias como de bibliotecas.

Para la obtención de datos se cuenta con una cámara de videovigilancia web. Sin duda existen muchas tecnologías de captura utilizables en control de calidad, como pueden ser sensores de imagen retroiluminados, tecnología CMOS o cámaras multiespectrales. Dado que el cliente fija el modelo de cámara (ya instalada en planta) el estudio de viabilidad ha de adaptarse para ser fiel al problema a resolver. De todas formas, no es un mal sensor ya que se trata de un CMOS 1080p a 30 fps, siendo los sensores CMOS una alternativa de alto coste y gran calidad.

2.3. Estructura del sistema

La solución de automatización se basa en el uso de una cámara web que monitorice las entradas de materia prima, trasladando imágenes a un equipo remoto donde se ejecute un código capaz de determinar que grado de maduración tiene el producto entrante en cada instante, para tal labor es adecuado el uso de un modelo de inteligencia artificial para predicciones relativas a la clasificación de imágenes.

El sistema propuesto para solucionar el problema se subdivide a su vez en 3 subsistemas, los cuales se detallan a continuación definiendo sus funciones, el flujo de información entre los mismos, así como las entidades responsables de su desarrollo en caso de estar fuera del alcance del TFM.

En el diagrama de bloques de la figura 2 se detalla gráficamente la arquitectura indicada. Una vez se captan los datos y se preparan de forma conveniente en el bloque 1, éstos alimentan los dos subsistemas restantes. Para el bloque 3 conforman el conjunto de características, típicamente conocido en ciencia de datos con el símbolo "X". En el bloque 2 se etiquetan los datos procedentes del bloque 1 y se genera el vector objetivo, comúnmente representado como "Y". Una vez el bloque 3 cuenta con el conjunto de características y con el vector objetivo, ya puede comenzar a entrenar el modelo deseado.

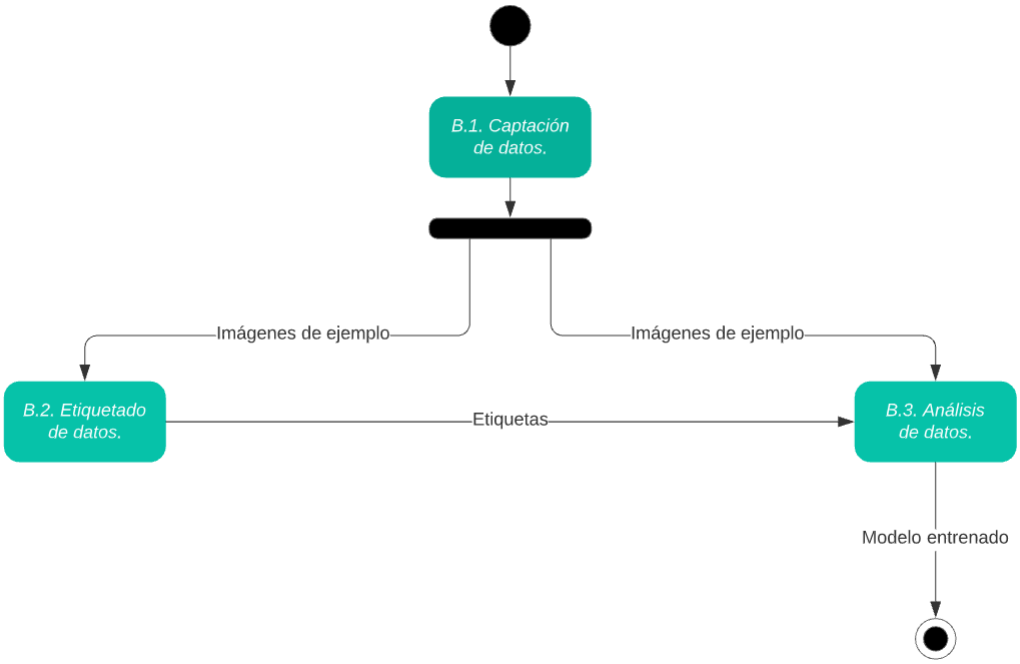

Figura 2: Diagrama de bloques.

2.3.1. Bloque 1: Captura de datos.

Este bloque se encarga de obtener los datos y aplicar las operaciones iniciales necesarias para su aprovechamiento por los bloques siguientes. El layout de captación de datos viene impuesto y acarrea operaciones de preprocesado, pues la cámara no ofrece una vista cenital del producto en cinta e incluye en las imágenes múltiples objetos y fondos sin interés (ver figura 3). Se propone incluir un preprocesado de imágenes, transformando las mismas para obtener una visión lo más aproximada a la vista cenital del producto, con el menor número de elementos superfluos que sea posible.

Figura 3: Imagen sin preprocesado.

Una vez conseguido un ajuste y preprocesado correcto, se ha de realizar un extenso monitoreo de entradas de producto para obtener el dataset de entrenamiento y test para los siguientes bloques. Dicho monitoreo se basa en la captación de vídeo para luego cortarlo en frames y componer el conjunto de ejemplos.

2.3.2. Bloque 2: Etiquetado de datos.

Como segundo bloque del sistema, será necesario un etiquetado de los datos obtenidos por el bloque anterior. Dado que se precisa un amplio conocimiento del proceso para realizar dicha tarea, este bloque se externaliza a un equipo de ingeniería agrónoma contratado por el cliente, que se encargará de determinar qué ejemplos de los recogidos corresponden a cada una de las categorías que el sistema de IA debe aprender.

2.3.3. Bloque 3: Análisis de datos.

El tercer y último bloque es el encargado del Machine Learning. Una vez se cuente con conjuntos de ejemplos de cada categoría, este sistema ha de aprender de ellos para determinar ante una nueva imagen, qué grado de maduración tienen los frutos presentes en ella. Una vez entrenado ha de ser capaz de clasificar las nuevas imágenes con que se alimente el sistema.

Dado que el planning del proyecto real se extiende mucho más allá de las fechas previstas para este TFM, para el estudio de viabilidad de este bloque se utilizará un dataset de pruebas obtenido de la red, basado en cambios de coloración de tomates los cuales presentan unas características diferenciales muy similares a las vistas en las aceitunas.

2.4. Planificación temporal

En el diagrama de la figura 4 se presenta la planificación semanal del proyecto a realizar durante las prácticas en empresa. Se consideran 5 horas de trabajo por día hábil y 5 días semanales laborables, en total 25 horas semanales, condiciones presentes en el convenio entre universidad, alumno y empresa para dichas prácticas. La semana 1 comienza el 28 de septiembre de 2020, terminando la presente planificación el día 19 de marzo de 2021 se la semana 23.

Tarea / Semana	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Recogida de objetivos	✓	✓																					
Estudio del layout y el hardware implicado		✓	✓																				
Definición de bloques			✓	✓																			
Estudio de alternativas para bloque 1					✓	✓																	
Desarrollo bloque 1							✓																
Pruebas bloque 1								✓															
Conclusiones bloque 1								✓															
Estudio de alternativas para bloque 2									✓	✓	✓												
Desarrollo bloque 2												✓	✓	✓	✓								
Pruebas bloque 2																✓							
Conclusiones bloque 2																	✓						
Documentación																		✓	✓	✓	✓		
Correcciones de documentación																						✓	✓

Figura 4: Planificación semanal de tareas del proyecto.

2.5. Contenido del documento

En los sucesivos apartados se analizarán primeramente los precedentes del problema propuesto, viendo soluciones que se hayan documentado para casos cercanos y analizándolas críticamente. En base a toda la información manejada y a los conocimientos del campo de estudio, se propondrá después un método describiéndolo en detalle. Tras ello, tocará describir qué experimentos se han realizado con dicho método, que herramientas y datos se han utilizado y cómo se ha procedido. Dichos experimentos arrojarán unos resultados, los cuales se presentarán para luego finalizar con una discusión que compruebe si se han logrado los objetivos, determinando la adecuación de todas las decisiones tomadas para el fin establecido.

De los tres bloques del sistema presentados anteriormente se abordan dos ya que el etiquetado lo realiza una entidad externa, por tanto los apartados de trabajos previos, método y experimientos se repetirán de forma separada para cada uno de esos dos bloques. La discusión, las conclusiones y el presupuesto final se realizarán de forma conjunta para ambos subsistemas.

3. Bloque 1: Captura de datos.

3.1. Problema

Para la recolección de datos, en el proyecto real de Phoenix Contact se ha dispuesto una cámara sobre la cinta de transportadores por la que entra el producto en planta. Dicha cinta, como se aprecia en la figura 5, es ascendente y el plano en que está contenida no es perpendicular al eje de la cámara, por lo que se busca un preprocesado de corrección de la perspectiva encargado de transformar las imágenes en una visión cenital del producto sobre la cinta. Dicha operación ofrece un beneficio extra, ya que se ven recortados numerosos elementos del fondo, objetos mal definidos y zonas de iluminación irregular que no tienen interés y podrían comprometer el rendimiento de la red neuronal.

Figura 5: Posición cámara.

3.2. Estado del arte

Durán Ferreras en [2] expone distintos algoritmos de rectificación de imágenes aéreas, los cuales se sintetizan a continuación. Dichos algoritmos se consideran en base a su utilidad para corregir imágenes como las captadas por aviones, drones o satélites. Aunque en el caso del presente proyecto las imágenes vienen de una cámara fija terrestre, si que existe ámbito común en el sentido de que se toman en altura respecto de los objetos de interés con una cierta perspectiva.

Rectificación polinómica: Se define una relación polinómica entre los pixels de la imagen original y los de la imagen transformada deseada, cuyos coeficientes se calculan en base a puntos de referencia en la imagen original, de los cuales se conoce la posición que han de tener en la imagen transformada. Se consigue eliminar el efecto de la perspectiva, fin deseado en el proyecto abordado. Decidir el orden del polinomio es crítico, pues determina la calidad del resultado final afectando negativamente tanto cuando se excede como cuando se reprime su valor.

Rectificación proyectiva: Transformación geométrica la cual requiere ajustar 8 parámetros. Algoritmo útil para procesar vistas aéreas de terreno plano y para imágenes de fachadas de edificaciones.

Rectificación basada en modelo del sensor: Se modela el sensor para luego realizar transformaciones sin necesidad de identificar puntos de referencia. Existen modelos físicos muy rigurosos y de alta precisión, los cuales también suponen un gran gasto computacional y también un escalado costoso en el caso de aumentar el número de sensores, ya que cada uno necesitará su propio modelo físico. Como ejemplo de este tipo de modelos se destaca la ecuación de colinearidad. También se utilizan modelos generalizados no basados en parámetros físicos, como es la calibración, con los que se controla la complejidad del modelo y la sencillez de modelado.

Homografía Se trata de una aplicación entre dos planos en que a cada punto del plano origen le corresponde un punto en el plano destino, ocurriendo de igual manera en el caso de las rectas. La pertenencia de un punto a una recta también se conserva en el plano destino. Esta transformación se asienta en el supuesto de que el terreno fotografiado es plano y el plano al que se proyecta es paralelo a la superficie. Para el cálculo se obtiene una matriz de transformación "H" que se aplica a los puntos de la imagen original para dar lugar a la imagen transformada. Mediante este algoritmo se consigue asignar el nivel de gris de un punto de la imagen original al punto adecuado en la imagen rectificada.

Durán Ferreras en [2] decide utilizar la homografía por ser un modelo sencillo que no requiere calibración de la cámara, adecuado para un procesado automatizado para gran número de imágenes. En la figura 6 se presenta una imagen de referencia ya rectificada que sirve de control para los experimentos siguientes.

Figura 6: Imágen rectificada de referencia. Durán Ferreras en [2].

El autor busca recuperar las orientaciones cartográficas del terreno como objetivo del procesado. En la figura 7 se puede apreciar que en las imágenes aéreas tomadas y no procesadas (lado izquierdo de la figura), la pista de aterrizaje no tiene la orientación de la imagen de control. Sin embargo, las corregidas (lado derecho), despreciando una rotación de 90 grados aplicada por el autor al formatear el documento, sí que mantiene la orientación de pista de la imagen de control. Mediante la homografía ha sido posible recuperar las orientaciones de las imágenes respecto a un sistema de referencia común.

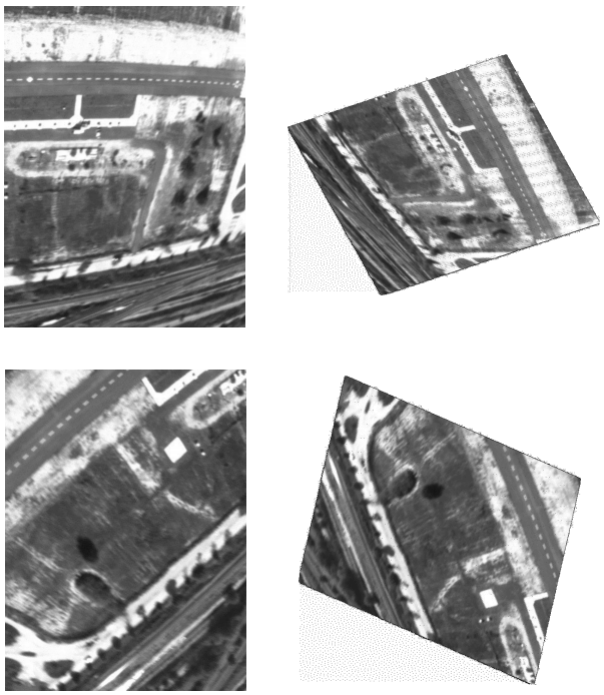

Figura 7: Imágenes aéreas rectificadas mediante homografía. Durán Ferreras en [2].

