Eduardo García Castrejón

Reporte Proyecto Final

Problema: Clasificación de imágenes de pinturas de los mejores artistas de la historia

Planteamiento:

Se tiene el dataset de una colección de pinturas de 50 artistas a lo largo de la historia, obtenida en kaggle: https://www.kaggle.com/datasets/ikarus777/best-artworks-of-all-time

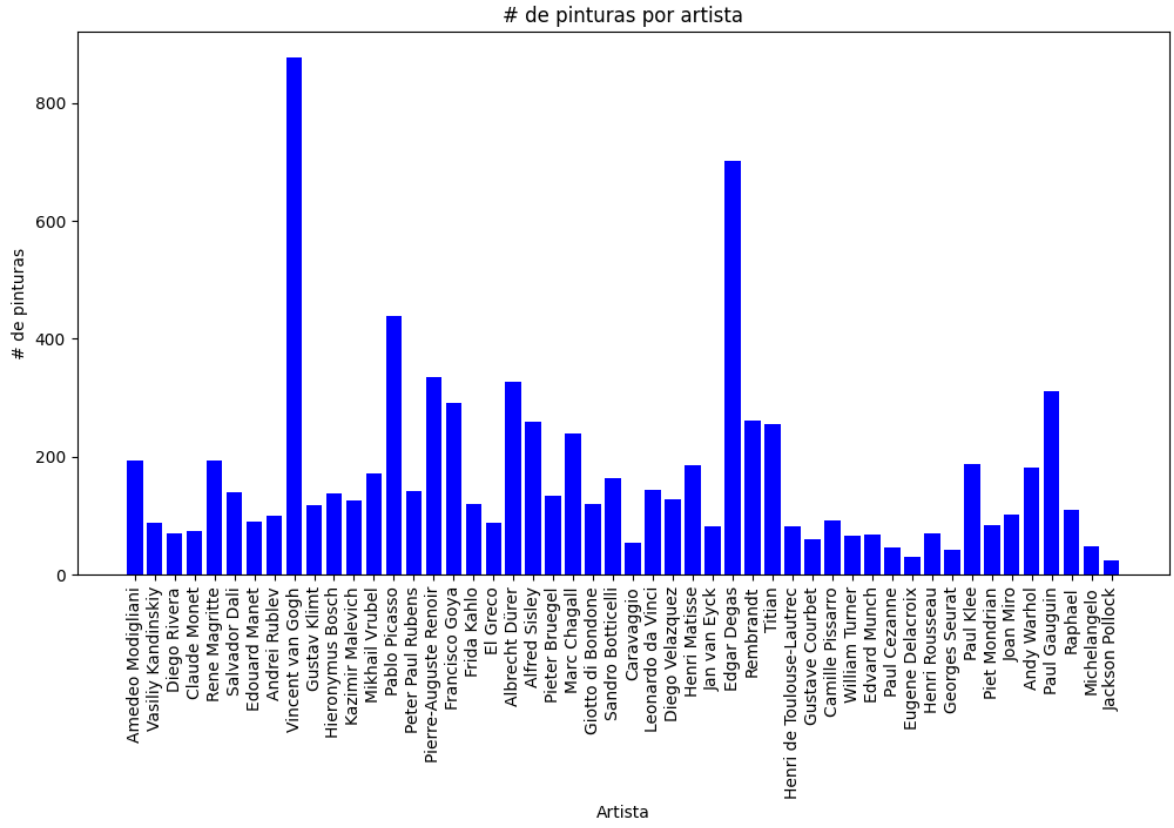
El contenido de este dataset es el siguiente:

artists.csv Información de cada artista incluyendo el número de pinturas dentro del dataset

images.zip Colección de imágenes en tamaño completo, divida en folders de cada artista.

