UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DISEÑO, IMPLEMENTACIÓN Y EVALUACIÓN DE TÉCNICAS DE CLASIFICACIÓN DE NARANJAS NAVELES MEDIANTE TÉCNICAS DE VISIÓN POR COMPUTADOR

AUTOR: SERGIO MOYA MARÍN

TUTOR: ANTONIO JOSÉ SÁNCHEZ SALMERÓN

Seleccion

Curso Académico: 2020-21

AGRADECIMIENTOS

A mi familia por estar siempre.

RESUMEN

El objetivo de este trabajo consiste en el diseño, implementación y evaluación de diferentes técnicas de clasificación de imágenes basadas en visión artificial con la finalidad de poder analizar la posible automatización del proceso de control de calidad de naranjas Navel en línea, durante el proceso de encajado en un almacén.

Se parte de un conjunto de datos etiquetados suministrado por el instituto de automática e informática industrial (ai2). Se trata de un "*dataset*" adquirido en un almacén en el que se dispone de un sistema de visión artificial que captura 5 imágenes por naranja navel durante el paso por railes que conducen a las naranjas a su encajado final. Cada secuencia de imágenes está etiquetada por su categoría.

Para el desarrollo del trabajo, se han diseñado técnicas de clasificación, por ejemplo: clasificador bayesiano, KNN, redes neuronales, etc. Estos métodos de clasificación se implementarán con lenguaje de programación Python y sus librerías Numpy, OpenCV, Pytorch, etc.

Para finalizar, se han evaluado los resultados obtenidos utilizando criterios como la tasa de acierto y coste computacional.

Palabras clave: Automatización, Visión Artificial, Control de calidad, Clasificación de naranjas naveles.

RESUM

L'objectiu d'aquest treball consisteix en el disseny, implementació i avaluació de diferents tècniques de classificació d'imatges basades en visió artificial amb la finalitat de poder analitzar la possible automatització del procés de control de qualitat de taronges Navel en línia, durant el procés d'encaixat en un magatzem.

Es parteix d'un conjunt de dades etiquetades subministrat per l'institut d'automática i informàtica industrial (ai2). Es tracta d'un data set adquirit en un magatzem en el qual es disposa d'un sistema de visió artificial que captura 5 imatges per taronja naval durant el pas per rails que condueixen a les taronges al seu encaixat final. Cada seqüència d'imatges està etiquetada per la seua categoria.

Per al desenvolupament del treball, s'han dissenyat tècniques de classificació, per exemple: classificador bayesià, KNN, xarxes neuronals, etc. Aquests mètodes de classificació s'implementaran amb llenguatge de programació Python i les seues llibreries Numpy, OpenCV, Pytorch, etc.

Per a finalitzar, s'han evaluat els resultats obtinguts utilitzant criteris com la taxa d'encert i cost computacional.

Paraules clau: Automatització, Visió Artificial, Control de qualitat, Classificació de taronges naveles.

ABSTRACT

The objective of this project is the design, implementation and evaluation of different image classification techniques based on artificial vision in order to analyse the possible automation of the quality control process of Navel oranges during the packing process in a storage.

The starting point is a labelled dataset supplied by the institute of automatics and industrial informatics (ai2). It is a dataset acquired in a storage with an artificial vision system that captures 5 images per navel orange during the passage along the rails that lead the oranges to their final packing. Each sequence of images is labelled by category.

For the development of the work, classification techniques had been designed, e.g., Bayesian classifier, KNN, neural networks, etc. these classification methods will be implemented with Python programming language and its libraries: Numpy, OpenCV, Pytorch, etc.

Finally, the results obtained had been evaluated using criteria such as hit rate and computational cost.

Keywords: Automation, Artificial Vision, Quality Control, Classification of navel oranges.

Índice de la memoria

Listado de figuras……9

Listado de ecuaciones……10

Listado de gráficas……10

Capítulo 1: Introducción……13

1.1 Introducción general……13
1.1.1. Visión artificial……13
1.1.2. Motivación……14
1.2 Inteligencia Artificial……14
1.2.1. ¿Qué es la Inteligencia artificial? ……14
1.2.2. Campos de la IA……14
1.2.3. "Machine Learning" ……14
1.2.3.1. ¿Qué es el "Machine Learning"?.....14
1.2.3.2. Tipos de aprendizaje……14
1.2.3.3. Algoritmos de "Machine Learning"……16
1.2.3.4. 1.2.3.4. ML + Visión artificial……16
1.2.4. Redes Neuronales Artificiales……16
1.2.4.1. ¿Qué es una Red Neuronal Artificial?.....16
1.2.4.2. Arquitectura de una red neuronal……16
1.2.4.3. Perceptrón……18
1.2.4.4. Perceptrón multicapa……18
1.2.5. Entrenamiento……19
1.2.5.1. Descenso del gradiente……19
1.2.6. Red neuronal convolucional……21
1.2.6.1 Capas convolucionales……21
1.2.6.2 Convolución……21
1.2.6.3 Conexión entre las capas convolucionales y el perceptrón multicapa...22
1.3 Visión artificial en una planta de manipulación de cítricos……23
1.3.1. Línea de manipulación de cítricos……23
1.3.2. Visión artificial en la selección……25

Capítulo 2: Objetivos……27

2.1 Objetivo general……27
2.2 Objetivos específicos……27
2.3 Criterios de valoración de resultados……27

Capítulo 3: Métodos……29

3.1 "Dataset" de imágenes……29
3.1.1. Etiquetado de las imágenes……30
3.1.2. Preprocesamiento de las imágenes……30
3.2 Red neuronal convolucional ResNet……31
3.3 Ajuste de hiperparámetros……34
3.3.1. Número de épocas……34
3.3.2. Tamaño del lote……35
3.3.3. Optimización……35

3.3.4. Ajuste de hiperparámetros……..……36
3.4 Ejecución……..……36
3.4.1. Pytorch……37
3.4.2. Entorno de ejecución……..……38

Capítulo 4: Resultados……..……39

4.1 “Dataset” RGB……39
4.2 “Dataset” IR……41
4.3 “Dataset” con concatenación vertical RGB + IR……43
4.4 “Dataset” con concatenación horizontal RGB + IR……45
4.5 OPENCV vs ResNet……47
4.5.1. Coste temporal……47
4.5.2. Precisión……..…48
4.6 Análisis de resultados……...…48
4.6.1. Coste temporal……48
4.6.2. Precisión…….49
4.6.3. OpenCV VS ResNet18……...49

Capítulo 5: Conclusiones…….50

Bibliografía……...…52

Índice del presupuesto

Contenido del presupuesto……...……55

- Coste personal……..…56
- Coste Material Inventariable……...56
- Presupuesto de Ejecución……56

Listado de figuras

Figura 1. Aprendizajes supervisado y no supervisado. Fuente: propia……15

Figura 2. Campeones del mundo de ajedrez y go contra una inteligencia artificial. Fuente: https://blog.andresnunez.com/inteligencia-artificial-el-reto-evolutivo-de-la-educacion-superior/ ……..16

Figura 3. Esquema de una neurona artificial con su equivalente biológico. Fuente: https://www.um.es/LEQ/Atmosferas/Ch-VI-3/F63s4p3.htm…..…….17

Figura 4. Ilustración de la función *ReLu* y su ecuación. Fuente: propia…….…….17

Figura 5. Esquema Perceptrón. Fuente: https://medium.com/nerd-for-tech/flux-prediction-using-single-layer-perceptron-and-multilayer-perceptron-cf82c1341c33…..…….18

Figura 6. Esquema de un perceptrón multicapa con una función *Softmax*. Fuente: https://juansensio.com/blog…….……19

Figura 7. Algoritmo de "*Backpropagation"*. Fuente: https://medium.com/latinxinai/un-lego-a-la-vez-explicando-la-matemática-de-como-las-redes-neuronales-aprenden-ae582ab91da6……20

Figura 8. Derivadas parciales de la función de pérdida. Fuente: propia…….…….20

Figura 9. Ejemplo de función de pérdida. Fuente: https://juansensio.com/blog…..…….21

Figura 10. Esquema del procesamiento de una imagen al pasar por una capa convolucional. Fuente: Michigan Online…….…….22

Figura 11. Aplanamiento de la última capa de convolución y la entrada al perceptrón multicapa. Fuente: propia……..…….23

Figura 12. Fruta en un supermercado que ha pasado por una línea de manipulación de alimentos. Fuente: Línea de manipulación: lavado, tratamiento, selección, calibrado y envasado…….…..23

Figura 13. Certificados de calidad alimentaria. Fuente: http://www.jcolomer.com/es/certificacion-brc--ifs.aspx…….……24

Figura 14. Esquema de una línea de manipulación de alimentos. Fuente: …….…….24

Figura 15. Sistema óptico de selección en una línea de manipulación de cítricos. Fuente: Línea de manipulación: lavado, tratamiento, selección, calibrado y envasado…….…….26

Figura 16. Muestra del "dataset" de naranjas Navel. Fuente: propia……..…….26

Figura 17. Muestra de imagen RGB del "dataset" de naranjas Navel. Fuente: propia……29

Figura 18. Muestra de imagen RGB del "dataset" de naranjas Navel. Fuente: propia……....29

Figura 19. Capturas de naranjas Navel para el "dataset ". Fuente: propia……..…….30

Figura 20. Concatenación de capturas para crear el elemento de entrenamiento. Fuente: propia……....30

Figura 21. Concatenación vertical de imágenes RGB e IR del "dataset". Fuente: propia……....31

Figura 22. Concatenación horizontal de imágenes RGB e IR del "dataset". Fuente: propia…...31

Figura 23. Resultados del ganador del reto ILSVRC a lo largo del tiempo. Fuente: propia …. 32

Figura 24. Comparativa entre redes neuronales convolucionales planas con diferente número de capas. Fuente: Deep Residual Learning for Image Recognition……..33

Figura 25. Esquema “Residual Learning”. Fuente: Deep Residual Learning for Image Recognition……..…….33

Figura 26. Redes neuronales condiferentes arquitecturas. Fuente: Deep Residual Learning for Image Recognition …….…..34

Figura 27. diferencias entre "batch", "mini-batch" y "stochastic" gradient descent. Fuente:…..35

Figura 28. Optimización de pesos. Fuente: https://juansensio.com/blog ……....36

Figura 29. Logo de Pytorch. Fuente: https://navaneethsdk.medium.com/getting-started-with-pytorch-4fa1a5ee1502 ……...……..37

Figura 30. Logo de Google Colab. Fuente: https://www.cursosgis.com/que-es-google-colab/ ……..…37

Listado de ecuaciones

Ecuación 1. Función de pérdida de entropía cruzada. Fuente: propia……..….20

Listado de gráficas

Gráfica 1. Función de coste "dataset" RGB de entrenamiento. Fuente: propia……....40

Gráfica 2. Función de coste "dataset" RGB de entrenamiento y validación. Fuente: propia…...40

Gráfica 3. Precisión de la red con el "dataset" RGB de entrenamiento y validación. Fuente: propia……40

Gráfica 4. Matriz de confusión de la red entrenada con el "dataset" RGB. Fuente: propia….....41

Gráfica 5. Función de coste "dataset" IR de entrenamiento. Fuente: propia……42

Gráfica 6. Función de coste "dataset" IR de entrenamiento y validación. Fuente: propia…...…42

Gráfica 7. Precisión de la red con el "dataset" IR de entrenamiento y validación. Fuente: propia……42

Gráfica 8. Matriz de confusión de la red entrenada con el "dataset" IR. Fuente: propia…….43

Gráfica 9. Función de coste "dataset" concatenado verticalmente de entrenamiento. Fuente: propia……44

Gráfica 10. Función de coste "dataset" concatenado verticalmente de entrenamiento y validación. Fuente: propia……...……44

Gráfica 11. Precisión de la red con el "dataset" concatenado verticalmente de entrenamiento y validación. Fuente: propia……44

Gráfica 12. Matriz de confusión de la red entrenada con el "dataset" concatenado verticalmente. Fuente: propia……...45

Gráfica 13. Función de coste "dataset" concatenado horizontamente de entrenamiento. Fuente: propia……...….46

Gráfica 14. Función de coste "dataset" concatenado horizontalmente de entrenamiento y . Fuente: propia validación……..…….46

Gráfica 15. Precisión de la red con el "dataset" concatenado horizontalmente de entrenamiento y validación. Fuente: propia……46

Gráfica 16. Matriz de confusión de la red entrenada con el "dataset" concatenado horizontalmente. Fuente: propia……...47

Gráfica 17. Comparación temporal ResNet VS OpenCV. Fuente: propia……...47

Gráfica 18. Comparación precisión ResNet VS OpenCV. Fuente: propia……...…48

Gráfica 19. Comparación temporal entre “datasets” de entrenamiento. Fuente: propia……..48

Gráfica 20. comparación precisión entre “datasets” de validación. Fuente: propia…….49

DOCUMENTO I: MEMORIA

Capítulo 1: Introducción

En este primer apartado se expondrá el marco teórico de este proyecto. En él introduciremos los conceptos de Inteligencia Artificial, Visión Artificial, Aprendizaje Automático y redes neuronales que son sus grandes pilares. Se expondrá con detenimiento el funcionamiento y tipos de redes neuronales. Y, finalmente, se verá cómo funciona una línea de manipulación de cítricos y como se implementaría un sistema óptico basado en redes neuronales convolucionales para resolver un problema de clasificación.

1.1. Introducción general:

Si hay un campo que en los últimos 10 años no hemos parado de escuchar es el de la Inteligencia Artificial (IA). Esta amplia subdisciplina de la automática se nos ha presentado como una revolución para el mundo en todos los aspectos debido a su posible aplicación de forma transversal en nuestras vidas.

Los asistentes programados por empresas como Apple, Google y Amazon han sido los primeros en instalarse en nuestras vidas. Y aunque al principio solo eran capaces de hacer búsquedas en internet y hacer tareas simples en nuestros teléfonos, han evolucionado para poder controlar algunos aparatos en hogares domotizados, analizar nuestras rutinas para optimizar el uso de la batería y otras muchas cosas.

Está claro que este no es el único uso de la inteligencia artificial, pero si es un primer paso, una primera inclusión de esta tecnología que nos permite a todos conocerla y tolerar su inclusión en diferentes aspectos de nuestras vidas, como ya lo están haciendo con las predicciones del tiempo, que hace uso de superordenadores para modelizar.

En el horizonte podemos encontrar un gran número de proyectos muy interesantes que nos brinda esta tecnología. Con la expansión de las redes 5G, las "*Smart Cities*" son un proyecto de domotización de una ciudad. Una ciudad que sepa adaptarse a como la viven sus ciudadanos: tráfico, frecuencia del transporte público y publicidad de eventos en función a tus gustos son algunas de las tareas a llevar a cabo.

La inteligencia artificial es una realidad en muchas empresas del mundo tecnológico, ayudando la gestión de las mismas y en la creación de nuevos productos simulando resultados: arquitecturas de chips, asistentes entrenándose a sí mismos, redes neuronales que simulan diferentes números de neuronas y capas para optimizar su funcionamiento, etc. Pero al igual que hicieron las maquinas durante la revolución industrial, la IA puede ser introducida en todo tipo de empresa para automatizar procesos que nunca habríamos pensado como el uso de nuestro criterio. Dando lugar al nacimiento de un nuevo tipo de industria, la industria 4.0.

Este es un proyecto con el nombre de la Comunidad Valenciana, ya que en él uniremos dos campos de la inteligencia artificial (Visión Artificial y *"Machine Learning"*) para la clasificación de la naranja Navel en función de la calidad de su aspecto. De esta forma se separarán en 5 categorías diferentes, siendo la 1 la que tenga mejor color y menos imperfecciones y la 5 la que más desperfectos contenga.

1.1.1. Visión artificial:

El campo de la visión artificial consiste en la comprensión de una imagen o un entorno por parte de una máquina.

Es un campo muy conocido dentro de la inteligencia artificial, ya que es algo que tenemos integrado ya en nuestros "*smartphones*" (el reconocimiento facial) o hemos oído hablar de los coches autónomos. Estos últimos utilizan algoritmos de visión artificial para el reconocimiento de coches, carriles y señales, dando la posibilidad al conductor de no tener que prestar atención a la carretera. Incluso para detectar si el conductor está cansado o sufre alguna indisposición.

1.1.2. Motivación:

El propósito de este proyecto es el propio aprendizaje de esta tecnología, como implementarla y el alcance de la misma.

Quiero ver como puede ser utilizada en todo tipo de industria y desmentir la necesidad de procesos sofisticados y costosos para la inclusión de la inteligencia artificial en todo tipo de empresas. Resumiendo, mostrar la transversalidad y plasticidad de esta tecnología.

1.2. Inteligencia Artificial

1.2.1. ¿Qué es la Inteligencia artificial?

"La Inteligencia Artificial (IA) es la subdisciplina del campo de la informática que busca la creación de máquinas que puedan imitar comportamientos inteligentes" – Tatiana Grapsas, Rock content.

Dichas inteligencias artificiales pueden ser clasificadas en dos grandes grupos: débiles y fuertes. Las débiles son las que el número de tareas que pueden realizar es limitado y las únicas que conocemos actualmente. Las fuertes pueden aplicarse a una gran variedad de tareas y a las que la ficción les ha dado forma.

