

# Coloración de imágenes

Álvaro Felipe Pérez<sup>1</sup>, Miguel Gómez Prieto<sup>1</sup>, Hugo Gómez-Caraballo Lopez-Romero<sup>1</sup> y Carlos Serrano Pinós<sup>1</sup>

<sup>1</sup>Estudiante CDIA, UPM, Madrid

## Resumen

La colorización automática de imágenes en escala de grises constituye un desafío relevante en visión por computador, al requerir la generación de información cromática plausible a partir de datos incompletos. En este trabajo se presenta un análisis comparativo de diferentes arquitecturas de aprendizaje profundo aplicadas a la colorización de imágenes, incluyendo Autoencoders (AE), Variational Autoencoders (VAE), redes UNet, Generative Adversarial Networks (GAN), modelos de difusión y enfoques guiados por texto. Los experimentos se realizaron sobre el conjunto de datos STL-10 y un dataset complementario de flores, con el objetivo de evaluar la capacidad de cada modelo para reconstruir color de manera realista y coherente con el contenido semántico. Se discuten las ventajas y limitaciones de cada aproximación, así como la influencia de la resolución y el tipo de condicionamiento en la calidad perceptual de los resultados. Los hallazgos contribuyen a comprender el potencial y las fronteras actuales de las técnicas de colorización basadas en deep learning.

**Keywords:** Image colorization, Autoencoder, UNet, Generative Adversarial Network, Diffusion models, Text-guided coloration, Computer vision

## Introducción

La colorización automática de imágenes en escala de grises ha sido un reto clásico en el campo de la visión por computador y el procesamiento digital de imágenes. Desde sus primeras aplicaciones en la restauración de fotografías históricas hasta su uso actual en generación de contenido digital, este problema plantea una dificultad fundamental: reconstruir información cromática que no está presente en los datos originales. La tarea no consiste únicamente en añadir color de manera arbitraria, sino en generar tonalidades plausibles y coherentes con la semántica de la escena, preservando al mismo tiempo la estructura y los detalles de la imagen.

Con el auge del aprendizaje profundo, la colorización ha experimentado un avance significativo. Las arquitecturas modernas permiten aprender representaciones latentes que capturan tanto la estructura como la distribución cromática de los datos, ofreciendo resultados mucho más realistas que los métodos tradicionales basados en reglas o en interpolación manual. En este contexto, diferentes enfoques han sido explorados: desde **Autoencoders (AE)** y **Variational Autoencoders (VAE)**, que aprenden a comprimir y reconstruir imágenes en espacios latentes, hasta **redes UNet**, que aprovechan conexiones de salto para preservar detalles espaciales durante la reconstrucción.

Por otro lado, las **Generative Adversarial Networks (GAN)** han demostrado un gran potencial en la generación de imágenes fotorrealistas, gracias a la dinámica competitiva entre generador y discriminador. En colorización, las GAN permiten producir colores más vivos y naturales, aunque a menudo requieren un entrenamiento cuidadoso para evitar inestabilidades. Más recientemente, los **modelos de difusión** han emergido como una alternativa poderosa, capaces de generar imágenes de alta calidad mediante un proceso iterativo de refinamiento desde ruido hacia datos estructurados. Estos modelos, al operar en espacios latentes comprimidos por un VAE, han mostrado resultados sobresalientes en tareas de síntesis y edición de imágenes, incluyendo la colorización.

Un aspecto adicional que ha cobrado relevancia es el **condicionamiento mediante texto**. La posibilidad de guiar el proceso de colorización con descripciones lingüísticas abre la puerta a aplicaciones creativas y personalizadas, donde el usuario puede especificar estilos, paletas cromáticas o incluso contextos narrativos. Aunque este enfoque introduce complejidad adicional, también amplía el rango de control sobre el resultado final.

En este trabajo se presenta un análisis comparativo de estas arquitecturas aplicadas a la colorización de imágenes, utilizando como base el conjunto de datos **STL-10**, ampliamente empleado en tareas de clasificación y representación visual, y un dataset complementario de flores, que aporta diversidad cromática y riqueza semántica. Además, se explora la hipótesis de que una **segmentación previa a la colorización** puede mejorar la coherencia de los resultados, al proporcionar información estructural más precisa sobre los objetos presentes en la escena.

La introducción de este estudio busca situar la colorización automática dentro del panorama actual de la inteligencia artificial aplicada a la visión por computador, destacando tanto su relevancia práctica como su interés científico. A través de la comparación de AE, VAE, UNet, GAN, modelos de difusión y enfoques guiados por texto, junto con la incorporación de segmentación como paso previo, se pretende ofrecer una visión integral de las técnicas más representativas y de su impacto en la calidad perceptual de las imágenes generadas.

## Antecedentes

### Autoencoder

Comenzaremos nuestro proyecto probando ciertos modelos sencillos para ver su desempeño sobre el dataset STL-10 [1]. En este apartado hablaremos sobre dos diferentes tipos de autoencoders cuya única entrada es una imagen en blanco y negro y su salida es la imagen reconstruida a color.

