PROYECTO Reconocimiento de caracteres alfanuméricos escritos a mano

Julian Ricardo Salazar Duarte

Universidad Nacional
Medellín
2024

Pregunta de investigación

¿Cuál es el rendimiento de diferentes algoritmos para la tarea de reconocer caracteres alfanuméricos escritos a mano?

Objetivos:

- Desarrollar y comparar diferentes modelos y técnicas de *machine learning* para la clasificación de caracteres escritos a mano.

- Analizar los errores de clasificación más comunes y proponer estrategias para mejorar el rendimiento del sistema en casos difíciles.

- Evaluar el rendimiento de los modelos desarrollados y seleccionar el mejor modelo en función de las métricas de evaluación.

Análisis de los datos

Para realizar el proyecto, se utilizó el dataset de Kaggle: https://www.kaggle.com/datasets/nabeel965/handwritten-words-dataset [1]. Este *dataset* cuenta con una carpeta img que contiene imágenes de caracteres alfanuméricos en inglés, y un documento english.csv que mapea cada imagen con su respectiva clase. En total, existen 62 clases correspondientes a las letras de a-z y A-Z, y a los números de 0 a 9. El dataset está distribuido con 55 elementos de cada clase, para un total de 3,410 imágenes.

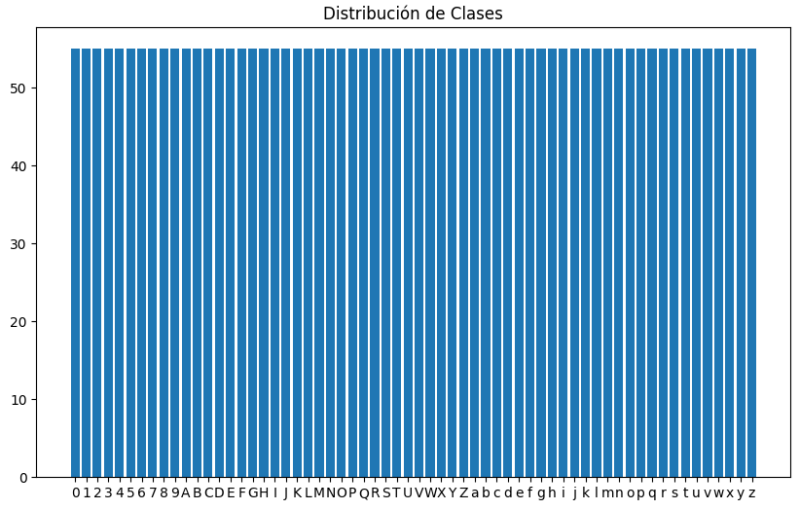

Figura 1. Distribución de clases.

Se utilizaron técnicas como *Label Encoder* para mapear todas las clases a números y convertirlas en un formato categórico *one-hot* en el caso de las redes neuronales convolucionales. Las imágenes fueron normalizadas, transformadas a escala de grises y redimensionadas a un tamaño de 32x32 píxeles.

