Generación de imágenes con redes neuronales generativas adversarias profundas

Rodrigo Dávalos Alarcón

9 de diciembre de 2024

Resumen

Este es el resumen de la investigación. Breve descripción de los objetivos, metodología y resultados.

1. Introducción

La clasificación de imágenes es una parte importante de la inteligencia artificial. Y si se desarrolla lo suficiente, la clasificación de imágenes puede llegar a reconcer objetos en tiempo real, reconcer patrones, ser inclusive mejor que los humanos en ciertas tareas. Sus aplicaciones pueden ir desde la medicina hasta la conducción de vehiculos autónomos. En esta investigación veremos la clasificación de pinturas y se estudiará como es que un conjunto de redes convolucionales puede lograr reconcer patrones que a los humanos nos puede ser dificiles de explicar (y por ende de programar).

1.1. Contexto

En la actualidad las inteligencias artificiales han avanzado demasiado. Hoy lo que mas impacto genera sin duda son las inteligencias artificilaes generativas. Las inteligencias artificiales para clasificación y predicción existen desde hace mucho tiempo. Uno de los primeros algoritmos para clasificación y predicción es el algoritmo de Native Bayes [1]. Este algoritmo puede funcionar para clasificación binaria y de multiples clases. Este algoritmo data del año 1760. Uno de los primeros programas de clasificación que usan redes convolucionales data del año 1980 llamado LeNet usado para el reconocimiento de dígitos [2]. De hecho existen muchas arquitecturas para las CNN's[3]. Entre ellas se encuentran:

- Alex Net
- Le Net
- VGG
- Google Net
- Res Net

Es elección de cada persona escojer el modelo que más le convenga o adaptar uno ya creado (crear un modelo suele ser más difícil).

El modelo que usaremos en esta investigación se basa en el modelo usado por el usuario **mkkoehler** de la plataforma `www.kaggle.com` [4]. Este modelo es un modelo simple de redes convolucionales que ayudan a ejemplificar como funcionan.

1.2. Motivación

La principal motivación de esta investigación radica en la curiosidad por explorar la relación entre las inteligencias artificiales y las imágenes. La generación de imágenes es, sin duda, uno de los temas más fascinantes en este campo de investigación. Sin embargo, antes de adentrarse plenamente en el proceso de generación, resulta fundamental comprender a fondo el funcionamiento de las redes neuronales convolucionales. Este es, precisamente, otro de los objetivos de este proyecto: conocer cómo operan las redes convolucionales y cómo una máquina es capaz de identificar patrones. A partir de este entendimiento, se busca, en un futuro cercano, comprender cómo estas redes pueden generar nuevas imágenes basándose en los patrones previamente aprendidos.

1.3. Objetivo general

Comprender el funcionamiento de una red neuronal convolucional y su aplicación en la clasificación de imágenes.

1.4. Objetivos específicos

1. Analizar el funcionamiento de una red neuronal convolucional.

2. Implementar el código de una red neuronal convolucional para la clasificación de imágenes.

3. Interpretar los resultados obtenidos con el fin de extraer conclusiones significativas.

4. Explicar el proceso de clasificación de imágenes utilizando una red neuronal convolucional.

5. Proponer mejoras al código previamente implementado.

6. Modificar el código existente para optimizar su rendimiento.

7. Incorporar los métodos sugeridos por el docente, como el Análisis de Componentes Principales (PCA) y el aprendizaje no supervisado, y justificar su aplicación o la falta de aplicabilidad.

1.5. Estructura del documento

Este documento contendrá no solo texto, si no también ejecución de código para que se tenga un documento sólido y que elimine la necesidad de ir de los archivos `.pdf` al `.ipynb`. Sin embargo se proporcionará los links respectivos al repositorio cuando sea necesario.

2. Marco teórico

Se presentan los conceptos fundamentales que que se usarán en el trabajo de investigación, en particular aquellos conceptos relacionados con inteligencia artificiales, redes neuronales convolucionales y los requisitos para la implementación y modificación del código.

2.1. Antecedentes

En la investigación A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects [5] podemos ver lo que es la evolución de las redes neuronales convolucionales. Estas redes se desarrollaron teóricamente hacer mucho tiempo. Las personas en ese tiempo no creían que fuera posible que una computadora diferencie un gato de un perro. Sin embargo con la evolución de la tecnología estas ideas se volvieron a retomar. El término convolucional viene por primera vez en la red LeNet en 1989. En un principio, luego de lo que fue la primera red neuronal, se empezaron a proponer arquitecturas más complejas que tenían múltiples capas ocultas que por esa época hacian difícil el cálculo para el ajuste de los pesos. Luego, por la década de los 2010 se empezaron a desarrollar nuevas arquitecturas multipropósito, para reconcer objetos, para la segmentación, y detección de imágenes.

No es coincidencia que las redes neuronales convolucionales se usen para las imagenes. Desde el principio se pude hallar una conexión entre ellas y nuestra visión. Cada neurona artificial corresponde a una neurona biológica y cada kérnel que se usa en la convolución es en realidad un receptor diferente que capta una cosa diferente como los colores, los bordes, etc. Las redes convolucionales tiene muchas ventajas, por ejemplo, se necesitan menos neuronas, y estas neuronas pueden compartir el mismo peso. Esto facilita el proceso de entrenamiento ya que no se necesita ajustar muchos parámetros.

Hoy en día usar estas redes es mucho más fácil que años atrás. Se pueden ejecutar en nuestros ordenadores, y si no es así, se pueden ejecutar en la nube. Las redes convolucionales han pasado ya ha trabajar en videos, lo cúal supone un gasto computacional mayor que cuando se trabaja en imágenes, sin embargo, esperamos que en el futuro esta tecnología sea accesible a todos como lo es hoy la tecnología que estudiamos.

2.2. Conceptos clave

2.2.1. Red neuronal

Una red neuronal es un modelo matemático inspirado en la estructura biológica del cerebro humano, compuesto por neuronas artificiales organizadas en capas. Cada neurona realiza una combinación ponderada de sus entradas, les aplica una función de activación, y pasa el resultado a la siguiente capa. Su objetivo principal es aprender una función $f : \mathbb{R}^n \to \mathbb{R}^m$, donde n es el número de características de entrada y m el número de clases o valores de salida.

La salida de una neurona en la capa l se define como:

$$z_i^{(l)} = \sum_{j=1}^{n^{(l-1)}} w_{ij}^{(l)} a_j^{(l-1)} + b_i^{(l)},$$

donde:

- $w_{ij}^{(l)}$ son los pesos
- $b_i^{(l)}$ es el sesgo

- $a_j^{(l-1)}$ es la salida de la neurona j en la capa anterior

2.2.2. Convolución

La convolución es una operación matemática fundamental en las redes neuronales convolucionales (CNN). Consiste en aplicar un filtro (kernel en inglés) a una entrada, como una imagen, para extraer características específicas. Matemáticamente, se define como:

$$s(t) = (x * w)(t) = \int_{-\infty}^{\infty} x(\tau)w(t-\tau)\,d\tau,$$

donde:

- x es la entrada
- w es el filtro
- t es la posición actual

En el contexto discreto (como las imágenes), se calcula como:

$$s(i,j) = \sum_m \sum_n x(i+m, j+n) \cdot w(m,n).$$

2.2.3. Función de activación

Las funciones de activación introducen no linealidades en la red neuronal, permitiendo que la red aprenda relaciones complejas. Ejemplos comunes incluyen:

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

Estas funciones transforman la salida lineal de una neurona en un valor dentro de un rango específico, como $[0, 1]$ o $[-1, 1]$.

2.2.4. Tasa de aprendizaje

La tasa de aprendizaje (η) es un hiperparámetro que controla la magnitud de los ajustes realizados en los pesos de la red durante el entrenamiento. Si es demasiado alta, el modelo puede no converger; si es demasiado baja, el aprendizaje será muy lento. En el descenso de gradiente, la actualización de los pesos se calcula como:

$$w = w - \eta \nabla L(w),$$

donde: - $\nabla L(w)$ es el gradiente del error con respecto a los pesos w.

2.2.5. Descenso de gradiente

El descenso de gradiente es un algoritmo de optimización utilizado para minimizar la función de pérdida de un modelo. Se basa en calcular el gradiente de la función de pérdida y ajustar los pesos en la dirección opuesta al gradiente:

$$w^{(t+1)} = w^{(t)} - \eta \nabla L(w^{(t)}),$$

donde η es la tasa de aprendizaje.

Existen variantes como:

- **Stochastic Gradient Descent (SGD)**: Utiliza un subconjunto aleatorio de datos para calcular el gradiente
- **Mini-Batch Gradient Descent**: Combina el SGD con pequeños lotes de datos

2.2.6. TensorFlow

TensorFlow es una biblioteca de código abierto para el desarrollo y entrenamiento de modelos de aprendizaje automático. Permite implementar operaciones matemáticas en estructuras llamadas tensores. Su diseño basado en grafos computacionales facilita la paralelización y optimización.

2.2.7. Notación tensorial

Los tensores son generalizaciones de matrices a dimensiones superiores. En TensorFlow, las operaciones tensoriales se expresan matemáticamente como:

- Vector (1-dimensional): $\mathbf{v} \in \mathbb{R}^n$
- Matriz (2-dimensional): $\mathbf{M} \in \mathbb{R}^{m \times n}$
- Tensor (N-dimensional): $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times \ldots \times d_N}$

2.2.8. Entrenamiento

El entrenamiento de un modelo implica ajustar los pesos para minimizar una función de pérdida utilizando un algoritmo de optimización como el descenso de gradiente. Cada iteración incluye:

1. **Propagación hacia adelante**: Calcular la salida del modelo
2. **Cálculo de la pérdida**: Comparar la predicción con el objetivo
3. **Retropropagación**: Ajustar los pesos usando el gradiente

2.2.9. Validación

La validación es el proceso de evaluar el rendimiento de un modelo en un conjunto de datos separado del entrenamiento. Se utiliza para prevenir el sobreajuste y seleccionar los mejores hiperparámetros. La métrica de validación más común es la precisión, definida como:

$$\text{Precisión} = \frac{\text{Número de predicciones correctas}}{\text{Número total de predicciones}}.$$

2.3. Palabras clave

- Convolución
- Clásificación de imágenes
- TensorFlow
- Inteligencia Artificial

3. Metodología

La metodología de la presente investigación se describe a continuación, detallando el diseño, los instrumentos, y las técnicas utilizadas para llevar a cabo el estudio.

3.1. Diseño de la Investigación

Esta investigación sigue un enfoque cuantitativo y experimental, complementado con elementos de aprendizaje por reproducción. Su propósito es analizar el funcionamiento de un modelo de red neuronal generativa adversaria profunda (GAN), replicar su implementación y estudiarlo para luego poder hacer algunas modificaciones.