### U-Net Autoencoder

El modelo implementado sigue una arquitectura tipo U-Net simétrica [2]. A diferencia de un autoencoder secuencial estándar, esta red incorpora conexiones de salto (skip connections) que unen las capas del codificador con sus correspondientes en el decodificador. La arquitectura se compone de tres bloques funcionales detallados a continuación:

- Encoder: recibe un tensor de dimensiones  $(96 \times 96 \times 1)$ . Consta de 3 bloques secuenciales.

Cada bloque aplica una convolución de  $3 \times 3$  con activación ReLU y padding "same" para mantener las dimensiones, seguida inmediatamente de una operación de Max Pooling que reduce el tamaño espacial a la mitad.

- Cuello de botella: comprime al máximo la información ( $12 \times 12$ ). Se utiliza una capa convolucional de 256 filtros para conectar ambos bloques.
- Decoder: aplicamos UpSampling2D para duplicar el tamaño de la imagen cada vez. Uno de sus detalles más importantes es que se concatena información con capas del encoder permitiendo al modelo recordar la ubicación exacta de los bordes y texturas que suelen perderse durante la compresión del Max Pooling.  
Finalmente, se utiliza una capa convolucional con activación sigmoide que proyecta el resultado a los tres canales R, G, B.

## Residual U-Net Autoencoder

Este segundo modelo propuesto sustituye las convoluciones estándar por Bloques Residuales e introduce mecanismos de normalización y submuestreo aprendible [3]. A continuación se detallan sus componentes clave:

- Bloques residuales: Esta es la unidad fundamental de la red. A diferencia de una capa convolucional simple que intenta aprender una función  $H(x)$  directamente, este bloque intenta aprender el residuo  $F(x)$ , de tal forma que la salida sea  $H(x)=F(x)+x$ .
- Encoder: similar al anterior
- Decoder: se diferencia del anterior porque utiliza capas deconvolucionales en lugar de capas upsampling. Las capas deconvolucionales tienen pesos entrenables. La red aprende la mejor manera de aumentar la resolución de la imagen, recuperando texturas y bordes de forma más inteligente que una simple interpolación bilineal.  
Finalmente, se utiliza una capa convolucional con activación sigmoide que proyecta el resultado a los tres canales R, G, B.

## Justificación de modelos

Se descarta el uso de autoencoders secuenciales simples debido a que el proceso de compresión provoca una pérdida irreversible de información espacial. Esto provoca el fenómeno de color bleeding, donde el color se desborda de los contornos al no existir una referencia clara de los bordes durante la reconstrucción.

Por el contrario, la elección de las arquitecturas U-Net y ResU-Net se fundamenta en su capacidad para resolver este compromiso:

- Recuperación Espacial: Las skip connections reintroducen los detalles de alta resolución bordes y texturas directamente desde la entrada, garantizando que el color se aplique con

cierta precisión sobre la estructura original.

- Capacidad de Aprendizaje: La inclusión de bloques residuales permite aumentar la profundidad de la red sin degradar el entrenamiento, facultando al modelo para aprender relaciones de color complejas y sutiles que un modelo superficial no podría captar.

## Entrenamiento

Ambos modelos se compilan utilizando el optimizador Adam y la función de pérdida MSE, estándar para regresión de píxeles. Sin embargo, para la ResU-Net se ha reducido el learning rate a 0.0005 para garantizar la estabilidad del gradiente dada su mayor profundidad.

Además, es interesante comentar la diferencia significativa en la carga computacional debido a la complejidad de las arquitecturas:

- Modelo U-Net: menos de 2 min/epoch. Al ser más ligero, permite un entrenamiento más extenso (40 epochs).
- Modelo ResU-Net: sobre 8 min/epoch. La inclusión de bloques residuales, normalización por lotes y mayor número de filtros (hasta 512) incrementa drásticamente las operaciones por ciclo, lo que ha obligado a limitar el entrenamiento.

Para acelerar el entrenamiento de ambos modelos se ha implementado una política de Mixed Precision (FP16).

## Resultados

Mostramos en la siguiente tabla las métricas obtenidas en test de ambos modelos:

Cuadro 1: Comparación de métricas entre U-Net y ResU-Net

Métrica	U-Net	ResU-Net
SSIM	0.9323	0.9360
PSNR	25.3312	25.0441
LPIPS	0.1345	0.1466

Los resultados cuantitativos muestran un rendimiento muy parejo entre ambos modelos, con ligeras variaciones atribuibles a la diferencia en el tiempo de entrenamiento (40 épocas vs 5 épocas):

- **SSIM (estructura):** La ResU-Net (0.9360) supera ligeramente a la U-Net. Esto confirma que los bloques residuales y la mayor capacidad de la red ayudan a reconstruir mejor la estructura y los bordes de la imagen, incluso con menos entrenamiento.
- **PSNR y LPIPS (fidelidad y percepción):** La U-Net obtiene mejores puntuaciones (mayor PSNR, menor LPIPS). Esto se debe a que, al haber entrenado durante 40 épocas, ha tenido tiempo de converger hacia una solución más estable y minimizar el error medio, mientras que la ResU-Net (con solo 5 épocas) quizás aún no haya alcanzado su potencial máximo.