1.2.2. Campos de la IA:

Lo que entendemos como comportamiento inteligente es lo que les da nombre a los diferentes campos de esta disciplina: la capacidad de moverse y adaptarse al entorno, es la robótica; la comprensión de imágenes y entornos, la visión artificial; la capacidad de comprender el lenguaje, es el Procesamiento de Lenguaje Natural, "*Natural Language Processing*" (NPL); la conversión de voz a texto o el texto a voz, y la capacidad de aprendizaje, el Aprendizaje Automático, *"Machine Learning*" (ML). Siendo este último y el de visión artificial los protagonistas en este proyecto.

1.2.3. *"Machine Learning"*

1.2.3.1. ¿Qué es el *"Machine Learning"*?

Como se ha comentado anteriormente, el ML es el campo de la IA que dota de capacidad de aprendizaje a las máquinas. Entendiendo como aprendizaje la mejora de resultados en una tarea a través de la experiencia.

1.2.3.2. Tipos de aprendizaje:

Para que una inteligencia artificial obtenga experiencia es necesario entrenamiento. Los más utilizados son:

- **Aprendizaje supervisado:** en este tipo de entrenamiento se proporciona una serie de entradas a nuestro algoritmo junto con las salidas esperadas a cada input. El objetivo de

este método es que, durante el entrenamiento, la red neuronal sea capaz de modificar sus parámetros para adaptar lo mejor posible las salidas obtenidas con las esperadas, siendo capaz de predecir su "etiqueta" con tan solo la información entrante.

- **Aprendizaje no supervisado:** a diferencia del aprendizaje supervisado, a las redes neuronales que sigan este método de entrenamiento no se les proporcionará la salida esperada a cada input, sino que la red debe ser capaz de encontrar relaciones estructurales entre los "*inputs*". De esta forma creará un modelo que relacione matemáticamente las entradas de datos proporcionadas durante el entrenamiento.

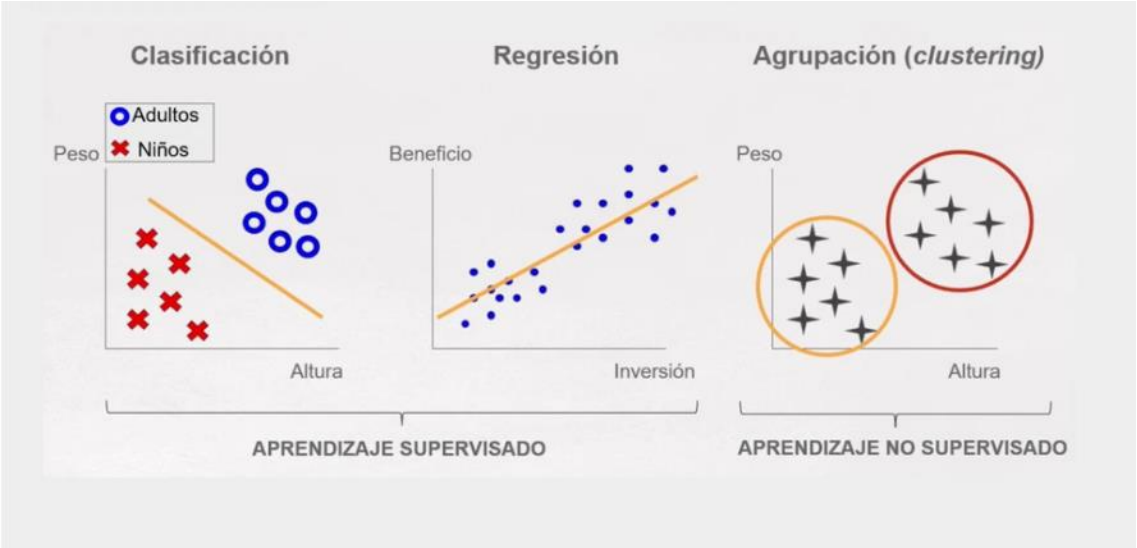


Figura 1. Aprendizajes supervisado y no supervisado

- **Aprendizaje reforzado:** probablemente este método de trabajo es el que levante más interés. No porque sea el más utilizado o el que mejores resultados obtenga, sino porque utiliza un método de aprendizaje que es muy similar a la forma en como los seres humanos aprendemos. Este cuenta con un sistema de señales de recompensa o penalización que se activan en función de los resultados a diferentes tareas.

  A nivel de publicidad de la inteligencia artificial han tenido un papel muy importante, ya que es el método utilizado para que las máquinas aprendan a jugar a juegos, incluso llegando a batir a los mejores jugadores humanos. Algunos de los ejemplos más conocidos es el de Deep Blue, el ordenador diseñado por IBM, que fue capaz de ganar al entonces campeón del mundo de ajedrez Kaspárov, o AlphaGo que dominó el juego del Go sorprendiendo al mundo por su solidez en un juego mucho más complicado que el ajedrez.

Figura 2. Campeones del mundo de ajedrez y go contra una inteligencia artificial

Para la tarea de clasificación que vamos a desarrollar más adelante implementaremos un entrenamiento supervisado a nuestro algoritmo de *"Machine Learning"*.

1.2.3.3 Algoritmos de *"Machine Learning"*

Hay diferentes maneras de hacer que una maquina pueda aprender en base a su experiencia como: arboles de decisión, algoritmos de regresión, bayesianos, etc. Sin embargo, los más populares y los que han producido los mayores avances en el campo de *"Machine Learning"* son, sin duda, las redes neuronales. Y será con el que trabajaremos en este proyecto.

1.2.3.4. ML + Visión artificial:

Como se ha dicho anteriormente, los dos campos de la IA que son los protagonistas en este trabajo son la visión artificial y el *"Machine Learning"*. Nuestro algoritmo será entrenado para clasificar imágenes de naranjas de tipo Navel pero para que nuestra máquina aprenda será necesario realizar una serie de transformaciones para pasar la información que obtiene la imagen a datos numéricos.

La tecnología que usaremos, las redes neuronales, necesitan entradas digitales para su funcionamiento. Sin embargo, en un principio estos datos estarán en forma de imagen, por lo que serán necesarias algunas transformaciones. Más adelante, veremos las peculiaridades de nuestro algoritmo, que le diferencia de otras redes neuronales.

1.2.4. Redes Neuronales Artificiales:

1.2.4.1. ¿Qué es una Red Neuronal Artificial?

Las redes neuronales artificiales son algoritmos de *"Machine Learning"* que consisten en neuronas ordenadas en capas que procesan la información que entra y generan una salida correspondiente a la entranda. Reciben este nombre porque intentan imitar artificialmente la forma de procesamiento de la información en sistemas nerviosos biológicos. A diferencia de cómo opera un ordenador, el cerebro humano es un sistema complejo, no lineal y paralelo. Esta forma de procesamiento simultáneo es lo que una red neuronal artificial trata de imitar.

La arquitectura de este algoritmo consiste en un conjunto de neuronas ordenadas en capas e interconectadas que generan una red que es capaz de aprender en base a su experiencia, plástica y adaptable, tolerante a fallos y con un comportamiento no lineal que le permite procesar información procedente de otros fenómenos no lineales.

1.2.4.2. Arquitectura de una red neuronal:

- **Neurona**: es el elemento más pequeño de este algoritmo y un conjunto de ellas es lo que formará nuestra red neuronal. Matemáticamente hablando, una neurona es simplemente una función lineal: $y = w \cdot x + w_0$, una forma básica de obtener una salida proporcional a

una entrada. Cada función lineal tendrá el mismo número de variables que entradas y unos pesos que variarán con el entrenamiento de la red y una variable independiente. Pero como la suma de una serie de funciones lineales da como resultado una función lineal y las redes neuronales tienen un carácter no lineal es necesario que pasen por una función de activación.

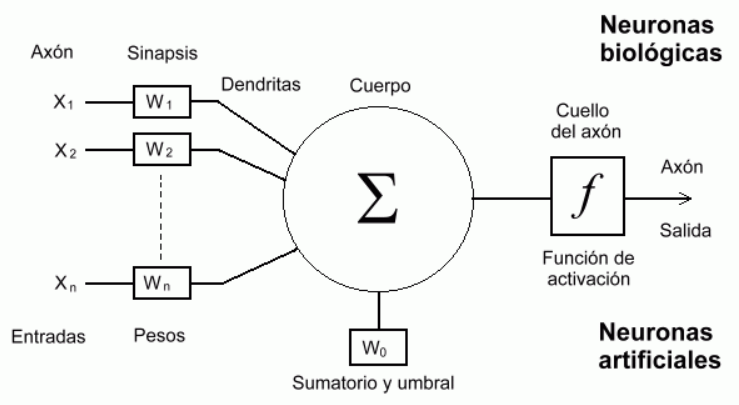


Figura 3. Esquema de una neurona artificial con su equivalente biológico

- **Función de activación**: convierten la función lineal de la neurona en no lineal para mantener el carácter individual de la propia neurona en la red.

  Las funciones de activación pueden ser lineales, para modelos de regresión lineal, y no lineales que adaptan a la no linealidad de los datos de trabajo. Las funciones no lineales más usadas en el ámbito del *"Machine Learning"* son: la función escalonada, sigmoide, la tangente hiperbólica y la función ReLu. La función escalonada es muy usada en tareas de clasificación binaria. La función sigmoide, como su rango es de 0 a 1 se interpreta como una probabilidad y se usa en las últimas capas para la clasificación final. Mientras, la función ReLu permite un entrenamiento más rápido de las redes neuronales. Es por eso que es la más usada en este campo y la que utilizaremos en el código de nuestro algoritmo.

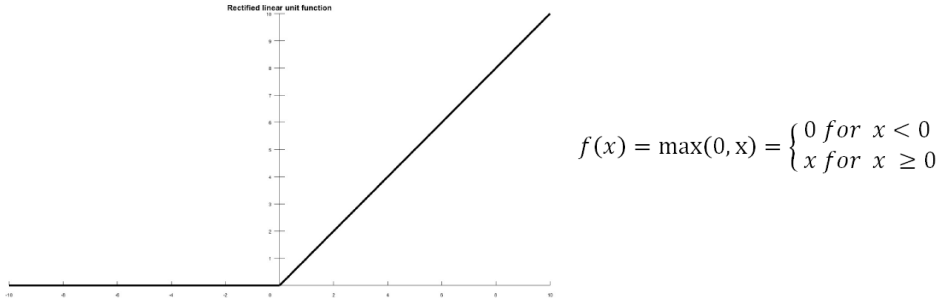


Figura 4. Ilustración de la función ReLu y su ecuación

- **Capa**: es el conjunto de neuronas que no están conectadas entre sí y que se encargan de marcar la jerarquía en el aprendizaje de la red neuronal. Conforme la red está siendo entrenada estas capas se van especializando en la detección de ciertas características.

  Para dar solución a problemas complejos, el número de neuronas en la red debe ser mayor, organizándose también en un número mayor de capas. Esta evolución de la red ha dado lugar a los algoritmos conocidos como: algoritmos de "*Deep Learning*".

1.2.4.3. Perceptrón:

Es la forma básica de nuestro sistema. El esquema del perceptrón nos muestra el flujo de datos en el paso por una neurona. La neurona recibe el vector de entrada de datos (x), que puede ser tanto de la entrada de la red o de una capa anterior, la multiplica por un vector de pesos (w), suma el resultado de todas las entradas por sus pesos correspondientes y el resultado pasa por la función de activación.

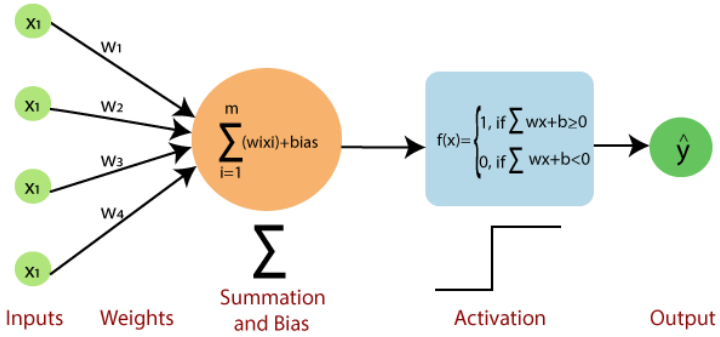


Figura 5. Esquema Perceptrón

1.2.4.4. Perceptrón multicapa:

El esquema del perceptrón multicapa muestra la forma que solemos relacionar con una red neuronal. En este vemos la organización de las neuronas en capas y como se conectan unas con otras. La organización de las capas es la siguiente:

1) **Capa de entrada**: es la primera capa de la red. En esta capa el número de neuronas debe ser el mismo que el número de entradas se proporcionen a la red.
2) **Capas ocultas**: con un mínimo de 1 capa hasta un número ilimitado de ellas, las capas centrales de la red neuronal son las que se encargan en mayor parte del aprendizaje. A mayor número de capas, mayor es la complejidad del algoritmo. Conforme van apareciendo capas el aprendizaje se vuelve más abstracto, de forma que las primeras capas sean capaces de identificar características más elementales, mientras que las siguientes sean capaces de detectar formas y objetos más complejos dando lugar a un aprendizaje jerarquizado. La interconexión entre capas no sigue unas reglas específicas más que cada neurona debe recibir y dar información a otra neurona. Las neuronas no tienen por qué estar conectadas a las capas adyacentes (aunque es lo más común) sino que pueden saltarse capas o conectarse a capas anteriores.
3) **Capa de salida**: el número de neuronas utilizadas en la última capa depende de la finalidad de la red neuronal. En nuestro caso vamos a realiza una tarea de clasificación de la naranja navel en 5 subcategorías, por lo que el número de neurona en la última capa será de 5. El objetivo es que cuando la red neuronal reciba una imagen genere una salida mayor en la neurona a la que corresponda la categoría de la naranja del input. Más adelante veremos cómo evitaremos que nuestro algoritmo se confunda al obtener salidas de un valor similar gracias a la función "*Softmax"*.

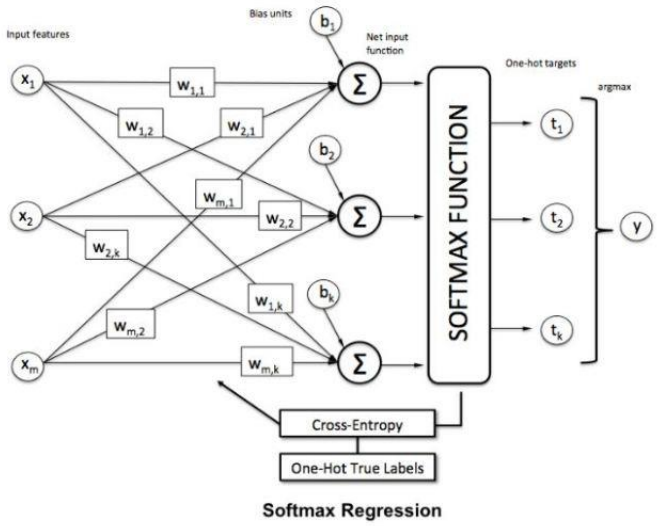


Figura 6. Esquema de un perceptrón multicapa con una función Softmax

Cuando una red neuronal evoluciona en su complejidad integrando un gran número de neuronas y capas, es capaz de realizar tareas cada vez más complejas aprendiendo de forma jerarquizada y por niveles. Esto quiere decir que durante el entrenamiento las primeras capas de la red se especializan, por ejemplo, en la detección de distintos elementos para que las siguientes se especialicen en tareas más abstractas, como en la distribución de los elementos a detectados. A estos algoritmos más complejos son a los que se les llama algoritmos de aprendizaje profundo, "*deep learning*"

1.2.5. Entrenamiento:

Para adaptar a una red neuronal a la tarea que va a desempeñar es necesario entrenarlas con algunos de los métodos de aprendizaje que hemos visto anteriormente. Tras este entrenamiento, la red neuronal habrá creado un modelo capaz de hacer predicciones con los datos de entrada en la red. A continuación, veremos cómo es la creación de estos modelos, las técnicas y los cambios que se producen en la red neuronal para adecuarla a las tareas a las que se va a dedicar.

El papel de la red neuronal es generar una respuesta a partir de una entrada de datos. A nivel elemental, la forma en la que conseguimos esto en una neurona es con una función lineal, aunque posteriormente será modificada por la función de activación escogida. Es en el interior de esas funciones lineales donde encontramos la clave del aprendizaje de una red neuronal. Estas poseen una serie de pesos (*w*) que pueden ser modificados para variar la respuesta dada a una serie de datos de entrada, "*inputs*".

Ya que modificar los pesos de forma aleatoria en busca de mejores resultados sería una tarea muy ineficiente, es necesaria la aplicación de técnicas que nos indique cómo y en qué neuronas deben ser los cambios en los pesos. Para que los cambios en los pesos sean consecuentes a los resultados obtenidos utilizaremos el algoritmo del descenso del gradiente.

1.2.5.1. Descenso del gradiente:

El algoritmo del descenso del gradiente parte de la selección una función de pérdida. En nuestro caso es capaz de cuantificar la desviación de la etiqueta obtenida y la esperada.

$$\text{Loss} = -\frac{1}{\substack{\text{output}\\\text{size}}} \sum_{i=1}^{\substack{\text{output}\\\text{size}}} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)$$

Ecuación 1. Función de pérdida de entropía cruzada.

En primer lugar, es necesario el volcado de información desde la entrada de la red para generar las salidas correspondientes con los pesos por defecto de la red o iniciados aleatoriamente. Una vez se ha completado el flujo normal de información se hará una comparación de los resultados que hemos obtenido y los deseados.

Una vez obtenemos los resultados, mediante la derivada de la función de coste que hemos comentado antes somos capaces de ver cómo afectan los pesos de la última capa de la red neuronal en el resultado, lo que afectará a la variación de estos pesos. Este algoritmo puede propagarse hacia las capas anteriores aplicando la regla de la cadena llamando a este proceso Retropropagación o "*Backpropagation*".