En la figura 8 se aprecia otro efecto como es la deformación por perspectiva, entre la imagen captada (lado izquierdo) y su versión procesada (lado derecho), se presentan cambios en el área de las parcelas delimitadas por los caminos así como de la longitud relativa entre un camino y otro.

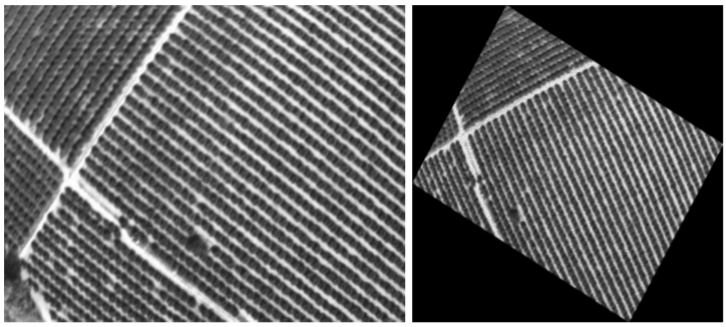

Figura 8: Imágen aérea corregida mediante homografía. Durán Ferreras en [2].

Se trata de una aplicación distinta a la necesaria en este proyecto, pero ofrece un buen ejemplo de rectificación de imagen, centrándose en el uso de la homgrafía para su resolución. Los condicionantes relativos a contar con un gran número de imágenes a procesar y la necesidad de hacerlo automática y ligeramente en cuanto a coste computacional es común entre ambos proyectos por lo que las decisiones tomadas serán de interés en este estudio.

Haciendo énfasis en el caso de la homografía, Maulion presenta interesantes ejemplos en [3]. Se trata de casos en que se busca una visión cenital de un objeto o conjunto de objetos, problema muy cercano al que se busca dar solución.

En el estudio se describe homografía como proyección de puntos de un plano a otro, mostrando las ecuaciones básicas y su representación gráfica espacial (ver figura 9 y Ec1). Se representan dos planos en el espacio, uno de origen (representado con símbolo y coordenadas prima) y otro de destino al que se desean proyectar los puntos.

$$s \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad (Ec1)$$

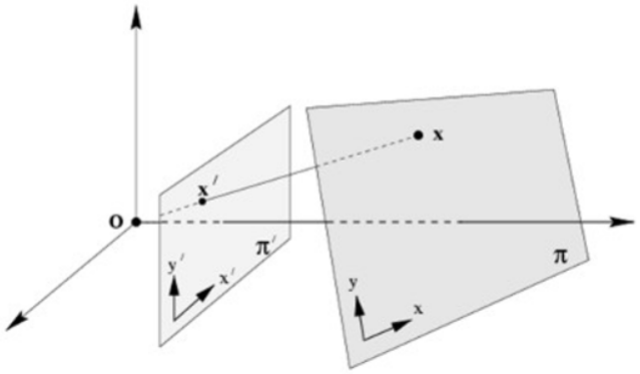

Figura 9: Definición matemática y gráfica de la homografía. Maulion en [3].

El autor toma un mínimo 4 puntos de referencia en el plano origen y sus puntos correspondientes del plano destino, con objeto de averiguar qué relación matemática se presenta entre los pares de puntos, relación que describirá la matriz de homografía que pretende obtener. En algunos casos la selección de los puntos de referencia es sencilla, como es el caso de las esquinas del tablero de ajedrez de la figura 10 y en otros casos como el de la figura 11, al no estar la pista de baloncesto completa en la imagen, el autor se apoya en una plantilla (figura 12) y selecciona puntos de corte de líneas reglamentarias del suelo de la cancha para puntos origen y destino.

Figura 10: Imagen de partida de tablero de ajedrez. Maulion en [3].

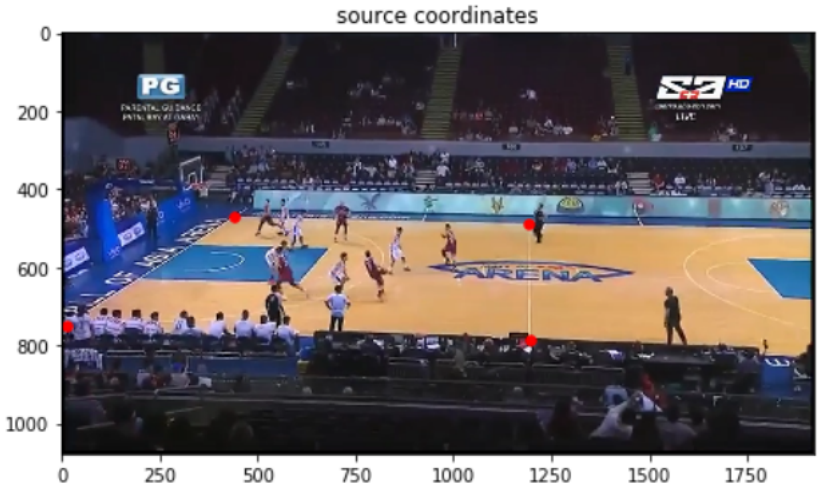

Figura 11: Imagen de partida de cancha de baloncesto. Maulion en [3].

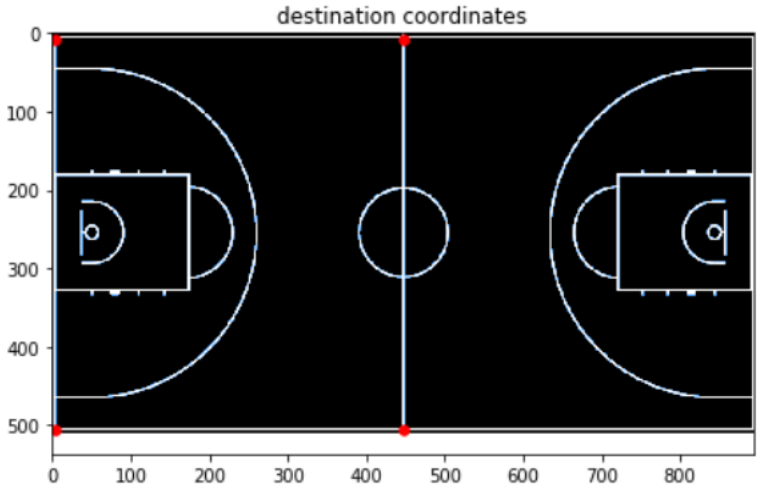

Figura 12: Plantilla de cancha de baloncesto. Maulion en [3].

Los resultados son satisfactorios, en ambos casos se consigue una vista muy cercana a la cenital (figuras 13 y 14). El recurso de la plantilla de la cancha, aprovechando las medidas estándar que caracterizan a estos emplazamientos, resulta muy interesante ya que simplifica algo que de otra manera requeriría esfuerzo, sobre todo para determinar unos puntos destino que no causen una deformación añadida al resultado final.

En los jugadores de baloncesto (figura 14) se aprecia notablemente ese efecto indeseado que viene de modelar como plano algo que no lo es, ya que los jugadores sobresalen ampliamente sobre la cancha, siendo ésta el supuesto plano origen. Dependerá del propósito del proyecto y del tamaño de los objetos en altura sobre el plano que estas deformaciones sean o no críticas.

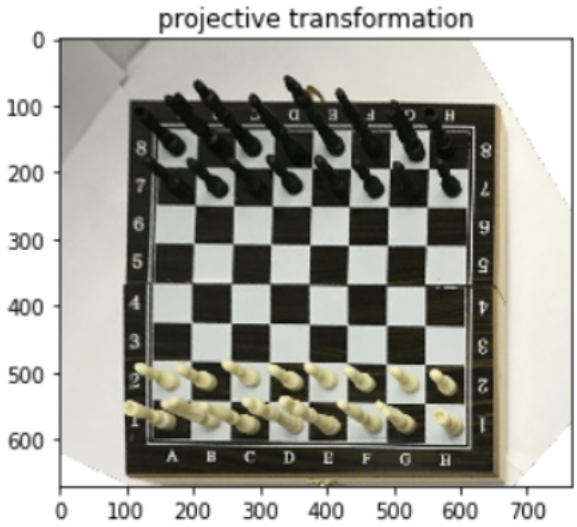

Figura 13: Imagen transformada de tablero de ajedrez. Maulion en [3].

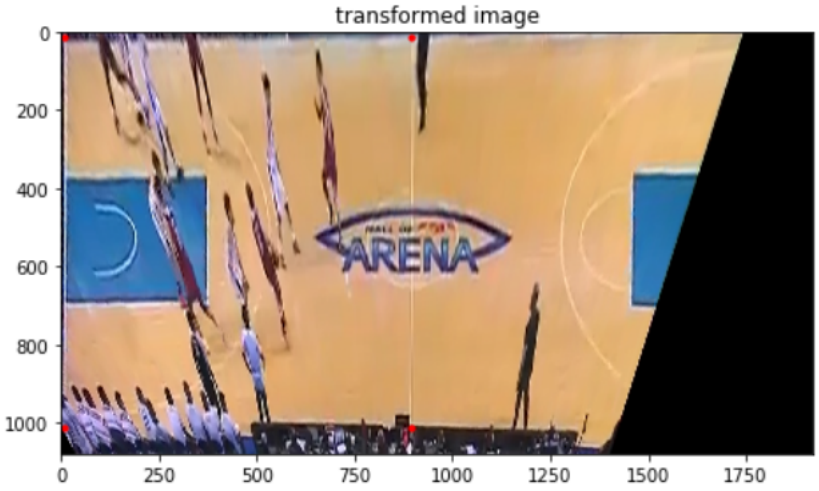

Figura 14: Imagen transformada de cancha de baloncesto. Maulion en [3].

Se trata sin duda de ejemplos de alto aprovechamiento para el problema a resolver por el bloque 1 que serán tenidos muy en cuenta en apartados posteriores, ya que cumplen con las necesidades y anticipan los problemas con los que habrá que lidiar más adelante.

3.3. Método

La solución propuesta se basa en hallar una transformación homográfica que proyecta puntos de un plano no perpendicular a la cámara (el de la cinta) al plano imagen. Como se indica en la Ec2, se trata de buscar una matriz de transformación "H" capaz de transformar los puntos del plano de la cinta de coordenadas p1 = (X,Y), en puntos del plano imagen expresados en las coordenadas homogéneas p2 = (u, v, w). Tal como se aprecia en la Ec3, la matriz "H" tiene 8 incógnitas y puede estimarse con un mínimo de 4 parejas de puntos de cada uno de los dos planos. La recolección de los pares de puntos se basa en buscar un punto fijo en la imagen original, que a poder ser delimite el área de

interés y emparejarlo con un punto de la imagen transformada deseada en el cual debería transformarse, repitiendo el proceso 4 veces.

$$\begin{bmatrix} \widetilde{u} \\ \widetilde{v} \\ \widetilde{w} \end{bmatrix} = \begin{bmatrix} H \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \qquad (Ec2)$$

$$\begin{bmatrix} \widetilde{u} \\ \widetilde{v} \\ \widetilde{w} \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} h_1^T \\ h_2^T \\ h_3^T \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \qquad (Ec3)$$

En las ecuaciones Ec4 y Ec5 se indica la conversión de las coordenadas homogéneas de los puntos del plano imagen a coordenadas cartesianas, dividiendo las dos primeras componentes por la tercera, así como las operaciones vectoriales devenidas de ecuaciones anteriores necesarias para su cálculo. En las ecuaciones Ec6 y Ec7 se expresa la proyección de puntos en ambas direcciones una vez conocida la matriz de transformación "H" (la cual admite inversa), entendiendo el punto "p2" como un punto del plano imagen y el punto "p1" como un punto del plano de la cinta (en la nomenclatura del problema que nos ocupa).

$$u = \frac{\widetilde{u}}{\widetilde{w}} = \begin{bmatrix} \widetilde{p_1}^T & 0_{1\times 3} & -u\widetilde{p_1}^T \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} \qquad (Ec4)$$

$$v = \frac{\widetilde{v}}{\widetilde{w}} = \begin{bmatrix} 0_{1\times 3} & \widetilde{p_1}^T & -v\widetilde{p_1}^T \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} \qquad (Ec5)$$

$$\widetilde{p_2} = H\widetilde{p_1} \qquad (Ec6)$$

$$\widetilde{p_1} = H^{-1}\widetilde{p_2} \qquad (Ec7)$$

De cara a estimar la matriz "H", se conforma la matriz "A" de coeficientes expuesta en la Ec8. Para componer "A" se utilizan los 4 puntos de cada plano necesarios para estimar "H" y se completa la matriz con submatrices de ceros con una disposición tal cual se indica en la ecuación.

$$A = \begin{bmatrix} \widetilde{p_1}^T & 0_{1\times3} & -u\widetilde{p_1}^T \\ 0_{1\times3} & \widetilde{p_1}^T & -v\widetilde{p_1}^T \end{bmatrix} \qquad (Ec8)$$

La resolución del sistema puede alcanzarse con descomposición en valores singulares SVD (Ec9), siendo "D" una matriz diagonal compuesta de los valores propios de "A" y las matrices "U" y "V" matrices ortonormales en que las columnas, se corresponden con los vectores singulares asociados a los valores singulares de "A". Los resultados presentes en "V" se utilizan componer finalmente "H" ordenados tal como se aprecia en la Ec10.

$$A = UDV^T \qquad (Ec9)$$

$$H = \begin{bmatrix} V(1,9) & V(4,9) & V(7,9) \\ V(2,9) & V(5,9) & V(8,9) \\ V(3,9) & V(6,9) & V(9,9) \end{bmatrix} \qquad (Ec10)$$

Para obtener la imagen transformada, solo resta aplicar la matriz de transformación a todos los puntos de la imagen original, de la forma indicada en Ec6, para obtener una imagen completamente transformada.