Observamos a continuación ciertas imágenes:

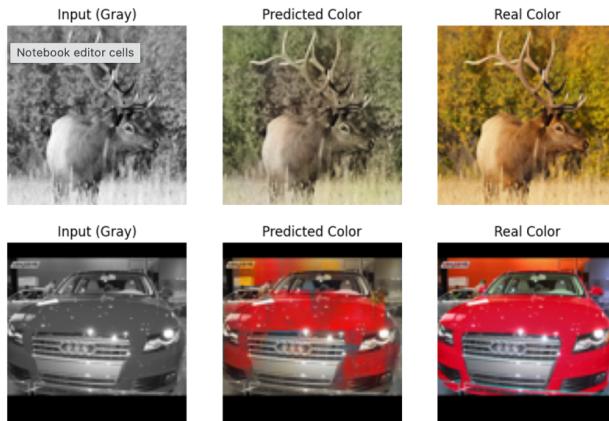


Figura 1: Imágenes generadas con U-Net

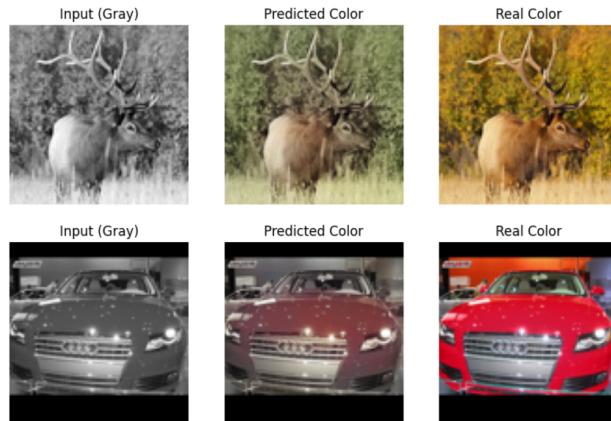


Figura 2: Imágenes generadas con ResU-Net

Vemos una mayor vivacidad y acierto con los colores de la U-Net que corresponde con los mejores resultados obtenidos en PSNR y LPIPS, pero vemos que su estructura es algo peor porque sus bordes coloreados son algo desacertados.

## Conclusiones

El análisis comparativo entre las dos arquitecturas propuestas revela un claro compromiso entre complejidad y rendimiento.

Si bien la ResU-Net demostró una superioridad estructural latente (mejor SSIM) gracias a sus bloques residuales, su elevado coste computacional limitó la convergencia cromática en comparación con la U-Net, que logró mayor fidelidad de color gracias a un entrenamiento más extenso.

A pesar de las limitaciones intrínsecas del modelo autoencoder y la pérdida MSE, que tienden a generar tonos desaturados ante la incertidumbre, los resultados cualitativos han superado las expectativas iniciales para modelos de esta complejidad,

## VAE

En este apartado, probaremos dos diferentes VAE sobre el mismo dataset y cuya entrada y salida es la misma que los autoencoders anteriores.

### VAE Denso

Este primer modelo implementa un VAE clásico con un cuello de botella denso [4]:

- Encoder: Rompe la estructura espacial de la imagen. Tras las convoluciones, utiliza Flatten para convertir el mapa de características 2D en un vector plano 1D, comprimiéndolo todo mediante capas Dense.
- Latent Space: Es un vector de números, sin altura ni anchura.
- Decoder: Debe reconstruir.<sup>el</sup> espacio desde cero. Utiliza una capa densa para expandir el vector y luego Reshape para forzarlo a tener forma de imagen antes de aplicar las deconvoluciones.

### VAE Convolucional

Este modelo implementa un Autoencoder Variacional que mantiene la topología 2D de la imagen a lo largo de todo el proceso, evitando las capas densas que destruyen la información espacial. Sus componentes técnicos clave son:

- Entrenamiento Estocástico: Implementa el Truco de Reparametrización

$$z = \mu + \sigma \cdot \epsilon$$

separando la aleatoriedad para permitir el cálculo de gradientes y el entrenamiento de la red.

- Encoder Aprendible: Sustituye el MaxPooling fijo por Convoluciones con Stride, otorgando a la red la capacidad de aprender cómo reducir la dimensión sin perder información estructural crítica. Utiliza activación LeakyReLU para garantizar la estabilidad del flujo de gradientes.
- Reconstrucción: El decodificador recupera la resolución original mediante Convoluciones Transpuestas, proyectando los detalles de color directamente desde los mapas de características espaciales del cuello de botella.

