

Trabajo Práctico Final - Clasificador de Doodles

Manuela Gomez Pazos y Martina Grünewald

Profesores: Gastón Castro y Esteban Uriza

Abstract—.

En este trabajo se abordó el problema de la clasificación automática de dibujos realizados a mano alzada, utilizando un subconjunto del dataset *Quick, Draw!* de Google. Se desarrolló e implementó una arquitectura de red neuronal convolucional (CNN) adaptada a imágenes de baja resolución en escala de grises, caracterizadas por su alto nivel de abstracción visual. El modelo alcanzó un rendimiento robusto, superando el 98% de precisión promedio, incluso frente a dibujos con variaciones significativas de trazo y estilo. Para mejorar su generalización, se incorporaron técnicas de *data augmentation*, y se realizaron distintos análisis de interpretabilidad mediante Grad-CAM, mapas de calor promedio por clase y reducciones de dimensionalidad (t-SNE y UMAP), que permitieron examinar cómo la red estructura su espacio latente. Finalmente, se desarrolló una aplicación interactiva que permite a cualquier usuario realizar un dibujo y obtener una predicción en tiempo real, evidenciando la solidez del modelo frente a entradas completamente nuevas. Los resultados obtenidos reflejan el potencial de las CNN para comprender dibujos realizados de manera libre y esquemática, aportando evidencia valiosa sobre sus capacidades en el dominio de la visión artificial.

1. Introducción

El reconocimiento automático de dibujos a mano —o *doodles*— representa un desafío particular dentro del campo de la visión artificial, ya que estos dibujos suelen carecer de color, textura y detalles de fondo, mostrando únicamente una abstracción de la forma. Recientemente, Google publicó el dataset *QuickDraw*, que contiene millones de *doodles* de centenas de categorías, ofreciendo una oportunidad única para explorar modelos de clasificación en un conjunto de dibujos con gran variabilidad de trazos y estilos.

En este trabajo proponemos un sistema de clasificación basado en una red neuronal convolucional sencilla (SimpleCNN) entrenada sobre un subconjunto de 12 clases del dataset *QuickDraw*. Para mejorar su robustez frente a variaciones en el dibujo, incorporamos técnicas de *data augmentation* (rotaciones, escalados y traslaciones). Además, evaluamos su capacidad para generalizar a *doodles* generados en tiempo real desde un canvas de dibujo interactivo, demostrando que el modelo puede predecir satisfactoriamente nuevos dibujos fuera del conjunto de test original.

A lo largo de este informe se presenta el marco teórico que fundamenta nuestro enfoque, se detallan los pasos de preprocessamiento, diseño de la arquitectura y entrenamiento del modelo, y se muestran los resultados cuantitativos y cualitativos obtenidos, así como técnicas de interpretabilidad (Grad-CAM) y análisis de características (t-SNE/UMAP) para comprender las representaciones internas de la red.

2. Marco Teórico

2.1. Clasificación de dibujos vs. imágenes naturales

En visión por computadora, la mayoría de los avances se han centrado en el procesamiento de imágenes naturales, es decir, aquellas capturadas mediante cámaras digitales que contienen información rica en color, textura, iluminación y contexto. Sin embargo, el reconocimiento de dibujos a mano alzada (sketches) plantea desafíos particulares: estas representaciones son inherentemente abstractas, monocromáticas y carecen de fondo o textura. Como resultado, los modelos deben aprender a reconocer conceptos visuales a partir de representaciones extremadamente simplificadas.

Esta diferencia implica que técnicas efectivas en imágenes naturales pueden fallar al enfrentarse a dibujos, donde la variabilidad de estilos individuales genera alta dispersión intraclasa y baja separabilidad interclase.

2.2. Aprendizaje profundo vs. métodos tradicionales

Los métodos tradicionales de clasificación de imágenes consistían en dos etapas separadas: extracción manual de características y clasificación. Por ejemplo, se utilizaban descriptoros como HOG (Histogram of Oriented Gradients) o SIFT (Scale-Invariant Feature Transform), seguidos por clasificadores como SVM o k-NN.

Con el aprendizaje profundo, este pipeline fue reemplazado por modelos end-to-end, como las redes neuronales convolucionales (CNN), que aprenden representaciones directamente desde los datos, optimizando conjuntamente extracción de características y clasificación.