3.4. Experimentos

Para desarrollar la solución propuesta se ha preparado un entorno Windows 10 profesional. Se ha contado con la distribución de Python "Anaconda", usando de forma intensiva sus paquetes Jupyter Notebook y Spyder. Sobre dichas herramientas se ha procedido a desarrollar código Python orientado a correr en entorno Windows.

En cuanto a bibliotecas externas, el desarrollo se basa en la utilización de Open-CV, todo un referente en cuanto a procesamiento de imagen. Otras bibliotecas básicas como Matplotlib y Numpy sirven de apoyo para cálculos y representaciones de resultados.

Los datos utilizados han sido las primeras imágenes obtenidas del proyecto real aún sin etiquetado, ya que en esta etapa no es requerido un categorizado de las mismas. En la figura 15 puede verse un ejemplo de captura dado el layout de cámara predefinido por el cliente. Se toman 4 puntos (mínimo necesario) de referencia de la imagen original, puntos que encierren el producto sobre los "escalones" más cercanos a la cámara, región de interés, ya que los más alejados no se ven con detalle por la distancia y el movimiento de la cinta. Dado que los raíles verdes laterales no se mueven y tienen muescas intermitentes, sirven para fijar estos 4 puntos que van representados en azul en la imagen. Luego los 4

puntos del plano destino a emparejar con los ya vistos serán las 4 esquinas de la imagen, ya que el propósito es que la región de interés ocupe toda la vista.

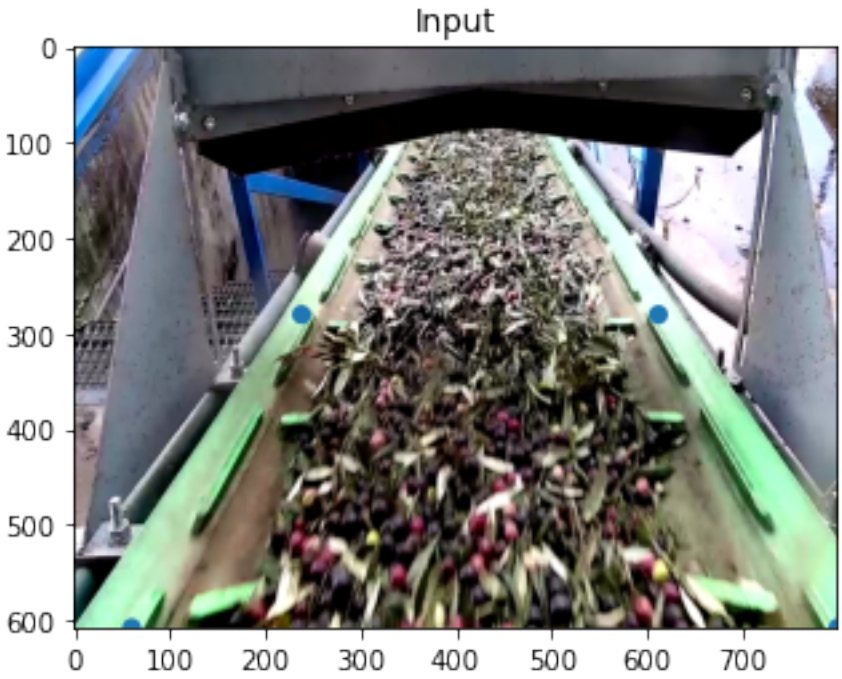

Figura 15: Vista original y 4 puntos origen.

Antes de procesar con Open-CV es necesario tomar una pequeña precaución. La información de color de una imagen cargada suele estar ordenada matricialmente de la forma "r,g,b" sin embargo, los métodos Open-CV a utilizar esperan una ordenación "b,g,r". Para evitar distorsiones indeseadas del color, problema inmensamente sensible para este proyecto, será necesario modificar ese orden mediante métodos propios de la biblioteca antes de procesar y revertir posteriormente al ordenamiento de información de color original para exportar y representar la imagen.

No se trata de un problema crítico de cara a hallar la matriz de homografía, pero sí lo es de cara a aplicarla para transformar las imágenes ya que se ha de recalcular el color en cada pixel de la imagen transformada teniendo en cuenta el color presente en la imagen original, teniendo también muy presente que la veracidad de los datos de color es de máxima importancia para el bloque de análisis de datos.

Una vez definidas las 4 parejas de puntos se halla la matriz de transformación "H" en base a ellos y una vez obtenida, se aplica la misma a todos los puntos de la imagen original para generar la imagen con perspectiva corregida.

3.5. Resultados

En la figura 16 se representa una comparativa de la imagen original y la transformación. En azul se representan en una y otra imagen los 8 puntos emparejados, los cuales ayudan a comprender el efecto obtenido. Como se puede apreciar en la parte derecha, se consigue una vista muy cercana a la cenital y todos los elementos presentes, se reducen a los escalones más cercanos de la cinta con producto y los raíles que contienen los puntos

de referencia. El número de elementos indeseados, fondos irregulares y zonas cambiantes ajenas a la cinta se ha reducido casi por completo.

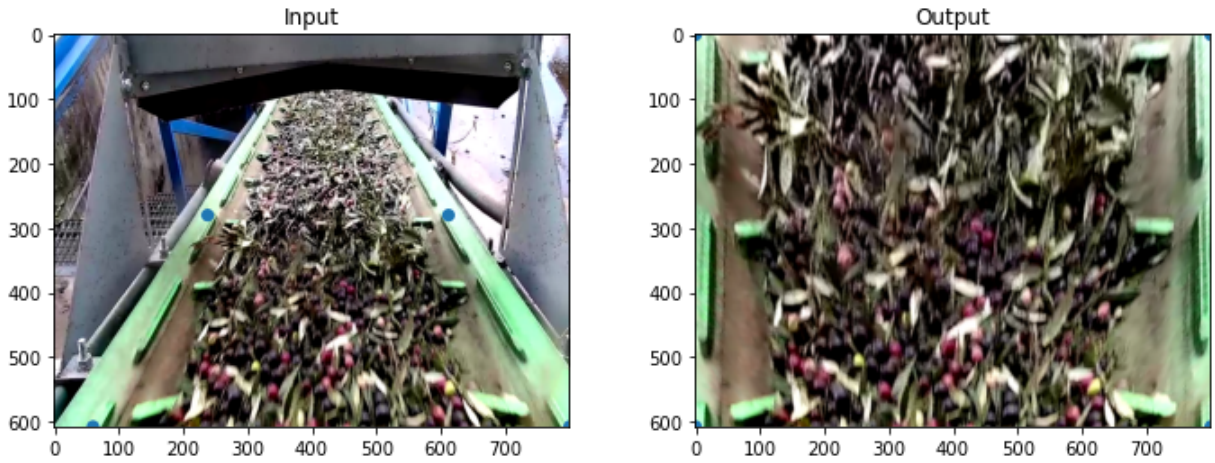

Figura 16: Comparativa entre las imágenes original (izquierda) y transformada (derecha).

4. Bloque 3: Análisis de datos.

4.1. Estado del arte

4.1.1. Redes neuronales convolucionales (CNN)

En los estudios previos se presentan múltiples soluciones basadas en redes neuronales convolucionales. De cara a comprender las soluciones propuestas se procede a repasar los principales conceptos de dicha tecnología, así como la función de los principales tipos de capas que componen estas redes.

La arquitectura de red neuronal convolucional (CNN), es un referente hoy en día en el campo de la inteligencia artificial sobre conjuntos de datos de formato imagen. Estas redes son muy utilizadas para procesar imágenes buscando características locales en grupos de pixels como pueden ser contornos, colores y sus variaciones. Cada neurona de la primera capa se encarga de un grupo de entradas, por ejemplo, de una sección de imagen de 3x3 pixels. Luego se pueden ir añadiendo sucesivas capas para detectar más características y aquellas menos evidentes que pueden pasar por alto arquitecturas más simples.

Las convoluciones son aplicaciones de un filtro lineal o kernel que la red va aprendiendo progresivamente durante el entrenamiento. Se dispone de un kernel de un determinado tamaño, por ejemplo 3X3 para imágenes en escala de grises o 3X3X3 para imágenes RGB. Dicho kernel se inicializa con unos valores aleatorios para ir luego optimizándose los mismos durante el entrenamiento mediante la técnica de Backpropagation (realimentaciones hacia atrás), con el objetivo de aprender finalmente unos valores del kernel lo más adecuados para poder etiquetar con éxito nuevos ejemplos entre las clases entrenadas. Se dispone de mútiples filtros o kernels locales que se encargan cada uno de una parte de la imagen, se realiza la convolución pasando el filtro centrado en cada píxel de la sección de imagen y se salvan los resultados.

Luego, sobre esos resultados se aplica una función de activación, por ejemplo ReLu (satura los valores negativos a cero) para obtener finalmente un mapa de características. Estas características extraídas son la "visión"que la red tiene de una imagen, características muy comunes en una de las clases de entrenamiento servirán para que diferencie esa clase de otras y serán claves para clasificar nuevos ejemplos en los que estén presentes.

Como se puede apreciar en la figura 17, los valores del kernel se multiplican uno a uno por sus análogos de la región de imagen donde se aplica y los 9 resultados parciales se promedian. El valor final se coloca en una matriz de resultados (del tamaño de la imagen), en la misma posición en la que se centra el kernel en la matriz de píxels de imagen.

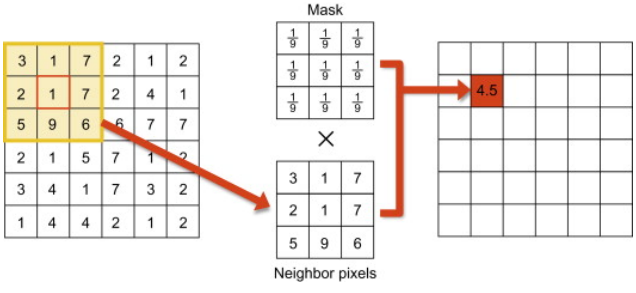

Figura 17: Ejemplo de convolución en escala de grises con filtro 3X3. Jeong et al. en [4].

De cara a profundizar más en las redes CNN y con objeto de alcanzar una mayor comprensión de las soluciones expuestas, a continuación se describen los principales tipos de capas que construyen estas arquitecturas.

InputLayer: Capa de entrada que recibe las matrices en las que se representan las imágenes. Ha de estar convenientemente configurada para el tamaño de imagen en píxels que va a recibir, ya sea el propio de los datos o el resultante tras un reescalado. También ha de estar preparada para la tercera dimensión de dichas matrices, la relativa al color según sea RGB o escala de grises.

Conv2D: Se encarga de extraer características de la imagen. Realiza operaciones de convolución entre una región de la imagen y un filtro (con tamaños coincidentes), dando como resultado un entero. Se va pasando el filtro por toda la imagen y obteniendo los resultados locales, dichos resultados componen la salida de la capa. La activación ReLu hace que los valores negativos se saturen a cero.

MaxPooling2D: Capa de agrupación encargada de la reducción del volumen espacial de las imágenes de entrada. Disminuye el coste computacional de aplicar capas sucesivas sobre los datos. En el caso de VGG-19, la primera capa MaxPooling2D usada pasa de un volumen 100X100X64 a 50X50X128 con una reducción del volumen espacial del 50 %.

Flatten: Capa de aplanamiento que convierte los datos a una dimensión. En el caso de las capas añadidas tras VGG-19, los datos de entrada a la capa Flatten tienen unas dimensiones de 3X3X512, almacenando por tanto de forma matricial 4608 datos numéricos. Tras aplanar se obtiene un conjunto de dimensión 1 y tamaño 4608.

Dense: Capa profundamente conectada, es decir, cada neurona de una capa densa recibe la salida de todas las neuronas de la capa anterior. Es la capa más utilizada en cualquier modelo CNN. En ella se realiza un producto de una matriz de parámetros entrenables por el vector de entradas a la neurona. Sirven para someter la salida de la capa anterior a operaciones de rotación, traslación o escalado. El número de salidas es configurable lo que es de gran utilidad en capas finales de la red para adaptarse al problema abordado.

Dropout: Capa de regularización orientada a prevenir el sobre-entrenamiento de la red. Limita el crecimiento desmesurado del modelo permitiendo que éste sea más flexible ante nuevos datos y no se limite a aprender los destinados a entrenamiento. Dicho efecto se consigue ignorando de forma aleatoria algunas neuronas de la red, despreciando sus contribuciones y no aplicando sus actualizaciones de pesos.

