UNIVERSIDAD MILITAR NUEVA GRANADA

MODELO DE CLASIFICACIÓN DE IMÁGENES CON MLP

DOCENTE

DIEGO RENZA TORRES

ESTUDIANTES

EDWIN ESTEBAN PINEDA BOCIGA
JAVIER STIVEN MUÑOZ CORTES

PROGRAMA

INGENIERIA EN TELECOMUNICACIONES

FECHA

26/08/2025

INTRODUCCIÓN

El perceptrón multicapa es un modelo de red neuronal artificial compuesto por capas de neuronas conectadas de manera densa, el cual ha sido ampliamente utilizado en tareas de clasificación. Este tipo de red se caracteriza por su capacidad para aprender representaciones no lineales de los datos mediante funciones de activación y retro propagación del error durante el entrenamiento. En el caso del conjunto de datos BloodMNIST, perteneciente a la colección MedMNIST, se dispone de imágenes biomédicas de células sanguíneas a baja resolución (28×28 píxeles), organizadas en diferentes clases. Este escenario resulta adecuado para evaluar el desempeño de un MLP en la clasificación de imágenes médicas simples, ya que permite analizar su eficacia al trabajar con datos visuales reducidos y al mismo tiempo explorar su potencial como herramienta de apoyo en aplicaciones de diagnóstico automatizado.

CRITERIOS DE DISEÑO

Para llevar a cabo el modelo, se tuvo en cuenta la distribución de las imágenes entregada por el MedMNIST, la cual cuenta con una resolución de 28×28 a 3 canales, con los siguientes datos de entrenamiento:

MedMNIST2D	Modalidad de datos	Tareas (# Clases/Etiquetas)	# Muestras	# Entrenamiento / Validación / Prueba
BloodMNIST	Microscopio de células sanguíneas	Multiclase (8)	17.092	11.959 / 1.712 / 3.421

Para la selección de hiperparámetros se tuvo en cuenta la cantidad de neuronas utilizadas, la capas, la función de activación, tamaño del lote y tasa de aprendizaje, el optimizador seleccionado fue Adam, debido a su alta convergencia y ajuste de tasa de aprendizaje, por cual se usó 30 épocas para evidenciar el avance del modelo, el tamaño del lote se tomó una lote de 16 a 32, con el fin de tener el resultado más generalizado y rápido posible, al usar un lote grande, converge más rápido pero no compara con las suficientes imágenes, al usar un lote pequeño, el modelo puede tardar un poco más pero se realiza una comparación más extensa.

En general, la selección de los hiperparámetros se basó en velocidad y generalización de los datos, precisión y resultados obtenidos de las pruebas realizadas para cada uno de los casos.

CARACTERISTICAS DEL MODELO

Para el modelo MLP realizado, se usó en todos los casos el lote completo de entrenamiento, es decir, se tuvo en cuenta las 11959 imágenes brindadas, además se priorizó la precisión del modelo esperando un resultado mayor al 80% y se realizaron pruebas con 3 a 4 capas en cada caso, observando el compartimiento del modelo también se evidenció la capacidad de aprendizaje de patrones minimizando la memorización de imágenes de entrenamiento mediante la comparación de imágenes de validación.

SELECCIÓN DE MODELOS

Se hizo la construcción de 3 modelos, los 2 primeros modelos se realizaron bajo los criterios mencionados y sin el apoyo de ninguna herramienta, a los cuales se le realizaron variaciones en el tamaño de los lotes, la tasa de aprendizaje y la cantidad de capas usadas obteniendo resultados similares entre ellos, al realizar el tercer modelo se usó una herramienta llamada optuna, un framework de selección automática de hiperparámetros diseñada para machine learning, a la cual se le puede asignar los parámetros que se necesiten y luego del número de pruebas seleccionado entregará los hiperparámetros con mejor desempeño.

En este caso, a la herramienta se le permitió realizar variaciones en:

- Capa 1 de 128 neuronas a 1024 realizando pasos de 8 neuronas, el optimizador y su función de activación ('relu', 'tanh', 'sigmoid', 'selu')
- Capa 2 de 64 neuronas a 1024 realizando pasos de 8 neuronas, el optimizador y su función de activación ('relu', 'tanh', 'sigmoid')
- Tasa de aprendizaje entre 0.00001 a 0.01
- Tamaño del lote de 16 a 128
- Se realizaron 20 pruebas determinar el mejor conjunto de hiperparámetros