### Justificación de modelos

Estos modelos se implementan para abordar la ambigüedad intrínseca de la colorización porque un objeto gris puede tener múltiples colores válidos.

Mientras que los modelos deterministas tienden a promediar colores generando tonos apagados ante la duda, los VAEs aprenden una distribución de probabilidad, permitiendo generar colores más vivos y diversos.

Además, la creación del modelo VAE convolucional se justifica porque al mantener un espacio latente tensorial no aplanado, preserva la correlación espacial de los píxeles, resultando en una reconstrucción geométrica superior.

## Entrenamiento

Ambos modelos se basan en dos términos fundamentales en su función de pérdida: un término de reconstrucción y un término de regularización mediante divergencia KL. Sin embargo, en cada modelo lo implementamos de diferente forma

El primer modelo, utiliza MSE para medir la discrepancia entre la imagen reconstruida y la imagen real en color. Tras observar los resultados de este modelo, decidimos implementar MAE en el segundo modelo porque penaliza de forma lineal los errores, lo que tiende a producir reconstrucciones más nítidas.

Por otra parte, ambos modelos emplean la misma expresión para la divergencia KL. Sin embargo, en el primer modelo la KL se añade directamente a la pérdida total porque es un espacio latente relativamente pequeño.

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \mathcal{L}_{\text{KL}}$$

Esto ayuda a que la distribución latente sea estable y bien organizada.

En el segundo modelo la KL se multiplica por un peso  $\lambda$  muy pequeño.

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \lambda \cdot \mathcal{L}_{\text{KL}}$$

La razón es que el espacio latente ahora es mucho más grande. Si se utilizara la KL sin ponderación, esta dominaría por completo la pérdida, impidiendo que el modelo aprenda a reconstruir correctamente.

El tiempo de entrenamiento de ambos modelos es muy similar (2 min/epoch). Para estos modelos se ha desactivado Mixed Precision por sencillez.

## Resultados

Mostramos en la siguiente tabla las métricas obtenidas en test de ambos modelos:

Cuadro 2: Comparación de métricas entre VAE Denso y VAE Convolutional

Métrica	VAE Denso	VAE Convolucional
SSIM	0.0600	0.9338
PSNR	6.3406	25.1147
LPIPS	0.6884	0.1485

La comparación cuantitativa evidencia una diferencia abismal en el rendimiento de ambas arquitecturas, validando la importancia crítica de la preservación espacial en el espacio latente:

Los valores obtenidos (SSIM: 0.06, PSNR: 6.34) indican un colapso estructural casi total del VAE Denso. Al aplanar los mapas de características y utilizar capas densas, el modelo pierde la correlación espacial de los píxeles, resultando en una salida que no logra reconstruir ni siquiera la forma de los objetos, comportándose prácticamente como ruido aleatorio.

Por el contrario, el modelo totalmente convolucional alcanza un SSIM de 0.9338, situándose al mismo nivel de calidad que los modelos autoencoder. Esto demuestra que mantener la topología 2D en el cuello de botella es un requisito indispensable para la colorización.

La métrica perceptual confirma lo anterior (LPIPS): el VAE Denso (0.6884) genera imágenes irreconocibles para el ojo humano, mientras que el CVAE (0.1485) produce resultados coherentes y naturales.

Con una inspección visual observamos reflejados los resultados obtenidos cuantitativamente por las métricas:

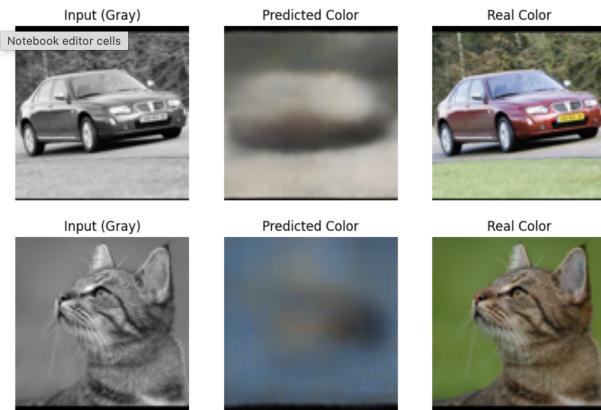


Figura 3: Imágenes generadas con VAE Denso

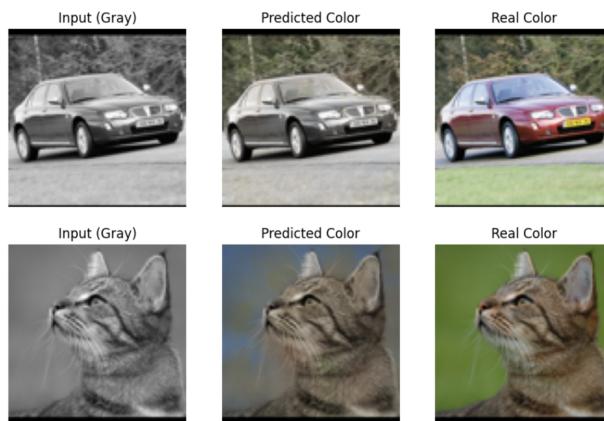


Figura 4: Imágenes generadas con VAE Convolucional

Observamos una imagen muy borrosa del primer VAE indicando que no ha sido capaz de recons-

truir la imagen. Con el segundo VAE sí tenemos una reconstrucción óptima de la imagen pero la colorización no es correcta.