3.2. Datos usados para el entrenamiento

Los datos usados para el entrenamiento provienen de un dataset de `www.kaggle.com`, más concrétamente de un dataset llamado **Best Artworks of All Time** [6]. Este dataset pesa 2.32 Gb y contiene imágenes de las pinturas de 49 artistas:

1. Albrecht Durer (328 imágenes)
2. Alfred Sisley (259 imágenes)
3. Amedeo Modigliani (193 imágenes)
4. Andrei Rublev (99 imágenes)
5. Andy Warhol (181 imágenes)
6. Camille Pissarro (91 imágenes)
7. Caravaggio (55 imágenes)
8. Claude Monet (73 imágenes)
9. Diego Rivera (70 imágenes)
10. Diego Velazquez (128 imágenes)
11. Edgar Degas (702 imágenes)
12. Edouard Manet (90 imágenes)
13. Edvard Munch (67 imágenes)
14. El Greco (87 imágenes)
15. Eugene Delacroix (31 imágenes)
16. Francisco Goya (291 imágenes)
17. Frida Kahlo (120 imágenes)
18. Georges Seurat (43 imágenes)
19. Giotto di Bondone (119 imágenes)
20. Gustav Klimt (117 imágenes)
21. Gustave Courbet (59 imágenes)
22. Henri Matisse (186 imágenes)
23. Henri Rousseau (70 imágenes)
24. Henri de Toulouse-Lautrec (81 imágenes)
25. Hieronymus Bosch (137 imágenes)
26. Jackson Pollock (24 imágenes)
27. Jan van Eyck (81 imágenes)
28. Joan Miro (102 imágenes)
29. Kazimir Malevich (126 imágenes)
30. Leonardo da Vinci (143 imágenes)
31. Marc Chagall (239 imágenes)
32. Michelangelo (49 imágenes)
33. Mikhail Vrubel (171 imágenes)
34. Pablo Picasso (439 imágenes)
35. Paul Cezanne (47 imágenes)
36. Paul Gauguin (311 imágenes)

37. Paul Klee (188 imágenes)
38. Peter Paul Rubens (141 imágenes)
39. Pierre-Auguste Renoir (336 imágenes)
40. Piet Mondrian (84 imágenes)
41. Pieter Bruegel (134 imágenes)
42. Raphael (109 imágenes)
43. Rembrandt (262 imágenes)
44. Rene Magritte (194 imágenes)
45. Salvador Dali (139 imágenes)
46. Sandro Botticelli (164 imágenes)
47. Titian (255 imágenes)
48. Vasiliy Kandinskiy (88 imágenes)
49. Vincent van Gogh (877 imágenes)
50. William Turner (66 imágenes)
51. Marc Chagall (239 imágenes)
52. Michelangelo (49 imágenes)
53. Mikhail Vrubel (171 imágenes)
54. Pablo Picasso (439 imágenes)
55. Paul Cezanne (47 imágenes)
56. Paul Gauguin (311 imágenes)
57. Paul Klee (188 imágenes)
58. Peter Paul Rubens (141 imágenes)
59. Pierre-Auguste Renoir (336 imágenes)
60. Piet Mondrian (84 imágenes)
61. Pieter Bruegel (134 imágenes)
62. Raphael (109 imágenes)
63. Rembrandt (262 imágenes)
64. Rene Magritte (194 imágenes)
65. Salvador Dali (139 imágenes)
66. Sandro Botticelli (164 imágenes)
67. Titian (255 imágenes)
68. Vasiliy Kandinskiy (88 imágenes)
69. Vincent van Gogh (877 imágenes)
70. William Turner (66 imágenes)

Las distintas imágenes no tienen el mismo tamaño, por lo tanto hay que preprocesarlas antes de usarlas en el entrenamiento.

Figura 1: Avenue de l'Opera Rain Effect de Camille Pissarro. 1898.

Una muestra del dataset la encontramos en la Figura 1. Las distintas imágenes del dataset pertenecen a diferentes artistas de diferentes épocas. El más antiguo es Giotto di Bondone del año 1267–1337 y el más reciente es Andy Warhol del año 1928–1987.

3.3. Técnicas de Análisis de Datos

Los datos serán analizados con diferentes herramientas, pero uso

3.4. Instrumentos utilizados

Los instrumentos utilizados en esta investigación incluyen:

- **VSCode**: Como entorno de desarrollo para escribir y ejecutar el código.
- **Python**: Lenguaje de programación utilizado para la implementación del modelo GAN.
- **TensorFlow**: Biblioteca para la creación y entrenamiento de redes neuronales.
- **Numpy**: Biblioteca que usa los recursos de C para facilitar el cálculo con matrices.

- **Matplotlib**: Biblioteca para la graficación y mostrado de imágenes.
- **Pandas**: Biblioteca para el manejo eficiente de datos tabulares.
- **Pillow**: Biblioteca para la manipulación y mostrado de imágenes.
- **Keras**: API de redes neuronales escrita en Python, una interfaz de alto nivel.
- **Jupyter Nootebooks**: Aplicación que permite compilar y ejecutar código interactivo.
- **Anaconda**: Gestor de paquetes de Python orientado a la ciencia de datos.

La versión de `TensorFlow` es la 2.12.3 Por lo tanto, no se está usando la última versión disponible, si no la que es compatible con los demás paquetes según el administrador de paquetes de `Anaconda`.

3.5. Procedimiento

El procedimiento consistirá en tomar un modelo básico de GAN disponible en línea y adaptarlo a nuestras necesidades específicas. Primero, se realizará la preparación de los datos de entrenamiento utilizando un dataset de pinturas impresionistas. Luego, se entrenará el modelo, ajustando los parámetros según sea necesario para mejorar la calidad de las imágenes generadas. Finalmente, se evaluará la calidad de las imágenes generadas mediante métricas de similitud visual y análisis subjetivo.

3.6. Ética de la Investigación

Para esta investigación, se utilizarán pinturas de dominio público, específicamente del movimiento impresionista, lo que garantiza que no se infringirán derechos de autor. Además, se tomará en cuenta el uso ético de los datos, respetando las normativas vigentes sobre el uso de obras artísticas.

3.7. Limitaciones

Las principales limitaciones de esta investigación incluyen la falta de recursos computacionales avanzados, lo que puede limitar la capacidad de entrenamiento de redes neuronales complejas. Además, el conocimiento sobre redes neuronales es limitado, por lo que se intentará adaptar un código preexistente para ajustarlo a las necesidades del proyecto.

4. Implementación

4.1. Preprocesamiento

Las siguiente páginas pertenecen a la exportación de un archivo `Jupyter Notebooks` a `.pdf`.

1 Preprocesamiento

Para este apartado lo que haremos será hacer 3 tipos de preprocesamiento:

- Escalar las imágenes a 300x300
- Aumentar datos (data augmentation)
- Normalizar de 0 a 1
- Matriz OneHotEncoder

Veremos luego que este preprocesamiento manual es innecesario ya que muchas herramientas de tensorflow pueden hacerlo.

1.1 Escalar las imágenes a 300x300

Para escalar todas la imágenes lo que haremos será tomas cadas una de las imágenes de la carpeta `images` y copiarlas en otra carpeta con el nombre `resized_images`. Por motivos de ahorro de CPU este y los demás métodos solo se aplicarán a una imagen de cada carpeta.

Estas son las librerías que usaremos:

[19]:
```
from PIL import Image, ImageEnhance
import numpy as np
import pandas as pd
import os
import random
```

[2]:
```
def resize(input_path, output_path):
    dirs = os.listdir(input_path)
    for dirpath, dirnames, filenames in os.walk(input_path):
        if len(filenames) > 0:
            filename = filenames[0]
            if filename.lower().endswith(('.png', '.jpg', '.jpeg')):
                file_path = os.path.join(dirpath, filename)

                relative_dir = os.path.relpath(dirpath, input_path)
                new_dir = os.path.join(output_path, relative_dir)
                os.makedirs(new_dir, exist_ok=True)

                im = Image.open(file_path)
                imResize = im.resize((300, 300))

                save_path = os.path.join(new_dir, f'{os.path.
 ↪splitext(filename)[0]} resized.jpg')
                imResize.save(save_path, 'JPEG', quality=90)
                print(f'Imagen guardada en: {save_path}')
```

[3]:
```
resize('images', 'resized_images')
```

```
Imagen guardada en: resized_images/Eugene_Delacroix/Eugene_Delacroix_16
resized.jpg
```

```
Imagen guardada en: resized_images/Claude_Monet/Claude_Monet_54 resized.jpg
Imagen guardada en: resized_images/Edvard_Munch/Edvard_Munch_41 resized.jpg
Imagen guardada en: resized_images/Sandro_Botticelli/Sandro_Botticelli_159
resized.jpg
Imagen guardada en: resized_images/Paul_Klee/Paul_Klee_139 resized.jpg
Imagen guardada en: resized_images/Albrecht_Du êrer/Albrecht_Du êrer_127
resized.jpg
Imagen guardada en: resized_images/Georges_Seurat/Georges_Seurat_34 resized.jpg
Imagen guardada en: resized_images/Jackson_Pollock/Jackson_Pollock_18
resized.jpg
Imagen guardada en: resized_images/Edgar_Degas/Edgar_Degas_305 resized.jpg
Imagen guardada en: resized_images/Rene_Magritte/Rene_Magritte_129 resized.jpg
Imagen guardada en: resized_images/Andrei_Rublev/Andrei_Rublev_28 resized.jpg
Imagen guardada en: resized_images/Pierre-Auguste_Renoir/Pierre-
Auguste_Renoir_261 resized.jpg
Imagen guardada en: resized_images/Hieronymus_Bosch/Hieronymus_Bosch_26
resized.jpg
Imagen guardada en: resized_images/Henri_de_Toulouse-Lautrec/Henri_de_Toulouse-
Lautrec_67 resized.jpg
Imagen guardada en: resized_images/Pablo_Picasso/Pablo_Picasso_375 resized.jpg
Imagen guardada en: resized_images/Andy_Warhol/Andy_Warhol_139 resized.jpg
Imagen guardada en: resized_images/Kazimir_Malevich/Kazimir_Malevich_22
resized.jpg
Imagen guardada en: resized_images/Gustav_Klimt/Gustav_Klimt_29 resized.jpg
Imagen guardada en: resized_images/Joan_Miro/Joan_Miro_81 resized.jpg
Imagen guardada en: resized_images/Albrecht_Dürer/Albrecht_Dürer_43
resized.jpg
Imagen guardada en: resized_images/Camille_Pissarro/Camille_Pissarro_1
resized.jpg
Imagen guardada en: resized_images/Diego_Rivera/Diego_Rivera_34 resized.jpg
Imagen guardada en: resized_images/Marc_Chagall/Marc_Chagall_194 resized.jpg
Imagen guardada en: resized_images/Frida_Kahlo/Frida_Kahlo_41 resized.jpg
Imagen guardada en: resized_images/Henri_Rousseau/Henri_Rousseau_58 resized.jpg
Imagen guardada en: resized_images/Amedeo_Modigliani/Amedeo_Modigliani_27
resized.jpg
Imagen guardada en: resized_images/Peter_Paul_Rubens/Peter_Paul_Rubens_74
resized.jpg
Imagen guardada en: resized_images/El_Greco/El_Greco_14 resized.jpg
Imagen guardada en: resized_images/Paul_Gauguin/Paul_Gauguin_250 resized.jpg
Imagen guardada en: resized_images/Salvador_Dali/Salvador_Dali_109 resized.jpg
Imagen guardada en: resized_images/Henri_Matisse/Henri_Matisse_158 resized.jpg
Imagen guardada en: resized_images/Paul_Cezanne/Paul_Cezanne_38 resized.jpg
Imagen guardada en: resized_images/Titian/Titian_129 resized.jpg
Imagen guardada en: resized_images/Raphael/Raphael_33 resized.jpg
Imagen guardada en: resized_images/Vincent_van_Gogh/Vincent_van_Gogh_504
resized.jpg
Imagen guardada en: resized_images/Alfred_Sisley/Alfred_Sisley_225 resized.jpg
Imagen guardada en: resized_images/William_Turner/William_Turner_8 resized.jpg
```

```
Imagen guardada en: resized_images/Rembrandt/Rembrandt_251 resized.jpg
Imagen guardada en: resized_images/Piet_Mondrian/Piet_Mondrian_35 resized.jpg
Imagen guardada en: resized_images/Caravaggio/Caravaggio_28 resized.jpg
Imagen guardada en: resized_images/Michelangelo/Michelangelo_37 resized.jpg
Imagen guardada en: resized_images/Jan_van_Eyck/Jan_van_Eyck_35 resized.jpg
Imagen guardada en: resized_images/Gustave_Courbet/Gustave_Courbet_34
resized.jpg
Imagen guardada en: resized_images/Pieter_Bruegel/Pieter_Bruegel_43 resized.jpg
Imagen guardada en: resized_images/Giotto_di_Bondone/Giotto_di_Bondone_27
resized.jpg
Imagen guardada en: resized_images/Francisco_Goya/Francisco_Goya_268 resized.jpg
Imagen guardada en: resized_images/Diego_Velazquez/Diego_Velazquez_12
resized.jpg
Imagen guardada en: resized_images/Vasiliy_Kandinskiy/Vasiliy_Kandinskiy_42
resized.jpg
Imagen guardada en: resized_images/Edouard_Manet/Edouard_Manet_74 resized.jpg
Imagen guardada en: resized_images/Mikhail_Vrubel/Mikhail_Vrubel_7 resized.jpg
Imagen guardada en: resized_images/Leonardo_da_Vinci/Leonardo_da_Vinci_56
resized.jpg
```

