CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Diposit Digital de Documents de la UAB

Etiquetado automático de los colores de una imagen con redes neuronales

Luis Condes Díaz

Resumen— En este proyecto se propone una herramienta de clasificación automática de los colores en imágenes basada en redes neuronales. El objetivo es obtener una clasificación que permita identificar los diferentes colores que aparecen en una imagen. Para realizar dicho clasificador se han analizado la precisión de las múltiples arquitecturas de redes neuronales simples. Posteriormente se ha desarrollado un conjunto de funciones y una interfaz gráfica para poder aplicar el clasificador entrenado y se han realizado pruebas de clasificación para evaluar el rendimiento. Las redes neuronales entrenadas durante el proyecto permiten realizar clasificación de los colores en imágenes con unos resultados favorables en un tiempo aceptable, aunque todavía existe margen para poder mejorar.

Palabras clave— Etiquetado, Imagen, Clasificación, Visión, Color, Redes Neuronales, Color Naming, RGB, CIELab, Matlab

Resum— En aquest projecte es proposa una eina de classificació automàtica dels colors en imatges basada en xarxes neuronals. L'objectiu principal és obtenir una classificació que pugui identificar els diferents colors que poden aparèixer en una imatge. Per poder desenvolupar aquest classificador s'ha analitzat la precisió de diverses arquitectures de xarxes neuronals simples. Posteriorment s'ha desenvolupat un conjunt de funcions i una interfície gràfica per poder aplicar el classificador ja entrenat i s'han realitzat proves de classificació per avaluar el rendiment. Les xarxes neuronals entrenades durant el projecte permeten realitzar classificació dels colors en imatges amb uns resultats favorables en un temps acceptable, tot i que encara hi ha possibilitat d'obtenir millores.

Paraules clau— Etiquetatge, Imatge, Classificació, Visió, Color, Xarxes Neuronals, Color Naming, RGB, CIELab, Matlab

Abstract— In this project we developed an automated color classification tool for images using neural networks. The main objective is to obtain a classification model good enough to allow identifying different colors inside natural images. To accomplish this, we analyzed the accuracy of the training in multiple simple neural network architectures. We followed with the development of a library of functions and a graphic user interface that allows applying the trained classification model to user provided images and performed multiple tests to evaluate the performance. The obtained performance from the trained neural networks during this project is satisfactory with an acceptable time, but there's always room for improvement.

Index Terms— Labeling, Image, Classification, Vision, Color, Neural Networks, Color Naming, RGB, CIELab, Matlab

1 Introducción

El ser humano es capaz de percibir un sinfín de colores, desde los tonos más brillantes de rojo hasta el azul más oscuro pasando por infinidad de diferentes tonalidades. Sin embargo, cuando hablamos con nuestros amigos o conocidos podemos encontramos disparidad de opinión a la hora de nombrar ciertas tonalidades de color. Esto es debido a como cada individuo posee una percepción única del color de la luz que pasa a través de sus ojos.

Estas diferencias tienen múltiples fuentes, siendo la más importante de carácter natural y que proviene directamente del funcionamiento del ojo humano. El ojo humano está constituido por diversas partes con funciones muy diferenciadas y, en concreto, uno de los principales elementos responsables del procesamiento de la luz se encuentra en el interior de la retina. Sobre la retina, es proyectada una imagen a través de todo el ojo. La luz proyectada provoca reacciones bioquímicas en células llamadas conos y bastones que se encuentran en el interior de la retina, representadas en la Figura 1.

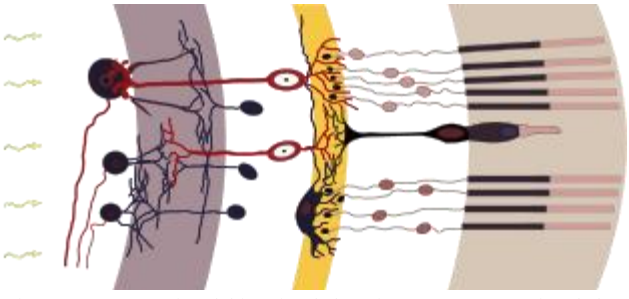

Fig. 1. Representación del interior de la retina. La representación de los conos y los bastones puede verse en el lado derecho de la imagen.

- *E-mail de contacto: luis.condesdiaz@gmail.com*
- *Mención realizada: Computación*
- *Trabajo tutorizado por: Robert Benavente Vidal e Ivet Rafegas Fonoll Departament de ciències de la computació*
- *Curso 2015/2016*

Estas células son las encargadas de retransmitir cambios de color y luminosidad en el nervio óptico. Existen tres tipos de conos en función de las longitudes de onda a la que ofrecen una respuesta más fuerte. Esto hace que cada tipo de cono tenga afinidad a ciertos colores. En función del número de conos de color de cada tipo y de cómo están ordenados dentro de la retina, la percepción de color puede ser más afín a ciertas tonalidades antes que otras. Si hay tonalidades de color muy diferentes y muy juntas, al proyectarse en nuestra retina es posible que se produzca distorsión del color detectado por los conos de color. Si la luminosidad es muy baja los conos de color no son capaces de realizar su función y sólo recibimos información visual de los bastones. Los bastones son células que ofrecen respuesta en condiciones de baja luminosidad, pero no proporcionan información de color.

Además de estas razones naturales, existen diferencias culturales y sociales en las que ciertos colores son más importantes que otros o tienen significado. A lo largo de la historia, han existido lenguajes que ni siquiera contienen palabras para expresar ciertas tonalidades específicas de color que si existen en otros lenguajes.

Estas diferencias hacen que exista disparidad a la hora de identificar o clasificar la información de color. Este trabajo pretende utilizar aprendizaje computacional basado en redes neuronales simples para aproximar un modelo que pueda resolver el problema de la clasificación de color en tiempo razonable.

La clasificación de color tiene múltiples aplicaciones prácticas, como el etiquetado de imágenes para su posterior organización por color, la toma de decisiones en sistemas autónomos a través de imágenes y que requieran información del color o incluso sistemas de asistencia para personas con problemas de percepción de color.

