Universidade da Coruña

Facultad de Ingeniería Informática

Sistema para clasificar obras de arte según su estilo

Proyecto Aprendizaje Automático

Autores:
David Esteban Martínez, Jorge García Pombo, Samuel Ramos Varela,
Alejandro Esteban Martínez e Iván Pérez Longa

Marzo 2022

Índice

0. Preludio

Nuestro proyecto está compuesto por los apartados siguientes:
Parte 1: Introducción.

Se describe el problema a resolver y se abordan temas como la importancia del mismo, las ventajas de su resolución y los objetivos del trabajo.

Parte 2: Descripción del problema, restricciones y Base de Datos.

Contiene un análisis más detallado del problema a resolver, dando detalles como las posibles limitaciones, base de datos utilizada o herramientas empleadas.

Parte 3: Análisis Bibliográfico.

Engloba estudios relacionados con el nuestro bien por su similitud en el tema como en las herramientas o metodologías empleadas.

Parte 4: Desarrollo.

Se centra en nuestra solución propuesta para el problema descrito. Incluye el preprocesado de los datos y nuestras decisiones de implementación.

1. Introducción

El mercado del arte mueve 1890 millones de euros, el doble que hace 10 años. Aparte del obvio valor económico en el que se tasan estas obras también tienen inherentemente una importancia cultural incalculable. Teniendo esto en mente, se estima una pérdida de 600.000 piezas entre 1933 y 1945. En tan solo 12 años el patrimonio cultural desaparecido es inmenso. A lo largo de nuestra historia incidentes como este se han repetido hasta la saciedad, desde la destrucción de Cartago en 149 a.C hasta el incendio del Museo Nacional de Brasil en 2018. Así pues, para evitar en la medida de lo posible el echarnos las manos a la cabeza con otra desgracia como estas, creemos que sería muy útil contar con una base de datos global de arte.

Nuestro proyecto tiene como finalidad, conseguir diferenciar los estilos artísticos y ser capaces de identificarlos en piezas concretas, para posteriormente poder clasificarlas según estos. Esta taxonomía podría ser de gran ayuda para agilizar sistemas de digitalización y almacenamiento de arte digital, ya sean de entidades concretas (museos o similares) o bases de datos de carácter general como la que mencionamos anteriormente. En conjuntos pequeños el impacto no sería muy relevante, pero en agrupaciones de mayor número contar con un programa capaz de diferenciar las corrientes artísticas ahorraría mucho tiempo en el filtrado de obras. Esto será de gran utilidad para profesionales relacionados con el mundo del arte. Por ejemplo, podría ayudar a investigar sobre una determinada época, a analizar las características de una obra o a proporcionar un mayor numero de recursos y ejemplos para la docencia. Además de esto el sistema podría ayudar a implementar otros que, profundizando en el análisis de características de los estilos, recomendasen obras para aficionados del arte, o incluso que aplicasen estilos a otras imágenes u obras.

Esta memoria esta estructurada siguiendo los baremos y recomendaciones de la asignatura. En primer lugar, una introducción para poner en contexto y dar una idea general del proyecto y de las ventajas que tendría su implementación. Posteriormente una descripción más detallada de la cuestión que nos atañe, junto con las restricciones del problema y la información de la base de datos que hemos construido, las propiedades de los datos y las condiciones en las que la hemos elaborado. En tercer lugar un análisis bibliográfico en el que se detallan otros estudios que ya han intentado resolver el problema que tenemos entre manos. Para finalizar, el desarrollo, donde nos centramos en nuestra manera de abordar la creación del sistema y las decisiones de diseño que hemos tenido que tomar.

Haremos uso de RR.NN.AA (Redes Neuronales Artificiales) [AJO+19], un modelo computacional basado en el funcionamiento de las neuronas humanas, que desde su concepción se ha utilizado para el reconocimiento de patrones. Probaremos diferentes arquitecturas, jugando con el número de neuronas y capas ocultas.

Probaremos también diferentes kernels y parámetros en SVM (Máquinas de Vectores de Soporte), una serie de técnicas y algoritmos de aprendizaje supervisado con muy buenos resultados en problemas de regresión, y sobre todo clasificación. En el siguiente articulo se discute la eficacia de las SVM en la clasificación de imágenes [AVTS11].

Muchas tecnologías de machine learning hoy en día no son especialmente buenas en la evaluación de que características son necesarias a partir de los datos sin estructurar (en nuestro caso imágenes), derivando en que los algoritmos puedan ser sensibles a datos sin importancia. Es por esto que los Árbol de decisión son necesarios en el procesado de imágenes para la extracción de características. Probaremos diferentes valores de profundidad para lograr los mejores resultados posibles. Aquí [LCD17] se hace una comparación de los Árbol de decisión (en concreto el algoritmo C4.5, visto en clase), con algunas de las otras técnicas ya mencionadas, como kNN o MLP Mixer.

También experimentaremos con kNN (k-Nearest Neighbours), un método de Aprendizaje Vago y supervisado basado en la disposición espacial de los elementos de diferentes clases con respecto al elemento que se quiere clasificar, calculando que clase es la más adecuada según los k elementos mas cercanos. Este método ve su mayor eficacia en problemas de clasificación, y para lograr la mejor solución posible probaremos distintos valores de k. En el siguiente artículo se explora la selección de características a través de kNN para la clasificación de imágenes [ZCW+20].

En la última aproximación se utilizará Deep Learning, el modelo elegido será Redes Neuronales Convolucionales (CNN), un modelo de gran utilidad en visión artificial que logra combinar la extracción de características y la clasificación a través de una mayor profundidad de capas y filtrando las imágenes originales para extraer tambien las características espaciales, de gran importancia en la clasificación de imágenes. Experimentaremos con diferentes arquitecturas de CNNs. Podemos ver aquí [WFW21] una comparativa entre técnicas de deep learning (concretamente CNN) y otras técnicas de machine learning más tradicionales, como SVM; en el campo de la clasificación de imágenes y evaluando la precisión de cada modelo.

2. Descripción del problema, restricciones y Base de Datos

2.1. Descripción del problema

El problema al que nos enfrentamos es la clasificación de obras de arte, concretamente cuadros, con diferentes estilos de pintura. Empezaremos con dos estilos fácilmente diferenciables como son el Pop Art y el Barroco para posteriormente incluir estilos más "ambiguos" para nuestro sistema. Para esto desarrollaremos y probaremos diferentes técnicas en cada aproximación. Nos basamos en diversas publicaciones acerca de las tecnologías mas utilizadas en la clasificación de imágenes para informarnos sobre los distintos algoritmos y modelos que emplearemos [LW07]. Nuestro procedimiento para medir la calidad del funcionamiento será comparar las asignaciones de nuestro sistema con los estilos ya fijados convencionalmente, expresando la tasa de acierto en un porcentaje de sencilla comprensión. Nuestro metrica objetivo es, entonces, el porcentaje de acierto en la clasificación de las obras.

2.2. Restricciones

Para que la tarea se pueda desarrollar satisfactoriamente, tendremos que especificar una serie de restricciones sobre las imágenes que utilizaremos.

- Las imágenes no deben incluir ningún margen o marco
- Deben tener un tamaño mínimo que permita distinguir las características de la obra (perdiendo la mínima calidad posible con respecto al original)
- La obra de la imagen no debe estar distorsionada, rotada o bajo el efecto de un filtro
- En caso de ser fotografías, tomadas en condiciones de luz neutra, sin ningún tipo de reflejo ni destello sobre la obra
- Pueden tener tamaños y orientaciones dispares

Utilizaremos imágenes que pensamos que pueden representar bien cada estilo, y dentro de éste abarcar la mayor variedad posible, pero siempre manteniendo las características en común que delimitan las dichas corrientes artísticas.

2.3. Base de Datos

En la primera aproximación distinguiremos entre los estilos Barroco y Pop Art, con alrededor de 80 imágenes de composiciones de cada uno. En las consiguientes iteraciones se podrían añadir más corrientes junto a las piezas que los representan.

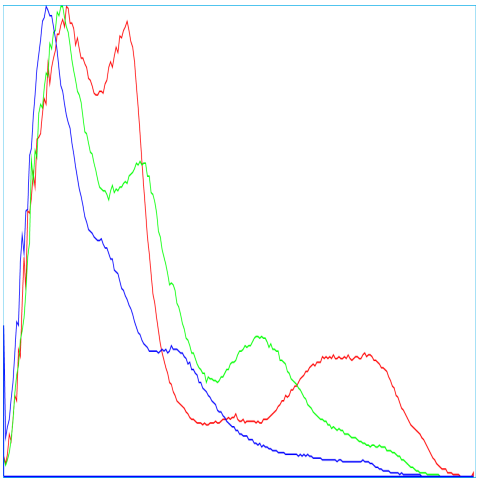
Nuestra base de datos ha sido creada a partir de imágenes de Internet. Al tratarse de obras de arte ha sido sencillo encontrar un gran número de ellas a una calidad aceptable y que cumplan las restricciones anteriormente explicadas.

Las obras de Pop Art se encuentran en las siguientes direcciones web: WGA (`https://www.wga.hu`) y WikiArt (`https://www.wikiart.org/`)
La página Web Gallery of Art es un museo virtual creado en 1996 con el objetivo de facilitar un uso eficiente de internet para la educación visual. Sus imágenes tienen un formato JPG con compresión IJG 67, tamaño habitual de 150-300 kbyte, 900-1400 pixel.
WikiArt recoge obras de arte de museos, universidades, ayuntamientos y otras instituciones de más de 100 países; la mayoría de estas fuera de la vista del público

Los cuadros barrocos son de la siguiente página: Ranker (`https://www.ranker.com/list/baroque-paintings/reference`) Esta es una página web que almacena gran cantidad de imagenes de distintos temas, y permite a los usuarios acceder a ellas y valorarlas.