2.3. Redes Neuronales Convolucionales (CNN)

Las CNN están diseñadas para procesar datos con estructura espacial, como imágenes. Una CNN típica está compuesta por capas convolucionales, funciones de activación, capas de *pooling* y capas totalmente conectadas (*fully connected*).

Convolución

La operación de convolución se define como:

$$S(i, j) = (X * K)(i, j) = \sum_m \sum_n X(i + m, j + n) \cdot K(m, n) \quad (1)$$

donde X es la imagen de entrada y K es el kernel o filtro. Esta operación permite detectar patrones locales (bordes, esquinas, formas) mediante pesos compartidos.

Pooling

Las capas de *pooling* reducen la dimensión espacial de las representaciones intermedias, conservando información relevante. El *Max Pooling* se define como:

$$Y(i, j) = \max_{(m,n) \in \text{ventana}} X(i + m, j + n) \quad (2)$$

Capas totalmente conectadas

Las capas totalmente conectadas (*fully connected*) funcionan como clasificador, utilizando la representación extraída por las capas anteriores.

2.4. Regularización y arquitectura

Para evitar el sobreajuste (*overfitting*), se emplean técnicas como:

- **Dropout:** apaga aleatoriamente un porcentaje de neuronas durante el entrenamiento.
- **Early Stopping:** detiene automáticamente el entrenamiento cuando no se observa mejora en la validación durante varias épocas consecutivas.
- **Batch Normalization:** normaliza la salida de cada capa, estabilizando y acelerando el entrenamiento.
- **Data Augmentation:** genera variaciones de los datos originales mediante transformaciones geométricas.

2.5. Interpretabilidad: Grad-CAM

Grad-CAM (*Gradient-weighted Class Activation Mapping*) permite visualizar qué regiones de una imagen contribuyeron más a la predicción de una clase específica. Se calcula ponderando los mapas de activación con los gradientes de la clase predicha:

$$\alpha_k = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k} \quad (3)$$

donde α_k es el peso del mapa de activación A^k , y^c es la salida de la red para la clase c , y Z es el número de posiciones espaciales.

Luego, el mapa Grad-CAM se obtiene como:

$$L_{\text{Grad-CAM}}^c = \text{ReLU} \left(\sum_k \alpha_k A^k \right) \quad (4)$$

2.6. Visualización de representaciones internas: t-SNE y UMAP

Para analizar cómo la red organiza internamente los ejemplos, se utilizan técnicas de reducción de dimensionalidad como t-SNE y UMAP. Estas proyectan vectores de características de alta dimensión (por ejemplo, salidas de capas internas) a un espacio bidimensional interpretable.

t-SNE

t-SNE (*t-distributed Stochastic Neighbor Embedding*) preserva la estructura local del espacio original, agrupando ejemplos similares. Su desventaja es que puede ser sensible a los hiperparámetros y no escala bien con muchos datos.

UMAP

UMAP (*Uniform Manifold Approximation and Projection*) también preserva relaciones locales pero incorpora información global. Tiende a formar clústeres más compactos y es computacionalmente más eficiente que t-SNE, lo que lo hace más adecuado para conjuntos de datos grandes.

3. Método

El desarrollo del trabajo se llevó a cabo en distintas etapas iterativas, comenzando por modelos sencillos y evolucionando hacia una arquitectura más compleja, con análisis tanto cualitativos como cuantitativos. El enfoque se centró en resolver la tarea de clasificación multiclas de dibujos pertenecientes al conjunto de datos *QuickDraw*, abordando los desafíos asociados a la variabilidad de estilo y simplicidad estructural de los *doodles*.

3.1. Primera aproximación: red convolucional básica

Como primer paso, se implementó una red neuronal convolucional de arquitectura mínima. Si bien este modelo mostró un rendimiento razonable sobre un subconjunto reducido de clases, se observó una clara fragilidad frente a variaciones en los trazos, especialmente cuando el grosor de línea difería del presente en los datos de entrenamiento. Esto evidenció la necesidad de una arquitectura más profunda, capaz de capturar patrones espaciales jerárquicos y representar la variabilidad estructural de los dibujos.

