Herramienta de ayuda en la detección de las características de la ropa por imágenes

Antonio Ocaña Mellado

Resumen—Este proyecto consiste en el desarrollo de una herramienta capaz de detectar las características de una prenda de ropa a partir de una imagen, pudiendo así asistir a las personas con disminución visual a vestirse de forma sencilla. La herramienta debe ser capaz de detectar que prenda de ropa se muestra en escena, que tipo de patrón sigue y de que color es. Para ello se han entrenado 2 redes neuronales convolucionales distintas, una destinada a detectar la prenda y otra encargada de detectar el patrón. Una vez detectada la prenda, mediante el algoritmo KMeans se extraen los colores presentes. Finalmente, la herramienta desarrollada detecta un total de 10 prendas de vestir, 5 patrones y 120 colores.

Paraules clau—DeepLearning, Visión por computador, Red Neuronal Convolucional, KMeans, VGG16, Dataset, GroundTruth, Finetunning.

Abstract—This project is about developing a tool able to detect the specific characteristics of clothes in order to help people with low vision dressing up in an easier way. The tool must be able to detect what kind of clothes are in each photograph, what pattern we can see on that image and what are the main colors on that garment. In order to achieve that, 2 convolutional neural networks have been trained, one of them aimed to detect what kind of clothes are and the other one, performs the pattern detection. To sum up, once that the clothes are detected, using a KMeans algorithm, the colors can be extracted. Finally, the developed tool is able to detect 10 differents clothes, 5 patterns and 120 colors.

Index Terms—DeepLearning, Machine visión, Convolutional Neural Network, KMeans, VGG16, Dataset, GroundTruth, Finetunning.

1 Introducción

EXISTEN un gran número de herramientas que estudian que moda se sigue en cada país o cuál es la tendencia de moda actual, sin embargo ninguna herramienta está orientada en la asistencia a las personas invidentes. Conseguir un sistema capaz de detectar qué pieza de ropa, de qué color y con que patrón está estampado supondría un gran avance para poder ayudar a un gran número de personas.

El objetivo principal es poder asistir a las personas con disminución visual en la labor de vestirse o escoger ropa a partir del desarrollo de una aplicación móvil capaz de captar qué tipo de prenda, qué patrón tiene estampado y de que color és a partir de una imagen.

1.1 Estado del arte

La detección y clasificación de prendas de vestir a día de hoy está muy avanzada, hay estudios [1] en los que se desarrollan redes neuronales destinadas a detectar la zona geográfica de una determinada foto basándose en el estilo y la moda de las personas que aparecían en la imagen. Otros estudios ayudan a los influencers a elegir vestimentas con las que poder agradar a un mayor número de personas, como puede observarse en la Figura 1 a la influencer de la imagen le recomiendan cambiar ligeramente su estilo.

Más allá de estudios también se comienza a popularizar las aplicaciones para dispositivos móvil que detectan distintos objetos. Entre estas aplicaciones se puede observar Google Lens, una herramienta capaz de detectar cualquier objeto y enlazarte con enlaces relacionados en tu nagador. En la misma temática podemos encontrar también aplicaciones como CamFind, que mediante una fotografía capturan las prendas de vestir y devuelve al usuario un conjunto de direcciones web donde comprar las prendas detectadas. Amazon Shopping y ASOS son más aplicaciones que detectan las prendas de vestir y ofrecen pasarelas de pago.

Fig. 1. Sistema recomendador de prendas para influencers.

- *E-mail de contacte: antonio.ocana@e-campus.uab.cat*
- *Menció realitzada: Computació*
- *Treball tutoritzat per: Marta Prim Sabrià*
- *Curs 2020/21*

Hay compañías, como *Codefactory SL* que desarrollan herramientas para ayudar a las personas invidentes, sin embargo, la mayoría de las herramientas están enfocadas en identificar colores.

Como se ha visto hasta el momento existen diversas tecnologías basadas en el reconocimiento de la vestimenta o en la detección de colores, sin embargo, ninguna de estas aplicaciones está orientada la asistencia a la vestimenta para personas con dificultades visuales, permitiéndoles saber si la camisa es roja y de cuadros.

2 Planificación

Para la planificación del proyecto se ha seguido la estructura mostrada en la Figura 2. El proyecto se compone en 3 bloques diferenciados en: Preparar datasets, Entrenar las redes neuronales y Extraer colores. A su vez estos 3 bloques se dividen en 6 entregas:

2.1 Primera entrega

En la primera entrega se efectuó un estudio previo para conocer el estado del arte del proyecto y las tecnologías y, a partir de él, elaborar la base del proyecto

2.2 Segunda entrega

En la segunda entrega se elaboró gran parte del bloque 1, se prepararon los datasets necesarios para poder entrenar la red neuronal encargada de detectar las prendas de vestir en una determinada escena.