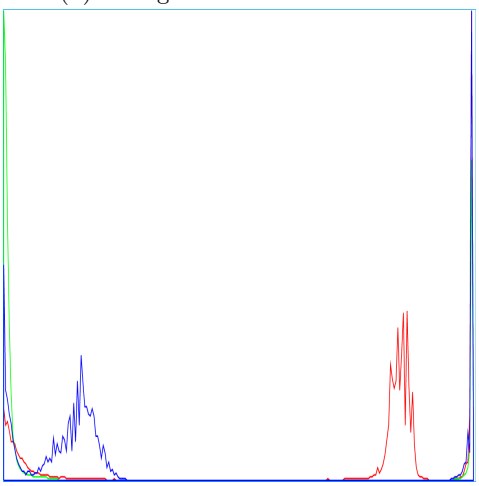
A la hora de buscar características diferenciadoras, nos hemos centrado en diversas propiedades. Con respecto a los colores destacamos: la cantidad de ellos, pues las piezas Barrocas suelen tener una paleta de colores mayor; la predominancia de tonos oscuros, ya que las obras Pop Art son más claras generalmente; la intensidad, pues el Pop Art suele tener más potencia; la disparidad en la escala cromática, ya que el Barroco tiene más colores pero el Pop Art más distintos entre sí y la similitud entre los colores contiguos en el cuadro. Por otra parte, también trabajaremos con las formas geométricas apreciables, pues el Pop Art suele presentar formas angulosas.

(a) Ejemplo de cuadro Barroco.



(b) Histograma RGB del Barroco

(c) Ejemplo de cuadro Pop Art.



(d) Histograma RGB del Pop Art

3. Análisis Bibliográfico

Numerosos estudios intentan resolver el problema de clasificar pinturas según sus estilos de forma automática, usando diversas técnicas para ello. Por ello, hemos encontrado varios artículos que consideramos interesantes.

Por ejemplo, en [INS+21] se utilizaron ViT y MLP Mixer, dos arquitecturas diferentes de Deep Learning, alcanzando un 39 % de precisión con una base de datos de 21 estilos diferentes. A pesar del bajo porcentaje de precisión, este sube bastante dependiendo del estilo y la técnica, con casi un 70 por ciento de precisión en el estilo japonés Ukiyo-e usando ViT.

Un estudio anterior, [LNY17], tiene más éxito, llegando al 63 % de precisión con 25 estilos, usando para ello, además de Deep Learning, una Red Neuronal Residual y varios reentrenos hasta llegar al resultado.

En estudios como [CG16] se han realizado experimentos extrayendo una gran variedad de caracteristicas de las imagenes, comprobando cuales permiten obtener los mejores resultados. Es muy interesante como analizan las sinergias entre características. Finalmente obtuvieron un 77.57 % de acierto al clasificar en 7 estilos.

Además de artículos de investigación, también hemos encontrado otras alternativas, como la creación de una aplicación de movil (`https://medium.com/swlh/art-detective-ai-for-art-style-recognition-4632e05c3496`) en [All20] capaz de llevar a cabo la distinción de 5 estilos distintos de pintura y presentarlo de forma sencilla al usuario.

Volviendo al empleo de Deep Learning, se estudia uso el uso de distintas configuraciones de CNNs, en [Alk21] llegando a una precisión del 42 %.

También encontramos estudios en los que se prueban varias técnologias distintas, como en [Oom18]. Donde emplean 82.000 imágenes pertenecientes a 25 estilos y obtienen resultados que rondan el 50 %. En otros como [HSSD17], se intentan reconocer el estilo de ilustradores específicos. Lo cual suena aún más complicado a primera vista, pero aún así se logra alcanzar un 90 % de acierto con deep learning.

Un caso un tanto distinto pero que también encontramos interesante es el sistema mencionado en [PHO19]. El objetivo del sistema es encontrar plagios de obras. Aunque no trata la clasificación como tal, las características que emplea para el reconocimiento de obras sí pueden sernos útiles. El sistema crea un mapa de calor para estudiar el trazo de las pinceladas.

Varios de los artículos hacen la observación de que una mayor base de datos sería beneficiosa, por lo que en nuestro caso este déficit de patrones será aún más marcado.

Concluir que, visto lo visto, el problema que tenemos entre manos es cuanto menos complicado, pero con una dificultad fácilmente escalable con el número de estilos que se intentan clasificar. Es obvio que cuantas más corrientes artísticas, más le costará a nuestro sistema diferenciar correctamente las obras, y mejor tendrán que ser las características de información con las que trabajemos. Aun así y vistas nuestras limitaciones tanto de tiempo como de conocimiento, esperamos obtener resultados decentes en las primeras aproximaciones con pocos estilos, lo que nos permitirá ir complicando el problema poco a poco, no hasta el punto de estos estudios que trabajaban con muchos estilos, pero sí con una diversidad artística un poco más modesta.

4. Desarrollo

Después de trabajar en los tutoriales de clase para tener un mayor conocimiento de Julia y sentirnos más cómodos en nuestro entorno de trabajo, empezamos el desarrollo de nuestro sistema de clasificación de arte. En los apartados siguientes se suceden las iteraciones que fuimos desarrollando, divididas en tres secciones, una pequeña descripción introductoria dónde explicamos los objetivos y novedades de la aproximación, un apartado de resultados dónde se exponen los datos recopilados y una discusión final sobre el estado y rumbo del proyecto. Cabe destacar que durante todo el desarrollo utilizamos Cross-validation para minimizar el azar en la partición de los conjuntos de datos.

4.1. Aproximación 1:

4.1.1. Descripción:

Para una primera solución buscaremos extraer las características necesarias de los cuadros para poder diferenciar entre Barroco y Pop Art.

Escogemos estos dos estilos por sus marcadas diferencias tanto en trazo como en color y forma. Posteriormente buscaremos ampliar el abanico de estilos pero para empezar estos son los más antagónicos.

(a) Ejemplo de Pop Art.

(b) Ejemplo de Barroco.

En un primer lugar comenzamos analizando la intensidad de los colores rojo, verde y azul, pues el Pop Art presenta colores mucho más saturados que su contraparte Barroca, con tonalidades más apagadas. Para esto, calculamos la media de la cantidad de cada uno de estos colores en cada imagen, y sumamos estas medias, dando los siguientes resultados:

	Rojo	**Azul**	**Verde**
Pop Art	48.336433	38.02643	43.178528
Barroco	28.400236	18.272198	22.592474

Cuadro 1: Resultados de las medias acumuladas.

Teniendo la misma cantidad de imágenes en cada uno de los dos estilos, 86, podemos ver que las pinturas del estilo Pop Art presenta colores más intensos, con una media acumulada mayor en los tres colores que en el estilo Barroco.

Por esto decidimos que es una buena aproximación para distinguir entre estos dos estilos empezar por la intensidad de los colores.

4.1.2. Resultados:

Comenzamos probando diferentes arquitecturas de Redes Neuronales Artificiales. Suponemos que los resultados obtenidos son muy bajos pues nuestro problema no es linealmente separable (lo justificamos más adelante).

		AVG	Desviación Tipica
RR.NN.AA **[1]**	**Accuracy (%)**	51.8750	11.8289
	F1 (%)	67.6115	10.0276
RR.NN.AA **[2]**	**Accuracy (%)**	52.0955	8.6291
	F1 (%)	68.1096	7.7336
RR.NN.AA **[10]**	**Accuracy (%)**	51.9117	14.5537
	F1 (%)	67.2188	13.1030
RR.NN.AA **[315]**	**Accuracy (%)**	52.3854	9.9601
	F1 (%)	61.755	12.445
RR.NN.AA **[1,1]**	**Accuracy (%)**	51.3857	15.2412
	F1 (%)	66.5326	13.7325
RR.NN.AA **[2,2]**	**Accuracy (%)**	51.8749	13.9576
	F1 (%)	67.0508	13.4646
RR.NN.AA **[4,3]**	**Accuracy (%)**	52.0588	10.0148
	F1 (%)	67.9642	8.5804
RR.NN.AA **[12,16]**	**Accuracy (%)**	52.2058	13.4703
	F1 (%)	67.6539	11.9218

Cuadro 2: Resultados de RR.NN.AA con k-fold

Podemos ver que aún cambiando la arquitectura, el porcentaje de acierto al clasificar cuadros ronda el 50 %, es decir, clasifica de manera aleatoria.

En las Máquinas de Vectores de Soporte (SVM), no se aprecia una variación destacable con el cambio de la variable C, como se puede ver en la tabla a continuación. Variamos tambien la Radial Basis Function por Lineal kernel, aunque se consiguen resultados similares.
Podemos ver que los resultados rondan el 87 % en todos los casos probados, con desviacion típica cerca del 8 %, valores más que aceptables para nuestro problema.

		AVG	Desviación Tipica
SVM(C 1)	**Accuracy (%)**	87.2426	9.3492
	F1 (%)	87.2738	9.4831
SVM(C 2)	**Accuracy (%)**	87.8676	4.9203
	F1 (%)	88.2792	4.9873
SVM(C 3)	**Accuracy (%)**	87.2794	8.9035
	F1 (%)	87.2025	8.3115
SVM(C 6)	**Accuracy (%)**	86.6544	9.4116
	F1 (%)	86.9702	8.9050
SVM(C 10)	**Accuracy (%)**	87.9044	8.5914
	F1 (%)	89.3417	7.0422
SVM(C 25)	**Accuracy (%)**	86.771	6.774
	F1 (%)	87.412	9.123
SVM(C 1), kernel lineal	**Accuracy (%)**	86.5073	11.7685
	F1 (%)	87.3265	11.1376
SVM(C 5), kernel lineal	**Accuracy (%)**	86.5808	6.9954
	F1 (%)	87.2847	6.6340

Cuadro 3: Resultados de SVM

Con respecto a Árbol de decisión, los resultados no tienen apenas variación con el cambio de la

profundidad máxima, como se puede ver en la tabla a continuación. Así pues se consigue accuracy y F1 que ronda el 84 % con un 7 % de desviación típica.