1) Modelo #1

```
x_train_Mod = x_train[1:11959,:,:,:]
y_train_Mod = y_train[1:11959,:]

#Hiperparámetros
batch_s, lr, num_epochs = 16, 0.0005, 30
optimizerf = tf.keras.optimizers.Adam(learning_rate=lr)

#Modelo
model = Sequential([
                    Flatten(),  # 28*28=784
                    Dense(240, activation='relu'),
                    Dense(192, activation='sigmoid'),
                    Dense(8, activation='softmax')
])
```

2) Modelo #2

```
#Datos
x_train_Mod = x_train[1:11959,:,:,:]
y_train_Mod = y_train[1:11959,:]
#Hiperparámetros
batch_s, lr, num_epochs = 32, 0.001, 30
optimizerf = tf.keras.optimizers.Adam(learning_rate=lr)

#Modelo
model = Sequential([
                    Flatten(),  # 28*28=784
                    Dense(128, activation='relu'),
                    Dense(8, activation='softmax')
])
```

3) Modelo #3

```
x_train_Mod = x_train[1:11959,:,:,:]
y_train_Mod = y_train[1:11959,:]
weight_decay1 = 0.000001
#Hiperparámetros
batch_s, lr, num_epochs = 16, 3.568382753188613e-05, 30
optimizerf = tf.keras.optimizers.Adam(learning_rate=lr)

#Modelo
model = Sequential([
                    Flatten(),  # 28*28=784
                    Dense(832,activation='tanh',kernel_regularizer=
tf.keras.regularizers.l2(weight_decay1)),
                    Dense(512,activation="relu"),
                    Dense(8, activation='softmax')
])
```

GENERALIZACIÓN

Los resultados obtenidos durante el entrenamiento del modelo MLP aplicado al conjunto BloodMNIST muestran un comportamiento estable tanto en la pérdida como en la exactitud, lo cual refleja una adecuada capacidad de generalización. En la curva de loss, se observa un descenso progresivo de la función de costo en el conjunto de entrenamiento y un comportamiento similar en el conjunto de validación, con diferencias moderadas entre ambas métricas. Esto indica que el modelo logra aprender patrones relevantes de los datos sin presentar un sobreajuste evidente.

De manera complementaria, la curva de accuracy evidencia un incremento sostenido en el rendimiento del modelo a lo largo de las épocas. La exactitud en entrenamiento alcanza valores cercanos, mientras que la exactitud en validación se mantiene con oscilaciones leves, pero sin pérdida significativa de desempeño. La proximidad entre ambas curvas confirma que el modelo es capaz de clasificar correctamente ejemplos no vistos, lo que valida su eficacia para la tarea de clasificación de imágenes de células sanguíneas en baja resolución.