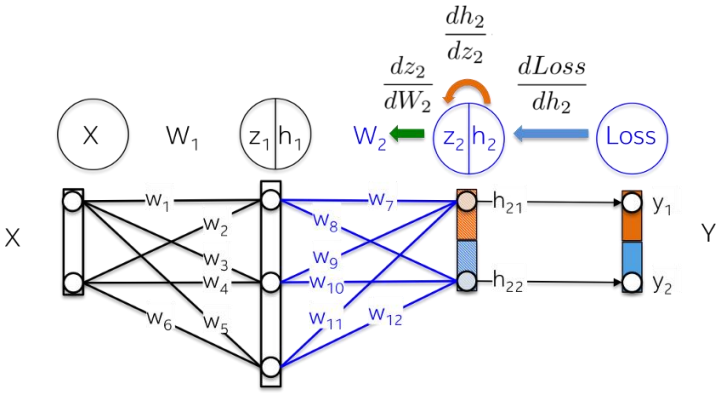


Figura 7. Algoritmo de Backpropagation.

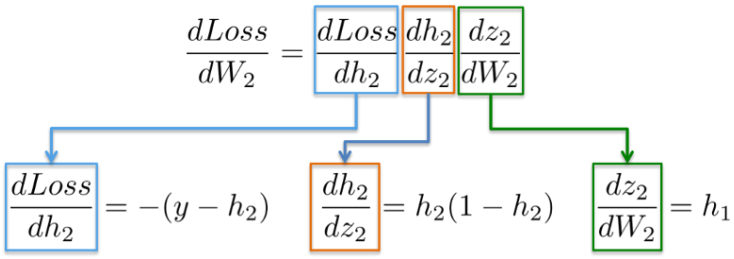


Figura 8. Derivadas parciales de la función de pérdida

El algoritmo de descenso por gradiente no nos garantiza encontrar el valor óptimo global y puede tomar como valor óptimo un mínimo local. Por eso es necesario que se realicen múltiples entrenamientos para encontrar el mejor resultado posible en las predicciones. Incluso ajustar algunos hiperparámetros que utilizará nuestro algoritmo para modificar los pesos.

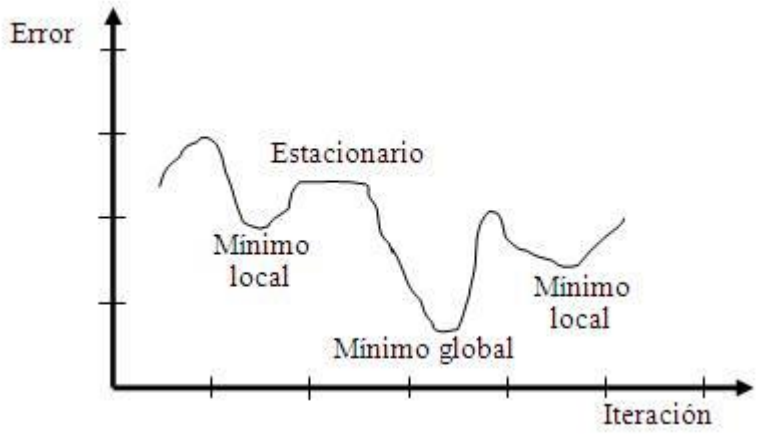


Figura 9. Ejemplo de función de pérdida.

1.2.6. Red neuronal convolucional

Las redes neuronales convolucionales con la comunión entre la Visión Artificial y el ML. El "*input*" de las redes neuronales que hemos presentado son una cadena de valores numéricos de una dimensión. Sin embargo, los datos que tenemos de entrada son imágenes RGB, y en blanco y negro en el espectro infrarrojo, por lo que no puede relacionar la información de un pixel con la posición en la que se encuentra ni con la información de los pixeles de su alrededor. Es por eso que antes del paso de la información por un perceptrón multicapa es necesario que sufra algunas transformaciones espaciales.

El funcionamiento de este tipo de red neuronal tiene algunas similitudes con el cerebro humano. Al igual que en la corteza visual primaria, se encargan de la detección de bordes y su orientación. Esto, aunque parezca muy simple, será el primer paso hacia la detección de características mucho más complejas.

La transformación de las imágenes se lleva a cabo dentro de la propia red neuronal, solo que es necesario implementar un tipo especial de capas: las capas convolucionales.

1.2.6.1. Capas convolucionales:

Las redes neuronales convolucionales consisten en múltiples capas de filtros o "*kernerls"* de una o más dimensiones que se encargarán de realizar múltiples transformaciones en la imagen inicial.

Las imágenes pasarán sucesivamente por una fase de extracción de características, compuesta por neuronas convolucionales, luego hay una reducción por muestreo y finalmente tendremos neuronas de perceptrón como hemos descrito anteriormente.

La fase de extracción de características se compone de capas alternas de neuronas convolucionales y neuronas de reducción por muestreo. Las imágenes que pasan sucesivamente por estas capas van disminuyendo su dimensionalidad de forma que las ultimas neuronas son menos sensibles a perturbaciones en los "*inputs"*, pero son activadas por características cada vez más complejas.

1.2.6.2. Convolución:

En primer procesamiento de la imagen se realizará una normalización de los colores de la imagen convirtiendo los valores entre 0 y 255 a valores entre 0 y 1.

Después, el proceso de convolución consiste en multiplicar matricialmente los valores de cada "*kernel"* por los valores de un conjunto de píxeles normalizados haciendo un barrido por toda la

imagen. El producto resultante de la convolución será una nueva imagen llamada mapa de características.

Siguiendo un ejemplo, una imagen de 28x28 en blanco y negro necesitará 28x28x1=784 pesos convolucionales por cada "*kernel"* y el triple si se trata de una imagen a color. Por lo que si tenemos una primera capa con 32 "*kernels"* el número de pesos utilizados sería de 25.088, lo que supone mucho poder computacional. Es por eso que es necesaria la nombrada fase de reducción por muestreo.

En la fase de muestreo, la convolución del "*kernel"* pasa por la ya conocida función de activación. La más utilizada es la *ReLu*, que eliminará los valores negativos obtenidos. El mapa de características resultante es al que realizaremos el muestreo o "*subsampling"*.

Hay diversos tipos de muestreo, pero el más usado es el "*Max-Pooling"*. Esta técnica de muestreo agrupa un conjunto más o menos pequeño de datos de entrada y solamente se queda con la que mayor valor tiene. Preservando así las características más importantes que ha detectado cada filtro.

Finalmente, la salida de la capa convolucional tendrá como resultado tantas imágenes o mapa de características como "*kernels"* había en la fase de extracción de características, pero de un tamaño tantas veces menor como el tamaño de muestreo (e.g., un tamaño de muestreo 2x2 supone dividir el tamaño de la imagen entre 4)

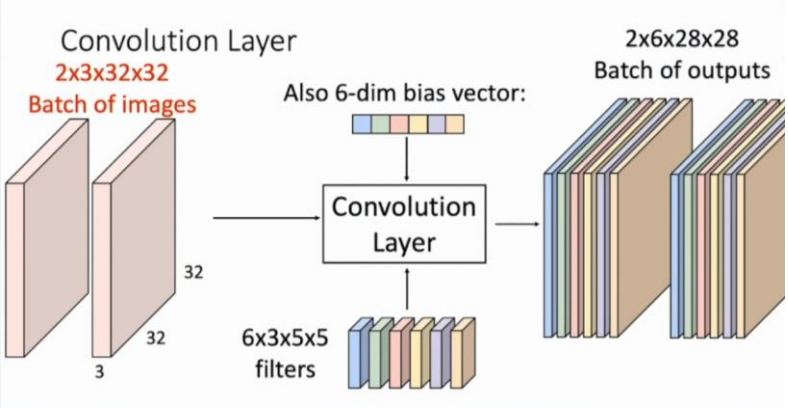


Figura 10. Esquema del procesamiento de una imagen al pasar por una capa convolucional

1.2.6.3. Conexión entre las capas convolucionales y el perceptrón multicapa:

Conforme la imagen pase sucesivamente por diferentes capas de "Max-Pooling" el tamaño de los mapas de características será cada vez menor aumentando el número de ellas. Sin embargo, los mapas de características resultantes son tridimensionales, por lo que se les aplicará un aplanamiento o *flattering*, creando un vector resultante de los mapas que será el *input* del perceptrón multicapa ya conocido.

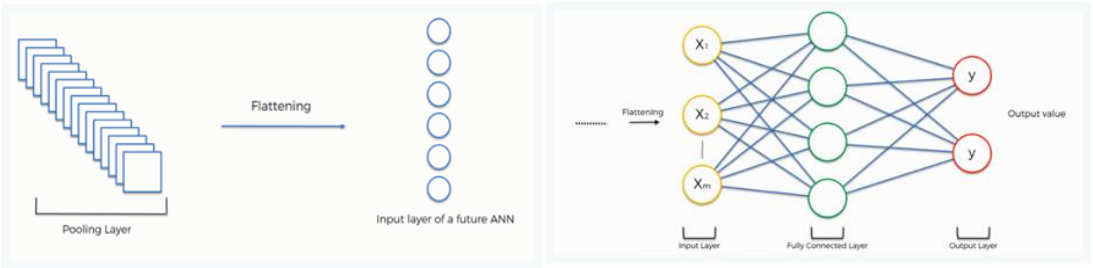


Figura 11. Aplanamiento de la última capa de convolución y la entrada al perceptrón multicapa

1.3. Visión artificial en una planta de manipulación de cítricos:

Este proyecto tiene como fin demostrar la posible implementación de la visión artificial en una línea de manipulación de cítricos de la variedad Navel. Para entender cómo esta tecnología puede integrarse en el proceso es necesario conocer el funcionamiento de la línea de manipulación de cítricos y las tareas que puede realizar dentro de la misma.

1.3.1. Línea de manipulación de cítricos:

Una línea de manipulación de cítricos cuenta con una serie de procesos y sus correspondientes equipos con el fin de preparar el producto obtenido del campo y encajarlo para su futura comercialización. Los procesos que van a llevarse a cabo serán: la limpieza de la fruta, tratamientos con fungicidas y desinfectantes, selección de la fruta por su calidad en función de su aspecto, tamaño y peso, y su empaquetado correspondiente. Todos estos procesos deben garantizar la correcta comercialización de la fruta, la cual está reglamentada en la UE 543/2011, que tiene el fin de ofrecer al consumidor un producto de calidad, sin defectos externos, limpios y con una buena presentación.

Figura 12. Fruta en un supermercado que ha pasado por una línea de manipulación de alimentos

Además, los principales comercializadores de fruta exigen a estas centrales hortofrutícolas unos protocolos de calidad y seguridad alimentaria certificando al producto como que cumple una serie de características que consideran relevantes para los consumidores, para la calidad, la salud o el medio ambiente (BRC, IFS, …).

Figura 13. Certificados de calidad alimentaria

Siguiendo el recorrido que realiza la naranja en la línea de manipulación podemos distinguir diferentes etapas:

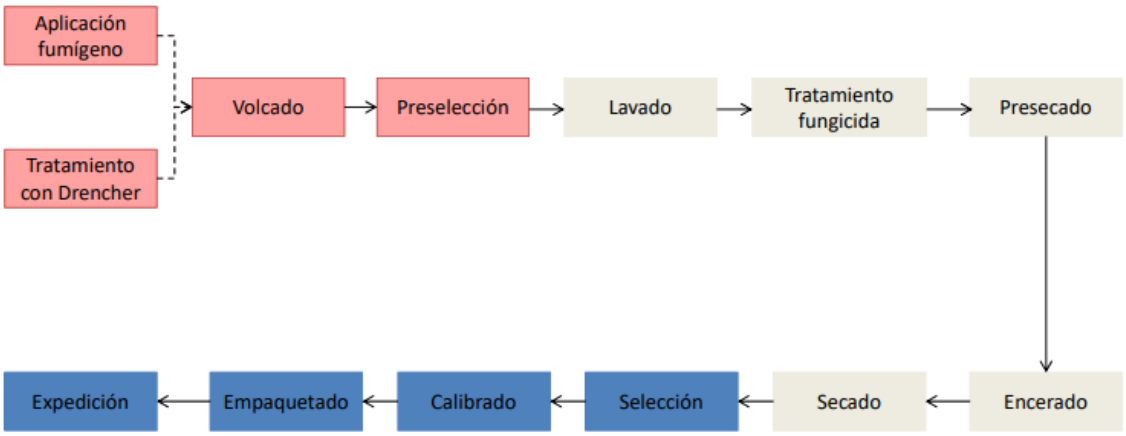


Figura 14. Esquema de una línea de manipulación de alimentos.

En un tratamiento previo se desinfecta la fruta mediante un lavado en el "*Drencher"* que circula agua por la fruta todavía en los palés con una dosis fungicidas y aditivos alimentarios. Como alternativa de tratamiento en seco está el Fumígeno, que aplica el fungicida y aditivos alimentarios autorizados directamente sobre la fruta. Después, se hace el volcado de la naranja, que consiste en el despaletizado y vaciado o volcado en agua de la naranja y se monta la fruta en un sistema de transporte por el que pasará por los siguientes procesos.

Todavía en el tratamiento previo, se hace una primera inspección de la fruta en la que se elimina manualmente los productos no comerciales (mal estado o calibre no comercial). Tras esta primera selección se pasa a la fase de lavado.

La fase de lavado es necesaria por la normativa de calidad y los protocolos de seguridad de los clientes por los que obligan a comercializar los frutos limpios. El lavado elimina polvo, suciedad, insectos, esporas, pesticidas y otros contaminantes de la superficie de los frutos. Se aplica el detergente directamente con unas boquillas sobre el fruto y este se retira con agua que puede contener algún desinfectante que no deje residuo en la fruta. Como alternativa más ecológica están las lavadoras de fruta. Utilizando detergentes ecológicos, boquillas de menor caudal y recirculando parte del agua consiguen resultados similares con un menor impacto ambiental. Además, el agua procedente de este proceso es tratada con ozono para eliminar los residuos fitosanitarios.

En la siguiente fase se prepara el producto para el consumidor. En las líneas de manipulación es habitual la aplicación de fungicidas con el objetivo de que la fruta aguante en buen estado el

mayor tiempo posible. También se busca presentar el producto con el mejor aspecto posible, ya que el consumidor lo asocia a un producto de buena calidad. Para ello se aplica una fina capa de cera que además reduce la perdida de líquido de la fruta y alarga el tiempo de vida comercial en la que además puede contener el fungicida y eliminar el paso anterior. Tras la aplicación del fungicida y/o la cera se pasa por túneles de secado que queman gas para que los productos aplicados se fijen a la superficie de la fruta.

La última fase de la línea consiste en la selección manual de la fruta desechando cualquiera que tenga algún tipo de desperfecto. Es necesaria una buena iluminación y zonas de selección delimitadas. Como todo proceso manual, resulta lento e ineficiente, y puede ser un cuello de botella en la línea de manipulación. Es por eso que ya se están incluyendo procesos de selección ópticos como el desarrollado en este proyecto para la automatización de esta etapa. También se derivan algunas piezas al análisis de sus características como el estado de madurez y acidez de la pieza de fruta. Después, en el calibrado unas máquinas clasifican y conducen la fruta por su tamaño y en ocasiones color hacia la sección final de envasado, que prepara al producto para su comercialización.

1.3.2. Visión artificial en la selección:

Con el fin de sustituir el proceso de selección manual de la fruta, en las líneas de manipulación de alimentos se empiezan a implementar sistemas de visión artificial que son capaces de reconocer la pieza de fruta y sus características para clasificarla.

La introducción de sistemas de selección ópticos en las líneas de manipulación de cítricos comenzó con programas que utilizando librerías de procesamiento de imágenes como *OpenCV* y con unos parámetros impuestos son capaces de hacer las labores de detección de defectos y clasificación. El procedimiento que siguen es hacer una segmentación de la imagen separando los píxeles defectuosos y otras características como el tamaño. A partir de las características extraídas y sus etiquetas se realiza la clasificación mediante clasificadores que están incluidos en la propia librería de *OpenCV*. Sin embargo, el tener que especificar todas y cada una de las características que pueden tener las piezas de fruta y los rangos en los que tienen que estar estas mismas hacen que sea una labor muy ineficiente y, como veremos en los resultados, ineficaz.

Los sistemas de selección que hacen uso de las redes neuronales convolucionales son entrenados con un amplio conjunto o *"dataset"* de imágenes de piezas de fruta son capaces que extraer características como el tamaño, color, formas, defectos de la fruta y otras muchas más. Esta tecnología podría ser implementada tanto al principio de la línea, cuando se hace una revisión de la fruta apta o no apta a la línea, como en la selección final y calibración de la fruta. De esta forma se evitan los procedimientos manuales, se reducen los tiempos de paso del producto por esta etapa y se evitan instrumentos mecánicos de medición.