## Conclusiones

La experimentación con modelos variacionales evidencia que la preservación de la topología en el espacio latente es un requisito indispensable, habiendo logrado el VAE convolucional rescatar la coherencia estructural tras el colapso del enfoque denso. Sin embargo, pese a esta validación arquitectónica, es necesario señalar que los resultados visuales finales no logran superar la calidad obtenida por los autoencoders. La restricción de regularización impuesta por la Divergencia KL, sumada a la optimización de pérdidas píxel a píxel, tiende a producir imágenes con texturas suavizadas y menor definición cromática. Esta limitación perceptiva justifica el siguiente paso en la investigación: la transición hacia las Redes GAN, cuyo mecanismo de discriminación busca romper precisamente esta barrera de borrosidad para forzar la generación de detalles de alta frecuencia y un realismo visual superior.

## GAN

Tras evaluar los modelos deterministas (Autoencoders y VAEs), exploramos ahora una arquitectura GAN para abordar la colorización desde este enfoque. A diferencia de los modelos anteriores, cuyo objetivo es minimizar un error de reconstrucción, las GAN introducen el discriminador que fuerza al generador a producir imágenes más realistas. Esto permite obtener resultados más vivos y definidos, especialmente en regiones donde la decisión del color es ambigua.

Nuestro modelo sigue una estructura similar al método *pix2pix*, donde el objetivo es aprender una transformación de imágenes en escala de grises (canal L del espacio Lab) a imágenes a color (canales a y b). La tarea se formula como un problema de *image-to-image translation*.

## Arquitectura del Generador (U-Net)

Para el generador empleamos una U-Net, ya que sus conexiones de salto permiten mantener la estructura espacial con gran fidelidad. En el contexto de la colorización, esto es especialmente importante: aunque el color puede ser ambiguo, la ubicación de los bordes y texturas debe conservarse con precisión.

El generador recibe como entrada un tensor  $(128 \times 128 \times 1)$  correspondiente al canal de luminancia L normalizado. Su estructura se divide en:

- **Encoder:** consiste en cinco bloques de *downsampling*, implementados con convoluciones de  $4 \times 4$  y stride 2. Cada bloque duplica el número de filtros (64, 128, 256, 512, 512), reduciendo la imagen hasta un mapa de  $4 \times 4$ . Se utiliza LeakyReLU como activación y BatchNorm en todos los niveles excepto el primero.
- **Decoder:** está formado por cuatro bloques de *upsampling* mediante *Conv2DTranspose*. Los dos primeros incluyen *dropout* para introducir ruido estructurado y evitar que el generador memorice patrones del conjunto de entrenamiento. Cada capa del decoder concatena su

salida con la activación correspondiente del encoder (skip connection), restaurando así la información espacial perdida.

- **Salida:** una convolución transpuesta final de  $4 \times 4$  produce una imagen  $(128 \times 128 \times 2)$  con activación  $tanh$ , que representa los canales a y b ya normalizados.

Esta arquitectura permite que el modelo coloree respetando forma, contornos y textura, incluso en regiones donde un autoencoder simple tiende a producir manchas o difuminados.

## Arquitectura del Discriminador (PatchGAN)

El discriminador evalúa si una imagen generada es realista comparándola con la imagen real. En lugar de clasificar la imagen completa, utilizamos un discriminador PatchGAN, que divide la imagen en parches locales y decide para cada uno si el contenido luminancia/color es coherente.

Su estructura consiste en:

- Concatenación de la entrada: el discriminador recibe una pareja formada por la imagen en escala de grises (L) y la imagen color (real o generada), que se combinan en un tensor  $(128 \times 128 \times 3)$ .
- Tres bloques de *downsampling* con convoluciones de  $4 \times 4$ , stride 2 y LeakyReLU, que reducen progresivamente la resolución y capturan texturas locales.
- Un nivel adicional de convolución sin *downsampling*, que amplía el campo receptivo sin perder detalle.
- Una última convolución produce un mapa de activación  $(14 \times 14 \times 1)$  donde cada valor indica si el parche correspondiente parece real o generado.

PatchGAN es especialmente útil porque penaliza los errores de coloración finos (bordes, texturas, transiciones) sin imponer restricciones globales demasiado rígidas.

