

Clasificación de Imágenes de Cartas usando Deep Learning

Gabriela Chimali Nava Ramírez | A01710530

09/11/2025

ABSTRACT La clasificación automática de imágenes es un desafío en la visión por computadora porque requiere arquitecturas capaces de aprender características complejas y muy variables. En este trabajo, se implementó y comparó una arquitectura de Deep Learning personalizada frente a estrategias de Transfer Learning. Contrario a la hipótesis inicial, se observó que una CNN básica entrenada desde cero (v1) superó significativamente en desempeño a la implementación directa de Transfer Learning con MobileNetV2 (v2), evidenciando una fuerte brecha de dominio entre las imágenes de ImageNet y los patrones geométricos de las cartas. Sin embargo, mediante la aplicación de técnicas de Fine Tuning (v4) y optimización avanzada con callbacks (v5), se logró adaptar la arquitectura pre-entrenada para alcanzar una precisión superior y una mayor capacidad de generalización.

1. INTRODUCCIÓN

El uso de algoritmos de *Deep Learning*, específicamente las Redes Neuronales Convolucionales (CNNs), ha revolucionado el campo de la visión por computadora. Ya que permiten extraer patrones jerárquicos de las imágenes, desde bordes simples hasta formas complejas como números o letras, siendo fundamentales para tareas de clasificación. Una de las técnicas más efectivas y eficientes computacionalmente es el *Transfer Learning* (aprendizaje por transferencia), que permite aprovechar el conocimiento de modelos pre-entrenados en datasets masivos para aplicarlos a tareas más específicas, reduciendo significativamente el tiempo de entrenamiento y los datos necesarios.

En este reporte, se plantea la aplicación y mejora progresiva de un modelo de *Deep Learning* (aprendizaje profundo) para la clasificación de un conjunto de datos de 53 clases de cartas. Se documenta la evolución a través de cinco versiones, iniciando con una CNN Simple (v1) que estableció una línea base sólida.

Posteriormente, se exploró *Transfer Learning* con *MobileNetV2* (v2), el cual mostró inicialmente un desempeño inferior al modelo básico, pues características pre-entrenadas en objetos naturales no son directamente transferibles a símbolos abstractos (cartas). Así fue necesario refinar el modelo mediante capas densas (v3), *Fine Tuning* (v4) y un entrenamiento extendido con tasa de aprendizaje adaptativa (v5) para superar la barrera de rendimiento de la CNN Simple.

2. DESCRIPCIÓN DEL DATASET

Este proyecto se llevó a cabo utilizando el conjunto de datos “*Cards Image Dataset-Classification*”, el cuál contiene imágenes de cartas de baraja.

2a.- Estructura de Datos

El *dataset* está pre dividido en tres carpetas: *train* (entrenamiento), *valid* (validación) y *test* (prueba).

Todas las imágenes son de 224x224x3 píxeles en formato jpg y están recortadas de forma que sólo una carta sea visible, ocupando el 50% de los píxeles del espacio total. El modo es RGB.



Figura 1. Ejemplo de las cartas en train

i. Variable Objetivo (Clases)

La variable dependiente es categórica, con 53 clases únicas que representan cada tipo de carta. Ejemplo:

- *ace of clubs*
 - *king of diamonds*
 - *seven of hearts*
 - *joker*

ii. Volumen de Datos

- **Entrenamiento:** 7624 imágenes
 - **Validación:** 265 imágenes (5 por clase)
 - **Prueba:** 265 imágenes (5 por clase)

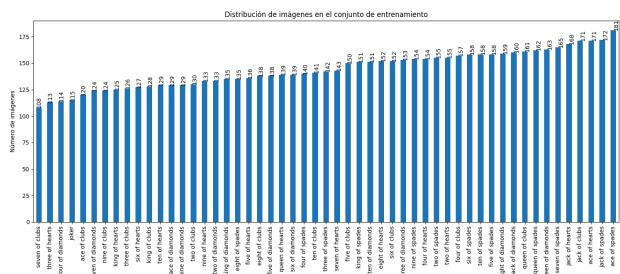


Figura 2. Distribución de clases en train

Como observamos, la representación de clase no es uniforme, por lo que posteriormente será importante contemplar técnicas para manejar este desbalance.

3. PREPROCESAMIENTO Y TRANSFORMACIÓN DE DATOS

3a.- Carga y exploración de datos

Se utiliza *ImageDataGenerator* de Keras para adecuar las imágenes al formato requerido por la arquitectura, apliando las siguientes transformaciones:

- **Normalización**, reescalando los píxeles dividiendo los valores de los píxeles por 224 (1./224). Esto transformó el rango de intensidad original [0, 255] a un rango normalizado [0, 1], esto ayuda a estabilizar los gradientes y acelera la convergencia durante el entrenamiento.
- **Redimensionando** todas las entradas a una resolución de 224 x 224 píxeles, preparándolas para la capa de entrada de la red.

3b. Data Augmentation (Aumento de Datos)

Para mejorar la capacidad de generalización del modelo y evitar el sobreajuste, se implementa una estrategia de aumento de datos exclusivamente en el conjunto de entrenamiento simulando variaciones del mundo real. Estas transformaciones consienten en:

- **Rotación ($\pm 20^\circ$) y Cizallamiento (5%)**: Simula la inclinación natural de la mano o de la carta sobre una superficie no plana.
- **Desplazamiento (10%) y Zoom (10%)**: Son cartas que no están perfectamente centradas o cuentan con variaciones por distancia de la cámara.
- **Volteo Horizontal (Flip)**: Aprovecha la simetría lateral presente en la mayoría de los diseños duplicando la variedad de orientaciones sin perder información.
- **Ajuste de Brillo ([0.8, 1.2])**: Se introdujo una variación multiplicativa de intensidad del $\pm 20\%$. Esto permite al modelo generalizar bajo diferentes condiciones de iluminación (sombra, luz artificial o luz natural) sin depender del contraste original de la imagen.

4. CONSTRUCCIÓN DEL MODELO BASE

Para establecer un punto de partida y evaluar la complejidad del problema, se diseñó una primera arquitectura (v1) basada en una Red Neuronal Convolutacional (CNN) sencilla, entrenada desde cero.

