Universidades de Burgos, León y Valladolid

Máster universitario

Inteligencia de Negocio y Big Data en Entornos Seguros

TFM del Máster Inteligencia de Negocio y Big Data en Entornos Seguros

Aplicación de Técnicas de Machine Learning para la Predicción de Diferencias de Color Percibidas

Presentado por Pablo Gallardo Fuentes
en Universidad de Burgos — 18 de julio de 2025
Tutor: Pedro Latorre Carmona
Co-tutor: Rafael Huertas Roa

Resumen

En este trabajo se aborda el problema de la estimación de diferencias de color percibidas por el ojo humano a partir de datos experimentales, con el objetivo de evaluar si los modelos basados en aprendizaje automático pueden superar a las fórmulas tradicionales como CIEDE2000 en términos de precisión perceptual. Para ello, se ha hecho uso de cinco bases de datos validadas por expertos en colorimetría, las cuales contienen pares de colores representados en el espacio CIELAB junto con sus correspondientes diferencias perceptuales obtenidas mediante ensayos visuales con observadores humanos.

A partir de estos datos, se ha desarrollado un conjunto de modelos de predicción, entre los que destaca una red neuronal artificial (ANN) entrenada con PyTorch. Dicho modelo ha mostrado un rendimiento superior al de la fórmula estándar CIEDE2000 en la métrica de evaluación STRESS. El proceso incluyó una normalización de los datos, optimización de hiperparámetros, la evluación de distintos tipos de modelos supervisados y la validación en diferentes bases de datos para asegurar la robustez de los resultados.

Los resultados obtenidos sugieren que el enfoque basado en datos puede capturar de forma más precisa las sutilezas de la percepción visual del color, lo que abre la puerta al desarrollo de métricas más adaptadas a contextos específicos. Finalmente, se proponen líneas de trabajo futuro centradas en la mejora del preprocesamiento, la incorporación de nuevos conjuntos de datos, y la exploración de arquitecturas de modelos más complejas.

Descriptores

machine learning, redes neuronales artificiales, máquinas de soporte vectorial, random forest, diferencias de color, percepción del color, CIEDE2000

Abstract

This work addresses the problem of estimating color differences perceived by the human eye based on experimental data, with the aim of evaluating whether machine learning models can outperform traditional formulas like CIEDE2000 in terms of perceptual accuracy. To this end, five databases validated by experts in colorimetry were used. These databases contain pairs of colors represented in the CIELAB color space along with their corresponding perceptual differences, obtained through visual experiments with human observers.

Based on these data, a set of predictive models were developed, among which an artificial neural network (ANN) trained with PyTorch stands out. This model showed superior performance compared to the standard CIEDE2000 formula using the STRESS evaluation metric. The process included data normalization, hyperparameter optimization, evaluation of various types of supervised models, and validation across different datasets to ensure the robustness of the results.

The findings suggest that data-driven approaches can more accurately capture the subtleties of human color perception, paving the way for the development of metrics better suited to specific application contexts. Finally, future work is proposed in the direction of improved preprocessing, integration of new datasets, and exploration of more complex model architectures.

Keywords

machine learning, artificial neural networks, support vector machine, random forest, color differences, color perception, CIEDE2000

Índice general

Índice de figuras

Índice de tablas

Memoria

Introducción

La percepción del color es, en muchos sentidos, una de las manifestaciones más fascinantes y complejas de la experiencia humana. Aunque la física pueda describir el color como una propiedad de la luz, lo que realmente vemos —y sentimos— va mucho más allá de longitudes de onda y espectros. El color es una construcción subjetiva, profundamente influida por nuestro sistema visual, nuestra memoria, nuestro entorno y hasta por nuestra cultura.

En el día a día, apenas lo notamos. Pero en sectores donde el color importa —y mucho—, como el diseño gráfico, la cosmética, la industria textil o la automoción, una mínima variación en tonalidad puede marcar la diferencia entre el éxito comercial y el rechazo más absoluto. Un rojo ligeramente más apagado en un envase, o un azul que no coincide exactamente con lo esperado, pueden generar devoluciones, pérdidas económicas o incluso dañar la imagen de una marca. Y, sin embargo, seguir confiando exclusivamente en fórmulas tradicionales para medir estas diferencias de color —como si la percepción humana pudiese encerrarse del todo en una ecuación— parece cada vez más insuficiente.

Es aquí donde entra en juego un recurso valiosísimo: los datos psicofísicos. Se trata de conjuntos cuidadosamente recopilados mediante experimentos, donde observadores expertos emiten juicios sobre cuánto difieren perceptivamente dos colores bajo condiciones controladas. Estos datos no son solo números; son una ventana al modo en que las personas realmente ven el color. Y nos ofrecen una base sobre la que construir algo más poderoso que una fórmula cerrada: un modelo que aprenda a ver como vemos nosotros.

La inteligencia artificial, y más concretamente el aprendizaje automático, ha demostrado ser especialmente prometedora en este tipo de retos. Cuando las reglas explícitas no bastan —cuando el fenómeno es demasiado sutil,

demasiado humano— los algoritmos pueden aprender directamente de la experiencia. Es decir, de los datos. A través del entrenamiento con grandes volúmenes de ejemplos, estos modelos son capaces de detectar patrones que se escapan tanto a la intuición humana como a los métodos clásicos.

Este trabajo se adentra precisamente en ese camino: combinar los principios fundamentales de la ciencia del color con el potencial de las técnicas modernas de machine learning. A lo largo de este proyecto, exploraremos si es posible entrenar modelos capaces de predecir la diferencia de color entre pares cromáticos de una forma más cercana a cómo lo haría un observador humano. Para ello, trabajaremos con varios conjuntos de datos psicofísicos y aplicaremos algoritmos de regresión supervisada —desde modelos clásicos hasta redes neuronales profundas— con el objetivo de acercarnos, tanto como sea posible, a la realidad de la percepción visual.

La utilidad de este enfoque va mucho más allá del interés académico. Un modelo capaz de predecir con precisión cómo percibimos las diferencias de color podría utilizarse para mejorar la calibración de pantallas, optimizar flujos de impresión o incluso diseñar sistemas de recomendación cromática personalizados. En definitiva, se trata de tender un puente entre los datos y la experiencia, entre el código y la percepción.

Objetivos

El presente Trabajo de Fin de Máster tiene como objetivo general el estudio, desarrollo y evaluación de modelos de aprendizaje automático para la predicción de la diferencia de color percibida por el ser humano, planteando una alternativa a las fórmulas matemáticas tradicionales utilizadas en este ámbito. Se busca así aprovechar la capacidad de los modelos basados en datos para capturar fenómenos de naturaleza subjetiva como la percepción visual, con el fin de mejorar la precisión y adaptabilidad de los sistemas actuales de gestión de color.

A partir de este objetivo general, se derivan los siguientes objetivos específicos:

- **Construcción y entrenamiento de modelos predictivos**: Implementar y entrenar distintos modelos de regresión supervisada capaces de aproximar la diferencia de color percibida, utilizando para ello datos psicofísicos empíricos como referencia. Entre los modelos considerados se incluyen métodos de tipo ensemble como *Random Forest*, algoritmos de margen como *Support Vector Regressor* (SVR) y aproximaciones no lineales de alta capacidad como las redes neuronales artificiales.
- **Evaluación comparativa del rendimiento**: Evaluar sistemáticamente el rendimiento de los modelos entrenados mediante métricas estándar como el erro cuadrático medio (*Mean Squared Error*, MSE). Además, se propone comparar dichos resultados con fórmulas clásicas de diferencia de color ampliamente aceptadas en la literatura, como CIEDE2000, con el objetivo de determinar en qué medida los modelos de aprendizaje automático pueden igualar o superar el rendimiento de estas aproximaciones tradicionales desde una perspectiva perceptual.

- **Diseño de un entorno experimental reproducible**: Desarrollar un entorno de experimentación robusto y replicable que abarque todas las fases del flujo de trabajo: carga y preparación de los datos, técnicas de normalización, entrenamiento de los modelos, validación y análisis de resultados. Para ello, se emplean herramientas y bibliotecas consolidadas en el ecosistema Python, como *scikit-learn*, XGBoost, *TensorFlow* o *PyTorch*.

- **Estudio del impacto del particionado de datos**: Analizar el efecto de diferentes estrategias de particionado sobre la capacidad de generalización de los modelos, comparando el enfoque tradicional de división en conjuntos *train/validation/test* con técnicas de validación cruzada (*cross-validation*).

- **Aplicabilidad futura e integración industrial**: Sentar bases para una futura integración de estos modelos en sistemas industriales de gestión del color, entornos de diseño gráfico o cadenas de fabricación donde la diferencia cromática percibida sea un factor determinante. Se plantea así una vía hacia soluciones más centradas en el usuario final, con el uso de inteligencia artificial como ventaja competitiva frente a métodos deterministas clásicos.

Conceptos teóricos

3.1. Colorimetría y diferencias de color

Para cuantificar el color de manera objetiva y consistente, la Comisión Internacional de la Iluminación (CIE, por sus siglas en francés) desarrolló el espacio de color CIELAB (L*a*b*). Esta organización se dedica al estudio de la luz, color y la iluminación, y se encarga de establecer normas y recomendaciones en relación con la percepción visual, la colorimetría y la medición de la luz [16].

El sistema de color CIELAB (también denomidado color Lab o simplemente Lab) busca representar las dimensiones de la percepción del color de una forma más uniforme desde el punto de vista perceptual. En el sistema CIELAB, la diferencia entre dos colores puede calcularse como la distancia euclídea entre sus coordenadas, lo que da lugar al valor conocido como ΔE. Esta medida objetiva de diferencia de color ha sido ampliamente adoptada en distintos ámbitos industriales y científicos debido a su capacidad para aproximarse, en cierta medida, a la percepción humana.

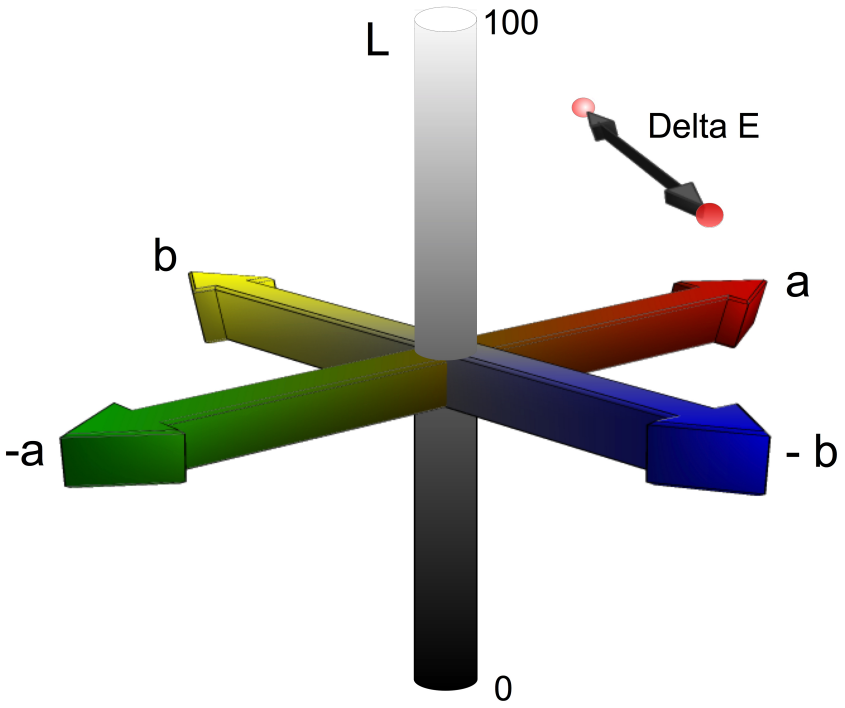


Figura 3.1: Espacio de color CIELAB. [9]

En la Figura 3.1 se muestra una representación del espacio de color CIELAB. Se trata de un espacio tridimensional, donde cada eje representa un parámetro utilizado para representar un color en colorimetría. Este espacio ha sido ideado para que las variaciones de color se correspondan, en la medida de lo posible, con diferencias perceptibles por el ojo humano. Sin embargo, uno de sus principales inconvenientes es su limitada capacidad para reflejar la complejidad de la percepción visual en contextos reales, especialmente bajo condiciones variables de iluminación u observación.