4.1.2. Transfer learning

En el campo de la inteligencia artificial, los trabajos previos no solo sirven de inspiración para los desarrollos, sino que los modelos entranados en otros proyectos pueden integrarse en el código para potenciar el mismo mediante las técnicas de transfer learning. Se trata del aprovechamiento de modelo CNN entrenado que haya sido utilizado para resolver un problema parecido, en este caso de clasificación de imágenes.

Una de las vías es usar dicha red solo re-entrenando su capa de salida, aunque también es posible utilizarla (eliminando su capa de salida) como una subred dentro de una nueva arquitectura, añadiendo nuevas capas tras ella. Este segundo enfoque permite, mayor margen de maniobra para incrementar el tamaño de la red y para tratar un posible sobreentrenamiento mediante regularización.

Las técnicas de transfer learning resultan beneficiosas cuando el volumen de datos empleado para el entrenamiento previo de la red aprovechada es mucho mayor al que se dispone para resolver el nuevo problema. Si se dispone de muchos datos es posible re-entrenar toda la red realizando un "ajuste fino" de los pesos ya inicializados en el pre-entrenamiento. El ajuste fino siempre ha de realizarse con datos del mismo tipo y procedencia que los futuros datos a clasificar en producción.

A continuación, se describen algunos de los modelos pre-entrenados más populares y potentes del estado del arte.

ResNet: Las distintas variantes de "Residual Neural Network" se diferencian entre sí por el dígito de su nomenclatura, relativo al número de capas de profundidad. Se trata de redes neuronales convolucionales entrenadas con más de un millón de datos del banco de imágenes "ImageNet" como clasificador de 1000 categorías. Entre dichas clases se cuentan múltiples objetos cotidianos y razas de animales.

Su característica principal son las conexiones de salto (ver figura 18), basadas en que a una capa "l" no llegue únicamente la salida de la capa anterior "l-1" sino también la de la capa "l-2" saltándose la capa "l-1" que se ve silenciada respecto a su predecesora, la cual ve amplificada la relevancia de su salida. Esta filosofía, aporta beneficios cuando el aumento de la profundidad de redes convencionales deja de beneficiar y empieza a perjudicar el rendimiento. La base teórica de estas redes está inspirada en ciertas capas de neuronas cerebrales humanas, en las que también se producen estas conexiones de salto.

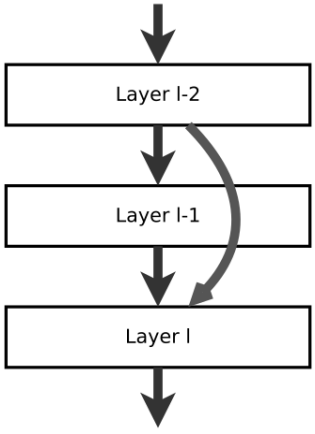

Figura 18: Conexiones de salto de ResNet. Wikipedia contributors en [5].

VGG: Estas arquitecturas fueron propuestas por Simonyan y Zisserman, del Visual Geometry Group de la universidad de Oxford, de ahí sus siglas identificativas. Se trata de modelos que una vez más, difieren en versiones de distinta profundidad de capas e igualmente entrenados con el banco de imágenes ImageNet.

En la figura 19 pueden apreciarse todas las variantes desarrolladas incluidas sus estructuras. Se trata de arquitecturas sencillas, con un número de capas convolucionales 3X3 determinado por la versión VGG, tras ellas una capa de agrupación máxima (maxpool), tres capas totalmente conectadas (FC) y un clasificador (soft-max) como capa final. Las versiones más pequeñas fueron desarrolladas primero y sentando la base inicial sus "hermanas mayores", utilizando el pre-entrenamiento continuamente con cada nueva variante.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figura 19: Varientes de VGG. Goswami en [6].

GoogleNet: Se trata de una arquitectura de red neuronal convolucional profunda de 27 capas en total, construida como variante de Inception Net, otra arquitectura de red neuronal convolucional desarrollada por Google. Enmarcada de nuevo en las competiciones del banco de datos Imagenet y campeona en la competición de 2014 "ILSVRC 2014 classification challenge".

En esta red se incluyen "clasificadores auxiliares" que previenen el sobreentrenamiento y además, se trata de otra solución distinta a la expuesta en Resnet para evitar problemas en el crecimiento de las redes profundas. Estos clasificadores solo trabajan en entrenamiento, realizando clasificaciones en secciones medias de la arquitectura y aportando datos de pérdidas en base a sus predicciones, agregándolos de forma ponderada a las pérdidas totales del entrenamiento.

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Figura 20: Arquitectura de Googlenet. Alake en [7].

AlexNet: Alexnet es una arquitectura de redes neuronales convolucionales muy popular, también entrenada con el banco de datos de Imagenet (ImageNet LSVRC-2010), desarrollada con el objetivo de clasificar 1.2 millones de imágenes entre más de 1000 categorías y quedando por delante de sus competidoras con notable distancia (año 2012).

Se compone de 8 capas, 5 capas convolucionaes y 3 capas FC contando en la última capa con activación softmax (clasificador) para discernir entre las 1000 clases. Los detalles de la arquitectura se presentan en la figura 21.

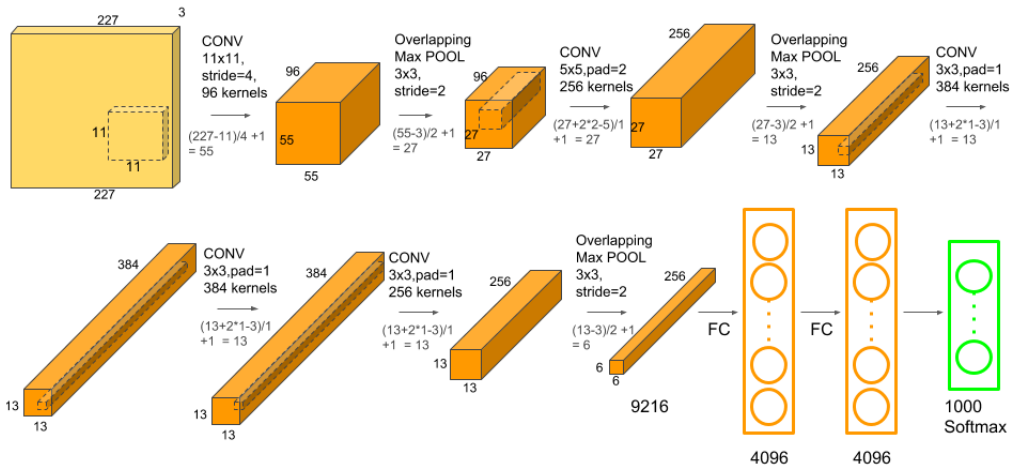

Figura 21: Arquitectura de Alexnet. Kurama en [8].

4.1.3. Estudios previos

Sri et al. abordan en [9] el problema del control de calidad de plátano según su estado de maduración. Para resolver de forma automática dicho problema se establecen dos vías, el análisis químico y la visión artificial. Los análisis químicos suponen desprender tejido del fruto, son invasivos y destructivos, por contra la visión por computador es totalmente inocua para el producto.

Dentro del campo de visión por computador, destacan la técnica de deep learning basada en redes neuronales convolucionales (CNN) como opción novedosa, frente a técnicas clásicas de visión.

En la figura 22 se aprecia la estructura del sistema CNN propuesto, detallando las capas implementadas en la red. Para evitar sobreentrenamiento, los autores aplican técnicas de aumento de datos (traslaciones) sobre las imágenes para generar otras nuevas. Utilizan un 80 % de los datos para entrenamiento, así como el 20 % restante para test y aunque indican que se realiza un pre-procesado a las imágenes antes de alimentar el modelo, no se detalla en qué se basa. La primera capa de la red CNN es una convolucional con kernels 3X3X7 con activación ReLU, seguida de una capa max pooling. A continuación, colocan otra capa convolucional con kernels 3X5X5 de nuevo con activación ReLU y capa max pooling. Tras ellas una nueva capa convolucional con kernels 3X5X5 con activación ReLu, una capa max pooling y finalmente la capa totalmente conectada (FC) con activación ReLu.

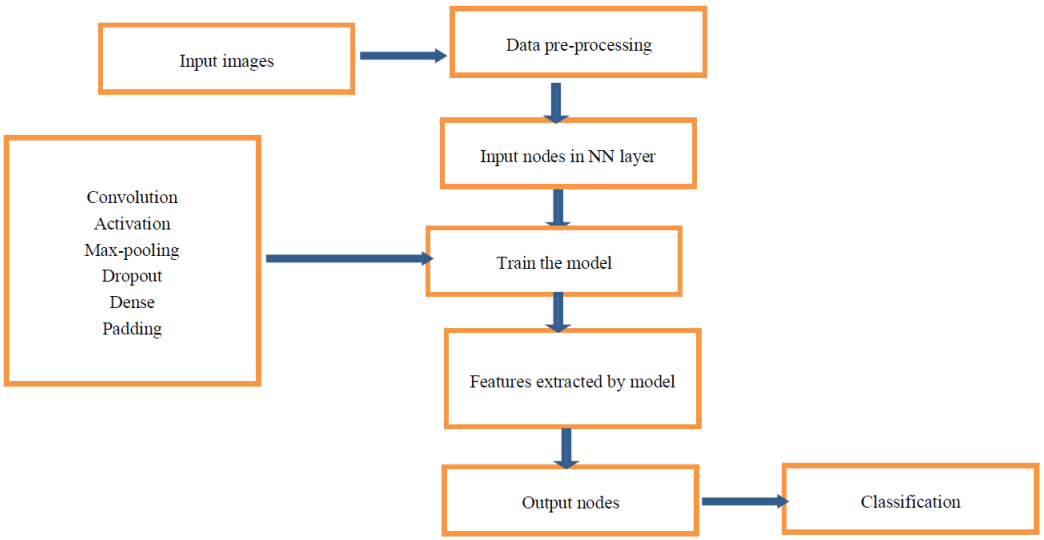

Figura 22: Arquitectura del sistema de clasificación de plátano. Sri et al. en [9]

El sistema CNN consigue un 92 % de aciertos en entrenamiento y un 80 % en test, siendo este el mejor rendimiento obtenido en comparación con el arrojado por otras técnicas no comentadas anteriormente en el paper. El rendimiento baja considerablemente de entrenamiento a test, lo que puede indicar un sobreentrenamiento de la red. El sobreentrenamiento es un efecto indeseado que se produce cuando el modelo se adapta en exceso a los ejemplos de entrenamiento, siendo poco flexible a datos nuevos reservados para validación o test. Dichos inconvenientes pueden paliarse con regularización (técnica que limita el

crecimiento del modelo) o reduciendo las iteraciones de entrenamiento, aunque en algunos casos extremos también puede indicar escasez de datos o diferencias de naturaleza de los datos de las particiones manejadas.

Los resultados de la red CNN son comparados con pruebas con algunos modelos preentrenados, asociados a técnicas de transfer learning como es el caso de Resnet, Densenet y Squeezenet. Los resultados de test de estas alternativas son peores que los de la arquitectura CNN construida de cero, 60 %, 55 % y 62 % de acierto respectivamente. Es algo posible ya que los modelos preentrenados no tienen por qué ser los más adecuados para nuestro problema concreto. Si mejoran el rendimiento de una red nueva será provechoso basarse en ellos, pero no tiene por qué ser así. En todo caso, en el paper no se detalla si esos modelos se han re-entrenado aislados o si se han añadido nuevas capas entorno a él, lo cual puede ayudar a sacarles rendimiento.

En las conclusiones resumen el rendimiento del modelo en base a la tasa de acierto de entrenamiento, aunque deberían ser las métricas de la fase de test las que den imagen del potencial de la solución. El 80 % de acierto en test no es despreciable e indica que las redes convolucionales se adecuan al problema de clasificación de plátano por maduración.

El control de calidad de papaya mediante percepción humana consume tiempo y es destructivo. Behera et al. buscan en [10] un sistema automático y no destructivo para realizar dichas labores de forma objetiva y sin error humano.

El estudio abre dos vías para resolver el problema, siendo éstas machine learning clásico por un lado y redes neuronales convolucionales (CNN) con técnicas de transfer learning por el otro. Dentro de ambas vías consideran distintos modelos y comparan su rendimiento. Utilizan 300 imágenes distribuidas uniformemente en 3 categorías de maduración, las cuales pueden apreciarse en la figura 23, tomando el 70 %, 20 % y 10 % para entrenamiento, validación y test respectivamente. El grado de maduración "a" se refiere a un estado muy poco maduro, el grado "b" a una maduración media y el grado "c" a una maduración intensa.

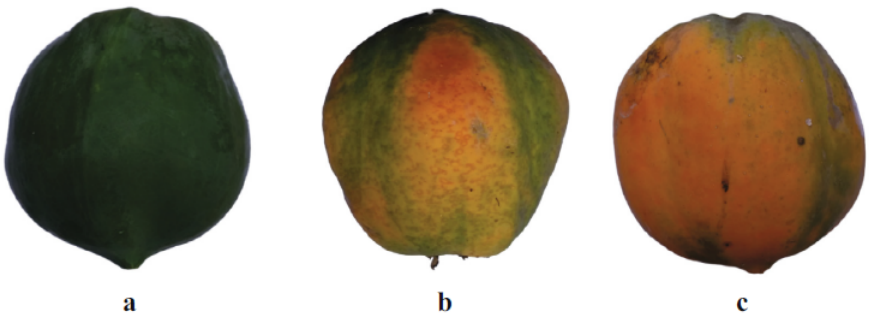

Figura 23: Grados de maduración de Papaya. Behera et al. en [10].