1.2 Aumentar datos

Para aumentar datos a nuestro dataset lo que haremos será usar 5 formas para modificar las diferentes imágenes sin modificar su escencia. Cada uno de los siguiente modificadores aplica su efecto y aplitud del mismo de manera aleatoria, de tal manera que el modelo no crea que hay patrones en estos cambios.

1.2.1 Flip horinzontal y vertical

Lo que hace esto es voltear la imagen como si se tratase de un reflejo de un espejo-

[4]:
```python
def random_flip(img):
    # Flip horizontal
    if random.random() > 0.5:
        img = img.transpose(Image.FLIP_LEFT_RIGHT)
    # Flip vertical
    if random.random() > 0.5:
        img = img.transpose(Image.FLIP_TOP_BOTTOM)
    return img
```

1.2.2 Rotación

Esta si es la rotación común que gira la imagen respecto a su centro.

[5]:
```python
def random_rotation(img, max_rotation=25):
    # Rotación aleatoria entre -max_rotation y max_rotation grados
    angle = random.uniform(-max_rotation, max_rotation)
    img = img.rotate(angle)
    return img
```

1.2.3 Zoom

Acerca la imagen, y no necesariamente hacia el centro de la misma.

[6]:
```
def random_zoom(img, zoom_factor=0.2):
    # Recorte y redimensionamiento para aplicar un efecto de zoom
    width, height = img.size
    crop_width = int(width * (1 - zoom_factor))
    crop_height = int(height * (1 - zoom_factor))

    left = random.randint(0, width - crop_width)
    upper = random.randint(0, height - crop_height)

    img_cropped = img.crop((left, upper, left + crop_width, upper +␣
↪crop_height))
    img_resized = img_cropped.resize((width, height))

    return img_resized
```

1.2.4 Translación

Mueve la imagen de manera que esta quede con otra centralización. Puede quedar más para arriba, abajo, derecha o izquierda.

[7]:
```
def random_translation(img, max_tx=0.3, max_ty=0.3):
    # Traslación aleatoria de la imagen
    width, height = img.size
    tx = random.uniform(-max_tx, max_tx) * width
    ty = random.uniform(-max_ty, max_ty) * height

    img = img.transform(
        (width, height),
        Image.AFFINE,
        (1, 0, tx, 0, 1, ty),
        resample=Image.BICUBIC
    )
    return img
```

1.2.5 Contraste

El contraste es la medida que indica que tan distanciados están unos tonos de otros. Tanto como el alto contraste como el bajo puede hacer que la imagen pierda información. Sin embargo, esto permitiria al modelo reconcer patrones fuera de los tonos usados en las pinturas.

[8]:
```
def random_contrast(img, max_contrast=0.2):
    # Aumentar o disminuir el contraste de la imagen
    enhancer = ImageEnhance.Contrast(img)
    factor = random.uniform(1 - max_contrast, 1 + max_contrast)
    img = enhancer.enhance(factor)
```

```
    return img
```

1.2.6 Aplicar los efectos de manera aleatoria

Conviene aplicar los distintos efectos de forma aleatoria ya que si no, se podría estar creando patrones que no queremos que influyan en el aprendizaje de nuestro modelo. De momento se generarán 5 imagenes por cada imagen pasada como parámetro

[9]:
```
def generate_images_from_image(img, num_images=5):
    generated_images = [img]

    for _ in range(num_images - 1):
        new_img = img.copy()

        # Aplicar una combinación aleatoria de las transformaciones
        if random.random() > 0.5:
            new_img = random_flip(new_img)
        if random.random() > 0.5:
            new_img = random_rotation(new_img)
        if random.random() > 0.5:
            new_img = random_zoom(new_img)
        if random.random() > 0.5:
            new_img = random_translation(new_img)
        if random.random() > 0.5:
            new_img = random_contrast(new_img)

        generated_images.append(new_img)

    return generated_images
```

1.2.7 Procesar dataset

[10]:
```
def generate_images(input_path, output_path):
    dirs = os.listdir(input_path)
    for dirpath, dirnames, filenames in os.walk(input_path):
        if len(filenames) > 0:
            filename = filenames[0]
            if filename.lower().endswith(('.png', '.jpg', '.jpeg')):
                file_path = os.path.join(dirpath, filename)

                relative_dir = os.path.relpath(dirpath, input_path)
                new_dir = os.path.join(output_path, relative_dir)
                os.makedirs(new_dir, exist_ok=True)

                im = Image.open(file_path)
                generate_images_list = generate_images_from_image(im)

                for i, generate_image in enumerate(generate_images_list):
```

```
                    save_path = os.path.join(new_dir, f'{os.path.
↪splitext(filename)[0]} {i} generated.jpg')
                    generate_image.save(save_path, 'JPEG', quality=90)
                    print(f'Imagen guardada en: {save_path}')
```