Este artículo está dividido en siete apartados. En primer lugar se repasarán cuáles han sido las aportaciones más importantes a la resolución del problema de la clasificación de color y de las dificultades encontradas. A continuación se explicarán conceptos clave para este artículo, como en que consiste el problema de la clasificación de color y el funcionamiento de una red neuronal. Acto seguido se explicará la resolución que se le ha dado al problema y los resultados conseguidos. Finalmente, extraeremos las conclusiones y un conjunto de propuestas de mejora para el futuro.

2 Objetivos del trabajo

El objetivo principal de este trabajo es el desarrollo y la documentación de un conjunto de funciones que permita, a través de redes neuronales, hacer clasificación del color en función de sus nombres en cada uno de los píxeles de una imagen con un tiempo y una precisión aceptables.

Además del objetivo principal, existen una serie de objetivos secundarios:

- Estudiar el estado actual del problema y las soluciones ya propuestas.
- Realizar el análisis del comportamiento de la clasificación a través de métricas como *accuracy* o *confusion matrix.*
- Diseñar y desarrollar una interfaz gráfica que facilite el acceso a las funciones desarrolladas.
- Comparar los resultados con los de otros estudios a través de una base de datos estándar.

3 Estado del arte

A lo largo de los últimos 15 años, de la mano de la evolución de los sistemas computacionales han aparecido estudios con resultados muy interesantes resolviendo el problema de la clasificación de color en el ámbito científico como el estudio realizado en 2006 por R. Baldrich *et al.* [1] en el que este artículo se basa y en el que se resuelve el problema de la clasificación de color a través de conjuntos difusos [14] que definen los lindes entre las clases de color en espacios de color concretos o el estudio realizado en 2009 por van de Weijer *et al.* [2] donde se utiliza el método del análisis sintáctico probabilístico latente (en inglés: Probabilistic Latent Syntactic Analysis o PLSA [3]) para aprender la información de color de imágenes extraídas de Google Imágenes con etiquetas de color potencialmente incorrectas o confusas. También artículos muy recientes como el artículo publicado en la conferencia internacional del Instituto de Ingenieros eléctricos y electrónicos (en inglés: Institute of Electrical and Electronics Engineers o IEEE) de 2015 por Y. Wang *et al.* [4] donde solucionan el problema de la clasificación de color utilizando redes neuronales convolucionales [5] y el artículo publicado por Y. Liu *et al.* [6] donde a través de la segmentación de una imagen en imágenes más pequeñas pueden propagar las etiquetas de cada uno de los colores para detectar como varía un color cuando está ensombrecido/iluminado y así reducir notablemente el impacto de la iluminación y la reflexión antes de clasificar el color de una imagen.

4 Clasificación de color

El problema de la clasificación de color consiste en nombrar un color descrito en cualquier espacio de color con su correspondiente nombre en cualquier idioma. Desde el punto de vista computacional, no es sencillo hacer una conversión de un valor de color a un nombre concreto, ya que existen múltiples formas de describir un color y además existen cientos de nombres de colores con diferencias variables. A nivel de percepción, podríamos llamar rojo a tonos granates o burdeos, pero sin embargo difícilmente diremos que un tono lima o anaranjado puedan corresponder a un amarillo. Limitar el número de clases de color es un punto clave en la resolución del problema ya que cada color tiene un área sobre el espacio diferente. Esto hace complicada la definición de los lindes que existen entre los colores, puesto que afectarán de

manera directa a cómo de buena será nuestra clasificación. La mejor clasificación, por tanto, será la que se acerque más a una descripción humana.

4.1 Percepción

Como se ha introducido al principio del artículo, cada individuo tiene una percepción única del color debido a la arquitectura del ojo humano. Esto hace que la descripción humana del color pueda tener variaciones de una persona a otra. Esto dificulta la definición de un modelo perfecto de color con el que comparar cualquier posible clasificación de color, pero se puede aproximar. Por suerte, el Centro de Visión por Computador de la Universidad Autónoma de Barcelona realizó un estudio de la percepción de color humana en 2006 [1] dónde personas reales describían el color que veían en un entorno controlado y estos resultados fueron modelados en un conjunto difuso que define los lindes existentes entre once colores concretos en la percepción de color humana. Este modelo lo utilizaremos de base para generalizar una solución al problema de la clasificación de color.

4.2 Espacios de color

Dentro del mundo de la informática y los sistemas computacionales el color aparece de manera notable en diseño asistido por ordenador, edición e impresión de imágenes. Actualmente disponemos un amplio abanico de formatos de representación de color para poder preservar la información de color de nuestras imágenes favoritas.

Uno de los formatos más antiguos de representación de color (y a estas alturas, un estándar) es el RGB. El formato de color RGB está compuesto por tres canales de color Rojo (Red), Verde (Green) y Azul (Blue) ya que representa todos los colores a través de combinaciones de estos tres. Como hemos visto anteriormente, la composición de colores a partir de tres componentes también se realiza en el ojo humano, y en tonalidades parecidas al RGB. Es fácil pensar que el RGB es un formato uniforme para representar la percepción del color, pero no es así. En RGB, la representación de la luminosidad se hace a través de la saturación de cada uno de los canales de color, siendo el color blanco la saturación máxima en los tres canales. Esto hace que los tonos grises, blancos y negros tengan menor representación en el espacio y, por tanto, no uniforme. Por esta misma razón, existen espacios de color orientados a hacer uniforme el color de manera perceptual donde la luminosidad es representada con un canal único.

Uno de los espacios de color más utilizado que tiene un canal de luminosidad separado es el CIELab, donde el color se representa en un plano mediante dos coordenadas (a, b) como podemos observar en la Figura 3.

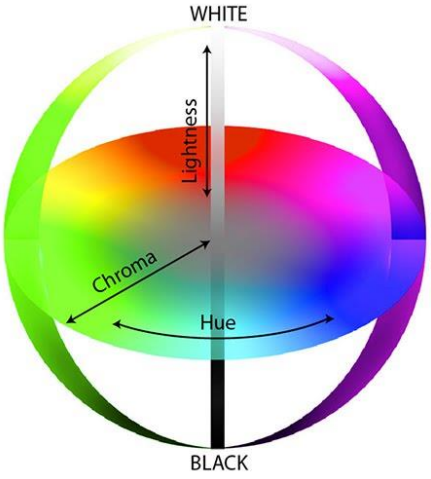

Fig. 3. Representación de colores en el espacio de color CIELab

Aunque el espacio de color CIELab sea uno de los más apropiados para la representación de la percepción del color, no está demasiado extendido como estándar de representación del color y es más utilizado en entornos de investigación. Por esta misma razón durante este estudio se dará una solución para la clasificación de color en el espacio de color RGB y otra en el espacio de color CIELab.