## Función de pérdida

La pérdida total combina dos términos:

- **Pérdida adversaria** ( $\mathcal{L}_{GAN}$ ): obliga al generador a producir imágenes indistinguibles para el discriminador. Está implementada mediante binary crossentropy.
- **Pérdida L1** ( $\mathcal{L}_{L1}$ ): calcula el MAE entre los canales de color reales y generados. Esto estabiliza el entrenamiento y evita colores fuera de rango o inconsistentes.

La pérdida global se define como:

$$\mathcal{L}_G = \mathcal{L}_{GAN} + \lambda \cdot \mathcal{L}_{L1},$$

donde utilizamos  $\lambda = 100$ .

## Entrenamiento

Tanto el generador como el discriminador se entrena con el optimizador Adam ( $\text{lr} = 2\text{e}-4$ ,  $\beta_1 = 0,5$ ), como se suele recomendar para que el entrenamiento de la GAN sea estable.

Durante cada iteración:

1. El generador predice los canales  $a$  y  $b$  a partir del canal  $L$ .
2. El discriminador recibe tanto la imagen real ( $L + ab$ ) como la generada.
3. Se calcula la pérdida, la pérdida L1 y los gradientes de ambos modelos.
4. Se actualizan las dos redes de forma independiente.

Entrenamos el modelo durante 40 épocas utilizando un subconjunto del dataset STL-10 (20.000 imágenes no etiquetadas para entrenamiento, 1.000 para validación y 3.000 para test). Todas las imágenes se transforman al espacio Lab mediante *skimage* y se normalizan a valores entre  $[-1, 1]$ . El tamaño final utilizado es  $128 \times 128$ , lo que equilibra detalle y tiempo de cómputo.

## Resultados

En las primeras épocas las colorizaciones muestran tonos desaturados y cierta inconsistencia en regiones complejas. A medida que avanza el entrenamiento, el modelo aprende a producir colores más vivos y transiciones suaves.

Frente a los autoencoders, la GAN genera imágenes perceptualmente más atractivas, con mayor viveza y contraste. Sin embargo, también muestra cierta variabilidad en zonas ambiguas, consecuencia directa de la optimización adversaria y del carácter condicional del modelo.

## Colorización con modelos de difusión

### Colorización Guiada por Texto

El objetivo es desarrollar un modelo capaz de colorear imágenes utilizando como guía una descripción textual. La idea central es que la red no solo aprenda a reconstruir colores, sino que además incorpore información semántica explícita presente en el texto (por ejemplo: “pétalos rojos”, “centro amarillo”, etc.).

Inicialmente se intentó utilizar el dataset original con animales y vehículos del resto de apartados del proyecto. Sin embargo, dicho conjunto de datos no incluía descripciones textuales ni etiquetas, lo que dificultaba enormemente la incorporación de información lingüística. Se intentaron dos estrategias para suplir esta ausencia:

1. **Generar descripciones automáticamente mediante un captioner preentrenado.**  
Resultó poco útil porque al trabajar con imágenes en escala de grises, el captioner no dis-

ponía de pistas sobre los colores reales y la información que producía era demasiado general, no aportaba señales relevantes al proceso de colorización que no entendiera ya el propio AE colorizador ya que no mencionaba nada sobre colores. Además, era más costoso de ejecutar por que requería modelo pesado durante la preparación del dataset para generar los textos, ralentizando el flujo de trabajo.

### 2. Clasificar cada imagen según su color dominante.

Este método tampoco funcionó: muchas imágenes contenían varios colores relevantes, o el color dominante procedía del fondo en vez del objeto de interés. Esto generaba etiquetas incorrectas además de prompts muy simples (nombres de los colores mayoritarios exclusivamente) que luego en la inferencia no serían iguales y hacía que el modelo no mejorase respecto al autoencoder base.

Tras comprobar que ninguno de estos dos enfoques producía beneficios reales, decidimos recurrir a un dataset ya anotado con descripciones ricas y con referencias explícitas a colores.

## Dataset empleado

Seleccionamos el dataset `Oxford Flowers102`, que incluye para cada imagen de la flor correspondiente, una descripción textual detallada. Estas descripciones suelen mencionar explícitamente los colores de los pétalos, hojas o centros, lo cual resulta ideal para un modelo de colorización guiada.

Para asegurar que el texto proporcionado realmente aportaba información cromática, construimos una lista de palabras relacionadas con colores (alrededor de 40 términos, incluyendo principalmente nombres de colores y tonalidades) y solo se conservaron aquellas imágenes cuyo texto contenía al menos una palabra de la lista. Con esto garantizamos que el modelo recibe mensajes relevantes sobre el color que queremos obtener.

## Arquitectura del modelo

El modelo desarrollado se basa en un **UNet** con condicionamiento textual mediante capas FiLM. Se trata de un diseño autoencoder con skip-connections, que permite preservar detalles espaciales mientras el texto influye en la reconstrucción del color.