[11]:
```
generate_images('resized_images', 'generated_images')
```

```
Imagen guardada en: generated_images/Eugene_Delacroix/Eugene_Delacroix_16
resized 0 generated.jpg
Imagen guardada en: generated_images/Eugene_Delacroix/Eugene_Delacroix_16
resized 1 generated.jpg
Imagen guardada en: generated_images/Eugene_Delacroix/Eugene_Delacroix_16
resized 2 generated.jpg
Imagen guardada en: generated_images/Eugene_Delacroix/Eugene_Delacroix_16
resized 3 generated.jpg
Imagen guardada en: generated_images/Eugene_Delacroix/Eugene_Delacroix_16
resized 4 generated.jpg
Imagen guardada en: generated_images/Claude_Monet/Claude_Monet_54 resized 0
generated.jpg
Imagen guardada en: generated_images/Claude_Monet/Claude_Monet_54 resized 1
generated.jpg
Imagen guardada en: generated_images/Claude_Monet/Claude_Monet_54 resized 2
generated.jpg
Imagen guardada en: generated_images/Claude_Monet/Claude_Monet_54 resized 3
generated.jpg
Imagen guardada en: generated_images/Claude_Monet/Claude_Monet_54 resized 4
generated.jpg
Imagen guardada en: generated_images/Edvard_Munch/Edvard_Munch_41 resized 0
generated.jpg
Imagen guardada en: generated_images/Edvard_Munch/Edvard_Munch_41 resized 1
generated.jpg
Imagen guardada en: generated_images/Edvard_Munch/Edvard_Munch_41 resized 2
generated.jpg
Imagen guardada en: generated_images/Edvard_Munch/Edvard_Munch_41 resized 3
generated.jpg
Imagen guardada en: generated_images/Edvard_Munch/Edvard_Munch_41 resized 4
generated.jpg
Imagen guardada en: generated_images/Sandro_Botticelli/Sandro_Botticelli_159
resized 0 generated.jpg
Imagen guardada en: generated_images/Sandro_Botticelli/Sandro_Botticelli_159
resized 1 generated.jpg
Imagen guardada en: generated_images/Sandro_Botticelli/Sandro_Botticelli_159
resized 2 generated.jpg
Imagen guardada en: generated_images/Sandro_Botticelli/Sandro_Botticelli_159
resized 3 generated.jpg
Imagen guardada en: generated_images/Sandro_Botticelli/Sandro_Botticelli_159
resized 4 generated.jpg
```

```
Imagen guardada en: generated_images/Paul_Klee/Paul_Klee_139 resized 0
generated.jpg
Imagen guardada en: generated_images/Paul_Klee/Paul_Klee_139 resized 1
generated.jpg
Imagen guardada en: generated_images/Paul_Klee/Paul_Klee_139 resized 2
generated.jpg
Imagen guardada en: generated_images/Paul_Klee/Paul_Klee_139 resized 3
generated.jpg
Imagen guardada en: generated_images/Paul_Klee/Paul_Klee_139 resized 4
generated.jpg
Imagen guardada en: generated_images/Albrecht_Du êrer/Albrecht_Du êrer_127
resized 0 generated.jpg
Imagen guardada en: generated_images/Albrecht_Du êrer/Albrecht_Du êrer_127
resized 1 generated.jpg
Imagen guardada en: generated_images/Albrecht_Du êrer/Albrecht_Du êrer_127
resized 2 generated.jpg
Imagen guardada en: generated_images/Albrecht_Du êrer/Albrecht_Du êrer_127
resized 3 generated.jpg
Imagen guardada en: generated_images/Albrecht_Du êrer/Albrecht_Du êrer_127
resized 4 generated.jpg
Imagen guardada en: generated_images/Georges_Seurat/Georges_Seurat_34 resized 0
generated.jpg
Imagen guardada en: generated_images/Georges_Seurat/Georges_Seurat_34 resized 1
generated.jpg
Imagen guardada en: generated_images/Georges_Seurat/Georges_Seurat_34 resized 2
generated.jpg
Imagen guardada en: generated_images/Georges_Seurat/Georges_Seurat_34 resized 3
generated.jpg
Imagen guardada en: generated_images/Georges_Seurat/Georges_Seurat_34 resized 4
generated.jpg
Imagen guardada en: generated_images/Jackson_Pollock/Jackson_Pollock_18 resized
0 generated.jpg
Imagen guardada en: generated_images/Jackson_Pollock/Jackson_Pollock_18 resized
1 generated.jpg
Imagen guardada en: generated_images/Jackson_Pollock/Jackson_Pollock_18 resized
2 generated.jpg
Imagen guardada en: generated_images/Jackson_Pollock/Jackson_Pollock_18 resized
3 generated.jpg
Imagen guardada en: generated_images/Jackson_Pollock/Jackson_Pollock_18 resized
4 generated.jpg
Imagen guardada en: generated_images/Edgar_Degas/Edgar_Degas_305 resized 0
generated.jpg
Imagen guardada en: generated_images/Edgar_Degas/Edgar_Degas_305 resized 1
generated.jpg
Imagen guardada en: generated_images/Edgar_Degas/Edgar_Degas_305 resized 2
generated.jpg
Imagen guardada en: generated_images/Edgar_Degas/Edgar_Degas_305 resized 3
generated.jpg
```

```
Imagen guardada en: generated_images/Edgar_Degas/Edgar_Degas_305 resized 4
generated.jpg
Imagen guardada en: generated_images/Rene_Magritte/Rene_Magritte_129 resized 0
generated.jpg
Imagen guardada en: generated_images/Rene_Magritte/Rene_Magritte_129 resized 1
generated.jpg
Imagen guardada en: generated_images/Rene_Magritte/Rene_Magritte_129 resized 2
generated.jpg
Imagen guardada en: generated_images/Rene_Magritte/Rene_Magritte_129 resized 3
generated.jpg
Imagen guardada en: generated_images/Rene_Magritte/Rene_Magritte_129 resized 4
generated.jpg
Imagen guardada en: generated_images/Andrei_Rublev/Andrei_Rublev_28 resized 0
generated.jpg
Imagen guardada en: generated_images/Andrei_Rublev/Andrei_Rublev_28 resized 1
generated.jpg
Imagen guardada en: generated_images/Andrei_Rublev/Andrei_Rublev_28 resized 2
generated.jpg
Imagen guardada en: generated_images/Andrei_Rublev/Andrei_Rublev_28 resized 3
generated.jpg
Imagen guardada en: generated_images/Andrei_Rublev/Andrei_Rublev_28 resized 4
generated.jpg
Imagen guardada en: generated_images/Pierre-Auguste_Renoir/Pierre-
Auguste_Renoir_261 resized 0 generated.jpg
Imagen guardada en: generated_images/Pierre-Auguste_Renoir/Pierre-
Auguste_Renoir_261 resized 1 generated.jpg
Imagen guardada en: generated_images/Pierre-Auguste_Renoir/Pierre-
Auguste_Renoir_261 resized 2 generated.jpg
Imagen guardada en: generated_images/Pierre-Auguste_Renoir/Pierre-
Auguste_Renoir_261 resized 3 generated.jpg
Imagen guardada en: generated_images/Pierre-Auguste_Renoir/Pierre-
Auguste_Renoir_261 resized 4 generated.jpg
Imagen guardada en: generated_images/Hieronymus_Bosch/Hieronymus_Bosch_26
resized 0 generated.jpg
Imagen guardada en: generated_images/Hieronymus_Bosch/Hieronymus_Bosch_26
resized 1 generated.jpg
Imagen guardada en: generated_images/Hieronymus_Bosch/Hieronymus_Bosch_26
resized 2 generated.jpg
Imagen guardada en: generated_images/Hieronymus_Bosch/Hieronymus_Bosch_26
resized 3 generated.jpg
Imagen guardada en: generated_images/Hieronymus_Bosch/Hieronymus_Bosch_26
resized 4 generated.jpg
Imagen guardada en: generated_images/Henri_de_Toulouse-
Lautrec/Henri_de_Toulouse-Lautrec_67 resized 0 generated.jpg
Imagen guardada en: generated_images/Henri_de_Toulouse-
Lautrec/Henri_de_Toulouse-Lautrec_67 resized 1 generated.jpg
Imagen guardada en: generated_images/Henri_de_Toulouse-
Lautrec/Henri_de_Toulouse-Lautrec_67 resized 2 generated.jpg
```

```
Imagen guardada en: generated_images/Henri_de_Toulouse-
Lautrec/Henri_de_Toulouse-Lautrec_67 resized 3 generated.jpg
Imagen guardada en: generated_images/Henri_de_Toulouse-
Lautrec/Henri_de_Toulouse-Lautrec_67 resized 4 generated.jpg
Imagen guardada en: generated_images/Pablo_Picasso/Pablo_Picasso_375 resized 0
generated.jpg
Imagen guardada en: generated_images/Pablo_Picasso/Pablo_Picasso_375 resized 1
generated.jpg
Imagen guardada en: generated_images/Pablo_Picasso/Pablo_Picasso_375 resized 2
generated.jpg
Imagen guardada en: generated_images/Pablo_Picasso/Pablo_Picasso_375 resized 3
generated.jpg
Imagen guardada en: generated_images/Pablo_Picasso/Pablo_Picasso_375 resized 4
generated.jpg
Imagen guardada en: generated_images/Andy_Warhol/Andy_Warhol_139 resized 0
generated.jpg
Imagen guardada en: generated_images/Andy_Warhol/Andy_Warhol_139 resized 1
generated.jpg
Imagen guardada en: generated_images/Andy_Warhol/Andy_Warhol_139 resized 2
generated.jpg
Imagen guardada en: generated_images/Andy_Warhol/Andy_Warhol_139 resized 3
generated.jpg
Imagen guardada en: generated_images/Andy_Warhol/Andy_Warhol_139 resized 4
generated.jpg
Imagen guardada en: generated_images/Kazimir_Malevich/Kazimir_Malevich_22
resized 0 generated.jpg
Imagen guardada en: generated_images/Kazimir_Malevich/Kazimir_Malevich_22
resized 1 generated.jpg
Imagen guardada en: generated_images/Kazimir_Malevich/Kazimir_Malevich_22
resized 2 generated.jpg
Imagen guardada en: generated_images/Kazimir_Malevich/Kazimir_Malevich_22
resized 3 generated.jpg
Imagen guardada en: generated_images/Kazimir_Malevich/Kazimir_Malevich_22
resized 4 generated.jpg
Imagen guardada en: generated_images/Gustav_Klimt/Gustav_Klimt_29 resized 0
generated.jpg
Imagen guardada en: generated_images/Gustav_Klimt/Gustav_Klimt_29 resized 1
generated.jpg
Imagen guardada en: generated_images/Gustav_Klimt/Gustav_Klimt_29 resized 2
generated.jpg
Imagen guardada en: generated_images/Gustav_Klimt/Gustav_Klimt_29 resized 3
generated.jpg
Imagen guardada en: generated_images/Gustav_Klimt/Gustav_Klimt_29 resized 4
generated.jpg
Imagen guardada en: generated_images/Joan_Miro/Joan_Miro_81 resized 0
generated.jpg
Imagen guardada en: generated_images/Joan_Miro/Joan_Miro_81 resized 1
generated.jpg
```

```
Imagen guardada en: generated_images/Joan_Miro/Joan_Miro_81 resized 2
generated.jpg
Imagen guardada en: generated_images/Joan_Miro/Joan_Miro_81 resized 3
generated.jpg
Imagen guardada en: generated_images/Joan_Miro/Joan_Miro_81 resized 4
generated.jpg
Imagen guardada en: generated_images/Albrecht_Dürer/Albrecht_Dürer_43 resized
0 generated.jpg
Imagen guardada en: generated_images/Albrecht_Dürer/Albrecht_Dürer_43 resized
1 generated.jpg
Imagen guardada en: generated_images/Albrecht_Dürer/Albrecht_Dürer_43 resized
2 generated.jpg
Imagen guardada en: generated_images/Albrecht_Dürer/Albrecht_Dürer_43 resized
3 generated.jpg
Imagen guardada en: generated_images/Albrecht_Dürer/Albrecht_Dürer_43 resized
4 generated.jpg
Imagen guardada en: generated_images/Camille_Pissarro/Camille_Pissarro_1 resized
0 generated.jpg
Imagen guardada en: generated_images/Camille_Pissarro/Camille_Pissarro_1 resized
1 generated.jpg
Imagen guardada en: generated_images/Camille_Pissarro/Camille_Pissarro_1 resized
2 generated.jpg
Imagen guardada en: generated_images/Camille_Pissarro/Camille_Pissarro_1 resized
3 generated.jpg
Imagen guardada en: generated_images/Camille_Pissarro/Camille_Pissarro_1 resized
4 generated.jpg
Imagen guardada en: generated_images/Diego_Rivera/Diego_Rivera_34 resized 0
generated.jpg
Imagen guardada en: generated_images/Diego_Rivera/Diego_Rivera_34 resized 1
generated.jpg
Imagen guardada en: generated_images/Diego_Rivera/Diego_Rivera_34 resized 2
generated.jpg
Imagen guardada en: generated_images/Diego_Rivera/Diego_Rivera_34 resized 3
generated.jpg
Imagen guardada en: generated_images/Diego_Rivera/Diego_Rivera_34 resized 4
generated.jpg
Imagen guardada en: generated_images/Marc_Chagall/Marc_Chagall_194 resized 0
generated.jpg
Imagen guardada en: generated_images/Marc_Chagall/Marc_Chagall_194 resized 1
generated.jpg
Imagen guardada en: generated_images/Marc_Chagall/Marc_Chagall_194 resized 2
generated.jpg
Imagen guardada en: generated_images/Marc_Chagall/Marc_Chagall_194 resized 3
generated.jpg
Imagen guardada en: generated_images/Marc_Chagall/Marc_Chagall_194 resized 4
generated.jpg
Imagen guardada en: generated_images/Frida_Kahlo/Frida_Kahlo_41 resized 0
generated.jpg
```

```
Imagen guardada en: generated_images/Frida_Kahlo/Frida_Kahlo_41 resized 1
generated.jpg
Imagen guardada en: generated_images/Frida_Kahlo/Frida_Kahlo_41 resized 2
generated.jpg
Imagen guardada en: generated_images/Frida_Kahlo/Frida_Kahlo_41 resized 3
generated.jpg
Imagen guardada en: generated_images/Frida_Kahlo/Frida_Kahlo_41 resized 4
generated.jpg
Imagen guardada en: generated_images/Henri_Rousseau/Henri_Rousseau_58 resized 0
generated.jpg
Imagen guardada en: generated_images/Henri_Rousseau/Henri_Rousseau_58 resized 1
generated.jpg
Imagen guardada en: generated_images/Henri_Rousseau/Henri_Rousseau_58 resized 2
generated.jpg
Imagen guardada en: generated_images/Henri_Rousseau/Henri_Rousseau_58 resized 3
generated.jpg
Imagen guardada en: generated_images/Henri_Rousseau/Henri_Rousseau_58 resized 4
generated.jpg
Imagen guardada en: generated_images/Amedeo_Modigliani/Amedeo_Modigliani_27