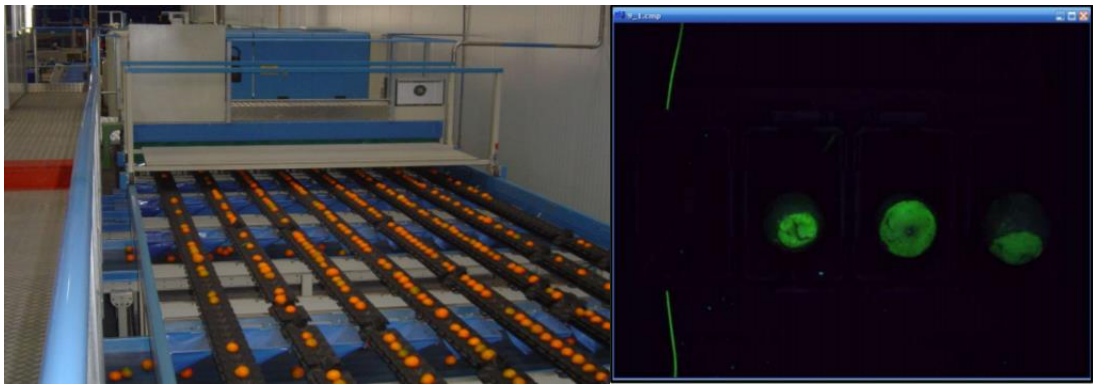


Figura 15. Sistema óptico de selección en una línea de manipulación de cítricos

Los elementos necesarios para la introducción de esta tecnología serían un sistema de captura óptica adaptado a la producción de la línea, un ordenador que ejecutara el programa que contenga la red neuronal y un sistema de clasificación que actúe en función de la respuesta de la red.

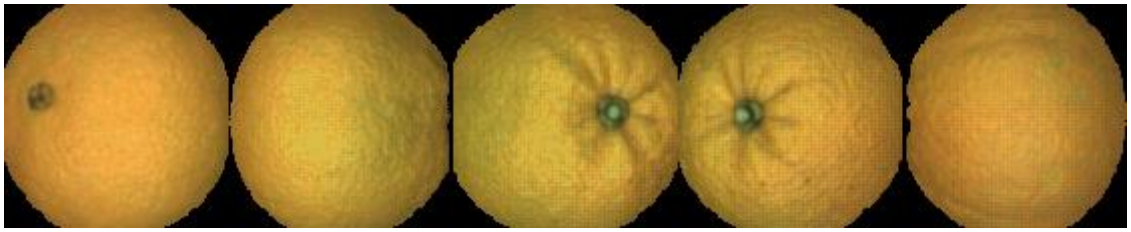


Figura 16. Muestra del "dataset" de naranjas Navel

Las tareas a realizar por el sistema de selección óptico son la eliminación de piezas de fruta no aptas para la línea que por su estado podrían comprometer los requisitos se seguridad alimentaria de la línea, etiquetar la fruta que ya ha sido encerada, secada y lista para su comercialización, y dar las instrucciones a un sistema de clasificación mecánico en función de la etiqueta asignada.

Capítulo 2: Objetivos

En este apartado se fijarán los objetivos a conseguir tanto al final del proyecto como los objetivos intermedios que servirán de puntos de control durante el desarrollo del mismo. También se fijarán una serie de criterios para evaluar la validez del algoritmo.

2.1. Objetivo general:

El objetivo principal de este proyecto es el diseño, implementación y evaluación de un sistema óptico de clasificación en una línea de manipulación de cítricos basado en visión artificial, apoyándose en algoritmos de redes neuronales convolucionales.

2.2. Objetivos específicos:

Para el correcto desarrollo de este trabajo será necesario seguir una metodología que alcanzar las siguientes metas:

- Elaboración de un "*dataset*" de imágenes de naranjas Navel para el entrenamiento de la red neuronal.
    - Toma de imágenes de cada pieza de fruta de forma que se haga un visionado total de la superficie del cítrico.
    - Creación de un criterio de etiquetado
    - Asignación de etiquetas a la muestra
- Diseño de un programa que ejecute un algoritmo de entrenamiento de la red neuronal convolucional escogida y a su vez, con una muestra del *"dataset"* no utilizado para el entrenamiento, cuantifique la eficacia del propio algoritmo.
    - Elegir una red neuronal convolucional preentrenada y adaptarla a la tarea a realizar: la clasificación final de las naranjas por categorías.
    - Encontrar los parámetros óptimos para la tarea de clasificación.
- Valoración de los resultados obtenidos tras el entrenamiento
    - Se aplicarán los criterios de evaluación de los resultados.
- Valoración económica del proyecto.

2.3. Criterios de valoración de resultados:

Para poder hacer una valoración objetiva de los resultados se comprobará el cumplimiento de algunos criterios que deben adecuarse las condiciones descritas en los objetivos específicos formulados anteriormente:

- Coste temporal:
    - El tiempo de duración de los entrenamientos del algoritmo afectará directamente al coste de económico de desarrollo del proyecto.
    - Para llegar al objetivo de igualar el ritmo de producción tradicional es necesario saber el tiempo que tarda el algoritmo en hacer la predicción.
- Precisión de la predicción:
    - La precisión del algoritmo marcará su validez para la tarea de clasificación. Se estudiará tanto la precisión con la muestra de entrenamiento como la de validación y se sacarán conclusiones en función del resultado final.
    - Se calculará la precisión asociada a cada etiqueta.

- Comparación con otras tecnologías:
    - Comparación del rendimiento entre OpenCV

Capítulo 3: Métodos

En este apartado se hablará de la metodología que se ha seguido para el desarrollo del proyecto. Se hablarán del proceso de etiquetado y las trasformaciones que han sufrido las imágenes del "dataset". También se hablará de la red neuronal implementada y cómo se ha adaptado el entrenamiento a los diferentes "datasets" formados. Finalmente, se introducirá el framework Pytorch, las diferentes herramientas que contiene y el entorno de ejecución del programa.

3.1. "Dataset" de imágenes:

Para crear el *"dataset"* con el que desarrollaremos nuestro algoritmo se realizará una ráfaga de 5 fotografías de la misma pieza de fruta de diferentes partes de la superficie para identificar cual es el nivel de desperfectos que tiene la naranja. La fruta irá sobre unos raíles de rodillos y con cierta separación con otras piezas de fruta. Será necesario que el dispositivo de captura de imagen tenga un amplio ancho campo sobre el raíl y, aprovechando la forma esférica de la naranja, la pieza de fruta avanzará rodando sobre los rodillos y será posible tomar imágenes de distintas partes de la superficie utilizando una cámara fija. En nuestro caso, además de una imagen a color, se realizará la captura del espectro infrarrojo de la pieza de fruta por lo que cada captura se hará con dos cámaras diferentes.

La razón por la que este "*dataset*" contiene imágenes en el espectro infrarrojo es que las zonas podridas de la fruta son más visibles en este espectro que a color, por lo que nuestro algoritmo a parte de poder detectar defectos visibles en la piel también podrá detectar la fruta en mal estado y retirarla.

La resolución de cada captura es de 112x112 píxeles, pero para asociar todas las capturas de la ráfaga a la misma etiqueta se creará un elemento de resolución 112x560 con las 5 imágenes de la misma pieza de fruta concatenadas horizontalmente.

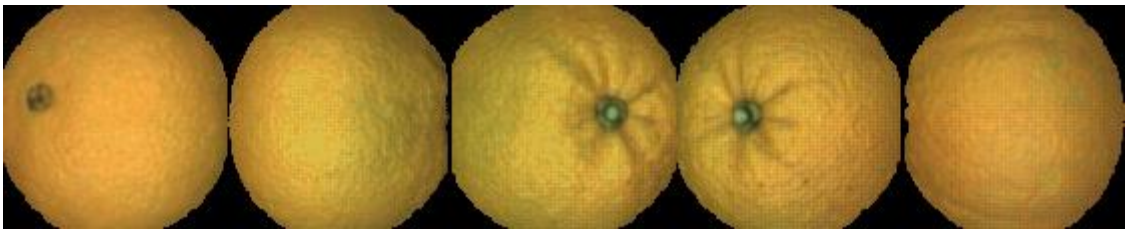


Figura 17. Muestra de imagen RGB del "dataset" de naranjas Navel

El *"dataset"* con el que se ha trabajado cuenta con las imágenes de 10.745 naranjas tanto a color como en el espectro infrarrojo. Por lo que finalmente se trabajará con un *"dataset"* de 21.490 imágenes.

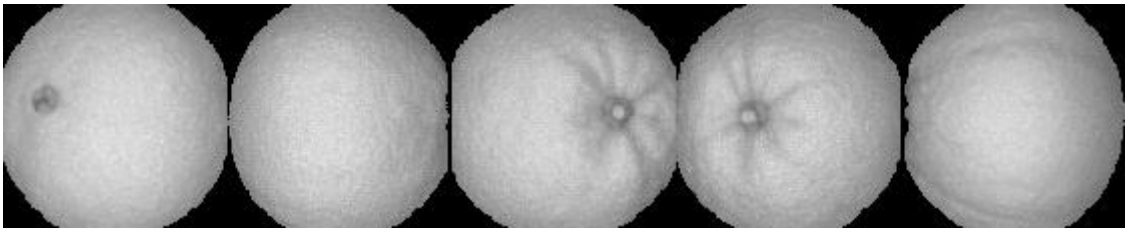


Figura 18. Muestra de imagen RGB del "dataset" de naranjas Navel

3.1.1. Etiquetado de las imágenes:

El etiquetado del "*dataset*" ha sido un proceso manual llevado a cabo por varios etiquetadores. El primer paso para el etiquetado es la creación de un criterio de etiquetado. En este caso la valoración de la calidad de la fruta ha seguido los estándares internacionales para la fruta y verdura (Anexo). Una vez escogido un criterio a seguir, se hace un etiquetado individual y después se hace una puesta en común para corregir las discrepancias que puedan ocasionarse. Este es un procedimiento iterativo que tiene como objetivo crear un criterio sólido de etiquetado tratando de eliminar la subjetividad del etiquetador.

La clasificación de la naranja Navel se realizará en categorías de la 1 a la 5. Se ordenarán por calidad de la pieza en la que la mejor calidad será la 1 y la peor la 5. Se juzgará el tamaño, forma y desperfectos en la superficie de la misma.

En el *"dataset"* con el que se trabajará la clase 1 es el tipo de naranja que encontramos en los supermercados. Esta tiene un color muy vivo, una forma totalmente redonda y con apenas daños en la superficie.

Por el contrario, la categoría 5 presenta numerosos defectos. En esta categoría se encuentran los productos que no tienen un estado de maduración adecuado y presentan grandes zonas verdes en lugar del naranja característico de la fruta. También encontramos en esta categoría la fruta que tiene grandes deformidades que ha sufrido durante su crecimiento en el campo y aquellas que en el proceso de recogida o durante su paso por la línea de manipulación de cítricos han sufrido daños muy visibles en la superficie de su piel.

El criterio escogido para el etiquetado de las clases intermedias es más flexible. En ellas vemos ejemplos de piezas de fruta que tienen algunos desperfectos visibles que en función de su presencia y gravedad se etiquetan en una mayor o menor categoría. Son categorías muy susceptibles a la subjetividad de quien las etiqueta y, como ya veremos más adelante, serán las más difíciles de clasificar.

3.1.2. Preprocesamiento de las imágenes:

Anteriormente se ha comentado que los elementos que compones nuestro *"dataset"* son 5 imágenes concatenadas horizontalmente de la misma pieza de fruta con una resolución de 112x112 píxeles de cada imagen y una resolución final del elemento 112x560 píxeles.

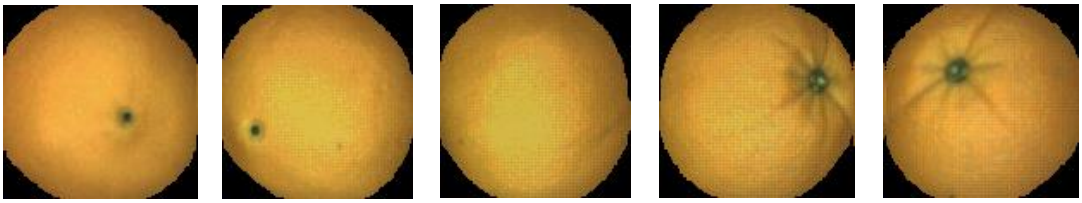


Figura 19. Capturas de naranjas Navel para el "dataset "

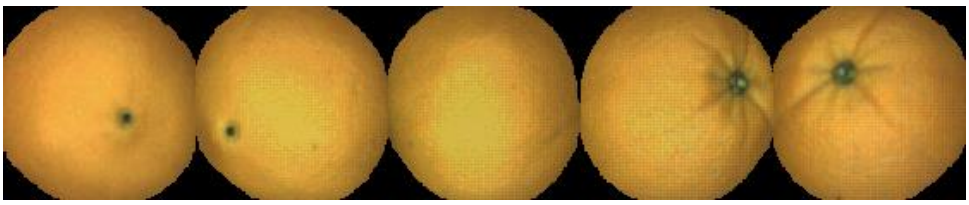


Figura 20. Concatenación de capturas para crear el elemento de entrenamiento

Durante los diferentes tipos de entrenamientos se han realizado ensayos de prueba y error para cuáles eran las imágenes más adecuadas para el entrenamiento de la red neuronal convolucional. Las pruebas han sido: red entrenada solo por las imágenes a color, solo entrenada por imágenes en espectro infrarrojo, entrenada por ambos tipos de imágenes por separado y finalmente con las imágenes concatenadas verticalmente y horizontalmente.

Las imágenes a color están almacenadas en formato "*ppm*" mientras que las imágenes en el espectro infrarrojo se encuentran en formato "*pgm*". Para poder trabajar con los dos tipos de imágenes a la vez, primero, será necesaria la conversión de las imágenes a un formato común. Con el fin de no perder la calidad del *"dataset"* original se transformarán al formato "*bmp*" (*Windows bitmap*) que tiene la capacidad de guardar imágenes de 24 bits (16.7 millones de colores) y 8 bits (256 tonos de gris). Este objetivo se ve alcanzado al no ver modificado el peso y calidad de las imágenes en diferentes formatos. Una vez ya contamos con el *"dataset"* con el mismo formato empiezan los diferentes entrenamientos.

Sin entrar en detalles de la precisión de las predicciones, el primer entrenamiento se realizó con las imágenes RGB (*Red Green and Blue*) del *"dataset"* consiguiendo un resultado que cumpliría muchos requisitos. En la prueba con las imágenes infrarrojas se ha visto un descenso notable de la precisión y un tiempo de ejecución del entrenamiento ligeramente menor pero que no compensa el bajo rendimiento. Después, buscando un mejor rendimiento de la red utilizando los dos tipos de imágenes. Se plantea la hipótesis de que la red está aprendiendo de ambos tipos de fotografías, por lo que es posible que al entrenar con las dos juntas la red pueda aprender características de un tipo de imagen que no tenga el otro. Por esa razón se plantea la creación de un nuevo elemento de entrenamiento Una única entrada de datos que contenga la información a color e infrarrojo de la imagen de la naranja. Esto se implementará concatenando verticalmente y horizontalmente las imágenes de la misma pieza de fruta asociando las dos imágenes a la misma etiqueta, como cuando se ha concatenado la ráfaga de 5 imágenes de la misma pieza de fruta. El elemento utilizado para el aprendizaje en estos casos tendrá una resolución final de 224x560 píxeles y 112x1120 píxeles, respectivamente.

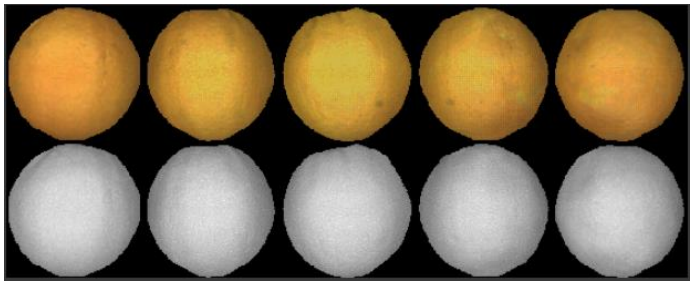


Figura 21. Concatenación vertical de imágenes RGB e IR del "dataset"

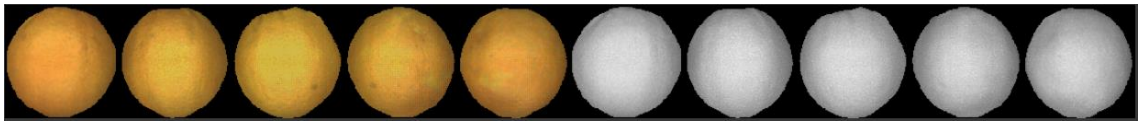


Figura 22. Concatenación horizontal de imágenes RGB e IR del "dataset"

3.2. Red neuronal convolucional ResNet:

El objetivo de este trabajo es diseñar e implementar un sistema de clasificación asistido por el algoritmo de una red neuronal convolucionalFalse. En lugar de diseñar la red desde cero se lleva a cabo una técnica conocida como "*transfer learning*".

Actualmente, se tiene a disposición de todo el mundo un gran abanico de redes neuronales que son de libre utilización. La técnica del "*transfer learning"* permite adaptar los parámetros de una red neuronal preentrenada para adaptarla a un *"dataset"* específico. La creación de una red neuronal convolucional no es un proceso trivial. No todas las redes son iguales, existen diferentes arquitecturas que pueden variar su rendimiento dependiendo del *"dataset"* con el que trabajemos.

Las primeras redes se dieron a conocer en el reto ILSVRC (*ImageNet Large Scale Visual Recognition Challenge*). Un reto que consistía en diseñar un algoritmo que fuera capaz de clasificar el *"dataset" ImageNet*, que cuenta con más de 1,2 millones de imágenes de 1000 clases diferentes. En 2010 se empezaron a ver grandes avances en la precisión de los algoritmos. Los ganadores de ese año consiguieron clasificar el *"dataset"* acercándose al 25% de error. Sin embargo, fue en 2012 cuando la aparición de las redes neuronales convolucionales, como las que conocemos hoy en día, entraron en escena. La primera red que se dio a conocer fue la red *AlexNet*, creada por Alex Krizhevsky en colaboración con Ilya Sutskever y Geoffrey Hinton, y fue la ganadora del reto en 2012 bajando de un 26% a un 16,4% la tasa de error en la clasificación.