1) Modelo #1

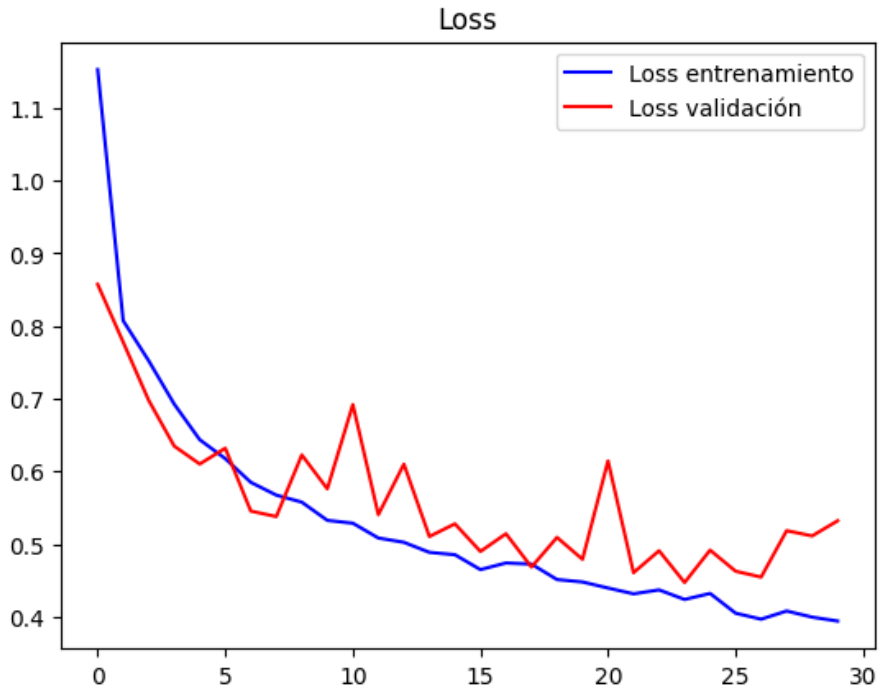

Imagen 1. Función de perdidas Modelo #1

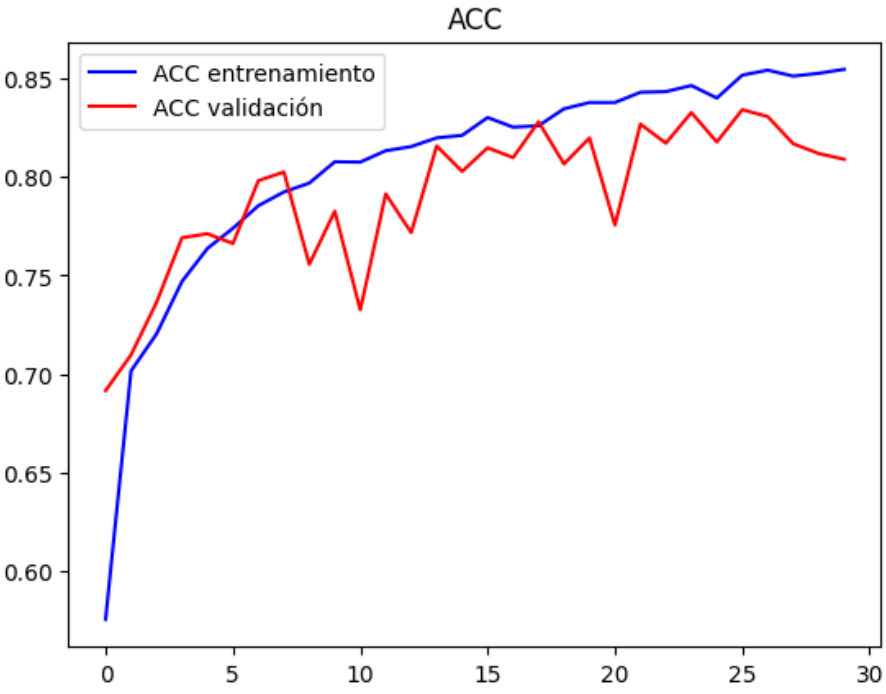

Imagen 2. Función de precisión Modelo #1

Como se puede observar, en ambas imágenes el modelo presenta al inicio una buena identificación de las imágenes, a medida que el modelo avanza comienza a presentar un estancamiento en su aprendizaje, lo cual puede significar que al modelo se le está dificultando el proceso de identificación de nuevos patrones.