3.2. Arquitectura final: SimpleCNN

Con el objetivo de mejorar la capacidad de generalización y robustez del modelo, se desarrolló una arquitectura más profunda, denominada *SimpleCNN*, compuesta por tres bloques convolucionales con estructura repetitiva. Cada bloque contiene dos capas convolucionales con kernel de 3×3 y padding de un píxel (para mantener el tamaño espacial), seguidas de funciones de activación ReLU y normalización por lotes (*BatchNorm*). Al final de cada bloque se incorpora una capa de *MaxPooling* de tamaño 2×2 para reducir la dimensión espacial, y una capa de *dropout* para regularización.

En total, la sección convolucional realiza seis convoluciones y tres reducciones espaciales. A continuación, el modelo aplana (*flatten*) la salida tridimensional en un vector de características cuya dimensión se calcula dinámicamente durante la construcción del modelo. Este vector se procesa mediante un *head* totalmente conectado compuesto por tres capas lineales: una capa de 256 unidades, otra de 128, y finalmente una capa de salida con tantas neuronas como clases (12 en este caso), aplicando activaciones ReLU y *dropout* intermedio para evitar sobreajuste.

3.3. Aumento de datos y entrenamiento

Dado que los dibujos en *QuickDraw* presentan una gran variabilidad de estilo y forma, se incorporó un esquema de *data augmentation* para mejorar la capacidad de generalización del modelo. Las imágenes de entrenamiento fueron transformadas dinámicamente mediante una serie de operaciones aleatorias que simulan perturbaciones naturales al dibujar:

- Rotación aleatoria ($\pm 10^\circ$)
- Traslación en hasta un 10% del ancho y alto
- Inversión horizontal
- Reescalamiento y conversión a tensor

Estas transformaciones fueron aplicadas usando la librería `torchvision.transforms`, permitiendo expandir la diversidad del conjunto de entrenamiento sin necesidad de nuevos datos.

Se construyó un Dataset personalizado y se utilizaron DataLoaders para cargar los datos en lotes de tamaño 64. El entrenamiento se llevó a cabo durante un máximo de 10 épocas, utilizando la función de pérdida de entropía cruzada (*CrossEntropyLoss*) y el optimizador Adam. Se implementó una estrategia de *early stopping* con paciencia de 3 épocas sin mejora en la pérdida de validación, y se monitorearon métricas de pérdida y exactitud durante todo el proceso. Al finalizar, se guardó automáticamente el modelo con mejor desempeño.

3.4. Interpretabilidad y análisis de representaciones

Al alcanzar un desempeño satisfactorio en términos cuantitativos, se abordó la interpretación de las decisiones del modelo mediante dos enfoques complementarios: visualización de patrones característicos por clase y generación de mapas de activación específicos por imagen.

En primer lugar, se calculó un promedio de activación por clase a partir de las imágenes de entrenamiento. Para cada clase, se computó la media de los dibujos en formato de matriz de píxeles, generando un “prototipo promedio” que visualiza los trazos más frecuentes.

En segundo lugar, se utilizó *Grad-CAM* (Gradient-weighted Class Activation Mapping) para generar mapas de atención sobre ejemplos individuales. Esta técnica se aplicó sobre la séptima capa convolucional del modelo, y consiste en ponderar cada canal de activación con el gradiente de la clase predicha. Los resultados se superpusieron sobre las imágenes originales utilizando una paleta de colores e incluyendo una barra de intensidad, permitiendo visualizar si el modelo estaba enfocando correctamente las regiones relevantes del dibujo.

3.5. Visualización del espacio latente

Para explorar la estructura del espacio de representaciones internas de la red, se utilizaron técnicas de reducción de dimensionalidad: t-SNE y UMAP. Se extrajeron los vectores de características de la última capa convolucional del modelo, y se proyectaron a dos dimensiones para su visualización.

El análisis incluyó muestras balanceadas por clase, se evaluó también la influencia de la cantidad de ejemplos seleccionados por clase en la claridad de los clusters, observando que UMAP escalaba mejor con mayor volumen de datos, mientras que t-SNE ofrecía mayor fiabilidad local.

3.6. Clasificación en tiempo real: interfaz de dibujo en vivo

Finalmente, se desarrolló una aplicación interactiva que permite a los usuarios dibujar en tiempo real utilizando un canvas. La interfaz fue implementada con *tkinter*, y está conectada directamente al modelo entrenado. Una vez que el usuario traza un *doodle*, la imagen es binarizada, centrada, redimensionada a 32×32 , convertida a tensor y normalizada. Luego, es pasada por el modelo y se muestra la predicción.