Las tres coordenadas cartesianas del sistema CIELAB se corresponden con:

- **Luminosidad (L*)**: como se observa en la Figura 3.1 va desde 0, negro absoluto, hasta 100, blanco absoluto. Indica la claridad, valores más altos, u oscuridad, valores más bajos, de un color.
- **Componente a***: La coordenada "a*" define el eje rojo-verde. Los valores negativos de a* indican una tendencia hacia el verde, mientras que los valores positivos indican una tendencia hacia el rojo.

- **Componente b***: Esta coordenada represente el eje azul-amarillo. Los valores negativos de b* indican una tendencia hacia el azul, mientras que los valores positivos indican una tendencia hacia el amarillo. Cuando los valores de las componentes a* y b* se aproximan a cero al mismo tiempo, los colores resultantes tienden a tonalidades grises o neutras.

En la actualidad, el estudio de las diferencias de color no solo está limitado a la teoría perceptual y física, sino que se ha convertido en un desafío multidisplinar que involucra grandes cantidades de datos y procesos computacionales cada vez más complejos. En este ámbito, el uso de bases de datos han permitido el almacenamiento, procesamiento, normalización de múltiples estudios que aportan grandes cantidades de datos. Esto permite almacenar muestras de colores, sus variaciones, sus correspondientes percepciones de los usuario y variaciones matemáticas.

En este trabajo se han utilizado cinco bases de datos obtenidas a partir de experimentos controlados y validados por expertos en colorimetría. Cada una de ellas contiene registros que incluyen, como mínimo, pares de colores representados por sus coordenadas L*a*b* en el espacio de color CIELAB, así como sus correspondientes diferencias perceptuales ΔV (diferencia visual observada) y ΔE_{00} (calculada mediante la fórmula CIEDE2000).

La fórmula CIEDE2000 [12] es el modelo propuesto por la Commission Internationale de l'Éclairage (CIE) para cuantificar la diferencia perceptual entre dos colores expresados en el espacio CIELAB. Representa una evolución respecto a fórmulas anteriores, CIE76 y CIE94, al incorporar correcciones que mejoran la correspondencia entre los valores numéricos y la percepción visual humana. En concreto, introduce ajustes para tener en cuenta fenómenos como la variación en la sensibilidad a diferencias de color en distintas regiones del espacio cromático. El resultado es un valor escalar, ΔE_{00}, que estima cuán diferente es un par de colores desde el punto de vista del observador promedio.

Para visualizar mejor, en la Tabla 3.1 muestro cuales son los datos necesarios para poder entrenar a los modelos con un ejemplo de una de las bases datos.

Para clarificar la estructura de datos utilizados, en la Tabla 3.1 se presentan y describen los campos principales que componen los registros de las bases de datos. Cada observación está asociada a un identificador de par (*Pair*), que distingue cada combinación de colores comparados. Los colores se representan mediante sus coordenadas en el espacio CIELAB: L_1^*,

Pair	$\mathbf{L}_1^*$	$\mathbf{a}_1^*$	$\mathbf{b}_1^*$	$\mathbf{L}_2^*$	$\mathbf{a}_2^*$	$\mathbf{b}_2^*$	$\boldsymbol{\Delta V}$	$\boldsymbol{\Delta E}_{00}$
50	61,929	−10,558	40,607	61,895	−9,016	37,991	1,539	1,362
51	61,929	−10,558	40,607	61,999	−8,712	40,372	1,364	1,190

Tabla 3.1: Muestra de datos experimentales de la base de datos COMCorreg

a_1^*, b_1^* para el primer color y L_2^*, a_2^*, b_2^* para el segundo. Estas coordenadas corresponden a los ejes de luminosidad (L*), rojo-verde (a*) y azul-amarillo (b*), diseñados para reflejar la percepción humana del color. La variable ΔV recoge la diferencia visual percibida por los observadores en los experimentos originales, mientras que ΔE_{00} representa la diferencia de color calculada utilizando la fórmula estandarizada CIEDE2000.

El estudio de las diferencias de color es fundamental por varias razones. Los sistemas de codificación de color como CIELAB y la calibración de dispositivos dependen en gran medida de la capacidad de cómo entendemos estas diferencias. Por lo tanto, abordar el problema de las diferencias de color no solo es relevante desde un punto de vista técnico, sino también desde una perspectiva práctica.

Existen distintas fórmulas de diferencias de color para asegurarse que los resultados se correlacionen lo mejor posible con la percepción visual humana. A lo largo de la historia se han venido utilizando distintas fórmulas para cuantificar la relación entre la diferencia de color percibida por el observador y la calculada por una fórmula. Una de ellas ha sido ECIEDE2000 o el índice combinado PF/3. Sin embargo, este índice presenta gran complejidad y la incapacidad de determinar si la mejora de una nueva fórmula de color es estadísticamente significativa.

Para solucionar estas limitaciones, en el estudio [3] se propuso un nuevo índice llamado STRESS (*Standardized Residual Sum of Squares*). El STRESS se presenta como una medida más simple y con mejores propiedades matemáticas que los índices anteriores.

La fórmula de stress se define como se puede ver en Ecuación 3.1

$$STRESS = \left(\frac{\sum (\Delta E_i - F_1 \Delta V_i)^2}{\sum F_1^2 \Delta V_i^2} \right)^{1/2} \tag{3.1}$$

Donde

$$F_1 = \frac{\sum \Delta E_i^2}{\sum \Delta E_i \Delta V}$$

- ΔE es la diferencia de color calculada para el par de colores i.
- ΔV es la diferencia de color visual o percibida para el par de colores i.
- F_1 es un factor de escala que minimiza el valor de STRESS, ajustando los valores ΔV a la escala de ΔE.

Una de las principales ventajas de STRESS es que su valor está acotado en el rango de 0 a 1 (o de 0 % a 100 %), donde 0 indica una concordancia perfecta.

Además, a diferencia del PF/3, el STRESS permite realizar inferencias estadísticas mediante pruebas F para determinar si la diferencia en el rendimiento entre dos fórmulas de color es significativa. Esto se logra porque el ratio de los valores de STRESS al cuadrado entre dos fórmulas es idéntico al ratio de la varianzas de error residual utilizado en las pruebas F.

Gracias a estas propiedades, como su simplicidad, su robustez matemática, su capacidad para la inferencia estadística y su capacidad para la comparación entre distintas fórmulas o modelos, se ha escogido este índice evaluar el rendimiento de los distintos modelos entrenados frente a la fórmulas.

3.2. Aprendizaje automático

El trabajo se apoya en diversos conceptos y herramientas propias del aprendizaje automático y la inteligencia artificial, en particular aquellos orientados a la predicción de valores continuos o de regresión. En esta sección se describen los fundamentos teóricos de los modelos probados, los procedimientos de validación aplicados y las tecnologías empleadas para su implementación. **??**

El aprendizaje automático se diferencia de la programación tradicional en que no se establecen reglas explícitas para que un sistema realice una tarea, si no que a través de los datos, el aprendizaje automático es capaz de crear modelos que aprenden patrones, relaciones y comportamientos. El principal objetivo de este campo es generar modelos que sean capaces de generalizar a partir de unos datos de entrenamiento para realizar predicciones o tomar decisiones sobre datos nuevos nunca antes vistos. Esa disciplina ha impulsado avances revolucionarios en diversos campos como la voz, la visión por computador, los sistemas de recomendación o la conducción autónoma.

El aprendizaje automático se puede enmarcar dentro de tres paradigmas principales:

- **Aprendizaje Supervisado**: El modelo aprende a partir de un conjunto de datos etiquetado, es decir, donde cada ejemplo de entrada está acompañado de su salida correcta.

- **Aprendizaje No Supervisado**: El modelo trabaja con datos sin etiquetar. Este tipo se centra en buscar estructuras, patrones o anomalías inherentes en los datos.

- **Aprendizaje por Refuerzo**: El modelo aprende a tomar decisiones secuenciales interactuando con un entorno, recibiendo recompensas o castigos en función de las decisiones tomadas.

El problema abordado se encuadra dentro del paradigma del aprendizaje supervisado, al disponer de conjuntos de datos etiquetados que permiten al modelo aprender una relación entre las variables de entrada y la variable de salida. Particularmente, el trabajo se plantea como un problema de regresión, ya que la variable a predecir se trata de una variable continua.

Se han explorado diferentes modelos de aprendizaje automático, tanto tradicionales como basado en redes neuronales. A continuación se describe la base teórica de estos modelos :

- **Random Forest** (Bosque aleatorio) [11]: Es un algoritmo de tipo *ensemble* basado en la combinación de múltiples árboles de decisión. El objetivo de introducir distintos árboles individuales era reducir la varianza producida por cada árbol sin aumentar significativamente el sesgo. Cada árbol es entrenado sobre un subconjunto de los datos seleccionado mediante *bootstrap*, y en cada nodo se selecciona un subconjunto de características para encontrar la mejor división. Esta aleatoridad mejorar la capacidad de generalización del conjunto. En tareas de regresión, el resultado final del modelo se obtiene como la media de las predicciones de todos los árboles. Se puede observar un ejemplo visual del proceso en la Figura 3.2.

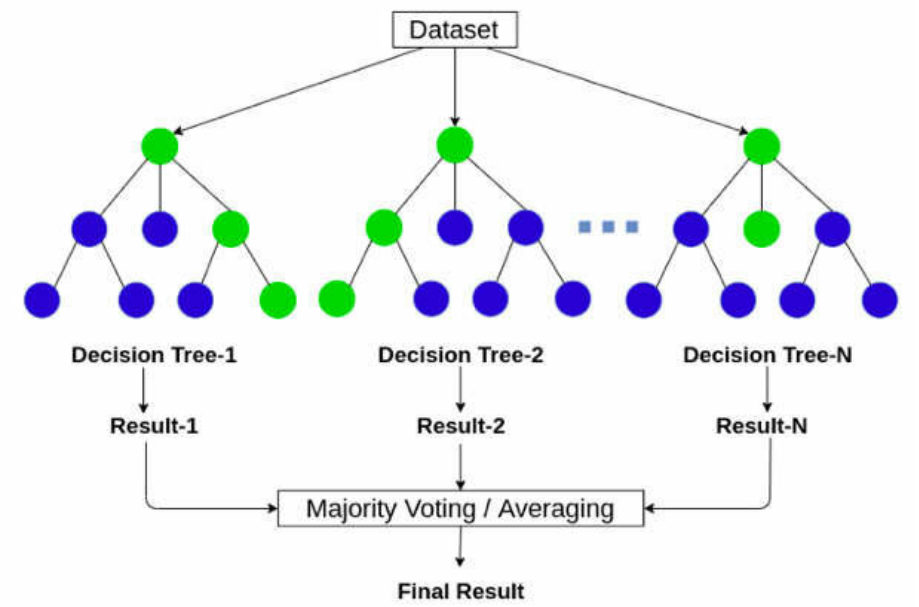


Figura 3.2: Random Forest. [4]

Este tipo de modelos destacan por evitar el sobreajuste (*overfitting*) frente a los árboles individuales, soportar datos no lineales y relaciones complejas como puede ser la predicción de la diferencia de color por el ojo humano y el manejo automático de las variables que el modelo considere irrelevantes. Por otra parte, tiene una interpretabilidad complicada, y aunque no ha sido el caso, puede tener poca eficiencia en tiempo de entrenamiento con grandes volúmenes de datos si no se paraleliza adecuadamente.

- **Support Vector Reggressor** (SVR) [15]: Es una extensión de las máquinas de vectores de soporte al dominio concreto de la regresión. El objetivo es encontrar un función $f(x)$ que desvíe como máximo ϵ de los valores buscados, en este caso ΔV, para todos los puntos del conjunto de entrenamiento. A diferencia de los modelos más tradicionales que buscan minimizar el error cuadrático medio entre las predicciones y los valores reales, el SVR introduce este concepto ϵ. La idea fundamental es penalizar solo aquellos valores que se encuentran fuera de este valor de tolerancia predefinido. SVR se centra en ajustar su función de regresión de manera que la mayoría de los datos de entrenamiento queden dentro del margen. Los datos que se quedan fuera contribuyen a la función de decisión final y se conocen como vectores de soporte. Esta caracterñistica otorga al SVR una muy buena eficiencia computacional

y una menor propensión al sobreajuste. SVR es capaz de modelar relaciones no lineales mediante el truco del kernel (kernel trick). La idea es mapear los datos de entrada a un espacio de característica de mayor dimensión, donde una relación lineal pueda separar los datos de manera efectiva.