Esta consta de dos capas convolucionales básicas, *Conv2D*, seguidas de *MaxPooling2D* para reducir la dimensionalidad y un clasificador con capas densas *Flatten* y *Dense*.

Este modelo mostró un desempeño limitado por lo que se explorará una arquitecturas más robustas para distinguir entre 53 clases con patrones visuales similares y variaciones significativas entre diseños. (Gráficas completas en la sección 10b. en RECURSOS)

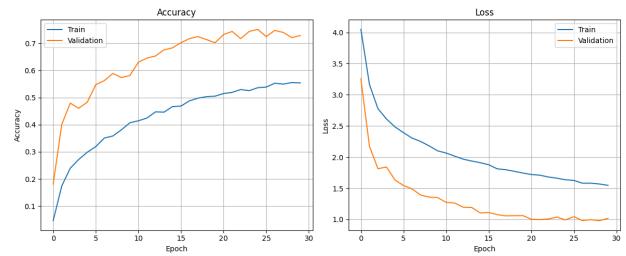


Figura 3. Curvas de Accuracy y Pérdida (Loss) para la arquitectura base

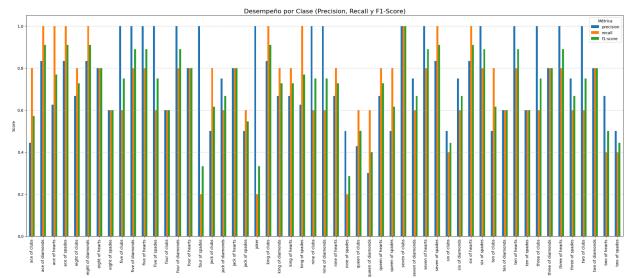


Figura 4. Métricas por Clase para la arquitectura base, una CNN simple (v1)

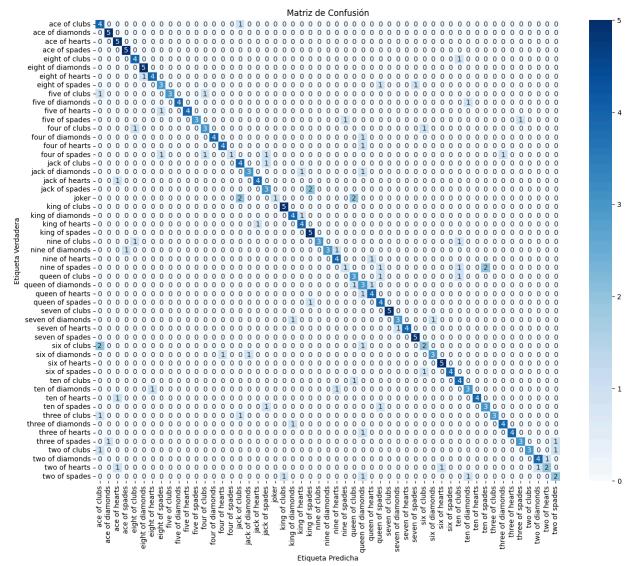


Figura 5. Matriz de confusión por Clase para la arquitectura base

5. MODIFICACIONES DEL MODELO

A partir del modelo base (CNN simple), se implementaron modificaciones incrementales para superar sus limitaciones y evaluar la viabilidad del *Transfer Learning*.

En la sección 10a. en *RECURSOS* se puede observar un resumen de la evolución de las métricas por cada versión.

5a. Versión 2: Transfer Learning (MobileNetV2) y Data Augmentation

Se sustituyó la base convolucional propia por *MobileNetV2*, un modelo pre-entrenado en el dataset *ImageNet* (1.4 millones de imágenes), congelando sus pesos para utilizarlos como extractores de características fijas.

En esta parte se agregaron al generador de imágenes todas las transformaciones descritas en la sección 3b. Data Augmentation para evitar la memorización de datos no cambiantes.

Para reducir el riesgo de sobreajuste. Se insertaron capas de *Dropout* (0.4) después de cada capa densa, que apagan aleatoriamente el 40% de las neuronas en cada paso.

Aún no se observa un incremento significativo en el *Accuracy* que pasa de **v1 = 71%** a **v2 = 51%** en test, como para afirmar que las características aprendidas en *ImageNet* transferibles a las cartas son mejores que el modelo base.

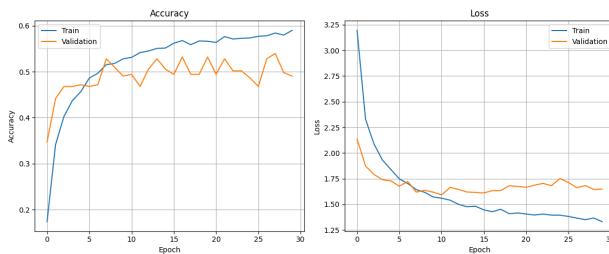


Figura 6. Curvas de Accuracy y Pérdida (Loss) para la versión 2

Figura 7. Métricas por Clase para la versión 2

5b. Versión 3: Ajuste de la Arquitectura (Capa Densa)

Para intentar mitigar el bajo desempeño de la v2, se añadió una capa densa intermedia de 256 neuronas antes de la salida.

Con la hipótesis de que la falta de capacidad de clasificación era el problema. Sin embargo, el desempeño apenas mejoró con *Accuracy* de **53%** en test. Además

como en la v2 identifica fielmente **9 clases**, mejorando simplemente de **3 a 2 clases** con 0% de *Accuracy* respecto a la v2. Esto confirma que el problema raíz es la extracción de características del modelo congelado y no el clasificador.