5 Redes neuronales

Una red neuronal es un sistema de aprendizaje computacional capaz de aproximar un modelo matemático de manera precisa si le podemos proporcionar suficientes datos de entrada y de salida. La red neuronal recibe los datos de entrada y produce una salida. Esta salida se compara con la salida correcta para los datos proporcionados y se genera un modelo de error. A partir de este modelo de error la red neuronal es capaz de modificar valores en su estructura para acercarse al resultado correcto a partir del modelo de error. Dado un número suficiente de datos de entrada y de iteraciones sobre todos los datos de entrada, la red neuronal es capaz de ajustarse hasta lograr una precisión adecuada en la salida para todos los datos de entrada introducidos. Cuando esto ocurre, decimos que la red neuronal ha sido entrenada. Para poder asegurar un buen entrenamiento hay que tener en cuenta tres factores: los datos de entrada, la arquitectura de la red y la distribución del error en la red. Una vez entrenada, la red neuronal se puede utilizar para resolver el problema sin necesidad de entrenar de nuevo.

5.1 Arquitectura

Una red neuronal define su arquitectura a partir de su unidad mínima: el perceptrón. Un perceptrón es una unidad de cálculo que intenta modelar el comportamiento de una neurona "natural", similares a las que constituyen del cerebro humano.

Un perceptrón tiene un número indefinido de entradas y el mismo número de pesos.

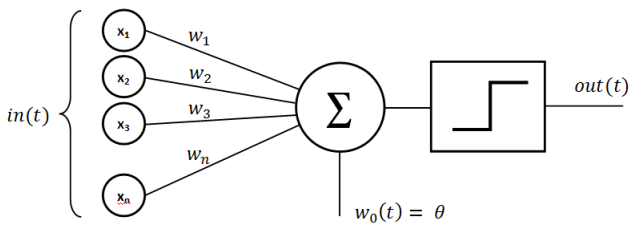

Fig. 4. Estructura de un perceptrón. A la izquierda podemos ver las entradas del perceptrón ponderadas por un conjunto de pesos W y evaluada por la función de activación a la derecha.

Cada peso multiplica a su valor de entrada correspondiente, acto seguido todos los valores de entrada se suman y se comprueba que esta suma supera cierto linde en la función de activación. El grado de excitación del perceptrón dependerá de la evaluación de la función de activación. Este grado de excitación es la salida del perceptrón. Esta estructura la podemos ver en la figura 4.

Dentro de una red neuronal, los perceptrones se apilan en capas. Estas capas forman la red neuronal. Todos los perceptrones de una capa pueden conectar sus salidas con todas las entradas de los perceptrones de la siguiente capa. La primera capa es la capa de entrada o input layer y tendrá tantos perceptrones como número de características o propiedades distintas se quiere representar los datos de entrada. La última capa es la capa de salida u output layer y tendrá tantas salidas como el número de clases en las que uno desea modelar el conjunto de datos de entrada. Todas las capas intermedias se consideran capas ocultas.

En las redes neuronales simples se coloca una única capa oculta, pero se pueden colocar cualquier número de capas para poder realizar aprendizaje profundo.

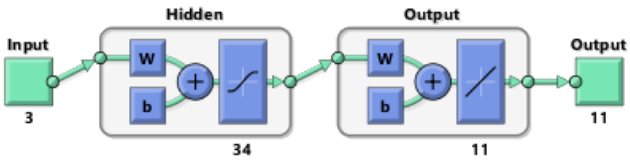

Fig. 5. Imagen de la arquitectura de red neuronal utilizada para resolver el problema. Imagen generada por Matlab.

El número de perceptrones de la capa de entrada viene definido por el número de datos diferentes en los datos de entrada, mientras que el número de perceptrones de la capa de salida depende del número de datos diferentes en los datos de salida. Podemos ver la arquitectura de una red neuronal simple en la figura 5. El número de perceptrones de las capas ocultas para una aproximación óptima varía con la naturaleza de cada problema.

5.2 Entrenamiento

El entrenamiento de una red neuronal se puede partir en dos partes igual de importantes. La primera parte corresponde a la inicialización de la red y sus parámetros mientras que la segunda parte corresponde a la propagación del error y la selección de nuevos datos de entrenamiento. Durante la inicialización de la red es importante saber con qué pesos inician cada uno de los perceptrones de la red neuronal. Estos valores usualmente es mejor iniciarlos de manera aleatoria, pero esto dificulta dar significado a los resultados. Los parámetros de la red neuronal indicarán cuándo se debe detener el entrenamiento.

La neural network toolbox de Matlab utiliza hasta cinco parámetros que pueden detener el entrenamiento. Estos parámetros son: número de iteraciones, valor del gradiente, valor adaptativo del algoritmo de Levenberg–Marquardt [7] (si se utiliza en el método de propagación escogido), numero de pesos utilizados de manera efectiva y suma del error cuadrático. Los límites de estos parámetros se pueden establecer antes del entrenamiento para mejorar los resultados. En la segunda parte tenemos la elección del método de propagación del error y sus parámetros. La neural network toolbox de Matlab nos ofrece hasta cuatro tipos de propagación de error: Levenberg–Marquardt backpropagation [8], Bayesian Regularization backpropagation, Scaled conjugate gradient backpropagation [9] y Resilient backpropagation [11], además de permitirnos construir un método personalizado de propagación del error. Elegir el método correcto de propagación del error depende del problema y de los recursos de los que podamos disponer.