Estos modelos tienen buena capacidad de generalización, incluso cuando se entrenan con pocos datos. Además se tiene un control explícito sobre el margen de error ϵ al ser un parámetro. Por otra parte, el modelo depende de ajustar estos parámetros de forma correcta para su funcionamiento en cada caso concreto y se puede volver costoso computacionalmente para grandes conjuntos de datos.

Las máquinas de soporte vectorial se utilizan en múltiples campos como la bioinformática, el procesamiento de imágenes o la predicción de series temporales financieras. Su capacidad para manejar datos de alta dimensionalidad y su robustez frente a outliers lo convierten en una opción atractiva para problemas de regresión complejos.

En el contexto de este trabajo, la elección de SVR se justifica por su capacidad para capturar patrones complejos, su solidez teórica y la existencia de librerías software que facilitan su implementación y evaluación.

- **XGBoost** (Extreme Gradient Boosting): XGBoost es una implementación optimizada del algoritmo de gradient boosting, ampliamente reconocido por su eficiencia computacional, capacidad de generalización y excelentes resultados en tareas de regresión y clasificación supervisada. Este algoritmo ha sido adoptado de forma masiva en competiciones de ciencia de datos y en entornos industriales por su robustez y velocidad.

  A diferencia de métodos tradicionales de boosting, XGBoost incorpora mejoras específicas tanto a nivel de algoritmo como de ingeniería de software, entre las que destacan:

  - Regularización L1 y L2: A diferencia del Gradient Boosting clásico, XGBoost incluye términos de regularización que penalizan modelos complejos y ayudan a evitar el overfitting.
  - Poda posterior del árbol (*post-pruning*): Los árboles se expanden completamente y luego se podan hacia atrás si la ganancia esperada no compensa la complejidad añadida. Esto permite un control más fino sobre la estructura del modelo.

- Cálculo de la ganancia basado en segundo orden: XGBoost utiliza tanto el gradiente como el Hessiano (derivada de segundo orden) de la función de pérdida para calcular la ganancia de las divisiones, lo cual acelera la convergencia y mejora la precisión.

- Manejo eficiente de valores perdidos: El algoritmo infiere automáticamente la mejor dirección para las muestras con valores faltantes, lo que permite entrenar modelos sin imputar previamente los datos.

- Paralelización del entrenamiento: XGBoost permite una paralelización eficiente de la construcción de los árboles, lo que lo hace muy escalable incluso con grandes volúmenes de datos.

Desde el punto de vista matemático, el objetivo de XGBoost es minimizar una función de pérdida diferenciable, como puede ser MSE para regresión, más un término de regularización que penaliza la complejidad del modelo.

En el presente trabajo, XGBoost se ha utilizado como uno de los modelos de regresión supervisada para predecir las diferencias de color. Su capacidad para manejar relaciones no lineales y su resistencia al overfitting lo convierten en un candidato ideal para este tipo de problema.

- **Redes Neuronales Artificiales** (ANN, Artifficial Neural Network) [6] [5]: Son modelos inspirados en la arquitectura del cerebro humano. Están compuestas por unidades llamadas *neuroanas artificiales* organizadas por capas.

  Una neurona artificial es la unidad básica de procesamiento de una red neuronal y está inspirada en el comportamiento de las neuronas biológicas. Su función principal es recibir entradas, procesarlas mediante una combinación lineal y producir una salida no lineal que será transmitida a otras neuronas. Como se puede ver en la Figura 3.3, la neurona recibe X_1, X_2, ..., X_n entradas que están ponderadas por sus correpondientes pesos w_1, w_2, ..., w_n. También puede incorporar un sesgo b que permite ajustar el resultado de la combinación lineal independientemente de las entradas.

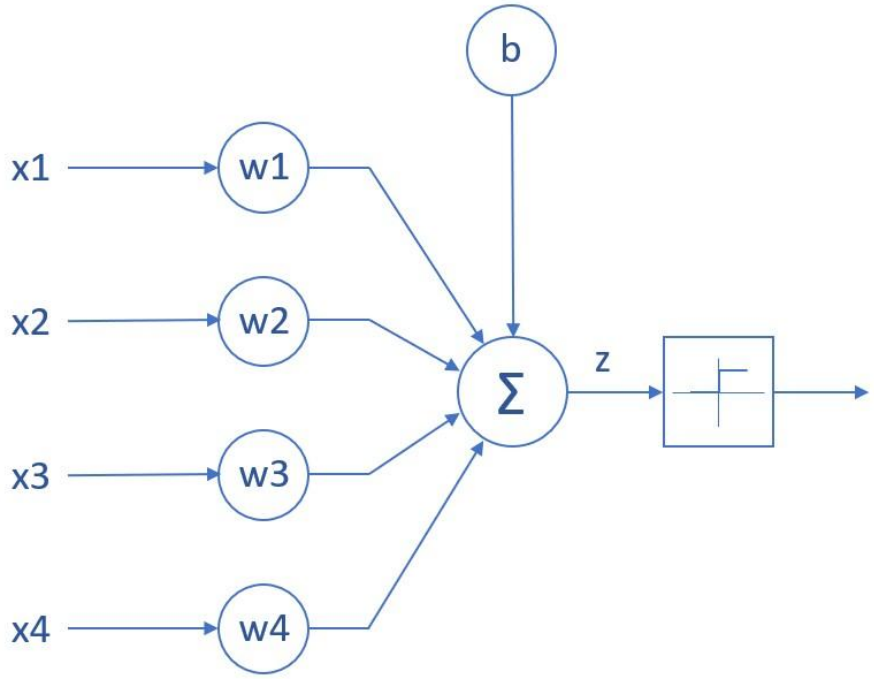


Figura 3.3: Estructura de una neurona artificial

La combinación lineal ponderada de las entradas más el sesgo se puede definir como en la Ecuación 3.2:

$$z = \sum_{i=1}^{n} w_i x_i + b \tag{3.2}$$

Una vez calculado el valor resultante z se pasa por una función de activación no lineal $\phi(z)$. Esta función determina la salida final de la neurona, que denominaremos a. Algunos ejemplos comunes de funciones de activación son los siguientes:

- ReLU (Rectified Linear Unit): $\phi(z) = max(0, z)$
- Sigmoide: $\phi(z) = \frac{1}{1+e^{-z}}$
- Tanh (Tangente hiperbólica): $\phi(z) = tanh(z)$

Por lo tanto, cada neurona de la red tendrá como propósito aprender representaciones no lineales de los datos de entrada. Al junta muchas neuronas en capa y ajustar sus pesos mediante el entrenamiento, la red puede aproximar funciones más complejas con precisión.

La arquitectura más básica para tareas de regresión es el perceptón multicapa (MLP) que consiste en una capa de entrada, una o varias capas ocultas intermedias y una capa de salida.

- Capa de entrada (Input Layer): Recibe los datos iniciales o las características del problema. Se utilizan normalmente valores normalizados y datos que han sido limpiados para seguir cierto formato. EL número de neuronas de esta capa se corresponde con el número de variables de entrada del problema o el número de características.
- Capas ocultas (Hidden Layers): Son las capas intermedias que conectan la entrada y la salida. La clave de estas capas es el uso de funciones de activación no lineales, como puede ser la función Sigmoide, la tangente hiperbólica (tanh) o la Unidad Lineal Rectificada (ReLU). Estas funciones permiten a la red aprender relaciones no lineales y complejas entre las variables. Una red puede tener una o varias capas ocultas. Cuando se tienen varias se suele denominar aprendizaje profundo o *Deep Learning*.
- Capa de salida (Output Layer): Produce el resultado final del modelo. El número de neuronas de esta capa dependerá de la tarea a realizar. Para tareas de regresión esta capa tendrá una única neurona sin función de activación que producirá el resultado final. En cambio, en tareas de clasificación multiclase, existirán tantas neuronas como clases contenga el problema, y además tendrán una función de activación como puede ser *softmax*.

El objetivo de una red neuronal artificial es minimizar o reducir una función de pérdida o función de coste, que mide la diferencia entre las predicciones realizadas por la red ($\hat{y}_i$) frente a los valores reales (y_i). Una función de pérdida muy utiliza en problemas de regresión es el Error Cuadrático Medio (MSE) que se define de la siguiente como en Ecuación 3.3:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{3.3}$$

Para lograr minimizar este error, se utilizan algoritmos de optimización basados en el descenso del gradiente. Algunos de estos algorimos son:

- SGD: Su lógica es simple y directa. Calcula el gradiente, actualiza los pesos en la dirección adecuada y utiliza una tasa de aprendizaje (*learning rate*) fija y única.

- *Adam* (Adaptive Moment Estimation): Es algo más sofisticado. Tiene *Momentum* manteniendo en "memoria"las direcciones de los gradientes pasados y tiene una tasa de aprendizaje adaptativa que es individual para cada parámetro.

Estos algoritmos, entre otros, ajustan los pesos de las conexiones entre neuronas, w_i, mediante el algoritmo de retropropagación del error (*backpropagation*), que calcula el gradiente de la función de pérdida con respecto a cada peso de la red y lo actualiza en la dirección que reduce el error.

Las redes neuronales se han vuelto un modelo de aprendizaje automático muy popular por varias razones.

- Alta capacidad: ANN pueden modelar relaciones extremadamente complejas que otros algoritmos no son capaces de capturar.
- Flexibilidad: La arquitectura variables de la red, número de capas, neuronas, funciones de activación, puede adaptarse a la complejidad del problema.
- Gran rendimiento: Suelen alcanzar un resultado preciso en muchas tareas, sobre todo cuando se dispone de grandes volúmenes de datos.

Aún así, también tienen algunos inconvenientes que no se deben pasar por alto. Se deben utilizar técnicas de regularización, como Dropout, para evitar el sobreajuste de la red y que sea capaz de generalizar, no solo aprender los datos de entrenamiento. El entrenamiento puede ser muy lento y requerir de hardaware especializado, y algunas veces costoso, como son las GPUs. Son considerados a menudo modelos de caja negra. Esto quiere decir que son poco interpretables ya que es difícil de entender cómo llegan a la predicción final realizada.

Técnicas y herramientas

Este capítulo tiene como objetivo presentar las principales técnicas metodológicas y herramientas utilizadas en el desarrollo del proyecto. Se describen tanto los enfoques de modelado aplicados como las bibliotecas y entornos de desarrollo empleados. Cuando ha sido pertinente, se han considerado distintas alternativas, valorando sus características más relevantes y justificando las decisiones adoptadas en función de criterios como precisión, facilidad de uso, compatibilidad con el entorno de trabajo y adecuación a los objetivos del estudio. No se pretende ofrecer una revisión exhaustiva de cada opción, sino proporcionar una visión general de los fundamentos necesarios, junto con referencias clave que permitan al lector ampliar su comprensión en caso de interés.

4.1. Técnicas de validación

Una parte fundamental en cualquier proyecto de aprendizaje automático es la correcta validación de los modelos construidos, ya que de ella depende la fiabilidad de las métricas de evaluación y la capacidad de generalización del sistema.

Dos de las técnicas más habituales en este contexto son la división en conjuntos de entrenamiento, validación y prueba (*train/validate/test* split) y la validación cruzada (*cross-validation*). A continuación se describen sus principales características ventajas e incovenientes.

Train/Validate/Test Split

Esta estrategia consiste en particionar el conjunto de datos en tres subconjuntos mutuamente excluyentes como se observa en la Figura 4.4:

- **Train set**: Este conjunto se utiliza para ajustar los parámetros del modelo.

- **Validation set**: Este conjunto se encarga de seleccionar hiperparámetros, prevenir el sobreajuste y tomar decisiones sobre la arquitectura o configuración concreta del modelo

- **Test set**: Este conjunto se reserva para la evaluación final, simulando el comportamiento del modelo en datos nunca antes vistos.

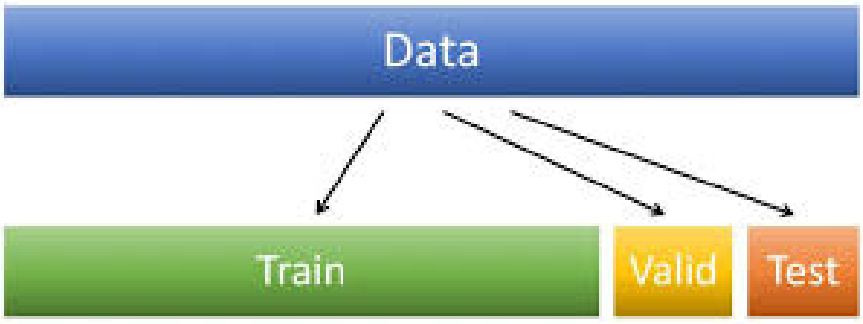


Figura 4.4: División en entrenamiento, validación y test

Esta técnica permite separar claramente el proceso de entrenamiento y evaluación, lo que resulta especialmente útil cuando se dispone de suficientes datos o cuando se utilizan varios conjuntos con propósitos diferenciados.