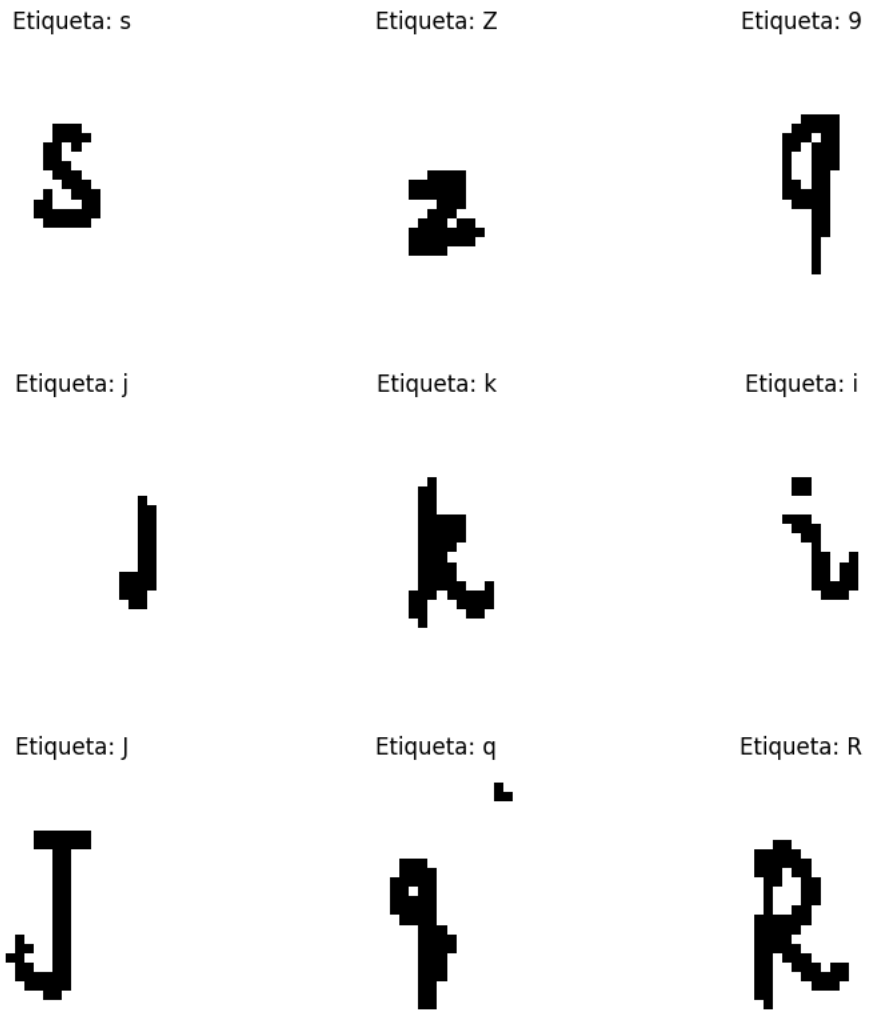

Figura 2. ejemplo de algunas imágenes re-dimensionadas

Se realizó un aumento de datos sobre las imágenes (5 por cada imagen), donde de manera aleatoria se aplicó una rotación de entre -15° y 15°, un zoom de ±10% y un desplazamiento horizontal y vertical de ±10%. Para rellenar los nuevos píxeles, se utilizó *nearest*, que consiste en rellenar los nuevos píxeles en función de los píxeles vecinos. Con el aumento de datos, el total de elementos fue de 20,490 imágenes. Por último, se dividió el *dataset* en un 60% para datos de entrenamiento (12,276), 20% para validación (4,092) y 20% para prueba (4,092).

Figura 3. ejemplo de imagen aumentada

Modelos de machine learning

Para probar el rendimiento de cada modelo se usaron librerías tales como Scikit-learn para la implementación de los modelos SVC, Random Forest y para el análisis de errores y rendimientos de todos los modelos; TensorFlow/Keras para la construcción, entrenamiento y optimización de las redes neuronales convolucionales; MLflow para rastrear los experimentos, versiones de modelos y facilitar la replicación de resultados; Optuna para la búsqueda de hiperparámetros; y librerías como Seaborn, Matplotlib, NumPy y Pandas para manipular y mostrar los datos e información obtenida durante la evaluación de los modelos. La métrica para comparar los diferentes modelos fue la *accuracy*. Primero, se realizó un entrenamiento con los valores por defecto de las librerías utilizadas y, en el caso de la red neuronal convolucional, se utilizó una arquitectura arbitraria. Se probaron tres tipos de modelos:

SVC (C-Support Vector Classification): el modelo base, con los parámetros por defecto: C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None.

Para mejorar el rendimiento del modelo, se realizó una búsqueda de hiperparámetros para los siguientes parámetros: C (entre 0.001 y 10), gamma con los valores 'auto', 'scale', 0.1 y 0.5, kernel con los valores 'rbf', 'linear', 'poly' y degree en caso de que el valor del kernel fuera 'poly', usando valores enteros entre 2 y 5.

Al finalizar la búsqueda de hiperparámetros, el mejor rendimiento se obtuvo con los valores C: 5.269903391488448, gamma: scale, kernel: rbf.