Ciertos métodos de propagación de errores ofrecen mejores resultados que otros cuando tenemos pocos datos de entrada o datos de entrada con mucho ruido, como es el método de la regularización bayesiana. A cambio, la regularización bayesiana hace un uso más intensivo de la memoria mientras que otros métodos como el método de Levenberg-Marquardt son menos intensivos en la memoria necesaria para utilizarlos, pero requieren datos de entrada más grandes y con menos ruido para ofrecer el mismo resultado. Volver a seleccionar datos de entrada durante el entrenamiento es una muy buena manera de asegurar que todos los datos de entrada se van a utilizar. Una red neuronal necesita al menos dos sets de datos a utilizar: el set de entrenamiento y el set de validación.

Es recomendable que los datos de entrenamiento y los datos de validación sean diferentes. Un elegante truco que se puede hacer es dividir un conjunto de datos de entrada en sets más pequeños e intercambiarlos después de cada iteración. De esta manera todos los datos del conjunto de datos pasan en algún momento por ambos sets de entrenamiento y validación, pero nunca en los dos durante la misma iteración.

6 Propuesta de resolución

Para poder solucionar el problema de la clasificación de color se ha utilizado Matlab para producir una conjunto de funciones y una interfaz gráfica para la clasificación de color con redes neuronales. Matlab provee un conjunto de funciones para crear y entrenar redes neuronales llamada Neural Network Toolbox. Este conjunto de herramientas además nos permiten exportar una red neuronal ya entrenada a una función de Matlab que podemos llamar posteriormente con cualquier color de entrada y la función nos devolverá su clasificación.

Como datos de entrada para entrenar las redes neuronales utilizaremos un conjunto de datos basado en conjuntos difusos [1] realizado por el Centro de Visión por Computador de la Universidad Autónoma de Barcelona. Este conjunto de datos de entrada contiene la clasificación de cada color realizada en un estudio de percepción humana con el color representado en los espacios de color RGB y CIELab y seguido de once porcentajes de pertenencia de rango 0 a 1 en once colores básicos seleccionados por ser los más comunes en todos los lenguajes del planeta. Estos colores son: Rojo, Rosa, Marrón, Naranja, Amarillo, Verde, Azul, Violeta, Negro, Gris y Blanco. Este formato define las entradas y salidas de las redes neuronales, teniendo tres perceptrones (uno por canal de color) en la capa de entrada y once perceptrones (uno por cada porcentaje de color) en la capa de salida. Estos datos de entrada están conformados por cuatro conjuntos de colores: dos sets pequeños de 387 elementos y dos sets intermedios de aproximadamente 140000 elementos representados tanto en RGB como en CIELab.

Para poder encontrar los parámetros adecuados para las redes neuronales se decidió entrenar en primer lugar los sets pequeños y entrenar los sets más grandes sólo cuando se hubiera encontrado una arquitectura que ofrezca buenos resultados. Para poder comprobar si el aprendizaje de una red neuronal ha sido satisfactorio, hay diversas métricas que nos pueden ayudar a medirlo. Durante este proyecto se han utilizado dos métricas: Accuracy y Confusion Matrix.

La métrica de precisión (Accuracy) evalúa cómo ha ido el aprendizaje comparando cómo se ha etiquetado un conjunto de datos con sus etiquetas reales y obtiene un porcentaje de aciertos y un porcentaje de errores.

La matriz de confusión (Confusion Matrix) es una matriz cuadrada que contiene cada posible etiqueta en la clasificación tanto en las filas como en las columnas. Cada valor de la matriz representa cuantos elementos que pertenecen a la etiqueta fila se han clasificado como la etiqueta columna. Idealmente, la matriz debería ser una matriz diagonal donde todos los elementos que pertenecen a todas las etiquetas fila han sido clasificadas en las mismas etiquetas columna. Esto significa que todos los elementos se han etiquetado correctamente. Esta matriz nos permite ver qué etiquetas están mejor o peor entrenadas y la población de cada etiqueta. Un ejemplo de esta matriz lo podemos ver en la figura 6.

	Rojo	Rosa	Mar.	Nar.	Ama.	Ver.	Azul	Vio.	Neg.	Gris	Bla.
Rojo	8262	122	36				137	258		5	
Rosa	151	5684	89	101				40		9	4
Mar.	90	225	3611	101	76					1	
Nar.		87	31	8786	163					3	
Ama.			19	130	44034	111					2
Ver.					348	23475	654		2		
Azul	25					168	27621	610	1		
Vio.	285	56					812	10734			1
Neg.	50		117		170	97	18		154	84	
Gris	19	49	51	14	443	445	156	90		695	
Bla.		22		120	85	161	2	115		42	249

Fig. 6. Confusion matrix de la red neuronal del espacio de color RGB. Los números miden cantidad de colores clasificados. Los colores en orden de las filas y las columnas son: Rojo, Rosa, Marrón, Naranja, Amarillo, Verde, Azul, Violeta, Negro, Gris y Blanco.

El primer entrenamiento se hizo proporcionando un 70% aleatorio del set pequeño como datos de entrenamiento y un 15% como datos de validación, utilizando el método de regularización bayesiana como propagación de error y limitando el número de iteraciones a 1000. La arquitectura inicial consiste en una sola capa oculta de 33 perceptrones (perceptrones de entrada por perceptrones de salida). Para encontrar qué arquitectura es la más adecuada, se ha creado un programa capaz de construir redes neuronales simples con un número concreto de perceptrones en la capa oculta, entrenar estas redes y almacenar su precisión para poder compararlas. Un breve resumen de las arquitecturas probadas se puede ver en la figura 7.

Número de capas ocultas:	1, 2
Número de perceptrones:	5, 15, 16, 17… , …50
Número de elementos:	387, 138700, 140600
Propagación de errores:	Levenberg-Marquardt, Regularización Bayesiana
Espacio de color:	RGB, CIELab
División de los datos:	70-30, 60-40, 50-50

Fig. 7. Resumen de las arquitecturas probadas en la clasificación.

La arquitectura escogida finalmente ha sido con una sola capa oculta y 34 perceptrones, ya que obtuvo el mejor resultado de precisión sobre RGB.