Cross-validation

La validación cruzada es una técnica robusta que consiste en dividir el conjunto de datos en k subconjuntos o *folds*. De esta forma se entrena el modelo k veces, usando en cada iteración $k - 1$ particiones para el entrenamiento y la partición restante para la validación (Figura 4.5). Al final, se promedian las métricas obtenidas para otorgar un resultado final.

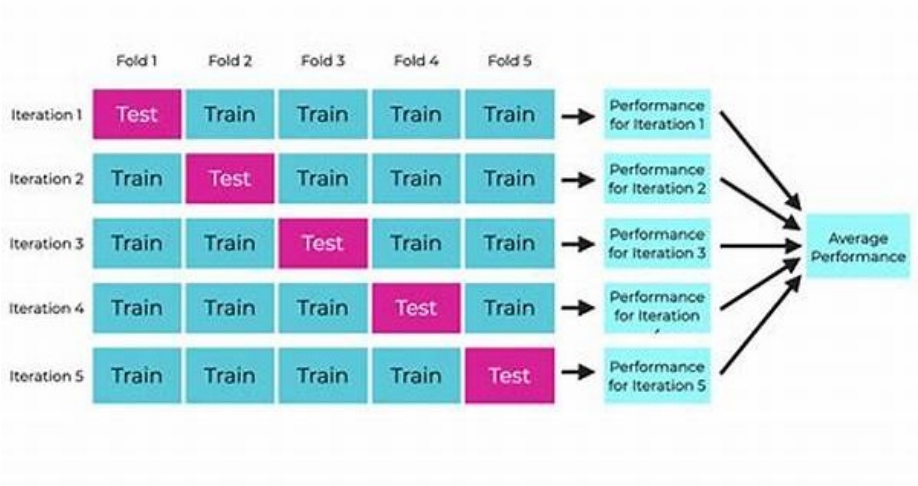


Figura 4.5: Validación cruzada. [8]

Este enfoque es útil cuando el tamaño del dataset es limitado. Se aprovecha de forma más eficiente todo el conjunto de datos y es capaz de producir estimaciones más estables y menos dependientes de una sola partición. Sin embargo, esta técnica presenta un mayor coste computacional, al entrenar el modelo varias veces, y puede conllevar una mayor complejidad en su implementación. Estos factores se acentúan más con modelos pesados o que requieren de varias etapas de procesamiento.

4.2. Bases de datos

Para la construcción y validación de los modelos de predicción de diferencia de color, se ha seleccionado un conjunto representativo y diverso de bases de datos experimentales reconocidas en la literatura especializada. Estas bases de datos aportan información de alta calidad y relevancia, necesaria para evaluar con rigor los métodos propuestos y comparar su desempeño con fórmulas establecidas como CIEDE2000. A continuación, se describen las bases de datos utilizadas:

- **COMCorreg-DiffColor**: Es una base de datos orientada a la evaluación de diferencias de color percibidas por humanos. Su principal propósito es proporcionar información experimental que permita comparar, validar o mejorar fórmulas de diferencia de color como CIEDE2000.

- **BIGC**: Son las siglas de Beijing Institute of Graphic Communication. Esta base de datos contiene mediciones detalladas de diferencias de color bajo distintas condiciones [7].
- **lcamDIN99WDC (Icam)**: Desarrollada en el departamento de Ingeniería de Materailes en Technical University of Liberec por el investigador Michal Vik. Este conjunto de datos se centra en aportar información específica sobre percepciones de diferencia de color en aplicaciones industriales.
- **MMB**: denominada Morley-Munn-BIllmeyer. Es una base ampliamente citada en estudios de colorimetría que ofrece datos experimentales para la evaluación de modelos de diferencia de color [10].
- **WangHan**: Desarrollada en Colour, Imaging and Design Research Centre de University of Leeds por el investigador WangHan. Esta base contribuye con datos obtenidos bajo condiciones controladas para el análisis de diferencias cromáticas.

A partir de este conjunto, tengo una base diversa y suficiente para entrenar y validar distintos modelos y comprobar su desempeño a la hora de predecir la diferencia de color, permitiéndome explorar si los modelos mejoran las fórmulas actuales. Esta diversidad permite explorar la capacidad de generalización de los modelos y su potencial mejora sobre los métodos actuales para el cálculo de diferencias de color.

Finalmente, en este trabajo se ha optado por una estrategia basada en división en conjuntos de entrenamiento, validación y prueba, tras hacer pruebas con ambas técnicas, debido a varias razones técnicas y experimentales:

1. Disponibilidad de bases de datos separadas: He utilizado distintas bases de datos que han sido obtenidas con distintos propósitos. La base de datos COMCorreg la he empleado como conjunto de entrenamiento y validación. Esta decisión la he tomado debido a su amplitud, calidad y cobertura perceptual que la convierte en un recurso adecuado y completo para el aprendizaje del modelo. Por otra parte, he reservado los conjuntos BIGC, IcamDIN99DWC, MMB y WangHan exclusivamente para pruebas sobre datos no vistos, permitiendo así una evaluación más realista y generalizable al rendimiento del modelo.
2. Simulación de condiciones reales: Este enfoque permite simular un escenario práctico donde el modelo siempre se entrena sobre un conjunto

de datos específico y completo, y luego se prueba sobre observaciones completamente nuevas. Esto sigue las líneas del proyecto de evaluar la generalización a diferentes contextos.

3. Coste computacional y escalabilidad: Dado que algunos modelos utilizados, como las redes neuronales, requieren tiempos de entrenamiento considerables, el uso de validación cruzada completa resultaría más costoso computacionalmente, sin aportar grandes beneficios significativos.

4. Claridad y separación de roles: El uso de conjuntos completamente disjuntos para entrenamiento y pruebas proporciona una separación conceptual clara y facilita la trazabilidad de los experimentos realizados.

4.3. Bibliotecas y frameworks

Durante el desarrollo del proyecto, se han empleado diversas herramientas y librerías especializadas en aprendizaje automático y ciencia de datos, cada una seleccionada por su idoneidad para resolver aspectos concretos del problema planteado. A continuación, se describen las más relevantes, así como una comparación entre las alternativas consideradas y una justificación de las elecciones realizadas.

Scikit-learn

Scikit-learn es una de las bibliotecas más consolidadas y utilizadas en el ámbito del aprendizaje automático clásico en Python. Ofrece una amplia gama de algoritmos de clasificación, regresión, clustering o reducción de dimensionalidad, además de herramientas para la evaluación y validación de modelos o escalado de datos [14].

En este proyecto, scikit-learn ha sido utilizada para implementar y entrenar modelos como *Random Forest* y *Support Vector Regressor (SVR)*, así como para tareas auxiliares como la división de datos en conjuntos de entrenamiento y validación, y el cálculo de métricas de evaluación. Su sintaxis clara, extensa documentación y comunidad y alta interopaerabilidad con otras bibliotecas del ecosistema Python justifican su elección.

XGBoost

XGBoost (eXtreme Gradient Boosting) es una biblioteca optimizada para el entrenamiento eficiente de modelos de árboles potenciados mediante gradiente. Se ha consolidado como una herramienta de referencia en competiciones de ciencia de datos debido a su capacidad para ofrecer altos niveles de rendimiento y generalización en tareas con datos tabulares.

Dentro del presente trabajo, se ha construido el modelo XGBRegressor, aprovechando su capacidad de controlar el sobreajuste mediante técnicas como el shrinkage, la selección de submuestras (*subsampling*) y el control de la profundidad de los árboles. Su rendimiento superior frente a otras técnicas de regresión, así como su eficiencia computacional y facilidad de integración con scikit-learn, justifican su inclusión como uno de los modelos principales evaluados.

TensorFlow y Keras

TensorFlow es una plataforma de desarrollo de modelos de aprendizaje profundo mantenida por Google. Junto con Keras, su interfaz de alto nivel, ofrece un entorno robusto, modular y altamente escalable para la creación de redes neuronales artificiales [1].

En este proyecto se ha empleado TensorFlow con Keras para el diseño y entrenamiento de modelos de redes neuronales artificiales. La elección de esta herramienta se fundamenta en su madurez, la amplitud de su documentación, su integración con utilidades para el seguimiento del entrenamiento y la facilidad con la que permite prototipar modelos de forma rápida. La curv de aprendizaje relativamente baja de Keras, en comparación con otras librerías más flexibles pero menos accesibles, fue un factor determinante para su elección.

PyTorch

PyTorch es una biblioteca de aprendiaje profundo desarrollada por Facebook, ampliamente adoptada en el ámbito académico por su enfoque dinámico y flexible en la construcción de redes neuronales [13].Su modelo de ejecución *eager* resulta especialmente útil en contextos donde se requiere un control detallado del flujo de datos y los cálculos.

Dentro del proyecto, PyTorch se ha integrado como herramienta complementaria para tareas experimentales relacionadas con redes neuronales, aprovechando su sintaxis intuitiva y su eficiente gestión del entrenamiento

con GPUs. Su creciente popularidad en el ámbito académico y profesional respalda su inclusión como entorno de referencia para aprendizaje profundo.

Aspectos relevantes del desarrollo del proyecto

5.1. Normalización de los datos de entrada

En lugar de realizar una transformación explícita de los colores se optó por una representación alternativa basada en diferencias cuadráticas entre los componentes de cada par de colores y la media aritmética.

Cada observación del conjunto de datos está compuesta por dos colores como se observa en Tabla 3.1. En lugar de usar estos 6 parámetros como entrada a los distintos modelos, he normalizado estas variables utilizando las diferencias al cuadrado.

$$\Delta L^{*2} = (L_1^* - L_2^*)^2 \qquad \Delta a^{*2} = (a_1^* - a_2^*)^2 \qquad \Delta b^{*2} = (b_1^* - b_2^*)^2$$

Además de esto, para que el modelo sepa por que zona del espacio de color se encuentran los puntos, ya que puede no ser lo mismo para un tono rojizo que para un tono verdoso, el modelo también recibe la media de cada magnitud.

$$\bar{L^*} = \frac{L_1^* + L_2^*}{2} \qquad \bar{a^*} = \frac{a_1^* + a_2^*}{2} \qquad \bar{b^*} = \frac{b_1^* + b_2^*}{2}$$

Otro aspecto a tener en cuenta a la hora del entrenamiento, las etiquetas, ΔV, eran pasadas al cuadrado para que el modelo tuviese más fácil para encontrar relaciones entre los datos de entrada y los datos de salida. Es decir para el ejemplo de la Tabla 3.1, los datos que recibiría el modelo de entrenamiento serían los de la tabla Tabla 5.2

Pair	ΔL^{*2}	$\bar{L}^{*}$	Δa^{*2}	$\bar{a}^{*}$	Δb^{*2}	$\bar{b}^{*}$	ΔV^{2}
50	0,001	61,912	2,378	−9,787	6,843	39,299	2,369
51	0,005	61,964	3,408	−9,635	0,055	40,490	1,860

Tabla 5.2: Muestra de datos de entrada para el modelo de entrenamiento

5.2. Optimización de arquitectura e hiperparámetros

Uno de los aspectos fundamentales en el desarrollo de modelos de aprendizaje automático es la adecuada configuración de los hiperparámetros y arquitectura del modelo, ya que estos influyen directamente en la capacidad predictiva, la generalización y la eficiencia computacional de los modelos. Con el objetivo de automatizar y optimizar este proceso, en este proyecto he utilizado la biblioteca **Optuna**.

Optuna es un software de código abierto diseñado específicamente para la optimización automática de hiperparámetros mediante técnicas de *AutoML* ([2]). Su filosofía de diseño se basa en una construcción definida por el usuario mediante una función objetivo, lo que permite una gran flexibilidad en la definición del espacio de búsqueda, así como en la lógica de entrenamiento y evaluación de modelos.

El proceso de optimización sigue una estrategia de búsqueda bayesiana mediante algoritmos de tipo *Tree-structured Parzen Estimator* (TPE), los cuales permiten seleccionar de forma inteligente los siguientes conjuntos de hiperparámetros a probar, a partir del histórico de evaluaciones anteriores. Esto hace que la búsqueda sea más eficiente que un barrido sistemático como *grid search* o incluso aleatorio (*random search*).

