Recibido 18 OCT. 2023 *ReCIBE, Año 12 No.2, NOV. 2023*

Aceptado 02 ENE. 2024

Estudio comparativo de algoritmos para el reconocimiento de dígitos manuscritos. Caso: MNIST

Comparative Study of Algorithms for Handwritten Digit Recognition. Case: MNIST

Yasiel Conde Bernal[1]

Dr. Saul Lazcano Salas[1]

Maricela Quintana López[1]

Saturnino Job Morales Escobar[1]

Asdrúbal López Chau[1]

[1]Maestría en Ciencias de la Computación, Universidad Autónoma del Estado de México

RESUMEN

El aprendizaje de máquinas dispone de varios algoritmos que ofrecen buenos resultados de exactitud en labores de clasificación de texto; no obstante, existen otros factores como el consumo de recursos que pueden influir o determinar en la elección de un algoritmo sobre otro. Por lo anterior, el presente trabajo ofrece una comparación entre varios algoritmos de clasificación como son k vecinos más cercanos, regresión logística, Nayve Bayes, máquina de soporte de vectores, árbol de decisión, bosque aleatorio, perceptrón multicapa y red neuronal convolucional, que toma en cuenta varios aspectos que permiten determinar cuál algoritmo ofrece el mejor balance entre el consumo de tiempo y almacenamiento garantizando a su vez altos valores de la métrica de exactitud. Para el estudio se empleó el conjunto de datos MNIST correspondientes a los dígitos manuscritos del cero al nueve y se emplearon técnicas de búsqueda exhaustiva de hiperparámetros, cotejando los resultados de las métricas de clasificación exactitud, precisión, exhaustividad y F1-score, así como el consumo de recursos de tiempo y memoria.

Palabras claves: MNIST, redes neuronales, reconocimiento de patrones, métricas de clasificación, algoritmos de clasificación

ABSTRACT

Machine learning has several algorithms that offer good accuracy results in text classification tasks; however, there are other factors such as resource consumption that can influence or determine the choice of one algorithm over another. Therefore, the present work offers a comparison between several classification algorithms such as k nearest neighbors, logistic regression, Nayve Bayes, support vector machine, decision tree, random forest, multilayer perceptron and convolutional neural network, which takes into account several aspects that allow us to determine which algorithm offers the best balance between time and storage consumption, while guaranteeing high values of the accuracy metric. For the study, the MNIST data set corresponding to the handwritten digits from zero to nine was used and exhaustive hyperparameter search techniques were used, comparing the results of the classification metrics accuracy, precision, recall and F1-score, as well as the consumption of time and memory resources.

Keywords: MNIST, neural networks, pattern recognition, classification metrics, classification algorithms

1. INTRODUCCIÓN

El reconocimiento de patrones (RP) es un fenómeno que se da de forma natural en seres vivos, especialmente en humanos, que desde que nacen ya comienzan a reconocer y a apreciar el entorno alrededor, a través de los sentidos: el tacto, el olfato, el oído, el gusto y la vista. De forma general, se puede decir que el reconocimiento de patrones es el mecanismo con el que se cuenta para distinguir unas cosas de otras, relacionar cosas semejantes, formar grupos de cosas, tomar y explicar decisiones, describir objetos, entre otras.

En un sentido más formal, se puede definir al reconocimiento de patrones como “la ciencia que se ocupa de los procesos sobre ingeniería, computación y matemáticas relacionados con objetos físicos y/o abstractos, con el propósito de extraer información que permita establecer propiedades de o entre conjuntos de dichos objetos” (Ruiz-Shulcloper et al., 1999). Durante las últimas décadas, se han reportado interesantes resultados en el reconocimiento de patrones, entre los métodos tradicionales más exitosos se encuentran los de enfoque estadístico, como las reglas de decisión de Bayes paramétricas y no paramétricas, las máquinas de vectores de soporte, los algoritmos de refuerzo, entre otros (Zhang et al., 2020).

Las aplicaciones de aprendizaje automático en campos como la visión por computadora, la robótica, el reconocimiento de voz y el procesamiento del lenguaje natural, generalmente tienen como objetivo emular y acercarse lo más posible al desempeño humano (Braga-Neto, 2020). Por su parte las redes neuronales artificiales (RNA) también han jugado un papel importante en el desarrollo de la inteligencia computacional. Las RNA son cada vez más atractivas, efectivas, eficientes y exitosas para lograr el reconocimiento de patrones (RP) en muchos problemas.

El presente trabajo tiene como objetivo, realizar un estudio comparativo entre diferentes algoritmos empleados para el reconocimiento de patrones en imágenes con caracteres manuscritos, así como el ajuste de los parámetros de cada uno para obtener las mejores métricas de clasificación con base en los hiperparámetros seleccionados, empleando para ello la base de datos MNIST. Se manejarán las métricas de exactitud, precisión, exhaustividad y f1-score, además del lenguaje de programación Python y las librerías keras, tensorflow, sklearn, numpy, pandas y matplotlib para la implementación de los algoritmos.

Los algoritmos que se analizarán en el estudio comparativo serán K-vecinos más cercanos (KNN por sus siglas en inglés), Naive Bayes (NB), regresión logística (LR), árboles de decisión (DT), bosque aleatorio (RF), perceptrón multicapa (MLP) y red neuronal convolucional (CNN). Para la identificación de los parámetros se utilizará la técnica de grid search que los encuentra buscando exhaustivamente a través de un subconjunto especificado manualmente del espacio de hiperparámetros del algoritmo de destino.

Los resultados sentarán las bases para el desarrollo de una aplicación móvil que potencie las habilidades comunicativas de las personas sordociegas mediante el reconocimiento de símbolos manuscritos específicos de la comunidad, permitiendo a estos individuos comunicar necesidades básicas. Cabe resaltar que en esta población se dificulta la comunicación (aunque no se imposibilitan), puesto que, siendo personas ciegas o débiles visuales, son también sordas o débiles auditivas, requiriendo de apoyo para realizar actividades que parecieran sencillas a un individuo sin la discapacidad como por ejemplo buscar información, dialogar, interactuar con el entorno o desplazarse justo como el resto de las personas y este tipo de apoyo suele variar de un sordociego a otro. Por lo que el presente estudio aportará una guía para la elección de un algoritmo de reconocimiento de patrones y parámetros optimizados que permita obtener un nivel de exactitud en la clasificación superior al 97% y una alta precisión por símbolo, manteniendo de manera simultánea el menor consumo de memoria y tiempo de ejecución posible, con el fin de poder ser utilizada en dispositivos móviles.

2. DESARROLLO

La clasificación de textos es una tarea de fundamental importancia en diversas actividades y ha ido ganando terreno gracias a los desarrollos recientes en los campos de la minería de textos y el procesamiento del lenguaje natural. La velocidad a la que se crea actualmente la información textual ha superado por mucho las soluciones manuales para estas tareas, lo que significa que esos métodos no solo son útiles, sino también estrictamente necesarios (Gasparetto et al., 2022).

En el presente trabajo, se emplea el conjunto de datos modificado del Instituto Nacional de Estándares y Tecnología (MNIST), propuesto en 1994 como una combinación de dos de las bases de datos del NIST: la especial 1 y la especial 3, constituidas estas 2 por dígitos escritos por estudiantes de secundaria y empleados de la Oficina del Censo de los Estados Unidos, respectivamente (Yann, 1998). Se tomó la decisión de emplear este conjunto de datos debido a que es muy empleado en el reconocimiento de caracteres (Baldominos Gómez et al., 2019), llegando a ser llamado el “Hello World” del Machine Learning (Merenda et al., 2020). Otro punto a favor de la elección de este conjunto de datos es que es de acceso público, aspecto que beneficia su amplio reconocimiento dentro de la comunidad científica y que lo posiciona como un punto de referencia ideal a la hora de analizar algoritmos de clasificación de imágenes y texto. Las características de las imágenes de MNIST son de igual forma similares a las que se emplearán en la aplicación móvil para la comunidad sordociega, ya que ambas poseen igual dimensión, se encuentran en escala de grises y se corresponden con caracteres manuscritos.