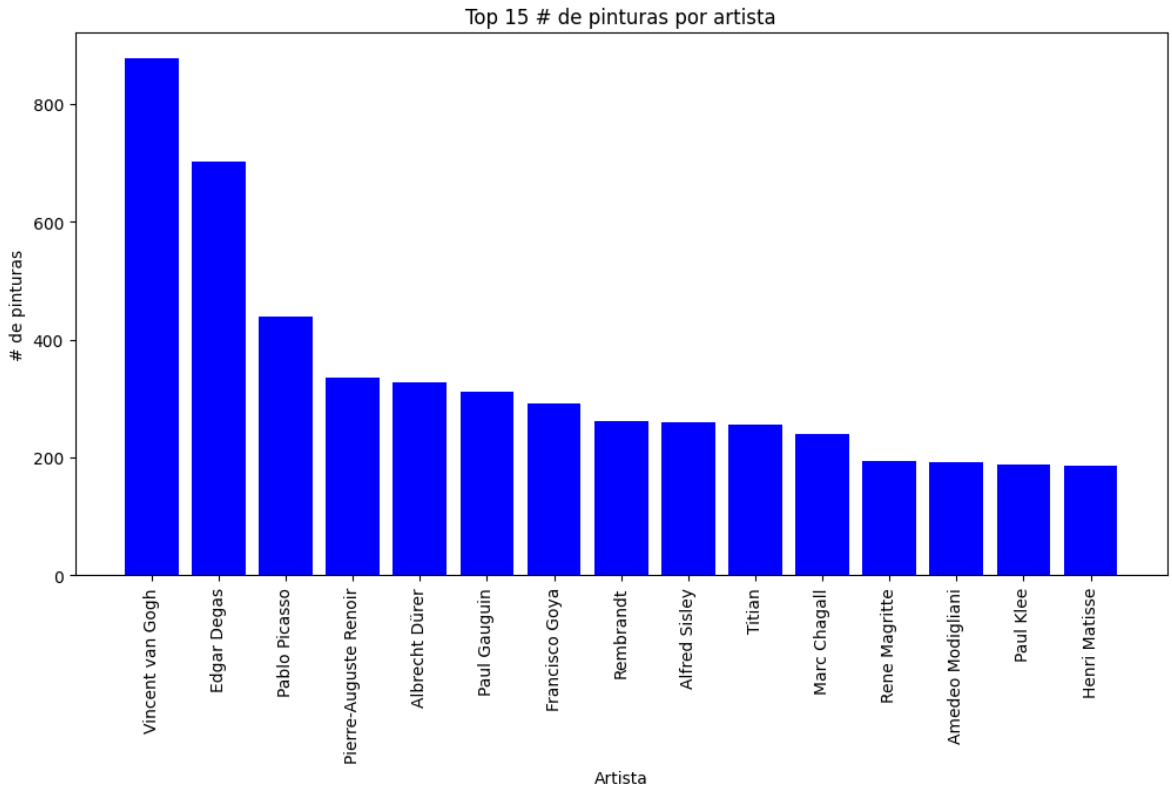
El plan es diseñar un modelo en Python de Deep Learning que sea capaz de reconocer el artista al analizar características esenciales en sus pinturas, tales como patrones geométricos o la paleta de colores.

En la siguiente figura se muestra un histograma que señala la frecuencia de cada clase en el dataset.



Los principales desafíos de este problema son la presencia de un desbalance y un número grande de clases. Además, es posible que las pinturas de cada artista no sean suficientemente identificables debido a la falta de unicidad en el estilo del artista o un estilo inconsistente a lo largo de todas las obras de algún artista.

Así que, se ha de limitar el alcance de este proyecto, acortando el número de clases a 15 artistas, específicamente las 15 clases con más datos. En el siguiente histograma se muestra los 15 pintores elegidos:



Objetivos:

- General: crear un modelo CNN que sea capaz de identificar las pinturas de 15 artistas.
- Específico: Evaluar el desempeño del modelo.