### Embedding textual

Para transformar las descripciones en vectores utilizables por la red, empleamos un **modelo CLIP preentrenado** (`openai/clip-vit-base-patch32`). Únicamente se utilizó su codificador textual para obtener un embedding robusto, compacto y semánticamente rico del texto que luego pasar al AE colorizador. Además, al ser un modelo transformer grande, el embedding representa no solo palabras de color, sino también la estructura semántica del texto, proporcionando al modelo una guía contextual (por ejemplo: si el color se refiere a los pétalos, al centro, a las hojas, etc.).

### Estructura general

- **Encoder:** cuatro bloques convolucionales que reducen progresivamente la resolución y extraen características visuales de nivel creciente.

- **Bottleneck:** un bloque central donde se combina la información visual comprimida con la información textual y se enriquece.
- **Decoder:** cuatro bloques de deconvolución que reconstruyen la imagen a color, fusionando la información procesada con los mapas del encoder mediante conexiones residuales. El último bloque del decoder produce la imagen RGB resultante (salida en 3 canales).

### Bloques convolucionales

Cada bloque convolucional del modelo está compuesto por dos operaciones principales. Una capa convolucional, que permiten extraer y transformar características visuales de forma progresiva. Y una capa de normalización por batch, aplicada tras las convoluciones para estabilizar la distribución de activaciones y facilitar el entrenamiento.

A partir de esta estructura común, el modelo utiliza dos variantes según la etapa. En el encoder, los bloques emplean convoluciones estándar que reducen la resolución espacial de la imagen, capturando patrones cada vez más globales. En el decoder, los bloques recurren a convoluciones transpuestas, que aumentan la resolución y permiten reconstruir la imagen final a partir de las representaciones comprimidas.

### Condicionamiento textual

El componente clave del modelo son las capas **FiLM** (Feature-wise Linear Modulation). Estas capas permiten que el texto modifique directamente la activación de las capas convolucionales mediante dos parámetros aprendidos:  $\gamma$  que controla la amplificación o atenuación por canal, y  $\beta$  que desplaza la activación por canal.

La operación es:

$$\text{FiLM}(X) = X \cdot (1 + \gamma) + \beta$$

donde cada canal de la activación se ajusta en función del embedding textual. logrando que si el texto menciona “petals are bright red”, las FiLM potencian las características asociadas a tonos rojizos en regiones que visualmente parezcan pétalos; si describe “dark green leaves”, favorece la aparición de verdes más apagados en la zona de hojas.

El condicionamiento se aplica por primera vez en el bottleneck, donde se controla la representación más comprimida de la imagen, y luego se continua aplicando en las etapas del decoder, donde se ajusta el color reconstruido de forma más localizada según los FiLM, que actúan integrando la semántica lingüística en el flujo visual del UNet.

### Entrenamiento

Durante el proceso de entrenamiento se utilizó una función personalizada `collate_fn` para garantizar que cada imagen queda alineada con su descripción textual antes de introducirse en el modelo.

Para la etapa de aprendizaje se empleó **MSE (Mean Squared Error)** como función de pérdida, comparando píxel a píxel la imagen generada con la imagen real. Para tarea de colorización resulta

ser de las más útiles ya que consiste en aproximar colores continuos. El modelo se entrenó sobre la partición de entrenamiento y se validó periódicamente para monitorizar su rendimiento.

## Resultados

Para evaluar el desempeño del modelo se definió un procedimiento de testeo que carga el modelo entrenado y la sección de los datos reservada para pruebas. Los resultados se presentan en forma de tríos de imágenes: (imagen en gris, predicción generada, referencia real) acompañados del prompt textual.

Los resultados muestran que el modelo es capaz de producir una colorización muy buena por regiones salvo en algunos casos concretos donde confunde los pétalos con hojas. Suponemos que debido a que las descripciones textuales contienen detalles sobre colores y partes de la flor, se logra que el modelo aplique los colores en las zonas adecuadas gracias a que la información del texto complementa la estructura visual.

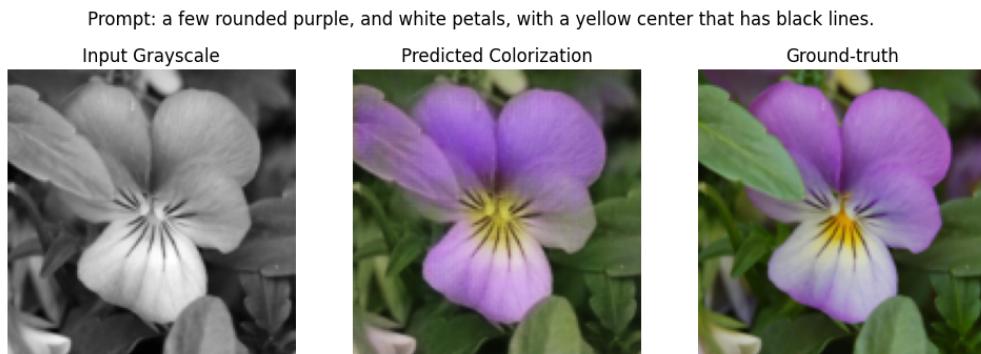


Figura 5: Ejemplo de resultado de colorización guiada por texto (1).