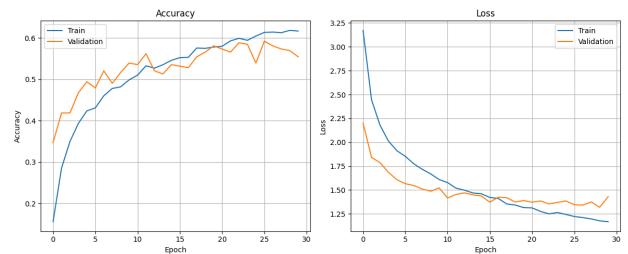


Figura 8. Curvas de Accuracy y Pérdida (Loss) para la versión 3

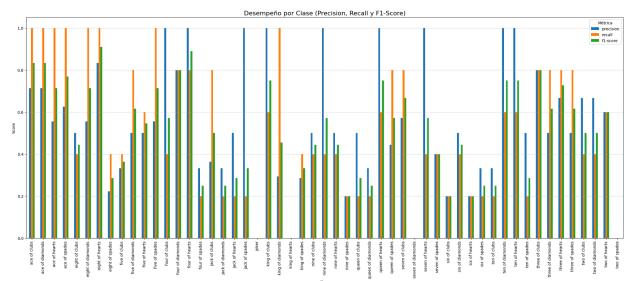


Figura 9. Métricas por Clase para la versión 3

5c. Versión 4: Fine Tuning

Debido a que no se observaba una mejora significativa de *Transfer Learning* sobre la CNN simple (v1). Se deduce que las características que busca *MobileNetV2*, como texturas, degradados suaves, entre otras formas y texturas más complejas, no son las más útiles para distinguir las cartas que son geométricas, de bajo contraste y simbólicas.

Es por esto para permitir que las últimas capas convolucionales de *MobileNetV2* se adapten a las formas de las cartas y no se limite a usar lo que ya sabe, se implementó una estrategia de descongelamiento progresivo de los últimos bloques convolucionales para permitir que los pesos se actualicen mediante *Backpropagation*. Y se usó un *Learning Rate* muy bajo (1e-5) para evitar destruir la información pre-entrenada.

Igualmente, dado el tamaño del dataset de aproximadamente 7,000 imágenes, se eliminan las capas densas para evitar sobreajuste, evitando que el modelo memorice ruido o factores muy específicos, en lugar de generalizar.

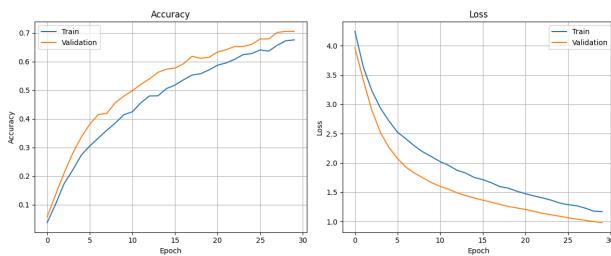


Figura 10. Curvas de Accuracy y Pérdida (Loss) para la versión 4

Figura 11. Métricas por Clase para la versión 4

5d. Versión 5: Agregar épocas y Optimización con de entrenamiento

Dado que se observa que el modelo mejora lenta y progresivamente con *Fine Tuning* y puede seguir mejorando, se extendió el entrenamiento permitiendo hasta 100 épocas. Para evitar el sobreajuste y entrenar el modelo de forma más estable y eficiente, se implementaron los siguientes callbacks:

- **ModelCheckpoint:** Guarda el modelo en el archivo “classifier_model.keras” únicamente cuando obtiene la mejor precisión al evaluar el conjunto de validación (save_best_only=True), buscando siempre el valor máximo (mode='max') y mostrando cuando ocurre (verbose=1).
- **EarlyStopping:** Detiene el entrenamiento cuando la pérdida al evaluar el conjunto de validación deja de mejorar al menos 0.001 durante 5 épocas consecutivas. Además restaura los mejores pesos alcanzados (restore_best_weights = True) para guardarlos en el archivo final.
- **ReduceLROnPlateau:** Reduce la tasa de aprendizaje cuando la pérdida en validación no mejora durante 5 épocas (patience=5), multiplicándose por 0.2, sin permitir que baje de 1e-7.

Estas tres funciones trabajan juntas para mejorar la estabilidad, eficiencia y calidad final del modelo entrenado, supervisando su comportamiento y tomando acciones sin tener que intervenir manualmente.

6. EVALUACIÓN DE RESULTADOS

El modelo final (v5) fue entrenado durante un máximo de 100 épocas, teniendo el mejor resultado en la 95.

6a. Estabilidad y Convergencia (Gráficas de Accuracy y Loss)

Como se aprecia en las curvas de aprendizaje de la **Figura 12**, a diferencia de la versión con Transfer Learning congelado (v2), que se estancaba pronto alrededor del 55%, la curva de entrenamiento de la v5 muestra un ascenso constante y suave hasta alcanzar un 92.52%. La curva de validación (línea naranja) logra acompañarla hasta situarse en un 85.66%. Aunque existe una brecha de aproximadamente 7%, se mantiene estable a lo largo de las épocas, lo que indica que el Dropout (0.4) está cumpliendo su función de mitigar el sobreajuste.

Por otro lado, la gráfica de pérdida confirma la estabilidad del entrenamiento. Se observa una convergencia hacia valores mínimos (Loss Train: 0.29, Loss Validation: 0.48). La ausencia de picos o divergencias bruscas en la validación se atribuye al Learning Rate reducido (1e-5), por lo que el optimizador desciende con pasos cortos hacia el mínimo global del error.

Finalmente, el entrenamiento se detuvo automáticamente cerca de la época 95 antes de completar las 100 épocas programadas para evitar degradar el desempeño en datos nuevos.

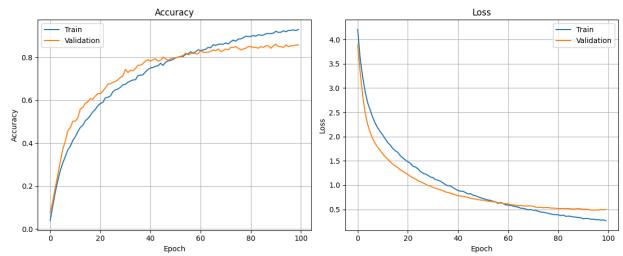


Figura 12. Curvas de Accuracy y Pérdida (Loss) para la versión final

6b. Desempeño Detallado por Clase

Las clases correspondientes a cartas numéricas bajas como “Ace of Spades, Two of Clubs” y cartas de corte estándar “King, Queen” mostraron un desempeño óptimo, alcanzando puntuaciones F1 cercanas a 1.0, logrando un balance casi perfecto entre Precisión (pocos falsos positivos) y Recall (pocos falsos negativos). El modelo aprendió a identificar eficazmente la forma del índice (número/letra en la esquina) y la densidad de píxeles única de las figuras reales.

Por otro lado, el desempeño para cartas intermedias del mismo color no es el mejor. Cartas como “Seven of Hearts” contra ”Eight of Diamonds”. Parece priorizarse el color rojo sobre la forma geométrica del palo,

generando falsos positivos cruzados. Sin embargo, la tasa de error se mantiene controlada <15%.