La base de datos MNIST está compuesta por un total de 70 000 imágenes en formato de 28x28 pixeles en escala de grises y presenta una versión dividida en 60 000 datos para la etapa de entrenamiento y 10 000 para la de pruebas, subconjuntos estos que serán empleados en el presente trabajo, la descripción resumen se muestra en la tabla 1. La base de datos cuenta con un total de 10 clases que se corresponden con los dígitos del 0 al 9, en la figura 1 se muestra la distribución por clases en el conjunto de entrenamiento.

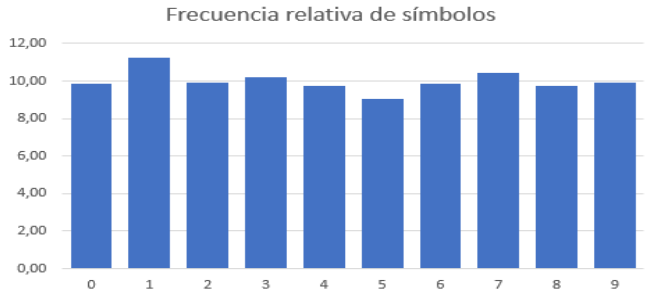

Figura 1. Frecuencia relativa de dígitos del conjunto de entrenamiento

Las clases se encuentran aceptablemente balanceadas, sobrepasando la media los símbolos 1, 3 y 7. En el caso del conjunto de prueba, la figura 2 muestra la frecuencia relativa por cada dígito.

Figura 2. Frecuencia relativa de los datos del conjunto de prueba

La proporción de clases en el conjunto de pruebas se encuentra balanceada en lo posible; no obstante, superan la media los dígitos 1, 2, 3, 7 y 9. El conjunto de entrenamiento está compuesto por aproximadamente el 85,71% de los registros y el de prueba por el 14,29%.

Conjunto de datos	**Objetos**	**Tamaño (pixeles)**
x_train	60 000	28*28 = 784
x_test	10 000	28*28 = 784
Total	70 000	

Tabla 1. Estructura de los conjuntos de entrenamiento y prueba

2.1 ALGORITMOS DE CLASIFICACIÓN

Para el aprendizaje automático se han desarrollado varios tipos de algoritmos entre los cuales se encuentran: MLP (Multilayer Perceptron), K-NN (K-Nearest Neighbor), NB (Naive Bayes), SVM (Support Vector Machines), DT (Decision Tree), entre otros (Llumiquinga Almeida, 2022). En la presente investigación se opta por el uso de los antes mencionados por ubicarse entre los más usados para la clasificación de textos (Gasparetto et al., 2022) y por presentar elevado desempeño en tareas de reconocimiento de patrones (Sen et al., 2020). Se incluye además en el comparativo la red neuronal convolucional (CNN por sus siglas en inglés) ya que durante la última época se ha destacado notablemente su empleo debido al elevado rendimiento que demuestran en tareas de clasificación de imágenes (Li et al., 2021). A continuación, se analizarán brevemente cada uno de los algoritmos citados.

2.1.1 K-Vecinos más cercanos (KNN)

Es un algoritmo de clasificación supervisado no paramétrico, que mediante el cálculo de distancias permite identificar a los vecinos más cercanos de un punto de consulta dado, de modo que se pueda asignar una etiqueta de clase a ese punto (Lévano-Rodriguez & Cerdán-León, 2022). KNN integra al conjunto de algoritmos de “aprendizaje perezoso”, esto significa que almacena solamente el conjunto de datos de entrenamiento sin pasar realmente por una fase de entrenamiento (Ehsani & Drabløs, 2020). Esto deja un gran trabajo a la etapa de predicción o clasificación, al realizar el cálculo durante esta etapa. El algoritmo emplea para su funcionamiento un conjunto de métricas de distancia como las que se listan a continuación:

- Distancia euclidiana
- Distancia de Minkowski
- Distancia de Manhattan
- Distancia de Chebyshev
- Similitud de coseno
- Coeficiente de Jaccard
- Distancia de Hamming

KNN es fácil de implementar y posee pocos hiperparámetros lo que constituye una ventaja, sin embargo, es costoso computacionalmente y lento, además sufre de la maldición de la dimensionalidad, uno de los principales problemas a tener en cuenta en la fase de clasificación, debido a que demanda una mayor cantidad de datos para entrenar el modelo en relación con la dimensión del vector de características. (Naranjo, 2022) (no está clara la idea)

2.1.2 Naïve Bayes

Es otro método de aprendizaje automático empleado para la resolución de problemas de clasificación. Este algoritmo es de tipo probabilístico y sienta sus bases en el teorema de Bayes. Asume que los predictores en un modelo Naïve Bayes son condicionalmente independientes o no están relacionados con ninguna otra característica del modelo. También supone que todas las características contribuyen por igual al resultado. El algoritmo funciona bien con tipos de datos heterogéneos y también con valores perdidos, debido al tratamiento independiente de cada variable predictiva para la construcción del modelo (Griffis et al., 2016).

Este método utiliza una variable score para apoyar el proceso de clasificación, en lugar de utilizar sólo el método de la probabilidad (Gutiérrez Esparza et al., 2017). Es un algoritmo sencillo de implementar y con parámetros fáciles de estimar, pero se afecta a frecuencia cero: la frecuencia cero ocurre cuando una variable categórica no existe dentro del conjunto de entrenamiento, y al estar basado en probabilidades, sufre cuando aparecen casos raros.

2.1.3 Regresión logística (LR)

La regresión logística es un instrumento estadístico empleado para el análisis de datos. Su origen se remonta a la década de los sesenta (Confield, Gordon y Smith 1961). LR se utiliza para describir datos y explicar la relación entre una variable binaria dependiente y una o más variables independientes nominales, ordinales, de intervalo o de relación. Los modelos LR generalmente incluyen solo las variables que se consideran "importantes" para predecir un resultado, normalmente, la predicción ofrece un número finito de resultados (Kirişci, 2019).

2.1.4 Máquina de soporte vectorial (SVM)

La máquina de soporte vectorial es un método popular de aprendizaje automático para clasificación, regresión y otras tareas de aprendizaje (Luu et al., 2020). Es una técnica de clasificación destacada, ampliamente utilizada desde sus inicios. Fue introducida por primera vez por Cortés y Vapnik en 1995 para problemas de clasificación binaria. SVM busca encontrar hiperplanos que determinen el límite de decisión que permita clasificar los puntos de datos en dos clases (Tanveer et al., 2022).

SVM, en su forma básica, es un clasificador binario lineal, que identifica un único límite entre dos clases. La SVM lineal asume que los datos multidimensionales son linealmente separables en el espacio de entrada. En particular, las SVM determinan un hiperplano óptimo (una línea en el caso más simple) para separar el conjunto de datos en un número discreto de clases predefinidas usando los datos de entrenamiento (Sheykhmousa et al., 2020).

2.1.5 Árboles de Decisión (DT)

DT es un enfoque de aprendizaje automático supervisado para resolver problemas de clasificación y regresión mediante la división continua de datos en función de un determinado parámetro. Las decisiones están en las hojas y los datos se dividen en los nodos. En el árbol de clasificación, la variable de decisión es categórica (resultado en forma de Sí/No) y en el Árbol de regresión, la variable de decisión es continua. DT tiene las siguientes ventajas: es adecuado para problemas de regresión y clasificación, es de fácil interpretación, presenta facilidad de manejo de valores categóricos y cuantitativos, es capaz de rellenar los valores faltantes en los atributos con el valor más probable y posee alto rendimiento debido a la eficiencia del árbol de algoritmo transversal (Ray, 2019).

2.1.6 Bosque aleatorio (RF)

Modelo propuesto por Kam Ho (1995) de Laboratorios Bell, y posteriormente desarrollado por Breiman (2001) quien presentó el modelo totalmente desarrollado. RF es una combinación de árboles predictores tal que cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de estos, empleando varios modelos para resolver un único problema y combinando sus resultados para producir una solución precisa (Alarcón Flores, 2017).

Sorprendentemente, la combinación de muchos árboles de decisión basados en datos muestreados aleatoriamente funciona bien en la práctica porque la aleatoriedad ayuda a que el modelo evite el ajuste excesivo de los datos; sin embargo, su velocidad de predicción puede ser lenta si contiene una gran cantidad de árboles, afectando el rendimiento del modelo, ya que este depende del parámetro que representa la profundidad de los árboles aleatorios en el mismo (Cho et al., 2019).