Metodología:

La metodología se basa en técnicas comunes de DeepLearning, centrado en las redes convolucionales, utilizando herramientas de Tensorflow y Keras en programación Python.

El uso de Redes Neuronales Convolucionales se justifica por su capacidad de extraer características esenciales de una imagen con el fin de realizar una clasificación categórica de manera eficiente.

- **Capas Convolucionales y de Pooling**
- **Capas Completamente Conectadas**
- **Función de Activación**: ReLU
- **Capa de Salida**: Softmax
- **Función de Pérdida:** Entropía cruzada categórica, adecuada para problemas de clasificación multicategórica.

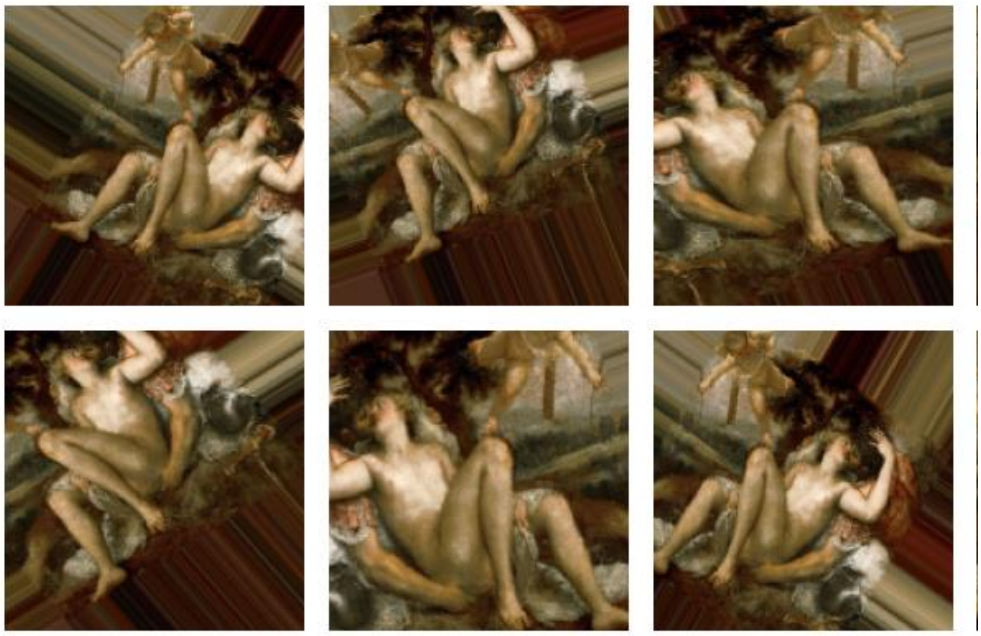
- **Optimizador**: RMSdrop

El proceso empleado se propuso en los siguientes pasos:

* **Organización de folders:** Organizar las carpetas de las imágenes, con tal de tener folders dedicados al entrenamiento, validación y prueba. Se escribió una función en Python donde dividía de manera aleatoria un folder en dos carpetas en una razón dada. De esta manera la distribución se pensó como:

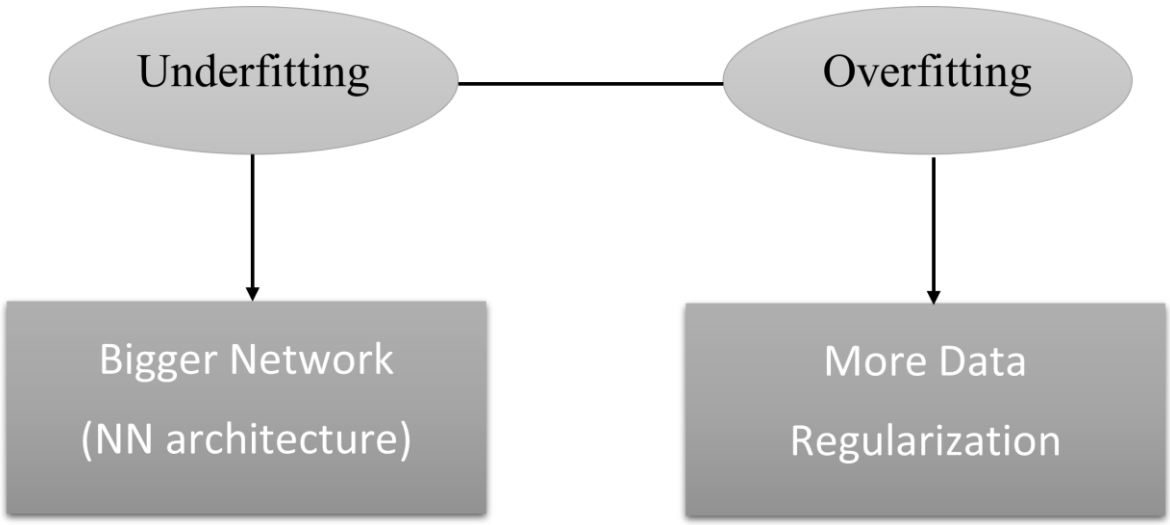
- **test: 15%**
- **train/val: 85%**
    - train: 75%
    - validation: 25%

* **Preprocesamiento:** Con keras.ImageDataGenerator se decodificaba el contenido JPEG a cuadrículas de pixeles RGB y son convertidos a vectores de punto flotante normalizados de las imágenes con una dimensión (256, 256, 3). Para el generador en entrenamiento, se especificó un aumento de datos como rotaciones, traslaciones, escalado y espejado (un ejemplo en la sig figura) para aumentar la variabilidad y robustez del modelo.



* **Diseño de arquitectura del modelo CNN:** Con Keras.models se diseña la arquitectura del modelo. Ahora bien, es importante dar con una arquitectura de tal forma que el modelo no presente casos donde las curvas de precisión y pérdida en los datos de entrenamiento difieren en los datos de validación. Si la curva de pérdida y la curva de precisión en validación no baja y aumenta respectivamente junto con las curvas en entrenamiento, es un caso de Overfitting; el modelo se ajustó demasiado a los datos de entramiento. El caso contrario es donde ocurre underfitting, el modelo no está aprendiendo lo suficiente.

El método a seguir para dar con una arquitectura óptima, se explica en la siguiente figura:



Al entrenar un modelo propuesto y si se observa un caso de overfitting, entonces lo que se tiene que hacer es implementar regularización al modelo o añadir más datos. En este problema, para aumentar datos solo se puede incrementar los parámetros de Data Augmentation en preprocesamiento, pero no es recomendable porque puede alterar características importantes del artista. Entonces para combatir el overfitting, se implementaron las siguientes herramientas de regularización, experimentando con distintos parámetros.

* Dropout (Mandar a valor cero los pesos de un número de neuronas en cada capa densa completamente conectada)

* lrate decay (Implementar un decaimiento en la razón de aprendizaje a medida que aumentan las épocas)

* BatchNormalization (Normalizar la salida de activación después de cada capa de convolución)