La única clase que no parece identificar es la de “Joker”. A diferencia de los palos y números que siguen patrones de texto estandarizados, este tipo de carta presenta una varianza visual extrema entre barajas, colores, estrellas o textos adicionales simples. Esta falta de características comunes consistentes impidió que la red neuronal encontrara un patrón reconocible, sugiriendo para futuras iteraciones un dataset específico y más amplio sólo para esta clase.

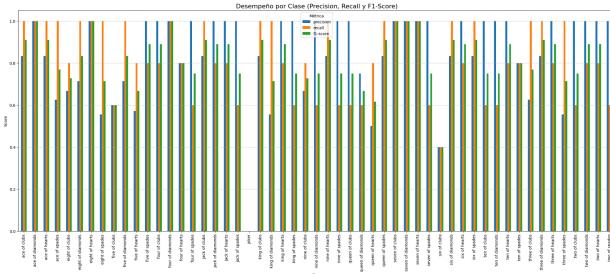


Figura 13. Métricas por Clase para la versión final.

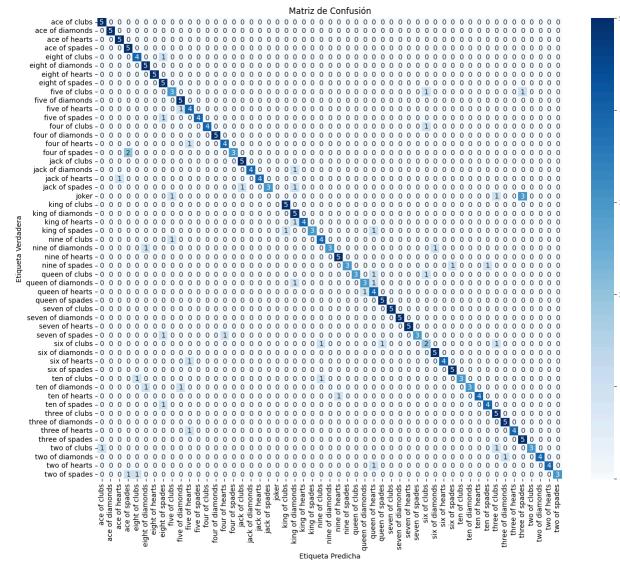


Figura 14. Matriz de confusión por Clase para la versión final.

6c. Análisis de Errores por Jerarquía

i. Confusión por Palo

Al analizar la matriz de confusión agrupada por palos **Figura 15**, se detectó que los errores residuales se concentran entre pares que comparten la misma tonalidad.

Corazones vs. Diamantes (color rojo) representa la tasa más alta de confusión. Al compartir el color rojo y tener una forma inferior similar (vértice en punta hacia abajo), el modelo tiende a intercambiar estas clases cuando la

resolución de la imagen no permite distinguir claramente la parte superior del corazón.

Picas vs. Tréboles (color negro) se observa un comportamiento similar, pero con menor frecuencia. Esto sugiere que el modelo distingue mejor la diferencia superior entre las tres hojas del trébol y la hoja única de la pica, en comparación con la sutileza entre la forma de corazones y diamantes.

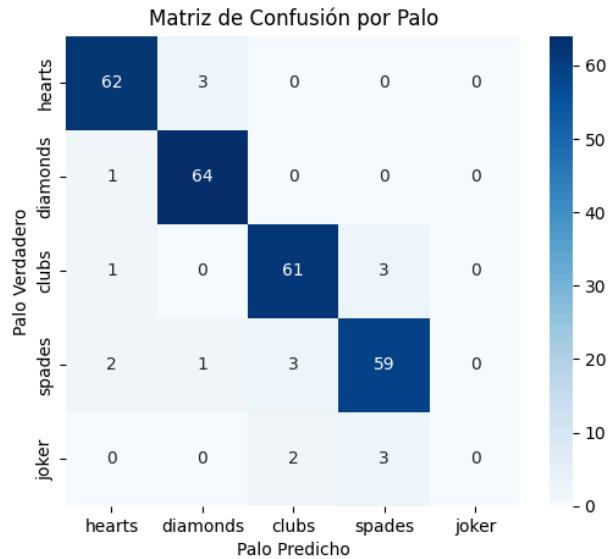


Figura 15. Matriz de confusión por Palo para la versión final.

ii. Confusión por Valor

La clasificación por valor numérico (**Figura 16**) muestra un desempeño global más robusto que la clasificación por palos, aunque presenta dos clústeres específicos de error:

Las cartas de corte o figuras presentan la mayor confusión entre sí (J, Q, K). Posiblemente por la alta densidad de información visual en el centro de la carta. Al redimensionar la imagen los detalles finos que distinguen a un Rey (King) de una Reina (Queen) se difuminan, dejando patrones muy similares en distribución de píxeles dificultando su diferenciación.

Se detectó una tasa de error notable entre los números 6 y 9. Probablemente un efecto secundario directo del Data Augmentation. Al entrenar con rotaciones aleatorias un 6 invertido comienza a invadir el espacio de la clase 9, generando ruido confundiendo al modelo en estas clases específicas.

Los valores numéricos bajos (A, 2, 3) y el 10 son fácilmente distinguibles. Estos patrones poseen pocos símbolos claramente separados sobre fondo blanco, lo que facilita que los filtros convolucionales extraigan bordes nítidos y características geométricas únicas.

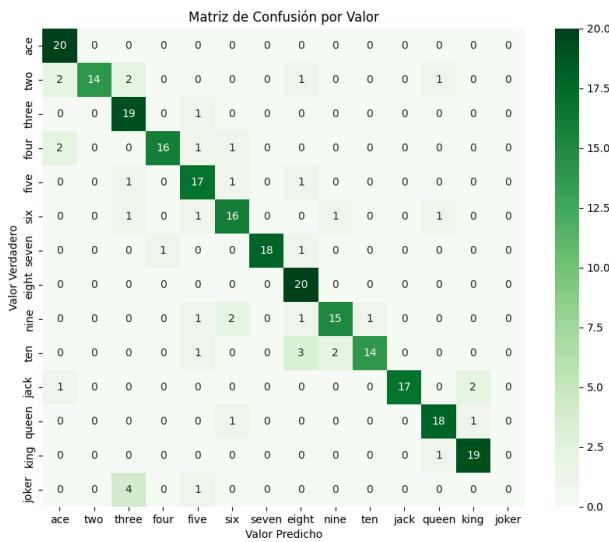


Figura 16. Matriz de confusión por Valor para la versión final.