2.1.7 Perceptrón multicapa (MLP)

Los perceptrones multicapa forman un tipo de red neuronal, compuestos por un sistema de neuronas o nodos simples interconectados, o sea, un modelo que representa un mapeo no lineal entre un vector de entrada y un vector de salida. Los nodos están conectados por pesos y señales de salida que son una función de la suma de las entradas al nodo modificadas por una simple función no lineal de transferencia o activación (Gardner & Dorling, 1998).

2.1.8 Red neuronal convolucional (CNN)

Las redes neuronales convolucionales (CNN) fueron introducidas por primera vez por Fukushima en 1998. Están formadas por neuronas, donde cada neurona tiene un peso y un sesgo que se pueden aprender. Contiene una capa de entrada, una capa de salida y múltiples capas ocultas, donde la capa oculta consta de una capa convolucional, una capa de agrupación, una capa totalmente conectada (FC) y varias capas de normalización.

La capa convolucional aplica una operación de convolución para fusionar dos conjuntos de información. Imita la retroalimentación de una neurona individual a los estímulos visuales. La capa de agrupación se utiliza para reducir la dimensionalidad, asociando la salida del grupo de neuronas en una capa con la neurona individual. La capa FC conecta cada neurona de una capa con cada neurona de otra capa. Su propósito principal es clasificar las imágenes de entrada en varias clases, según los conjuntos de datos de entrenamiento (Dhillon & Verma, 2020).

CNN ha estado alcanzado grandes logros y se ha convertido en una de las redes neuronales más representativas en el campo del aprendizaje profundo. La visión artificial basada en CNN ha permitido a las personas lograr lo que se consideraba imposible en los últimos siglos, como el reconocimiento facial, los vehículos autónomos, los supermercados de autoservicio y los tratamientos médicos inteligentes (Li et al., 2021).

2.2 Métricas de clasificación

Las métricas de clasificación son útiles para evaluar y comparar diferentes modelos de clasificación o técnicas de aprendizaje automático, permiten comparar su rendimiento y analizar su comportamiento ajustando diferentes parámetros. Algunas métricas se basan en los resultados de la matriz de confusión (Grandini et al., 2020) ya que estos permiten medir el rendimiento de un algoritmo de aprendizaje automático (Martínez-Toro et al., 2019). Las métricas de evaluación se emplean además para medir y resumir la calidad del clasificador entrenado cuando se prueba con los datos no vistos. En el presente trabajo se utilizan las métricas listadas a continuación para comparar el rendimiento de cada uno de los modelos propuestos.

- Exactitud (Accuracy): es la suma de las predicciones verdaderas, tanto positivas como negativas, dividida por el número total de predicciones hechas. Permite saber qué tan exacto es el resultado y ofrece información sobre los posibles errores que se pueden encontrar en la clasificación (Hernández et al., 2022).
- Precisión (Precision): es la fracción de elementos positivos verdaderos dividida por el número total de unidades predichas como positivas. Expresa la proporción de unidades que el modelo predice que son positivas y en realidad lo son, o sea, esta métrica dice cuánto se puede confiar en el modelo cuando predice que un individuo es positivo. En otras palabras, dentro de todas las predicciones positivas del clasificador, qué porcentaje de estos en realidad son positivos (Llumiquinga Almeida, 2022).
- Exhaustividad (Recall): se emplea para medir la fracción de elementos positivos clasificados correctamente, mide la capacidad del modelo para encontrar todas las unidades positivas en el conjunto de datos; en otras palabras, mide la proporción de verdaderos positivos frente a falsos negativos.
- F1-score: representa la media armónica de precisión y exhaustividad, que proporciona una medida equilibrada del rendimiento general de un clasificador. Por ejemplo, si la precisión es del 80% y la recuperación es del 70%, la puntuación F1 es del 75% (Kalmegh & Padar, 2023). Esta es una métrica muy utilizada en problemas en los que el conjunto de datos a analizar está desbalanceado.

Se seleccionaron las métricas listadas porque permiten medir de manera imparcial y rigurosa el desempeño de los algoritmos de clasificación. Las métricas seleccionadas en el trabajo resultan útiles para probar la capacidad de un clasificador de clases múltiples, permitiendo comparar el rendimiento de modelos diferentes o analizar el comportamiento del mismo modelo ajustando diferentes parámetros (Grandini et al., 2020), además son frecuentemente utilizadas por investigadores, especialmente en conjuntos de datos con clases balanceadas (Borja-Robalino et al., 2020).

2.3 Hiperparámetros de algoritmos propuestos

El ajuste de hiperparámetros de un algoritmo de aprendizaje automático (ML) es una característica del proceso de aprendizaje del modelo. Los hiperparámetros suelen ser de varios tipos, discretos, categóricos y continuos, por lo que el mecanismo de ajuste es diferente entre los algoritmos de ML. Grid search es un estándar de optimización de parámetros existente en el aprendizaje automático, se basa en una búsqueda exhaustiva a través de un subconjunto, definido manualmente, del espacio de hiperparámetros del algoritmo de aprendizaje.

Existen otras propuestas, como la búsqueda aleatoria, la optimización bayesiana y la optimización de gradientes. Sin embargo, debido a su facilidad de ejecución y paralelización, su durabilidad en espacios de baja dimensión, grid search prevalece como el estado del arte para la optimización de hiperparámetros (Belete & Huchaiah, 2022). En la tabla 2 se muestra la configuración de hiperparámetros definida para cada algoritmo.

Algoritmo	Hiperparámetros
KNN	**n_neighbors:** [1, 3, 5, 7, 9 ,11] **metric:** [euclidean, manhattan, minkowski, chebyshev, cosine, jaccard, hamming] **weights:** [uniform, distance] **leaf_size:** [5, 10, 20, 30, 35, 40, 45, 50] **algorithm:** [brute, kd_tree, ball_tree]
NB	**var_smoothing:** np.logspace(0, -9, num=1000)
LR	**penalty:** [l1, l2, elasticnet] **C:** np.logspace(-3,3,60) **class_weight:** [balanced, None] **solver:** [lbfgs, liblinear, newton-cg, newton-cholesky, sag, saga]
SVM	**kernel:** [rbf, poly, sigmoid] **gamma:** [1e-1, 1e-2,1e-3, 1e-4,1e-5, 1e-6, scale, auto] **C:** [1, 10, 100, 1000] **decision_function_shape:** [ovo, ovr]
DT	**criterion:** [gini, entropy, log_loss] **splitter:** [best, random] **min_samples_split:** [2,3,4,5,6,7,8,16] **class_weight:** [balanced, None] **min_weight_fraction_leaf:** np.linspace(0, 0.5, 100)
RF	**n_estimators:** [10,20,30,40,50,60,70,80,100,200,500] **max_features:** [None, sqrt, log2] **max_depth:** [4,5,6,7,8] **criterion:** [gini, entropy] **min_samples_split:** [2,4,8,16,32] **class_weight:** [balanced, None]}
MLP	**Activation:** [relu, tanh, sigmoid, hard_sigmoid, linear] **optimizer:** [SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax, Nadam] **epochs:** [1, 10, 20, 30, 40, 50] **learning_rate:** [0.1, 0.0001, 0.01] **batch_size:** [10, 20, 50, 80, 100] **initializer:** [lecun_uniform, normal, he_normal, he_uniform] **dropout_rate:** [0.3, 0.2, 0.8] **num_unit:** [5, 10, 20]

Tabla 2. Configuración de hiperparámetros por algoritmo

Luego de ejecutado el algoritmo de grid search para encontrar los parámetros óptimos, según los hiperparámetros propuestos para cada algoritmo y la base de datos analizada se obtuvieron los resultados mostrados en la tabla 3.