resized 0 generated.jpg
Imagen guardada en: generated_images/Amedeo_Modigliani/Amedeo_Modigliani_27
resized 1 generated.jpg
Imagen guardada en: generated_images/Amedeo_Modigliani/Amedeo_Modigliani_27
resized 2 generated.jpg
Imagen guardada en: generated_images/Amedeo_Modigliani/Amedeo_Modigliani_27
resized 3 generated.jpg
Imagen guardada en: generated_images/Amedeo_Modigliani/Amedeo_Modigliani_27
resized 4 generated.jpg
Imagen guardada en: generated_images/Peter_Paul_Rubens/Peter_Paul_Rubens_74
resized 0 generated.jpg
Imagen guardada en: generated_images/Peter_Paul_Rubens/Peter_Paul_Rubens_74
resized 1 generated.jpg
Imagen guardada en: generated_images/Peter_Paul_Rubens/Peter_Paul_Rubens_74
resized 2 generated.jpg
Imagen guardada en: generated_images/Peter_Paul_Rubens/Peter_Paul_Rubens_74
resized 3 generated.jpg
Imagen guardada en: generated_images/Peter_Paul_Rubens/Peter_Paul_Rubens_74
resized 4 generated.jpg
Imagen guardada en: generated_images/El_Greco/El_Greco_14 resized 0
generated.jpg
Imagen guardada en: generated_images/El_Greco/El_Greco_14 resized 1
generated.jpg
Imagen guardada en: generated_images/El_Greco/El_Greco_14 resized 2
generated.jpg
Imagen guardada en: generated_images/El_Greco/El_Greco_14 resized 3
generated.jpg
Imagen guardada en: generated_images/El_Greco/El_Greco_14 resized 4
generated.jpg
```

```
Imagen guardada en: generated_images/Paul_Gauguin/Paul_Gauguin_250 resized 0
generated.jpg
Imagen guardada en: generated_images/Paul_Gauguin/Paul_Gauguin_250 resized 1
generated.jpg
Imagen guardada en: generated_images/Paul_Gauguin/Paul_Gauguin_250 resized 2
generated.jpg
Imagen guardada en: generated_images/Paul_Gauguin/Paul_Gauguin_250 resized 3
generated.jpg
Imagen guardada en: generated_images/Paul_Gauguin/Paul_Gauguin_250 resized 4
generated.jpg
Imagen guardada en: generated_images/Salvador_Dali/Salvador_Dali_109 resized 0
generated.jpg
Imagen guardada en: generated_images/Salvador_Dali/Salvador_Dali_109 resized 1
generated.jpg
Imagen guardada en: generated_images/Salvador_Dali/Salvador_Dali_109 resized 2
generated.jpg
Imagen guardada en: generated_images/Salvador_Dali/Salvador_Dali_109 resized 3
generated.jpg
Imagen guardada en: generated_images/Salvador_Dali/Salvador_Dali_109 resized 4
generated.jpg
Imagen guardada en: generated_images/Henri_Matisse/Henri_Matisse_158 resized 0
generated.jpg
Imagen guardada en: generated_images/Henri_Matisse/Henri_Matisse_158 resized 1
generated.jpg
Imagen guardada en: generated_images/Henri_Matisse/Henri_Matisse_158 resized 2
generated.jpg
Imagen guardada en: generated_images/Henri_Matisse/Henri_Matisse_158 resized 3
generated.jpg
Imagen guardada en: generated_images/Henri_Matisse/Henri_Matisse_158 resized 4
generated.jpg
Imagen guardada en: generated_images/Paul_Cezanne/Paul_Cezanne_38 resized 0
generated.jpg
Imagen guardada en: generated_images/Paul_Cezanne/Paul_Cezanne_38 resized 1
generated.jpg
Imagen guardada en: generated_images/Paul_Cezanne/Paul_Cezanne_38 resized 2
generated.jpg
Imagen guardada en: generated_images/Paul_Cezanne/Paul_Cezanne_38 resized 3
generated.jpg
Imagen guardada en: generated_images/Paul_Cezanne/Paul_Cezanne_38 resized 4
generated.jpg
Imagen guardada en: generated_images/Titian/Titian_129 resized 0 generated.jpg
Imagen guardada en: generated_images/Titian/Titian_129 resized 1 generated.jpg
Imagen guardada en: generated_images/Titian/Titian_129 resized 2 generated.jpg
Imagen guardada en: generated_images/Titian/Titian_129 resized 3 generated.jpg
Imagen guardada en: generated_images/Titian/Titian_129 resized 4 generated.jpg
Imagen guardada en: generated_images/Raphael/Raphael_33 resized 0 generated.jpg
Imagen guardada en: generated_images/Raphael/Raphael_33 resized 1 generated.jpg
Imagen guardada en: generated_images/Raphael/Raphael_33 resized 2 generated.jpg
```

```
Imagen guardada en: generated_images/Raphael/Raphael_33 resized 3 generated.jpg
Imagen guardada en: generated_images/Raphael/Raphael_33 resized 4 generated.jpg
Imagen guardada en: generated_images/Vincent_van_Gogh/Vincent_van_Gogh_504
resized 0 generated.jpg
Imagen guardada en: generated_images/Vincent_van_Gogh/Vincent_van_Gogh_504
resized 1 generated.jpg
Imagen guardada en: generated_images/Vincent_van_Gogh/Vincent_van_Gogh_504
resized 2 generated.jpg
Imagen guardada en: generated_images/Vincent_van_Gogh/Vincent_van_Gogh_504
resized 3 generated.jpg
Imagen guardada en: generated_images/Vincent_van_Gogh/Vincent_van_Gogh_504
resized 4 generated.jpg
Imagen guardada en: generated_images/Alfred_Sisley/Alfred_Sisley_225 resized 0
generated.jpg
Imagen guardada en: generated_images/Alfred_Sisley/Alfred_Sisley_225 resized 1
generated.jpg
Imagen guardada en: generated_images/Alfred_Sisley/Alfred_Sisley_225 resized 2
generated.jpg
Imagen guardada en: generated_images/Alfred_Sisley/Alfred_Sisley_225 resized 3
generated.jpg
Imagen guardada en: generated_images/Alfred_Sisley/Alfred_Sisley_225 resized 4
generated.jpg
Imagen guardada en: generated_images/William_Turner/William_Turner_8 resized 0
generated.jpg
Imagen guardada en: generated_images/William_Turner/William_Turner_8 resized 1
generated.jpg
Imagen guardada en: generated_images/William_Turner/William_Turner_8 resized 2
generated.jpg
Imagen guardada en: generated_images/William_Turner/William_Turner_8 resized 3
generated.jpg
Imagen guardada en: generated_images/William_Turner/William_Turner_8 resized 4
generated.jpg
Imagen guardada en: generated_images/Rembrandt/Rembrandt_251 resized 0
generated.jpg
Imagen guardada en: generated_images/Rembrandt/Rembrandt_251 resized 1
generated.jpg
Imagen guardada en: generated_images/Rembrandt/Rembrandt_251 resized 2
generated.jpg
Imagen guardada en: generated_images/Rembrandt/Rembrandt_251 resized 3
generated.jpg
Imagen guardada en: generated_images/Rembrandt/Rembrandt_251 resized 4
generated.jpg
Imagen guardada en: generated_images/Piet_Mondrian/Piet_Mondrian_35 resized 0
generated.jpg
Imagen guardada en: generated_images/Piet_Mondrian/Piet_Mondrian_35 resized 1
generated.jpg
Imagen guardada en: generated_images/Piet_Mondrian/Piet_Mondrian_35 resized 2
generated.jpg
```

```
Imagen guardada en: generated_images/Piet_Mondrian/Piet_Mondrian_35 resized 3
generated.jpg
Imagen guardada en: generated_images/Piet_Mondrian/Piet_Mondrian_35 resized 4
generated.jpg
Imagen guardada en: generated_images/Caravaggio/Caravaggio_28 resized 0
generated.jpg
Imagen guardada en: generated_images/Caravaggio/Caravaggio_28 resized 1
generated.jpg
Imagen guardada en: generated_images/Caravaggio/Caravaggio_28 resized 2
generated.jpg
Imagen guardada en: generated_images/Caravaggio/Caravaggio_28 resized 3
generated.jpg
Imagen guardada en: generated_images/Caravaggio/Caravaggio_28 resized 4
generated.jpg
Imagen guardada en: generated_images/Michelangelo/Michelangelo_37 resized 0
generated.jpg
Imagen guardada en: generated_images/Michelangelo/Michelangelo_37 resized 1
generated.jpg
Imagen guardada en: generated_images/Michelangelo/Michelangelo_37 resized 2
generated.jpg
Imagen guardada en: generated_images/Michelangelo/Michelangelo_37 resized 3
generated.jpg
Imagen guardada en: generated_images/Michelangelo/Michelangelo_37 resized 4
generated.jpg
Imagen guardada en: generated_images/Jan_van_Eyck/Jan_van_Eyck_35 resized 0
generated.jpg
Imagen guardada en: generated_images/Jan_van_Eyck/Jan_van_Eyck_35 resized 1
generated.jpg
Imagen guardada en: generated_images/Jan_van_Eyck/Jan_van_Eyck_35 resized 2
generated.jpg
Imagen guardada en: generated_images/Jan_van_Eyck/Jan_van_Eyck_35 resized 3
generated.jpg
Imagen guardada en: generated_images/Jan_van_Eyck/Jan_van_Eyck_35 resized 4
generated.jpg
Imagen guardada en: generated_images/Gustave_Courbet/Gustave_Courbet_34 resized
0 generated.jpg
Imagen guardada en: generated_images/Gustave_Courbet/Gustave_Courbet_34 resized
1 generated.jpg
Imagen guardada en: generated_images/Gustave_Courbet/Gustave_Courbet_34 resized
2 generated.jpg
Imagen guardada en: generated_images/Gustave_Courbet/Gustave_Courbet_34 resized
3 generated.jpg
Imagen guardada en: generated_images/Gustave_Courbet/Gustave_Courbet_34 resized
4 generated.jpg
Imagen guardada en: generated_images/Pieter_Bruegel/Pieter_Bruegel_43 resized 0
generated.jpg
Imagen guardada en: generated_images/Pieter_Bruegel/Pieter_Bruegel_43 resized 1
generated.jpg
```

```
Imagen guardada en: generated_images/Pieter_Bruegel/Pieter_Bruegel_43 resized 2
generated.jpg
Imagen guardada en: generated_images/Pieter_Bruegel/Pieter_Bruegel_43 resized 3
generated.jpg
Imagen guardada en: generated_images/Pieter_Bruegel/Pieter_Bruegel_43 resized 4
generated.jpg
Imagen guardada en: generated_images/Giotto_di_Bondone/Giotto_di_Bondone_27
resized 0 generated.jpg
Imagen guardada en: generated_images/Giotto_di_Bondone/Giotto_di_Bondone_27
resized 1 generated.jpg
Imagen guardada en: generated_images/Giotto_di_Bondone/Giotto_di_Bondone_27
resized 2 generated.jpg
Imagen guardada en: generated_images/Giotto_di_Bondone/Giotto_di_Bondone_27
resized 3 generated.jpg
Imagen guardada en: generated_images/Giotto_di_Bondone/Giotto_di_Bondone_27
resized 4 generated.jpg
Imagen guardada en: generated_images/Francisco_Goya/Francisco_Goya_268 resized 0
generated.jpg
Imagen guardada en: generated_images/Francisco_Goya/Francisco_Goya_268 resized 1
generated.jpg
Imagen guardada en: generated_images/Francisco_Goya/Francisco_Goya_268 resized 2
generated.jpg
Imagen guardada en: generated_images/Francisco_Goya/Francisco_Goya_268 resized 3
generated.jpg
Imagen guardada en: generated_images/Francisco_Goya/Francisco_Goya_268 resized 4
generated.jpg
Imagen guardada en: generated_images/Diego_Velazquez/Diego_Velazquez_12 resized
0 generated.jpg
Imagen guardada en: generated_images/Diego_Velazquez/Diego_Velazquez_12 resized
1 generated.jpg
Imagen guardada en: generated_images/Diego_Velazquez/Diego_Velazquez_12 resized
2 generated.jpg
Imagen guardada en: generated_images/Diego_Velazquez/Diego_Velazquez_12 resized
3 generated.jpg
Imagen guardada en: generated_images/Diego_Velazquez/Diego_Velazquez_12 resized
4 generated.jpg
Imagen guardada en: generated_images/Vasiliy_Kandinskiy/Vasiliy_Kandinskiy_42
resized 0 generated.jpg
Imagen guardada en: generated_images/Vasiliy_Kandinskiy/Vasiliy_Kandinskiy_42
resized 1 generated.jpg
Imagen guardada en: generated_images/Vasiliy_Kandinskiy/Vasiliy_Kandinskiy_42
resized 2 generated.jpg
Imagen guardada en: generated_images/Vasiliy_Kandinskiy/Vasiliy_Kandinskiy_42
resized 3 generated.jpg
Imagen guardada en: generated_images/Vasiliy_Kandinskiy/Vasiliy_Kandinskiy_42
resized 4 generated.jpg
Imagen guardada en: generated_images/Edouard_Manet/Edouard_Manet_74 resized 0
generated.jpg
```

```
Imagen guardada en: generated_images/Edouard_Manet/Edouard_Manet_74 resized 1
generated.jpg
Imagen guardada en: generated_images/Edouard_Manet/Edouard_Manet_74 resized 2
generated.jpg
Imagen guardada en: generated_images/Edouard_Manet/Edouard_Manet_74 resized 3
generated.jpg
Imagen guardada en: generated_images/Edouard_Manet/Edouard_Manet_74 resized 4
generated.jpg
Imagen guardada en: generated_images/Mikhail_Vrubel/Mikhail_Vrubel_7 resized 0
generated.jpg
Imagen guardada en: generated_images/Mikhail_Vrubel/Mikhail_Vrubel_7 resized 1
generated.jpg
Imagen guardada en: generated_images/Mikhail_Vrubel/Mikhail_Vrubel_7 resized 2
generated.jpg
Imagen guardada en: generated_images/Mikhail_Vrubel/Mikhail_Vrubel_7 resized 3
generated.jpg
Imagen guardada en: generated_images/Mikhail_Vrubel/Mikhail_Vrubel_7 resized 4
generated.jpg
Imagen guardada en: generated_images/Leonardo_da_Vinci/Leonardo_da_Vinci_56
resized 0 generated.jpg
Imagen guardada en: generated_images/Leonardo_da_Vinci/Leonardo_da_Vinci_56
resized 1 generated.jpg
Imagen guardada en: generated_images/Leonardo_da_Vinci/Leonardo_da_Vinci_56
resized 2 generated.jpg
Imagen guardada en: generated_images/Leonardo_da_Vinci/Leonardo_da_Vinci_56
resized 3 generated.jpg
Imagen guardada en: generated_images/Leonardo_da_Vinci/Leonardo_da_Vinci_56
resized 4 generated.jpg
```