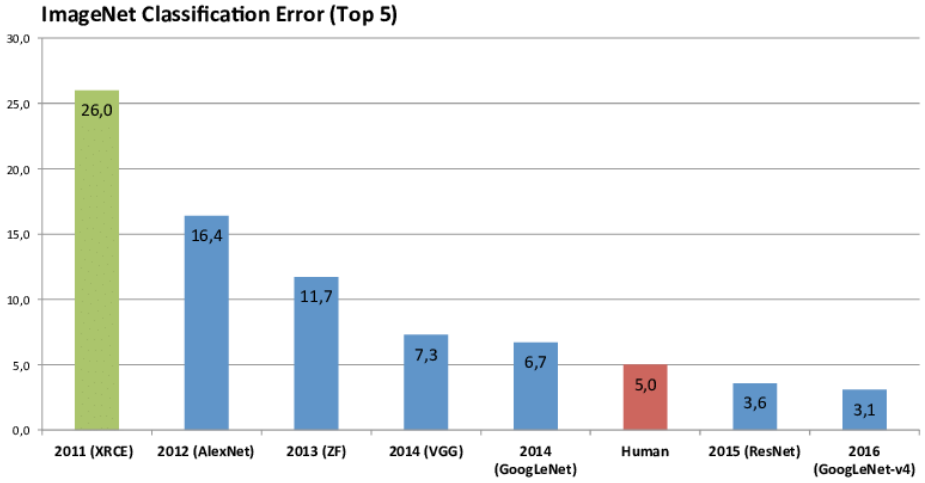


Figura 23. Resultados del ganador del reto ILSVRC a lo largo del tiempo

AlexNet era una primera red neuronal que contaba con un input de 227x227, 5 capas de convolución, muestreo con *Max-Pooling*, 3 capas de neuronas totalmente conectadas (*fully connected*) y la función *ReLu* como función de activación. A partir de la presentación de *AlexNet* se popularizó el uso de este tipo de redes y no tardaron en aparecer modelos que conseguían mejorar el rendimiento de *AlexNet*. Las más conocidas fueron las ganadoras del ILSVRC del año 2014 primero *VGG* y después *GoogleLeNet* que superaron las entonces 8 capas de la red de *AlexNet* hasta las 19 y 22 capas respectivamente. Además, estas redes se diferencias de la de *AlexNet* en que deja de lado la rigidez y el diseño manual de la red, puesto que sobre las nuevas arquitecturas existen una serie de hiperparámetros que permiten ajustar nuestro modelo.

Aparentemente una red neuronal con un mayor número de capas puede aprender un a mayor cantidad de características del *"dataset"* con el que se entrena y con ello conseguir una menor tasa de error. Sin embargo, los ensayos no dicen lo mismo. La práctica nos dice que al aumentar el número de capas de un algoritmo el modelo se vuelve más difícil de entrenar. Esto sucede porque al añadir capas el gradiente del algoritmo de *backpropagation* explicado anteriormente se vuelve más pequeño. La solución que se dio a este problema fue la normalización intermedia del gradiente. Esto solucionó la dificultad de aprendizaje de las redes, pero mostró se segundo gran problema al añadir capas a nuestra red: la degradación de la red. Este fenómeno que los resultados del entrenamiento sean peores cuanto mayor es el número de capas, independientemente de sus

características o el *"dataset"*. Lo que nos dice nuestro problema es que las nuevas capas están deshaciendo el entrenamiento de las capas anteriores. La solución a este inconveniente fue el aprendizaje residual, "*residual learning"*. El aprendizaje residual consiste en crear una arquitectura en la que una red que tenga un buen comportamiento al añadirle capas no aumente la tasa de error. La clave de esta arquitectura es la creación de derivaciones ("*bypasses"*) entre las diferentes capas convolucionales que, en el caso de que las características aportadas por una capa convolucional den un mayor error que en las capas anteriores, saltará la capa llegando a la siguiente capa los mapas de características aportados en las capas con mejores resultados.

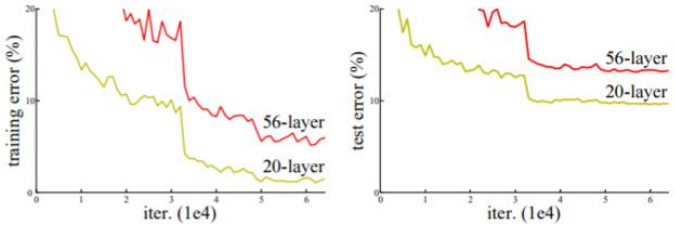


Figura 24. Comparativa entre redes neuronales convolucionales planas con diferente número de capas

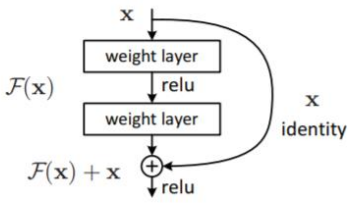


Figura 25. Esquema "Residual Learning"

Esta es la estrategia que siguió el equipo de *Microsoft Research*, liderado por Kaiming He, para crear la red *ResNet* (*Residual Net*) que consiguió solucionar los problemas que se habían encontrado todos los desarrolladores de redes neuronales convolucionales hasta el momento rompiendo la barrera del número de capas presentando una red de 156 capas y superando por primera vez la tasa de acierto en el *"dataset"* de *ImageNet* de un ser humano estimada en el 5%.

Para este proyecto se hará uso de la red ResNet18, la red de arquitectura *ResNet* de 18 capas.

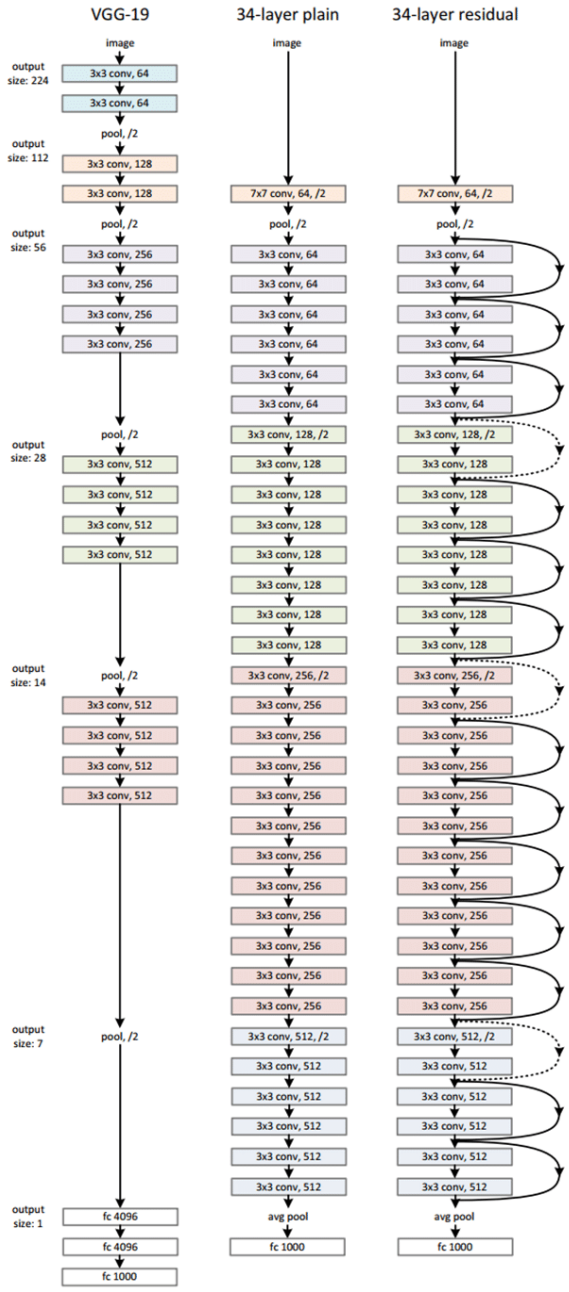


Figura 26. Redes neuronales condiferentes arquitecturas

3.3. Ajuste de hiperparámetros:

El ajuste de hiperparámetros forma parte del proceso de entrenamiento de la red. El ajuste suele ser un proceso manual en el que por ensayos de acierto y error podemos modificar el aprendizaje de la red.

3.3.1. Número de épocas:

Uno de los parámetros más intuitivos que tendremos que definir es el que nos marca el número de veces que se va a hacer el volcado de imágenes del *"dataset"* de entrenamiento sobre la red. Este parámetro recibe el nombre de número de épocas o "*epochs*". Este parámetro afecta muy directamente en el coste temporal del entrenamiento. En la práctica podemos ver que durante el entrenamiento los valores de precisión y de la función de coste se estabilizan, por lo que un mayor

número de iteraciones no significa que se produzcan mejores resultados. Es importante no sobreentrenar las redes neuronales.

3.3.2. Tamaño del lote:

En apartados anteriores se ha visto cómo las redes hacen uso del descenso del gradiente para variar los pesos de las neuronas, convolucionales y las del perceptrón, y así modificar las salidas de la red en función a nuestro *"dataset"*. Hacer el volcado de imágenes y calcular el gradiente de todas ellas en cada iteración consume una cantidad de cómputo absurdamente grande para el beneficio que podemos sacar de ello. La práctica habitual es el tomar un lote más o menos grande de imágenes para hacer estos cálculos. Esta técnica tiene el nombre de *mini-batch gradient descent*. Se le da el nombre de "mini-batch" refiriéndose a que la muestra es menor que el *"dataset"* completo a diferencia del *batch gradient descent*.

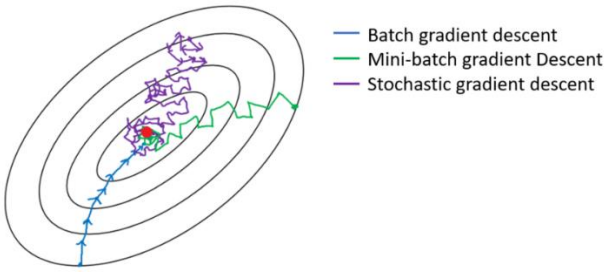


Figura 27. diferencias entre "batch", "mini-batch" y "stochastic" gradient descent

3.3.3. Optimización:

En cada iteración se hace el cálculo del gradiente que marcará los cambios en los pesos de la red, pero la personalización de este hiperparámetro no acaba aquí. Tenemos una serie de parámetros que pueden marcar cómo va a afectar el gradiente a la variación de pesos.

El primer parámetro es el ratio de aprendizaje, "*learning rate"*. Este parámetro se puede resumir en la ponderación del gradiente sobre los pesos. Un "*learning rate"* muy grande puede no llegar nunca al mínimo de la función de pérdida. Mientras, un "*learning rate"* demasiado bajo tiene un aprendizaje muy lento y es susceptible de tomar mínimos locales como mínimo global de la función.

El segundo parámetro también afecta directamente a los valores de los pesos de las neuronas, la variación del peso o "*weight-decay"*. A pesar de ajustar correctamente el "*learning rate"* puede que el gradiente en algunas ocasiones sea demasiado grande y produzca un cambio de pesos que lleve a una solución inestable del entrenamiento. El "*weight-decay"* limita cuánto puede variar los pesos de cada neurona.

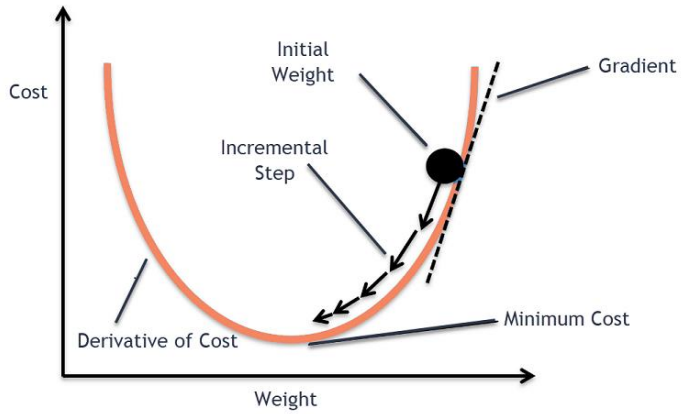


Figura 28. Optimización de pesos

Por último, hay ocasiones en las que para encontrar la mejor solución posible para el entrenamiento sin estancarse en mínimos locales (como pueden verse en la *Figura 10*) de la función de pérdida es necesario un "*learning rate"* variable durante el entrenamiento. Normalmente se desea un mayor "*learning rate"* al principio del entrenamiento para evitar los mencionados mínimos locales y un "*learning rate"* más bajo al final del entrenamiento para no saltarse el mínimo global de la función. Los parámetros son el tamaño de paso ("*step size"*), que indica cada cuantas iteraciones se va a modificar el "*learning rate"*, y el parámetro *gamma* (γ) que indica en tanto por 1 cuál va a ser la reducción del "*learning rate"*.

3.3.4. Ajuste de hiperparámetros:

A pesar de que el ajuste de hiperparámetros suele ser un proceso manual no quita que pueda llegar a ser un trabajo tedioso y que cree confusión y equivocaciones si no se hace forma muy meticulosa. Como alternativa, al igual que en este trabajo, podemos informatizar este proceso manual mediante diferentes técnicas de optimización automática de hiperparámetros como: búsqueda de cuadrícula, búsqueda aleatoria, optimización bayesiana, optimización basada en gradientes, optimización evolutiva, basado en la población y basada en parada anticipada, entre otros.

En este proyecto se van a estudiar múltiples usos del *"dataset"*, lo que conlleva que se necesiten diferentes hiperparámetros para cada caso. El uso de estas técnicas será muy beneficioso para el coste temporal del proyecto. La técnica en concreto que se va a utilizar en este proyecto será la optimización bayesiana, que crea un modelo probabilístico del mapeo de funciones desde los valores de los hiperparámetros hasta el objetivo evaluado en un conjunto de validación. La optimización bayesiana explora de forma eficiente el espacio de posibles opciones de hiperparámetros al decidir qué combinación explorar a continuación basándose en observaciones anteriores.

3.4. Ejecución:

A continuación, se nombrarán las herramientas utilizadas en este Trabajo de Fin de Grado. Para todas ellas hay alternativas, pero hemos creído que eran las más adecuadas.

Empezando por el lenguaje de programación, se ha elegido el más popular para tareas de *"Machine Learning"*, Python. Es de sobra conocido en este campo que este es un gran lenguaje para empezar y continuar tu trayectoria en *"Machine Learning"*. Es un lenguaje de muy alto nivel,

intuitivo y está apoyado por un gran número de librerías y otras herramientas, algunas de las cuales nos serán muy útiles durante el proyecto.

Lo segundo más importante es elegir el lugar dónde se va a desarrollar el programa. Lo más conveniente para el desarrollo de un programa desde cero es elegir un IDE (Entorno de Desarrollo Integrado) que nos permita ir probando pequeñas partes del código para detectar de forma más rápida y fácil los errores que pueda tener. Hay varios IDEs que permiten el desarrollo de programas en cuadernos con estas características, pero los utilizados han sido *Jupyter Lab* y *Google Colab*. Finalmente, se ha decantado por el uso de *Google Colab* por errores que han aparecido en *Jupyter* tras alguna actualización. Otras ventajas que nos da *Google Colab* es que tienes hardware potente para *"Machine Learning"* a tu disposición de forma gratuita, se puede realizar el trabajo desde cualquier tipo de dispositivo sea cual sea su gama y puedes hacer el uso de la gran parte de las librerías sin necesidad de instalarlas en tu dispositivo.

Las librerías son algo esencial en cualquier trabajo de programación. Añaden funcionalidades a la programación con un determinado lenguaje que no son nativas del mismo. Algunas de ellas son tan recurrentes que se descargan con el propio lenguaje. Aunque, para no obligar a hacer una instalación tan pesada de primeras, otras muchas librerías que son también muy recurrentes se pueden instalar desde la consola de comando del ordenador, algunas de ellas son: *Numpy* (agiliza los cálculos con matrices con sus *arrays*), *Matplotlib* (para graficar resultados) y *Pandas* (para trabajar con bases de datos en múltiples formatos). Por otro lado, están las librerías que contienen herramientas que son específicas para proyectos de *"Machine Learning"* que reciben el nombre de "*frameworks"*. Son herramientas que agilizan el diseño de los algoritmos de *"Machine Learning"*, e.g.: tienen formas específicas de cargar los *inputs* con sus propias etiquetas y tamaño de lote, contienen modelos descargables de redes neuronales preentrenadas, funciones de activación y de pérdida, y tienen funcionalidades que automatizan procesos que surgen durante el entrenamiento como el cálculo del gradiente de la red y la actualización de pesos. Las librerías más conocidas que realicen estas tareas son la librería de *TensorFlow* y la de *Pytorch*, la elegida para este proyecto.

Figura 29. Logo de Pytorch

Figura 30. Logo de Google Colab

3.4.1. Pytorch:

Pytorch es un *framework* de redes neuronales que nos facilita la tarea de diseñar, entrenar y poner en producción nuestros modelos de *Deep Learning*.

Pytorch integra muchas funcionalidades y una interfaz muy similar a la librería *Numpy*. Esto hace que el *framework* sea muy fácil de utilizar, ya que sabiendo utilizar la librería *Numpy*, que es muy común en los proyectos de Python, es bastante sencillo utilizar *Pytorch*. Algunas similitudes que tiene con la librería es el tener un tipo propio de vector con el que se puede trabajar de forma

similar, en *Numpy* es el array y en *Pytorch* el tensor, aunque guardan algunas diferencias importantes. Además, tiene funciones para pasar de un tipo al otro de forma muy rápida y sencilla.