Random Forest Classifier: el modelo base con los hiperparámetros por defecto: n_estimators=100, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None, monotonic_cst=None.

La búsqueda de hiperparámetros arrojó los siguientes valores óptimos: n_estimators: 464, max_depth: 47, min_samples_split: 7, min_samples_leaf: 1, max_features: 10.

Red neuronal convolucional (CNN): el modelo base de la CNN tiene capas convolucionales seguidas de *pooling* para extraer características, luego *flattening* y capas densas para la clasificación final. Se utilizó el optimizador Adam, con una tasa de aprendizaje (*learning rate*) de 0.001 y la función de pérdida categorical_crossentropy.

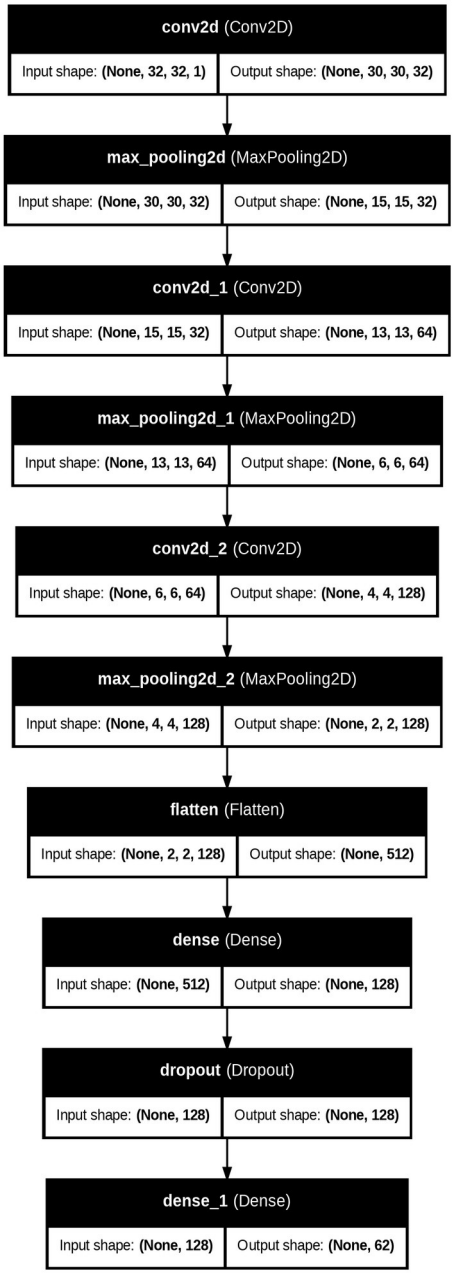

Figura 4. Arquitectura CNN base

Tras una búsqueda de hiperparámetros por el numero de filtros entre 16 y 28, el tamaño del kernel entre 3 y 7, el número de capas entre 1 y 4, *learning rate* entre 1e-5, 1e-2y un *dropout* entre 0 y 0.5, y función de activación entre relu, elu, selu, tanh el mejor modelo encontrado fue 'num_filters': 47, 'kernel_size': 5, 'num_layers': 4, 'learning_rate': 0.0010538504774863038, 'dropout_rate': 0.3770022049861306, 'activation': 'relu', esta arquitectura es más profunda que la anterior, con más capas convolucionales, uso consistente de normalización por lotes después de cada convolución, y un aumento gradual en el número de filtros.