Posteriormente se hizo el entrenamiento de ambas redes neuronales con el conjunto grande de datos, se aumentó el número de iteraciones a 2500 e incrementó la sensibilidad del gradiente. Dada la gran cantidad de datos, se aplicó reducción de memoria para poder procesarlos. Entrenar una red neuronal con el set grande y los recursos disponibles requería aproximadamente 8 horas de trabajo intensivo para el procesador del ordenador.[1]

Una vez encontrada una buena arquitectura, es importante cómo se procesa la información. Las funciones de Matlab generadas con la red neuronal entrenada admiten la entrada de un pixel o de una fila de píxeles. Introducir una fila completa de píxeles permite a Matlab optimizar las operaciones de clasificación como operaciones matriciales e incrementa la eficiencia de la clasificación. La función con la red neuronal entrenada ofrece once valores de salida del modelo. Estos valores representan pertenencia a una o varias clases. Para las funciones de la librería simplemente se ha escogido la etiqueta que mayor valor de pertenencia produce, pero de la red neuronal se puede extraer la información completa, existiendo casos en los que puede haber más de un valor de pertenencia con los valores bastante altos. Esto permite observar escenarios de incertidumbre para el modelo, igual que un ser humano.

Para poder clasificar toda una imagen, hay que recorrer todos los píxeles de la misma. Esto en imágenes de poca resolución puede ser sencillo, pero cuando el tamaño de las imágenes aumenta el número de píxeles a clasi-

1. *Estos datos fueron captados con una máquina equipada con 8 GB de memoria RAM DDR3 a 800 Mhz.*

ficar se dispara. Dada la naturaleza del problema, es posible reducir drásticamente el número de píxeles a clasificar si sólo escogemos un pixel de cada dos o incluso tres. Por ello, la función de clasificación de una imagen permite establecer un parámetro de paso que dicta cada cuantos píxeles se escoge uno para clasificación. Si la imagen es una fotografía, aumentar el paso a dos reduce a la mitad el tiempo de procesamiento y produce el mismo resultado con un muy pequeño margen de error. Para aumentar la precisión, la función de clasificación de una imagen puede utilizar una ventana de píxeles a los cuales aplica un filtro de desenfoque gaussiano hacia el centro. Posteriormente utiliza el pixel central de esta ventana y la red neuronal adecuada para poder clasificar el color. Esto le permite a la función tener en cuenta cómo los colores de los píxeles que hay alrededor de un pixel concreto pueden afectar a la percepción de color. El parámetro de ventana es completamente compatible con el parámetro de paso.

Finalmente y de manera adicional, se ha incluido una interfaz gráfica para un uso simple y rápido de la librería. También se han hecho test de detección de formas en una base de datos estándar formada por un conjunto de imágenes de objetos.

7 Implementación

Se han programado un conjunto de funciones en Matlab que aplican redes neuronales para la clasificación de color. Además, se ha creado una interfaz gráfica que hace aún más sencillo el uso de las funciones del trabajo.

Se han desarrollado funciones que generan resultados de clasificación de color para un pixel o para una imagen completa. También se han desarrollado múltiples funciones para interpretar los resultados. La documentación de estas funciones puede encontrarse en el anexo 1.

7.1 Interfaz Gráfica

La interfaz gráfica permite hacer uso de manera rápida y sencilla de las funciones de la librería. Podemos ver la interfaz gráfica en la figura 8. La interfaz gráfica está compuesta por cinco secciones: Acciones, Información Datos, Imagen, Imagen clasificada e Histograma.

En la sección de Acciones hay 5 botones. Al arrancar la interfaz sólo está disponible “Open Image File…” que nos permite cargar una imagen desde un fichero en el disco duro. Una vez cargada la imagen aparecerá en la sección Imagen, sus datos aparecerán en la sección Información y las opciones “Scan file as RGB” y “Scan file as LAB” quedarán disponibles. Estas opciones permiten clasificar la imagen cargada en uno de los dos espacios de color soportados. Los ficheros cargados siempre estarán en RGB, así que cuando se selecciona “Scan file as LAB” previamente se hace una conversión del fichero a CIELab antes de clasificarlo.

Después de clasificar la imagen, las opciones “Generate Histogram” y “Generate Class Image” quedarán disponibles. Estas opciones permiten, respectivamente, generar el histograma de los colores clasificados y generar la imagen de clasificación.

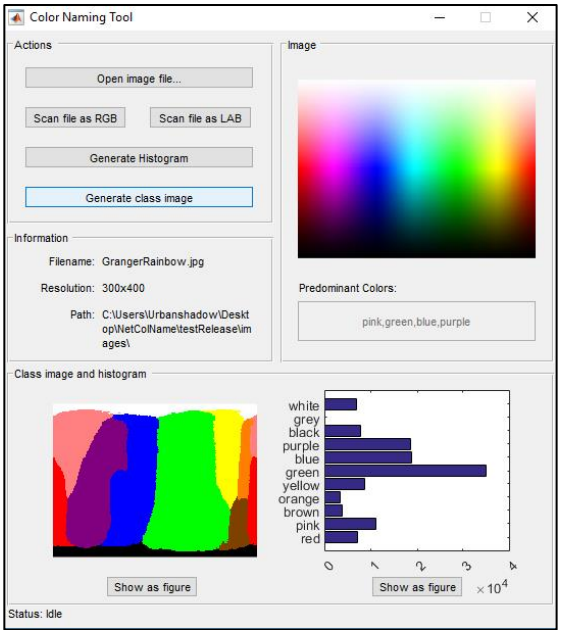

Fig. 8. Interfaz gráfica para la clasificación de imágenes.

Ambos resultados aparecerán en la parte inferior de la interfaz. Cada una tiene su correspondiente botón “Show as figure” que permite generar una figura Matlab para poder exportar o modificar el resultado.

Por último, en la esquina inferior izquierda hay un indicador de estado. Dependiendo del tamaño de las imágenes cargadas, su clasificación puede ser lenta. Para evitar colapsos de peticiones, el indicador de estado mostrará cuando se está trabajando de manera intensiva y cuando la interfaz vuelve a estar disponible para trabajar.

8 Resultados

Para poder evaluar los resultados, se han realizado pruebas de precisión con datos de tres fuentes: del análisis psicofísico realizado por R. Baldrich et al. [1] extendido mediante un modelo en el Centro de Visión por Computador, análisis cualitativos con imágenes naturales y un set de imágenes reales desarrollado por van de Weijer *et al.* [2].