7. PREDICCIONES

Para demostrar la utilidad práctica del modelo, se desarrolló una interfaz en Python ([Interfaz.ipynb](#)) que simula un entorno de producción.

7a. Preparación de la entrada

Importación el modelo final “classifier_model.h5” y el diccionario de clases “class_names.json”

Para ser consistente con el formato de las imágenes con las que fue entrenado el modelo, se agrega la función “obtener_prediccion” para redimensionar cualquier imagen de entrada a 224x224 y normalizar sus valores.

Se incluyen también las funciones para generar un reporte visual con la imagen de entrada, el valor de predicción más alto y su clase real, así como un gráfico de barras de las Top 5 probabilidades.

7b. Casos de Prueba

Verificando la capacidad del modelo se elaboraron diferentes escenarios de prueba para diferentes tipos de cartas y contextos.

i. Mismo valor en distintos palos

El modelo demostró capacidad para desacoplar la jerarquía visual de la carta, en la **Figura 17** con los “Aces”, identificando correctamente el número o letra independientemente del palo. Esto sugiere que la red neuronal prioriza la forma del carácter alfanumérico dominante para filtrar las posibles clases y luego refina la selección con el símbolo del palo.

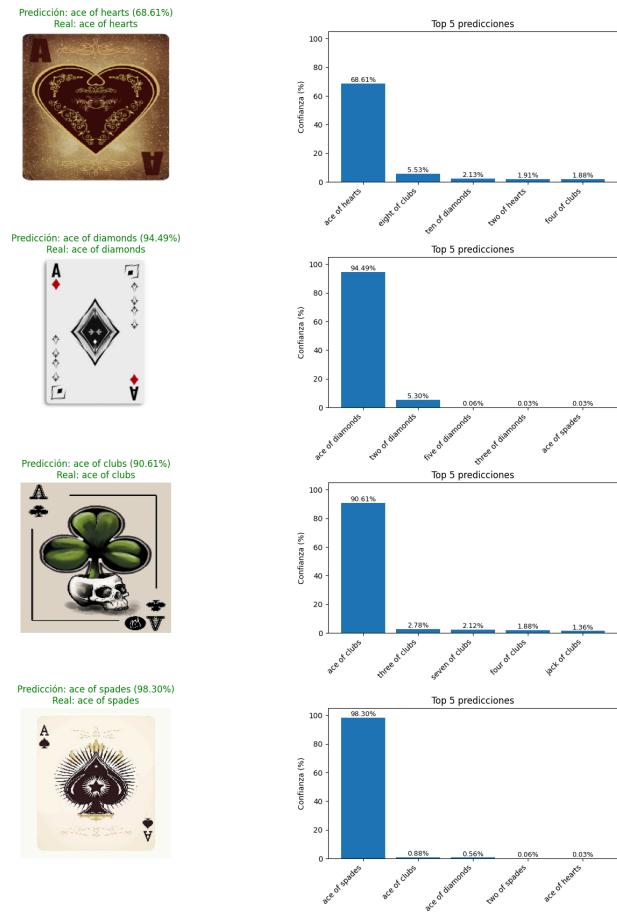


Figura 17. Prueba con imágenes inusuales de las clases con mejor desempeño.

ii. Mismo palo con distintos valores

Al evaluar cartas que comparten idénticos patrones de color y símbolo 6, 8, 9 de “Spades” en la **Figura 18**, el modelo clasificó correctamente las instancias, validando que la red no se limita a detectar la forma general del palo, sino que extrae características de los índices numéricos y las figuras de corte. Las probabilidades secundarias se mantuvieron predominantemente dentro de la misma familia de palo.

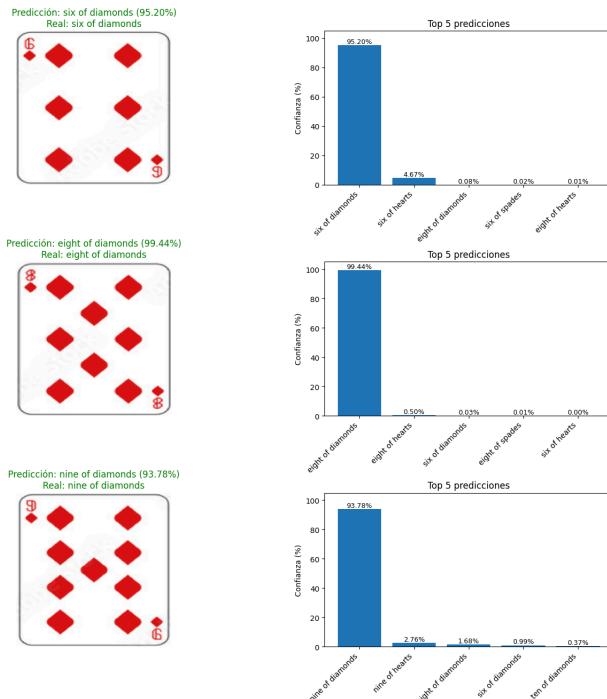


Figura 18. Prueba con imágenes inusuales de las clases con mejor desempeño.

iii. Imagen con rotación

La primera (arriba en la **Figura 19**) al contar con menos ruido por el fondo aunque está rotada, acierta con mayor confianza, la otra si bien es correcta, lo hace con menor confianza.

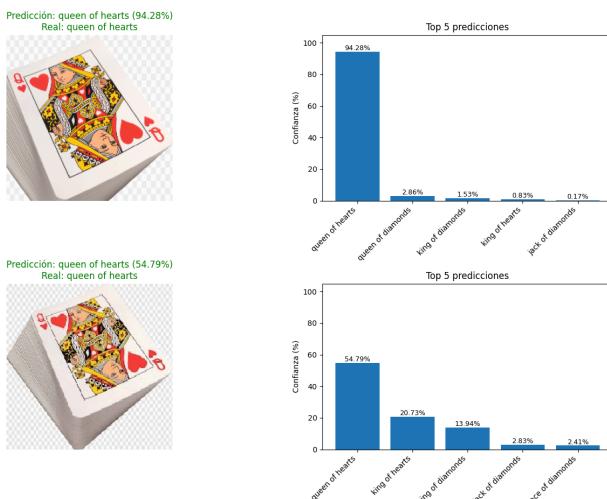


Figura 19. Prueba con imágenes rotadas

iv. Imagen muy similar al estándar de estilo de las cartas

La primera (arriba en la **Figura 20**) al no contar con ruido por el fondo acierta, la otra por el contrario se equivoca pero el valor real está entre el top de probabilidades.