Otra de las grandes funcionalidades que tiene *Pytorch* es el llamado *Autograd*. *Autograd* es una funcionalidad que nos proporciona la posibilidad de calcular derivadas de manera automática con respecto cualquier tensor. Esto es lo que le da a *Pytorch* un gran potencial para diseñar redes neuronales complejas y entrenarlas utilizando algoritmos de gradientes sin tener que calcular todas estas derivadas manualmente. Para poder llevar a cabo estas operaciones, *Pytorch* va construyendo de manera dinámica un grafo computacional. Cada vez que aplicamos una operación sobre uno o varios tensores se añaden al grafo computacional junto a la operación en concreto. De esta manera, si queremos calcular la derivada de cualquier valor con respecto a cualquier tensor, simplemente tenemos que aplicar el algoritmo de *backpropagation* en el grafo. Además, esta función se implementa en una sola acción llamando a la propiedad *backward* del tensor, e igual de sencillo es calcular la función de pérdida y actualizar los pesos de la red.

Por último, y probablemente más importante, *Pytorch* permite la programación con la GPU, lo que permite que un entrenamiento que puede tardar horas tarde unos minutos. El entrenamiento de redes neuronales hace uso de muchos productos matriciales, tarea increíblemente costosa para un procesador, pero trivial para una tarjeta gráfica. Las tarjetas gráficas multiprocesadores dentro de los propios ordenadores diseñados para acelerar los cálculos necesarios para renderizar una escena tridimensional en la pantalla del ordenador. Las tarjetas gráficas construyen una imagen bidimensional calculando la posición relativa de los elementos de una escena vistos desde una cámara virtual que representa nuestro punto de vista. Y, casualmente, se asemejan mucho a las operaciones que necesitamos para entrenar a nuestras redes neuronales. Es esta reducción de tiempos la que ha propiciado el desarrollo de esta tecnología y el aumento de sus aplicaciones

3.4.2. Entorno de ejecución:

Debido a las circunstancias ocasionadas por la Covid-19 este Trabajo de Final de Grado ha sido realizado íntegramente de forma remota. Tanto 1a asignatura que precede este proyecto como la confección y seguimiento del mismo ha sido a distancia. Esta situación ha hecho que los recursos para realizar el trabajo sean propios.

Al haber usado el IDE de *Google Colab* se ha evitado el uso intensivo de dispositivos propios evitando posibles problemas de mal funcionamiento y averías. La herramienta de Google ha permitido ejecutar el código en equipos alojados de forma gratuita, aunque contaba con la desventaja de tener un límite temporal de uso diario, que podía evadirse conectando cuentas diferentes al IDE y compartiendo con estas cuentas el cuaderno con el código. A pesar de este inconveniente, la estrategia utilizada disminuía mucho las molestias de tener una limitación diaria.

Otra posibilidad que nos ofrece este IDE es el no tener que descargar en dispositivos propios los recursos necesarios para la ejecución del código. Sin embargo, es necesaria la instalación y carga de datos cada vez que la sesión alojada termina.

Capítulo 4: Resultados

En este apartado se mostrarán los resultados obtenidos en la práctica y las comparaciones entre ellos utilizando los criterios descritos en el apartado de objetivos. Además, se compararán las dos tecnologías utilizadas.

4.1. "*Dataset*" RGB:

PARÁMETRO	**VALOR**
Learning rate	0,0004246581724198987
Tamaño de lote	60
Número máximo de épocas	20
Variación de pesos	0,0014226201812774092
Tamaño de paso	9
Gamma (γ)	0,6089548960972283

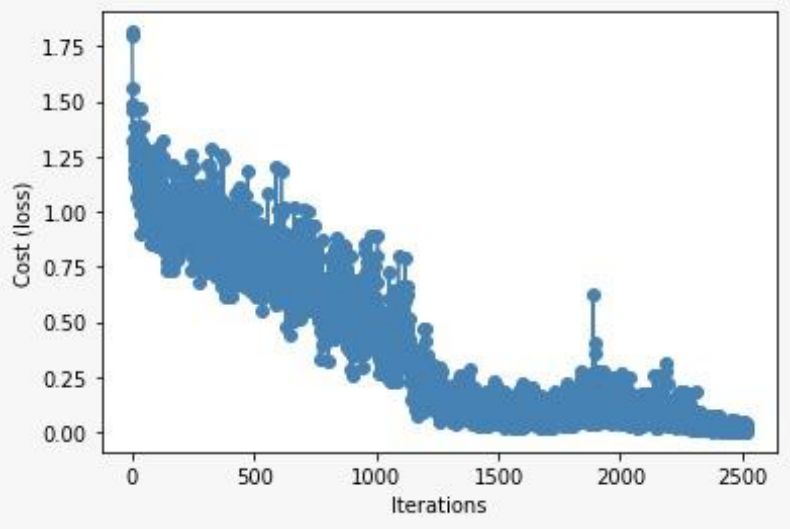
Tabla 1. Hiperparámetros de la red entrenada con el "dataset" RGB

Precisión de la red: 58,45%

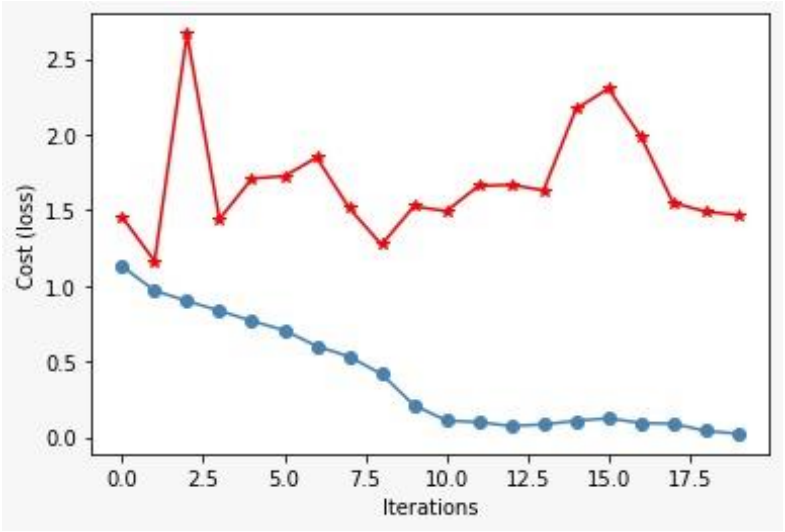
- **Clase 1:** 75,98%
- **Clase 2:** 57,47%
- **Clase 3:** 42,94%
- **Clase 4:** 48,85%
- **Clase 5:**71,08%

Coste temporal del entrenamiento:

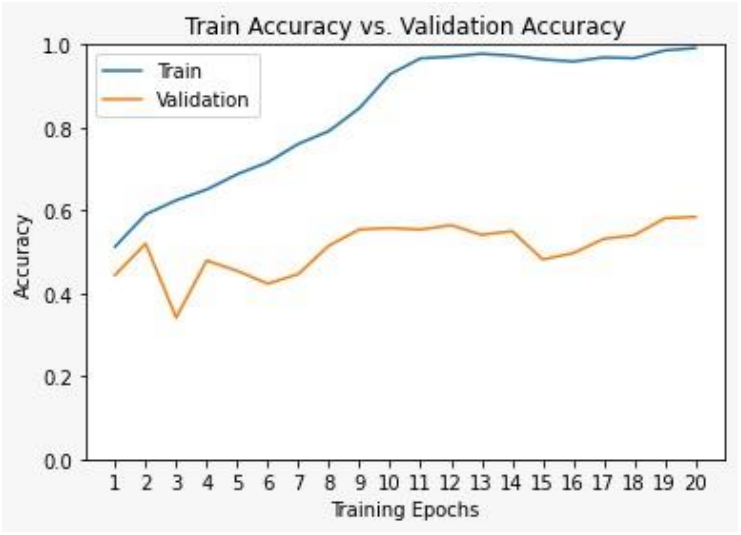
- Cálculo de hiperparámetros: 1 hora 6 minutos 52 segundos
- Entrenamiento de la red: 21 minutos 22 segundos



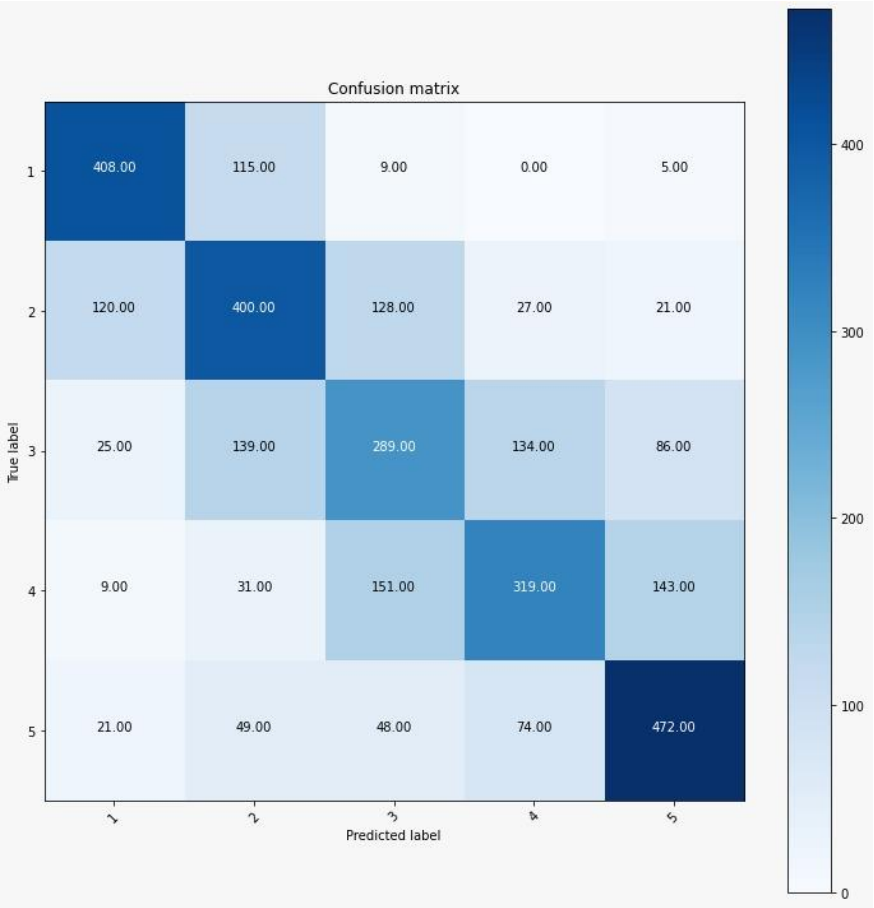
Gráfica 1. Función de coste "dataset" RGB de entrenamiento.



Gráfica 2. Función de coste "dataset" RGB de entrenamiento y validación



Gráfica 3. Precisión de la red con el "dataset" RGB de entrenamiento y validación



Gráfica 4. Matriz de confusión de la red entrenada con el "dataset" RGB

4.2. "*Dataset*" IR:

PARÁMETRO	**VALOR**
Learning rate	0,0003080745888934188
Tamaño de lote	95
Número máximo de épocas	25
Variación de pesos	0,002871317809059446
Tamaño de paso	15
Gamma (γ)	0,5199073006732008

Tabla 2. Hiperparámetros de la red entrenada con el "dataset" IR

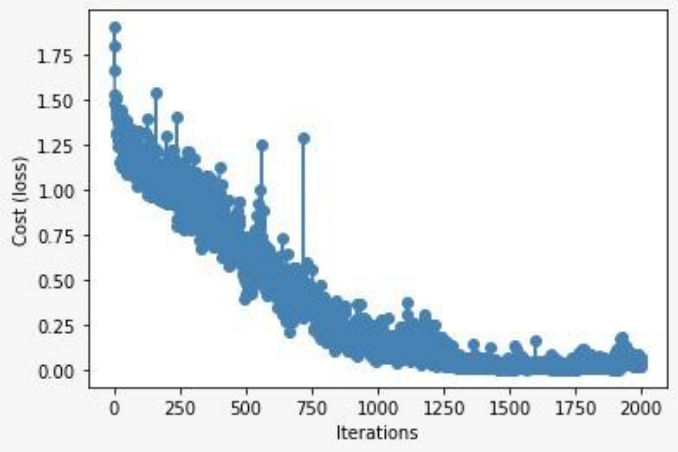
Precisión de la red: 49,33%

- **Clase 1:** 62,58%
- **Clase 2:** 44,78%
- **Clase 3:** 40,41%
- **Clase 4:** 38,81%

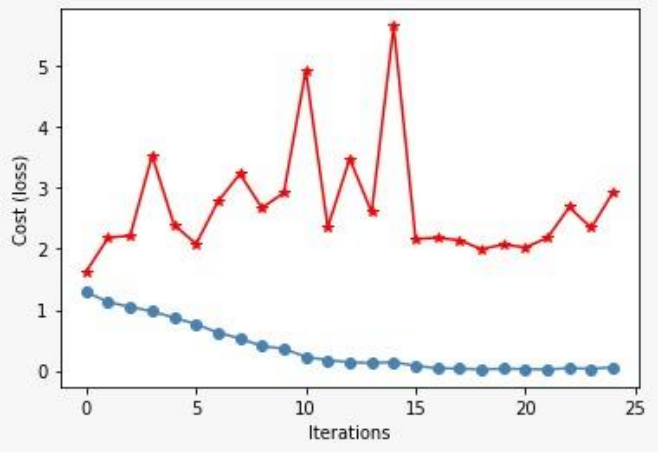
- **Clase 5:** 64,43%

Coste temporal del entrenamiento:

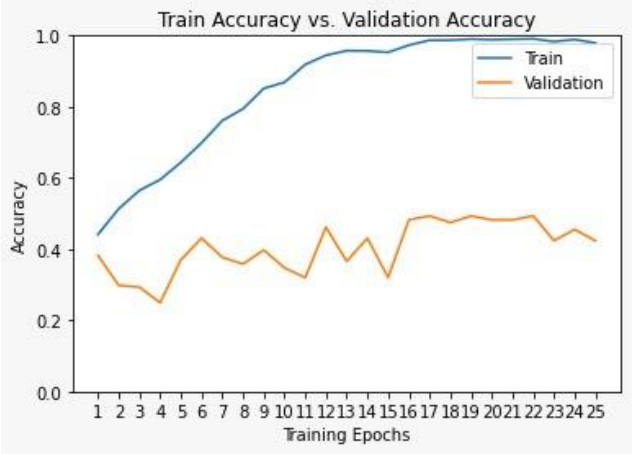
- Cálculo de hiperparámetros: 1 hora 7 minutos 30 segundos
- Entrenamiento de la red: 18 minutos 30 segundos



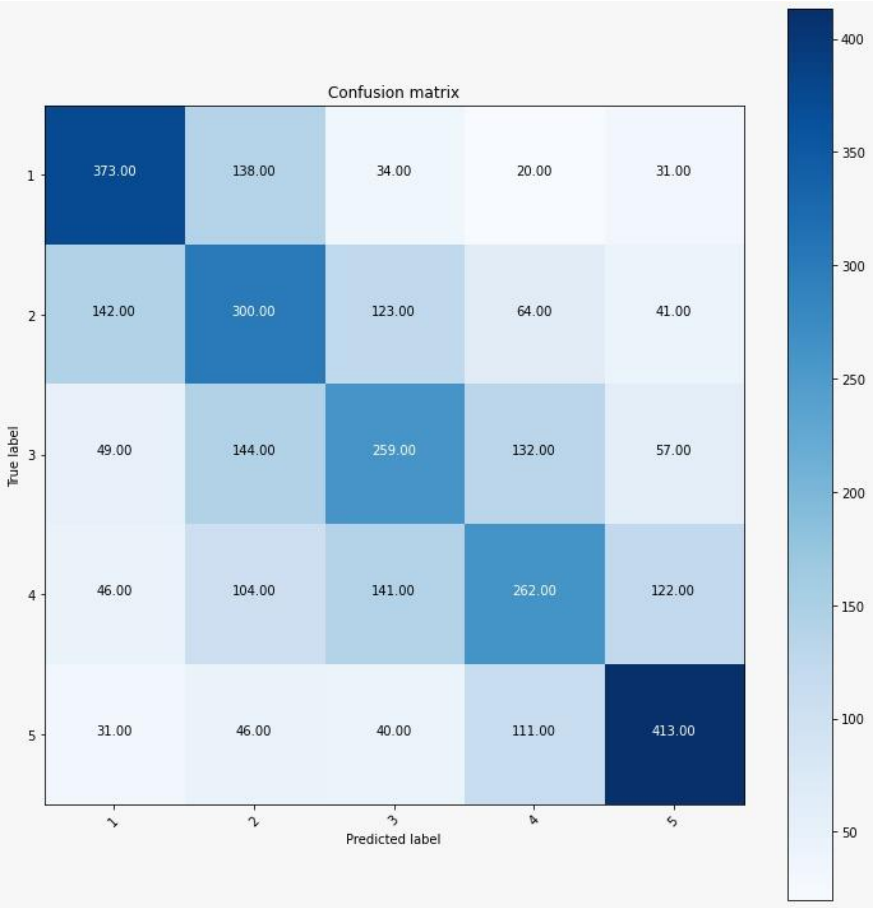
Gráfica 5. Función de coste "dataset" IR de entrenamiento