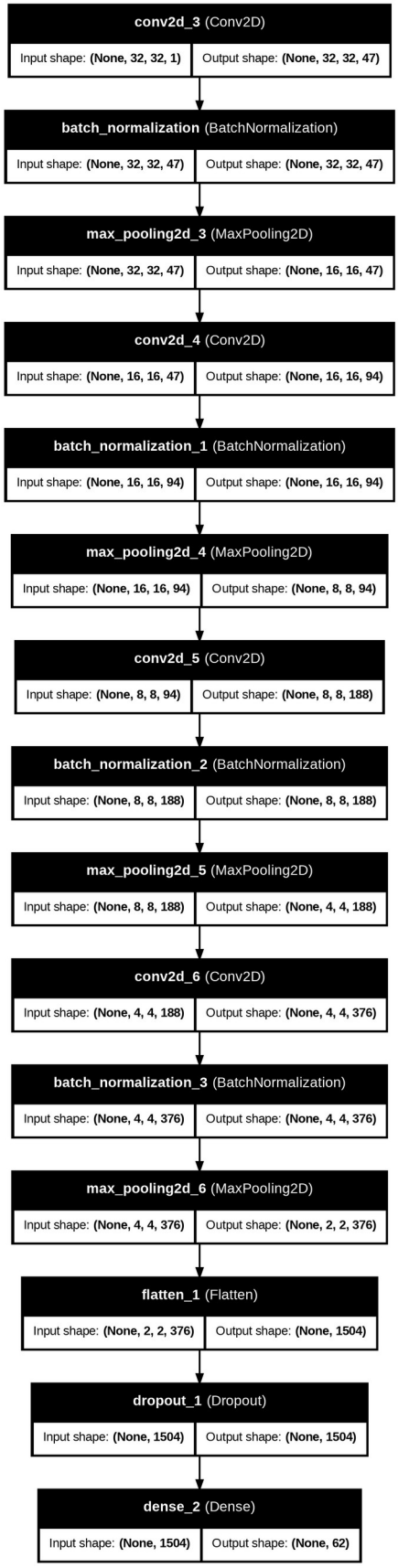

Figura 5. Arquitectura CNN mejorada

Este modelo tenia una varianza del 10% aproximadamente por lo que se buscó optimizar el modelo con una combinación de profundidad, normalización por lotes y *dropout* dando un enfoque para prevenir el sobreajuste.

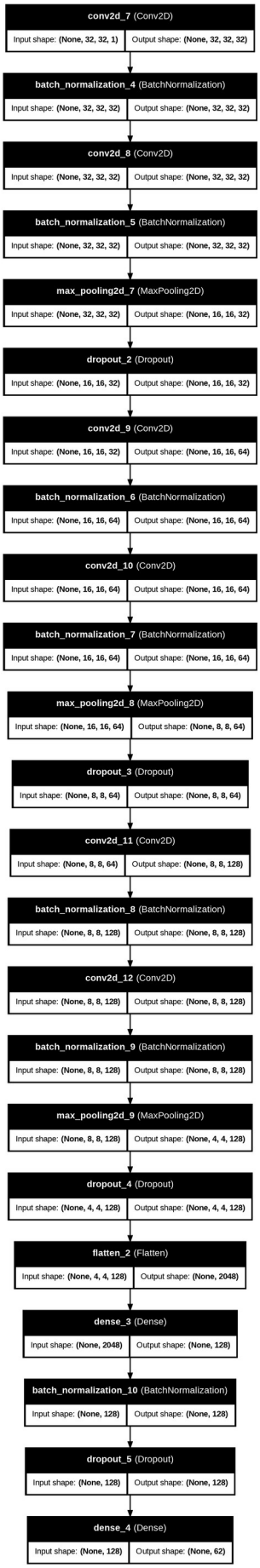

Figura 6. Arquitectura CNN optimizada

Al comparar los diferentes modelos según su precisión obtenemos los siguientes resultados:

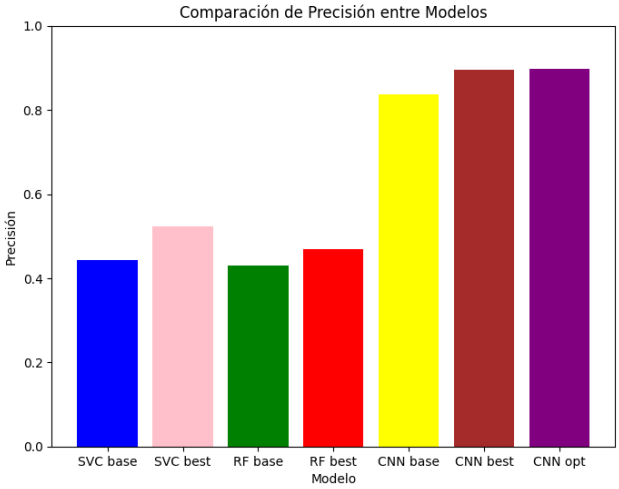

Figura 8. Precisión de los diferentes modelos

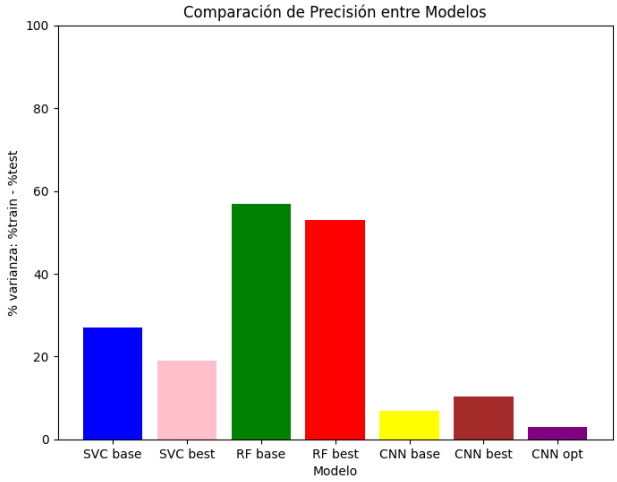

Figura 9. Varianza de los modelos

Las redes neuronales convolucionales (CNN) son claramente superiores en términos de precisión frente a SVC y Random Forest para este problema de reconocimiento de caracteres alfanuméricos, y con una varianza mucho menor entre los conjuntos de entrenamiento y prueba, lo que las hace más efectivas y confiables.
SVC muestra una mejora notable tras la optimización tanto en rendimiento como en varianza, sin embargo sigue siendo menos precisa que los modelos CNN.
Random Forest tiene el rendimiento más bajo y parece no beneficiarse mucho de la optimización, y muestra un alto nivel de varianza, lo que indica sobreajuste y baja generalización, sugiriendo que este tipo de modelo puede no ser el más adecuado para este tipo de datos o tarea.
El mejor Modelo encontrado evaluando y comparando los resultados es la red CNN optimizada (CNN optimized).

Con el mejor modelo obtenido se realizó un análisis de errores, con el fin de intentar optimizar aún más el modelo, para esto se analizó una matriz de confusión (figura 10), mostrando que las etiquetas que el modelo con 10 errores o más eran : o: 24, 0: 22, I: 22,c: 20, 1: 19, s: 19, p: 15, v: 14, O: 13, l: 11, r: 11, 9: 10, V: 10, q: 10, z: 10.
Para buscar reducir o minimizar estos errores se realizo un aumento de datos y se volvió a probar el modelo, dando como resultado un una mayor precisión y una reducción de las clases con una cantidad de errores mayor o igual a 10, que quedaron así: 0: 29, o: 22, c: 18, v: 14, 1: 13, I: 13, O: 13, s: 13, l: 10.
Las clases con mayor cantidad de errores son la Clase 0, con 29 errores, posiblemente debido a su similitud visual con las letras "o" y "O". La Clase "c" presenta 18 errores, lo que indica dificultades para clasificarla correctamente. Las Clases "I" y "s" registran 13 errores cada una, sugiriendo confusión con caracteres similares. Por otro lado, varias clases no presentan errores, como las clases 4, 7, D, E, F, H, N, Q, R, T, h y u, lo que demuestra un buen desempeño del modelo en estas áreas. Además, algunas clases tienen solo 1 o 2 errores, indicando un funcionamiento razonablemente bueno. Sin embargo, persisten confusiones entre caracteres similares, como la Clase "o" y la Clase 0, y la Clase "I" y la Clase "l", debido a su similitud visual. Esta confusión podría mejorarse con más datos o técnicas avanzadas de aumento de datos para hacer estas clases más distinguibles.