Los métodos de machine learning utilizados recurren a una extracción previa de características para luego clasificar las muestras en base a ellas. Por contra, los métodos de transfer learning se basan en CNN y trabajan con las imágenes en bruto.

Como se muestra en la figura 24, distintos métodos de ML basados en distintas características alcanzan el 100 % de acierto, con unos tiempos de entrenamiento por debajo de 1 segundo en muchos casos. Para comprender mejor el estudio, se procede a introducir los conceptos indicados para extracción de características y análisis de datos.

LBP: Local Binary Pattern es un sencillo y eficaz método de extracción de características, que opera en escala de grises sobre un píxel central y una vecindad 3X3. Compara el nivel de gris de cada píxel vecino con el del central y les va asignando un 1, en caso de que el nivel del gris del vecino supere el del central, o un cero en caso opuesto. De esta manera, retorna un patrón binario para cada vecindad (de ahí su nombre). Destaca por su ligereza computacional y su robustez frente a gradientes de iluminación.

HOG: Histogram of oriented gradients es otro método de extracción de características que al igual que LBP, trabaja en vecindades de tamaño de pixels variable y generalmente pequeño (del orden de las vecindades a 8 de LBP). Para cada vecindad computa un histograma de gradientes orientados, el cual normaliza posteriormente, para retornar un descriptor de cada vecindad.

GLCM: Gray Level Co-occurrence Matrix es un método estadístico de extracción de características. Opera sobre escala de grises y calcula la probabilidad con la que un píxel con nivel de gris ''A", es vecino inmediato (vecindad 3X3) de otro píxel con nivel de gris ''B". Con esa información genera un mapa de descriptores denominada matriz de co-ocurrencia.

KNN: K-Nearest Neighbors es un método de aprendizaje automático supervisado muy utilizado para clasificación. EL algoritmo genera un mapa de ejemplos con los datos de entrenamiento, en el que ubica el nuevo ejemplo a clasificar. La clase predicha se decide en base a qué clase pertenecen los ''k" vecinos más cercanos en dicho mapa. El valor de ''k" ha de optimizarse según los datos que se utilicen y puede ser crítico para el rendimiento del sistema. Se estudian varias variantes según la magnitud de ''k" (''fine", ''medium" y ''coarse") así como la versión ''weighted" que ponderan la distancia a los vecinos, aumentando la importancia de aquellos más cercanos al ejemplo a clasificar.

SVM: Support Vector Machines (SVM) es otra opción para el aprendizaje automático supervisado dirigido a regresión y clasificación. Sigue una filosofía similar a KNN ya que también representa el conjunto de entrenamiento en un plano y busca un hiperplano que los separe en dos clases, optimizando el mismo de forma que se maximice la distancia entre el hiperplano y los puntos más cercanos a él de ambas clases. Luego al evaluar un nuevo ejemplo, se determina a que lado del hiperplano ha de posicionarse. Para clasificaciones no binarias se eleva la dimensionalidad del espacio de entradas, se focaliza en el caso binario para facilitar la comprensión del algoritmo. Según la naturaleza matemática de la definición del hiperplano de separación, surgen variantes como lineal, cuadrática, gaussiana o cúbica.

Naïve Bayes: Se trata de un clasificador probabilístico que analiza un vector de características (observaciones) de los datos de entrenamiento, con objeto de determinar la probabilidad de un nuevo ejemplo de pertenecer a cada clase dada su naturaleza en las distintas características a estudio. El algoritmo trata de minimizar el coste de clasificar mal un ejemplo, buscando la regla óptima que minimice la probabilidad de error.

Feature	Classifier	Accuracy in %	AUC	Training time (in Second)
	Fine-KNN	100	1	2.5307
	Medium KNN	93.3	1	0.2722
	Coarse KNN	73.3	0.85	0.4482
	Weighted KNN	**100**	**1**	**0.1099**
LBP	Linear SVM	89.7	1	3.3836
	Quadratic SVM	100	1	0.5763
	Fine Gaussian SVM	100	1	0.2593
	Cubic SVM	100	1	0.2884
	Kernel Naïve Bayes	78	0.95	2.2277
	Fine-KNN	100	1	0.6123
	Medium KNN	100	1	0.1014
	Coarse KNN	62.7	0.83	0.1085
HOG	**Weighted KNN**	**100**	**1**	**0.0995**
	Linear SVM	97.3	1	0.7714
	Quadratic SVM	100	1	0.1903
	Fine Gaussian SVM	100	1	0.1614
	Cubic SVM	100	1	0.5644
	Kernel Naïve Bayes	97.3	1	1.3628
	Fine-KNN	100	1	0.5035
	Medium KNN	97.3	1	0.0807
	Coarse KNN	75	1	0.0864
	Weighted KNN	**100**	**1**	**0.0856**
GLCM	Linear SVM	97.3	1	0.5223
	Quadratic SVM	100	1	0.1803
	Fine Gaussian SVM	100	1	0.1575
	Cubic SVM	98.7	1	0.1738
	Kernel Naïve Bayes	96	1	0.6519

Figura 24: Resultados de machine learning. Behera et al. en [10].

Como se puede observar en la figura 25, dentro de las técnicas de transfer learning (ya descritas previamente en este apartado), el modelo VGG-19 destaca notablemente sobre los demás en tiempo de entrenamiento y en el número de iteraciones necesarias para optimizar el rendimiento en validación.

Dichas ventajas son de gran relevancia en el ámbito industrial, ya que una vez el sistema esté en marcha favorecen re-entrenamientos más ligeros, facilitando las intervenciones y dando lugar a paradas de producción más breves y ágiles.

Además, al no precisar extracción de características sobre las imágenes en el caso de los modelos CNN pre-entrenados, se aligera el futuro clasificador pudiendo ejecutarlo en dispositivos de características modestas (empotrados, PLCs), muy comunes en una planta industrial.

Pre-Trained model	Accuracy in %	No. of iteration to reach maximum validation accuracy	Training time (minute: Second)
ResNet101	95	15	2:33
ResNet50	100	20	2:42
ResNet18	100	15	2:09
VGG19	**100**	**10**	**1:52**
VGG16	100	20	2:28
GoogleNet	100	15	2:08
AlexNet	96.67	15	2:40

Figura 25: Resultados de transfer learning. Behera et al. en [10].

En general se obtienen muy buenos resultados por múltiples vías, se trata de un problema de resolución posible y puede abordarse por varios frentes, pudiendo elegir el más adecuado para cada situación. Aunque el dataset no es grande (300 ejemplos), solo se consideran 3 clases cuyas diferencias en cuanto a coloración de los frutos capturados son muy marcadas, lo que explica unos resultados tan buenos en un amplio abanico de técnicas.

El estudio de Mettleq et al. en [11] propone una clasificación de variedades de mango basada en redes neuronales convolucionales. En la figura 26 se pueden apreciar las diferencias visuales entre variedades de mango, con unos cambios de coloración muy similares a los observados en clasificaciones por maduración de otros frutos. En dicha imagen se presentan 5 variedades aunque son muchas más las existentes.

Sin embargo, el dataset utilizado para desarrollar el sistema propuesto solo cuenta con datos de 2 de esas variedades. El dataset cuenta con 916 imágenes, de las que los autores reservan un 10 % para validación y un 34 % para test, todas ellas de 150x150 píxels de tamaño.

Figura 26: Diferencias visuales entre variedades de mango. De La Cruz en [12].

En la figura 27 se especifica la arquitectura de capas utilizada para la red neuronal. Dado que se trata de una clasificación binaria impuesta por el dataset, la última capa tiene tamaño 1 suficiente para indicar a cual de las dos clases pertenece cada muestra.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d_1 (MaxPooling2	(None, 74, 74, 32)	0
conv2d_2 (Conv2D)	(None, 72, 72, 64)	18496
max_pooling2d_2 (MaxPooling2	(None, 36, 36, 64)	0
conv2d_3 (Conv2D)	(None, 34, 34, 128)	73856
max_pooling2d_3 (MaxPooling2	(None, 17, 17, 128)	0
conv2d_4 (Conv2D)	(None, 15, 15, 128)	147584
max_pooling2d_4 (MaxPooling2	(None, 7, 7, 128)	0
flatten_1 (Flatten)	(None, 6272)	0
dense_1 (Dense)	(None, 512)	3211776
dense_2 (Dense)	(None, 1)	513

Figura 27: Arquitectura de la red neuronal convolucional. Mettleq et al. en [11]

En la figura 28 se presentan los resultados, los cuales arrojan una clasificación perfecta en validación con unas pérdidas nulas. El dataset es amplio en número de elementos y el número de clases se reduce a solo 2, por lo que el modelo aprende a clasificar holgadamente dadas las gráficas que aporta el autor. Todo apunta a que el sistema sería capaz de diferenciar un número de clases mayor, acordemente con la diversidad de variedades que presenta el fruto. El problema propuesto resulta fácil de resolver para la red neuronal, se trata de un estudio de interés por lo similar de las características de coloración observadas y el modelo propuesto cumple bien con su cometido.

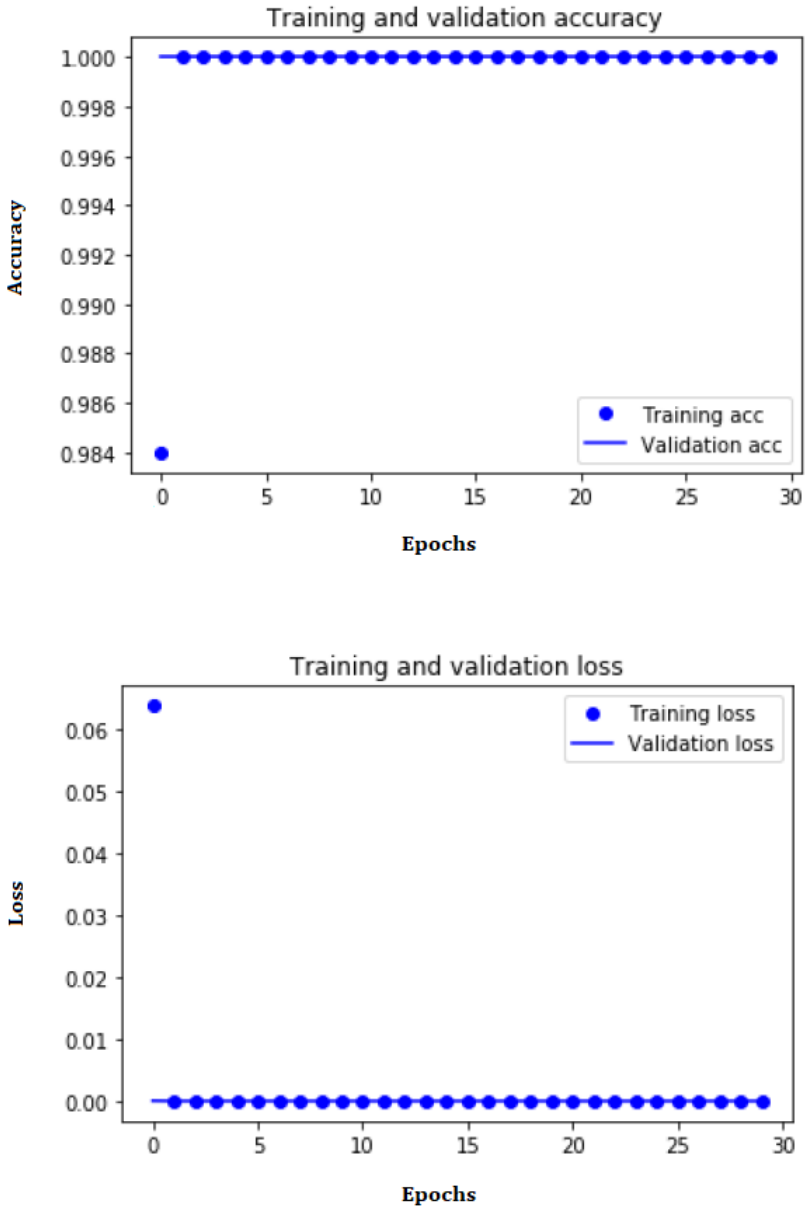

Figura 28: Resultados de rendimiento y pérdidas del sistema. Mettleq et al. en [11].

4.2. Método

4.2.1. Modelo

Vistos los trabajos previos y tratándose de técnicas presentes en ellos, con buenos resultados y buena adecuación para el problema a resolver, se propone aplicar al control de calidad de maduración de frutos una solución basada en redes neuronales convolucionales, asistida por técnicas de transfer learning.

La red VGG-19 entrenada con el banco de datos de Imagenet, presente en estudios previos y anteriormente descrita, ha sido elegida por su buen comportamiento en problemas similares para integrarse en la arquitectura del sistema mediante técnicas de transfer learning. En la figura 29 se detalla la arquitectura del modelo base (VGG-19 sin su clasificador final).