1.3 Normalizar los datos de 0 a 1

Muchos modelo funciona mejor cuando los datos están normalizados. Ya sea de -1 a 1, o de 0 a 1 (lo más común) los datos normalizados ayudan a que estos modelos puedan trabajar de manera más eficiente con los datos, si que estos pierdan la información relevante. Mismo caso que el anterior: Por motivos de costos computacionales, solo se trabajará sobre una imagen por carpeta. Hay que tomar en cuenta que muchas imágenes no se pueden guardar con valores de entre 0 y 1, si no, valores enteros de entre 0 y 255. Por lo tanto guardaremos las imágenes normalizadas como arreglos de numpy.

[13]:
```python
def normalize_and_resize(input_path, output_path):
    dirs = os.listdir(input_path)
    for dirpath, dirnames, filenames in os.walk(input_path):
        if len(filenames) > 0:
            filename = filenames[0]
            if filename.lower().endswith(('.png', '.jpg', '.jpeg')):
                file_path = os.path.join(dirpath, filename)

                relative_dir = os.path.relpath(dirpath, input_path)
```

```
                new_dir = os.path.join(output_path, relative_dir)
                os.makedirs(new_dir, exist_ok=True)

                im = Image.open(file_path)
                img_array = np.array(im)

                img_normalized = img_array / 255.0

                npy_save_path = os.path.join(new_dir, f'{os.path.
↪splitext(filename)[0]} normalized.npy')
                np.save(npy_save_path, img_normalized)
                print(f'Arreglo guardado en: {npy_save_path}')
```

[14]:
```
normalize_and_resize('generated_images', 'normalized_images')
```

```
Arreglo guardado en: normalized_images/Eugene_Delacroix/Eugene_Delacroix_16
resized 3 generated normalized.npy
Arreglo guardado en: normalized_images/Claude_Monet/Claude_Monet_54 resized 3
generated normalized.npy
Arreglo guardado en: normalized_images/Edvard_Munch/Edvard_Munch_41 resized 2
generated normalized.npy
Arreglo guardado en: normalized_images/Sandro_Botticelli/Sandro_Botticelli_159
resized 2 generated normalized.npy
Arreglo guardado en: normalized_images/Paul_Klee/Paul_Klee_139 resized 3
generated normalized.npy
Arreglo guardado en: normalized_images/Albrecht_Du êrer/Albrecht_Du êrer_127
resized 3 generated normalized.npy
Arreglo guardado en: normalized_images/Georges_Seurat/Georges_Seurat_34 resized
4 generated normalized.npy
Arreglo guardado en: normalized_images/Jackson_Pollock/Jackson_Pollock_18
resized 4 generated normalized.npy
Arreglo guardado en: normalized_images/Edgar_Degas/Edgar_Degas_305 resized 2
generated normalized.npy
Arreglo guardado en: normalized_images/Rene_Magritte/Rene_Magritte_129 resized 0
generated normalized.npy
Arreglo guardado en: normalized_images/Andrei_Rublev/Andrei_Rublev_28 resized 1
generated normalized.npy
Arreglo guardado en: normalized_images/Pierre-Auguste_Renoir/Pierre-
Auguste_Renoir_261 resized 4 generated normalized.npy
Arreglo guardado en: normalized_images/Hieronymus_Bosch/Hieronymus_Bosch_26
resized 3 generated normalized.npy
Arreglo guardado en: normalized_images/Henri_de_Toulouse-
Lautrec/Henri_de_Toulouse-Lautrec_67 resized 0 generated normalized.npy
Arreglo guardado en: normalized_images/Pablo_Picasso/Pablo_Picasso_375 resized 4
generated normalized.npy
Arreglo guardado en: normalized_images/Andy_Warhol/Andy_Warhol_139 resized 0
generated normalized.npy
Arreglo guardado en: normalized_images/Kazimir_Malevich/Kazimir_Malevich_22
```

```
resized 3 generated normalized.npy
Arreglo guardado en: normalized_images/Gustav_Klimt/Gustav_Klimt_29 resized 2
generated normalized.npy
Arreglo guardado en: normalized_images/Joan_Miro/Joan_Miro_81 resized 4
generated normalized.npy
Arreglo guardado en: normalized_images/Albrecht_Dürer/Albrecht_Dürer_43
resized 0 generated normalized.npy
Arreglo guardado en: normalized_images/Camille_Pissarro/Camille_Pissarro_1
resized 3 generated normalized.npy
Arreglo guardado en: normalized_images/Diego_Rivera/Diego_Rivera_34 resized 0
generated normalized.npy
Arreglo guardado en: normalized_images/Marc_Chagall/Marc_Chagall_194 resized 2
generated normalized.npy
Arreglo guardado en: normalized_images/Frida_Kahlo/Frida_Kahlo_41 resized 1
generated normalized.npy
Arreglo guardado en: normalized_images/Henri_Rousseau/Henri_Rousseau_58 resized
2 generated normalized.npy
Arreglo guardado en: normalized_images/Amedeo_Modigliani/Amedeo_Modigliani_27
resized 2 generated normalized.npy
Arreglo guardado en: normalized_images/Peter_Paul_Rubens/Peter_Paul_Rubens_74
resized 2 generated normalized.npy
Arreglo guardado en: normalized_images/El_Greco/El_Greco_14 resized 0 generated
normalized.npy
Arreglo guardado en: normalized_images/Paul_Gauguin/Paul_Gauguin_250 resized 0
generated normalized.npy
Arreglo guardado en: normalized_images/Salvador_Dali/Salvador_Dali_109 resized 3
generated normalized.npy
Arreglo guardado en: normalized_images/Henri_Matisse/Henri_Matisse_158 resized 0
generated normalized.npy
Arreglo guardado en: normalized_images/Paul_Cezanne/Paul_Cezanne_38 resized 2
generated normalized.npy
Arreglo guardado en: normalized_images/Titian/Titian_129 resized 4 generated
normalized.npy
Arreglo guardado en: normalized_images/Raphael/Raphael_33 resized 1 generated
normalized.npy
Arreglo guardado en: normalized_images/Vincent_van_Gogh/Vincent_van_Gogh_504
resized 3 generated normalized.npy
Arreglo guardado en: normalized_images/Alfred_Sisley/Alfred_Sisley_225 resized 0
generated normalized.npy
Arreglo guardado en: normalized_images/William_Turner/William_Turner_8 resized 1
generated normalized.npy
Arreglo guardado en: normalized_images/Rembrandt/Rembrandt_251 resized 4
generated normalized.npy
Arreglo guardado en: normalized_images/Piet_Mondrian/Piet_Mondrian_35 resized 0
generated normalized.npy
Arreglo guardado en: normalized_images/Caravaggio/Caravaggio_28 resized 2
generated normalized.npy
Arreglo guardado en: normalized_images/Michelangelo/Michelangelo_37 resized 2
```

```
generated normalized.npy
Arreglo guardado en: normalized_images/Jan_van_Eyck/Jan_van_Eyck_35 resized 3
generated normalized.npy
Arreglo guardado en: normalized_images/Gustave_Courbet/Gustave_Courbet_34
resized 0 generated normalized.npy
Arreglo guardado en: normalized_images/Pieter_Bruegel/Pieter_Bruegel_43 resized
3 generated normalized.npy
Arreglo guardado en: normalized_images/Giotto_di_Bondone/Giotto_di_Bondone_27
resized 1 generated normalized.npy
Arreglo guardado en: normalized_images/Francisco_Goya/Francisco_Goya_268 resized
1 generated normalized.npy
Arreglo guardado en: normalized_images/Diego_Velazquez/Diego_Velazquez_12
resized 3 generated normalized.npy
Arreglo guardado en: normalized_images/Vasiliy_Kandinskiy/Vasiliy_Kandinskiy_42
resized 0 generated normalized.npy
Arreglo guardado en: normalized_images/Edouard_Manet/Edouard_Manet_74 resized 2
generated normalized.npy
Arreglo guardado en: normalized_images/Mikhail_Vrubel/Mikhail_Vrubel_7 resized 4
generated normalized.npy
Arreglo guardado en: normalized_images/Leonardo_da_Vinci/Leonardo_da_Vinci_56
resized 4 generated normalized.npy
```

1.4 Matriz de One Hot Encoder

[22]:
```python
def one_hot_encoder_labels(input_path):
    dirs = sorted([d for d in os.listdir(input_path) if os.path.isdir(os.path.
 ↪join(input_path, d))])

    class_to_index = {class_name: idx for idx, class_name in enumerate(dirs)}

    data = []

    for dir_name in dirs:
        folder_path = os.path.join(input_path, dir_name)
        for filename in os.listdir(folder_path):
            if filename.lower().endswith(('.png', '.jpg', '.jpeg')):
                one_hot = [0] * len(dirs)
                one_hot[class_to_index[dir_name]] = 1
                data.append({'image': filename, 'one_hot': one_hot})

    df = pd.DataFrame(data)
    return df
```

[23]:
```python
df = one_hot_encoder_labels('images')
print(df)
```

```
                        image  \
0      Albrecht_Dürer_43.jpg
```

```
1     Albrecht_Dürer_153.jpg
2     Albrecht_Dürer_180.jpg
3      Albrecht_Dürer_29.jpg
4     Albrecht_Dürer_237.jpg
…                       …
8769    William_Turner_11.jpg
8770    William_Turner_36.jpg
8771    William_Turner_28.jpg
8772    William_Turner_55.jpg
8773    William_Turner_49.jpg

                                                  one_hot
0     [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
1     [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
2     [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
3     [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
4     [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
…                                                   …
8769  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
8770  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
8771  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
8772  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
8773  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …

[8774 rows x 2 columns]
```

Mucho de lo que hicimos anteriormente no es necesario, ya que Tensorflow puede realizar estos proprocesamientos de manera mucho más sencilla.