2.3 Tercera entrega

En esta entrega se desarrolló por completo la red neuronal convolucional encargada de detectar las prendas. Esta entrega supuso el mayor cambio en base a la planificación del proyecto inicial. En un primer análisis del problema, se decidió utilizar la red neuronal YoloV3, no obstante, debido a la falta de datasets públicos con *groundtruth* se decidió cambiar el enfoque a utilizar imágenes sencillas donde sólo apareciese una prenda en escena.

2.4 Cuarta entrega

Para la cuarta entrega se ejecutaron tareas tanto del bloque 1 como del bloque 2. Se preparó un dataset propio con más de 10.000 imágenes de distintas telas estampadas. Así mismo se desarrollo la red neuronal convolucional encargada de detectar los estampados de una determinada prenda de vestir. Esta entrega supuso el segundo cambio de enfoque. La idea principal era utilizar una red ya existente y ejecutar *finetuning* [3], sin embargo, debido a que el dataset contenia imágenes muy simples no daba una gran tasa de acierto. Por ello, se decidió implementar una red más sencilla y reducida.

2.5 Quinta entrega

En la quinta entrega se elaboró el bloque 3: el extractor de colores. Para la exctracción de colores se ha utilizado el algoritmo KMeans y una lista de 120 colores basados en la paleta de los lápices de cera. Además, esta entrega supone el nexo de los 3 bloques de desarrollo, vinculando la salida de cada uno para el bloque siguiente y poder así detectar en base a una escena que prenda aparece, que patrón tiene estampado y de que color es la pieza.

2.6 Sexta entrega

La sexta entrega se reserva para extraer evidencias del desarrollo y la capacidad de detección que presenta, la extracción de conclusiones y el desarrollo del informe final mostrando todo el trabajo efectuado.

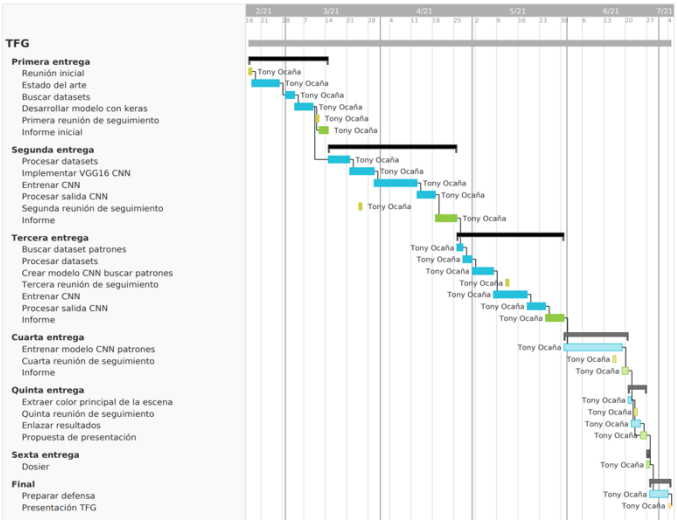

Fig. 2. Detalle de los 6 bloques durante la planificación del proyecto.

3 Metodología

Durante todo el proceso del desarrollo del proyecto se ha seguido la metodología de trabajo Kanban [2]. Kanban ofrece un sistema sencillo para poder llevar a cabo cualquier desarrollo. Se trata de un tablero dividido en el número de estados por lo que cada determinada tarea debe pasar hasta estar completamente terminada. Cada tarea se compone de una tarjeta en la que se especifica el objetivo de la tarea y de cuantas sub tareas se compone. Para identificar el estado de cada tarea se van colocando en el estado pertinente. Cómo se puede observar en la Figura 3 en el caso de este proyecto se han utilizado 3 estados: To Do, In Progress y Done. A continuación, se detalla cada estado.

To Do es el primer estado por el que pasan todas las tareas. El primer paso de todo proyecto es determinar las tareas que se deben ejecutar para lograr el objetivo, en este punto se añaden todas las tareas a esta sección.

In Progress es el segundo estado. A esta columna se avanza la tarea desde To Do en el momento en el que vamos a comenzar a desarrollar y no se avanza a la siguiente hasta que se haya terminado por completo. Como se puede observar en la Figura 3 en ese momento del desarrollo se encontraba aun en proceso el extractor de colores.

Done es el último de los estados. Cuando una tarea que se encontraba en In Progress se finaliza hay que mover la

tarjeta a este útimo bloque, de esta forma con un simple vistazo podemos saber que % de tareas quedan aun por resolver.

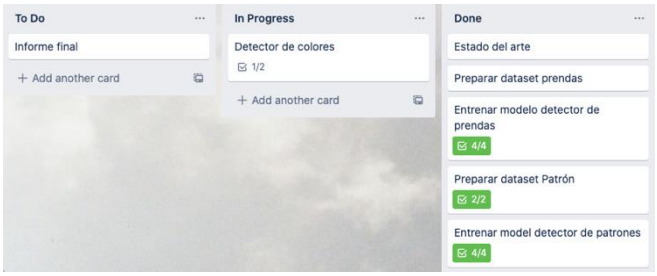

Fig 3: Tablero con las tareas del proyecto. En el se pueden ver los estados: To Do, In Progress y Done.