Model: "vgg19"

Layer (type)	Output Shape	Param #
input_5 (InputLayer)	(None, 100, 100, 3)	0
block1_conv1 (Conv2D)	(None, 100, 100, 64)	1792
block1_conv2 (Conv2D)	(None, 100, 100, 64)	36928
block1_pool (MaxPooling2D)	(None, 50, 50, 64)	0
block2_conv1 (Conv2D)	(None, 50, 50, 128)	73856
block2_conv2 (Conv2D)	(None, 50, 50, 128)	147584
block2_pool (MaxPooling2D)	(None, 25, 25, 128)	0
block3_conv1 (Conv2D)	(None, 25, 25, 256)	295168
block3_conv2 (Conv2D)	(None, 25, 25, 256)	590080
block3_conv3 (Conv2D)	(None, 25, 25, 256)	590080
block3_conv4 (Conv2D)	(None, 25, 25, 256)	590080
block3_pool (MaxPooling2D)	(None, 12, 12, 256)	0
block4_conv1 (Conv2D)	(None, 12, 12, 512)	1180160
block4_conv2 (Conv2D)	(None, 12, 12, 512)	2359808
block4_conv3 (Conv2D)	(None, 12, 12, 512)	2359808
block4_conv4 (Conv2D)	(None, 12, 12, 512)	2359808
block4_pool (MaxPooling2D)	(None, 6, 6, 512)	0
block5_conv1 (Conv2D)	(None, 6, 6, 512)	2359808
block5_conv2 (Conv2D)	(None, 6, 6, 512)	2359808
block5_conv3 (Conv2D)	(None, 6, 6, 512)	2359808
block5_conv4 (Conv2D)	(None, 6, 6, 512)	2359808
block5_pool (MaxPooling2D)	(None, 3, 3, 512)	0

Total params: 20,024,384
Trainable params: 0
Non-trainable params: 20,024,384

Figura 29: Arquitectura del modelo base VGG-19.

Como se comentó anteriormente, es posible añadir nuevas capas tras el modelo base de transfer learning. Dado que respecto a las capas del modelo base solo es posible decidir si se re-entrenan con los nuevos datos o no, estas capas añadidas posteriormente dan juego al desarrollador para poder cambiar, corregir o mejorar aspectos del sistema que tengan que ver con la arquitectura del modelo.

De esta manera, como se puede apreciar en la figura 30, tras el modelo base se añade una capa de aplanamiento que convierte la salida en 3 dimensiones 3x3x512 en una salida 1-D de tamaño 4608. A continuación, se acoplan varias capas densas (3 concretamente) que van reduciendo el tamaño de las salidas hasta tamaño 256. Tras ellas se encuentra una capa de regularización encargada de prevenir el sobreentrenamiento, para luego dar lugar a dos capas densas más, siendo la última de ellas la encargada del clasificador con activación softmax.

En la última capa de la red se ha de lograr un tamaño adecuado dado el número de clases, siendo necesario en este caso una salida de tamaño 9 dado que el dataset con el que trabaja está dividido en 9 categorías de ejemplos y por tanto, será dentro de una de las 9 categorías donde se pretenden clasificar los nuevos ejemplos. La red convolucional emite su juicio sobre qué clase corresponde a un nuevo ejemplo señalando una de esas 9 salidas de su última capa.

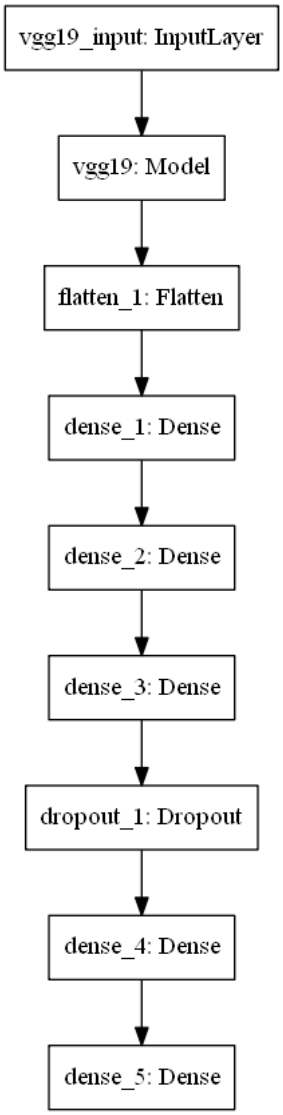

Figura 30: Arquitectura del modelo final.

Las capas del modelo base se congelan para evitar su re-entrenamiento, sus parámetros se marcan como no entrenables y todo el trabajo se centrará en las capas finales añadidas, reduciendo notablemente el tiempo de entrenamiento. Además, se utiliza un dataset aumentado basado en pequeñas rotaciones, volteos horizontales y variación de zoom, recurso que se aborda con más detalle en el siguiente epígrafe.

4.2.2. Aumento de datos

Un gran problema que siempre se encuentra al aplicar inteligencia artificial en el ámbito industrial es la escasez de datos utilizables, razón por la que el transfer learning resulta de interés. Si se alimenta el entrenamiento de un modelo con datos repetidos no se obtiene ninguna mejora, serán obviados, sin embargo existen técnicas de aumento de datos para generar datos aprovechables modificando los ya existentes. En el caso de datasets de imágenes, se realizan ligeras modificaciones como rotaciones, volteos, cambios de zoom, iluminación, etc. La imagen original y esa misma procesada con dichas operaciones, pasan a ser 2 imágenes distintas a ojos de la red aprovechando ambas para entrenar.

En el presente desarrollo, se aumenta el conjunto de datos de entrenamiento original compuesto por 5103 imágenes aplicando un volteo horizontal, una rotación de 2 grados en sentido aleatorio y una variación aleatoria del zoom en un rango de (0.9 , 1.1), siendo el valor 1 el zoom por defecto de la imagen original. De esta manera, por cada de imágenes originales se genera un nuevo lote de imágenes con las transformaciones aleatorias descritas, consiguiendo doblar el dataset de entrenamiento. Remarcar que dicho aumento solo se aplica a la partición de entrenamiento, ya que para unas métricas de rendimiento fieles, no conviene alterar la naturaleza de los datos de test por leve que sea la transformación.

4.3. Experimentos

Para desarrollar la solución propuesta se ha preparado un entorno Windows 10 profesional. Se ha contado con la distribución de Python "Anaconda", usando de forma intensiva sus paquetes Jupyter Notebook y Spyder. Sobre dichas herramientas se ha procedido a desarrollar código Python orientado a correr en entorno Windows sin contar con CUDA para ejecución en GPUs.

En cuanto a bibliotecas externas, el desarrollo se basa en la utilización de Keras y Tensorflow. De ellas se obtiene el modelo de IA importando la red preentrenada VGG-19, los modelos de capas añadidas, los manejadores de aumento e iteración de datos en formato imagen y otros objetos relativos a métricas y optimización de error. Para apoyar el entrenamiento del modelo y la gestión del dataset, se recurre a algunas funcionalidades de Sklearn y Open-CV, habiendo sido esta segunda la encargada de la totalidad del desarrollo de corrección de perspectiva en el bloque anterior.

Adicionalmente, se recurre a otras bibliotecas auxiliares para representación y resumen de datos, así como bibliotecas básicas comúnmente utilizadas en cualquier desarrollo Python: Matplotlib, Seaborn, Pandas, Math y Numpy.

Respecto al dataset, se ha optado por un conjunto de imágenes de tomate publicado por Oltean et al. en [13]. Se trata de un dataset muy completo de piezas de fruta sobre fondo neutro (blanco), fotografiadas durante rotaciones horizontales respecto a distintos perfiles. En la figura 31 puede apreciarse un ejemplo de cada una de las 9 clases utilizadas para entrenamiento y test. El dataset ofrece 6800 imágenes y además, se aumenta el conjunto de datos de entrenamiento (5103 imágenes) aplicando un volteo horizontal, una rotación de 2 grados en sentido aleatorio y una variación aleatoria del zoom en un rango de (0.9 , 1.1), siendo el valor 1 el zoom por defecto de la imagen original. Con las técnicas de aumento de datos se consigue un nuevo batch (lote) de imágenes de entrenamiento por cada batch original.

La relación entre las respectivas clases no se basa únicamente en el estado de maduración, también existen diferencias por variedad de tomate. Aunque la distinción de variedades no es objeto de este estudio, sirve para aumentar el número de clases complicando el problema. Además, en muchos casos las diferencias visuales según maduración y según variedad de raza de tomate son muy similares, por tanto no supone alejarse del objetivo.

Puede ilustrarse lo expuesto observando la figura 31, ya que el tomate amarillo es una mutación que afecta al color en tomates como el que tiene justo a su derecha. No se aprecia un cambio de forma relevante pero si un cambio de coloración, tal diferencia es la que habitualmente se observa en distintos estados de maduración.

Figura 31: Muestras de las 9 clases del dataset.

Se entrena con 5103 imágenes distribuidas en lotes de 32 y a lo largo de 100 epochs. Como se ha indicado anteriormente, las capas del modelo base se congelan y solo se entrenan las nuevas capas añadidas detrás, lo que favorece un entrenamiento mucho más ligero al pasar de 25.434.185 parámetros entrenables a 5.409.801.

Finalizado el entrenamiento del modelo, se serializa el objeto que lo describe en un fichero para futuras utilizaciones y se procede a realizar la fase de test. La parte del dataset que no se utilizó para entrenar, alimenta el modelo para que éste realice sus predicciones de test, en este caso 1697 imágenes. Cabe destacar que las imágenes de test no han sido objeto de técnicas de aumento de datos, ya que para evaluar el rendimiento del sistema se buscan ejemplos de idéntica naturaleza a los que el sistema tendrá que clasificar en producción, sin ninguna alteración sintética por leve que sea. Una vez realizadas las predicciones, se comparan éstas con las clases reales de cada imagen evaluada y se genera una matriz de confusión. Este proceso es el que permite conocer el rendimiento final del sistema propuesto. El modelo entrenado y serializado será validado si se acepta su rendimiento en la fase de test.

De esas pruebas en fase de test han de derivarse métricas que aporten luz sobre el rendimiento real del sistema. Para ello, es preciso asentar algunos conceptos que servirán para entender lo expuesto en el apartado de resultados.

La forma típica de representar los resultados de test es la matriz de confusión (figura 32). Dicha matriz contiene en sus cabeceras el compendio total de clases del problema, refiriéndose en un eje a las clases reales y en otro a las predichas por el modelo. Los ejemplos clasificados con éxito, es decir, aquellos en lo que coinciden la clase real y la

predicha, se quedan en la diagonal principal de la matriz facilitando enormemente la fácil comprensión de los datos. Los dígitos restantes se refieren a ejemplos mal clasificados. Según la ubicación de dichos "errores", se manejan algunas nomenclaturas presentes en la figura 32 y detalladas a continuación.

Figura 32: Ejemplos de matriz de confusión. Martinez Heras en [14].

TP: Positivos verdaderos. Ejemplos clasificados en clase 'X' siendo su verdadera clase la clase 'X'.

FP: Falsos positivos. Ejemplos clasificados en clase 'X' siendo otra su verdadera clase.

TN: Negativos verdaderos. Ejemplos no clasificados en clase 'X' siendo otra su verdadera clase.

FN: Negativos falsos. Ejemplos no clasificados en clase 'X' siendo su verdadera clase la clase 'X'.

Acompañando a la matriz de confusión suelen figurar métricas numéricas más compactas del rendimiento. La más conocida es la tasa de acierto "accuracy", que indica el porcentaje de ejemplos bien clasificados respecto al total de ejemplos ds test.

Sin embargo, se trata de una métrica que puede llevar a engaño en caso de contar con cantidades de ejemplos desbalanceadas según su clase real, es decir, si hay muchos más ejemplos de unas clases que de otras, lo bien o mal que funcione el modelo con las más pobladas afectará severamente al "accuracy", pudiendo dar una imagen equivocada del rendimiento general. Por ello, se cuenta con otras medidas de cara a describir de forma más completa los ensayos, algunas de las cuales se indican a continuación.

Precision: Métrica que da fe de la calidad del clasificado. Indica el porcentaje de ejemplos clasificados en una determinada clase (predicción) que han sido correctamente clasificados (realidad). En la figura 33 se presenta su fórmula, así como su significado sobre la matriz de confusión.

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

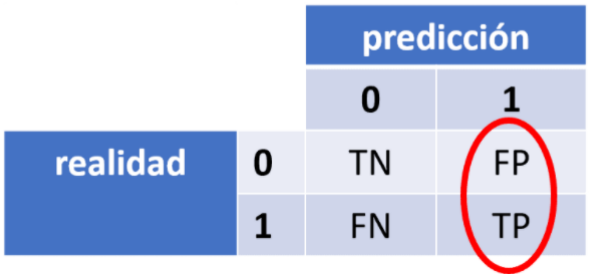

Figura 33: Precision. Martinez Heras en [14].

Recall: Métrica de exhaustividad (figura 34). Indica el porcentaje de ejemplos de una determinada clase (realidad) que el modelo es capaz de clasificar correctamente (predicción).

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

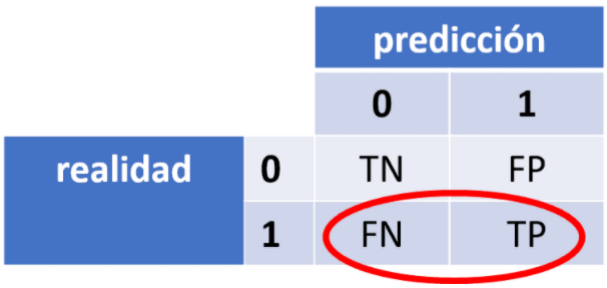

Figura 34: Recall. Martinez Heras en [14].