		AVG	Desviación Tipica
Árbol de decisión(mD 2)	**Accuracy (%)**	81.874	6.3558
	F1 (%)	85.774	6.211
Árbol de decisión(mD 4)	**Accuracy (%)**	83.0882	6.7145
	F1 (%)	84.1001	6.3784
Árbol de decisión(mD 5)	**Accuracy (%)**	84.1911	8.2500
	F1 (%)	84.0103	8.4983
Árbol de decisión(mD 6)	**Accuracy (%)**	85.5147	4.8810
	F1 (%)	85.3469	7.4832
Árbol de decisión(mD 10)	**Accuracy (%)**	84.2279	9.4314
	F1 (%)	85.1469	8.9726
Árbol de decisión(mD 13)	**Accuracy (%)**	85.112	6.224
	F1 (%)	84.239	3.995
Árbol de decisión(mD 25)	**Accuracy (%)**	83.514	5.220
	F1 (%)	82.557	5.447
Árbol de decisión(mD 147)	**Accuracy (%)**	85.399	4.802
	F1 (%)	85.778	6.225

Cuadro 4: Resultados de con k-fold

Centrándonos en k-Nearest Neighbours (kNN), los datos siguen sin variar apenas al cambiar el número de vecinos, como se muestra en la tabla a continuación. Así pues se consigue accuracy y F1 que ronda el 85 % con un 7 % de desviación típica. Resultados bastante similares a los obtenidos con el Árbol de decisión.

		AVG	Desviación Tipica
kNN (k = 3)	**Accuracy (%)**	86.6176	6.9674
	F1 (%)	87.3234	7.2434
kNN (k = 4)	**Accuracy (%)**	86.7647	8.1271
	F1 (%)	87.7687	8.3075
kNN (k = 7)	**Accuracy (%)**	86.5441	12.0749
	F1 (%)	88.2177	10.0262
kNN (k = 10)	**Accuracy (%)**	82.4632	8.1806
	F1 (%)	83.3870	9.1468
kNN (k = 20)	**Accuracy (%)**	87.354	11.457
	F1 (%)	86.5357	9.741
kNN (k = 100)	**Accuracy (%)**	85.941	9.665
	F1 (%)	87.227	11.452

Cuadro 5: Resultados de kNN con k-fold

4.1.3. Discusión:

Podemos concluir que en los dos estilos utilizados, Barroco y PopArt, la cantidad de rojo, azul y verde nos sirve para poder clasificar de forma satisfactoria los distintos cuadros. Esto se puede ver en el apartado de resultados, los cuales rondan el 85 % en todas las técnicas menos en RR.NN.AA.

Como comentábamos al inicio de esta aproximación, la gran diversidad de subestilos que hay dentro del PopArt, así como la técnica propia de cada artista; provocan que aunque no obtuviésemos una precisión perfecta, podamos considerar muy buenos los porcentajes obtenidos.

De todas maneras, los resultados se ven muy favorecidos por el hecho de que solo estamos trabajando con dos estilos y que existen numerosas diferencias entre ellos.

El principal problema que nos hemos encontrado ha sido con la técnica de RR.NN.AA, en la que la accuracy ronda el 50 %. Esto significa que está funcionando de forma aleatoria, así que intentaremos solucionar este tema para la siguiente iteración.

Probamos a entrenar una SVM con un valor de C = 10 000 000, y un kernel lineal, para intentar descubrir si el problema era linealmente separable. Tras registrar una media de accuracy = 86.8014 y de F1 = 84.3172 (muy similar a las demas pruebas de SVMs) podemos constatar que el problema no es linealmente separable.

Otra cuestión que nos ha llamado la atención es el hecho de que las precisiones apenas varían cuando cambiamos la arquitectura de cada técnica. Intentaremos averigüar el por qué para solucionarlo. Suponemos que al ser un problema no lineal, las tecnicas de machine learning tradicional no consiguen mejorar a pesar de aumentar los parametros.

De cara a la siguiente aproximación valoramos añadir un nuevo estilo que no esté tan drásticamente diferenciado de los dos anteriores. De este modo veremos que tal se comporta nuestro sistema y si tenemos que buscar una nueva característica diferenciadora.

4.2. Aproximación 2:

4.2.1. Descripción:

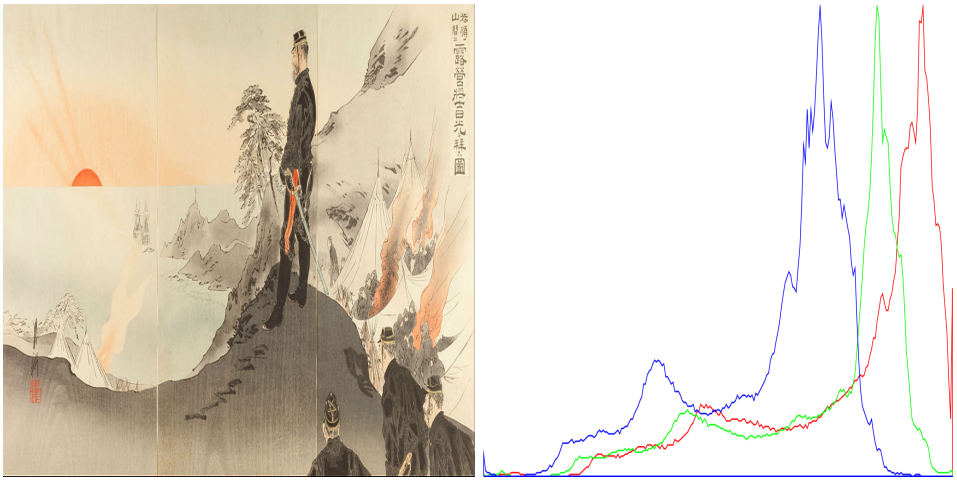
Como mencionabamos al final de la primera aproximación, para esta segunda añadiremos un nuevo estilo artístico a nuestro sistema. Este estilo es el ukiyo-e, de procedencia japonesa.

	Rojo	**Azul**	**Verde**
Pop Art	44.173706	34.940853	39.701782
Barroco	28.361208	18.263456	22.603714
Ukiyo-e	58.06354	42.358467	51.434998

Cuadro 6: Resultados de las medias acumuladas.

Lo hemos elegido porque lo consideramos bastante distinto tanto del barroco como del pop Art. Sin embargo, al contar con tres estilos las diferencias ya no son tan marcadas como antes y debemos trabajar más en las características diferenciadoras.

Es por esto que no nos sorprendería que los resultados empeorasen ligeramente.



(a) Ejemplo de Ukiyo-e.

(b) Histograma RGB del Ukiyo-e

4.2.2. Resultados:

Comenzamos probando diferentes arquitecturas de Redes Neuronales Artificiales.

		AVG	Desviación Tipica
RR.NN.AA **[1]**	**Accuracy (%)**	49.6116	7.8376
	F1 (%)	41.4293	10.2061
RR.NN.AA **[2]**	**Accuracy (%)**	59.4933	7.3390
	F1 (%)	54.4159	9.9374
RR.NN.AA **[10]**	**Accuracy (%)**	64.2983	8.3132
	F1 (%)	61.4544	9.3290
RR.NN.AA **[1,1]**	**Accuracy (%)**	38.5183	5.8634
	F1 (%)	25.0939	7.2033
RR.NN.AA **[315]**	**Accuracy (%)**	57.241	8.962
	F1 (%)	57.542	7.852
RR.NN.AA **[1,1]**	**Accuracy (%)**	38.5183	5.8634
	F1 (%)	25.0939	7.2033
RR.NN.AA **[1,2]**	**Accuracy (%)**	44.3787	7.8967
	F1 (%)	39.5475	8.0237
RR.NN.AA **[2,2]**	**Accuracy (%)**	49.0983	7.9307
	F1 (%)	39.7552	8.4847
RR.NN.AA **[4,3]**	**Accuracy (%)**	54.0883	8.9158
	F1 (%)	47.8416	9.7763
RR.NN.AA **[12,16]**	**Accuracy (%)**	58.5833	10.2443
	F1 (%)	53.3629	11.1710

Cuadro 7: Resultados de RR.NN.AA con k-fold

Todas las arquitecturas probadas tienen resultados similares , con accuracies de 40-60 % y desviaciones de cerca de 8-10. Esto quiere decir que los resultados son practicamente aleatorios, muy parecidos a los de la aproximacion anterior.

Cabe mencionar la arquitectura [10], que presenta los mejores resultados, llegando al 64 % de acierto, y manteniéndose cerca de esos valores con sucesivas ejecuciones del código; no es un resultado tan bueno como con las demás técnicas, pero si se empieza a acercar.

Las SVM muestran una mejoria sustancial con respecto a las RNAs, logrando una accuracy cercana a 70 % consistentemente. De todas formas no podemos observar ninguna mejoria por mucho que variemos el valor de C. Se aprecia también una pérdida de calidad con respecto a la anterior aproximación, debido a la adición del nuevo estilo Ukiyo-e.