La aplicación incluye animaciones visuales asociadas a cada clase (íconos flotantes) y un mensaje textual con el nombre del objeto detectado. Esta instancia permitió validar el comportamiento del

modelo frente a ejemplos completamente nuevos, confirmando su capacidad para generalizar a trazos, estilos y proporciones no vistas durante el entrenamiento.

4. Resultados

En esta sección se presentan los resultados obtenidos por el modelo SimpleCNN con Data Augmentation en distintas instancias de evaluación, tanto cuantitativa como cualitativa. Se reportan métricas de clasificación, análisis de entrenamiento, interpretabilidad y experimentación en condiciones reales.

4.1. Desempeño en test

El modelo alcanzó una *accuracy* de **98%** en el conjunto de test. La Tabla 1 muestra las métricas de precisión, recall y F1-score por clase.

Tabla 1. Reporte de clasificación en el conjunto de test

Clase	Precisión	Recall	F1-score	Soporte
apple	0.99	0.98	0.98	21708
butterfly	0.98	0.98	0.98	17700
cloud	0.98	0.97	0.97	18040
eye	0.97	0.98	0.98	18883
flower	0.97	0.96	0.97	21723
ice cream	0.97	0.97	0.97	18470
moon	0.95	0.96	0.95	18249
rainbow	0.99	0.99	0.99	19027
smiley face	0.98	0.98	0.98	18658
star	0.97	0.99	0.98	20643
sun	0.99	0.98	0.98	20067
tree	0.96	0.98	0.97	21708
Promedio (macro)	0.98	0.98	0.98	–
Promedio (ponderado)	0.98	0.98	0.98	234876

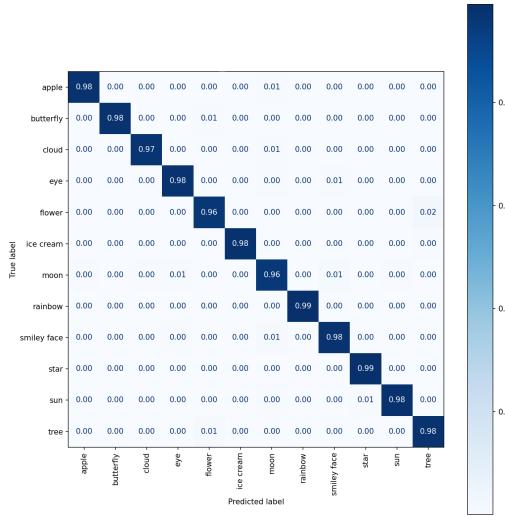


Figure 1. Matriz de confusión normalizada

Se puede observar un modelo expresivo, capaz de aprender representaciones significativas y separables incluso ante dibujos variados dentro de una misma categoría.

4.2. Curvas de entrenamiento

Se observó una convergencia estable del modelo. Las curvas de pérdida y *accuracy* se presentan en las Figuras 2 y 3.

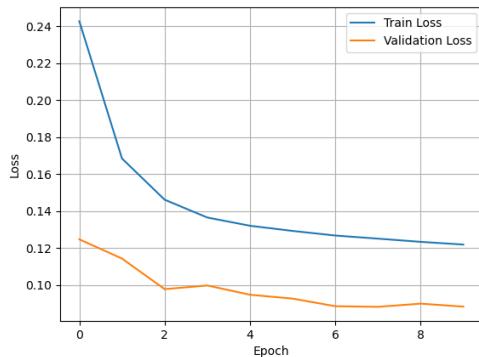


Figure 2. Pérdida de entrenamiento y validación por época

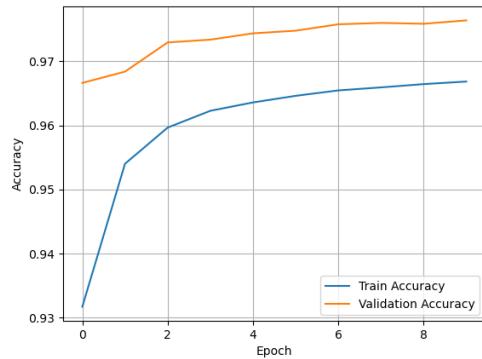


Figure 3. Exactitud de entrenamiento y validación por época

4.3. Interpretabilidad del modelo

Para explorar el funcionamiento interno de la red y comprender qué patrones influyen en sus decisiones, se emplearon técnicas de interpretabilidad visual. En particular, se utilizaron mapas de Grad-CAM, los cuales permiten visualizar las regiones de una imagen que más contribuyen a la predicción de una clase específica.