3.1 Herramientas software utilizadas

Todo el proyecto se ha desarrollado en Python 3.6 con un conjunto de librerías comúnmente utilizadas en el procesamiento de imágenes y redes neuronales. Se ha decidido utilizar Python 3.6 debido a que la mayoría de la información obtenida durante la fase de investigación estaba orientado a esta versión. A continuación, se especifican las librerías más importantes del desarrollo.

Numpy [4] ofrece una nueva forma de gestionar las matrices multidimensionales de forma sencilla. Además, viene con un conjunto de funciones matemáticas embedidas en el sistema para poder operar con las matrices de forma ágil. Se ha utilizado para gestionar las imágenes.

OpenCV [5] engloba un conjunto muy amplio de funciones definidas para poder procesar las imágenes: cargar, recortar, rotar…. Para el proyecto se ha utilizado principalmente para cargar la imagen, recortar la imagen y cambiar el espacio de color.

Keras [6] es una librería desarrollada por Google que permite generar redes neuronales de una forma muy abstracta y sencilla. Supone el componente principal del proyecto, con él se han desarrollado las 2 redes neuronales indicadas.

MatplotLib [7] ofrece un sistema para mostrar gráficas e imágenes. Durante el proyecto se ha utilizado para extraer las gráficas de los resultados y guardar las imágenes con el resultado de la predicción.

4 Desarrollo

Para llevar a cabo el desarrollo se ha dividido en tres fases principales:

Investigar datasets públicos: En esta fase se evaluaron los estudios hasta el momento y los datasets que utilizaban. La mayoría de los trabajos se basaban en tecnologías YOLO y Faster CNN para obtener reconocimiento instantáneo, sin embargo, la mayoría de los datasets publicados eran privados o no tenían el *groundtruth* asociado.

Entrenar los modelos: En esta fase se implementaron múltiples redes neuronales buscando cual és el modelo que mejor se adapta al conjunto de datos. Finalmente, se optó por efectuar *finetunning* para el modelo detector de prendas y un modelo más sencillo para detectar los estampados de las prendas.

Extraer colores: Se trata de la última fase, para desarrollarla se ha ejecutado el algoritmo KMeans. Este desarrollo está basado en la práctica de la asignatura Inteligencia Artificial cursada en segundo. A continuación, se detalla más extensamente cada una de las tareas indicadas.

4.1 Desarrollo de los datasets

Dataset de prendas: Para la base de datos necesaria para la detección de prendas de vestir se ha utilizado el dataset público *Clothing 5k Dataset* [8]. Se trata de una base de datos mantenida por los propios usuarios en la que en cada imagen solamente se muestra una única prenda. Debido a que se trata de una base de datos pública no contiene un gran número de clases distintas, únicamente cuenta con 10 clases.

Entre las clases disponibles se observa: Camisetas, Gorros, Zapatos, Chaquetas, Vestidos, Camisetas de manga larga, Pantalones, Pantalones cortos, Falda y Camisa. En la Figura 4 podemos observar la distibución de imágenes destinadas al entrenamiento, mientras que en la Figura 5 encontramos la distribución de las imágenes de validación.

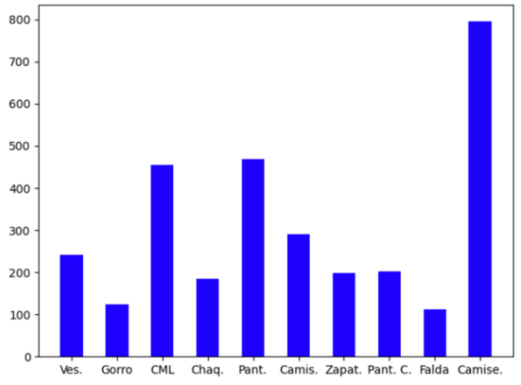

Fig. 4. Distribución de imágenes en cada una de las clases para el dataset encargado del entrenamiento. Observamos como la clase con más imágenes es la de Camisas.

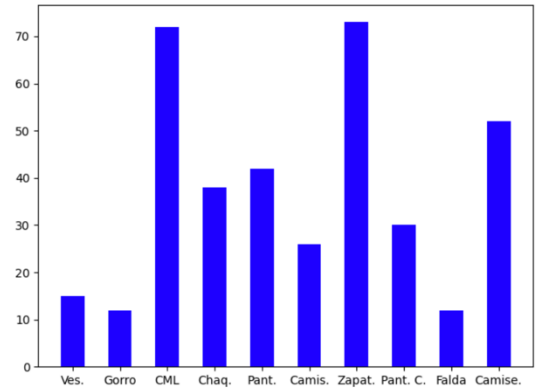

Fig. 5. Distribución de imágenes en cada una de las clases para el dataset encargado de la validación.