		AVG	Desviación Tipica
SVM(C 1)	**Accuracy (%)**	71.85	8.8067
	F1 (%)	71.7400	9.0087
SVM(C 2)	**Accuracy (%)**	71.85	8.6024
	F1 (%)	71.9569	7.9888
SVM(C 3)	**Accuracy (%)**	71.1000	12.8620
	F1 (%)	70.7443	13.8483
SVM(C 6)	**Accuracy (%)**	70.2333	8.0799
	F1 (%)	70.5328	7.6333
SVM(C 10)	**Accuracy (%)**	71.0166	10.2651
	F1 (%)	71.7400	10.6487
SVM(C 25)	**Accuracy (%)**	71.0666	8.0258
	F1 (%)	71.3686	7.4411
SVM(C 1), kernel lineal	**Accuracy (%)**	71.8678	11.7685
	F1 (%)	70.3564	9.0082
SVM(C 5), kernel lineal	**Accuracy (%)**	71.0377	8.5747
	F1 (%)	71.8762	7.9923

Cuadro 8: Resultados de SVM con k-fold

En Árbol de decisión, los resultados no tienen apenas variación con el cambio de la profundidad máxima, como se puede ver en las tablas a continuación. La accuracies y F1 rondan el 65-70 %, y la desviacion tipica cerca de 8-10. Los resultados empeoraron con respecto a la anterior aproximación, pero era lo esperado, ya que hemos añadido una nueva clase para la clasificacion y aun no hemos sacado nuevas características.

		AVG	**Desviación Tipica**
Árbol de decisión(mD 4)	**Accuracy (%)**	69.8999	8.4122
	F1 (%)	69.5636	8.6152
Árbol de decisión(mD 5)	**Accuracy (%)**	68.3166	9.4026
	F1 (%)	69.1309	8.8542
Árbol de decisión(mD 6)	**Accuracy (%)**	66.25	9.8861
	F1 (%)	64.8535	10.7109
Árbol de decisión(mD 10)	**Accuracy (%)**	66.65	8.7179
	F1 (%)	66.4651	9.3219
Árbol de decisión(mD 13)	**Accuracy (%)**	69.9166	8.1646
	F1 (%)	69.7440	8.5725
Árbol de decisión(mD 25)	**Accuracy (%)**	67.0666	8.5962
	F1 (%)	66.9203	8.8033
Árbol de decisión(mD 147)	**Accuracy (%)**	69.48333	9.0555
	F1 (%)	68.8667	9.8871

Cuadro 9: Resultados de con k-fold

Centrándonos en kNN, los datos siguen sin variar apenas al cambiar el número de vecinos, como se muestra en las tablas a continuación. Para todos los valores de k se muestran accuracies y F1 positivas, rondando del 70 % a 75 %, y con una desviacion tipica de 8 a 11 %, cabe decir que el valor de k que mejor resultados da es 7, con una acc = 73 y una desviación típica de apenas 7 %. Con respecto a la anterior aproximación los resultados han empeorado, pero era de esperar debido a la adición del nuevo estilo Ukiyo-e.

		AVG	**Desviación Tipica**
kNN (k = 3)	**Accuracy (%)**	70.66667	8.2910
	F1 (%)	70.0441	8.3603
kNN (k = 4)	**Accuracy (%)**	70.3	6.2191
	F1 (%)	70.0364	6.32577
kNN (k = 7)	**Accuracy (%)**	73.1333	6.9570
	F1 (%)	72.6649	7.1285
kNN (k = 10)	**Accuracy (%)**	73.083	6.579
	F1 (%)	72.484	7.0226
kNN (k = 20)	**Accuracy (%)**	72.3166	9.9908
	F1 (%)	71.2187	10.5057
kNN (k = 100)	**Accuracy (%)**	70.6	11.0775
	F1 (%)	70.1687	11.6352

Cuadro 10: Resultados de kNN con k-fold

4.2.3. Discusión:

En esta aproximación añadimos un tercer estilo, Ukiyo-e, de origen japonés que se caracteriza por colores vivos pero menos potentes que los del PopArt y con lineas marcadas con contornos gruesos.

Los resultados de esta aproximación son globalmente peores que en la anterior, ya que Ukiyo-e es un estilo que por las características que tiene puede enmarcarse en medio entre barroco y Pop Art, disminuyendo considerablemente la precisión con la mayoría de técnicas utilizadas.
Además debemos tener en cuenta que al tener 3 clases, los significados de los resultados ya no son los mismos. En la aproximación 1, un resultado del 50 % implicaría que el sistema tiene la misma eficacia

que una clasificación aleatoria. Sin embargo, ahora ese valor estará en el 33.3 %

Sin embargo sorprende el resultado con las RNA, en las que al usar arquitecturas como [2] o [10] vemos como los resultados mejoran ligeramente, dejando de ser 50 % como en la primera aproximación. Estos resultados siguen sin ser buenos, pero representan una mejora que no se ve en el resto de técnicas.

Por esto, en la siguiente aproximación trabajaremos en añadir una nueva característica que ayude a diferenciar el contraste de brillo entre las imágenes, usando la cantidad de gris en la imagen transformada a escala de grises. Anticipamos que esta característica debería ayudar a aumentar ligeramente la precisión.

4.3. Aproximación 3:

4.3.1. Descripción:

Siguiendo donde nos quedamos al final de la anterior aproximación, hemos decidido extraer una nueva característica de las imágenes que componen la base de datos, la cantidad de gris que presentan las fotos una vez pasadas a escala de grises. Vamos a utilizar pues la media de color gris de cada imagen una vez aplicado este filtro para intentar generalizar más.

	Gris
Pop Art	40.4968
Barroco	23.8388
Ukiyo-e	52.3843

Cuadro 11: Resultados de las medias acumuladas.

Consideramos esto para intentar mejorar los resultados, pues el nuevo estilo presenta cantidades de color similares a las del estilo Barroco.

4.3.2. Resultados:

Comenzamos probando diferentes arquitecturas de Redes Neuronales Artificiales. Usamos las mismas arquitecturas que en las anteriores aproximaciones para ser rigurosos en la experimentación

		AVG	**Desviación Tipica**
RR.NN.AA **[1]**	**Accuracy (%)**	50.865	10.3956
	F1 (%)	42.86	12.959
RR.NN.AA **[2]**	**Accuracy (%)**	59.3933	10.6149
	F1 (%)	54.8739	10.614
RR.NN.AA **[10]**	**Accuracy (%)**	62.92	7.1649
	F1 (%)	60.3824	8.7814
RR.NN.AA **[315]**	**Accuracy (%)**	52.2245	7.4151
	F1 (%)	48.326	5.124
RR.NN.AA **[1,1]**	**Accuracy (%)**	41.88	6.7857
	F1 (%)	30.575	8.478
RR.NN.AA **[2,2]**	**Accuracy (%)**	5.1300	4.84355
	F1 (%)	43.565	6.1563
RR.NN.AA **[4,3]**	**Accuracy (%)**	48.48000	8.6117
	F1 (%)	41.269	10.62666
RR.NN.AA **[12,16]**	**Accuracy (%)**	57.6083	5.1201
	F1 (%)	52.127	7.430

Cuadro 12: Resultados de RR.NN.AA con k-fold

Podemos ver que los resultados son similares si no idénticos a la aproximación anterior usando RR.NN.AA.

En las SVM, se repite esto, los resultados son tan cercanos a los de la anterior aproximación que podrían decirse que son los mismos.

		AVG	Desviación Tipica
SVM(C 1)	**Accuracy (%)**	71.1	8.54337
	F1 (%)	70.4613	8.964
SVM(C 2)	**Accuracy (%)**	72.6666	9.6609
	F1 (%)	72.342	10.0816
SVM(C 3)	**Accuracy (%)**	71.9333	7.4266
	F1 (%)	71.5625	7.475
SVM(C 6)	**Accuracy (%)**	71.51666	9.049
	F1 (%)	71.255	9.5877
SVM(C 10)	**Accuracy (%)**	72.2666	8.8175
	F1 (%)	71.751	9.0799
SVM(C 25)	**Accuracy (%)**	71.4999	5.4006
	F1 (%)	70.9408	6.12776
SVM(C 1), kernel lineal	**Accuracy (%)**	70.2728	9.9856
	F1 (%)	72.9947	9.0051
SVM(C 5), kernel lineal	**Accuracy (%)**	71.6882	7.2928
	F1 (%)	71.2244	8.7492

Cuadro 13: Resultados de SVM con k-fold

Con respecto a Árbol de decisión, se mantiene esta tendencia a la escasa variación en los resultados.

		AVG	Desviación Tipica
Árbol de decisión(mD 4)	**Accuracy (%)**	70.2499	8.8168
	F1 (%)	70.5615	9.1280
Árbol de decisión(mD 5)	**Accuracy (%)**	73.1	7.9784
	F1 (%)	71.8543	9.9934
Árbol de decisión(mD 6)	**Accuracy (%)**	65.4166	9.1203
	F1 (%)	65.0286	9.1506
Árbol de decisión(mD 10)	**Accuracy (%)**	71.05	10.4201
	F1 (%)	70.8583	10.3029
Árbol de decisión(mD 13)	**Accuracy (%)**	67.8666	7.0679
	F1 (%)	67.0859	7.3399
Árbol de decisión(mD 25)	**Accuracy (%)**	67.85	6.0921
	F1 (%)	67.5576	5.7485
Árbol de decisión(mD 147)	**Accuracy (%)**	66.2666	5.9975
	F1 (%)	65.9754	6.1759

Cuadro 14: Resultados de con k-fold

Centrándonos en kNN, confirmamos que el que los resultados no varien con respecto a los anteriores se da en todas las técnicas usadas.