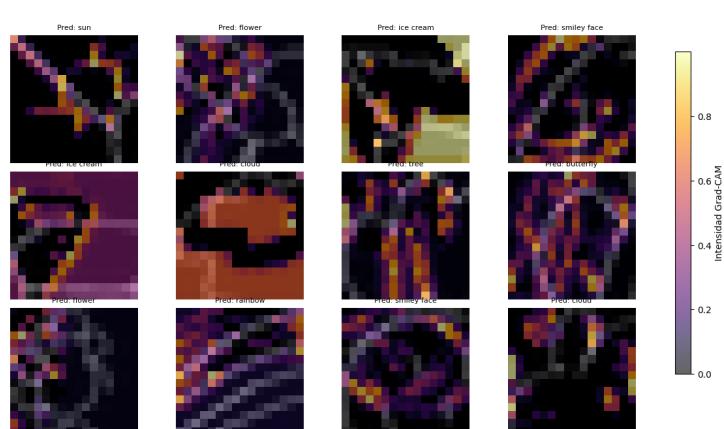


Figure 4. Ejemplos de Grad-CAM sobre imágenes del conjunto de test.

En los ejemplos de la Figura 4, puede observarse cómo las zonas resaltadas en tonos cálidos (rojos y amarillos) corresponden a regiones que tuvieron una mayor activación en la última capa convolucional. Esto sugiere que esas áreas contienen trazos visualmente relevantes que la red considera informativos para identificar el objeto dibujado.

Adicionalmente, se construyeron promedios de imágenes por clase, generando mapas de calor que evidencian los trazos más comunes dentro de cada categoría.

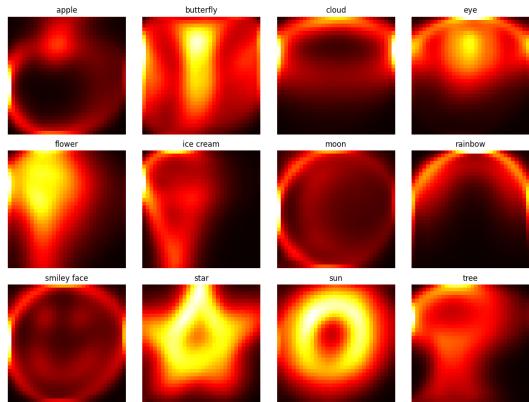


Figure 5. Mapas promedio por clase calculados sobre el conjunto de entrenamiento. Las zonas más brillantes corresponden a trazos que aparecen consistentemente en múltiples ejemplos de la misma clase.

En la Figura 5 se observan patrones de trazo recurrentes entre clases, lo cual sugiere ciertas similitudes estructurales que podrían contribuir a confusiones en la etapa de clasificación.

Ambas técnicas ofrecieron evidencia cualitativa de que la red aprendió representaciones semánticamente coherentes, focalizando su atención en regiones clave y capturando patrones estructurales característicos de cada tipo de doodle.

4.4. Visualización del espacio latente

Se emplearon técnicas de reducción de dimensionalidad para visualizar cómo el modelo organiza internamente los ejemplos.