Dataset de patrones: Para la base de datos de patrones se ha decidido elaborar un dataset propio debido a la inexistencia de ningún dataset público que contenga los posibles patrones que presenta una determinada prenda de vestir. Para obtener las imágenes se ha utilizado el motor de búsqueda Google y un plugin aplicable en el navegador Chrome: Download All Images [9]. Con este plugin se descargan todas las imágenes cargadas en Google. Una vez descargadas las imágenes se revisaron una por una para eliminar todas las que presenten ruido y puedan hacer que la red neuronal no entrene con la eficacia requerida.

Para que una imagen se considere válida y libre de ruido debe cumplir los siguientes requisitos:

- El tamaño mínimo de la imagen debe ser 100 x 100 pixels.
- La imagen debe responder al patrón correspondiente.
- La imagen no debe contener texto ni marcas de agua ni debe representar ningún objeto.

Las clases que componen el dataset son: Punteado, Floral, Rayado y Cuadros. En la Figura 6 podemos observar como se distribuyen las imágenes en sus respectivas clases para el conjunto de entrenamiento, mientras que en la Figura 7 podemos observar la distribución del dataset de validación.

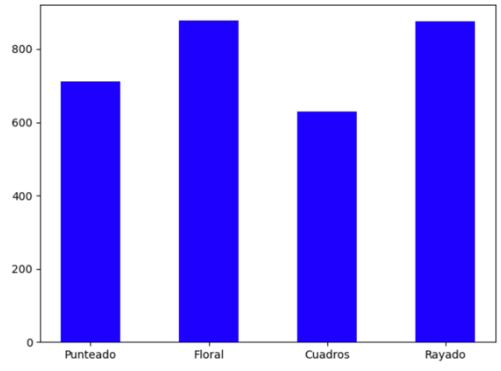

Fig. 6. Distribución dataset de patrones para entrenamiento.

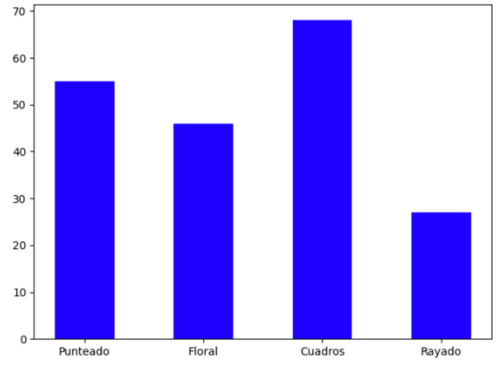

Fig. 7. Distribución dataset de patrones para validación.

4.2 Desarrollo de las redes neuronales

Para el desarrollo de las redes neuronales previamente se hizo un estudio de las redes actuales y su funcionalidad principal para poder adaptarlo a la detección de prendas de vestir. Existen un gran numero de redes destinadas a la detección de objetos en escenas de una forma muy rápida, como YoloV3 o Faster CNN. Estas redes ofrecen un accuracy final más bajo que una red como Imagenet, sin embargo, son mucho más rápidas detectando el objeto. Además de la velocidad ofrecen la posibilidad de detectar más de un objeto por escena. En un primer enfoque se decidió elaborar el proyecto en base a YoloV3, permitiendo al usuario detectar la ropa en vivo casi sin tiempo de espera. Debido a la falta de datasets para poder utilizar este tipo de red se optó por cambiar el enfoque y utilizar redes que detectasen una única prenda por escena. A continuación, se detallan las 2 redes neuronales empleadas para la detección de la prenda con su estampado.

Detector de prendas: Para la red neuronal convolucional encargada de detectar prendas de vestir en una determinada escena se ha optado por la red neuronal VGG16 [10]. VGG16 es una red neuronal capaz de detectar 1000 clases diferentes. Es una de las redes llamadas Very Deep Neural Network.

Keras ofrece el esqueleto de la red neuronal ya montado, pero sin entrenar, además, si se desea se puede inicializar con los pesos ya precargados, de forma que con apenas unas líneas de código se puede obtener un clasificador que reconozca 1000 clases distintas con un accuracy del 92,7%.

De base no es posible utilizar esta red para el cometido del proyecto, para adecuarla es necesario ejecutar *finetunning* y cambiar ligeramente la estructura de la red. Para ello nos centramos en la última capa, cambiando la capa con activación softmax a 1000 por una que solamente detecte 10 clases.

Una vez definida la red neuronal hay que preparar el dataset. Para ello se recorren todas las carpetas con las imágenes y se les asigna una clase: Camisa, Chaqueta… Una vez están todas las rutas y los labels cargados se procede a mezclar las posiciones, permitiendo así que la red neuronal aprenda de una forma más efectiva que si le pasaramos el dataset ordenado. Una vez se han mezclado las posiciones se cargan las imágenes, como se ha observado en la Figura 4 el conjunto de imágenes es muy pequeño, por ello se aplica la técnica de Data Augmentation [11]. De cada imagen se sacan 4 imágenes iguales en las que solamente se ha cambiado el ángulo de rotación.