Algoritmo	Hiperparámetros
KNN	**n_neighbors:** 1 **metric:** cosine **weights:** uniform **leaf_size:** 5 **algorithm:** brute
NB	**var_smoothing:** 0.009203731996618
LR	**penalty:** l1 **C:** 0.2758531617629184 **class_weight:** balanced **solver:** liblinear
SVM	**kernel:** rbf **gamma:** scale **C:** 100 **decision_function_shape:** ovo
DT	**criterion:** log_loss **splitter:** best **min_samples_split:** 3 **class_weight:** None **min_weight_fraction_leaf:** 0
RF	**class_weight:** None **criterion:** entropy **max_depth:** 8 **max_features:** log2 **min_samples_split:** 2 **n_estimators:** 500
MLP	**Activation:** relu **Optimizers:** SGD **Epochs:** 1 **Learning Rate**:0.1 **Batch Size:**20 **Initializers:** lecun_uniform **Dropout Rate**:0.3 **Number of Units:**10

Tabla 3. Mejores parámetros por algoritmo según hiperparámetros especificados

Posteriormente se procedió con la implementación de cada uno para comparar sus resultados con lo obtenidos con una red neuronal convolucional que será diseñada e implementada de igual forma en la presente investigación y con bases en la propuesta de arquitectura de red neuronal convolucional de (Wang et al., 2020)

2.4 Entrenamiento y prueba de algoritmos

Para el desarrollo de la aplicación se empleó una laptop con 8 gigabytes de RAM, un procesador de 64 bits Intel Core I7 2620M de segunda generación y un disco de estado sólido de 256 gigabytes. El sistema operativo empleado fue Microsoft Windows 11 de 64 bits, el entorno de programación seleccionado fue Spyder y la versión de Python fue 3.10.9.

2.4.1 K-Vecino más cercano

La exactitud del algoritmo KNN para el conjunto de entrenamiento fue de 1.00 con un tiempo de 0.61 segundos. Por su parte en la predicción del conjunto de pruebas el resultado de la exactitud fue de 0.97 (valor este un poco inferior al del entrenamiento) y la operación tardó un tiempo de 42.27 segundos. La matriz de confusión, mostrada en la figura 3 ofrece resultados de clasificación por símbolos que se emplearon para calcular las métricas de rendimiento.

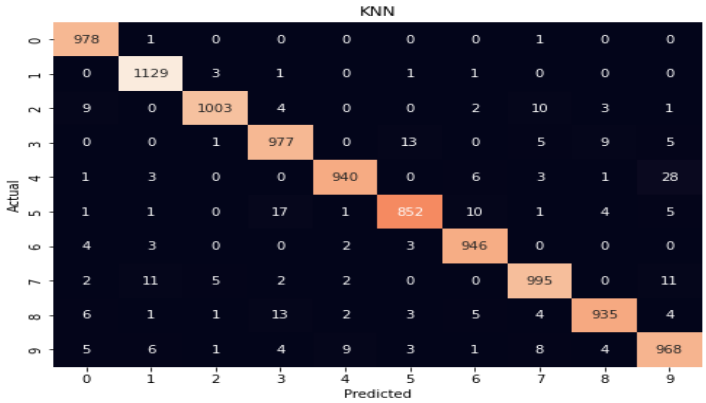

Figura 3. Matriz de confusión obtenida del conjunto de prueba mediante el algoritmo KNN

Como se puede apreciar en la figura el símbolo que más veces se clasificó incorrectamente fue el 4, que se clasificó como 9 en 28 oportunidades, seguido del 5 que se clasificó como 3 en 17 ocasiones. Los símbolos con mejores resultados de clasificación fueron el 0 y el 1. El dígito 2 obtuvo la mayor precisión, lo que indica que de los elementos clasificados como 2, el 99% realmente lo era; sin embargo, su métrica recall obtuvo valor inferior, esto sugiere que un 3% de los dígitos 2 fueron mal clasificados y un 97% bien clasificados. La tabla 4 ofrece un resumen de las métricas de clasificación calculadas a partir de la matriz de confusión, los valores fueron redondeados por exceso dejando dos lugares decimales.

Símbolo	Precisión	Recall	F1-Score
0	0.97	1.00	0.98
1	0.98	0.99	0.99
2	0.99	0.97	0.98
3	0.96	0.97	0.96
4	0.98	0.96	0.97
5	0.97	0.96	0.96
6	0.97	0.99	0.98
7	0.97	0.97	0.97
8	0.98	0.96	0.97
9	0.95	0.96	0.95

Tabla 4. Resumen de métricas para algoritmo KNN

2.4.2 Naïve Bayes (NB)

Luego de implementar el algoritmo NB y entrenarlo con ajuste óptimo obtenido según hiperparámetros definidos, el algoritmo ofreció una exactitud del 0.78, en un tiempo de 0.9754 segundos. La predicción de todo el conjunto de datos ofreció una exactitud de 0.79, ligeramente superior al entrenamiento, requiriendo para ello de un tiempo de 0.8727 segundos. En la figura 4 se puede apreciar la matriz de confusión.

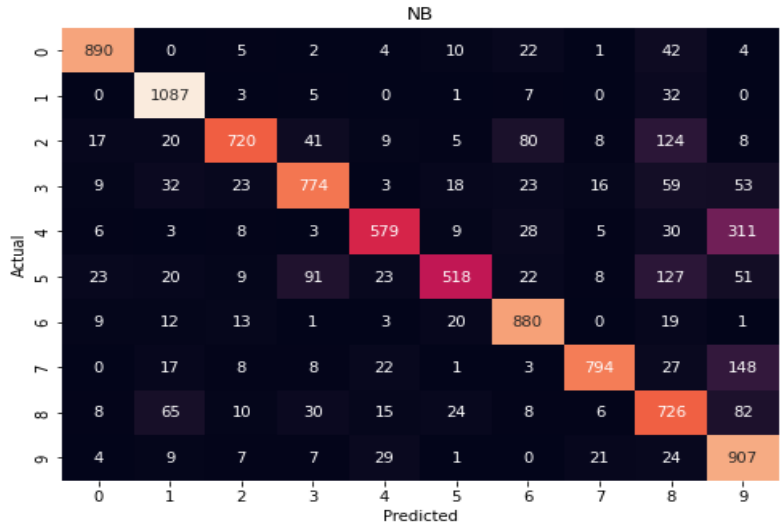

Figura 4. Matriz de confusión obtenida del conjunto de prueba mediante el algoritmo NB

El símbolo con mayor problema resultó ser el 5, que fue mal clasificado en 374 ocasiones, 127 veces de estas fue clasificado incorrectamente como 8. El símbolo con mejores resultados fueron el 0 y el 1, cabe resaltar que este para tampoco fue confundido entre sí en ningún momento. La tabla 5 ofrece el resultado detallado de métricas por símbolo.

Símbolo	Precisión	Recall	F1-Score
0	0.92	0.91	0.91
1	0.86	0.96	0.91
2	0.89	0.70	0.78
3	0.80	0.77	0.78
4	0.84	0.59	0.69
5	0.85	0.58	0.69
6	0.82	0.92	0.87
7	0.92	0.77	0.84
8	0.60	0.75	0.66
9	0.58	0.90	0.70

Tabla 5. Resumen de métricas para algoritmo NB

1.4.3 Regresión Logística

El algoritmo ofreció una exactitud de 0.92 durante el entrenamiento, consumiendo un tiempo de 39.5372 segundos. La predicción de todo el conjunto de pruebas le tomó un tiempo de 0.0420 segundos y ofreció una exactitud de 0.92 exactamente igual al entrenamiento. La matriz de confusión se muestra en la figura 5.

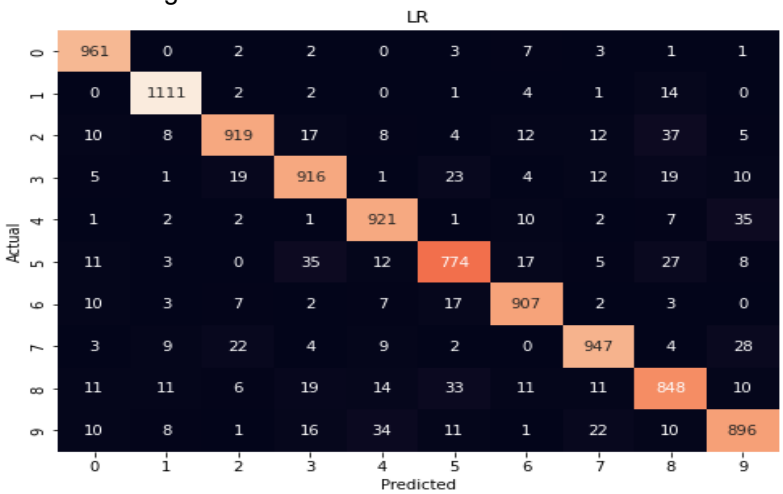

Figura 5. Matriz de confusión obtenida del conjunto de prueba mediante el algoritmo LR

El algoritmo ofreció una tasa aceptable de reconocimiento tanto durante el entrenamiento como durante la prueba. Los símbolos 5, 8, 9 y 2 ofrecieron la tasa más baja de reconocimiento correcto. La tabla 6 muestra un resumen de las métricas obtenidas de la matriz de confusión. Los símbolos mejores resultados de clasificación resultaron ser el 0 y el 1.