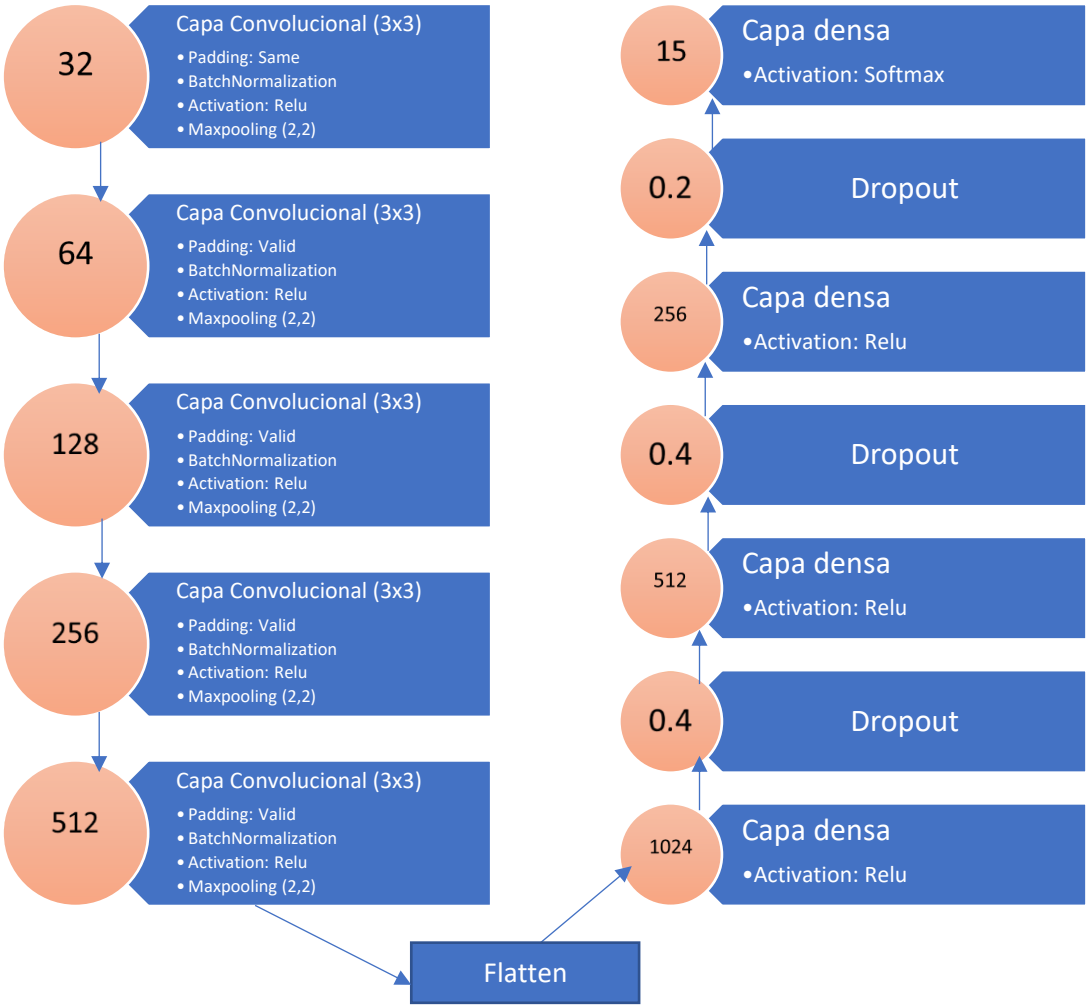
* L2 (se agrega un término de penalización a la función de pérdida)

En la presencia de underfitting, lo que se hace es aumentar la complejidad del modelo, y eso es, incrementar el número de capas convolucionales, así como el número de capas densas y neuronas.

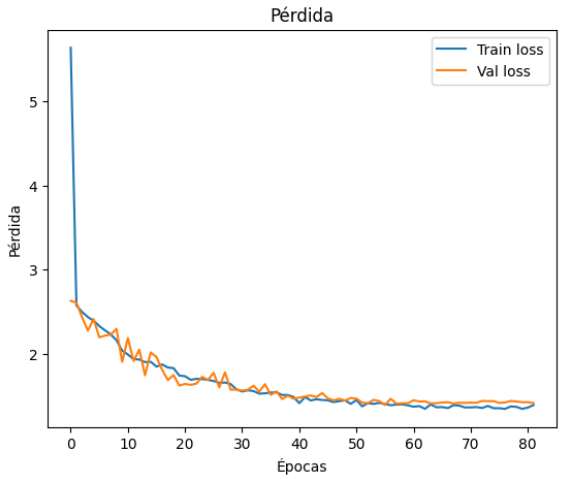
* ****Evaluación:**** Evaluar el modelo con Acuraccy, top-2 Acuraccy, las curvas de entrenamiento y con matrices de confusión.