La red neuronal VGG16 por defecto acepta imágenes cuadradas, por esta razón a medida que se van cargando las imágenes reales se les cambia el tamaño a 48 x 48 pixels, el tamaño más pequeño que acepta VGG16.

Este tamaño no ha sido arbitrario, se ha decidido un tamaño muy reducido debido a que el proyecto no necesita detectar materiales ni colores, simplemente nos importa la

forma de la prenda. También se ha optado por este tamaño por las limitaciones del ordenador de trabajo, se pueden ver las características en la Tabla 1.

TABLA 1
CARACTERÍSTICAS DEL ORDENADOR

Hardware	Características
Modelo	MacBook pro 2017
Procesador	Intel core i7 doble nucleo a 2,5GHz
Tarjeta Gráfica	Intel Iris plus Graphics 640
Memoria RAM	16 GB 2133 MHz LPDDR3

Finalizado el pre procesamiento de las imágenes se pasa a entrenar la red neuronal. Para el entrenamiento se procesan las imágenes en bloques de 256 (**batchsize** [12]) y, hasta que no se ha ejecutado el proceso 50 veces (**epochs**) no se actualiza el estado de los pesos. Como norma general se utilizan epochs de hasta 1000 iteraciones, sin embargo, debido a las limitaciones en el apartado gráfico mostradas en la Tabla 1 se ha decidido utilizar 50. Así mismo como se puede observar en la Figura 8, a partir de los 40 epochs el modelo se estabiliza a aproximadamente el 75% de accuracy en cuanto al conjunto de datos de validación. El error por otra parte, si observamos la Figura 9 podemos ver como el error mínimo se obtiene a los 25 epochs.

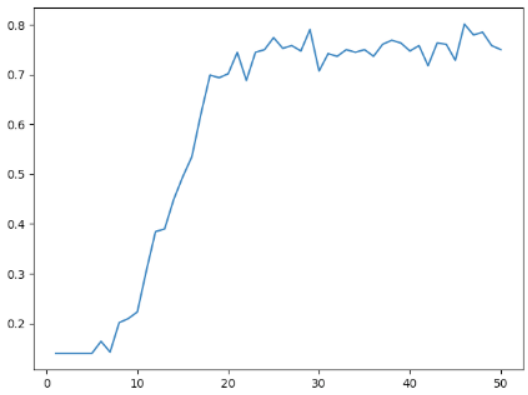

Fig. 8. Accuracy del modelo detector de prendas al entrenar.

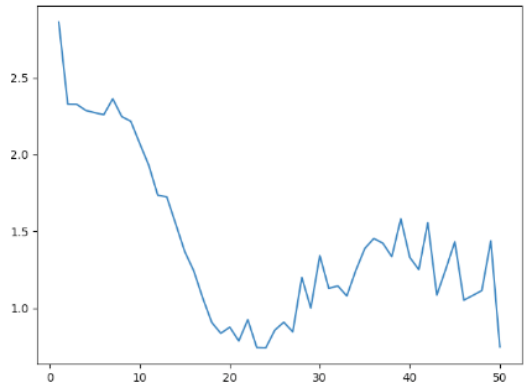

Fig. 9. Pérdida del modelo detector de prendas al entrenar.

Detector de patrones: Para la red neuronal encargada de detectar patrones en escena se desarrolló un modelo más sencillo que el anterior, un modelo secuencial [13]. Este modelo está basado en el expuesto por Keyur Rathod [14] para distinguir formas sencillas: Triangulos, Circulos y Cuadrados. A continuación, se detalla la composición de la red:

- Primera capa: Convolution2D con activación Relu.
- Segunda capa: MaxPooling2D.
- Tercera capa: Convolution2D con activación Relu.
- Cuarta capa: MaxPooling2D.
- Quinta capa: Flatten.
- Sexta capa: Dense de 56 caracteristicas con activación Relu.
- Última capa: Dense con las 4 clases finales a detectar con activación Relu.

Una vez definida la estructura base se detallará cada una de las distintas capas:

Convolution: Esta capa se encarga de recorrer la imagen en grupos de NxM efecutuando una operación matemática, usualmente se utiliza la multiplicación y el resultado se guarda en un nuevo filtro que es el que viaja hasta la entrada de la siguiente capa.

Relu: És una de las funciones de activación más utilizadas en redes neuronales convolucionales. Cuando a una neurona llega un valor negativo que no aporta ningún conocimiento la capa se le sustituye por un 0.

MaxPooling: Recorre la imagen con una matriz o stripe, em este de caso de 2 x 2 y combina los valores resaltados por la ventana para reducir así el número características que pasan a la siguiente capa.

Flatten: Se encarga de transformar una entrada de N dimensiones a una salida de 1 dimensión.

Dense: Interconecta todas las neuronas entre si de la capa anterior a las establecidas en esta nueva capa.