Símbolo	Precisión	Recall	F1-Score
0	0.94	0.98	0.96
1	0.96	0.98	0.97
2	0.94	0.89	0.91
3	0.90	0.91	0.91
4	0.92	0.94	0.93
5	0.89	0.87	0.88
6	0.93	0.95	0.94
7	0.93	0.92	0.93
8	0.87	0.87	0.87
9	0.90	0.89	0.90

Tabla 6. Resumen de métricas para algoritmo LR

1.4.4 Máquina de soporte vectorial

Este algoritmo ofreció una exactitud al conjunto de entrenamiento de 1.00 y la ejecución de toda la instrucción de entrenamiento al conjunto se tomó un tiempo de 217.6239 segundos. La predicción del conjunto de prueba tardó 96.3754 segundos y arrojó una exactitud de 0.98. La figura 6 muestra la matriz de confusión obtenida de la predicción de todo el conjunto de prueba. Los resultados de este algoritmo resultaron excelentes tanto para el conjunto de entrenamiento como para el de prueba, sin embargo, el tiempo requerido para realizar ambas labores fue muy alto. Las precisiones más bajas las obtuvieron los dígitos 5 y 8, con valores inferiores a 0,9. Los símbolos mejor clasificados fueron el 0 y el 1.

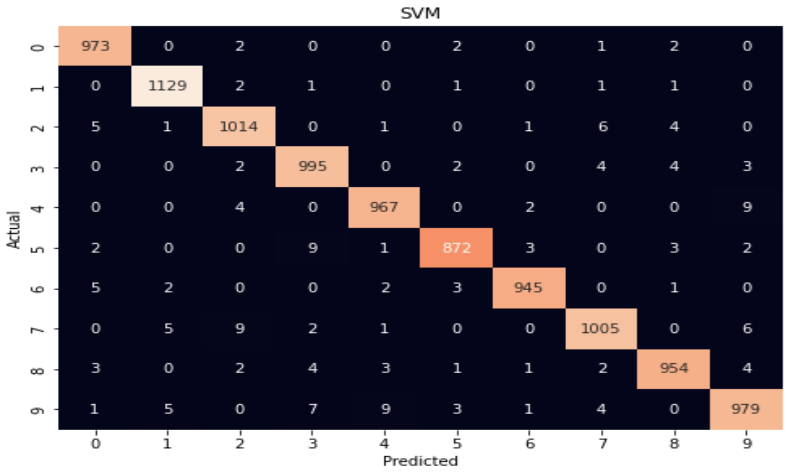

Figura 6. Matriz de confusión obtenida del conjunto de prueba mediante el algoritmo SVM

La tabla 7 muestra el resumen de las métricas de rendimiento de clasificación calculados a partir de la matriz de confusión.

Símbolo	Precisión	Recall	F1-Score
0	0.94	0.98	0.96
1	0.96	0.98	0.97
2	0.94	0.89	0.91
3	0.90	0.91	0.91
4	0.92	0.94	0.93
5	0.89	0.87	0.88
6	0.93	0.95	0.94
7	0.93	0.92	0.93
8	0.87	0.87	0.87
9	0.90	0.89	0.90

Tabla 7. Resumen de métricas para algoritmo SVM

1.4.5 Árbol de clasificación (DT)

El algoritmo ofreció una exactitud de 1.00 para el conjunto de entrenamiento, tardando en acometer la tarea, un tiempo de 21.97288 segundos. Para la predicción del conjunto de prueba, la exactitud descendió notablemente hasta 0.88 en un tiempo de 0.0270 segundos. La figura 7 muestra la matriz de confusión obtenida de la predicción del conjunto de prueba.

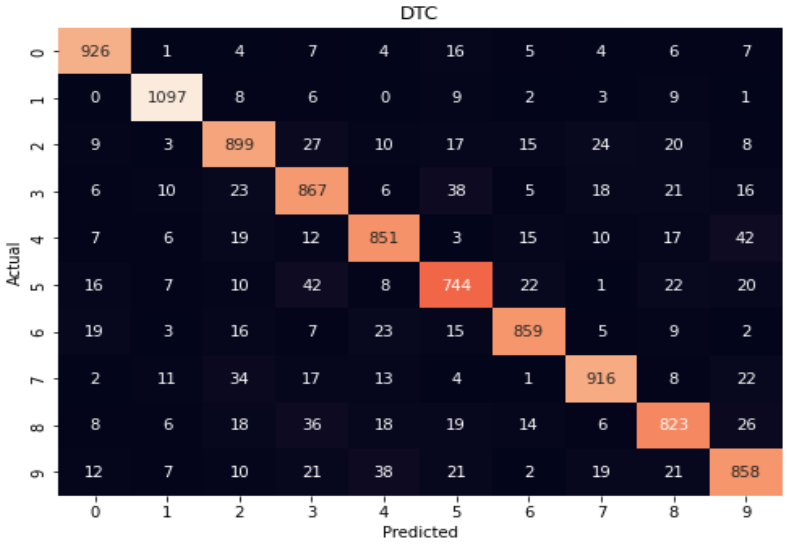

Figura 7. Matriz de confusión obtenida del conjunto de prueba mediante el algoritmo SVM

Nuevamente los símbolos mejor clasificados resultaron ser el 0 y el 1 y los que más problemas causaron durante la clasificación fueron el 5 y el 8. La tabla 8 ofrece los resultados de las métricas por cada símbolo.

Símbolo	Precisión	Recall	F1-Score
0	0.94	0.98	0.96
1	0.96	0.98	0.97
2	0.94	0.89	0.91
3	0.90	0.91	0.91
4	0.92	0.94	0.93
5	0.89	0.87	0.88
6	0.93	0.95	0.94
7	0.93	0.92	0.93
8	0.87	0.87	0.87
9	0.90	0.89	0.90

Tabla 8. Resumen de métricas para algoritmo DT

1.4.6 Bosque aleatorio (RF)

La exactitud para este algoritmo fue de 0.93 para el conjunto de entrenamiento, requiriendo para este proceso 14.8594 segundos. La predicción del conjunto de pruebas obtuvo una exactitud de 0.92 en un lapso de 0.2822 segundos. La figura 8 muestra la matriz de confusión ofrecida para la predicción del conjunto de pruebas por el algoritmo.

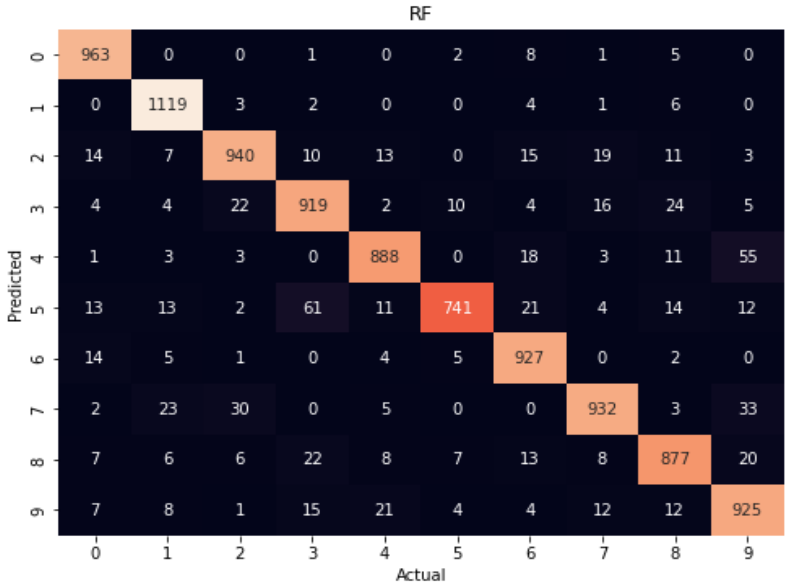

Figura 8. Matriz de confusión obtenida del conjunto de prueba mediante el algoritmo RF

Los símbolos que mayor confusión generaron durante el proceso de clasificación del conjunto de pruebas fueron el 3, 8, 9 y 5; por el contrario, los de mejores resultados fueron el 0 y el 1. Los tres símbolos más confundidos durante el proceso de clasificación fueron el 5 que clasificó como 3 en 61 ocasiones, el 4 que fue clasificado como 9 en 55 ocasiones y el 7 que fue clasificado como 9 en 33 ocasiones.