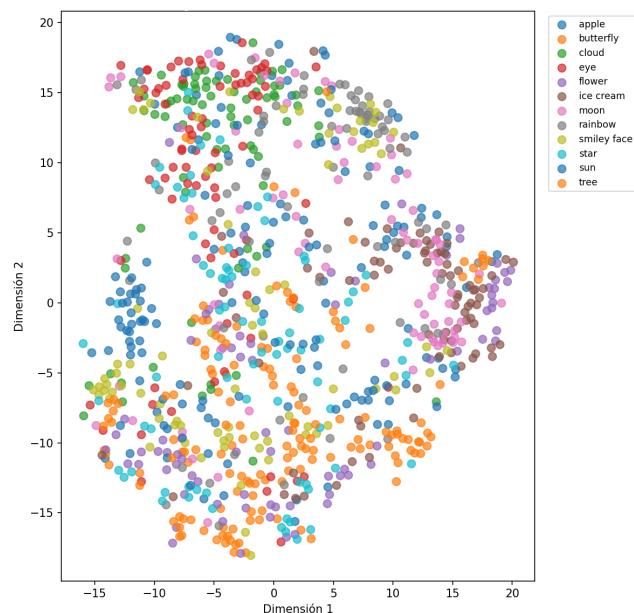


Figure 6. Visualización t-SNE de las activaciones internas

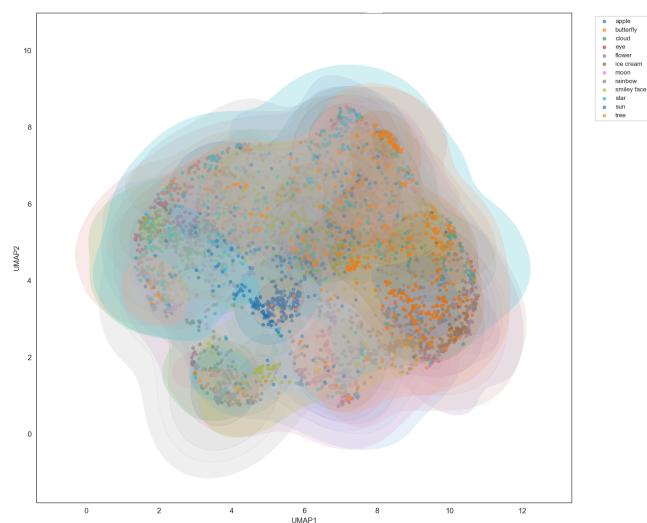


Figure 7. Visualización UMAP de las activaciones internas con sus contornos

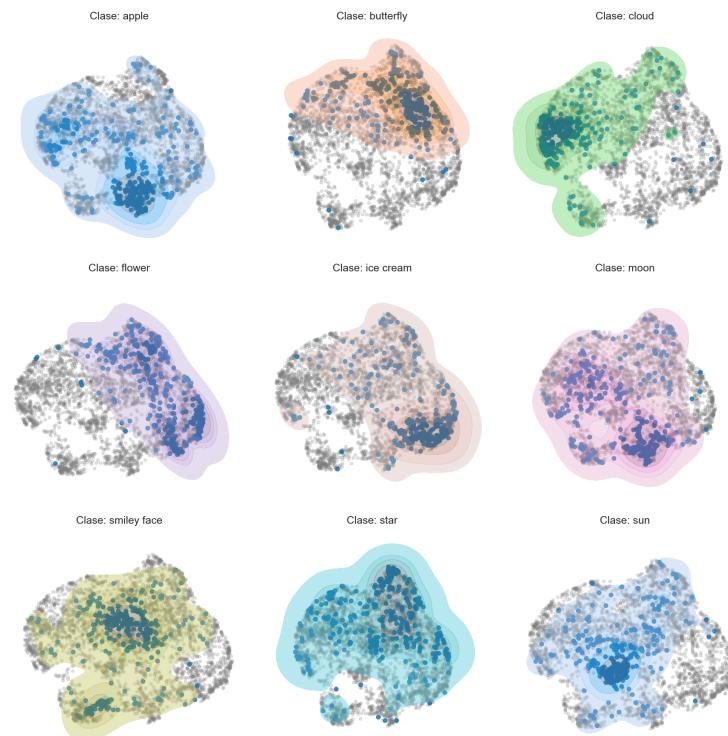


Figure 8. Visualización UMAP de las activaciones internas por clase

Las proyecciones mostraron una tendencia al agrupamiento de ejemplos dentro de cada clase, revelando zonas de concentración interna. Sin embargo, los límites entre clases no siempre estuvieron claramente definidos, lo que sugiere que las representaciones latentes capturan información semántica útil, aunque no completamente separable en todos los casos. Este análisis permitió validar que las activaciones del modelo organizaban el espacio de forma estructurada y útil para la clasificación, aunque no sea perfectamente separable.

4.5. Evaluación en condiciones reales

Finalmente, se evaluó la capacidad del modelo para clasificar dibujos hechos por humanos de forma libre en una interfaz personalizada.