Para cada uno de los modelos implementados -principalmente Random Forest, XGBoost y redes neuronales, y en menor medida SVR- se ha definido una función objetivo encargada de:

- Probar una combinación de hiperparámetros propuesta por el motor de Optuna, dentro de un rango limitado.
- Entrenar un modelo sobre el conjunto de entrenamiento.
- Evaluar su rendimiento en el conjunto de validación. A diferencia de enfoques tradicionales donde se emplean métricas estándar como

el error cuadrático medio, en este caso se ha seleccionado el índice STRESS como métrica de evaluación. Esta decisión ha sido tomada debido a la idoneidad de STRESS en contextos relacionados con la percepción de diferencias de color.

- Devolver el valor de dicha métrica al motor de optimización.

Un ejemplo típico de los hiperparámetros optimizados incluye:

- En Random Forest: número de estimadores y la profundidad máxima de cada estimador.
- EN SVR: la penalización C, el parámetro de insensibilidad ϵ y el tipo de núcleo.
- En redes neuronales: el número de capas ocultas, el número de neuronas por capa, la función de activación, la tasa de aprendizaja, el número de épocas de entrenamiento y el tipo de optimizador.

La utilización de Optuna me ha permitido encontrar configuraciones de alto rendimiento con un menor coste computacional, mejorando la eficiencia y rapidez del desarrollo y favoreciendo la reproducibilidad de los experimentos mediante el control de semillas.

Random Forest

En el caso del modelo *Random Forest*, implementado mediante la biblioteca *scikit-learn*, el proceso de optimización realizado con *Optuna* ha permitido identificar la configuración más adecuada para este problema específico. La combinación que ha ofrecido el mejor rendimiento en términos del índice STRESS corresponde a un número de estimadores (`n_estimators`) igual a 75 y una profundidad máxima del árbol (`max_depth`) de 13.

Esta configuración refleja un equilibrio entre capacidad de aprendizaje y control del sobreajuste. Un número de estimadores suficientemente elevado garantiza la diversidad entre los árboles individuales, mientras que una profundidad moderada limita la complejidad de cada árbol favoreciendo una mejor generalización.

Para asegurar la reproducibilidad de los resultados, se ha fijado una semilla aleatoria concreta (`random_state`). Esta práctica es especialmente relevante en métodos de tipo ensemble, dado que el ordenamiento y selección de subconjuntos aleatorios puede influir en el comportamiento del modelo final.

SVR

En el caso del modelo *Support Vector Regressor*, el proceso de optimización ha identificado como combinación óptima el uso de un núcleo polinómico (`kernel="poly"`), con un parámetro de penalización `C` aproximado a 0,011 y un valor de insensibilidad `epsilon` aproximado a 0,005.

El uso del núcleo polinómico permite modelar relaciones no lineales entre las características del color y la diferencia perceptual, lo que resulta apropiado en un contexto donde las transformaciones perceptuales no son lineales. La configuración elegida ofrece un compromiso adecuado entre precisión y complejidad, sin sobreajustar a los datos de entrenamiento.

El modelo SVR con esta configuración se ha evaluado sistemáticamente sobre los conjuntos de datos de validación y test, siguiendo el mismo procedimiento experimental empleado para los demás modelos.

XGBoost

Para el modelo *XGBoost*, se ha utilizado la implementación oficial proporcionada por la biblioteca `xgboost`, en combinación con la optimización de hiperparámetros mediante *Optuna*. El mejor rendimiento se ha obtenido con los siguiente valores:

- `n_estimators = 100`
- `max_depth = 3`
- `learning_rate = 0.1`
- `subsample = 0.8`

Esta configuración proporciona una estrategia conservadora, con árboles de baja profundidad y una tasa de aprendizaje moderada, lo cual permite una optimización más estable y una mejor capacidad de generalización. El subsampleo parcial introduce aleatoriedad en el proceso de entrenamiento, reduciendo el riesgo de sobreajuste.

Este modelo se ha entrenado y validado de forma consistente sobre los mismos conjuntos utilizados para el resto de modelos, lo que permite una comparación directa de su rendimiento.

ANN implementada en Tensorflow

En el caso de la red neuronal implementada con *TensorFlow*, el modelo óptimo corresponde a una red secuencial compuesta por cuatro capas densas: una primera capa de 6 neuronas con las características de entrada, una segunda capa de 64 neuronas con activación ReLU, una tercera capa de 32 neuronas también con ReLU, y una capa de salida con una única neurona para la regresión.

El modelo ha sido compilado con el optimizador `Adam` y la función de pérdida `mse` (error cuadrático medio), entrenado durante 100 épocas con un tamaño de batch de 16. Se ha establecido una semilla para garantizar la reproducibilidad de los resultados.

Esta arquitectura proporciona un balance entre expresividad y simplicidad, siendo capaz de capturar relaciones no lineales sin incurrir en sobreajuste, gracias al uso de técnicas como *early stopping*.

ANN implementada en PyTorch

La arquitectura definida con *PyTorch* consiste en una red neuronal multicapa (`MLPRegressor`) de cuatro capas ocultas: 110, 116 y 121 neuronas respectivamente, todas con activación ReLU, seguidas de una capa de salida lineal. Este diseño se ha determinado empíricamente mediante búsqueda de hiperparámetros y ha demostrado un alto rendimiento de predicción.

La red se entrena utilizando el optimizador `Adam` con una tasa de aprendizaje ajustada a `0.0061`, durante 100 épocas y con un tamaño de batch de 64. Como función de pérdida se ha empleado el error cuadrático medio (`MSELoss`).

Este modelo ha sido entrenado sobre GPU utilizando la biblioteca `torch`, mostrando un rendimiento competitivo respecto a otras aproximaciones, particularmente en datasets donde se requiere alta capacidad de representación.

Resumen de hiperparámetros óptimos

En la Tabla 5.3 se puede observar un resumen de la arquitectura e hiperparámetros seleccionados para las fases de entrenamiento y test posteriores.

Modelo	Hiperparámetros óptimos
Random Forest	`n_estimators = 75`, `max_depth = 13`, `random_state = 42(fijo)`
SVR	`kernel = "poly"`, `C = 0.011`, `epsilon = 0.005`
XGBoost	`n_estimators = 100`, max_depth = 3, `learning_rate = 0.1`, `subsample = 0.8`
Red neuronal (TensorFlow)	arquitectura = [6, 64-ReLU, 32-ReLU, 1], `opt = Adam`, `loss = MSE`, `epochs = 100`, `batch_size = 16`
Red neuronal (PyTorch)	arquitectura = [6, 110-ReLU, 116-ReLU, 121-ReLU, 1], `opt = Adam`, `lr = 0.0061`, `loss = MSE`, `epochs = 100`, `batch_size = 64`

Tabla 5.3: Resumen de hiperparámetros óptimos para cada modelo.

5.3. Evaluación segmentada de los resultados por rango de ΔV

Para realizar un análisis exhaustivo del rendimiento de los modelos de diferencia de color, no basta con limitarse a una métrica global agregada, ya que no todas las predicciones presentan la misma dificultad perceptual. El comportamiento de una fórmula puede variar significativamente dependiendo de la magnitud de la diferencia de color que se esté evaluando. Una fórmula puede ser muy precisa para diferencias pequeñas y sutiles, pero fallar en la predicción de diferencias grandes, o viceversa. Por este motivo, en este trabajo se llevará a cabo una evaluación estratificada, dividiendo los pares de colores en tres rangos distintos según la magnitud de la diferencia de color percibida por los observadores (ΔV) [3].

Este enfoque segmentado permitirá obtener una comprensión más profunda y matizada del rendimiento del modelo en diferentes contextos de aplicación. Los rangos definidos para el análisis son los siguientes:

- **Diferencias cercanas al umbral** $\Delta V < 1$: Este rango incluye diferencias de color muy pequeñas, aquellas que se encuentran en el umbral de la percepción humana. La evaluación en esta zona es de máxima importancia para apliaciones de control de calidad industrial, donde el objetivo es mantener una consistencia de color tan alta que las variaciones sean prácticamente imperceptibles. El rendimiento de un modelo en este intervalo indica su capacidad para establecer tolerancias colorimétricas precisas y fiables.

- **Diferencias suprathreshold pequeñas y medianas** $1 \leq \Delta V \leq 5$: Este rango abarca las diferencias de color que son claramente perceptibles, pero que no se consideran extremadamente grandes. Representa el escenario más común en muchas apliaciones industriales y de diseño, donde se comparan muestras con variaciones de color típicas. El análisis de este intervalo es crucial, ya que muchos conjuntos de datos experimentales, como los utilizados en el desarrollo de nuevas métricas, se concentran en estas magnitudes.
- **Diferencias suprathreshold grandes** $\Delta V > 5$: Este último rango corresponde a las diferencias de color grandes y evidentes, que cualquier observador puede identificar sin dificultad. Aunque el ajuste de estas diferencias no suele ser el objetivo principal en el control de calidad, su correcta modelización es importante para otras aplicaciones como la armonía de color, el diseño de interfaces o la recuperación de imágenes por contenido. La capacidad de un modelo para escalar correctamente estas grandes distancias perceptuales es un indicador de su robustez y de la validez de su espacio de color subyacente.

El análisis independiente en cada uno de estos tres rangos permitirá identificar las fortalezas y debilidades específicas del modelo, proporcionando una visión completa de su aplicabilidad práctica.

Resultados

El desarrollar de este Trabajo de Fin de Máster ha permitido explorar el potencial del aprendizaje automático como herramienta para la predicción de color percibida por el ser humano, un fenómeno de naturaleza compleja y subjetiva que tradicionalmente se ha abordado mediante fórmulas matemáticas estandarizadas. A lo largo del proyecto se han diseñado, implementado y evaluado distintos modelos supervisados con el objetivo de aproximar la percepción cromática desde un enfoque basado en datos psicofísicos.

6.1. Análisis comparativo entre modelos

Uno de los principales objetivos del proyecto ha sido evaluar la capacidad predictiva de distintos modelos de regresión supervisada aplicados al problema de la estimación de la diferencia de color percibida por humanos. Para ello, se han entrenado modelos como Random Forest, Support Vector Regressor, XGBoost y redes neuronales artificiales, utilizando como base de datos extensa y correctamente etiquetada como conjunto de entrenamiento. En esta sección se presentan los resultados cuantitativos obtenidos, comparando el rendimiento de los diferentes mediante distintas métricas.

	BIGC	**ICAM**	**MMB**	**WangHan**
Random Forest	0.225	0.522	0.370	0.308
XGBoost	0.375	0.337	0.193	0.376
Red neuronal TensorFlow	0.383	0.337	0.193	0.382
Red neuronal PyTorch	0.285	0.276	0.169	0.230

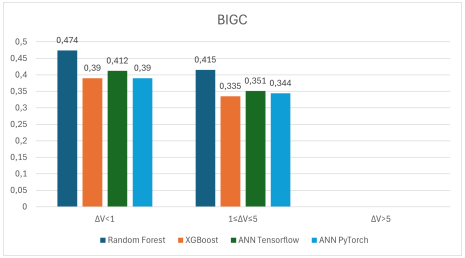
Tabla 6.4: STRESS obtenidos por distintos modelos en los conjuntos de datos de test.

Como se observa en la Tabla 6.4, el rendimiento de los modelos varía significativamente en función del conjunto de datos utilizado, lo cual sugiere diferencias notables en la distribución de los ejemplos. No obstante, pueden empezar a extraerse algunas conclusiones generales.

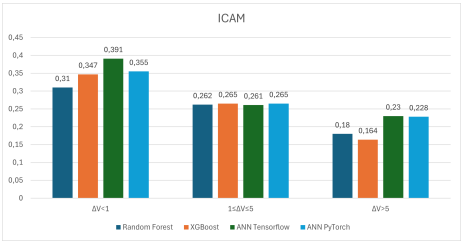
El modelo basado en Random Forest sufre bastantes variaciones entre distintas bases de datos, 0,179 de WangHan frente a 0,522 en ICAM, lo que evidencia una capacidad limitada para generalizar aquellos conjuntos que no son parecidos a los de entrenamiento. En contraste, XGBoost y la red neuronal implementada con Tensorflow presentan un rendimiento más estable entre bases de datos, lo que sugiere una mayor robustez. Sin embargo, el modelo más competitivo en STRESS global es la red neuronal entrenado con PyTorch, que alcanza los mejores resultados en tres de los cuatro conjuntos de test.