2) Modelo #2

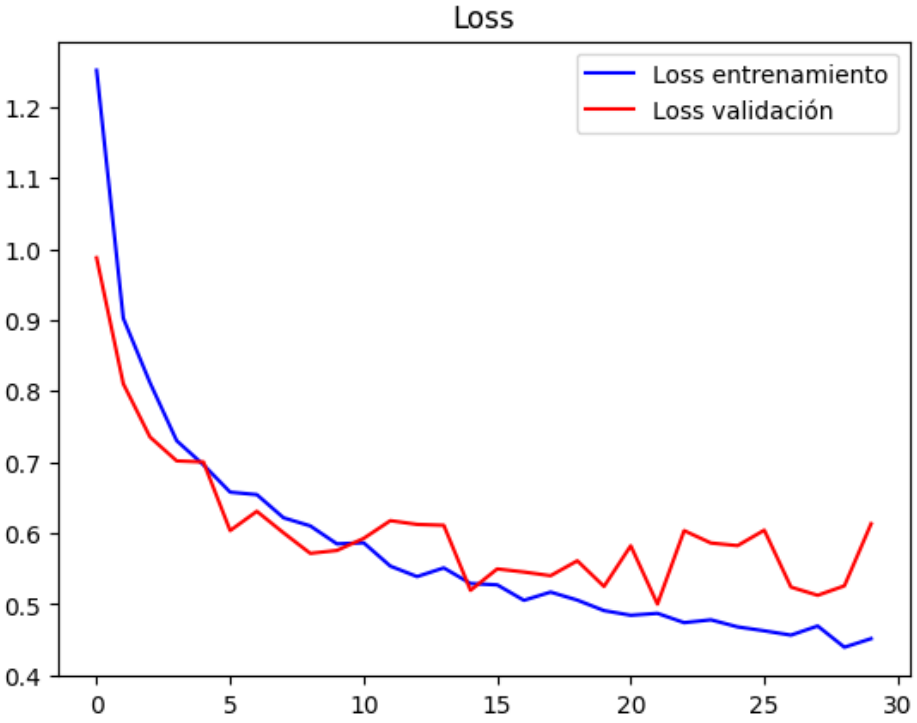

Imagen 3. Función de perdidas Modelo #2

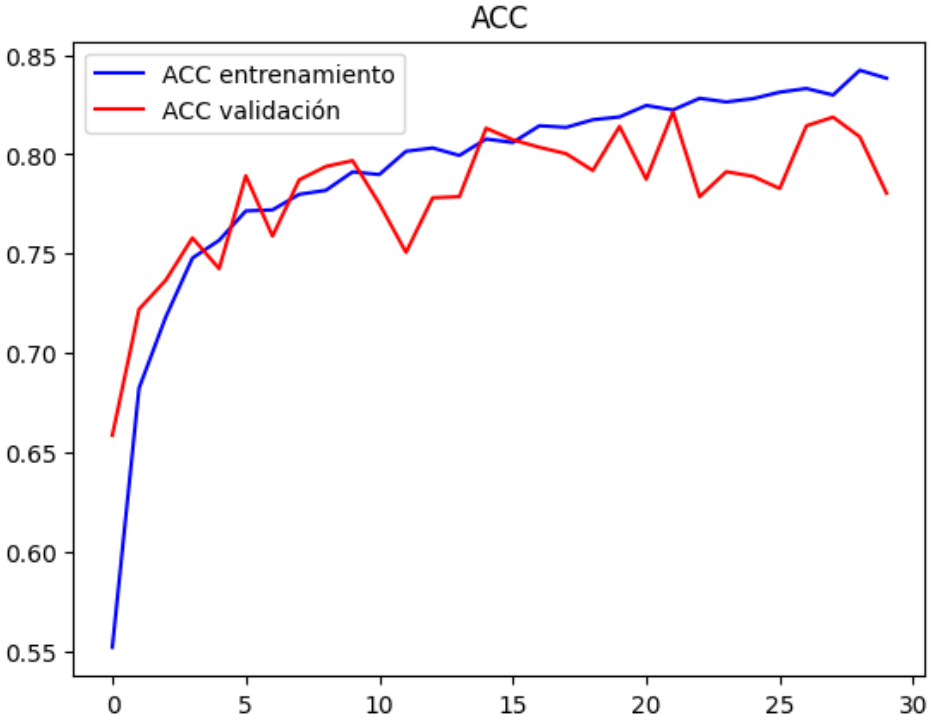

Imagen #4. Función de precisión Modelo #2

De igual forma, el modelo presenta un inicio bastante bueno, pero a medida que sube sufre un estancamiento similar al modelo #1.