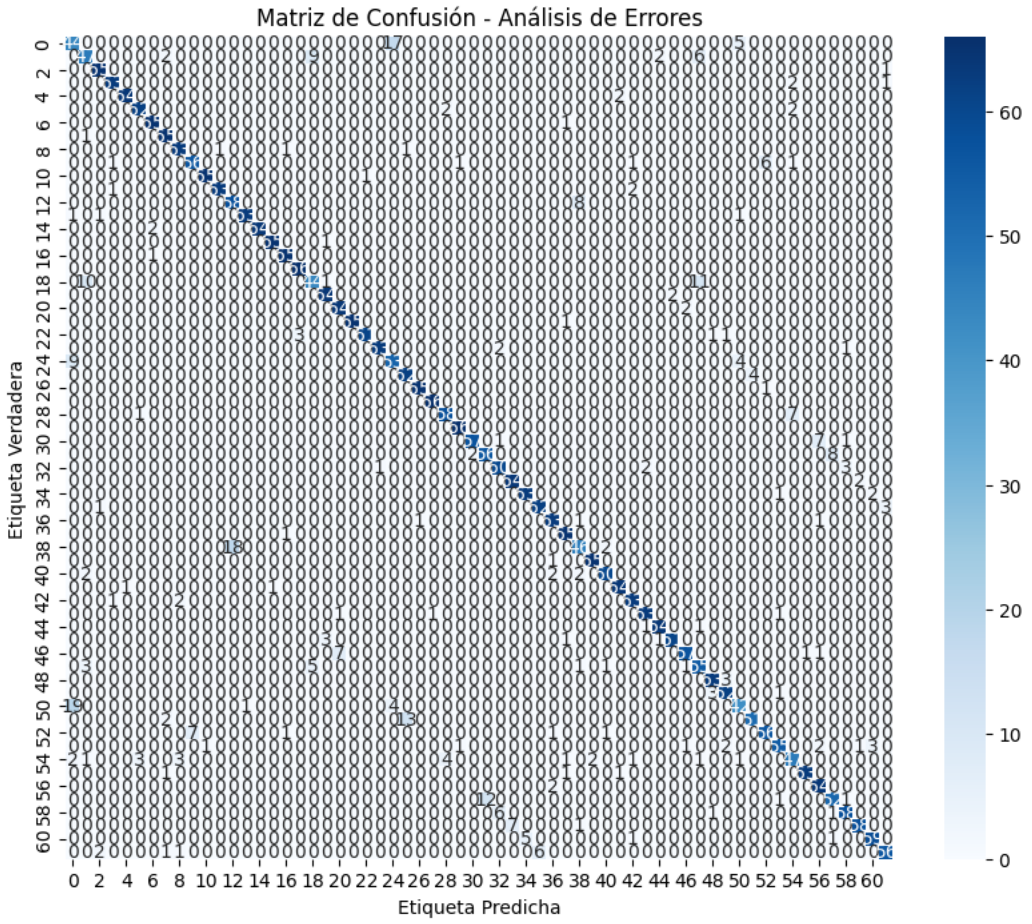

Figura 10. Matriz de Confusión

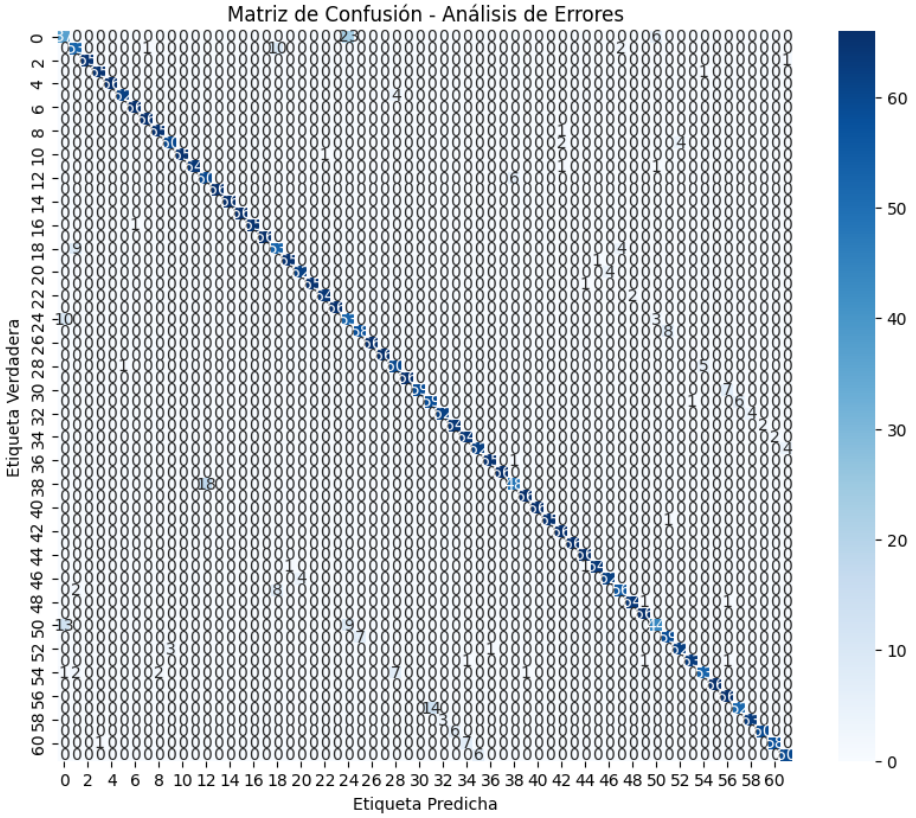

Figura 11. Matriz de Confusión tras entrenamiento con aumento de datos

Conclusión

La red neuronal convolucional (CNN) fue la que obtuvo el mejor rendimiento en la tarea de reconocimiento de caracteres escritos a mano. El uso de técnicas de optimización y análisis de errores permitió mejorar la precisión y reducir los errores de clasificación. Como trabajo futuro, se plantea entrenar las redes convolucionales con más datos, utilizar otros algoritmos como ViT, y emplear el algoritmo como parte de un programa más grande que permita reconocer palabras escritas a mano y, posteriormente, textos completos, utilizando técnicas como detección de objetos.

Ejecución del plan

Semana 1:

1. Configuración del entorno de trabajo

Semana 2:

2. Recolección y preprocesamiento de datos
3. Análisis exploratorio y preprocesamiento de datos
4. Implementación y optimización del modelo SVC
5.

4. Evaluación del modelo SVM optimizado
5.

Semana 3:

5. Evaluación del modelo SVM optimizado
6. Implementación del modelo Random Forest
7. Optimización y evaluación del modelo Random Forest

Semana 4:

8 . Entrenamiento y evaluación inicial de la red neuronal
9. Re-entrenar los modelos (SVC y Random Forest) buscando hiperparametros con optuna

Semana 5:

10. Optimización de hiperparámetros para la CNN
11. Evaluación final de la CNN optimizada

Semana 6:

12. Guardar los modelos usando mlflow
13. Comparación de todos los modelos implementados