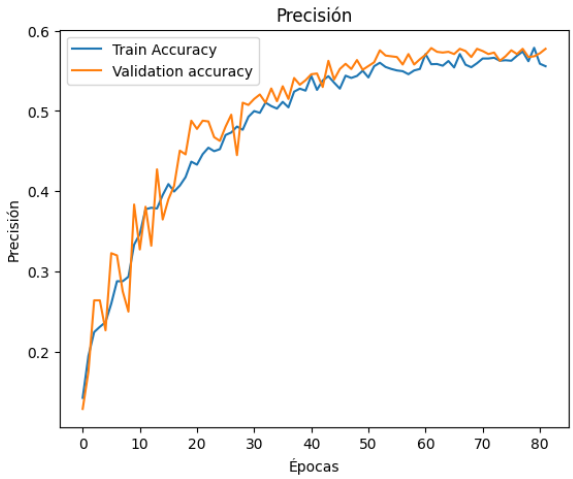
Resultados:

El mejor modelo tiene la siguiente Arquitectura:



A continuación, se presentan las curvas de aprendizaje, como se puede observar, el modelo se ajusta adecuadamente en los datos de entrenamiento y validación.

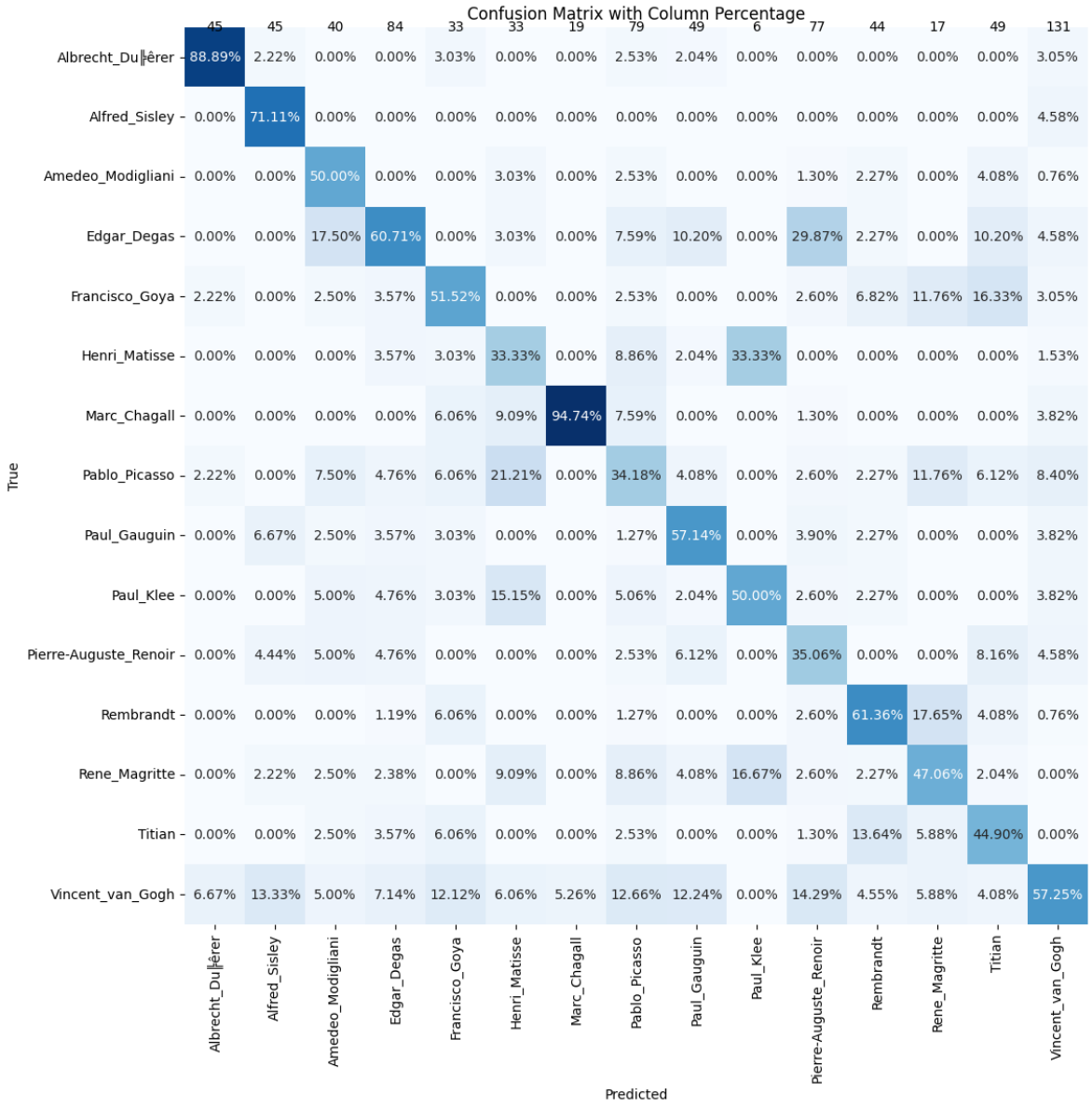




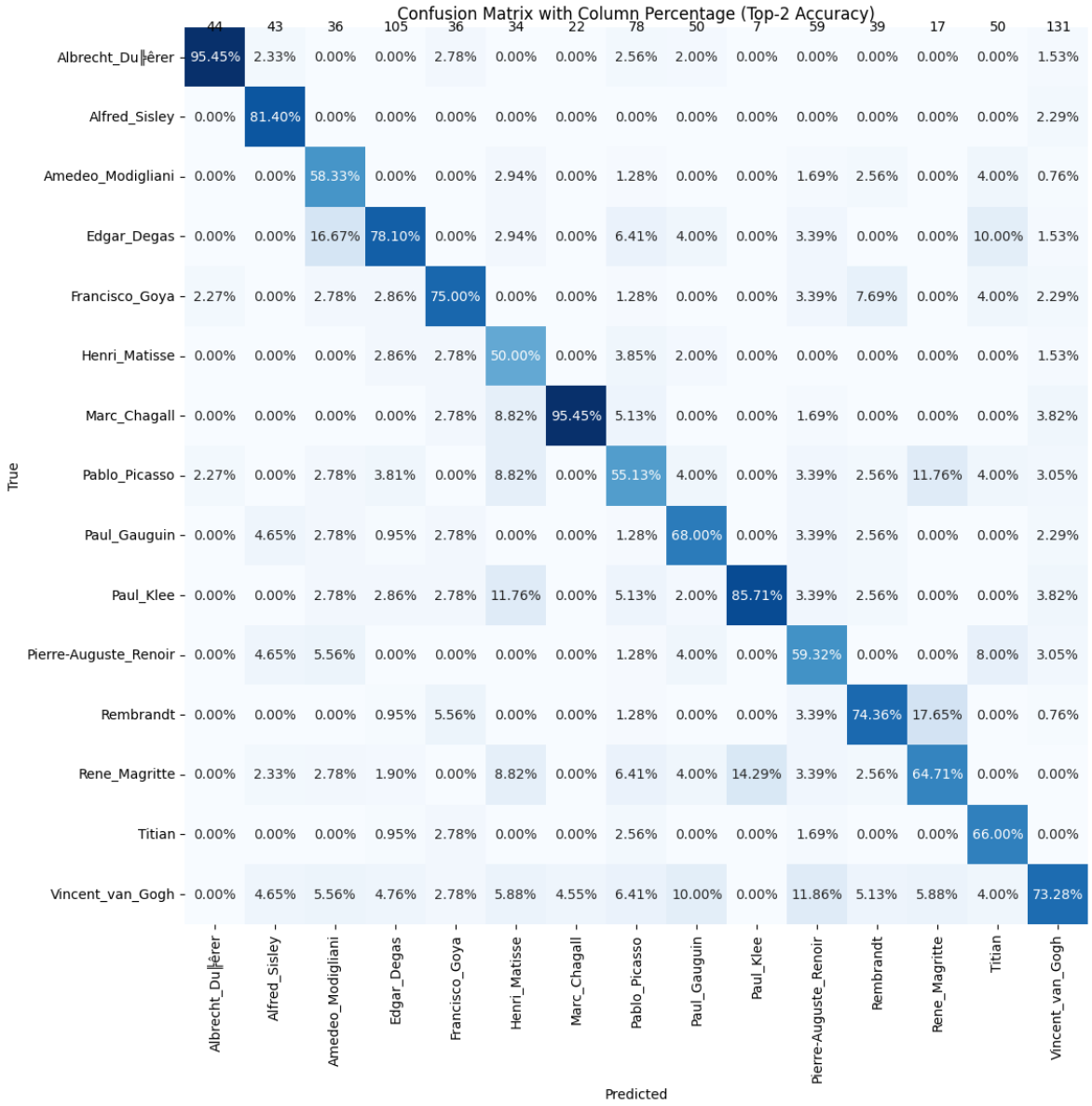
Se tuvieron los siguientes resultados:

- Test accuracy: 0.55
- Test top-3 accuracy: 0.79
- Test top-2 accuracy: 0.71

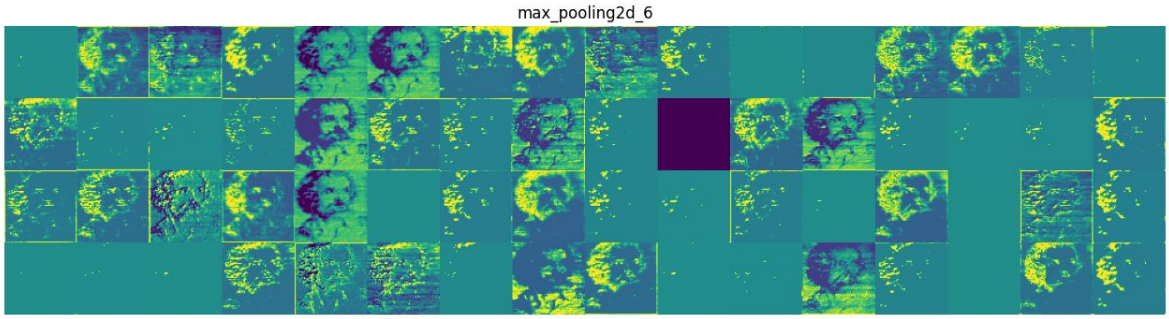
Pero al observar la matriz de confusión se puede notar un mejor desempeño para ciertos artistas que para otros, esto debido a que sus pinturas son más consistentes en un estilo único.



En la matriz de confusión al considerar las dos predicciones más probables, tenemos un desempeño notablemente mayor.



Como demostración del proceso de las capas de la convolución, tenemos una imagen perteneciente a una pintura de Albrecht Dullherer y la representación visual del resultado de la sexta capa de convolución:



Conclusiones:

- El dataset es muy susceptible al overfitting debido al bajo tamaño de este y por desbalance de clases
- El dataset es difícil de predecir debido a la baja exposición de datos al modelo CNN. También se puede argumentar que las características inherentes de cada pintura no son suficientemente únicas y son similares entre cada artista.
- Top-2 test accuracy para cada artista varía entre 50% y 95%. Esta diferencia se debe a características únicas y consistencia entre las pinturas de cada artista.
- Se necesitarán de otras herramientas más sofisticadas para mejorar el modelo que resultó ser muy complejo.