En conjunto, si bien los valores de STRESS absolutos no son directamente comparables con escalas de percepción humana, sí permiten establecer una jerarquía relativa de rendimiento. Estos resultados, no obstante, requieren ser complementados con un análisis segmentado por rango de diferencias perceptuales —como se presenta en la Sección 5.3— para obtener una visión más granular y ajustada al contexto de uso real de estos modelos.

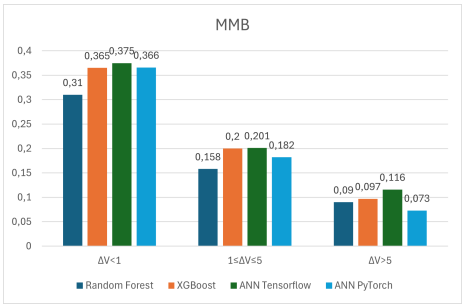
Para evaluar la capacidad de los modelos de aprendizaje automático en la predicción de diferencias de color perceptuales, se analizaron sus rendimientos segmentando los errores por tramos de diferencia visual: $\Delta V < 1$, $1 \leq \Delta V \leq 5$ y $\Delta V > 5$. Esta segmentación permite entender en qué regiones perceptuales los modelos fallan más o menos, lo cual es clave dado que la percepción humana es especialmente sensible en el entorno de ΔV bajos.



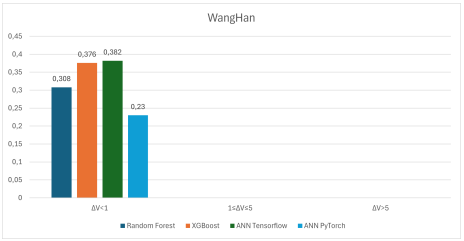
(a) BIGC



(b) ICAM



(c) MMB



(d) WangHan

Figura 6.6: STRESS segmentado por ΔV de cada modelo en los diferentes conjuntos de datos. Para una versión mas detalla consulte el Apéndice A

Los resultados que se muestran en la Figura 6.6 presentan una comparativa de rendimiento evaluados en los cuatro conjuntos de test. El rendimiento óptimo de cada modelo depende en gran medida del conjunto de datos y el rango ΔV analizado. Algunos de los conjuntos, como WangHan, no tienen valores en todos los rangos ΔV analizados.

Los modelos de redes neuronales artificiales (ANN) se posicionan frecuentemente como los de mejor rendimiento, especialmente en el rango más bajo ($\Delta V < 1$). Las redes entrenadas con Tensorflow presentan una ligera ventaja y mayor consistencia en varios casos, sin embargo, ANN PyTorch destaca con el mejor STRESS absoluto en el rango $\Delta V > 5$ para el dataset MMB, lo que sugiere una gran capacidad para manejar predicciones con grandes desviaciones.

XGBoost ofrece un modelo robusto y consistente en todos los datasets. Aunque no siempre logra el mejor resultado, se mantiene muy competitivo frente a las redes neuronales y supera consistentemente a Random Forest. Su rendimiento balanceado lo convierte en una opción muy fiable, destacando por ejemplo para ICAM en el rango $\Delta V > 5$.

Random Forest ofrece un buen rendimiento general pero le falla la capacidad de generalización como se puede observar con el mal rendimiento que tiene en el conjunto BIGC.

6.2. Comparación de redes neuronales frente a CIEDE2000

Más allá de la comparación entre modelos de aprendizaje automático, resulta fundamental valorar hasta qué punto estas aproximaciones pueden superar o complementar las fórmulas matemáticas clásicas utilizadas en la industria para medir diferencias de color. En particular, se ha tomado como referencia la fórmula CIEDE2000, ampliamente aceptada en aplicaciones industriales y normalizada por organismos como la CIE. En esta sección se expone un análisis detallado del modelo que ha ofrecido mejor rendimiento, las redes neuronales (concretamente la entrenada con PyTorch), comparando sus predicciones con las obtenidas mediante CIEDE2000 en distintas bases de datos de test. El objetivo es evaluar si el enfoque basado en datos logra captar mejor las sutilezas de la percepción visual subjetiva.

Rango $\Delta V < 1$

El modelo de red neuronal artificial entrenada en PyTorch demuestra un rendimiento excepcional en el rango $\Delta V < 1$, obteniendo mejores resultados que CIEDE2000 de forma sistemática como se observa en la Figura 6.7, destacando en el conjunto de datos WangHan.

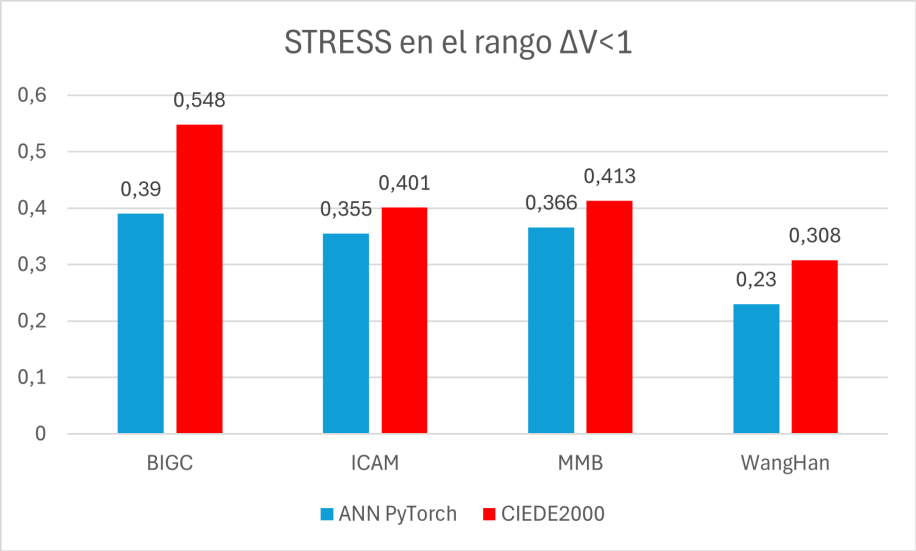


Figura 6.7: STRESS de ANN frente a CIEDE2000 en el rango $\Delta V < 1$

Se puede observar como aunque en el conjunto BIGC el STRESS predicho por el modelo aumenta, también lo hace, y más considerablemente, el STRESS obtenido por la fórmula CIEDE2000. En el conjunto WangHan, los datos se acoplan de manera más correcta a las predicciones hechas por el modelo.

El modelo es capaz de predecir diferencias de color apenas perceptibles por el ojo humano con menor STRESS que la fórmula CIEDE2000 ampliamente utilizada en la industria, lo que indica una alta precisión.

Rango $1 \leq \Delta V \leq 5$

En este rango intermedio, considerado crítico desde el punto de vista de la percepción visual, el rendimiento del modelo ANN de PyTorch presenta una interesante dualidad como se observa en la Figura 6.8. Por un lado, en las bases de datos BIGC y MMB el rendimiento del modelo supera con claridad el rendimiento de CIEDE2000. Con valores de STRESS de 0,344 frente a 0,516 y 0,182 frente 0,379 respectivamente, el modelo basado en aprendizaje profundo se adapta mejor a las características de estas dos bases de datos y refleja mejor la aproximación a las valoraciones humanas. Por otro lado, en la base de datos ICAM ambos modelos presentan un STRESS prácticamente idéntico, 0,265 para la red neuronal frente a 0,265 de ECIEDE2000. En este caso, el STRESS de la fórmula matemática supera en 0,004 al del modelo. Al ser un valor tan pequeño puede considerarse estadísticamente irrelevante.

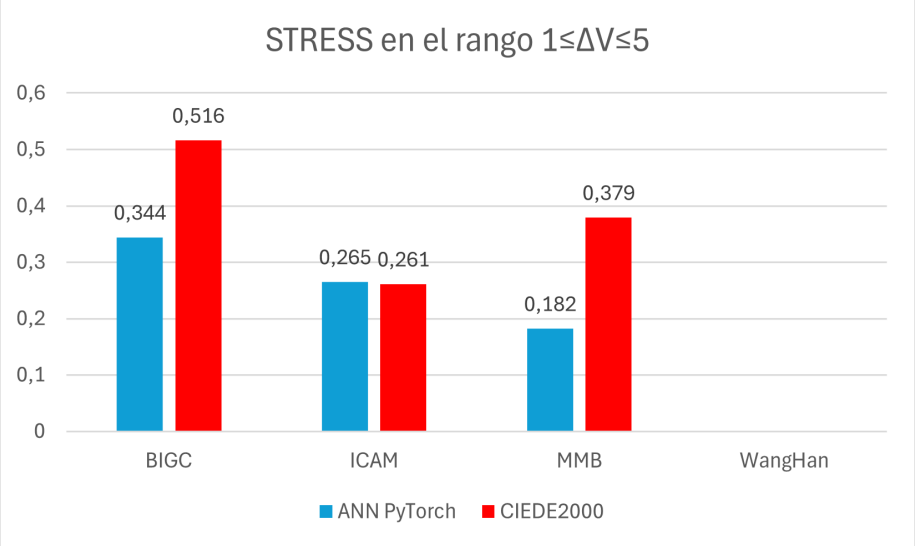


Figura 6.8: STRESS de ANN frente a CIEDE2000 en el rango $1 \leq \Delta V \leq 5$

En conjunto, el modelo de red neuronal artificial supera claramente a CIEDE2000 en dos de las tres bases de datos evaluadas. Este comportamiento evidencia una mayor robustez del modelo propuesto en la estimación de diferencias perceptuales intermedias, lo cual es de particular interés para aplicaciones en las que la fidelidad perceptual es crítica.

Rango $\Delta V > 5$

En este rango, las diferencias visuales son generalmente evidentes, incluso para observadores no entrenados. No todas las bases de datos tienen datos para evaluar este rango como se observa en la Figura 6.9, así que me centraré en el análisis de ICAM y MMB.

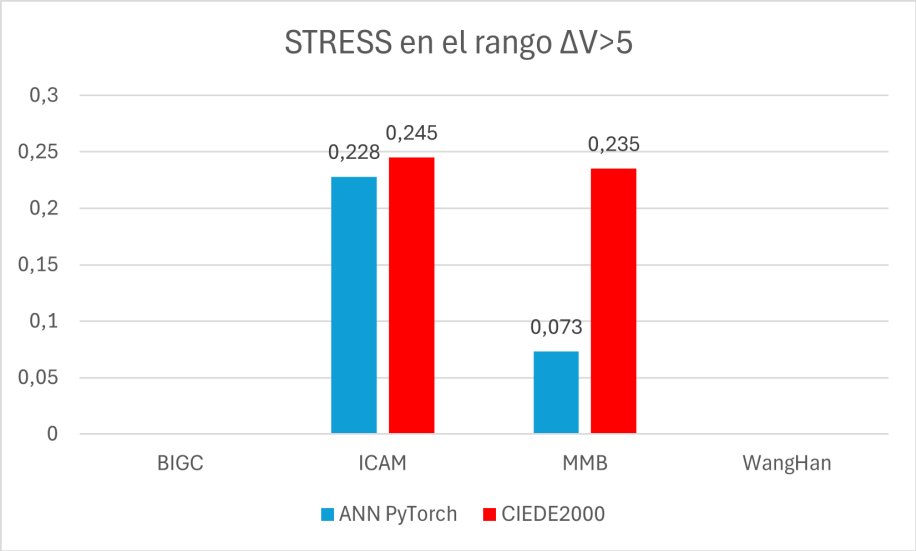


Figura 6.9: STRESS de ANN frente a CIEDE2000 en el rango $\Delta V > 5$

En la base ICAM el modelo obtiene un valor de STRESS ligeramente inferior al alcanzado por CIEDE2000, 0,228 frente a 0,245. Aunque la diferencia no es elevada, sugiere una mejor capacidad de generalización del modelo propuesto en situaciones de alta discrepancia cromática.

Los resultados son más concluyentes en la base de datos MMB. El modelo de ANN es capaz de reducir su STRESS a 0,073, un valor muy por debajo del obtenido por la fórmula matemática CIEDE2000 que se queda en 0,235.

A pesar de los buenos resultados de CIEDE2000, el modelo basado en aprendizaje profundo proporciona una representación más cercana a la percepción humana en escenarios donde las diferencias de color son claramente distinguibles.