8.1 Datos de análisis psicofísico

Los datos del análisis psicofísico están compuestos por una tonalidad de color, los porcentajes de pertenencia a cada una de las clases de color y la etiqueta real del color. Se han utilizado estos datos para comparar el resultado de las redes neuronales ya entrenadas con la etiqueta real del color. Haciendo uso de estos datos, las redes neuronales han sido capaces de conseguir un 76% de precisión en el espacio de color RGB y 77% de precisión en el espacio de color CIELab, ambos utilizando los sets de datos grandes descritos en el apartado 6.

8.2 Datos cualitativos

Utilizando la funcionalidad de la imagen clasificación podemos obtener información cualitativa del modelo. Esto se consigue etiquetando cada color con un color representativo, altamente saturado. De esta forma podemos ver visualmente qué áreas de la imagen han sido etiquetadas en la clasificación con cada color.

Para poder representar cómo de buena es la clasificación, se ha hecho un etiquetado de una imagen del arcoíris de Granger.

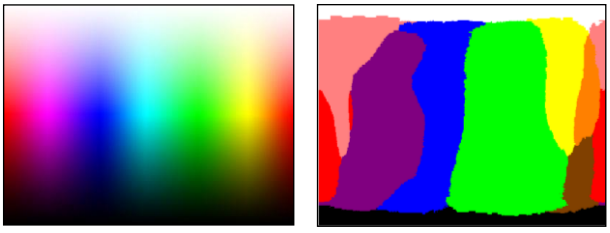

Fig. 9. Clasificación de un arcoíris de Granger. A la izquierda la imagen original, a la derecha la clasificación de la red neuronal en el espacio de color CIELab.

Como etiquetas se han establecido valores representativos en RGB para producir una imagen de salida con la clasificación. La comparación se puede ver en la figura 9. También se han probado diferentes fotografías de diferentes fuentes para probar la clasificación como podemos ver en la figura 10.

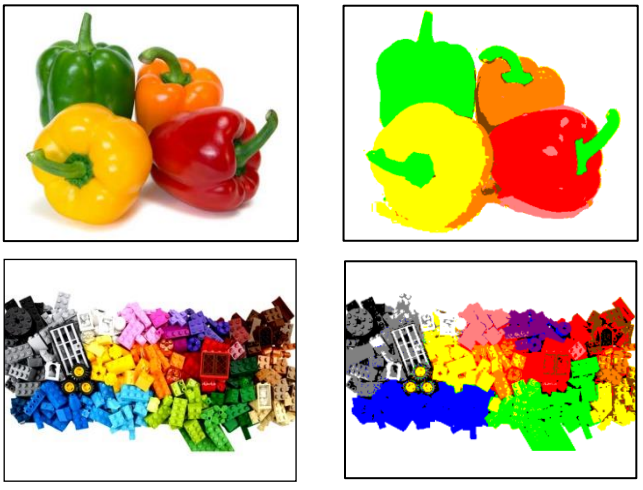

Fig. 10. Clasificación de colores en fotografías. A la izquierda las fotografías originales, a su derecha la clasificación de color.

En el estado actual la red neuronal nos permite encontrar de manera fiable cuales son los colores principales de una imagen cualquiera. Sin embargo, la velocidad a la que se procesa la imagen es un problema. Para poder clasificar cada uno de los píxeles de la imagen hay que pasar por todos ellos, esto hace que en imágenes grandes el proceso sea más costoso.

TABLA I
RELACIÓN DE RESOLUCIÓN CON VELOCIDAD DE CLASIFICACIÓN

Resolución de la imagen	Tiempo en segundos	Píxeles por Segundo (px/s)
300x300	3.1	29000
500x400	6.8	29400
900x563	16.8	30160
900x563 (step = 2)	4.3	29500
2560x1440	120.1	30690
2560x1440 (step = 2)	30.8	29900

Estos datos fueron captados con una máquina equipada con 8 GB de memoria RAM DDR3 a 800 Mhz. Imágenes con step = 2 clasifican la mitad del total de pixels; px = pixel, s = second

La velocidad de procesamiento de imágenes actual la podemos ver en la tabla 1. Mediante el parámetro de paso, podemos reducir drásticamente el número de píxeles a analizar, pero perdemos resolución en la clasificación.

8.3 Pruebas con imágenes reales

Para realizar pruebas con imágenes reales se ha utilizado el conjunto de imágenes de Ebay desarrollado por van de Weijer *et al.* [2] y que permite comprobar cómo de precisa es la clasificación de un color a través de fotografías de objetos concretos y máscaras que definen donde debería haber sido clasificado cierto color. El conjunto de imágenes contiene cuatro categorías de objetos: coches, vestidos, alfarería y calzado. Estos objetos están organizados por categoría y en las mismas once clases de color que han sido entrenadas en la red neuronal.

Cada color tiene 12 imágenes con sus respectivas máscaras para cada uno de los tipos de objeto teniendo así un total de 528 imágenes para probar el modelo. Podemos ver un ejemplo del set de imágenes en la figura 11.

Fig. 11. Ejemplo del set de pruebas de fotografías de Ebay. A la izquierda la imagen a clasificar. A su derecha la máscara correspondiente al color azul

El set contiene cuatro tipos diferentes de objetos y utiliza los mismos once colores de salida que las redes neuronales entrenadas.

Utilizando el mismo método que el utilizado para la comprobación de fotografías, podemos seleccionar uno de los colores y producir una máscara de la clasificación para poderla comparar con la incluida en el set.

La fórmula utilizada para calcular la precisión de la máscara generada es la siguiente:

$$accuracy = \frac{\sum_1^n (M_e \circ M_g)}{\sum_1^n M_e}$$

donde M_e es la imagen máscara compuesta por 1 y 0 que proviene de la base de datos de imágenes, M_g es la imagen máscara generada por la clasificación de la red neuronal también compuesta por 1 y 0 y n es el número de elementos en ambas matrices. La fórmula asume que M_g es del mismo tamaño que M_e y que ambas representan la información de la misma forma. La operación $\circ$ corresponde al producto elemento a elemento de las matrices.

De esta manera, y haciéndonos valer de histogramas, podemos comprobar la precisión que tienen las redes neuronales entrenadas en cada color y en cada tipo de objeto.