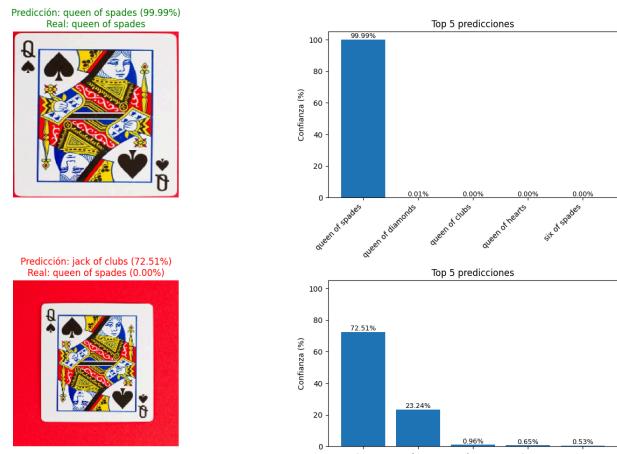


Figura 20. Prueba con imágenes similares al estilo estándar

El acierto en este ejemplo específico del joker en la **Figura 21** sugiere que el modelo reconoce eficazmente la representación estándar de la carta, fallando únicamente cuando el diseño gráfico se desvía drásticamente del promedio observado en el entrenamiento.

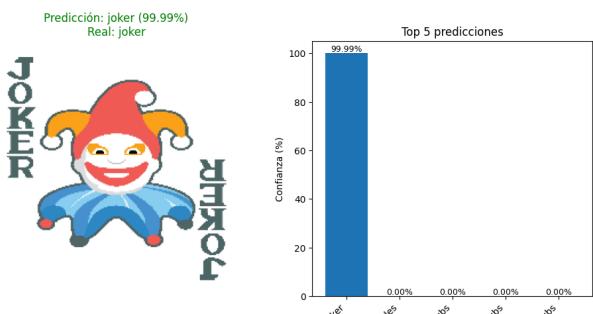


Figura 21. Prueba con peor clase de test (joker) con un estilo estándar

iv. Imagen muy variable al estilo estándar de las cartas

Al someter al modelo a cartas con diseños no tradicionales, se observó una notable disminución en la confianza de la predicción como en los ejemplos a continuación. Aunque la red logró clasificar correctamente la mayoría de las instancias, la dispersión de probabilidades entre clases evidencia que el modelo es sensible a la varianza por estilo.

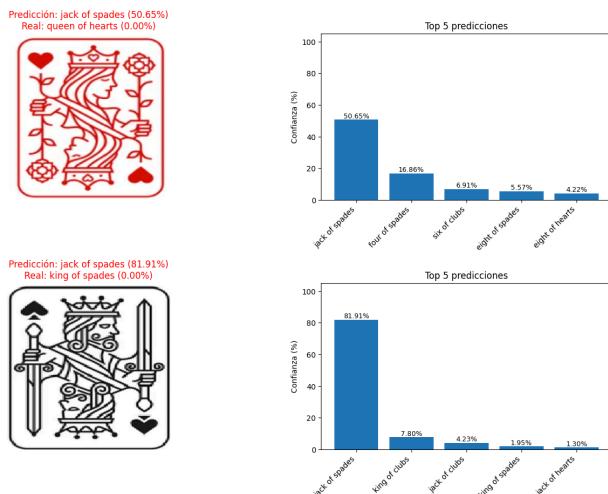


Figura 22. Prueba con imágenes muy variables estilo estándar

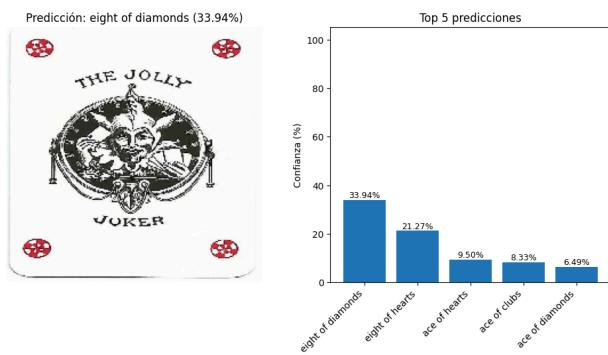


Figura 23. Prueba con peor clase de test (joker) con un estilo muy variable al estándar

8. CONCLUSIONES

El desarrollo de este proyecto permitió identificar que, contrario a la intuición inicial, la aplicación directa de *Transfer Learning*, una arquitectura compleja pre-entrenada *MobileNetV2* no es necesariamente superior a una CNN básica diseñada a medida cuando existe una brecha de dominio significativa. Mientras que la CNN simple (v1) logró adaptarse rápidamente a las formas geométricas de las cartas alcanzando un 71% de exactitud, el modelo pre-entrenado con *ImageNet* se estancó en 51%, evidenciando que las características aprendidas en estas fotos no son directamente transferibles a símbolos abstractos y planos. La mejora clave fue la implementación de una estrategia agresiva de *Fine Tuning* y optimización de hiperparámetros en la versión final (v5). Esta logró redirigir el conocimiento de la red hacia los bordes y patrones específicos de la baraja, superando finalmente la barrera del modelo base y alcanzando una exactitud en prueba del 81.13%.

Observaciones finales clave:

- El modelo aprendió a priorizar la lectura del número/letra sobre el símbolo del palo, mostrando gran precisión en la clasificación de valores pero cierta dependencia al distinguir palos del mismo color.
- La clase “Joker” representó el caso de fallo más crítico por la falta de patrones visuales consistentes.
- El uso de Data Augmentation fue efectivo, permitiendo que el modelo clasifique correctamente cartas inclinadas en la interfaz de prueba, simulando condiciones de uso real.