3) Modelo #3

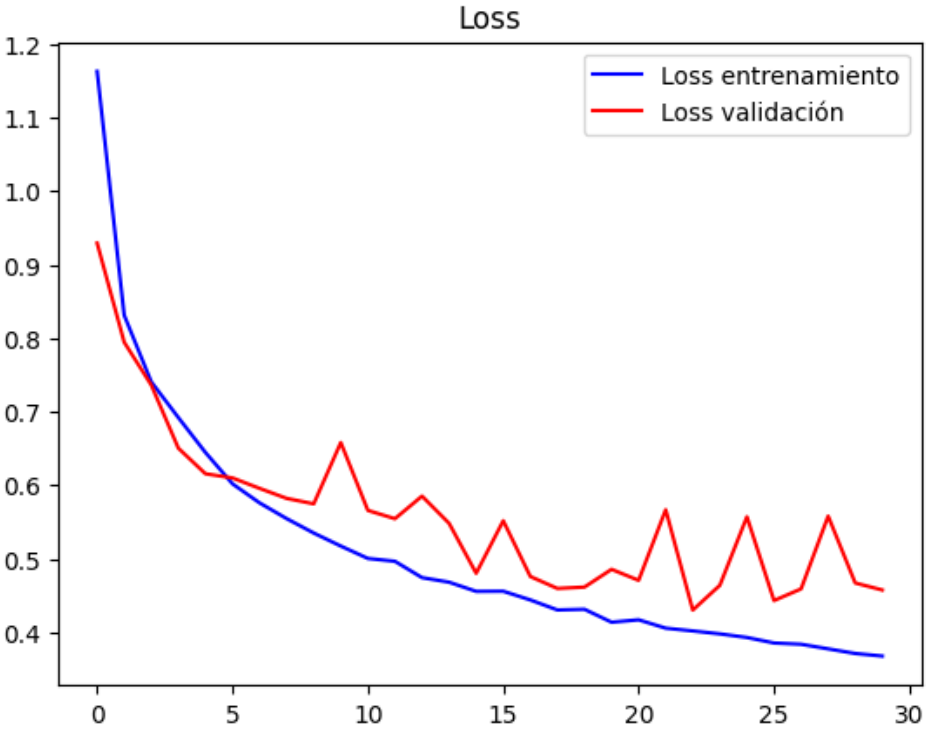

Imagen 5. Función de perdidas Modelo #3

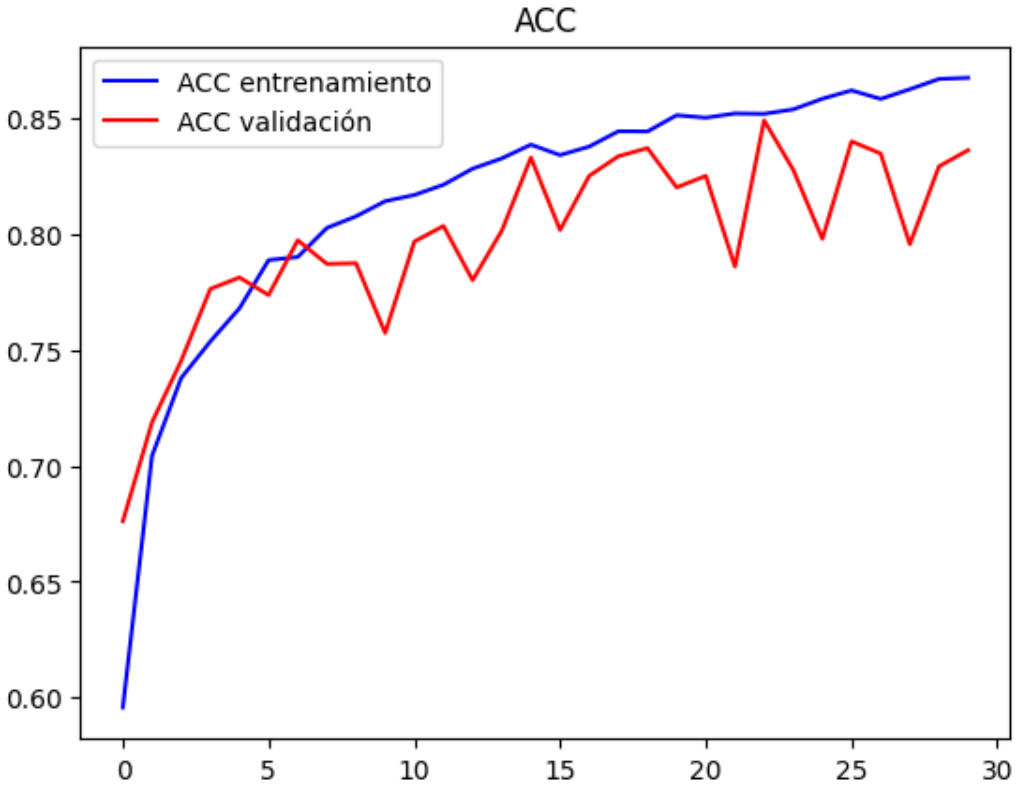

Imagen 6. Función de perdidas Modelo #3

Al usar la herramienta podemos observa una leve mejora en la precisión y las perdidas, aunque no es una mejora considerable, la herramienta brinda hiperparámetros con la posibilidad de entregar un mejor desempeño al momento de realizar las pruebas.

EVALUACIÓN DE RENDIMIENTO

Modelo	Precisión_test	Perdida_test	Precisión_validación	Perdida_validación
#1	0,8648	0,3755	0,8118	0,5054
#2	0,8697	0,3683	0,8410	0,4533
#3	0,8798	0,3397	0,8439	0,4310

Al revisar los datos obtenidos por cada modelo, se puede observar que la diferencia de la precisión entre los datos no es muy grande, no obstante, al hacer uso de la herramienta, se logró obtener un valor un poco mayor, ya que los primeros 2 modelos no superaban el 86 % de precisión mientras que el tercer modelo logró llegar en diferentes pruebas a 89%, pero en general se sus variaciones entre los datos de prueba y de validación son bastantes cercanos, lo que nos deja entender que el modelo puede realizar generalizaciones de manera correcta.

REFERENCIAS

[1] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A next-generation hyperparameter optimization framework,” in Proc. 25th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining (KDD), Anchorage, AK, USA, Jul. 2019, pp. 2623–2631.

[2] Veloz, O. (2025). Application and Evaluation of Neural Networks for Blood Cell Recognition with the BloodMNIST Database. https://doi.org/10.13140/RG.2.2.10588.09609