Comparación gráfica de predicciones de diferencias

Con el objetivo de evaluar la capacidad de los modelos desarrollados para aproximarse a la percepción visual humana, se presentan a continuación una serie de representaciones gráficas que comparan las predicciones realizadas por distintos enfoques —incluyendo modelos de aprendizaje profundo como ANN PyTorch y métricas tradicionales como CIEDE2000— frente a las valoraciones obtenidas mediante observación humana.

Estas gráficas permiten observar visualmente el grado de correlación entre las predicciones de cada modelo y los valores perceptuales reales,

representados mediante la variable ΔV. En el eje vertical se representa la diferencia de color predicha (ΔE), mientras que el eje horizontal muestra la diferencia visual estimada por los observadores. A través del análisis de las nubes de puntos, las regresiones lineales asociadas y su relación con la diagonal ideal ($x = y$), se pueden identificar patrones de sobreestimación, subestimación o ajuste adecuado por parte de cada método.

En el eje X se representa la variable ΔV que representa la variable predicha por los observadores. En el eje Y se representa la variable ΔE que representa los resultados predichos por el modelo o la fórmula CIEDE2000. Además, se representa las rectas de regresión de cada una de las dos componentes de la gráfica.

Esta comparación visual se complementa con el análisis numérico previo, aportando una perspectiva cualitativa que facilita la interpretación del rendimiento de los modelos en distintos rangos de diferencia perceptual.

A partir de la Figura 6.10, se observa que ambas aproximaciones muestran una tendencia creciente coherente con el incremento de ΔV, lo que refleja una capacidad razonable para capturar los cambios cromáticos. Aunque la pendiente de regresión del modelo de CIEDE2000 (roja) se aproxima más a la línea $x = y$, que sugiere una mejor correspondencia, también presenta una mayor variabilidad y dispersión sobre todo en rangos altos. Por su parte, la regresión de ANN PyTorch (azul) presenta una pendiente más moderada pero tiende a mantener sus predicciones cerca de la recta de dispersión. Esto hace que conforme aumenta ΔV, el modelo tiende a subestimar la diferencia perceptual del observador en la base de datos ICAM.

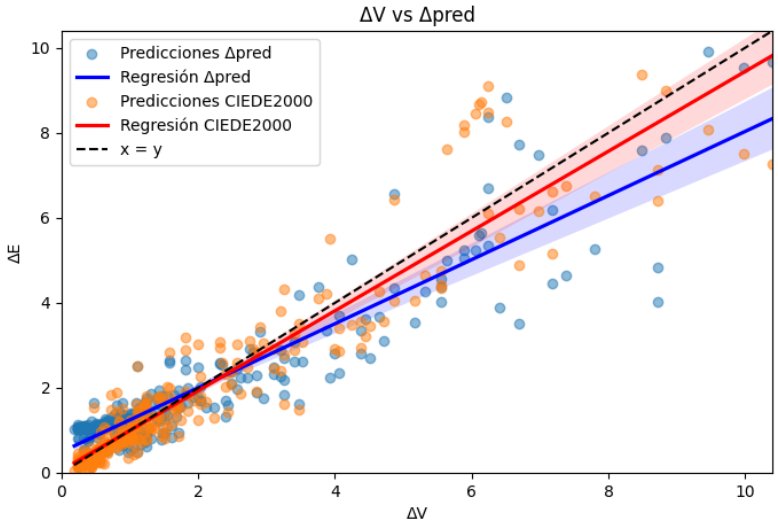


Figura 6.10: Comparación entre las diferencias perceptuales estimadas por observadores humanos (ΔV) y las predicciones obtenidas mediante el modelo ANN PyTorch (azul) y la métrica CIEDE2000 (naranja) en el conjunto ICAM

En conjunto, esta representación evidencia que la red neuronal ofrece predicciones conservadoras pero coherentes mientras que ECIEDE2000 tiende a proporcionar valores más elevados pero con una dispersión mayor. Estas diferencias pueden tener implicaciones prácticas dependiendo del nivel de precisión requerido para una tarea determinada.

En la Figura 6.11 se puede comparar visualmente la capacidad predictiva de cada enfoque, red neuronal frente a CIEDE2000, en el conjunto de datos MMB. Se observa que ambos métodos presentan una tendencia creciente, lo que indica una correlación positiva con los valores subjetivos ΔV. La pendiente de regresión del modelo ANN PyTorch se aproxima en mayor medida a la diagonal $x = y$ en comparación con la recta de regresión obtenida por CIEDE2000. Esto sugiere que, en el contexto del conjunto MMB, el modelo basado en aprendizaje profundo logra una mayor concordancia con las valoraciones humanas.

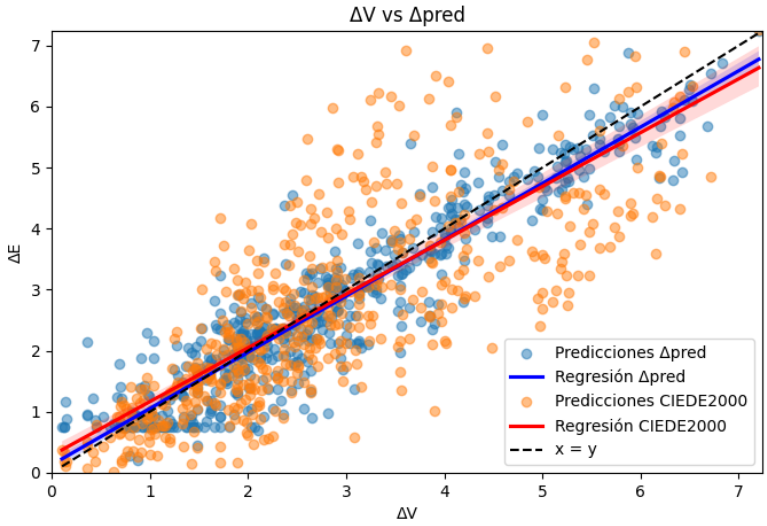


Figura 6.11: Comparación entre las diferencias perceptuales estimadas por observadores humanos (ΔV) y las predicciones obtenidas mediante el modelo ANN PyTorch (azul) y la métrica CIEDE2000 (naranja) en el conjunto MMB

En cuanto a la dispersión, es evidente que las predicciones del modelo de red neuronal artificial presentan una menor variabilidad en torno a la recta de regresión. Esta menor dispersión sugiere una mayor consistencia del modelo al predecir diferencias perceptuales para muestras similares. Las predicciones de CIEDE2000 exhiben una dispersión más elevada, lo que puede interpretarse como una menor capacidad de la fórmula para ajustarse con precisión a la subjetividad humana.

En resumen, para el conjunto MMB, el modelo de aprendizaje profundo muestra una mayor proximidad a la respuesta subjetiva media a la par que una dispersión menor, lo cual refuerza su adecuación como herramienta predictiva en este contexto específico.

Conclusiones y Líneas de trabajo futuras

Este apartado recoge, en primer lugar, las principales conclusiones derivadas del análisis de los resultados obtenidos, tanto en términos de rendimiento cuantitativo de los modelos como de su capacidad para generalizar frente a fórmulas tradicionales de diferencia de color.

Finalmente, se plantean distintas líneas de trabajo futuras que permitirían ampliar, mejorar o transferir los resultados obtenidos a nuevos contextos de aplicación. Estas propuestas se formulan con el objetivo de consolidar la utilidad práctica de los modelos desarrollados, así como de fomentar nuevas investigaciones que profundicen en la intersección entre percepción visual, aprendizaje automático e industria del color.

7.1. Conclusiones

- Mejor rendimiento general con PyTorch (sobresale en WangHan, lo que sugiere que es capaz de detectar bien las diferencias sutiles) - Rendimiento solido y fiable XGBoost - Diferencias grandes son más fáciles de predecir

El presente Trabajo de Fin de Máster ha abordado el estudio, desarrollo y evaluación de modelos predictivos orientados a estimar la diferencia de color percibida por el ojo humano, con el objetivo de analizar su capacidad para mejorar las fórmulas clásicas actualmente utilizadas en colorimetría, como CIEDE2000.

A partir del uso de un conjunto diverso y contrastado de bases de datos experimentales —incluyendo COMCorreg-DiffColor, BIGC, lcamDIN99WDC, MMB y WangHan— se ha logrado entrenar y validar modelos con una

base empírica sólida. Esta diversidad ha permitido evaluar el comportamiento de los modelos en distintos contextos y condiciones de visualización, garantizando una mayor robustez en los resultados obtenidos.

Los modelos desarrollados muestran un desempeño competitivo frente a las fórmulas estándar, y en ciertos casos superan sus resultados en términos de correlación con la percepción humana. Esto sugiere que el uso de técnicas de aprendizaje automático, adecuadamente entrenadas y validadas, puede ofrecer una vía prometedora para la mejora del cálculo de diferencias de color, especialmente en entornos donde la precisión perceptual es crítica.

El modelo que obtuvo el mejor rendimiento fue una red neuronal artificial (ANN) implementada con PyTorch, que superó a la fórmula CIEDE2000 en diversas métricas de evaluación, como la métrica STRESS. Este resultado refuerza el potencial de los enfoques basados en aprendizaje automático para capturar patrones complejos en la percepción del color. El modelo entrenado sobre XGBoost también ofrece un rendimiento sólido y fiable que puede ser útil en ciertas situaciones.

Asimismo, este trabajo ha puesto de manifiesto la importancia de considerar no solo métricas objetivas, sino también la estructura y naturaleza de los datos perceptuales. La interpretación de los resultados obtenidos, tanto en términos cuantitativos como cualitativos, ha evidenciado que aún existe margen de mejora en los modelos tradicionales y que las soluciones basadas en datos pueden complementar o incluso redefinir los enfoques existentes.

En definitiva, este estudio contribuye a la línea de investigación que busca integrar el conocimiento perceptual humano con técnicas computacionales avanzadas, abriendo la puerta a futuras investigaciones que profundicen en el uso de inteligencia artificial para la mejora de métricas psicofísicas en el ámbito de la ciencia del color.

7.2. Trabajo Futuro

A partir del desarrollo realizado en este trabajo, se identifican diversas líneas de investigación que podrían explorarse en el futuro para ampliar y profundizar los resultados obtenidos:

- **Normalización y preprocesamiento avanzado de datos**: Dado que cada base de datos utilizada proviene de experimentos distintos, con posibles diferencias en escalas, métodos de medición y condiciones de observación, sería conveniente establecer un proceso de normalización previo al entrenamiento de los modelos. Esto permitiría una

comparación más justa entre datos heterogéneos y podría mejorar la consistencia y rendimiento de los modelos entrenados.

- **Ampliación del conjunto de datos**: Incorporar nuevas bases de datos, especialmente aquellas con condiciones de visualización más diversas en cuanto a fondos, iluminantes u observadores. Esto permitiría mejorar la generalización de los modelos y validar su robustez en contextos más amplios.
- **Exploración de arquitecturas más complejas**: Aunque el modelo de red neuronal artificial ha mostrado un buen rendimiento, podría explorarse el uso de otras arquitecturas más avanzadas, como redes convolucionales (CNN) o modelos basados en atención.
- **Aplicaciones prácticas en tiempo real**: Estudiar la viabilidad de implementar los modelos entrenados en sistemas de control de calidad industrial o en software de edición de imagen, donde una predicción precisa de la diferencia de color tiene un impacto directo.
- **Comparación con otras métricas**: Evaluar los modelos propuestos frente a métricas recientemente desarrolladas o en desarrollo, como DE2000, o métricas perceptuales específicas de cada industria.