Símbolo	Precisión	Recall	F1-Score
0	0.95	0.98	0.96
1	0.93	0.99	0.96
2	0.93	0.91	0.92
3	0.89	0.92	0.90
4	0.93	0.91	0.92
5	0.96	0.87	0.91
6	0.93	0.96	0.94
7	0.93	0.90	0.92
8	0.91	0.88	0.90
9	0.88	0.91	0.90

Tabla 8. Resumen de métricas para algoritmo RF

1.4.7 Perceptrón Multicapa (MLP)

La estructura de la red neuronal multicapa es la mostrada en la figura 9, que utiliza un modelo secuencial. La capa de entrada cuenta con 512 neuronas y función de activación rectificación lineal. La segunda capa consta de 768 neuronas y función de activación rectificación lineal al igual que la capa precedente. La capa de salida consta de 10 neuronas y emplea función de activación softmax. La variable de las categorías y_train fue codificada mediante one hot encoding. La red neuronal propuesta cuenta con 803594 parámetros, todos estos entrenables. Se empleó el optimizador SGD, un total de 20 épocas y el tamaño del lote de validación seleccionado fue de 20.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_1 (Dense)             (None, 512)               401920

 dense_2 (Dense)             (None, 768)               393984

 dense_3 (Dense)             (None, 10)                7690

=================================================================
Total params: 803,594
Trainable params: 803,594
Non-trainable params: 0
```

Figura 9. Resumen del modelo de perceptrón multicapa

El algoritmo propuesto obtuvo una exactitud de 0.9946 durante el entrenamiento, el tiempo consumido durante este proceso fue de 434.8619 segundos, la validación por lote ofreció una exactitud del 0.9817. El proceso de prueba duró un tiempo de 1.9344 segundos y devolvió una exactitud de 0.98. La precisión del algoritmo fue muy buena para todos los dígitos obteniendo para cada uno un valor entre 0.97 y 0.99. El desempeño del modelo durante las 20 épocas del entrenamiento fue adecuado, la pérdida fue disminuyendo por etapas tanto para el conjunto de entrenamiento como para el de validación de lote, así como también la exactitud del algoritmo aumentó de manera progresiva en cada etapa. Para más información ver figura 10.

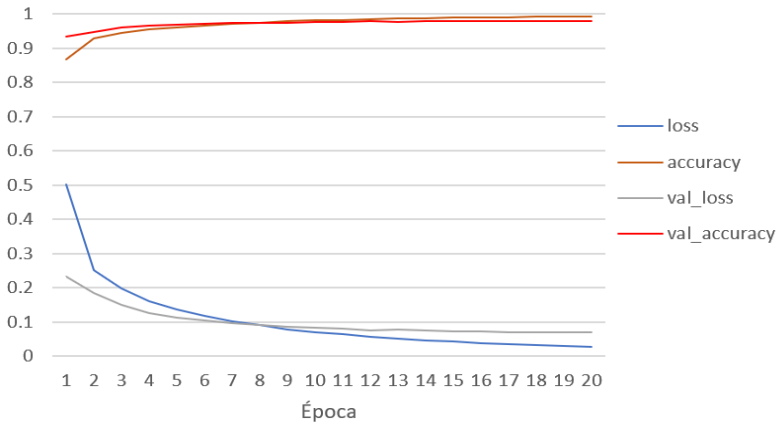

Figura 10. Gráfica de pérdida y exactitud por épocas

La figura 11 muestra la matriz de confusión obtenida para el conjunto de prueba. Por el eje vertical se pueden observar los valores reales y por el eje horizontal se encuentran los valores predichos por el algoritmo.

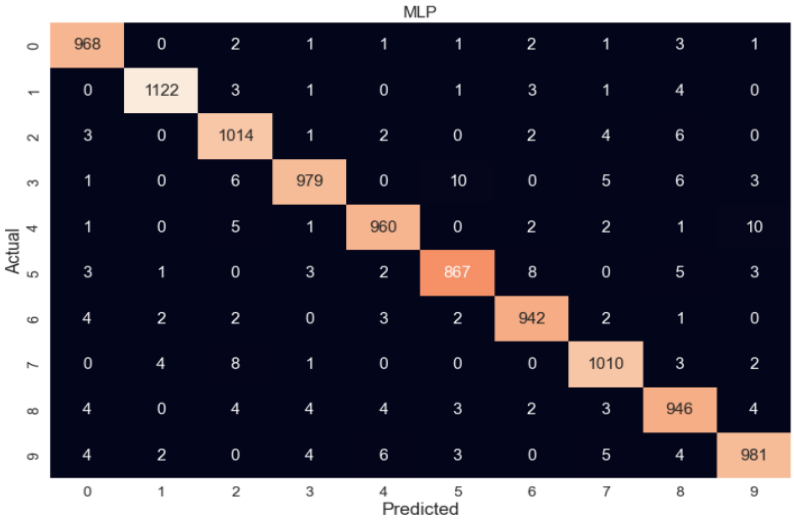

Figura 11. Matriz de confusión obtenida del conjunto de prueba mediante el algoritmo MLP

El símbolo con mejor tasa de reconocimiento fuer el 1 y el 0. Los símbolos más mal clasificados fueron el 3, el 8 y el 9. La tabla 9 muestra los valores de las métricas de precisión obtenidas a partir de la matriz de confusión.

Símbolo	Precisión	Recall	F1-Score
0	0.98	0.99	0.98
1	0.99	0.99	0.99
2	0.97	0.98	0.98
3	0.98	0.97	0.98
4	0.98	0.98	0.98
5	0.98	0.97	0.97
6	0.98	0.98	0.98
7	0.98	0.98	0.98
8	0.97	0.97	0.97
9	0.98	0.97	0.97

Tabla 9. Resumen de métricas para algoritmo MLP

2.4.8 Red neuronal convolucional (CNN)

La red propuesta posee una estructura como la mostrada en la figura 11. Esta compuesta por una primera capa convolucional con máscara de 3x3, con 64 neuronas, una segunda capa convolucional con iguales características, una capa oculta de 512 neuronas, otra capa oculta con 256 neuronas y todas estas con función de activación rectificación lineal. La capa de salida está compuesta por 10 neuronas y emplea la función de activación softmax.

```
Model: "sequential_29"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 64)        640

 conv2d_1 (Conv2D)           (None, 24, 24, 64)        36928

 max_pooling2d (MaxPooling2D  (None, 12, 12, 64)       0
 )