Gráfica 6. Función de coste "dataset" IR de entrenamiento y validación



Gráfica 7. Precisión de la red con el "dataset" IR de entrenamiento y validación



Gráfica 8. Matriz de confusión de la red entrenada con el "dataset" IR

4.3. "*Dataset*" con concatenación vertical RGB + IR:

PARÁMETRO	**VALOR**
Learning rate	0,00025644764415899826
Tamaño de lote	65
Número máximo de épocas	25
Variación de pesos	0,01783110159122258
Tamaño de paso	8
Gamma (γ)	0,7044377224896161

Tabla 3. Hiperparámetros de la red entrenada con el "dataset" concatenado verticalmente

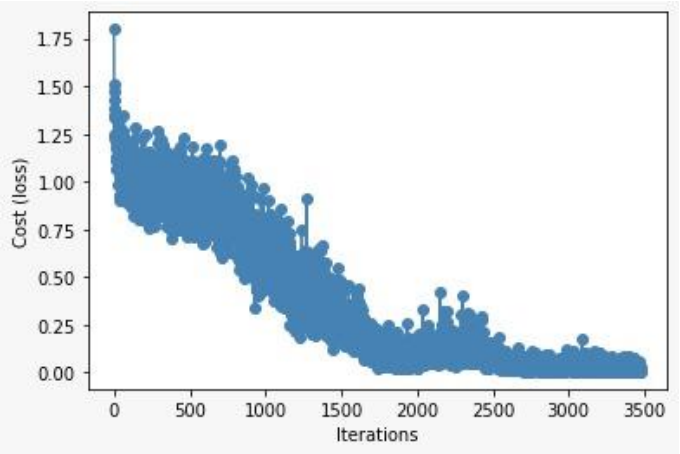
Precisión de la red: 58,11%

- **Clase 1:** 66,27%
- **Clase 2:** 58,11%
- **Clase 3:** 48,28%
- **Clase 4:** 51,74%

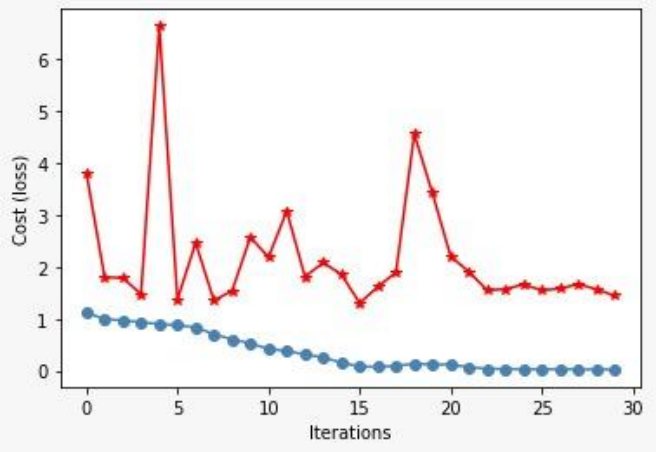
- **Clase 5:** 66,57%

Coste temporal del entrenamiento:

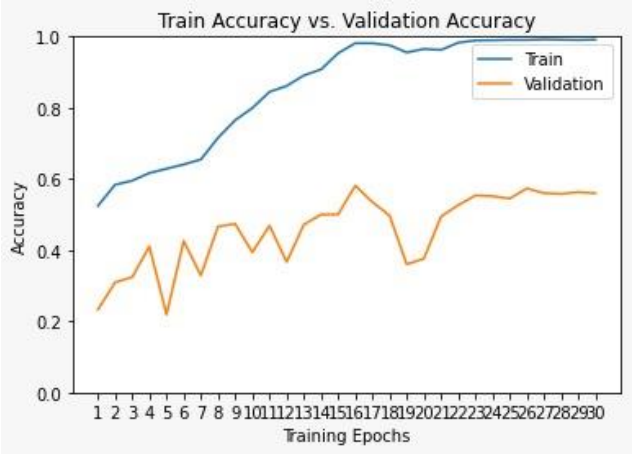
- Cálculo de hiperparámetros: 2 horas 10 minutos
- Entrenamiento de la red: 1 hora 30 minutos 18 segundos



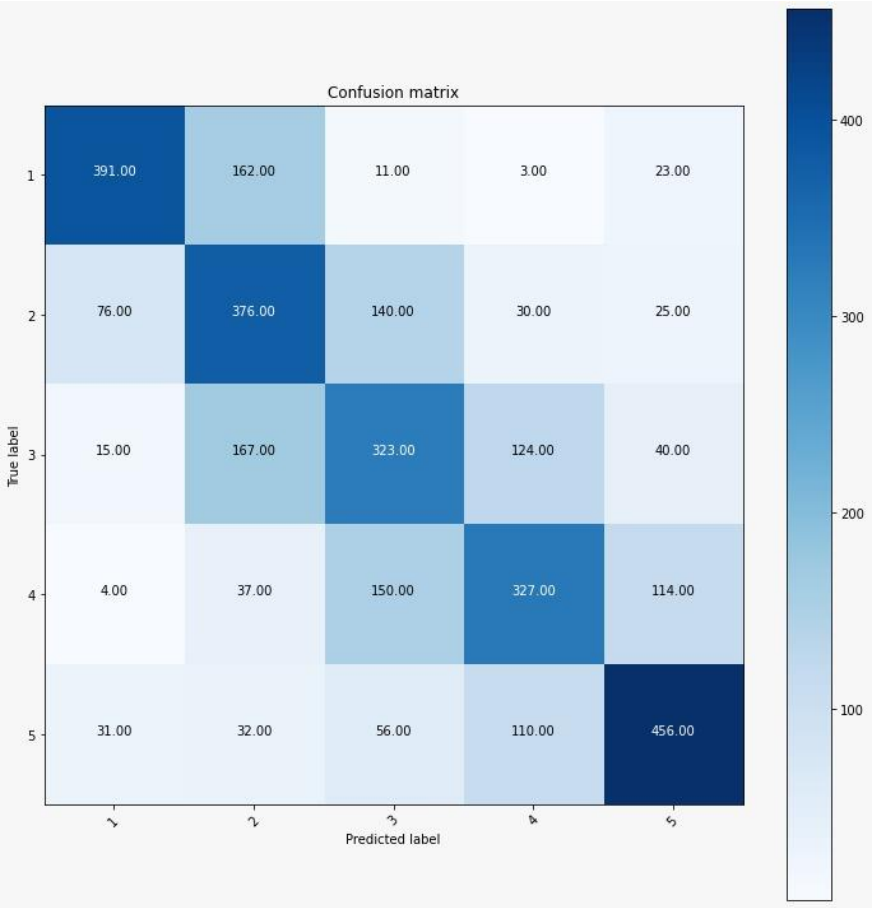
Gráfica 9. Función de coste "dataset" concatenado verticalmente de entrenamiento



Gráfica 10. Función de coste "dataset" concatenado verticalmente de entrenamiento y validación



Gráfica 11. Precisión de la red con el "dataset" concatenado verticalmente de entrenamiento y validación



Gráfica 12. Matriz de confusión de la red entrenada con el "dataset" concatenado verticalmente

4.4. "*Dataset*" con concatenación horizontal RGB + IR:

PARÁMETRO	**VALOR**
Learning rate	0,00028523911611220396
Tamaño de lote	116
Número máximo de épocas	25
Variación de pesos	0,03449915887099345
Tamaño de paso	15
Gamma (γ)	0,546921815544897

Tabla 4. Hiperparámetros de la red entrenada con el "dataset" concatenado horizontalemente

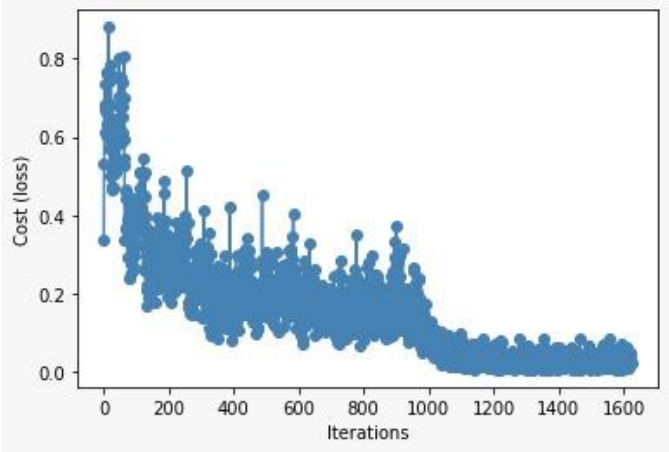
Precisión de la red: 64,16%

- **Clase 1:** 78,56%
- **Clase 2:** 71,16%
- **Clase 3:** 45,52%
- **Clase 4:** 62,46%

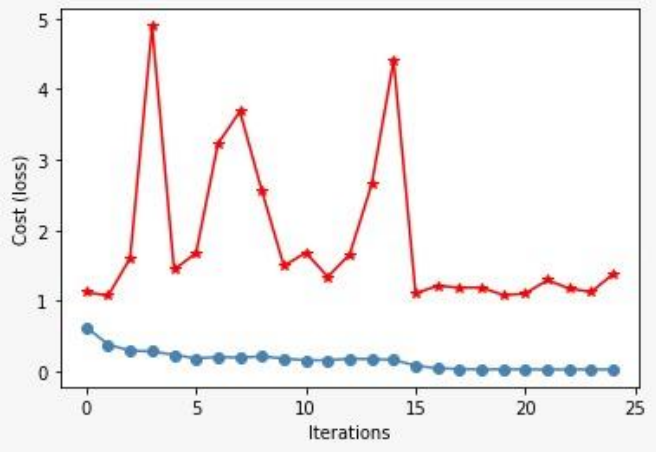
- **Clase 5:** 64,56%

Coste temporal del entrenamiento:

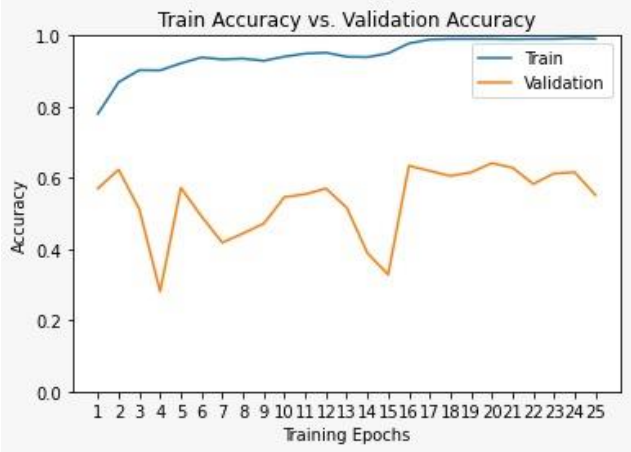
- Cálculo de hiperparámetros: 2 horas 9 minutos
- Entrenamiento de la red: 1 hora 10 minutos 22 segundos



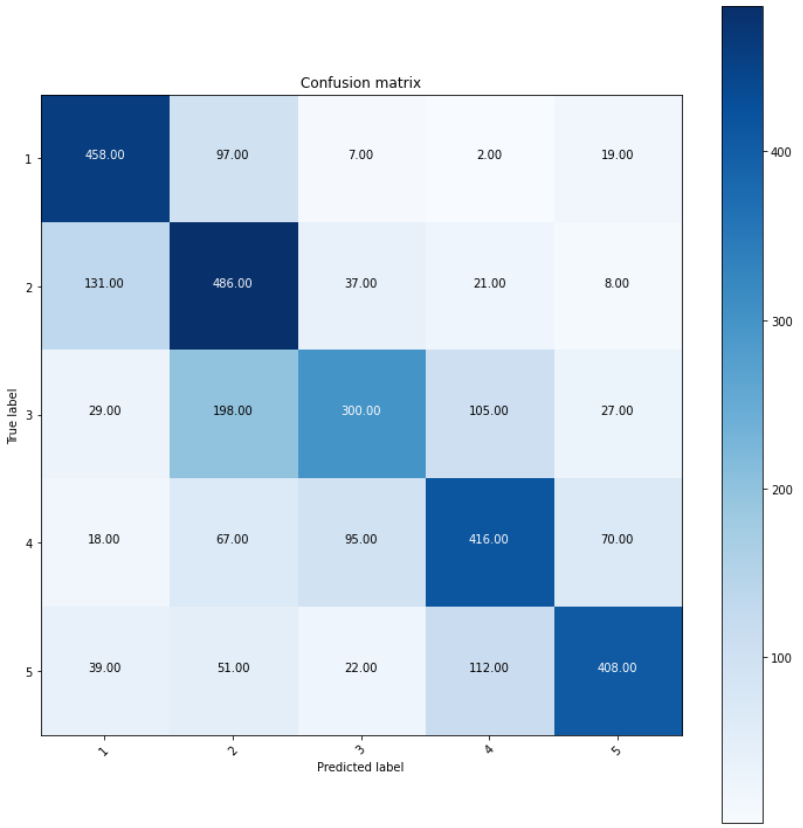
Gráfica 13. Función de coste "dataset" concatenado horizontamente de entrenamiento



Gráfica 14. Función de coste "dataset" concatenado horizontalmente de entrenamiento y validación



Gráfica 15. Precisión de la red con el "dataset" concatenado horizontalmente de entrenamiento y validación



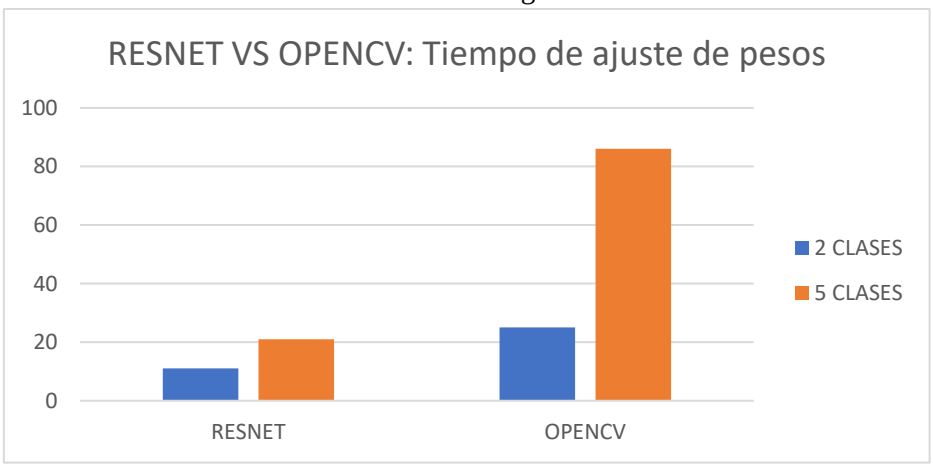
Gráfica 16. Matriz de confusión de la red entrenada con el "dataset" concatenado horizontalmente

4.5. OPENCV vs ResNet18:

Para poder comparar los dos tipos de tecnología se ha realizado la clasificación de 2 categorías (1 y 5) con el "*dataset*" RGB y después el de 5 categorías con ambas. Solo se compararán los tiempos de entrenamiento, puesto que el de ajuste de hiperparámetros tardó días en el caso de OpenCV al ser manual, y la precisión de los algoritmos.