Apéndices

Apéndice A

Gráficas de resultados

A.1. STRESS segmentado por ΔV de cada modelo en los diferentes conjuntos de datos

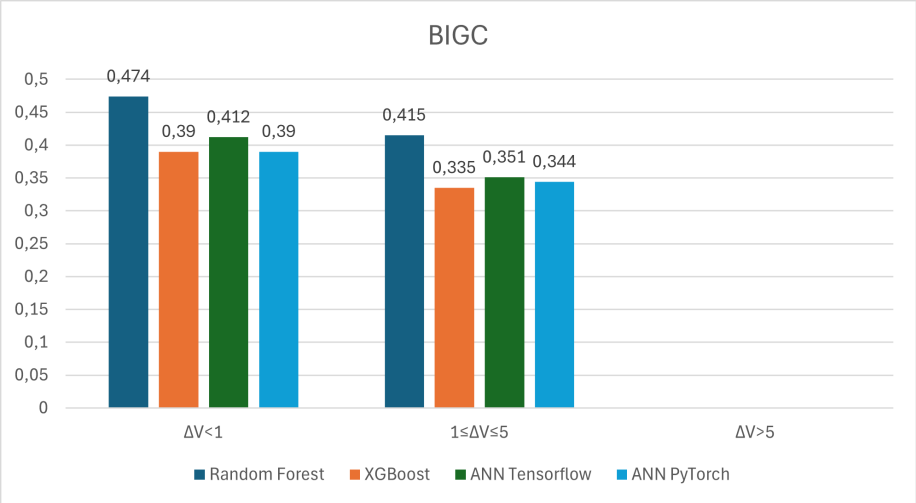


Figura A.1: STRESS de los modelos en BIGC segmentado por ΔV

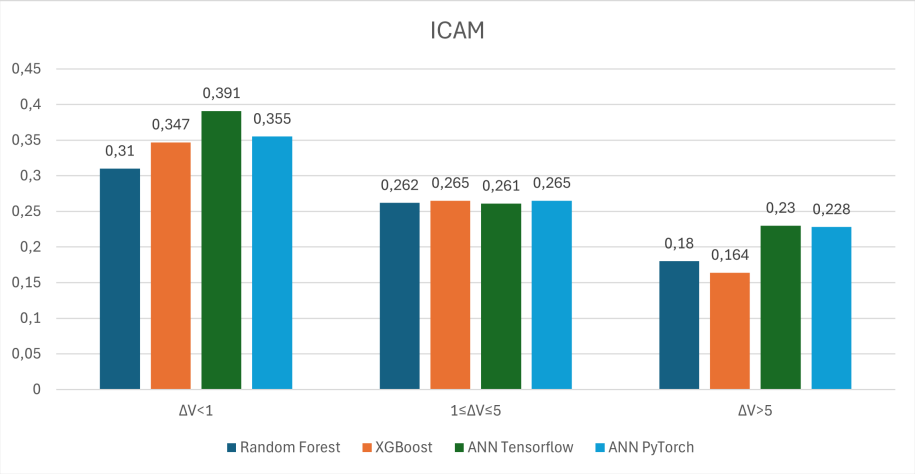


Figura A.2: STRESS de los modelos en ICAM segmentado por ΔV

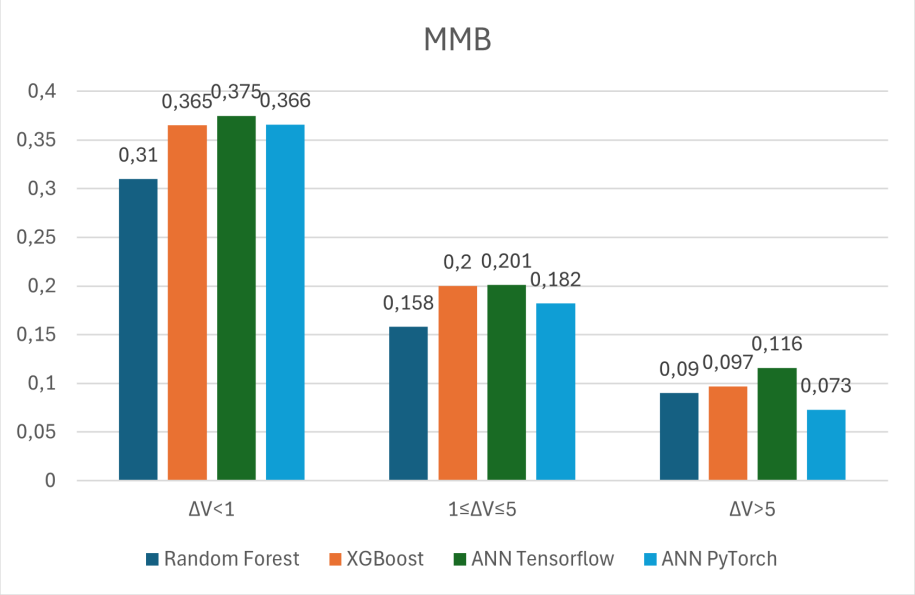


Figura A.3: STRESS de los modelos en MMB segmentado por ΔV

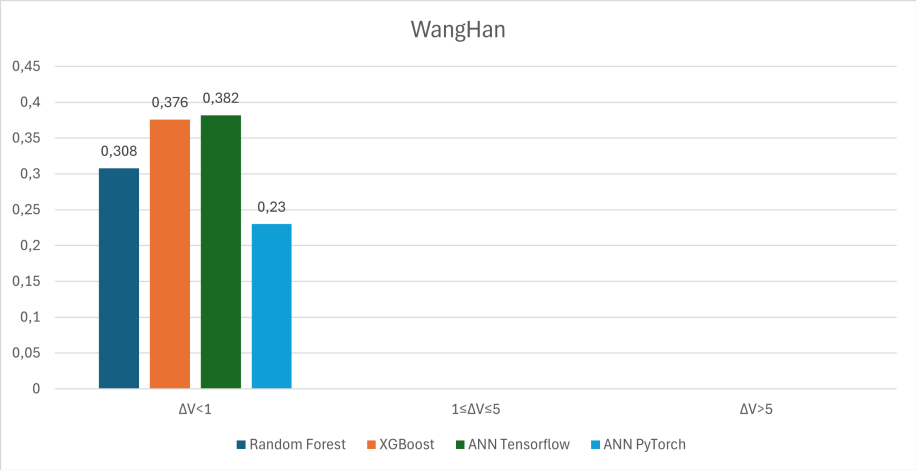


Figura A.4: STRESS de los modelos en WangHan segmentado por ΔV

A.2. Scatter plots de ANN PyTorch frente a CIEDE2000

En esta sección se incluyen los dos scatter plots de los otros conjuntos de datos para su visualización.

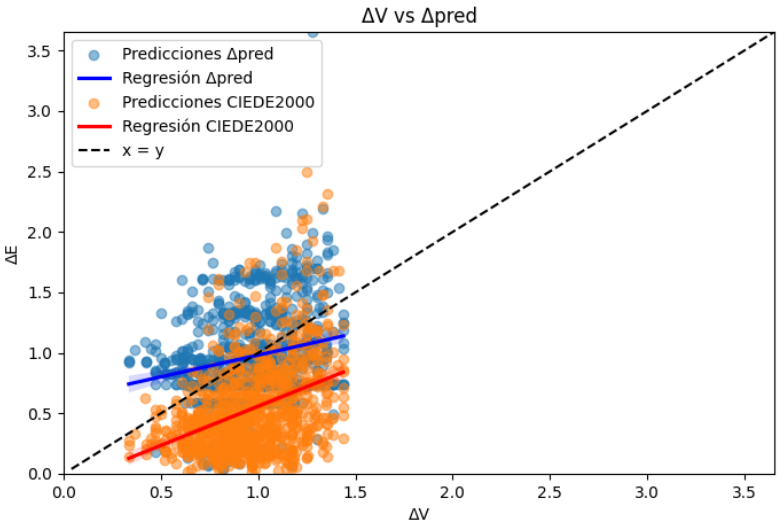


Figura A.5: Comparación entre las diferencias perceptuales estimadas por observadores humanos (ΔV) y las predicciones obtenidas mediante el modelo ANN PyTorch (azul) y la métrica CIEDE2000 (naranja) en el conjunto BIGC

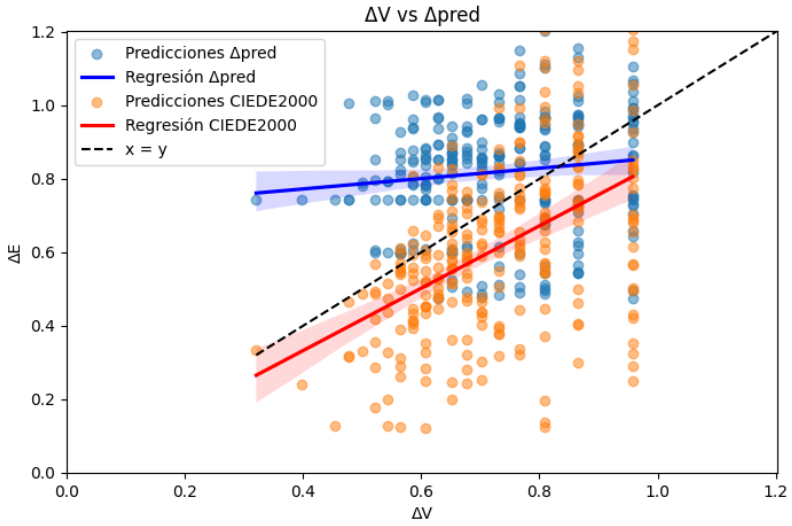


Figura A.6: Comparación entre las diferencias perceptuales estimadas por observadores humanos (ΔV) y las predicciones obtenidas mediante el modelo ANN PyTorch (azul) y la métrica CIEDE2000 (naranja) en el conjunto WangHan

Apéndice B

Manual de usuario

Este apéndice proporciona una visión general del uso del proyecto desarrollado. Para una guía más detallada sobre la instalación, ejecución y configuración del sistema, se recomienda consultar el archivo `README.md` disponible en el repositorio oficial del proyecto en GitHub.

El repositorio puede encontrarse en el siguiente enlace: https://github.com/PabloGallardoFuentes/TFM_PabloGallardo.git

En dicho documento se detallan los requisitos del entorno, los pasos para la instalación de dependencias. Además, se incluyen consideraciones sobre la estructura del proyecto y cómo reproducir los experimentos presentados en esta memoria.

Bibliografía

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 265–283, 2016. URL https://doi.org/10.5555/3026877.3026899.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631. ACM, 2019. doi: 10.1145/3292500.3330701.

[3] Huertas R. Melgosa M. Cui G. García, P.A. Measurement of the relationship between perceived and computed color differences. *Journal Of The Optical Society Of America A*, 2007. URL https://doi.org/10.1364/josaa.24.001823.

[4] V. T. Ha and P. T. Giang. Estudio experimental sobre la predicción de la vida útil restante de baterías de iones de litio basado en tres modelos de regresión para aplicación en vehículos eléctricos. *Applied Sciences*, 13 (13):7660, 2023. URL https://doi.org/10.3390/app13137660. Consultado el 2 de julio de 2025.

[5] Simon Haykin. *Neural Networks and Learning Machines*. Prentice Hall, 3rd edition, 2009.

[6] J. Heaton. Ian goodfellow, yoshua bengio, and aaron courville: Deep learning. *Genetic Programming and Evolvable Machines*, 19(1-2):305–307, 2017. doi: 10.1007/s10710-017-9314-z. URL https://doi.org/10.1007/s10710-017-9314-z.

[7] M. Huang, Y. Xi, J. Pan, R. He, and X. Li. Colorimetric observer categories for young and aged using paired-comparison experiments. *IEEE Access*, 8:219473–219482, 2020. doi: 10.1109/access.2020.3042817. URL https://doi.org/10.1109/access.2020.3042817.

[8] S. Jenkins. Métodos de validación cruzada para un poder predictivo mejorado. Information About Grapix, s.f. URL https://www.grapixai.com/cross-validation-methods-for-enhanced-predictive-power/. Consultada el 31 de junio de 2025.

[9] Just Paint. Delta e: Una clave para entender las lecturas de resistencia a la luz, s.f. URL https://justpaint.org/delta-e/. Consultado el 14 de julio de 2025.

[10] R. G. Kuehni. Color-tolerance data and the tentative cie 1976 l*a*b* formula. *Journal of the Optical Society of America*, 66(5):497, 1976. doi: 10.1364/josa.66.000497. URL https://doi.org/10.1364/josa.66.000497.

[11] Y. Liu, Y. Wang, and J. Zhang. New machine learning algorithm: Random forest. In *Lecture Notes in Computer Science*, pages 246–252. Springer, 2012. doi: 10.1007/978-3-642-34062-8_32. URL https://doi.org/10.1007/978-3-642-34062-8_32.

[12] M. R. Luo, G. Cui, and B. Rigg. The development of the cie 2000 colour-difference formula: Ciede2000. *Color Research & Application*, 26 (5):340–350, 2001. doi: 10.1002/col.1049. URL https://doi.org/10.1002/col.1049.

[13] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Srinath Chilamkurthy, Benoit Steiner, Lu Fang, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 32:8026–8037, 2019. URL https://arxiv.org/pdf/1912.01703.pdf.

[14] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. URL https://doi.org/10.5555/1953048.2078195.

[15] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004. doi: 10.1023/B:STCO.0000035301.49549.88. URL https://doi.org/10.1023/B:STCO.0000035301.49549.88.

[16] G. Wyszecki and W. S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. Wiley, 2nd edition, 2000.