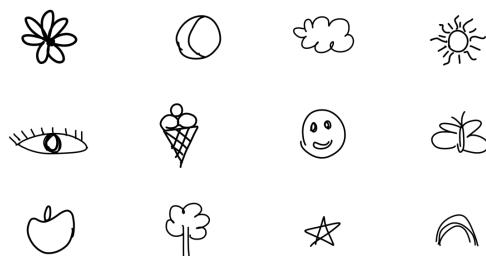


Figure 9. Doodles hechos a mano por una usuaria externa al dataset

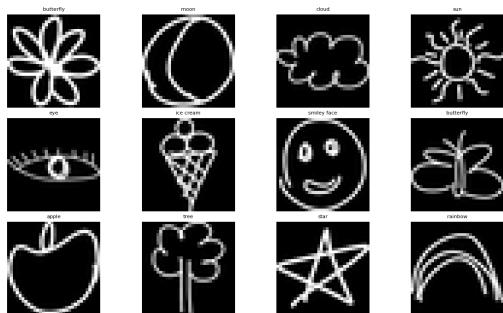


Figure 10. Predicciones del conjunto de doodles hechos por una usuaria externa al dataset

El modelo logró predecir correctamente la mayoría de los ejemplos. Algunas confusiones ocurrieron en clases con similitudes visuales (*por ejemplo: árbol vs flor*), pero en general se observó una buena generalización fuera del dominio del entrenamiento.

5. Conclusiones

En este trabajo se desarrolló un sistema de clasificación de dibujos a mano utilizando una red neuronal convolucional entrenada sobre un subconjunto del dataset *QuickDraw*. A lo largo del proceso, se avanzó desde arquitecturas simples hacia una red más robusta (*SimpleCNN*), que demostró ser capaz de generalizar no solo sobre el conjunto de test, sino también frente a ejemplos reales y no vistos, incluso aquellos generados en vivo por usuarios.

El uso de técnicas de *data augmentation* fue clave para mejorar la robustez del modelo ante pequeñas variaciones en escala, rotación o trazo, contribuyendo a evitar el sobreajuste y permitiendo al sistema mantener un rendimiento elevado en contextos diversos. Las evaluaciones cuantitativas mostraron una *accuracy* global del 98%, con métricas consistentes a través de todas las clases.

Además, se exploraron mecanismos de interpretabilidad como *Grad-CAM* y promedios por clase, que revelaron que la red aprendió a focalizar su atención en regiones informativas y a captar patrones representativos por categoría. Complementariamente, las proyecciones mediante t-SNE y UMAP permitieron visualizar cómo el modelo organiza internamente sus representaciones, mostrando agrupamientos internos coherentes aunque no perfectamente lineales entre clases.

Finalmente, la implementación de una aplicación de dibujo en tiempo real validó el uso del modelo en un entorno interactivo, comprobando su capacidad para reconocer trazos libres y variados. Estos resultados reflejan que incluso modelos relativamente simples, si son correctamente entrenados y analizados, pueden abordar con éxito tareas complejas como el reconocimiento de *doodles*, abriendo la puerta a aplicaciones creativas y educativas en el ámbito de la **visión artificial**.

6. Bibliografía

- Cátedra de Visión Artificial (I308), Universidad de San Andrés. *Materiales de clase: presentaciones y Jupyter Notebooks*. Campus Virtual, 2025.
- Google Creative Lab. *Quick, Draw! Dataset*. GitHub repository. Available at: <https://github.com/googlecreativelab/quickdraw-dataset> (Accessed: May, 2025).
- Scikit-learn developers. *t-SNE: t-distributed Stochastic Neighbor Embedding*. Documentación oficial de scikit-learn. Disponible en: <https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html> (Accedido: 30 de junio de 2025).
- McInnes, L., & Healy, J. *UMAP: Uniform Manifold Approximation and Projection*. Documentación oficial de umap-learn. Disponible en: <https://umap-learn.readthedocs.io/en/latest/> (Accedido: 30 de junio de 2025).
- Captum. *Guided Grad-CAM — Captum 0.6.0 documentation*. Disponible en: https://captum.ai/api/guided_grad_cam.html (Accedido: 30 de junio de 2025).