		AVG	Desviación Tipica
kNN (k = 3)	**Accuracy (%)**	70.3333	9.361
	F1 (%)	69.5064	9.86927
kNN (k = 4)	**Accuracy (%)**	70.683	3.2664
	F1 (%)	70.33985	3.22298
kNN (k = 7)	**Accuracy (%)**	75.0666	8.7330
	F1 (%)	74.2276	9.1946
kNN (k = 10)	**Accuracy (%)**	71.4666	7.254
	F1 (%)	70.32600	7.90166
kNN (k = 20)	**Accuracy (%)**	73.55	10.6260
	F1 (%)	72.4541	11.776
kNN (k = 100)	**Accuracy (%)**	71.8666	6.635
	F1 (%)	71.374	6.9955

Cuadro 15: Resultados de kNN con k-fold

4.3.3. Discusión:

Tras analizar los resultados obtenidos con la combinación de las características de cantidad de color y de gris de las imágenes de la base de datos, podemos concluir que el cambio en los resultados es tan pequeño, que la nueva característica no ha afectado realmente a la clasificación de los patrones, una posibilidad que podíamos intuir al inicio de esta aproximación. Esto probablemente se deba a que la característica añadida en esta aproximación, cantidad media de gris de las imágenes tras pasarlas a escala de grises, es redundante, es decir, la información que aporta ya está contenida en la característica utilizada hasta el momento, cantidad media de color. Por lo tanto SVM, y kNN siguen dándonos peores resultados que en la primera aproximación, cuando solo teníamos dos clases.

Aunque los resultados continúan siendo en parte satisfactorios, pues un 70 % de acierto con 3 clases distintas en un problema tan complejo como el de clasificación de pinturas es un resultado más que aceptable, vamos a utilizar una nueva característica, reemplazando la añadida en esta aproximación, los valores el escala de color HSV, HSV: HUE, Saturation y Value.

4.4. Aproximación 4:

4.4.1. Descripción:

Para esta aproximación, hemos añadido los valores de la escala de color HSV, reemplazando la característica de la escala de grises pues no conseguimos los resultados cuando la empleamos en la aproximación 3, probablemente esto se deba a que la información que aportaba era redundante.

HSV está formado por tres valores.

	HUE	**Saturation**	**Value**
Pop Art	142.96503	0.602644	0.64932925
Barroco	185.56796	0.3819065	0.30468854
Ukiyo-e	54.708275	0.2002211	0.8635214

Cuadro 16: Resultados de las medias acumuladas.

Comparamos entonces la media de los tres valores de la escala HSV: HUE, Saturation y Value para hacer la clasificación en base a estos nuevos valores, esperando obtener mejores resultados que la anterior aproximación.

4.4.2. Resultados:

Probamos diferentes arquitecturas de Redes Neuronales Artificiales. Usamos las mismas arquitecturas que en las anteriores aproximaciones para ser rigurosos en la experimentación:

		AVG	Desviación Tipica
RR.NN.AA **[1]**	**Accuracy (%)**	53.5599	8.8130
	F1 (%)	46.4117	10.2364
RR.NN.AA **[2]**	**Accuracy (%)**	70.3383	9.7381
	F1 (%)	66.0323	12.4224
RR.NN.AA **[10]**	**Accuracy (%)**	75.6866	7.0560
	F1 (%)	74.2213	8.0400
RR.NN.AA **[315]**	**Accuracy (%)**	47.4366	5.0238
	F1 (%)	37.0294	6.0837
RR.NN.AA **[1,1]**	**Accuracy (%)**	42.0783	5.5906
	F1 (%)	29.9126	7.0758
RR.NN.AA **[2,2]**	**Accuracy (%)**	51.9416	8.3943
	F1 (%)	44.0881	11.2454
RR.NN.AA **[4,3]**	**Accuracy (%)**	60.6033	9.3889
	F1 (%)	54.3232	11.1174
RR.NN.AA **[12,16]**	**Accuracy (%)**	67.2033	5.7092
	F1 (%)	64.1380	6.6955

Cuadro 17: Resultados de RR.NN.AA con k-fold

Pese a que esperábamos una mejora, no cóntabamos con que fuese tan notable, llegando a mejorar más de un 10 % en algunas arquitecturas, siendo la mejor con diferencia [10], con un 75 % de precisión. Estos resultados nos sorprenden gratamente pues son ampliamente mejores que en las anteriores aproximaciones usando RR.NN.AA.

Podemos apreciar que en las SVM esta mejoría se mantiene, con resultados sólidos de aproximadamente 82 % con una variedad de arquitecturas. Parece que utilizar el kernel lineal ofrece algún tipo de mejoría, obteniendo los mejores resultados de toda la aproximación con C = 25, con un 85 % de precisión.

		AVG	Desviación Tipica
SVM(C 1)	**Accuracy (%)**	82.7499	7.5175
	F1 (%)	82.7911	7.5637
SVM(C 2)	**Accuracy (%)**	82.7166	9.2789
	F1 (%)	82.5407	9.4642
SVM(C 3)	**Accuracy (%)**	81.4999	6.9482
	F1 (%)	81.3538	7.0511
SVM(C 6)	**Accuracy (%)**	82.6999	7.8888
	F1 (%)	82.5392	7.9240
SVM(C 10)	**Accuracy (%)**	82.7333	7.7774
	F1 (%)	82.8098	8.0522
SVM(C 25)	**Accuracy (%)**	82.3833	6.8141
	F1 (%)	82.4102	6.8875
SVM(C 100)	**Accuracy (%)**	82.7333	4.9933
	F1 (%)	82.6874	4.8326
SVM(kernel lineal C 1)	**Accuracy (%)**	83.0999	7.8379
	F1 (%)	83.1184	7.9831
SVM(kernel lineal C 5)	**Accuracy (%)**	84.3166	9.1794
	F1 (%)	84.2876	9.0764
SVM(kernel lineal C 25)	**Accuracy (%)**	85.15	10.9900
	F1 (%)	84.987	11.2026

Cuadro 18: Resultados de SVM con k-fold

Con respecto a Árbol de decisión, se mantiene la tendencia a la mejoría de resultados aunque levemente por debajo de las SVM. Nuestro punto cumbre parece ser con una mD 6 con un 81 % de acierto, empeorando con el aumento de esta variable.

		AVG	Desviación Tipica
Árbol de decisión(mD 4)	**Accuracy (%)**	80.7	6.0009
	F1 (%)	80.5757	5.5819
Árbol de decisión(mD 5)	**Accuracy (%)**	80.6333	9.9857
	F1 (%)	80.6069	9.9840
Árbol de decisión(mD 6)	**Accuracy (%)**	81.1333	4.5979
	F1 (%)	81.5365	4.3948
Árbol de decisión(mD 10)	**Accuracy (%)**	74.7166	5.5932
	F1 (%)	74.9661	5.4612
Árbol de decisión(mD 13)	**Accuracy (%)**	77.0833	7.1721
	F1 (%)	76.8999	7.2212
Árbol de decisión(mD 25)	**Accuracy (%)**	77.5333	7.2897
	F1 (%)	77.7556	7.1686
Árbol de decisión(mD 147)	**Accuracy (%)**	75.0666	6.3805
	F1 (%)	74.9237	5.7896

Cuadro 19: Resultados de con k-fold

Centrándonos en kNN, confirmamos la mejoría general en el total de los datos. En este caso, obtenemos valores cumbre similares a los de los Árbol de decisión, pero con un comportamiento más variable.

		AVG	Desviación Tipica
kNN (k = 3)	**Accuracy (%)**	79.8833	8.9684
	F1 (%)	79.6104	9.1932
kNN (k = 4)	**Accuracy (%)**	80.3	7.4840
	F1 (%)	80.0820	7.7293
kNN (k = 7)	**Accuracy (%)**	81.1333	7.9879
	F1 (%)	80.7061	8.8325
kNN (k = 10)	**Accuracy (%)**	78.7833	11.7562
	F1 (%)	78.4365	12.1113
kNN (k = 20)	**Accuracy (%)**	81.9166	8.5111
	F1 (%)	81.6082	8.5749
kNN (k = 100)	**Accuracy (%)**	77.5166	7.0791
	F1 (%)	76.1471	8.7014

Cuadro 20: Resultados de kNN con k-fold

4.5. Aproximación 5:

4.5.1. Descripción:

Para esta aproximación, hemos añadido una nueva categoría artística, el Realismo. Motivados por los buenos resultados de la anterior aproximación, hemos decidido forzar la maquinaria para ver como se desenvolvería esta en un escenario con 4 tipos diferentes de posibles clasificaciones.

	HUE	**Saturation**	**Value**
Pop Art	142.96503	0.602644	0.64932925
Barroco	185.56796	0.3819065	0.30468854
Ukiyo-e	54.708275	0.2002211	0.8635214
Realismo	54.708275	0.2002211	0.863521

Cuadro 21: Resultados de las medias acumuladas.

Hemos elegido este estilo porque consideramos que, aunque tiene personalidad propia, se asemeja mucho al barroco. Así pues, esperamos poner en un aprieto al sistema para ver como se desenvuelve.

(a) Ejemplo de Realismo. (b) Histograma RGB del Realismo

4.5.2. Resultados:

Probamos diferentes arquitecturas de Redes Neuronales Artificiales. Usamos las mismas arquitecturas que en las anteriores aproximaciones para ser rigurosos en la experimentación:

		AVG	Desviación Tipica
RR.NN.AA **[1]**	**Accuracy (%)**	38.0625	3.4676
	F1 (%)	29.6499	3.4876
RR.NN.AA **[2]**	**Accuracy (%)**	49.7288	6.8891
	F1 (%)	44.6083	7.8794
RR.NN.AA **[10]**	**Accuracy (%)**	56.0070	8.7427
	F1 (%)	52.2499	10.5469
RR.NN.AA **[315]**	**Accuracy (%)**	39.0393	4.1548
	F1 (%)	28.0285	4.9290
RR.NN.AA **[1,1]**	**Accuracy (%)**	30.4788	7.2799
	F1 (%)	18.3629	7.5556
RR.NN.AA **[2,2]**	**Accuracy (%)**	35.5312	6.9557
	F1 (%)	25.7831	8.1655
RR.NN.AA **[4,3]**	**Accuracy (%)**	43.5241	6.5456
	F1 (%)	37.1464	7.10797
RR.NN.AA **[12,16]**	**Accuracy (%)**	51.3155	7.2149
	F1 (%)	46.5077	8.3589

Cuadro 22: Resultados de RR.NN.AA con k-fold

Como habíamos previsto, todos los resultados caen alrededor de un 10-15 %, siendo la mejor con diferencia [10], con un 56 % de precisión. Estos resultados no nos sorprenden pues ciñendonos a los colores es francamente diferenciar una obra con la nueva categoría añadida. Un 56 % pese a que es un resultado muy mejorable no nos parece despreciable, y si nos centrásemos en características de formas también creemos que podríamos subir este porcentaje bastante.