 flatten (Flatten)           (None, 9216)              0

 dense_87 (Dense)            (None, 512)               4719104

 dense_88 (Dense)            (None, 256)               131328

 dense_89 (Dense)            (None, 10)                2570

=================================================================
Total params: 4,890,570
Trainable params: 4,890,570
Non-trainable params: 0
```

Figura 11. Resumen del modelo de red convolucional propuesta

El algoritmo ofreció exactitud de 0.9998 durante el entrenamiento en un tiempo prudencial de 13191 segundos. Luego de clasificado el conjunto de prueba la exactitud del algoritmo descendió hasta 98.30, esta operación se completó en un tiempo de 15.90 segundos. En la figura 12 se muestra la matriz de confusión resultante del análisis del conjunto de entrenamiento.

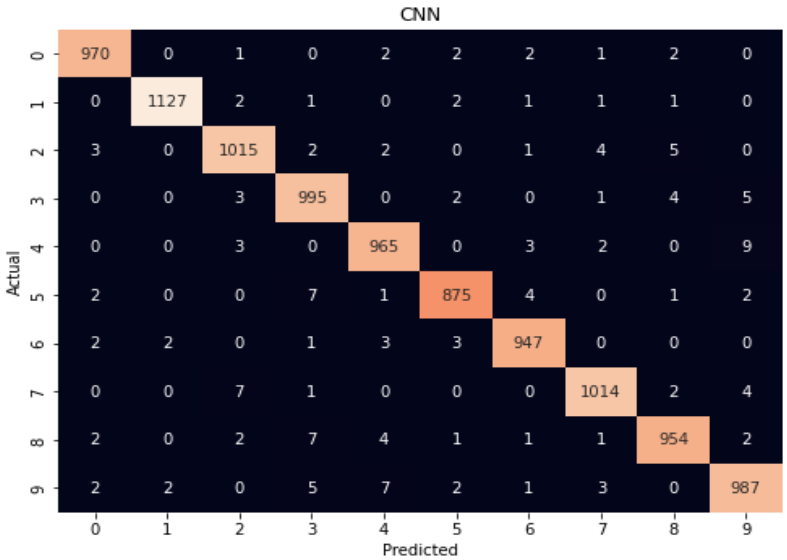

Figura 12. Matriz de confusión obtenida del conjunto de prueba mediante el algoritmo CNN

Los símbolos con mejor tasa de clasificación según los resultados de las métricas mostrado en la tabla 10 fueron el 1 y el 0. Los resultados fueron muy buenos para todos los símbolos, obteniendo valores de métricas entre 0.98 y 1.00 para cada uno.

Símbolo	Precisión	Recall	F1-Score
0	0.99	0.99	0.99
1	1.00	0.99	0.99
2	0.98	0.98	0.98
3	0.98	0.99	0.98
4	0.98	0.98	0.98
5	0.99	0.98	0.98
6	0.99	0.99	0.99
7	0.99	0.99	0.99
8	0.98	0.98	0.98
9	0.98	0.98	0.98

Tabla 10. Resumen de métricas para algoritmo CNN

3. Resultados

Luego de analizados los desempeños de los algoritmos se puede corroborar que varios de estos ofrecieron resultados de exactitud superior al 95% durante el entrenamiento, la figura 13 muestra un gráfico con los resultados de exactitud por cada modelo en las etapas de entrenamiento y prueba.

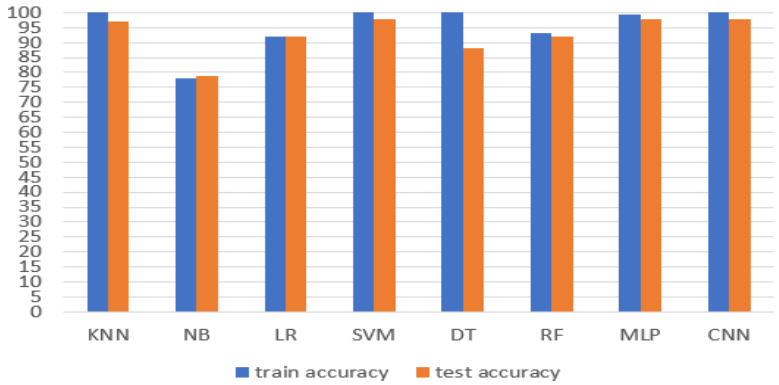

Figura 13. Muestra los resultados de exactitud por cada algoritmo durante las etapas de entrenamiento y prueba

Los algoritmos que superaron tanto para el conjunto el entrenamiento como para el de prueba el 95% de exactitud fueron KNN, SVM, MLP y CNN. Un caso particular fue el de DT que clasificó correctamente el 100% de los objetos durante el entrenamiento; sin embargo, su resultado descendió notablemente para el conjunto de prueba, lo que puede intuir en un modelo no ajustado completamente a los datos o sobreajuste, ya que el algoritmo memorizó los datos del entrenamiento no pudiendo clasificar correctamente nuevas muestras no analizadas durante el entrenamiento. Si se hace un análisis particularizado de la precisión que cada modelo estimó al conjunto de prueba por cada uno de sus símbolos, se puede corroborar que los modelos que mejores resultados tuvieron en esa métrica fueron de igual manera KNN, SVM, MLP y CNN. Los símbolos 0 y 1 obtuvieron el mayor valor de percisión en todos los algoritmos analizados. Ver figura 14 para mayor comprensión.

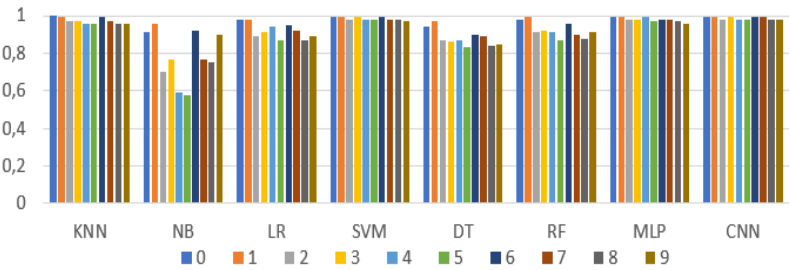

Figura 14. Precisión modelo/símbolos

Si se considera el consumo de tiempo por cada algoritmo, durante el entrenamiento, en orden ascendente se tiene la siguiente lista KNN, NB, RF, DT, LR, SVM, MLP y CNN, la figura 15 muestra un resumen del consumo de tiempo en escala logarítmica de los modelos.

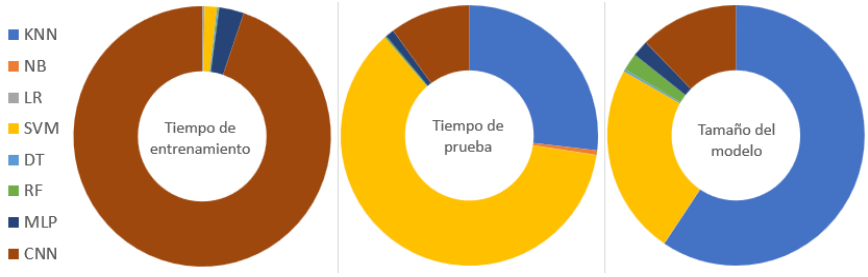

Figura 15. Consumo de tiempo y almacenamiento por modelo

En cuanto al tiempo consumido por los algoritmos para procesar los datos de prueba, los resultados por modelos en orden ascendente son los siguientes: DT, LR, RF, NB, MLP, CNN, KNN y SVM. Si se analiza el tamaño en kilobytes de los modelos en orden ascendente, se tiene la siguiente sucesión: LR, NB, DT, MLP, RF, CNN, SVM y KNN. Ver tabla 11 para más información.

Algoritmo	Tiempo de entrenamiento (segundos)	Tiempo de prueba (segundos)	Tamaño del modelo (kilobytes)
KNN	0.6127	42.2709	183986
NB	0.9755	0.8727	124
LR	39.5372	0.042	63
SVM	217.6239	96.3754	73582
DT	21.9729	0.027	810
RF	14.8569	0.3141	6693
MLP	434.8619	1.7201	6305
CNN	13190.9868	15.904	38249

Tabla 11. Consumo de recursos de los algoritmos analizados

4. Conclusiones

El presente estudio permitió analizar el desempeño de 8 algoritmos muy empleados en las ciencias de la computación para tareas de inteligencia artificial como clasificación y reconocimiento de patrones. Los resultados ofrecidos están sujetos a los parámetros establecidos a cada algoritmo, y estos a su vez están relacionados con la búsqueda exhaustiva mediante grid search sobre los hiperparámetros definidos al inicio de la investigación.

Se pudieron obtener las métricas de clasificación para cada algoritmo obteniendo como resultado que 4 algoritmos superaron el 97% de exactitud tanto durante el entrenamiento como durante la prueba, estos fueron KNN, SVM, MLP y CNN, ubicándolos como los de mejor desempeño para el conjunto de datos analizado MNIST con ligeras diferencias entre estos. La métrica de precisión de los símbolos durante el proceso de prueba del conjunto de datos mostró valores elevados para los 4 algoritmos anteriores resultados, lo mismo sucedió con la métrica de exhaustividad y F1-Score. Los tiempos de entrenamiento arrojaron una diferencia bien marcada entre los algoritmos como se aprecia en la figura 16, consumiendo la mayor cantidad de tiempo el algoritmo CNN, seguido por MLP, SVM y KNN en ese orden. En el caso del consumo de tiempo en la etapa de prueba, los algoritmos que más tiempo emplearon fueron SVM, KNN, CNN y MLP en ese orden. El tamaño del almacenamiento ocupado por el modelo entrenado fue otro elemento tomado en consideración, el modelo que más espacio consumió fue KNN, seguido por SVM, CNN y finalmente MLP.

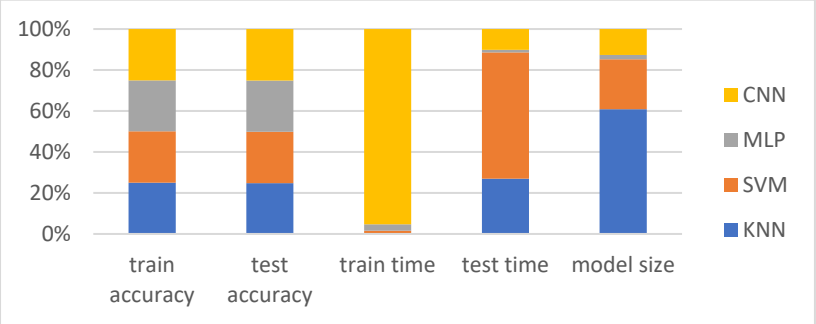

Figura 16. Resultados de exactitud, tiempo y memoria por algoritmo

El algoritmo que mejores resultados ofreció de manera general para las métricas, los tiempos de entrenamiento y prueba, así como de consumo de espacio en almacenamiento fue el de MLP, ofreciendo un balance entre todos los elementos medidos. Este algoritmo brindó una exactitud de 0.9946 y 0.98 para el entrenamiento y prueba respectivamente, consumiendo de manera moderada los recursos de tiempo y memoria, lo que lo convierte en el principal candidato a considerar para la implementación de una aplicación movil que garantice el equilibrio entre el consumo de tiempo y almacenamiento manteniendo valores altos de exactitud.