F1 Score: Valor que combina las métricas de precision y recall realizando su media armónica (figura 35).

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Figura 35: F1 Score. Martinez Heras en [14].

4.4. Resultados

Una vez finalizado el entrenamiento se recogen los datos que ilustran cómo ha ido el mismo, dicha información es de un alto valor por su potencial para detectar fallos en la fase de entrenamiento, de cara a corregirlos y re-entrenar el modelo. Luego se utilizan los datos reservados para test y se realiza una prueba de rendimiento final que da cuenta de la capacidad del sistema para funcionar en producción.

En las gráficas de la figura 36 se puede apreciar la evolución de las pérdidas y el porcentaje de acierto a lo largo del entrenamiento, entendiendo acierto como clasificar una imagen de una determinada clase en esa misma y no en otra.

Como se puede apreciar claramente en la representación de las pérdidas, el número de epochs es suficiente para apreciar cierta periodicidad, lo que lleva a pensar que un número mayor de epochs no arrojaría resultados radicalmente distintos a los ya recogidos.

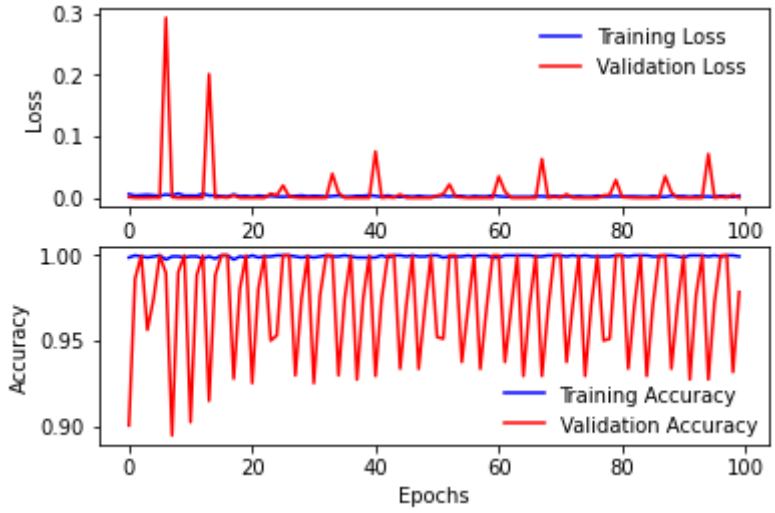

Figura 36: Representación de resultados de entrenamiento.

Una vez realizada la fase de test con aquella parte del dataset reservada para tal fin, se obtienen los resultados representados en la figura 37. La matriz de confusión detalla qué clase se predice para todos los ejemplos de cada clase utilizados en el test. Los valores de la diagonal principal corresponden a ejemplos con clases verdaderas y predichas coincidentes, quedando fuera de ella los errores, o sea, ejemplos cuya clase predicha no es igual que su clase verdadera. El mapa de calor en tonos azules junto con su leyenda, asisten al lector en la tarea de asimilar rápidamente los resultados expuestos. Además, se añade en el encabezado de la misma una medida en tanto por ciento del acierto total obtenido en test, la cual resume de forma compacta el rendimiento del sistema.

Se consigue un 97 % de "accuracy" en la fase de test. Dado que esta métrica típicamente utilizada puede ser engañosa, en la figura 38 se presentan otras métricas ya descritas que aseguran el buen rendimiento en test de la solución planteada, coincidiendo con unos altos valores de rendimiento por encima del 97 % en todos los casos.

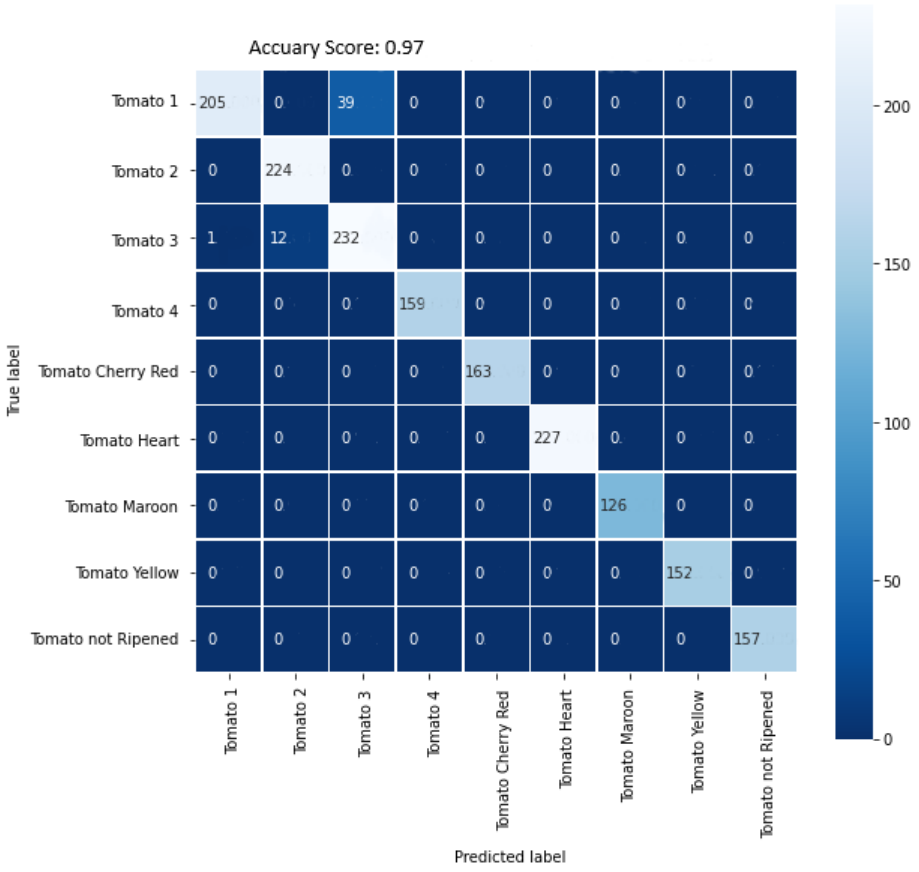

Figura 37: Matriz de confusión de test.

Clases	TP	FP	FN	Precision	Recall	F1 Score
Tomato 1	205	1	39	0,995	0,840	0,911
Tomato 2	224	12	0	0,949	1,000	0,974
Tomato 3	232	39	13	0,856	0,947	0,899
Tomato 4	159	0	0	1,000	1,000	1,000
Tomato Cherry Red	163	0	0	1,000	1,000	1,000
Tomato Heart	227	0	0	1,000	1,000	1,000
Tomato Maroon	126	0	0	1,000	1,000	1,000
Tomato Yellow	152	0	0	1,000	1,000	1,000
Tomato not ripened	157	0	0	1,000	1,000	1,000
Promedio	-	-	-	0,978	0,976	0,976

Figura 38: Métricas de rendimiento de test.

5. Discusión

5.1. Hardware y sistema operativo

Emplear un entorno Windows sin GPU no es lo óptimo en el campo de inteligencia artificial. Los entornos Linux ofrecen muchas ventajas tanto en desarrollo como en despliegue, ya que las actualizaciones periódicas son mucho menos peligrosas y críticas y en general, se trata de operativos más fiables y menos propensos a provocar paradas de producción. Muchos de ellos son de código abierto, por lo que el desarrollador tiene mucho más margen de maniobra a la hora de ajustar el operativo a sus necesidades.

Además, las GPU externas son de gran provecho pudiendo utilizarlas gracias a la biblioteca "Cuda", para conseguir un rendimiento muy superior al de un microprocesador en entrenamientos con imágenes, ya que estos procesadores gráficos están diseñados para trabajar con datos matriciales de imagen y vídeo. Sin embargo, el futuro uso de este estudio de viabilidad en un proyecto real de cliente donde se imponen estas restricciones de hardware y software obligan a tomar decisiones no óptimas, ya que si se va aumentando la distancia con el caso real, el estudio de viabilidad ve mermado su interés.

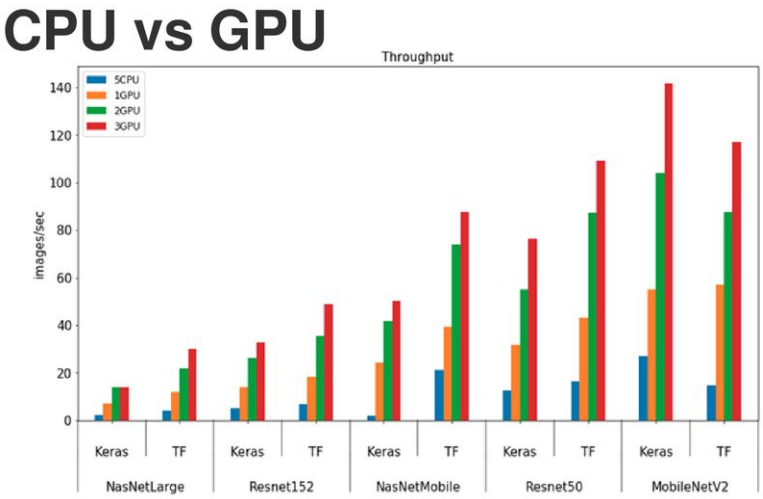

Figura 39: Desempeño de CPUs y GPUs en transfer learning. Salvaris et al. en [15]

5.2. Datasets

La utilización de un dataset distinto al del proyecto real, se decidió conforme al hecho de que aunque ya se tenían datos reales, eran escasos y no estaban aún etiquetados. El futuro cambio de dataset al conjunto de imágenes de aceituna en almazara, se establece como ampliación posterior al proyecto actual.

Las clases de dicho dataset de pruebas exceden el ámbito de la madurez, ya que aunque contienen clases de tomates que se diferencian entre sí por estar más o menos maduras, también se extiende el estudio a otras variaciones de color y forma relativas a otras variedades y mutaciones de tomates. Dicha extensión es intencionada, ya que el código

demostrará su robustez enfrentándose a un problema más complejo aumentando el número de clases, manteniendo una misma línea de características a detectar.

En el proyecto real solo se consideran 3 grados de maduración de aceituna, pero ha de entenderse que se trata de imágenes mucho menos idílicas. Mientras que el dataset de tomates se basa en imágenes de un fruto sobre fondo plano, en las imágenes de aceituna hay fondos irregulares y parcialmente cambiantes, muchos frutos a la vez, suciedad del fruto y también de la cámara en ciertas ocasiones. Por todo ello, cuanto más robustez se pueda obtener en el estudio de viabilidad complicando el problema, más seguridad se tendrá de que este problema es resoluble mediante la arquitectura propuesta.

Mediante las técnicas de aumento de datos se logra un nuevo lote de imágenes por cada lote original, doblando el dataset de entrenamiento y entrenando con el conjunto resultante. Si se entrena con dos imágenes idénticas el resultado no varía, el modelo desprecia el duplicado. Sin embargo, con leves cambios como una pequeña rotación, el modelo tiene en cuenta ambas versiones de una misma imagen para aprender, lo que es muy provechoso siempre y cuando se tenga cuidado de elegir bien las transformaciones a aplicar, ya que no deben modificar lo que realmente interesa aprender. En este caso, ninguna de las transformaciones aplicadas afecta a la coloración de los frutos, característica principal a extraer y aprender.

5.3. Pre-procesado

La transformación homográfica se eligió por ser una opción sencilla, computacionalmente ligera y que ofrece grandes ventajas al no necesitar calibrar la cámara para su cómputo. La camara de la almazara se ha manejado de forma remota vía web y no se contó con personal que pudiese asistir colocando imágenes de calibración por el layout, la necesidad de calibración hubiese requerido un desplazamiento a Córdoba durante las prácticas, con todas las complicaciones que ello conlleva en la actualidad.

Existe una fuente de error por modelar la cinta cargada como un plano ya que valga la redundancia, no es plana. Los frutos están sobre ella, sobresalen del plano y ello provocará deformaciones al proyectar al plano imagen, no obstante sigue siendo un método válido por el hecho de no necesitar medir ni detectar formas, al centrarse el clasificador en datos de coloración.

La homografía ha conseguido con éxito los dos propositos esperados. Por un lado, ha conseguido cambiar la vista del producto a una muy cercana a la cenital, eliminando la posible y engorrosa necesidad de modificar/trasladar el bastidor de la cámara para cambiar su enfoque. Por otro lado, ha eliminado prácticamente todo objeto, fondo y zona de iluminación irregular presente en la vista, convirtiendo un dataset muy alejado de lo idílico para el machine learning, en uno mucho más centrado y adecuado en el propósito del estudio.

5.4. Modelo

El estado del arte confirma que las redes neuronales convolucionales son una técnica muy adecuadas para clasificación de imágenes en problemas similares, ya sea con modelos construidos de cero o con transfer learning. El frecuente recurso en estudios previos a

esta tecnología, así como los buenos resultados vistos avalan su elección para el presente proyecto, decisión respaldada finalmente por los buenos resultados obtenidos.