Podemos apreciar que en las SVM este descenso se mantiene, aunque se siguen obteniendo resultados mejores que con las RR.NN.AA. Casi todas las arquitecturas obtienen un mismo porcentaje de acierto que ronda el 60 %, pero si tuviéramos que quedarnos con una sería con C3 ya que la desviación es la menor.

		AVG	Desviación Tipica
SVM(C 1)	**Accuracy (%)**	62.5100	6.4550
	F1 (%)	61.5201	6.3132
SVM(C 2)	**Accuracy (%)**	64.1028	5.7962
	F1 (%)	63.4324	5.5224
SVM(C 3)	**Accuracy (%)**	62.8225	8.3471
	F1 (%)	61.3448	9.0673
SVM(C 6)	**Accuracy (%)**	62.2076	5.9336
	F1 (%)	61.3096	5.9020
SVM(C 10)	**Accuracy (%)**	64.7177	9.4822
	F1 (%)	63.1268	9.1680
SVM(C 25)	**Accuracy (%)**	62.8427	7.9270
	F1 (%)	61.5164	8.9053
SVM(C 100)	**Accuracy (%)**	60.2822	8.1990
	F1 (%)	58.7409	9.6712
SVM(kernel lineal C 1)	**Accuracy (%)**	63.4778	7.6307
	F1 (%)	62.7238	6.8424
SVM(kernel lineal C 5)	**Accuracy (%)**	64.7883	9.6926
	F1 (%)	63.2660	10.0922
SVM(kernel lineal C 25)	**Accuracy (%)**	64.0826	10.3654
	F1 (%)	63.0858	10.3497

Cuadro 23: Resultados de SVM con k-fold

Con respecto a Árbol de decisión, se mantiene la misma tendencia que en las anteriores aproxima-

ciones y los resultados se situan levemente por debajo de las SVM. Nuestro punto cumbre parece ser con una mD 4 con un 62 % de acierto. Sorprende eso sí el aumento de la desviación típica, indicando que estos valores son poco estables.

		AVG	Desviación Tipica
Árbol de decisión(mD 4)	**Accuracy (%)**	62.2379	10.5237
	F1 (%)	60.7451	10.8681
Árbol de decisión(mD 5)	**Accuracy (%)**	57.7822	11.4809
	F1 (%)	56.8710	10.8562
Árbol de decisión(mD 6)	**Accuracy (%)**	55.7963	8.1253
	F1 (%)	55.4828	7.6427
Árbol de decisión(mD 10)	**Accuracy (%)**	53.0040	8.9005
	F1 (%)	52.7214	8.0326
Árbol de decisión(mD 13)	**Accuracy (%)**	55.7862	9.2054
	F1 (%)	55.9282	9.1883
Árbol de decisión(mD 25)	**Accuracy (%)**	50.8366	13.3321
	F1 (%)	51.7344	13.9837
Árbol de decisión(mD 147)	**Accuracy (%)**	58.3669	8.3593
	F1 (%)	58.3367	9.2069

Cuadro 24: Resultados de con k-fold

Centrándonos en kNN, se puede que ver que todos los resultados han empeorado claramente con la inclusión de una nueva categoría, haciendo más ambigua la clasificación. En este caso los resultados oscilan entre un 58 y un 64 %.

		AVG	Desviación Tipica
kNN (k = 3)	**Accuracy (%)**	60.5544	10.0303
	F1 (%)	59.3134	10.9699
kNN (k = 4)	**Accuracy (%)**	59.9697	10.1182
	F1 (%)	58.0965	11.1382
kNN (k = 7)	**Accuracy (%)**	64.1834	8.0600
	F1 (%)	63.0160	7.3008
kNN (k = 10)	**Accuracy (%)**	64.1330	4.1674
	F1 (%)	62.5414	4.7160
kNN (k = 20)	**Accuracy (%)**	63.8508	9.0670
	F1 (%)	62.2675	9.0500
kNN (k = 100)	**Accuracy (%)**	58.7298	7.0457
	F1 (%)	57.2659	8.1961

Cuadro 25: Resultados de kNN con k-fold

4.5.3. Discusión:

En esta aproximación, motivados por los buenos resultados de la anterior, decidimos arriesgarnos y poner a prueba nuestro sistema con un estilo artístico más, el Realismo.

Pese a que sabíamos con bastante certeza que nuestros resultados descenderían mucho con la inclusión de un estilo tan ambiguo, intentamos forzar la maquinaria para ver que tal se desenvolvería en un caso extremo. Al tener unos estilos mucho más ambiguos que incluso a alguno de nosotros nos ha costado identificar, nuestro sistema a perdido bastante en el porcentaje de acierto. Mientras en la anterior aproximación obteniamos en SVM una tasa de acierto del 82 %, ahora siguen siendo nuestra cumbre pero con aproximadamente un 20 % menos de acierto.

Mencionar también que pese a que en la anterior aproximación premiábamos la mejoría con las Redes Neuronales Artificiales (RR.NN.AA), ahora volvemos al punto de partida, con los resultados

más pobres de todos los análisis, cayendo de un 75 % en la anterior aproximación a un 55 % en esta.

Por todo ello concluimos que nuestro sistema no está todavía preparado como para la inclusión de estilos difusos, y que se debería trabajar más en la obtención de características diferenciadoras que ayuden en la clasificación, tema que se trabajará en posteriores aproximaciones.

4.6. Aproximación 6: Deep Learning

4.6.1. Descripción:

Para comprobar la efectividad del deep learning, hemos creado 10 arquitecturas diferentes. Cambiaremos los paddings, el numero de capas y las entradas y salidas de cada una, intentando experimentar con estos valores y ver que configuración es la mas adecuada para el problema que tenemos entre manos.

Para esta aproximación, clasificaremos las imágenes en solo tres categorías: barroco, pop Art y ukiyo-e. No incluimos realismo, porque en realidad esta aproximación fue realizada antes que la aproximación 5 (ya que esa era opcional y preferimos asegurar primero que teníamos tiempo para finalizar la de Deep Learning). Por tanto, a la hora de comparar resultados, debemos comparar esta aproximación con la número 4).

Antes de comenzar a analizar las imágenes, las tenemos que reducir, pues hacer computaciones con cientos de imágenes con sus tamaños originales requiere de un poder computacional, y sobre todo un tiempo, del que no disponemos, por no hablar de que muy probablemente haya mucha redundancia en imágenes tan grandes, y reducir el numero de píxeles no tendría un efecto tan detrimental para el sistema.

(a) Imagen original de barroco. (b) Imagen barroca tras un resize de 28*28.

En la figura anterior vemos el antes y después de una imagen ejemplo tras hacerle una reducción de píxeles, para poder servir de entrada a la primera capa. Decidimos que las imágenes tuviesen unas dimensiones de 28*28, pues vimos que es un tamaño bastante estándar para trabajar con imágenes en deep learning.

Dividiremos también las imágenes de entrada en dos conjuntos, entrenamiento y test. Para la fase de entrenamiento usaremos 224 fotos, mientras que en el test utilizaremos las 25 restantes. Designamos así mismo una semilla que nos permitirá poder distribuir las imágenes de la misma forma siempre, para contar con las mismas condiciones iniciales y asegurar la replicabilidad de nuestros experimentos.

4.6.2. Resultados:

En todas las configuraciones utilizaremos las mismas tres capas finales, reshape, Dense y softmax. La primera cambia las dimensiones del tensor (de 3D a 2D). Dense es una capa que simula un perceptron multicapa clásico, y transforma las entradas en 3 salidas, pues está clasificando entre tres clases (barroco, pop art y ukiyo-e). En la última capa se aplica la función softmax a las 3 salidas de Dense, y se devuelven tres valores entre 0 y uno, que suman 1 en total, y representan las probabilidades de la imagen de ser de cada clase.

Usaremos también siempre relu como función de transferencia de las capas convolucionales. Podremos entonces jugar con el padding, que impide que haya zonas de la imagen donde se apliquen menos filtros, y el propio número de filtros. El canal de entrada de la capa convolucional inicial siempre será 3, pues hay 3 valores de entrada ya que contamos con imágenes RGB. Podemos sin embargo cambiar el número de canales de salida.

La capa MaxPool calcula el maximo valor para cada filtro dividiendo la entrada (el mapa de características) en el eje X y en el eje Y por dos, para crear un mapa de características reducido.

1. Para la primera configuración, designamos un total de 9 capas, con tres capas convolucionales y un padding de 1 tanto en el eje X como en el Y.

Capas:
Conv((3, 3), 3 => 16, relu, pad=1), MaxPool((2, 2)),
Conv((3, 3), 16 => 32, relu, pad=1), MaxPool((2, 2)),
Conv((3, 3), 32 => 32, relu, pad=1), MaxPool((2, 2)),
3648, Dense(288 => 3), softmax

resultados	**train accuracy (%)**	**test accuracy (%)**
configuracion 1	100.0	84.0

Podemos ver que los resultados son buenos, llegando a un 84 % de precisión en el conjunto de test, que es el importante. Tenemos también un 100 % de precision en el conjunto de entrenamiento, lo cual provocó una parada prematura del proceso de training. A pesar de ser 84 % un buen valor de precisión, creemos que pudo haber un overfitting en el conjunto de training, por eso el alto valor del 100 %, y que puede haber mejoría si arreglamos esto cambiando algunos parámetros.