Softmax: Es la función de activación final del modelo. Se encarga de transformar la salida en una lista de probabilidades para detectar que porcentaje tiene un objeto de estar presente en la escena.

Para entrenar el modelo las imágenes deben preprocesarse para tener un tamaño de 128 x 128 pixels. Este tamaño es necesario para los patrones más complejos como el estampado floral, en el que si reducimos la imagen a un tamaño de 48 x 48 pixels se vería una imagen borrosa de la que la red neuronal no podría obtener ninguna información relevante y no entrenaría jamas.

Además de reducir el tamaño también se decidió cortar las imágenes. Para detectar un estampado en una prenda, como por ejemplo una camisa no es necesario buscar en

toda la imagen, podemos coger tan solo una porción. Para todas las prendas se recorta hasta quedarse con el 30% de la imagen central, salvo para los pantalones. Al encontrarse generalmente expandidos en la escena los pantalones tienden a dar falso positivo debido a que capta la escena de lo que hay entre cada pernera. Debido a ello en los pantalones se recorta para obtener el 30% superior de la imagen como puede observarse en la Figura 10.

Fig. 10. Pre procesamiento de un pantalón antes de pasar la imagen por la red neuronal encargada de detectar patrones.

Al necesitar un tamaño de imagen moderadamente grande la red neuronal con 50 epochs tarda aproximadamente 15 horas en completar el entrenamiento en un ordenador con las características mostradas en Tabla 1. Como se puede observar en la Figura 11, con el modelo mostrado se ha conseguido un accuracy del 77%, mientras que en la Figura 12 se puede observar como el error cae y se mantiene por debajo del 0,6.

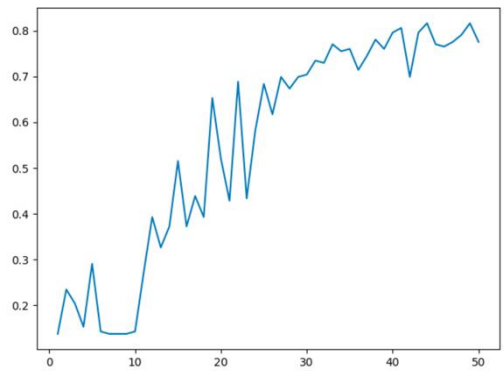

Fig. 11. Accuracy del modelo detector de patrones al entrenar.

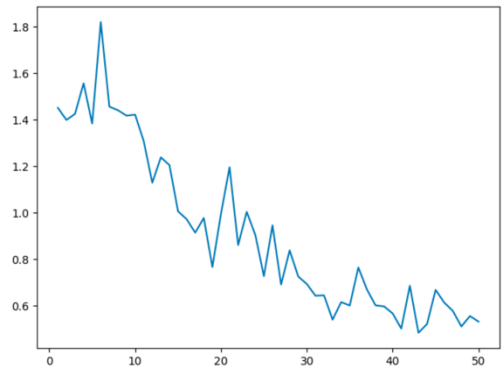

Fig. 12. Error del modelo detector de patrones al entrenar.

4.3 Desarrollo del extractor de colores

Para detectar los colores que predominan en una imagen se ha decidido utilizar el algoritmo KMeans [15] junto a la implementación del método de Elbow [16] para detectar el número de clusters/colores.

Para poder extraer los colores principales de una escena hay que seguir un conjunto de pasos:

Reestablecer dominio de colores: Al cargar la imagen, por defecto vene en BGR (Blue, Green, Reed) en lugar de en RGB (Red, Green, Blue). El primer paso es transformar la imagen al dominio que necesitamos para detectar correctamente los colores. Para ello se utilizó la librería OpenCV en la que se puede definir el espacio de color de una forma sencilla. En la Figura 13 se puede observar como sin el cambio se detectarían colores completamente distintos.

Fig. 13. Transformación del dominio de color de BGR a RGB

Cortar la imagen: La mayoría de las imágenes tienen una escena muy amplia con la prenda de vestir situada en el centro. Para retirar los bordes de una forma sencilla, se decidió cortar la imagen de tal forma que se obtuviese el 30% central de la imagen, tal como se puede observar en la Figura 14, para los pantalones cómo se puede observar en Figura 10 se obtiene el 30% superior.

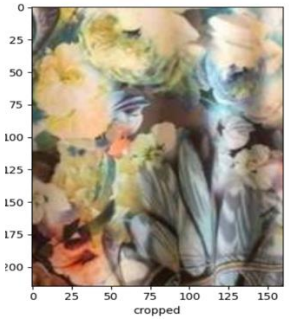

Fig. 14. Porción de imagen perteneciente al 30% válido para la detección de colores.