4.2. Justificación del modelo

El modelo presentado es un clasificador basado en una red neuronal convolucional (CNN) diseñada para identificar al artista de una pintura. Se basa en el ejemplo del usuario **mkkoehler** de la plataforma `www.kaggle.com` [4] A continuación, se explican las razones detrás de las decisiones de diseño del modelo y su configuración:

4.2.1. Razones para usar este modelo

Figura 2: Convolución aplicada sobre la imagen de un gato.

- **Problema de clasificación de imágenes**:

  El problema requiere analizar imágenes para clasificarlas según múltiples categorías (en este caso artistas). Las redes convolucionales son ideales para este propósito debido a su capacidad para detectar patrones espaciales y características visuales en imágenes.

  Como se puede ver en la Figura 2 la convolución actua como un filtro. Gracias a estos filtros es posible que una máquina pueda detectar cosas como los bordes, los colores, las formas, y puede ser que se abstraingan tanto que empieza a ver atributos que nosotros no detectamos, o no podemos nombrar con tanta facilidad, las llamadas características de alto nivel.

  Por ejemeplo, en un paper famosos se uso un dataset de 1.2 millones de imágenes. Sin embargo recuerda que a pesar de los grande logros que tuvo las CNN's en datasets como el MNIST, la implementación para dataset de alta calidad aun son computacionalmente costosas.

  La arquitectura del modelo expuesto en el paper se compone de 8 capas entrenadas, 5 convolucionales, y 3 capas totalmente conectadas.

  Esta es la arquitectura de AlexNet [7] (Figura 2). Y de hecho, el número de capas a usar en nuestro modelo como indica el siguiente fragmento, no sigue a unas reglas estrictas, si no más bien que se basan en otras arquitecturas ya probadas.

  > There are no strict rules on the number of layers which are incorporated in the network model. However, in most cases, two to four layers have been observed in different architectures including LeNet, AlexNet, and VGG Net [8]

  Así pues, tomaremos muchas cosas que ya estan en AlexNet como lo son:

  - **Data Augmentation**:
    Aumentar los datos artificialmente aplicando transformaciones que no afecten la escencia de las imágenes, como rotaciones, acercamiento y volteados
  - **Scaling**:
    Normalizar los valores de los píxeles de un rango de [0, 255] a [0, 1] para facilitar el entrenamiento del modelo.
  - **Primera capa convolucional**:
    La capa convolucional tendra 16 kernels de 3×3.
  - **Primera capa MaxPooling2D**:
    Esta capa reduce dimensionalidad al seleccionar el valor máximo en regiones específicas, reteniendo la información más importante mientras disminuye la carga computacional.
  - **Segunda capa convolucional**:
    La capa convolucional tendra 32 kernels de 3×3.
  - **Segunda capa MaxPooling2D**
  - **Tercera capa convolucional**:
    La capa convolucional tendra 64 kernels de 3×3.
  - **Tercera capa MaxPooling2D**
  - **Capa Dropout**:
    Esta capa apaga el 20 % de las neuronal aleatoriamente. Esto hace que las neuronas sean más robustas a la hora de detectar patrones y también ayuda ayuda a prevenir el sobreajuste

- **Capa de aplanamiento**:
  Convierte la salida tridimensional de las capas convolucionales en un vector unidimensional, permitiendo que las capas densas procesen la información.
- **Primera capa densa**
  Esta capa se conectará con todas las salidas anteriores, y tendrá 128 neuronas con la función de activación **ReLu**. Esta recibira cada una de las características que pudo extraer las capas convolucionales.
- **Segunda capa densa, salida**
  Esta capa tendrá la misma cantidad de clases como neuronas de salida. Hablamos de 49 clases. Cada neurona se conecta con todas la anterior y se activa cuando la imagen de entrada pertecene a la clase asignada a la neurona. Para esta última capa usaremos las función de activación **Softmax** para poder ver los porcentajes de salida.

Entonces nuestro modelo tendrá la siguiente estrucuta:

```
data_augmentation = keras.Sequential(
  [
    layers.experimental.
    .preprocessing.
      .RandomFlip
      ("horizontal_and_vertical"),
    layers.experimental.
    .preprocessing.
      .RandomRotation(0.25),
    layers.experimental.
    .preprocessing.
      .RandomZoom(0.2),
    layers.experimental.
    .preprocessing.
      .RandomTranslation(0.3,0.2),
    layers.experimental.
    .preprocessing.
      .RandomContrast(0.2)
  ]
)

CONV = 3

model = Sequential([
  data_augmentation,
  layers.experimental.
    .preprocessing.Rescaling(1./255),
  layers.Conv2D(16, CONV,
                padding='same',
                activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, CONV,
                padding='same',
                activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, CONV,
                padding='same',
                activation='relu'),
  layers.MaxPooling2D(),
  layers.Dropout(0.2),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes,
               activation='softmax')
])
```

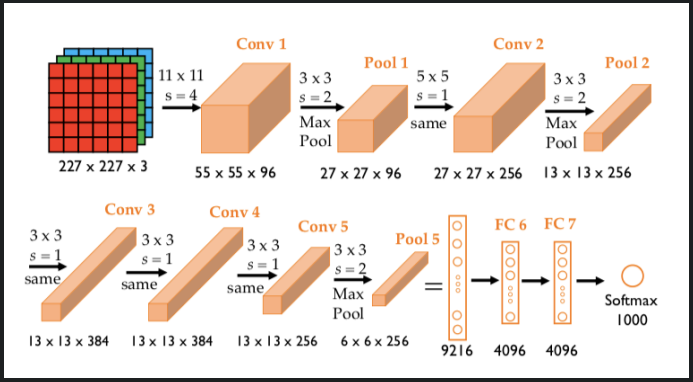

Figura 3: AlexNet. Diagrama de su arquitectura

- **Cantidad de datos**:

  Los datos que usaremos serán de un dataset de `www.kaggle.com` explicado en la sección 3.2. Con un dataset de 8000 pinturas de más de 49 artistas, un modelo convolucional implementado desde cero es manejable. Sin embargo si se tratara de un dataset con mayor cantidad de datos, lo mejor sería usar modelos ya entrenados.

  En base a el dataset de MNIST [9] que tiene 70.000 imágenes (60.000 para el entrenamiento y 10.000 para la validación) podríamos pensar que nuestro dataset es muy pequeño. Sin embargo hay que tomar en cuenta lo siguiente:

  1. Nuestro dataset usa colores RGB.
  2. Usamos data augmentation para poder enriquecer nuestro modelo.

3. Nuestros recursos computacionales son limitados.

Por lo tanto, en base a los anteriores podemos proseguir con la exposición.

4.2.2. Elección de las entradas y el tamaño de la imagen

- **Resolución de entrada (300x300)**:

  Las arquitecturas como AlexNet [7] tienen el tamaño de entreada de 227×227. Pero como se está trabajando con imágenes de pinturas debemos balancear el tamaño de entrada entre la calidad y la eficiencia. Es por ello que se escogió un tamaño de entrada de 300×300 que permite analizar un poco más de detalles sin sacrificar la eficiencia.

- **Tamaño de salida (número de artistas)**:

  El modelo finaliza con una capa `Dense(num_classes)` donde `num_classes` corresponde al número de artistas (49 en nuestro caso) (categorías) Esto permite asignar probabilidades a cada clase durante la clasificación. De hecho este no es el único cámino que existe. Se han intentando otro tipo de codificaciones, como por ejemplo la binaria en el dataset de MNIST [10]. En este último estudio revela que tanto la codificación One Hot Encoder como la binaria son igual de eficinetes, sin embargo la codificación binaria es un poco más complicada de decodificar, y por ende de entrenar. Es por ello que por simplicidad por simplicidad usaremos la codificación One Hot Encoder.

4.2.3. Configuración del entrenamiento

- **Batch size**: Este tamaño común en ejemplos de `www.kaggle.com` que balancea la estabilidad del gradiente y el uso eficiente de la memoria. Existen sin embargo investigaciones que sugieren un batch size mayor a 200 dependiendo los recursos computacionales [11]. Sin embargo, dado que nuestros recursos computacionales son limitados (no se cuenta con una GPU) reduciremos el tamaño del lote a 32 que es poco pesado para nuestra computadora, pero funciona.

- **Optimización con RMSprop**: El optimizador RMSprop ajusta dinámicamente la tasa de aprendizaje para cada parámetro en función de los gradientes recientes, lo que lo hace particularmente útil en redes neuronales profundas. Su funcionamiento se basa en los siguientes pasos:

  1. **Cálculo del gradiente**:

     $$g_t = \nabla_{\theta_t} L(\theta_t)$$

     Aquí, g_t es el gradiente del parámetro θ con respecto a la función de pérdida L en el paso t.

  2. **Promedio móvil del gradiente cuadrado**: RMSprop mantiene un promedio exponencial del cuadrado del gradiente:

     $$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2$$

     Donde:

     - γ: Tasa de decaimiento, típicamente $\gamma = 0{,}9$.
     - $E[g^2]_t$: Promedio móvil del gradiente cuadrado en el paso t.

  3. **Actualización del parámetro**: La actualización se realiza normalizando el gradiente con la raíz del promedio móvil, añadiendo un término pequeño para evitar divisiones por cero:

     $$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

     Donde:

     - η: Tasa de aprendizaje.
     - ϵ: Término de estabilidad numérica, típicamente $\epsilon = 10^{-8}$.

  Lo que hace a grandes rasgos cada una de las ecuaciones es:

  1. **Promedio móvil del gradiente cuadrado**: El término $E[g^2]_t$ acumula la información histórica del gradiente, permitiendo que RMSprop adapte la escala de las actualizaciones de los parámetros basándose en la magnitud del gradiente reciente.
  2. **Normalización adaptativa**: Al dividir por $\sqrt{E[g^2]_t + \epsilon}$, RMSprop reduce la magnitud de las actualizaciones en parámetros con gradientes grandes y las aumenta en parámetros con gradientes

pequeños. Esto permite que el modelo aprenda de manera más uniforme y evita que los gradientes grandes dominen el entrenamiento.

3. **Tasa de aprendizaje efectiva**: En cada paso, RMSprop calcula una tasa de aprendizaje efectiva específica para cada parámetro, adaptándola según el historial de gradientes.

El código que inicializa este optimizador es:

```
opti = tf.keras.optimizers.
  .RMSprop(momentum=0.1)
```

Aunque el parámetro momentum no aparece explícitamente en las ecuaciones de **RMSprop**, se incluye en la implementación de TensorFlow para mantener una interfaz unificada con otros optimizadores y para que el optimizador no se quede atrapado en mínimos locales. Existen más parámetros que se pueden especificar según la página de Keras. Además en la misma página nos dice el valor por defecto de la tasa de aprendizaje.