TABLA II
RESULTADOS DE LA CLASIFICACIÓN DE IMÁGENES REALES

Espacio de color	Coches	Vestidos	Alfarería	Zapatos
CIELab	51.5%	69.2%	58.3%	69.6%
RGB	51.7%	67.4%	58.1%	70.1%

Los porcentajes muestran la precisión obtenida en la forma de la máscara al compararla utilizando la fórmula superior.

Los resultados de la clasificación por objeto las podemos ver en la tabla 2. Utilizando el mismo método, podemos agrupar los porcentajes de precisión en los diferentes colores para poder ver las diferencias de precisión por color. Podemos ver diferencias entre los modelos de RGB y CIELab en la figura 12.

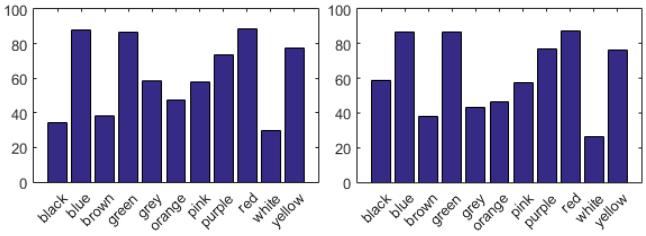

Fig. 12. Precisión de las redes neuronales en el test de imágenes de Ebay ordenadas por color. A la izquierda la precisión obtenida por la red neuronal en el espacio de color RGB, a su derecha la precisión obtenida por la red neuronal en el espacio de color CIELab.

En ambas imágenes podemos observar un patrón similar, donde el color azul, el color verde y el color rojo han obtenido los mejores resultados. Seguido de estos, aparecen el violeta y el amarillo. Comparando ambos resultados podemos ver la diferencia de precisión obtenida en el color negro, obteniendo una mayor precisión en la red neuronal del espacio CIELab; y en el color gris teniendo mayor precisión en el espacio RGB. También destaca la baja precisión del color blanco y del color marrón. Esto se debe a que precisamente los colores blanco, negro y marrón son los colores que tienen menor área de representación dentro de ambos espacios de color y esto hace que los límites de estos colores sean difíciles de detectar y acaben confundiéndose. En total, la clasificación de las redes neuronales para el conjunto de imágenes reales ha sido del 61.8% para RGB y del 62.2% para CIELab. Los resultados de precisión son mayores en CIELab gracias a la uniformidad del espacio, pero la definición de la forma de algunos objetos se ve muy afectada por las reflexiones, especialmente en las imágenes de coches.

9 CONCLUSIONES

Durante el transcurso de este trabajo hemos podido descubrir cómo el problema de la clasificación de color se ha afrontado en los últimos 15 años, hemos analizado las diferencias en el aprendizaje de múltiples arquitecturas de redes neuronales simples. Además, hemos podido generar una serie de funciones y una interfaz gráfica que facilitan el uso de las redes neuronales simples entrenadas para resolver el problema de la clasificación de color. Las redes neuronales entrenadas ofrecen un 76% y un 77% de precisión en la clasificación (para los espacios de color RGB y CIELab, respectivamente), suficiente para realizar tareas como detección de los colores principales de la imagen o asistir a otros sistemas de aprendizaje en su clasificación, todo ello en un tiempo razonable. Pese a que el tiempo de clasificación no es instantáneo, se han incluido formas de contrarrestar la pérdida de velocidad asociada al número de píxeles a clasificar.

Por tanto, considero que se han conseguido los objetivos previstos para este trabajo en el tiempo planificado. A corto plazo, el proyecto podría mejorar proporcionando datos de múltiples fuentes, para así mejorar la generalización de las redes neuronales. Agregar más datos que analizar también supondría aumentar el tiempo de entrenamiento así que se sugiere añadir más potencia de procesamiento a la hora de entrenar. Finalmente, para aumentar el rendimiento de la clasificación se propone trabajar en algoritmos de robustez a iluminación como el realizado por Y. Liu *et al.* [6] o se podrían implementar redes neuronales convolucionales que permitirían tener en cuenta como colores muy diferentes que se encuentran muy juntos distorsionan la percepción de su color, pudiendo así engañar al ojo humano. Actualmente existen librerías para Matlab muy potentes como Matconvnet [10] que permitirían aplicar redes neuronales convolucionales como una extensión del proyecto actual.

AGRADECIMIENTOS

En primer lugar me gustaría agradecer a Robert Benavente Vidal e Ivet Rafegas Fonoll, mis tutores del proyecto, su paciencia y su dedicación que han hecho este trabajo posible. Después me gustaría agradecer a mis padres y a mis suegros todo el cariño y apoyo que me han dado, ya que sin ellos hubiera sido imposible tan siquiera tener la oportunidad de cursar mi titulación o llegar a crear este documento.

Por último, y no por ello menos importante, me gustaría agradecer el trabajo del equipo docente de la Escuela de Ingeniería de la Universidad Autónoma de Barcelona. Sin su dedicación incluso en los tiempos más difíciles sería imposible educar y formar a nuevas generaciones de ingenieros.

BIBLIOGRAFÍA

[1] R. Baldrich, R. Benavente, M. Vanrell, A Dataset For Fuzzy Colour Naming. Color research and application, 31(1): 48-56, 2006.

[2] J. Van De Weijer, C. Schmid, J. Verbeek, and D. Larlus, "Learning color names for real-world applications," Image processing, ieee transactions on, vol. 18, iss. 7, pp. 1512-1523, 2009.

[3] Thomas Hofmann. Probabilistic latent semantic indexing. In proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '99, pages 50{57, New York, NY, USA, 1999. ACM.

[4] Y. Wang, J. Liu, J. Wang, Y. Li,H. Lu, Color names learning using convolutional neural networks, Image Processing (ICIP), 2015 IEEE International Conference, 2015.

[5] Y. LeCun, K. Kavukcuoglu, and C. Farabet. Convolutional Networks and Applications in Vision. International Symposium on Circuits and Systems, pp. 253-256, 2010.

[6] Y. Liu, Z. Yuan, B. Chen, J. Xue and N. Zheng, "Illumination Robust Color Naming via Label Propagation," *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, 2015, pp. 621-629.