Finalmente para llevar este sistema a un entorno de producción, el siguiente paso lógico podría ser migrar de una arquitectura de clasificación más avanzada como una de Detección de Objetos (como YOLO), para localizar y clasificar múltiples cartas simultáneamente en una mesa sin necesidad de recortes manuales.

9. REFERENCIAS

- [1] giose. (2025, octubre 29). *CardClassifier MobileNetV2 based*. Kaggle.
<https://www.kaggle.com/datasets/gpiosenka/cards-image-datasetclassification>
- [2] GeeksforGeeks. (2024, 10 de septiembre). *Multiclass image classification using Transfer learning*. GeeksforGeeks.
<https://www.geeksforgeeks.org/deep-learning/multiclass-image-classification-using-transfer-learning/>
- [3] TensorFlow (2024, 24 de abril). *Module: tf.keras.callbacks*. TensorFlow.
https://www.tensorflow.org/api_docs/python/tf/keras/callbacks
- [4] Weirich, A. (2024, 4 de julio). *Finetuning TensorFlow/Keras Networks: Basics Using MobileNetV2 as an Example*. Medium.
<https://medium.com/@alfred.weirich/finetuning-tensorflow-keras-networks-basics-using-mobilenetv2-as-an-example-8274859dc232>

10. RECURSOS

	CNN simple, dos capas convolucionales básicas, Conv2D, seguidas de MaxPooling2D para reducir la dimensionalidad y un clasificador con capas densas Flatten y Dense.	MultiNetV2 con Data Augmentation y Dropout de 0.4	Agregar capa Densa on 256 parámetros	Agregar Fine Tuning	Aumentar épocas y agregar callbacks: EarlyStopping, ModelCheckpoint y ReduceLROnPlateau
Métrica	Modelo v1	Modelo v2	Modelo v3	Modelo v4	Modelo v5
Accuracy train	68.37%	62.81%	61.73%	66.28%	92.52%
Accuracy validation	85.66%	55.47%	58.49%	69.81%	85.66%
Accuracy test	71.32%	51.32%	53.58%	70.19%	81.13%
Loss train	1.0723	1.137	1.2043	1.1798	0.2907
Loss validation	0.5269	1.431	1.4562	0.9716	0.4842
Loss test	1.1677	1.6239	1.5536	1.0714	0.6452
No. Épocas	30	30	30	30	En la época 95 de 100 alcanzó el mejor desempeño y restauró los pesos
Learning Rate	0.001	0.001	0.001	0.00001	0.00001
Clases que predice mejor en TEST	16 - 100%	9 - 100%	9 - 100%	19 - 100%	27 - 100%
Clases que no logra identificar en TEST	Identifica todas, la más baja es 30% (queen of diamonds)	3 - 0% (joker, king of hearts, seven of diamonds)	2 - 0% (seven of clubs, two of spades)	1 - 0% (joker)	1 - 0% (joker)

Figura 24. Tabla de métricas reportadas en diferentes versiones del modelo

10b. Gráficas de la versión 1: CNN Simple

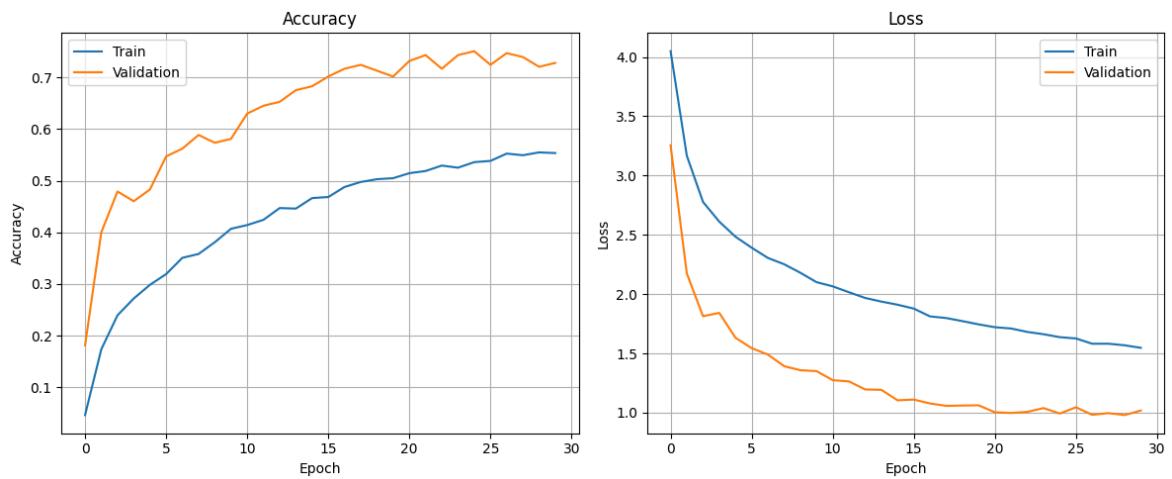


Figura 3. Curvas de Accuracy y Pérdida (Loss) para la arquitectura base

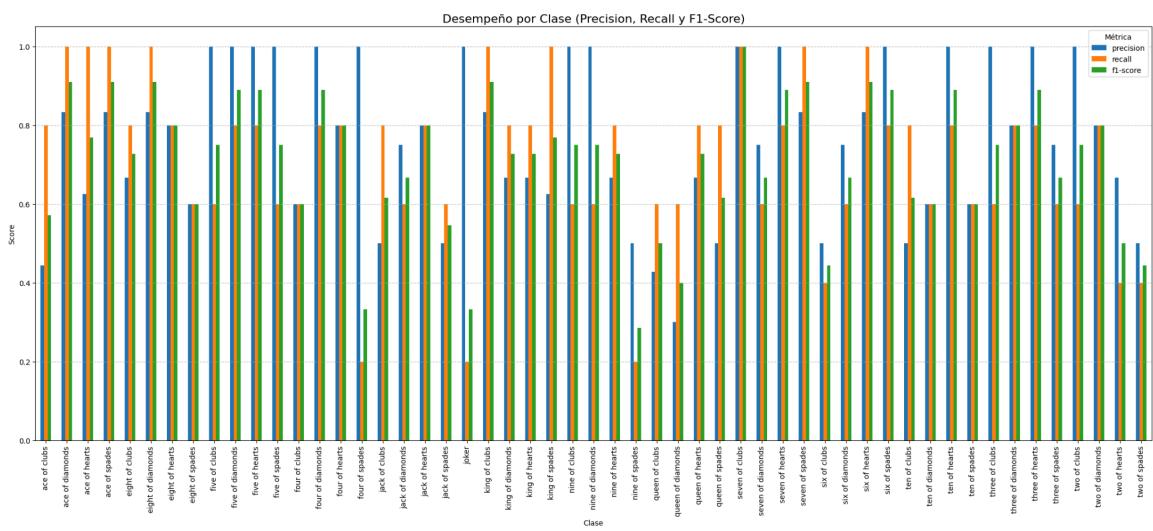


Figura 4. Métricas por Clase para la arquitectura base, una CNN simple (v1)