Detección de clusters: El siguiente paso consiste en cuantificar el número de colores principales que aparecen en la porción de imagen. Para obtener los colores se ha utilizado el método de Elbow presente en la librería *ScikitLearn*. Este algoritmo está basado en KMeans. Para obtener el número de colores presentes en una imagen basta con inicializar el método especificando cuantos colores

principales se requieren. En este caso se ha especificado un total de 11 colores. Elbow devuelve una lista con los 11 clusters demandados y la distancia que existe entre sus centroides. Como se puede observar en la Figura 15, para la porción de prenda anterior, se han detectado un total de 3 colores principales diferenciados.

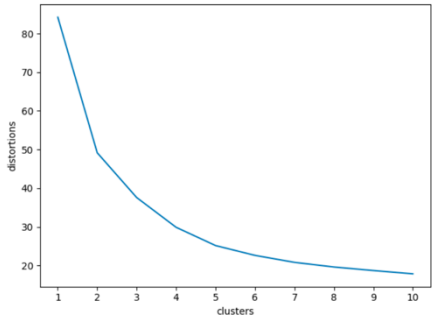

Fig. 15. Distancia media de cada pixel a su centroide.

Para detectar el mejor número de clusters, se recorre cada uno de los centroides pertenecientes a los clusters má ximos definidos, en este caso se estableció un total de 11, calculando la distancia entre cada uno de ellos. Según la distancia entre centroides se puede extraer cual és el número mínimo de colores necesarios para recrear la imagen. En caso que los centroides estén muy distanciados entre ellos más colores necesitará una imagen, mientras que si la distancia es insignificante menos colores serán necesarios.

Una vez obtenidas las distancias hay que buscar cual es la distancia mínima necesaria para discriminar colores. En este caso se ha establecido un *threshold* para una distancia mínima de 12 puntos.

Detección de colores: Una vez obtenido el número de colores, se puede extraer cuales son estos colores. Para ello se utiliza de nuevo KMeans de *ScikitLearn*. En este caso hay que inicializar el algoritmo diseccionando la imagen en sus correspondientes capas (R, G, B) y el número de colores obtenido anteriormente. Con esta llamada se obtiene una lista con tantos colores como se hayan obtenido. Una vez obtenida la lista de colores se procede a calcular la distancia entre el color detectado y 120 colores de referencia utilizados en el proyecto para encontrar el nombre del color.

5. Resultados

Tras entrenar cada uno de los modelos de forma separada y entrelazando los resultados por separado para devolver una única instancia en la que se detecte tanto la prenda, como el estampado como los colores presentes, se ha conseguido una tasa de aciertos elevada, en torno al 64%.

Como se puede observar en la Figura 16, el modelo es capaz de detectar patrones muy pequeños y colores muy cercanos entre ellos.

Además de detectar patrones, también es capaz de identificar si la prenda no contiene ningún patrón, en ese caso solamente se informa el tipo de prenda y el color que presenta tal como puede observarse en la Figura 17.

Fig 16. Falda con estampado rayado. Se ha detectado un patrón muy sutil con unos colores muy similares.

Fig. 17. Detección de unos zapatos lisos de color gris oscuro.

Pese a mantener un alto % de acierto la herramienta continúa teniendo algunos problemas al diferenciar algunas prendas. Debido a las pocas muestras para entrenar la primera red neuronal la herramienta tiene un alto acierto detectando camisetas, sin embargo, tal como puede observarse en la Figura 18 el 21% de las ocasiones que debe detectar un vestido la red neuronal lo confunde con una camiseta de manga larga. Este error no es únicamente debido a la falta de muestras, también incide que son dos prendas realmente similares. Aun con esa dificultad, el 79% restante detecta el vestido correctamente junto a los colores y al patrón. Como puede observarse en la Figura 19, detecta de forma correcta un vestido de flores.

Fig. 18. Vestido rayado con matices grises detectado como una camiseta de manga larga. Se puede observar como el sistema tiene problemas a la hora de diferenciar esas dos prendas.

Fig. 19. Vestido con estampado de floral detectado correctamente.

6. Conclusiones

El objetivo principal del proyecto era detectar la prenda de ropa que aparece en una determinada escena para poder asistir a una persona con una disminución visual elevada.

Pese obtener un 75% de acierto detectando prendas, un 77% detectando patrones y un 92% extrayendo colores, al combinar todas las tecnologías la tasa de acierto cae hasta llegar a un 64%.

Para poder haber conseguido una mayor tasa de acierto, es necesario contar con datasets que contengan un número mucho mayor de muestras.

Personalmente, el proyecto me ha parecido muy completo, engloba procesamiento de imágenes, redes neuronales y detección de colores. Además, entra la labor de poder ayudar a gente necesitada. Me llevo una maravillosa experiencia y unos conocimientos útiles en cuanto a Keras y sus redes neuronales

7. Lineas futuras

Este proyecto ha supuesto la base de un detector de prendas capaz de detectar con un acierto del 64% que prenda existe en la imagen, sin embargo, está limitado a una única prenda por imagen.