REFERENCIAS BIBLIOGRÁFICAS

Alarcón Flores, J. B. (2017). Modelos de minería de datos: Random forest y adaboost, para identificar los factores asociados al uso de las TIC (internet, telefonía Fija y televisión de paga) en los hogares del Perú. 2014. https://cybertesis.unmsm.edu.pe/handle/20.500.12672/7404

Baldominos Gómez, A., Sáez Achaerandio, Y., & Isasi Viñuela, P. (2019). A survey of handwritten character recognition with MNIST and EMNIST. https://e-archivo.uc3m.es/handle/10016/38329

Belete, D. M., & Huchaiah, M. D. (2022). Grid search in hyperparameter optimization of machine learning models for prediction of HIV/AIDS test results. International Journal of Computers and Applications, 44(9), 875-886.

Borja-Robalino, R., Monleon-Getino, A., & Rodellar, J. (2020). Estandarización de métricas de rendimiento para clasificadores Machine y Deep Learning. Revista Ibérica de Sistemas e Tecnologias de Informação, E30, 184-196.

Braga-Neto, U. (2020). Fundamentals of pattern recognition and machine learning. Springer.

Cho, G., Yim, J., Choi, Y., Ko, J., & Lee, S.-H. (2019). Review of machine learning algorithms for diagnosing mental illness. Psychiatry investigation, 16(4), 262.

Dhillon, A., & Verma, G. K. (2020). Convolutional neural network: A review of models, methodologies and applications to object detection. Progress in Artificial Intelligence, 9(2), 85-112.

Ehsani, R., & Drabløs, F. (2020). Robust Distance Measures for kNN Classification of Cancer Data. Cancer Informatics, 19, 1176935120965542. https://doi.org/10.1177/1176935120965542

Gardner, M. W., & Dorling, S. R. (1998). Artificial neural networks (the multilayer perceptron)—A review of applications in the atmospheric sciences. Atmospheric Environment, 32(14), 2627-2636. https://doi.org/10.1016/S1352-2310(97)00447-0

Gasparetto, A., Marcuzzo, M., Zangari, A., & Albarelli, A. (2022). A survey on text classification algorithms: From text to predictions. Information, 13(2), 83.

Grandini, M., Bagli, E., & Visani, G. (2020). Metrics for multi-class classification: An overview. arXiv preprint arXiv:2008.05756.

Griffis, J. C., Allendorfer, J. B., & Szaflarski, J. P. (2016). Voxel-based Gaussian naïve Bayes classification of ischemic stroke lesions in individual T1-weighted MRI scans. Journal of neuroscience methods, 257, 97-108.

Gutiérrez Esparza, G. O., Margain Fuentes, M. de L., Ramírez del Real, T. A., & Canul Reich, J. (2017). Un modelo basado en el Clasificador Naïve Bayes para la evaluación del desempeño docente. RIED. Revista iberoamericana de educación a distancia.

Hernández, A. Z., Rosales, G. A. G., Santiago, H. J. J., & Lee, M. M. (2022). Métricas de rendimiento para evaluar el aprendizaje automático en la clasificación de imágenes petroleras utilizando redes neuronales convolucionales. Ciencia Latina Revista Científica Multidisciplinar, 6(5), 4624-4637.

Kalmegh, S. R., & Padar, B. R. (2023). Empirical Study on Evaluation Metrics for Classification Algorithms. International Journal of Advanced Research in Science, Communication and Technology.

Kirişci, M. (2019). Comparison of artificial neural network and logistic regression model for factors affecting birth weight. SN Applied Sciences, 1(4), 378.

Lévano-Rodriguez, D., & Cerdán-León, F. E. (2022). Discriminación de masas mamográficas mediante K-Nearest Neighbor y atributos BIRADS. Revista Científica de Sistemas e Informática, 2(1), Article 1. https://doi.org/10.51252/rcsi.v2i1.225

Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). A survey of convolutional neural networks: Analysis, applications, and prospects. IEEE transactions on neural networks and learning systems.

Llumiquinga Almeida, E. P. (2022). Estudio comparativo de los algoritmos de clasificación supervisada empleando datos artificiales. [B.S. thesis]. Quito, 2022.

Luu, S. T., Nguyen, H. P., Van Nguyen, K., & Nguyen, N. L.-T. (2020). Comparison between traditional machine learning models and neural network models for vietnamese hate speech detection. 2020 RIVF International Conference on Computing and Communication Technologies (RIVF), 1-6.

Martínez-Toro, G. M., Rico-Bautista, D., & Romero-Riaño, E. (2019). Análisis comparativo de predicción dentro de bases de datos de cáncer: Una aplicación de aprendizaje automático. Revista Ibérica de Sistemas e Tecnologias de Informação, E17, 113-122.

Merenda, M., Porcaro, C., & Iero, D. (2020). Edge machine learning for ai-enabled iot devices: A review. Sensors, 20(9), 2533.

Naranjo, L. T. L. (2022). Diseño de una interfaz cerebro computador (BCI) para la interacción con un sistema de rehabilitación de miembro superior.

Ray, S. (2019). A quick review of machine learning algorithms. 2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon), 35-39.

Ruiz-Shulcloper, J., Guzmán Arenas, A., & Martínez-Trinidad, J. F. (1999). Enfoque lógico combinatorio al reconocimiento de patrones. I. Selección de Variables y Clasificación Supervisada, IPN, México.

Sen, P. C., Hajra, M., & Ghosh, M. (2020). Supervised Classification Algorithms in Machine Learning: A Survey and Review. En J. K. Mandal & D. Bhattacharya (Eds.), Emerging Technology in Modelling and Graphics (Vol. 937, pp. 99-111). Springer Singapore. https://doi.org/10.1007/978-981-13-7403-6_11

Sheykhmousa, M., Mahdianpari, M., Ghanbari, H., Mohammadimanesh, F., Ghamisi, P., & Homayouni, S. (2020). Support vector machine versus random forest for remote sensing image classification: A meta-analysis and systematic review. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 13, 6308-6325.

Tanveer, M., Rajani, T., Rastogi, R., Shao, Y.-H., & Ganaie, M. A. (2022). Comprehensive review on twin support vector machines. Annals of Operations Research, 1-46.

Wang, Y., Li, F., Sun, H., Li, W., Zhong, C., Wu, X., Wang, H., & Wang, P. (2020). Improvement of MNIST Image Recognition Based on CNN. IOP Conference Series: Earth and Environmental Science, 428(1), 012097. https://doi.org/10.1088/1755-1315/428/1/012097

Yann, L. (1998). The mnist database of handwritten digits. R.

Zhang, X.-Y., Liu, C.-L., & Suen, C. Y. (2020). Towards robust pattern recognition: A review. Proceedings of the IEEE, 108(6), 894-922.