Figura 6: Ejemplo de resultado de colorización guiada por texto (2).



Figura 7: Ejemplo de resultado de colorización guiada por texto (3).

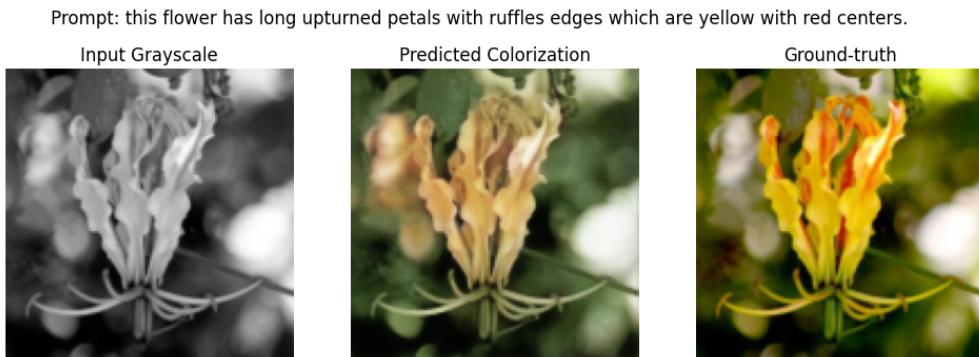


Figura 8: Ejemplo de resultado de colorización guiada por texto (4).

## Conclusiones

La colorización guiada por texto ha demostrado ofrecer resultados muy sólidos cuando se dispone de imágenes acompañadas de descripciones ricas y precisas, como ocurre en el caso de las flores. Consideramos que esta estrategia podría extenderse a otros dominios donde la colorización sea relevante; sin embargo, la principal limitación reside en la disponibilidad de *captions* de calidad, algo poco común en la mayoría de los datasets existentes. Pensamos que sería factible aproximarse a un colorizador más general para todos los usos si se adoptara un enfoque de entrenamiento similar al de modelos como CLIP, que aprovechan casi todas las imágenes que hay en Internet junto a su texto alternativo. No obstante, un esfuerzo de esta magnitud requeriría modelos de mayor capacidad y recursos computacionales sustancialmente superiores.

## Conclusión

## Help

## Tables

Tables must be centered on the page. A minimum font size of 8 points is recommended for all text within tables. An example of a table is shown below.

Note Name	Frequency (Hz)
F4	349.23
G4	392
A4	440
B4	493.88
C5	523.25

Cuadro 3: Note names and their corresponding frequencies.

## Captions

Captions should follow each figure or table and conform to the guidelines below:

- Please number figures and tables in order and make sure that image files can be clearly identified from their filenames.
- Images not created by the author must have a catalogue-style caption supplied as a separate list. Captions should include:
  - Maker
  - Title (in italics)
  - Date
  - Medium
  - Measurements
  - Location (Institution, Geographic Location)
  - Copyright statement
  - Courtesy line and/or photo credit

An example of a caption is shown below:

**Fig 1.** Yinka Shonibare, The Swing (after Fragonard), 2001, mannequin, cotton costume, 2 slippers, swing seat, 2 ropes, oak twig and artificial foliage, 3.3 x 3.5 x 2.2 m (Tate, London) © Yinka Shonibare; image courtesy Stephen Friedman Gallery, London.

## Equations

Equations should be placed on separate lines and numbered. An example is shown below.

$$f(x) = a_0 + \sum_{n=1}^{\infty} \left( a_n \cos \frac{n\pi x}{L} + b_n \sin \frac{n\pi x}{L} \right) \quad (1)$$

## Citas

Para citar le pedis a chat que os de el bibtex del paper, o sino en arxiv está y lo meteis en references.bib.

Luego citáis así: [5]

## Agradecimientos

...

## Referencias

- [1] Adam Coates, Andrew Ng y Honglak Lee. “An Analysis of Single-Layer Networks in Unsupervised Feature Learning”. En: *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2011, págs. 215-223.
- [2] Olaf Ronneberger, Philipp Fischer y Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. En: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer. 2015, págs. 234-241.
- [3] Zizhao Zhang, Qi Liu y Yefeng Wang. “Road Extraction by Deep Residual U-Net”. En: *IEEE Geoscience and Remote Sensing Letters* 15.5 (2018), págs. 749-753.
- [4] Diederik P. Kingma y Max Welling. “Auto-Encoding Variational Bayes”. En: *arXiv preprint arXiv:1312.6114* (2013).
- [5] Nir Zabari et al. “Diffusing Colors: Image Colorization with Text Guided Diffusion”. En: *arXiv preprint arXiv:2312.04145* (2023). DOI: 10.48550/arXiv.2312.04145. URL: <https://arxiv.org/abs/2312.04145>.