Una mejora necesaria para ofrecer un mejor servicio es cambiar las dos redes independientes y convertirlo en una red YoloV3. De esta forma podrían aparecer en escena muchas y diversas prendas de vestir y la herramienta sería capaz de detectarlas todas e incluso decir en que posición o donde se encuentran.

Con el cambio no sólo se gana en cuanto al número de objetos presentes, también supondría un incremento en la velocidad, de esta forma la persona invidente ya no estaría limitada a hacer una foto y procesarla, si no que podría enfocar con su teléfono móvil a un maniquí y el sistema responder en tiempo real que tipo de prenda lleva puesto.

Para llevar a cabo esta mejora previamente hay que preparar un dataset con muchas muestras y todas ellas con el *groundtruth* asociado.

Además de cambiar las redes neuronales actuales por una red Yolo, se podría desarrollar una aplicación móvil que explotase los recursos que ofrece este proyecto. De esta forma una persona con discapacidad visual, tan solo con su dispositivo móvil podría dirigir la cámara a su armario o a una sección concreta de una tienda de ropa y saber con exactitud que prendas de vestir aparecen en escena y en que posición se ubican.

Esta aplicación debería utilizar la tecnología *text 2 speech* para poder notificar al usuario mediante voz. Así mismo, la aplicación debería contar con soporte a *Google Assistant*, así el usuario controlaría la aplicación con la voz.

Agradecimientos

En primer lugar, quiero dar las gracias a mi pareja Marta, quien no ha dejado que me rindiese en ningún momento. Cada vez que me veía en el sofá me traía el ordenador espetando: ¡A trabajar! También agradecer a mi familia y amigos que durante todo este tiempo me han estado apoyando y ayudando en todo lo que han podido. Y, por último, pero no menos importante quiero agradecer a mi tutora Marta Prim por haber tenido mucha paciencia y haberme ido detrás a menudo para ver como avanzaba el proyecto.

Bibliografía

[1] Edgar Simo-Serra, Sanja Fidler, Francesc Moreno-Noguer y Raquel Urtasun. *Neuroaesthetics in Fashion: Modeling the Perception of Fashionability*. Último acceso el 24 de mayo de 2021.

[2] Redacción ABP. *¿En qué consiste la metodología Kanban y cómo utilizarla?*, 2021 en https://www.apd.es/metodologia-kanban/. Último acceso el 20 de junio de 2021.

[3] Victor Roman. *CNN Transfer Learning and Fine Tunning*, 2020 en https://towardsdatascience.com/cnn-transfer-learning-fine-tuning-9f3e7c5806b2. Último acceso el 15 de mayo de 2021.

[4] Numpy org. *What is Numpy*, 2021 en https://numpy.org/doc/stable/user/whatisnumpy.html. Último acceso el 25 de junio de 2021.

[5] Ramswarup Kulhary. *OpenCV Overview*, 2019 en https://www.geeksforgeeks.org/opencv-overview/. Último acceso el 25 de junio de 2021.

[6] Keras inc. *About Keras*. 2021 en https://keras.io/about/. Último acceso el 25 de junio de 2021.

[7] Karlijin Willems. *Matplotlib Tutorial: Python Plotting*, 2019 en https://www.datacamp.com/community/tutorials/matplotlib-tutorial-python. Último acceso el 17 de junio de 2021.

[8] Alexey Grigorev. *Clothing Dataset*, 2020 en https://medium.com/data-science-insider/clothing-dataset-5b72cd7c3f1f. Último acceso el 20 de mayo de 2021.

[9] Download All Images inc. *Download All Images Plugin*, 2021 en https://download-all-images.mobilefirst.me. Último acceso el 16 de mayo de 2021.

[10] Muneeb ul Hassan. *VGG16 – Convolutional Neural Network for Clasification and Detection*, 2018 en https://neurohive.io/en/popular-networks/vgg16. Último acceso el 29 de mayo de 2021.

[11] Arun Gandhi. *Data Augmentation | How to use Deep Learning when you have limited data*, 2021 en https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2. Último acceso el 30 de mayo de 2021.

[12] Jason Brownlee. *Difference between a Batch and an Epoch in a Neural Network*, 2018 en https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch. Último acceso el 26 de mayo de 2021.

[13] Juan Carlos Gonzales, *Tensor Flow para principiantes*, 2018 en https://www.apsl.net/blog/2018/02/02/tensor-flow-para-principiantes-vi-uso-de-la-api-keras. Último acceso el 20 de mayo de 2021.

[14] Keyur Rahod, Código fuente distribuido públicamente, 2018 en https://github.com/keyurr2/shape-classifier-cnn/blob/master/cnn.py. Último acceso el 24 de mayo de 2021.

[15] George Pipis. *Get the dominant colors of an image with kmeans*, 2020 en https://predictivehacks.com/get-the-dominant-colors-of-an-image-with-k-means. Último acceso el 12 de junio de 2021.

[16] Alind Gupta. *Elbow method for optimal value of k in KMeans*, 2021 en https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans. Último acceso el 10 de junio de 2021.