2. En esta segunda configuración cambiamos el número de canales de salida de las capas convolucionales, doblando el primero de 16 a 32, es decir, ahora se generan 16 filtros, y por consiguiente la próxima capa convolucional aceptará el doble de canales que antes, y generará 64 en vez de 32. Mantenemos el padding a 1.

Capas:
Conv((3, 3), 3 => 32, relu, pad=1), MaxPool((2, 2)),
Conv((3, 3), 32 => 64, relu, pad=1), MaxPool((2, 2)),
Conv((3, 3), 64 => 64, relu, pad=1), MaxPool((2, 2)),
3650, Dense(576 => 3), softmax

resultados	**train accuracy (%)**	**test accuracy (%)**
configuracion 2	100.0	80.0

Podemos ver que los resultados siguen siendo buenos comparados con las anteriores aproximaciones, pero observamos que empeoran con respecto a la anterior arquitectura de deep learning, quedándose en un 80 % de precisión en el conjunto de test, y un 100 % en el de trainig.

3. En esta iteración probamos a eliminar el padding de las capas convolucionales, y nos basamos en la arquitectura que hasta ahora mejores resultados ha tenido (la 1).

Capas:
Conv((3, 3), 3 => 16, relu), MaxPool((2, 2)),
Conv((3, 3), 16 => 32, relu), MaxPool((2, 2)),
Conv((3, 3), 32 => 32, relu), MaxPool((2, 2)),
3652, Dense(32 => 3), softmax

resultados	**train accuracy (%)**	**test accuracy (%)**
configuracion 3	100.0	84.0

Eliminando el padding no podemos observar ningún cambio en los resultados recibidos, la precisión de test se ha mantenido en 84 % y la de training en 100 %.

4. En la arquitectura 4 hemos intentado con esta configuración que efecto tendría sobre los resultados mantener igual en todas las capas el número de canales de entrada.

Capas:
Conv((3, 3), 3 => 3, relu, pad=1), MaxPool((2, 2)),
Conv((3, 3), 3 => 3, relu, pad=1), MaxPool((2, 2)),
Conv((3, 3), 3 => 3, relu, pad=1), MaxPool((2, 2)),
3654, Dense(27 => 3), softmax)

resultados	**train accuracy (%)**	**test accuracy (%)**
configuracion 4	25.8928	36.0

Como era de esperar, el resultado se ve afectado de manera considerable por estos parámetros, siendo la precision tan solo 25.8928 % en el conjunto de training y 36 % en el de test. Viendo estos pésimos valores no volveremos a utilizar una configuración de este estilo.

5. En esta arquitectura probaremos a reducir el número de capas, para ver si estábamos sobrecomplicando el problema. Nos quedamos pues con 4 capas, teniendo tan solo una capa convolucional, que genera 16 filtros.

Capas:
Conv((3, 3), 3 => 16, relu, pad=1), MaxPool((2, 2)),
3656, Dense(3136 => 3), softmax)

resultados	**train accuracy (%)**	**test accuracy (%)**
configuracion 5	100	92.0

Recibimos muy buenos resultados, los mejores hasta ahora con una precisión en test del 92 % y una precision del training del 100 %, puede ser que efectivamente estuviésemos sobrecomplicando el problema con tantas capas convolucionales, pero creemos que aún queda espacio para mejoría.

6. Seguimos la tendencia de simplificar las arquitecturas y la despojamos esta vez de capas convolucionales, dejándola con tan solo las tres últimas capas (reshape, perceptrón multicapa y softmax), comunes a cada configuración.

Capas:
3658, Dense(2352 => 3), softmax

resultados	train accuracy (%)	test accuracy (%)
configuracion 6	70.0892	44.0

Los resultados son malos con respecto a las otras arquitecturas (70.0892 % en el conjunto de training y un 44 % en el de test), pero esto era de esperar, pues esta arquitectura es equivalente a un perceptron multicapa clásico.

7. Contamos en esta arquitectura con 6 capas, 3 de ellas MaxPool. Esto es equivalente a hacer tres veces una reducción de las imágenes iniciales, 28*28 − > 14*14 − > 7*7 − > 3*3, y usar el vector 3*3*3 como entrada al perceptron multicapa, así que como mínimo debería dar mejores resultados que la anterior configuración, pues es lo mismo pero preparando los datos antes de dárselos al perceptrón.

Capas:
MaxPool((2, 2)), MaxPool((2, 2)), MaxPool((2, 2)),
3660, Dense(27 => 3), softmax)

resultados	train accuracy (%)	test accuracy (%)
configuracion 7	48.2142	60.0

La precisioń en el conjunto de entrenamiento es 48.2142 %, y en el de test es 60 %, con lo que nuestra hipótesis es correcta, es un poco mejor que la anterior, pero se puede ver que donde verdaderamente está la potencia en el deep learning es en las capas convolucionales.

8. Tras los resultados anteriores volvemos a hacer uso de capas convolucionales para intentar mejorar los resultados de la arquitectura 1. Esta vez cambiaremos los paddings para ver si tienen algún efecto en los resultados, y utilizaremos el perceptron multicapa con función sigmoidal.

Capas:
Conv((3, 3), 3 => 16, relu, pad=2), MaxPool((2, 2)),
Conv((3, 3), 16 => 32, relu, pad=(3, 4)), MaxPool((2, 2)),
Conv((3, 3), 32 => 32, relu, pad=(1, 5)), MaxPool((2, 2)),
4514, Dense(1152 => 3,), softmax

resultados	train accuracy (%)	test accuracy (%)
configuracion 8	74.5535	68.0

Obtenemos unos resultados bastante malos con respecto a otras arquitecturas, con 74.5535 % de train accuracy y 68 % de test accuracy. Es posible que cambiar los paddings de forma arbitraria sea detrimental para los resultados.

9. En esta arquitectura cambiaremos la manera en que los mapas de características se reducen, através de utiilizar diferentes parámetros de MaxPool, lo demás será exactamente igual a la primera arquitectura.

Capas
Conv((3, 3), 3 => 16, relu, pad=1), MaxPool((3, 3)),
Conv((3, 3), 16 => 32, relu, pad=1), MaxPool((4, 4)),
Conv((3, 3), 32 => 32, relu, pad=1), MaxPool((2, 2)),
4516, Dense(32 => 3), softmax)

resultados	train accuracy (%)	test accuracy (%)
configuracion 9	100.0	84.0

Los resultados son idénticos a los de la primera configuración (100 de training y 84 de test), de lo que podemos deducir que en este caso el tamaño de MaxPool no afecta.

10. En esta arquitectura optamos por complicar las cosas innecesariamente, utilizando un total de 21 capas, con 9 capas convolucionales, repitiendo 3 veces las tres primeras capas convolucionales, y todo con un padding de (2,2).

Capas:
Conv((3, 3), 3 => 16, relu, pad=2), MaxPool((2, 2)),
Conv((3, 3), 16 => 32, relu, pad=2), MaxPool((2, 2)),
Conv((3, 3), 32 => 3, relu, pad=2), MaxPool((2, 2)),
...
4518, Dense(12 => 3), softmax)

resultados	**train accuracy (%)**	**test accuracy (%)**
configuracion 10	59.8214	52.0

Como era de esperar, los resultados no son muy buenos, siendo el train acc. 59.8214 % y el test acc. 52 %. Es una situación muy común sobrestimar la complicación de los problemas y obtener por lo tanto malos resultados.

4.6.3. Discusión:

resultados	**train accuracy (%)**	**test accuracy (%)**
configuracion 1	100.0	84.0
configuracion 2	100.0	80.0
configuracion 3	100.0	80.0
configuracion 4	25.89285	36.0
configuracion 5	100.0	92.0
configuracion 6	70.0892	44.0
configuracion 7	48.2142	60.0
configuracion 8	74.5535	68.0
configuracion 9	100.0	84.0
configuracion 10	59.8214	52.0

Los resultados de cada configuración ya fueron explicados en profundidad en la sección de cada una, aún así queremos destacar en esta discusión general lo distintos que son los resultados en función de la configuración empleada. Una variabilidad mucho mayor que la que nos encontrábamos al modificar los parámetros de las otras técnicas empleadas.
En lugar de probar configuraciones parecidas unas a otras, preferimos probar con casos muy distintos para abarcar un mayor rango de resultados y aprender mejor que configuraciones benefician más al sistema.
A pesar de esta variabilidad, obtuvimos algunos resultados estupendos; como en la configuración 5.

5. Conclusiones

Como se ha descrito ya en diversas partes de esta memoria, a través de este estudio hemos intentado crear un sistema para la correcta clasificación de arte según su corriente artística. Este problema, pese a estar en un terreno un poco pantanoso en alguno de sus géneros (pues ni nosotros mismos hemos clasificado bien todas las imágenes), nos resulta muy interesante, pues podría ayudar tanto a museos como a otras entidades culturales en la creación y organización de plataformas online de cuadros.

Si nos fijamos en estudios de ámbitos u objetivos similares podemos apreciar unos resultados muy dispares dependiendo de la cantidad de estilos que se intentan clasificar y la amplitud de la base de datos usada para entrenar los distintos sistemas. Estas investigaciones se detallan en la sección 3.

Como se ha podido ver a través de las diversas iteraciones en el apartado anterior (Desarrollo), nuestra tarea de indagación ha pasado por diversas etapas, algunas con resultados positivos y otras con peores, pero igualmente alentadores. A modo de resumen de nuestra dirección del proyecto, resaltar que empezamos con solamente dos estilos para poder obtener una clasificación sencilla e ir habituándonos al equipo, entorno y problema. Cuando vimos que estábamos listos para un salto de dificultad, añadimos un nuevo estilo artístico, que hizo que nuestros resultados empeorasen. Como ya no nos valía con ver las medias acumuladas de los colores primarios, intentamos pasar las imágenes a una escala de grises y ver si con ello obteníamos una mejor clasificación. Desgraciadamente, este no fue el caso, y tuvimos que seguir buscando otra forma de mejorar estas estadísticas. Finalmente nos centramos en la escala de color comparando de una forma más completa los espectros de color de los cuadros. Esta mejora fue tan notable que nos empujó a añadir una corriente artística nueva. Así pues, vimos de nuevo nuestros resultados mermados por esta inclusión, y pensamos en la utilización de centroides para mejorar la clasificación, pero la limitación de tiempo nos impidió explorar este camino.