- **momentum**: `float`, defaults to 0.0. If not 0.0., the optimizer tracks the momentum value, with a decay rate equals to 1 - momentum.
- **learning_rate**: A `float`, a `keras.optimizers.schedules.LearningRateSchedule` instance, or a callable that takes no arguments and returns the actual value to use. The learning rate. Defaults to 0.001. [12]

- **Pérdida (SparseCategoricalCrossentropy)**: La pérdida **SparseCategoricalCrossentropy** se utiliza para problemas de clasificación multiclase donde las etiquetas de las clases son enteros (en lugar de vectores one-hot). Es especialmente útil cuando hay un número grande de clases, ya que reduce los requisitos de memoria y cómputo. Esta tiene una relación estrecha con la función de entropía cruzada estándar está definida como:

$$L_{\text{CCE}} = -\frac{1}{N}\sum_{k=1}^{N}\sum_{i=1}^{C} y_{k,i}\log(\hat{y}_{k,i})$$

Donde:

- N: Número de ejemplos en el lote.
- C: Número de clases.
- $y_{k,i}$: Valor de la etiqueta para el ejemplo k y la clase i (one-hot-encoded: 1 para la clase correcta, 0 para las demás).
- $\hat{y}_{k,i}$: Probabilidad predicha para el ejemplo k y la clase i.

En el caso de **SparseCategoricalCrossentropy**, no se utiliza un vector codificado one-hot ($y_{k,i}$), sino un entero t_k que representa el índice de la clase correcta para el ejemplo k. La fórmula se simplifica:

$$L_{\text{SparseCCE}} = -\frac{1}{N}\sum_{k=1}^{N}\log(\hat{y}_{k,t_k})$$

Donde:

- t_k: Índice entero de la clase correcta para el ejemplo k.
- $\hat{y}_{k,t_k}$: Probabilidad predicha para la clase t_k.

Así pues, ambas funciones son equivalentes si las etiquetas t_k se convierten en formato one-hot. La diferencia es puramente en términos de representación y eficiencia computacional.

$$L_{\text{CCE}} = L_{\text{SparseCCE}}$$

La distinción surge porque en **SparseCategoricalCrossentropy**, en lugar de sumar sobre todas las clases i, simplemente se accede directamente al valor de la probabilidad predicha para la clase correcta, t_k.

Esta pérdida generalmente se usa junto con una activación **softmax** en la última capa de la red neuronal. El softmax convierte las salidas z_i del modelo en probabilidades $\hat{y}_i$:

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}$$

Donde:

- z_i: Logits o puntuaciones brutas producidas por el modelo para la clase i.
- C: Número total de clases.

La combinación de softmax y entropía cruzada es eficiente, ya que se pueden calcular juntos de manera numéricamente estable mediante la opción `from_logits=True`.

En TensorFlow el código es:

```
loss = tf.keras.losses.
  .SparseCategoricalCrossentropy
  (from_logits=True)
```

- **Medida de precisión**:

  El **accuracy** o **precisión** es una métrica de evaluación utilizada en problemas de clasificación que mide la proporción de predicciones correctas respecto al total de predicciones. Matemáticamente, se define como:

$$\text{Accuracy} = \frac{\text{Número de predicciones correctas}}{\text{Número total de predicciones}}$$

  La fórmula matemática es:

  Dado un conjunto de datos con N muestras:

  1. y_{true}: etiquetas reales de las muestras.
  2. y_{pred}: etiquetas predichas por el modelo.

  La precisión se calcula como:

$$\text{Accuracy} = \frac{\sum_{i=1}^{N} \delta(y_{\text{pred}_i} = y_{\text{true}_i})}{N}$$

  donde :

$$\delta(y_{\text{pred}_i}, y_{\text{true}_i}) = \begin{cases} 1 & \text{si } y_{\text{pred}_i} = y_{\text{true}_i}, \\ 0 & \text{en caso contrario.} \end{cases}$$

  En TensorFlow, la métrica `accuracy` realiza internamente este cálculo utilizando las predicciones del modelo y las etiquetas reales. En este caso:

  1. El modelo produce un vector de probabilidades (logits) por clase, ya que se usa `SparseCategoricalCrossentropy (from_logits=True)`.
  2. `'accuracy'` convierte los logits en índices de clases predichas (aplicando `argmax` en el eje de las clases).
  3. Compara estas predicciones con las etiquetas reales para calcular la fracción de aciertos.

  El `accuracy` es una métrica adecuada cuando:

  - Las clases están balanceadas (aproximadamente el mismo número de muestras por clase).
  - La evaluación del modelo no necesita diferenciar entre tipos de errores (por ejemplo, en problemas médicos donde los falsos negativos pueden ser más graves que los falsos positivos).

- **Épocas**:

  Empezaremos con 20 épocas (ya que el ejemplo viene definido así). Si es que el modelo no convergiera, entonces se podría modificar levemente algunos de los hiperparámetros. Si es que si convergiera entonces lo que procedería es aumentar el número de épocas. En una primera ejecución se pudo evidenciar que el modelo podía distinguir algunas pinturas, sin embargo, no era tan preciso commo se esperaría.

- **Entrenamiento**

  Cuando usamos el método `model.fit` lo que TensorFlow hace es:

  1. **Forward Pass (Propagación hacia adelante)**:
     - La imagen pasa a través de las capas convolucionales (`Conv2D`), donde los kernels realizan convoluciones, detectando características como bordes o texturas.
     - Luego, las capas densas combinan estas características para hacer predicciones.
     - Finalmente, el modelo genera una salida con probabilidades para cada clase, gracias a la capa `Dense` con activación `softmax`.

     Matemáticamente:

     a) Para cada capa convolucional (`Conv2D`):

$$\mathbf{y}^{(l)} = \text{ReLU}\left(\mathbf{W}^{(l)} * \mathbf{x}^{(l-1)} + \mathbf{b}^{(l)}\right)$$

     donde:

     - $\mathbf{W}^{(l)}$ son los filtros o kernels de la capa l.
     - $*$ denota la operación de convolución.

- $\mathbf{x}^{(l-1)}$ es la salida de la capa anterior.
- $\mathbf{b}^{(l)}$ son los sesgos de la capa.
- $\text{ReLU}(z) = \text{máx}(0, z)$ es la función de activación.

b) Las capas densas (`Dense`) combinan las características extraídas:

$$\mathbf{y}^{(L)} = \text{ReLU}\left(\mathbf{W}^{(L)} \cdot \mathbf{x}^{(L-1)} + \mathbf{b}^{(L)}\right)$$

donde:

- $\cdot$ denota el producto matricial,
- $\mathbf{W}^{(L)}$ son los pesos de la capa L,
- $\mathbf{x}^{(L-1)}$ es la salida de la capa anterior.

c) La capa final usa `softmax` para generar probabilidades para cada clase:

$$\hat{y}_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

para cada $i = 1, \ldots, \text{num\_classes}$

donde:

- z_i es la entrada a la clase i,
- $\hat{y}_i$ es la probabilidad predicha para la clase i.

2. **Cálculo del error**:
   El modelo compara las predicciones con las etiquetas reales usando la función de pérdida (`SparseCategoricalCrossentropy` en este caso). Esto genera un valor de pérdida que indica qué tan incorrecta fue la predicción.
3. **Backward Pass (Propagación hacia atrás)**:
   - Se usa el algoritmo de **backpropagation** para calcular el gradiente de la pérdida con respecto a todos los parámetros entrenables (pesos de las capas densas y kernels de las convoluciones).
   - El optimizador (`RMSprop`) actualiza los valores de los parámetros para minimizar la pérdida.

   a) Calcula los gradientes para cada parámetro.

   b) Actualiza los valores de los parámetros en base al gradiente y su hiperparámetro de aprendizaje.

   Matemáticamente:

   a) **Gradientes**: Se calculan los gradientes de la pérdida con respecto a los parámetros entrenables ($\mathbf{W}$ y $\mathbf{b}$):
   - Para los pesos de la capa l:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}} = \delta^{(l)} \cdot (\mathbf{x}^{(l-1)})^\top$$

   - Para los sesgos de la capa l:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(l)}} = \delta^{(l)}$$

   - Donde $\delta^{(l)}$ es el gradiente del error propagado hacia atrás en la capa l, calculado como:

$$\delta^{(l)} = \frac{\partial \mathcal{L}}{\partial \mathbf{y}^{(l)}} \cdot f'(\mathbf{z}^{(l)})$$

   b) **Actualización de parámetros**: Los gradientes se utilizan para actualizar los parámetros con un optimizador (`RMSprop` en este caso). Usando la regla de descenso de gradiente:

$$\mathbf{W}^{(l)} \leftarrow \mathbf{W}^{(l)} - \eta \cdot \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}}$$

$$\mathbf{b}^{(l)} \leftarrow \mathbf{b}^{(l)} - \eta \cdot \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(l)}}$$

   donde η es la tasa de aprendizaje.

Los parámetros que se actualizan son:

1. Kernels en las capas convolucionales (`Conv2D`) $\mathbf{W}^{(l)}$ y $\mathbf{b}^{(l)}$
2. Pesos y sesgos en las capas densas (`Dense`) $\mathbf{W}^{(L)}$ y $\mathbf{b}^{(L)}$

Así hay algunas cosas que cabe señalar como:

1. TensorFlow permite que nosotros hagamos modificaciones a el método `fit()` creando una clase que herede de nuestro modelo y sobreescribiendo el método `train_step()` [13]
2. Del mismo modo TensorFlow permite poder hacer nuestros propios layers totalmente personalizados, inclusive con pesos no entrenables [**?**]

5. Resultados

Se pudo comprobar que con pequeñas modificaciones es posible generar pinturas

6. Discusión

Es muy difícil crear un modelo nuevo, en especial cuando no se tiene conocimiento suficiente sobre el mundo teórico de la inteligencia artificial. Es por ello que adapatamos un código que ya estaba en internet.

7. Conclusiones

El mundo de la inteligencia artificial está avanzado demasiado rápido. Muchas personas están asustadas y tiene motivos de estarlo.

Referencias

[1] Geoffrey I Webb, Eamonn Keogh, and Risto Miikkulainen. Naïve bayes. *Encyclopedia of machine learning*, 15(1):713–714, 2010.

[2] Mohammed Kayed, Ahmed Anter, and Hadeer Mohamed. Classification of garments from fashion mnist dataset using cnn lenet-5 architecture. In *2020 international conference on innovative trends in communication and computer engineering (ITCE)*, pages 238–243. IEEE, 2020.

[3] M Swapna, Yogesh Kumar Sharma, and BMG Prasadh. Cnn architectures: Alex net, le net, vgg, google net, res net. *Int. J. Recent Technol. Eng*, 8(6):953–960, 2020.

[4] Mkkoehler. Cnn to identify paintings, December 2020. URL `https://www.kaggle.com/code/mkkoehler/cnn-to-identify-paintings/notebook`.

[5] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 33(12):6999–7019, 2021.

[6] Best artworks of all time, March 2019. URL `https://www.kaggle.com/datasets/ikarus777/best-artworks-of-all-time`.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[8] Md Zahangir Alom, Tarek M Taha, Christopher Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C Van Esesn, Abdul A S Awwal, and Vijayan K Asari. The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*, 2018.

[9] Corinna Cortes Yann LeCun and Chris Burges. Handwritten digit database. URL `https://yann.lecun.com/exdb/mnist/`.

[10] Jinxin Wei and Zhe Hou. Binary encoding for label. *Authorea Preprints*, 2023.

[11] Pavlo M Radiuk. Impact of training set batch size on the performance of convolutional neural networks for diverse datasets. 2017.

[12] Keras Team. Keras documentation: Rmsprop. URL `https://keras.io/api/optimizers/rmsprop/`.

[13] URL `https://www.tensorflow.org/guide/keras/customizing_what_happens_in_fit`.