4.5.1. Coste temporal:

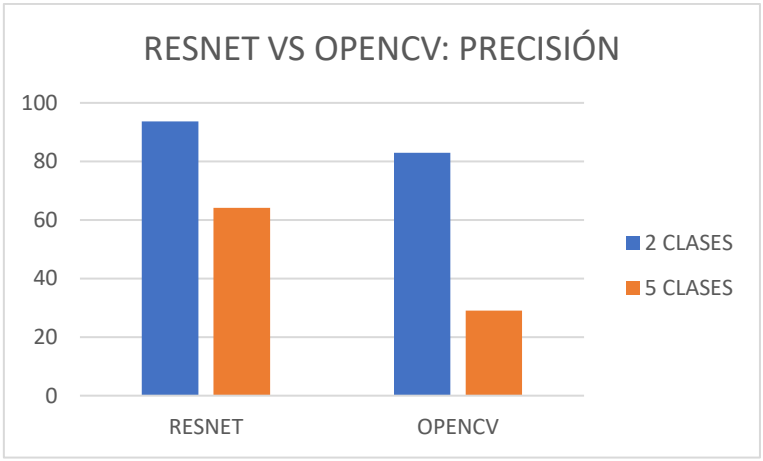
- **"*Dataset*" con 2 clases:** 39 minutos
- **"*Dataset*" con 5 clases:** 1 hora 26 minutos 36 segundos



Gráfica 17. Comparación temporal ResNet VS OpenCV

4.5.2. Precisión:

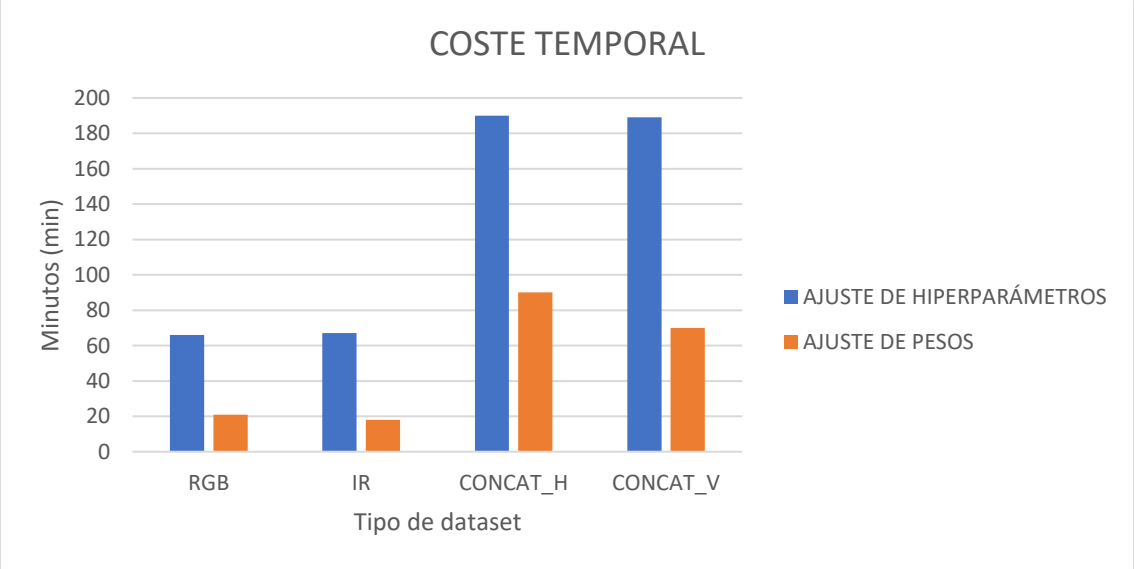
- **Tasa de acierto OpenCV (2 clases):** 83%
- **Tasa de acierto OpenCV (5 clases):** 29,04%
- **Tasa de acierto ResNet18 (2 clases):** 93,67%
- **Tasa de acierto ResNet18 (5 clases):** 64,16%



Gráfica 18. Comparación precisión ResNet VS OpenCV

4.6. Análisis de resultados:

4.6.1. Coste temporal:



Gráfica 19. Comparación temporal entre "datasets" de entrenamiento

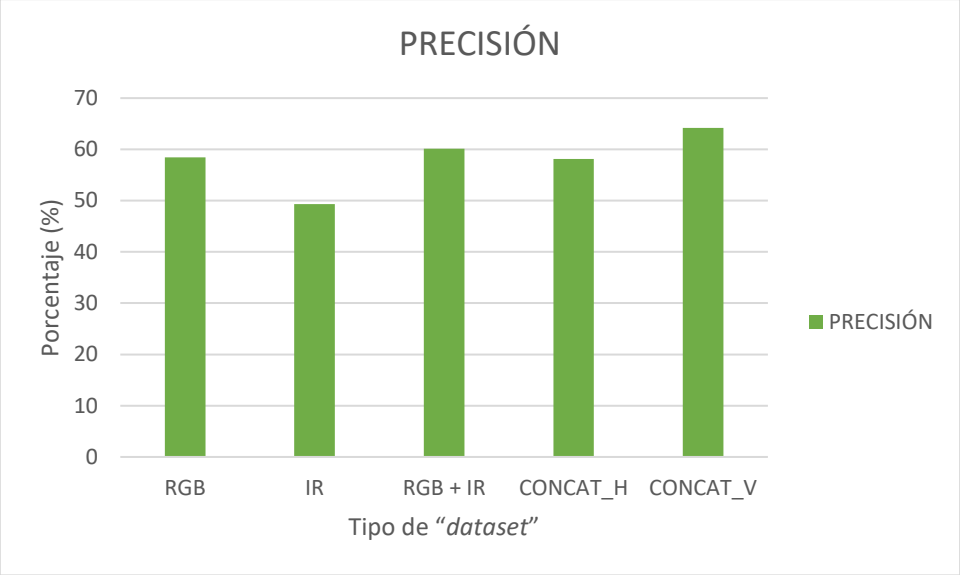
El coste temporal del algoritmo junta el tiempo invertido en el ajuste de hiperparámetros y el tiempo que el algoritmo tarda en estar entrenado. En el caso del ajuste de hiperparámetros, si hubiera sido un proceso manual, como es habitual, habría necesitado de múltiples entrenamientos para llegar a la solución óptima. En nuestro caso, el entrenamiento demandaba el ajuste de 6 hiperparámetros por lo que conseguir los parámetros más adecuados en el tiempo equivalente a 3 entrenamientos es algo impensable si se hiciera manualmente.

El tiempo que una red es entrenada es el tiempo que tarda el algoritmo en hacer múltiples volcados del "*dataset*", calcular la función de coste y su gradiente, y variar los pesos de la red. Este es un

proceso muy costoso en cuanto a potencia de cómputo. Sin embargo, el modelo entrenado que solo tiene que hacer el volcado de imágenes es capaz de etiquetar miles de ellas en tan solo unos segundos. De esta forma el proceso de selección y clasificación dejaría de ser un cuello de botella en la línea de producción.

Si comparamos los costes temporales entre la red neuronal y OpenCV se puede ver que la red neuronal es una tecnología más rápida. La principal razón de que las redes sean más rápidas es la ya comentada funcionalidad de Pytorch que permite entrenar las redes en la GPU.

4.6.2. Precisión:



Gráfica 20. comparación precisión entre "datasets" de validación

A pesar de realizar un gran número de entrenamientos no se ha conseguido grandes tasas de acierto para ninguno de los casos, alcanzando una tasa de acierto máxima del 64,16%. Tal y como muestran las matrices de confusión, la red sí que está aprendiendo, y en las categorías 1 y 5 consigue altas tasas de acierto (por encima del 70%). Sin embargo, en las categorías intermedias el algoritmo no es capaz de captar los criterios utilizados por los etiquetadores y se producen traspasos de predicciones entre categorías adyacentes. No obstante, las tasas de acierto que consigue la red son mucho mayores que las conseguidas con las herramientas de OpenCV.

4.6.3. OpenCV VS ResNet18:

Existen grandes diferencias en el coste temporal y la precisión entre estas tecnologías. Las redes neuronales, además de ser más rápidas de entrenar consiguen ser mucho más precisas que el clasificador de OpenCV.

Capítulo 5: Conclusiones

En este último apartado de la memoria se exponen las conclusiones a las que se han llegado tras los resultados obtenidos, posibles mejoras y limitaciones encontradas durante el desarrollo del proyecto.

En este Trabajo de Final de Grado se ha elaborado un "*dataset*" de imágenes de piezas de fruta y se han etiquetado en función de su calidad y a partir de él se han probado diferentes sistemas de selección y clasificación óptica con el fin de implementarlo en una línea de manipulación de cítricos y así sustituir un procedimiento manual.

Los objetivos específicos alcanzados son:

- Crear un "*dataset*" de imágenes de naranjas Navel con la captura de imágenes, la creación de un criterio basado en las recomendaciones de la OECD y el etiquetado en función de este.
- Elegir una arquitectura de red neuronal y diseñar un algoritmo de entrenamiento que clasifique correctamente por categorías.
- Valoración de los resultados obtenidos al final de proyecto teniendo en cuenta un criterio preestablecido que juzga el coste temporal y precisión de diferentes tecnologías.

El objetivo de las líneas de manipulación de cítricos es seleccionar y preparar un producto proveniente del campo para su comercialización siguiendo unos estrictos estándares de calidad de la forma más rápida y segura posible. La mayoría de los procesos a los que se somete la fruta están automatizados, sin embargo, todavía quedan procesos por automatizar que hasta ahora habían sido imposibles, ya que se tienen que aplicar ciertos criterios que en un principio no eran cuantificables.

Los nuevos sistemas de selección óptica son una tecnología muy prometedora en este campo. Esta tecnología apoyada por redes neuronales convolucionales es capaz de aplicar estos criterios aprendiendo de los nuestros. Como hemos visto, son capaces de obtener buenos resultados mejorando por mucho las herramientas de procesamiento y clasificación de imágenes como las que ofrece OpenCV.

Se pueden realizar múltiples cambios para mejorar este proyecto. En primer lugar, este trabajo se ha realizado utilizando las funciones de carga de datos de Pytorch que únicamente nos permite cargar imágenes para el entrenamiento. Habría sido interesante utilizar diferentes métodos de carga de datos que nos hubieran permitido crear elementos de entrenamiento con más de 3 capas (las capas RGB de las imágenes a color) que serían parecidos a los nombrados mapas de características. En segundo lugar, la mejora del etiquetado del "*dataset*". Además de que ha sido imposible por parte de la red neuronal de crear un criterio correcto para las categorías intermedias, se han encontrado fallos de etiquetado en varias ocasiones como: duplicidad de elementos en diferentes categorías, e imágenes mal clasificadas o características no capturadas por la imagen. Otra mejora del proyecto sería utilizar redes neuronales con mayor potencia. Se ha utilizado la red ResNet18 que cuenta con 18 capas, pero existen otras redes con muchas más capas, pero que demandan más potencia de cómputo. Por último, se podrían obtener mejores resultados si el tamaño y variedad del "*dataset*" de imágenes fuera mayor ya que podría identificar mejor las características de cada categoría.

Algunas de las limitaciones que se han enfrentado es la realización remota del trabajo. El no tener un entorno de ejecución ilimitado hace que no sean posibles algunos entrenamientos que necesitaran mayor potencia de cómputo y tiempo. Esto hace imposible la práctica de algunas técnicas. También se puede destacar los errores que se han visto en el "*dataset*" y que pueden confundir los criterios de la red neuronal. Finalmente, antes de implementar el optimizador bayesiano para el ajuste de hiperparámetros se estaba realizando con ensayos de prueba y error. Esta tarea junto con la alta duración de los entrenamientos se hacía muy costosa.

Bibliografía

[1] Benlloch, J. V., Agusti, M., Sanchez, A., & Rodas, A. (1995, October). Colour segmentation techniques for detecting weed patches in cereal crops. In Proc. of Fourth Workshop on Robotics in Agriculture and the Food-Industry (pp. 30-31).

[2] Sánchez, A. J., Albarracin, W., Grau, R., Ricolfe, C., & Barat, J. M. (2008). Control of ham salting by using image segmentation. Food Control, 19(2), 135-142.

[3] Sánchez, A., & Ramos, C. (2000). SEGUIMIENTO VISUAL DE OBJETOS UTILIZANDO TÉCNICAS DE PREDICCIÓN, XXI Jornadas de Automatica.

[4] Sanchez, A. J., & Marchant, J. A. (2000a). Fusing 3D information for crop/weeds classification. In Proceedings 15th International Conference on Pattern Recognition. ICPR-2000 (Vol. 4, pp. 295-298). IEEE.

[5] Ricolfe-Viala, C., & Sanchez-Salmeron, A. J. (2011, September). Optimal conditions for camera calibration using a planar template. In 2011 18th IEEE International Conference on Image Processing (pp. 853-856). IEEE.

[6] Ivorra, E., Amat, S. V., Sánchez, A. J., Barat, J. M., & Grau, R. (2014). Continuous monitoring of bread dough fermentation using a 3D vision Structured Light technique. Journal of Food Engineering, 130, 8–13. https://doi.org/10.1016/j.jfoodeng.2013.12.031.

[7] Ivorra, E., Sánchez, A. J., Camarasa, J. G., Diago, M. P., & Tardaguila, J. (2015). Assessment of grape cluster yield components based on 3D descriptors using stereo vision. Food Control, 50, 273-282.

[8] Verdú, S., Ivorra, E., Sánchez, A. J., Barat, J. M., & Grau, R. (2015a). Relationship between fermentation behavior, measured with a 3D vision Structured Light technique, and the internal structure of bread. Journal of Food Engineering, 146, 227–233. https://doi.org/10.1016/j.jfoodeng.2014.08.014.

[9] Grau, R., Sánchez, A. J., Girón, J., Iborra, E., Fuentes, A., & Barat, J. M. (2011). Nondestructive assessment of freshness in packaged sliced chicken breasts using SW-NIR spectroscopy. Food Research International, 44(1), 331-337.

[10] Ivorra, E., Sánchez, A. J., Verdú, S., Barat, J. M., & Grau, R. (2016). Shelf life prediction of expired vacuum-packed chilled smoked salmon based on a KNN tissue segmentation method using hyperspectral images. Journal of Food Engineering, 178, 110-116.

[11] Verdú, S., Ivorra, E., Sánchez, A. J., Barat, J. M., & Grau, R. (2015b). Study of high strength wheat flours considering their physicochemical and rheological characterisation as well as fermentation capacity using SW-NIR imaging. Journal of Cereal Science, 62, 31-37.

[12] E.M. Berti, A.J.S. Salmeron, F. Benimeli (2012a). Kalman filter for tracking robotic arms using low cost 3D vision systems. The Fifth International Conference on Advances in Computer–Human Interactions, Valencia, Spain, Jan. 30–Feb. 4, pp. 236-240.

[13] Berti, E.M.; Salmerón, A.J.S.; Benimeli, F. (2012b). Human-Robot Interaction and Tracking Using low cost 3D Vision Systems. Romanian J. Tech. Sci. Appl. Mech. 2012, 7, 1–15.

[14] Martinez, E.; Sanchez, A.; Ricolfe, C.; Nina, O. (2016). Human Pose Estimation for RGBD Imagery with Multi-Channel Mixture of Parts and Kinematic Constraints. WSEAS Trans. Comput. 2016, 15, 279–286.

[15] Martinez, E.; Sanchez-Salmeron, A.J.; Ricolfe-Viala, C. (2017a). 4D-DPM model for pose estimation using Kalman filter constraints. Int. J. Adv. Robot. Syst., 14, 1–13.

[16] Martinez, E.; Nina, O.; Sanchez, A.; Ricolfe, C. (2017b). Optimized 4D-DPM for Pose Estimation on RGBD Channels using polisphere models. In Proceedings of the 12th

International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, Porto, Portugal, 27 February–1 March 2017; Volume 5, pp. 281–288.

[17] Martinez-Berti, E., Sánchez-Salmerón, A. J., & Ricolfe-Viala, C. (2017). Dual quaternions as constraints in 4d-dpm models for pose estimation. Sensors (Switzerland).

[18] OECD (2010), Citrus Fruits, International Standards for Fruit and Vegetables, OECD Publishing, Paris.

[19] Stringfixer.com. (22-8-2021). https://stringfixer.com/es/Grid_search

[20] ICHI.PRO. (22-8-2021). https://ichi.pro/es/tutorial-rapido-uso-de-la-optimizacion-bayesiana-para-ajustar-sus-hiperparametros-en-pytorch-257248404255682

[21] Dr. Ernesto Conse Roca. (2018). Linea de manipulación: lavado, tratamiento, selección, calibrado y envasado. Curso tecnología postcosecha de cítricos y otros cultivos alternativos en la Comunidad Valenciana, Valencia, España, 2-2-2018. https://www.bibliotecahorticultura.com/wp-content/uploads/2018/01/CONESA-ROCA-Ernesto.-Febrero-2018.-Línea-de-manipulación-lavado-tratamiento-selección-calibrado-y-envasado-1.Presentación-.pdf

[22] Kaiming He, et.al. (2015). Deep Residual Learning for Image Recognition. 10-12-2015.

[23] Pytorch.org. (25-8-2021). https://pytorch.org

DOCUMENTO II: PRESUPUESTO

Contenido del Presupuesto

En esta parte del documento se hará la evaluación económica del proyecto "Diseño, implementación y evaluación de técnicas de clasificación de naranjas naveles mediante técnicas de visión por computador" acorde con el documento recomendado por la UPV: ·recomendaciones en la elaboración de presupuestos en actividade de I+D+I. Revisión 2018.

Un presupuesto tiene 3 partes diferenciadas:

- Coste personal:
  Es el coste de cada trabajador implicado en el proyecto calculado con la fórmula:
  $$\boldsymbol{Coste}(€) = \textbf{Coste horario}(€) \cdot \boldsymbol{Dedicacion\ en\ horas}$$
  Las horas dedicadas por cada participante han sido estimadas.
- Material inventariable:
  Las amortizaciones de equipos y licencias de software informático se calculan con la fórmula:
  $$\boldsymbol{Coste}(€) = \frac{\textbf{Tiempo de uso}(\textbf{meses}) \cdot \boldsymbol{Coste\ del\ equipo/software}}{\textbf{Periodo de amortización}\ (\textbf{años}) \cdot \mathbf{12}}$$

Clasificación económica del gasto	**Amortización (años)**
Adquisición de equipos para procesos de información	6
Adquisición de aplicaciones informáticas	6

- Presupuesto de ejecución

Coste de Personal

Núm	Denominación personal	Precio	Cantidad	Total
1	**Tutor/Profesor Catedrático de Universidad**	**51,4**	**30**	**1542**
2	**Graduado en ingeniería de Tecnologías Industriales**	**30**	**300**	**9000**
			Total Personal:	**10542**

Coste de Material Inventariable

Núm	Concepto	Precio	T. amortización	T. uso	Cantidad	Total
1	**Ordenador**	**1200**	**6**	**4**	**1**	**66,67**
2	**Conexión a internet**	**30**	**6**	**4**	**4**	**6,67**
3	**Sistema operativo Windows 10**	**145**	**6**	**4**	**1**	**8,06**
4	**Paquete ofimática office**	**69**	**6**	**4**	**1**	**3,83**
5	**Costes indirectos**				**10%**	**8,52**
				Total Material Inventariable:		**93,75**

Concepto	**Coste Total**
Coste de Personal	10542€
Coste de Material Inventariable	93,75€
Presupuesto de Ejecución Material	10635,75€

	Importe
Presupuesto de Ejecución Material	10635,75€
Gastos Generales (13%)	1382,64€
Beneficio Industrial (6%)	638,15€
Presupuesto de Ejecución por Contrata	12656,54€
IVA(21%)	2657,87€
	15314,42€

El coste total del proyecto asciende a **QUINCE MIL TRESCIENTOS CATORCE EUROS CON CUARENTA Y DOS CÉNTIMOS**.