Semana 7:

14. Análisis detallado de errores
15. Documentación
16. Revisión final

Implicaciones éticas

El algoritmo podría ser parte de programas más grandes que podrían traer riesgos en cuanto a la privacidad y seguridad de datos, esobre la protección de documentos sensibles, como registros médicos o financieros, ya que errores en el reconocimiento podrían llevar a brechas de privacidad o mal manejo de información confidencial. Además, el impacto ambiental, ya que aunque podría reducir el uso de papel al facilitar la digitalización, también podría aumentar el consumo de energía para el procesamiento computacional, especialmente si se requiere mantenimiento y re-entrenamiento constante de sistemas poco precisos.

Aspectos legales y comerciales

El algoritmo se puede usar como parte de un proceso más grande que permita la digitalización de documentos, es ideal para empresas que manejan grandes volúmenes de documentos escritos a mano, como bancos, aseguradoras y servicios de salud, permitiendo la automatización de procesos de archivo y gestión documental. En el ámbito educativo, como una herramienta de evaluación automática para profesores, facilitando su labor diaria. Además, como procesamiento de formularios para gobiernos y organizaciones que manejan formularios escritos a mano, como censos, encuestas y votaciones.

Referencias

[1] T. E. de Campos, B. R. Babu y M. Varma, "Character recognition in natural images," en Proceedings of the International Conference on Computer Vision Theory and Applications, Lisboa, Portugal, feb. 2009.