Consideramos los resultados obtenidos como altamente positivos, a pesar de que fueran relativamente bajos con la técnica de RR.NN.AA. Llegamos a obtener un 85 % de precisión con la técnica de SVM, y cerca de un 82 % con las otras técnicas, lo que para un problema tan complejo como el de separar obras de arte por estilo son resultados más que aceptables, llegando finalmente a dividir entre 3 estilos, Barroco, PopArt, y Ukiyo-e.

A lo largo de las aproximaciones hemos podido observar que SVM, Árbol de decisión y kNN han sido capaces de clasificar las obras con gran precisión, aunque poco a poco SVM ha demostrado dar los mejores resultados, y a pesar de inicialmente quedar RR.NN.AA muy rezagado, dando resultados inicialmente aleatorios en todas las arquitecturas que probamos, ha conseguido poco a poco acercarse, con un 75 % de precisión con arquitectura [10], que aún siendo eclipsados por los mejores valores de las otras técnicas, no dejan de destacar por su cuenta, demostrando que existe la posibilidad de que con más o mejores características llegue a igualar o incluso superar las otras técnicas utilizadas.

Con respecto a la aproximación de Deep Learning, los resultados nos muestran claramente que puede competir e incluso sobrepasar a todas las otras tecnologías. Vemos que en deeplearning hay una mayor variación de los resultados con respecto a los parámetros de entrada. Con tres estilos logra una precisión de entre 80 % hasta 90 % con las arquitecturas que mejor resultados dan. Cabe destacar también que su escalabilidad con respecto a las otras herramientas es mucho mayor, debido a su capacidad de funcionar en futuras expansiones sin tener que designar nuevas características.

En resumen, a pesar de tener algunos problemas obteniendo las características, pues una de las utilizadas, cantidad media de gris de las imágenes a escala de grises, no fue de ninguna ayuda al clasificar; estamos muy satisfechos con los resultados a los que hemos llegado, que aunque no son tan ambiciosos como los de algunos de los trabajos a los que hemos hecho referencia (menor cantidad de estilos y mucho menor número de patrones), nos ha servido para dar un paso por ese camino.

6. Trabajo Futuro

La clasificación de obras de arte es un campo que cuenta hoy en día con sistemas muy complejos y que se lleva perfeccionando desde las últimas dos décadas. Aunque nuestro sistema es bastante sencillo, podría ser fácilmente expandido en el futuro añadiendo los principales estilos de arte que se encuentran en los museos más importantes actualmente. Podríamos intentar, incluso, adivinar el autor de las obras, aunque sería mucho más complejo.

Es también interesante poder disponer de una base de datos global de arte, donde haya millones de obras ya clasificadas, y de la cual se puedan nutrir otros sistemas mas complejos como el DALL-E [RDN+22], una IA que se nutre de obras y referencias de todo tipo para crear obras de arte a partir de peticiones en lenguaje natural. Así pues, si contásemos con una base de datos más amplia que con la que trabajamos actualmente el proyecto se vería sin duda beneficiado por una diversidad que ahora solo estamos simulando, pues contamos con un número suficiente pero lejos de ser representativo de la inmensa diversidad de cada género.

El caso más novedoso, ambicioso y a la vez cercano al problema que estamos intentando resolver es la elaboración de un sistema que permita diferenciar si una obra es original de un autor o si es una falsificación, campo que está escasamente trabajado con el aprendizaje automático y que requiere de un estudio y conocimiento humano muy elevado.

Y aunque este sistema que hemos trabajado se base en clasificación de obras de arte, puede extenderse a otras áreas donde cada vez ven un uso más importante las técnicas de aprendizaje automático, como la médica y la automobilística. En el ámbito médico, es de actualidad la identificación de células cancerosas o sanas, a través de imágenes de pruebas de la zona a tratar. Y en el ámbito automobilístico se usa actualmente para el desarrollo de coches autónomos, pudiendo trabajar en diferenciar si en una calle está cruzando un peatón o cual es la velocidad máxima permitida en la zona a través de las señales horizontales y del suelo.

Glosario

Aprendizaje Vago (lazy learning) método de aprendizaje en el que la generalización más allá de los datos de entrenamiento es demorada hasta que se hace una pregunta al sistema. 3

Cross-validation En este método, se parte el conjunto de datos en k subconjuntos disjuntos y se realizan k experimentos. En el k-ésimo experimento, el subconjunto k se separa para realizar test, y los k-1 restantes se utilizan para entrenar, realizando un k-fold crossvalidation. Un valor habitual suele ser k=10, con lo que se tiene un 10-fold crossvalidation. Finalmente, el valor de test correspondiente a la métrica adecuada será el valor promedio de los valores de cada uno de los k experimentos. 8

Deep Learning conjunto de algoritmos de aprendizaje automático que intenta modelar abstracciones de alto nivel en datos usando arquitecturas computacionales que admiten transformaciones no lineales múltiples e iterativas de datos expresados en forma matricial o tensorial. 4, 7, 24, 29

HSV El modelo HSV, también llamado HSB, define un modelo de color en términos de sus componentes. 18

HUE El tono, matiz o tonalidad (en inglés Hue) es una de las propiedades o cualidades fundamentales en la propiedad de un color. 18, 21

Julia Julia es un lenguaje de programación homoicónico, multiplataforma y multiparadigma de tipado dinámico de alto nivel y alto desempeño para la computación genérica, técnica y científica, con una sintaxis similar a la de otros entornos de computación similares. 8

MLP Mixer Arquitectura para visión artificial basada exclusivamente en perceptrones multicapa (MLP). 3, 7

Red Neuronal Residual Red neuronal Artificial que utiliza conexiones o atajos para saltar por encima de diferentes capas. 7

RGB Composición del color en términos de la intensidad de los colores primarios de la luz. 6, 12, 21, 25

Saturation En la teoría del color, la saturación, colorido o pureza es la intensidad de un matiz específico. 18, 21

Value Se denomina valor a la amplitud de la luz que define el color; más cerca del negro, más bajo es el valor. 18, 21

Árbol de decisión Un árbol de decisión es un mapa de los posibles resultados de una serie de decisiones relacionadas.. 3, 9, 10, 14, 17, 20, 22, 23, 29

Siglas

CNN Redes Neuronales Convolucionales. 4, 7

kNN k-Nearest Neighbours. 3, 10, 14, 17, 18, 20, 21, 23, 29

RR.NN.AA Redes Neuronales Artificiales. 3, 9–13, 16, 19, 21–23, 29

SVM Máquinas de Vectores de Soporte. 3, 4, 9, 11, 13, 16–20, 22, 23, 29

ViT Transformadores de Visión. 7

Referencias

[AJO+19] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Muhammad Ubale Kiru, Usman Gana, Abdullai Aminu Kazaure, Humaira Arshad, Okafor Uchenwa Linus, Abubakar Malah Umar, and Dada. Comprehensive review of artificial neural network applications to pattern recognition, Oct 2019.

[Alk21] Moritz Alkofer. Using convolutional neural networks to compare paintings by style. *Intersect*, 15(1), 2021.

[All20] Allegra. Art detective: Ai for art style recognition. https://medium.com/swlh/art-detective-ai-for-art-style-recognition-4632e05c3496, Oct 2020.

[AVTS11] Saurabh Agrawal, Nishchal K. Verma, Prateek Tamrakar, and Pradip Sircar. Content based color image classification using svm. In *2011 Eighth International Conference on Information Technology: New Generations*, pages 1090–1094, 2011.

[CG16] Eva Cetinic and Sonja Grgic. Genre classification of paintings, Nov 2016.

[HSSD17] Samet Hicsonmez, Nermin Samet, Fadime Sener, and Pinar Duygulu. Draw: Deep networks for recognizing styles of artists who illustrate children's books, Apr 2017.

[INS+21] Lazaros Iliadis, Spyridon Nikolaidis, Panagiotis Sarigiannidis, Shaohua Wan, and Sotirios Goudos. Artwork style recognition using vision transformers and mlp mixer. *Technologies*, 10(1):2, 2021.

[LCD17] Han Liu, Mihaela Cocea, and Weili Ding. Decision tree learning based feature evaluation and selection for image classification. In *2017 International Conference on Machine Learning and Cybernetics (ICMLC)*, volume 2, pages 569–574, 2017.

[LNY17] Adrian Lecoutre, Benjamin Negrevergne, and Florian Yger. Recognizing art style automatically in painting with deep learning. *PMLR*, Nov 2017.

[LW07] D. Lu and Q. Weng. A survey of image classification methods and techniques for improving classification performance. *International Journal of Remote Sensing*, 28(5):823–870, 2007.

[Oom18] Emiel Oomien. Classification of painting style with transfer learning, Jul 2018.

[PHO19] Carina Popovici and Christiane Hoppe-Oehl. Can ai art authentication put an end to art forgery?, Sep 2019.

[RDN+22] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, Apr 2022.

[WFW21] Pin Wang, En Fan, and Peng Wang. Comparative analysis of image classification algorithms based on traditional machine learning and deep learning. *Pattern Recognition Letters*, 141:61–67, 2021.

[ZCW+20] Jiaxin Zhuang, Jiabin Cai, Ruixuan Wang, Jianguo Zhang, and Wei-Shi Zheng. Deep knn for medical image classification. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2020*, pages 127–136, Cham, 2020. Springer International Publishing.