[7] K. Levenberg "A Method for the Solution of Certain Non-Linear Problems in Least Squares". Quarterly of Applied Mathematics 2: 164–168, 1944

[8] H. Martin, M. Mohammad. Training feedforward networks with the Marquardt algorithm. IEEE transactions on Neural Networks, 1994, vol. 5, no 6, p. 989-993.

[9] M. Martin Fodslette. A scaled conjugate gradient algorithm for fast supervised learning. Neural networks, 1993, vol. 6, no 4, p. 525-533.

[10] A. Vedaldi, K. Lenc. MatconvNet -- Convolutional Neural Networks for MATLAB. Proceedings of ACM Int. Conf. on Multimedia, 2015

[11] R. Martin, B. Heinrich. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. En Neural Networks, 1993., IEEE International Conference On. IEEE, 1993. p. 586-591.

[12] M.Seaborn, L. Hepplewhite, and J. Stonham. "Fuzzy Colour Category Map for the Measurement of Colour Similarity and Dissimilarity", Pattern Recognition, 38(2):165-177, 2005.

[13] G. Wyszecki, W.S.Stiles. Color science: concepts and methods, quantitative data and formulae. John Wiley & Sons, 2nd edition, 1982.

[14] Y.Bengio, A.Courville, and P.Vincent. Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(8):1798-1828, 2013.

APÉNDICES

A1. DOCUMENTACIÓN DEL PROYECTO

Las funciones para Matlab incluidas con este trabajo son las siguientes:

NeuralNetworkRGB(Pixels)

Clasifica un color en el espacio de color RGB.

Parámetros:

Pixels: n x 3 uint8 array donde n es el número de píxeles diferentes a clasificar y 3 los tres canales en orden R, G, B.

Retorno:

n x 11 double array donde n es el número de colores introducidos para clasificar y 11 los valores de pertenencia a cada una de las 11 clases de color. El orden de los píxeles se conserva con respecto al parámetro de entrada.

NeuralNetworkLAB(Pixels)

Clasifica un color en el espacio de color CIELab.

Parámetros:

Pixels: n x 3 uint8 array donde n es el número de píxeles diferentes a clasificar y 3 los tres canales en orden L, a, b.

Retorno:

n x 11 double array donde n es el número de colores introducidos para clasificar y 11 los valores de pertenencia a cada una de las 11 clases de color. El orden de los píxeles se conserva con respecto al parámetro de entrada.

ColorProcessPixels(Pixels, Cspace = RGB)

Función de ayuda. Clasifica un color en el espacio de color especificado por parámetro.

Parámetros:

Pixels: n x 3 uint8 array donde n es el número de píxeles diferentes a clasificar y 3 los tres canales en orden.

Cspace: string que identifica el espacio de color. Si el string no es RGB, utilizará CIELab. Por defecto RGB.

Retorno:

n x 11 double array donde n es el número de colores introducidos para clasificar y 11 los valores de pertenencia a cada una de las 11 clases de color. El orden de los píxeles se conserva con respecto al parámetro de entrada. La función utilizada para clasificar depende del parámetro Cspace.

ColorProcessImage(Image, Cspace = RGB, step = 1, wsize = 1, convolute = no)

Función de ayuda. Clasifica una imagen. Por defecto la clasificación se realiza para todos los píxeles y en RGB.

Parámetros:

Pixels: n x m x 3 uint8 matrix donde n y m es el ancho y alto de la imagen y 3 los tres canales en orden.

Cspace: string que identifica el espacio de color. Si el string no es RGB, utilizará CIELab. Por defecto RGB.

Step: int que identifica cada cuantos píxeles se escoge uno para clasificar. Step influye horizontal y verticalmente, de forma uniforme. Step = 2 reduce la resolución de la imagen a la mitad. Por defecto a 1 (se analizan todos los píxeles de la imagen).

Wsize: int que define una ventana alrededor de un pixel escogido para realizar la convolución. Debe ser impar ya que un pixel central es necesario. Si se le pasa un número par, cambiará automáticamente al anterior número impar. La ventana hace que no se empiece a clasificar desde el primer pixel, puede reducir el número de píxeles de salida. Por defecto a 1 (no afecta).

Convolute: string que define si se va a realizar o no la convolución. En caso afirmativo genera un kernel de convolución gaussiano de tamaño Wsize. La convolución es aplicada en cada ventana, por canal. El pixel central en la ventana se envía a clasificar después de la convolución. Por defecto a 'no'.

Retorno:

n x m x 11 double array donde n y m es el ancho y alto de la imagen pasada por parámetro y 11 los valores de pertenencia a cada una de las 11 clases de color. El orden de los píxeles se conserva con respecto a la imagen original. La función utilizada para clasificar depende del parámetro Cspace.

ResultToImage(cdata)

Produce una imagen a partir del resultado de la clasificación. Utiliza colores saturados como etiqueta. Escoge el más alto de los valores de pertenencia. En caso de empate, escoge el primero de ellos.

Parámetros:

Cdata: n x m x 11 double matrix con los resultados de una clasificación. n y m definen el tamaño de la imagen de salida.

Retorno:

n x m x 3 uint8 matrix con la información de color de la imagen producida. El retorno de esta función puede ser visualizada con imshow() o guardada a un fichero con la función de Matlab imsave().

ResultToDictionary(cdata)

Produce una matriz de nombres a partir del resultado de la clasificación. Utiliza los nombres de los colores en inglés como etiqueta. Escoge el más alto de los valores de pertenencia. En caso de empate, escoge el primero de ellos.

Parámetros:

Cdata: n x m x 11 double matrix con los resultados de una clasificación. n y m definen el tamaño de la matriz de salida.

Retorno:

n x m cell matrix con el nombre del color correspondiente en inglés a cada pixel de la imagen original.

DictionaryLabelHist(dictionary)

Cuenta la población de nombres de una matriz de nombres producida por ResultToDictionary(cdata). El retorno de esta función puede ser utilizada con la función de Matlab bar() para producir un histograma.

Parámetros:

Dictionary: n x m cell matrix con nombres de colores en inglés producida por ResultToDictionary(cdata).

Retorno:

Data: 1 x 11 int array con el número total de cada etiqueta del diccionario pasado.

Heads: 1 x 11 cell array con el nombre de los colores en el orden devuelto.