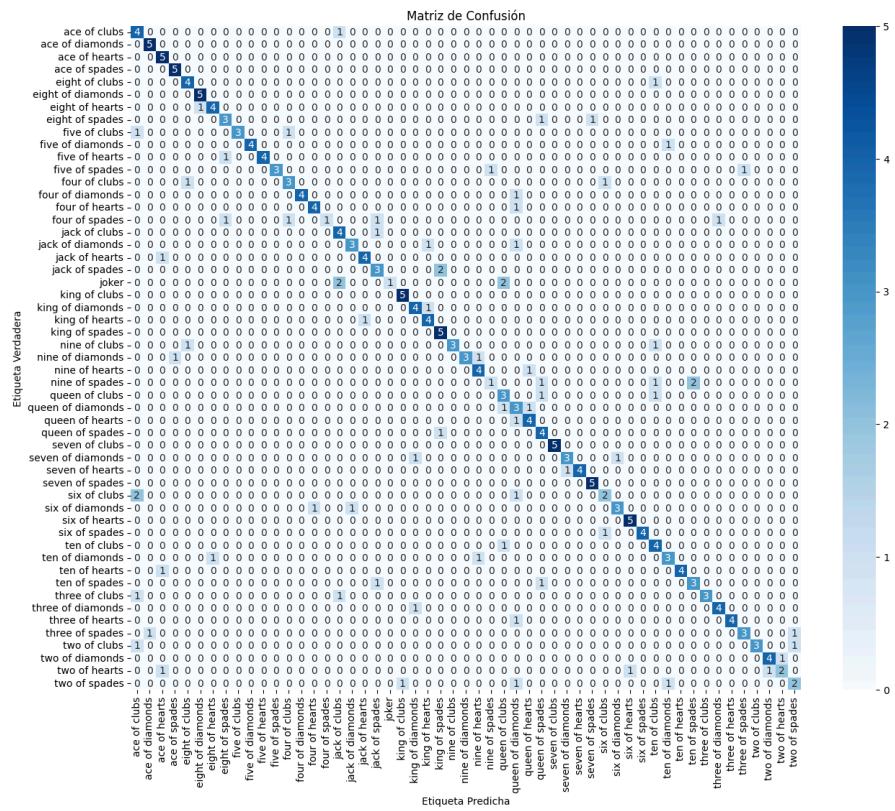


Figura 5. Matriz de confusión por Clase para la arquitectura base

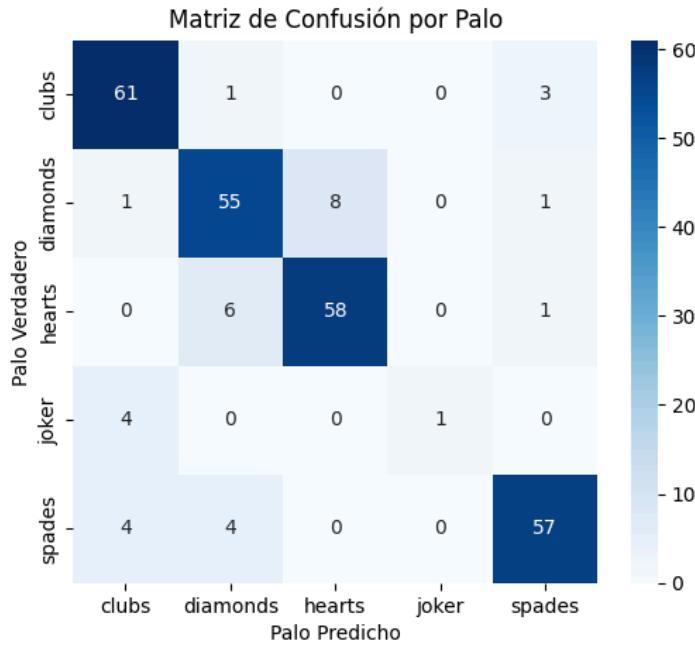


Figura 25. Matriz de confusión por Palo para la arquitectura base

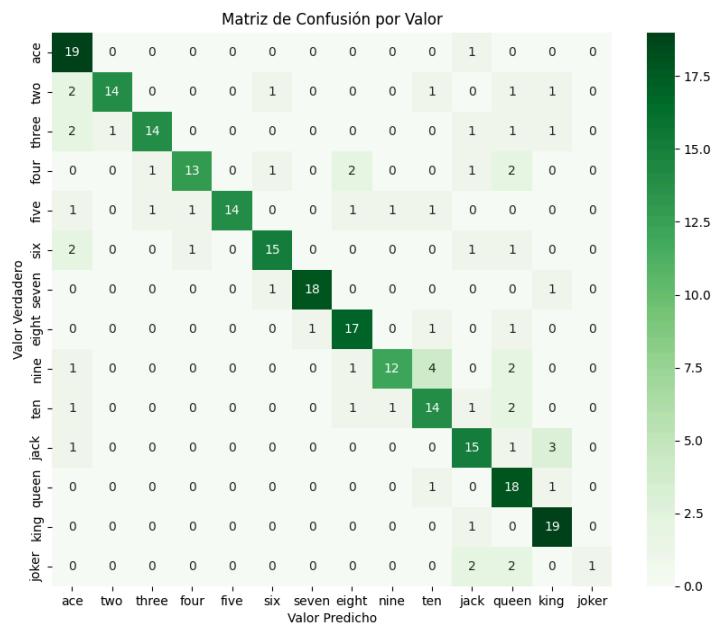


Figura 26. Matriz de confusión por Valor para la arquitectura base