Técnicas basadas en modelos de machine learning, los cuales aprenden en base a características extraídas con procesamiento de imagen también han demostrado ser de utilidad en el estado del arte. Sin embargo, dichas vías complican el código del decisor final, cargándolo con operaciones sobre las imágenes. Dado que se pretende correr el decisor en un empotrado en su despliegue en el proyecto real, son vías menos interesante dadas las limitaciones de recursos de estos dispositivos y también atendiendo a las repercusiones en cuanto a tiempos de ejecución. Además, las redes convolucionales son más novedosas, no han sido estudiadas en el máster y por todo ello tienen especial atractivo investigador en el entorno académico y empresarial.

Las técnicas de transfer learning no dejan de ser un potenciador de rendimiento, ya que se basa en aprovechar trabajo previo para aprovechar su entrenamiento. Dependiendo del problema a resolver, distintos modelos preentrenados pueden funcionar mejor o peor. VGG-19 fue un modelo presente en un estudio previo de los analizados, el que mejor funcionó en tasa de acierto y tiempo de entrenamiento entre los de su clase. VGG-16 fue la primera alternativa a probar en el problema actual pero no se consiguió superar el 70 % de acierto en test. Sin embargo, luego se usó VGG-19 y funcionó de forma impecable, algo que no tiene por qué estar asegurado al recurrir a estos modelos.

5.5. Entrenamiento y test

El entrenamiento, para el que se proporcionan al sistema 5103 imágenes uniformemente distribuidas en las 9 clases, se ha realizado sin re-entrenar el modelo base. Si a futuro hiciese falta un entrenamiento más exhaustivo, aun queda reservada la opción de descongelar todos esos coeficientes y realizar un ajuste fino de los mismos con los datos del problema a resolver. También queda disponible aplicar más aumento de datos, o unos pre-procesados mas agresivos en esos nuevos datos generados sintéticamente. Una vez más, se trata de dejar vías abiertas de cara a enfrentar el sistema a un problema más complejo tras el estudio.

En la fase de test se realizan una serie de predicciones con un conjunto de datos distinto al de entrenamiento (1697 imágenes), aunque de la misma naturaleza. En porcentaje de acierto logrado asciende a un 96.93 % de acierto. Se trata de un rendimiento a considerar, aunque para poder realizar aseveraciones sobre el mismo hay que tener en cuenta los requisitos del proyecto y la valoración del cliente. En el caso del proyecto real de aceituna sería un rendimiento extraordinario y más que suficiente, ya que en la actualidad la supervisión manual está cometiendo grandes errores de bulto de forma habitual. Las otras métricas de rendimiento confirman que la respuesta del sistema es buena, respaldando lo obtenido con la métrica de "accuracy" que como se comentó, en ocasiones puede ser engañosa.

Además, mirando con detenimiento la matriz de confusión de test, se puede apreciar que la mayor parte de los errores vienen de confundir ejemplos de "Tomato 1" con "Tomato 3" y ejemplos de "Tomato 3" con "Tomato 2" (ver figura 40). Teniendo localizadas las clases donde se localiza la mayor parte del error, más aún cuando una misma clase está presente en ambos casos, es posible focalizar sobre ella valorando la necesidad de diferenciar dichos ejemplos de los de las demás clases, incrementando los datos de entrenamiento de esa

variedad, eliminando datos mal etiquetados o revisando el efecto de las modificaciones de aumento de datos realizadas.

Figura 40: Ejemplos de las variedades de tomate problemáticas en test

No obstante, se está abordando un error del 3 % y las operaciones correctivas han de ser proporcionales e inocuas para las clasificaciones que se están realizando correctamente. Como se ha expuesto anteriormente, lo bien o mal que rinde un sistema se ha de medir conforme a lo que se requiere de él, dado el ecosistema y tiempo de respuesta exigido.

6. Conclusiones

Tras el estudio expuesto, se afirma que es posible detectar grados de maduración en frutos mediante el sistema propuesto. Se consigue un rendimiento excelente dejando aún vías abiertas para poder incrementar el mismo si el problema lo requiere. Si con los datos reales se obtiene un rendimiento cercano al conseguido con los datos de tomates, la mejora del control de calidad respecto al método manual utilizado hasta ahora en la almazara será abismal.

La transformación homográfica ha demostrado comportarse muy bien en layouts similares al presentado, una gran ayuda teniendo en cuenta que en una planta industrial las cámaras no siempre pueden estar donde uno querría, ya sea por temas físicos o por que se esta aprovechando una cámara ya montada con otros fines (videovigilancia, etc.).

El modelo preentrenado VGG-19 ha demostrado comportarse de forma excelente para el problema propuesto sin necesidad de reentrenar sus coeficientes, hecho a tener en cuenta para otros proyectos de estas características ya que no todos los modelos de transfer learning sirven para cualquier caso de estudio.

Analizando la matriz de confusión de test, de las 9 clases consideradas los ejemplos de test se clasifican sin a penas error alguno en 7 clases. En las dos clases restantes se ha fallado un 15.98 % y un 5.3 % de las veces respectivamente, pudiendo focalizar futuras iteraciones sobre los conjuntos de datos implicados y viendo posibles soluciones. Además, dichos errores obtenidos en el estudio de viabilidad no son relevantes para el proyecto real pues se trata de clases con coloración muy similar y variaciones de forma, cuando en el problema real las clases de fruto tienen forma común y diferencias claras en coloración.

Las ampliaciones a futuro pasan por la aplicación del sistema expuesto al proyecto real de clasificación de aceituna. La adaptación se basa por un lado en la reducción de clases de 9 a 3, modificando la última capa de la arquitectura en consecuencia. Luego se deberá repetir el entrenamiento y ajustarlo según los resultados obtenidos.

Por otro lado, el modelo entrenado definitivo utilizando todos los datos reales disponibles ha de utilizarse en un script Python que actúe como decisor para nuevas imágenes, corriéndolo siempre que sea posible en un dispositivo PLCNext AXC F 3152 sobre contenedor Balena-Engine. El tiempo de ejecución de dicho script debe decidir en como máximo 2 segundos, requisito impuesto por el cliente y tener un nivel de acierto suficiente para no cometer errores de bulto, ya que actualmente con supervisión manual se están almacenando en el mismo silo muchos kilogramos de aceituna de niveles de maduración opuestos (verde y negra), problema prioritario para su resolución.

La ejecución del decisor ha de integrarse en un sistema automatizado, que reaccione a las capturas periódicas lanzadas por detección de movimiento de la cámara Kedacom ya desplegada. La clase asignada a cada imagen alimenta decisiones de automatización ligadas al encaminamiento del producto, permitiendo separar en silos distintos frutos de distinta clase.

Para labores de etiquetado se cuenta con un equipo de ingeniería agrónoma, además de contar con datos de los rendimientos grasos de las remesas capturadas por las cámaras según su marca temporal, datos estrechamente ligados a la maduración, que servirán para asegurar un minucioso etiquetado de el dataset de entrenamiento y test.

7. Presupuesto

En la figura 41 se desglosa el presupuesto del proyecto. Se tienen en cuenta los materiales empleados para el trabajo descrito, los cuales se reducen al equipo de desarrollo y la cámara. Más adelante, con el paso del estudio de viabilidad al proyecto real, tocará añadir el PC industrial a desplegar en planta, coste no tenido en cuenta ya que aún no se ha pasado a esa fase. Luego se consideran los costes de personal por las horas invertidas dada la planificación, los gastos generales y beneficio industrial y por último, la tasa impositiva a aplicar al total. Con todo ello, el presupuesto final del proyecto impuestos incluidos asciende a 24.673,41 euros.

Hardware

Concepto	Coste
PC de desarrollo intel i5 8Gb RAM	400 €
Cámara Kedacom IPC2251-HNB-SIR80-Z6048	663 €
Totales	**1.063 €**

Horas facturadas

Concepto	Horas	Coste / Hora	Coste
Especificaciones iniciales	100	20,00 €	2.000,00 €
Diseño, desarrollo y pruebas del Bloque 1	100	30,00 €	3.000,00 €
Diseño, desarrollo y pruebas del Bloque 2	225	30,00 €	6.750,00 €
Documentación final	175	20,00 €	3.500,00 €
Totales	**600**	**100,00 €**	**15.250,00 €**

Total ejecución material del proyecto

Concepto	Presupuesto
Total hardware	1.063,00 €
Total horas facturadas	15.250,00 €
Totales	**16.313,00 €**

Total ejecución por contrata

Concepto	Presupuesto
Total ejecución material	16.313,00 €
Gastos generales (15%)	2.446,95 €
Beneficio industrial (10%)	1.631,30 €
Totales	**20.391 €**

Total ejecución por contrata tras impuestos

Concepto	Presupuesto
Total antes de impuestos	20.391,25 €
I.V.A. del 21%	4.282,16 €
Totales	**24.673,41 €**

Figura 41: Desglose del presupuesto del proyecto.

8. Bibliografía

1. Cañas, L. (2021, 31 marzo). Recién amueblado y listo para entrar a vivir!. Familia Luis Cañas.
   Recuperado de:
   https://www.familialuiscanas.com/web/category/luis-canas/
   Última consulta: 13/01/2021

2. Durán Ferreras, A. (2012, 24 febrero). Capítulo 5 - Percepción. Modelado, Control y Percepción en Sistemas Aéreos Autónomos. Trabajo fin de Máster. Universidad de Sevilla. España.
   Recuperado de:
   http://bibing.us.es/proyectos/abreproy/70314/direccion/Memoria%252F
   Última consulta: 21/04/2021

3. Maulion, M. (2021, 2 febrero). Homography Transform — Image Processing - Matt Maulion. Medium.
   Recuperado de:
   https://mattmaulion.medium.com/homography-transform-image-processing-eddbcb8e4ff7
   Última consulta: 21/04/2021

4. Jeong, W., Fatica, M. (2011). Medical Image Processing Using GPU-Accelerated ITK Image Filters. Neighboring Pixel - an overview — ScienceDirect Topics. ScienceDirect.
   Recuperado de:
   https://www.sciencedirect.com/topics/computer-science/neighboring-pixel
   Última consulta: 20/01/2021

5. Wikipedia contributors. (2021, 22 abril). Residual neural network. Wikipedia.
   Recuperado de:
   https://en.wikipedia.org/wiki/Residual_neural_network
   Última consulta: 20/04/2021

6. Goswami, S. (2020, 26 octubre). How to use a pre-trained model (VGG) for image classification. Medium.
   Recuperado de:
   https://towardsdatascience.com/how-to-use-a-pre-trained-model-vgg-for-image-classification-8dd7c4a4a517
   Última consulta: 20/04/2021

7. Alake, R. (2020, 23 diciembre). Deep Learning: GoogLeNet Explained - Towards Data Science. Medium.
   Recuperado de:
   https://towardsdatascience.com/deep-learning-googlenet-explained-de8861c82765
   Última consulta: 20/04/2021

8. Kurama, V. (2021, 9 abril). A Guide to AlexNet, VGG16, and GoogleNet. Paperspace Blog.
   Recuperado de:
   https://blog.paperspace.com/popular-deep-learning-architectures-alexnet-

vgg-googlenet/
Última consulta: 20/04/2021

9. Sri, M.K., Saikrishna, K., Kumar, V.V. (2020, 23 marzo). Classification of Ripening of Banana Fruit Using Convolutional Neural Networks, Anurag Group of Institutions, India.
Recuperado de:
https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3558355
Última consulta: 13/01/2021

10. Behera, S.K., Rath, A.K., Sethy, P.K. (2020, 3 mayo). Maturity status classification of papaya fruits based on machine learning and transfer learning approach, Department of Computer Science and Engineering, Veer Surendra Sai University of Technology, Odisha, Department of Electronics, Sambalpur University, Odisha, India.
Recuperado de:
https://www.sciencedirect.com/science/article/pii/S2214317320300044
Última consulta: 13/01/2021

11. Mettleq, A.S.A, Dheir, I.M., Elsharif, A.A., Abu-Naser, S.S. (2019, 12 diciembre), Mango Classification Using Deep Learning, Department of Information Technology,Faculty of Engineering & Information Technology, Al-Azhar University, Gaza, Palestine.
Recuperado de:
https://core.ac.uk/download/pdf/286027617.pdf
Última consulta: 13/01/2021

12. De La Cruz, A. (2018, 15 febrero), Variedades del mango. Mango con chile.
Recuperado de:
https://sites.google.com/site/mangoconchileconmangoconchile/variedades-del-mango
Última consulta: 13/01/2021

13. Oltean, M., Muresan, H. (2018, 20 octubre). Fruit recognition from images using deep learning, Acta Univ. Sapientiae, Informatica Vol. 10, Issue 1, pp. 26-42, 2018. Fruits 360 dataset. Mendeley Data.
Recuperado de:
https://data.mendeley.com/datasets/rp73yg93n8/1
Última consulta: 17/01/2021

14. Martinez Heras, J. (2020, 9 octubre). Precision, Recall, F1, Accuracy en clasificación - IArtificial.net
Recuperado de:
https://www.iartificial.net/precision-recall-f1-accuracy-en-clasificacion/#Precision_Precision
Última consulta: 16/04/2021

15. Salvaris, M., UZ, F.B. Deploying Deep Learning Models on GPU Enabled Kubernetes Cluster. SlidePlayer.
Recuperado de:
https://slideplayer.com/slide/17239727/
Última consulta: 20/01/2021