Clasificación de gestos de mano mediante señales EMG utilizando redes LSTM

De Mónica Monserrat Martínez Vásquez | A01710965

Inteligencia artificial avanzada para la ciencia de datos I (Gpo 101)

Fecha: 09/11/2025

1. INTRODUCCIÓN

El presente trabajo tiene como propósito desarrollar un modelo de deep learning capaz de clasificar gestos de la mano a partir de señales electromiográficas (EMG) multicanal. Estas señales, registradas a través de un brazalete con electrodos de superficie, representan la actividad eléctrica muscular asociada a distintos movimientos y posturas de la mano. El objetivo principal del proyecto es aplicar de manera efectiva los fundamentos teóricos y prácticos del aprendizaje profundo, utilizando un conjunto de datos real y un modelo implementado íntegramente con el framework PyTorch.

En primer lugar, se seleccionó un conjunto de datos de EMG que contiene registros de ocho canales correspondientes a diferentes gestos manuales. Posteriormente, se diseñó una estrategia de preprocesamiento que incluye limpieza de datos, normalización por canal mediante StandardScaler, y segmentación temporal en ventanas deslizantes de longitud fija, garantizando que las muestras de cada ventana pertenezcan a una misma clase. Este procedimiento permitió transformar las secuencias continuas en instancias adecuadas para entrenamiento supervisado.

El modelo propuesto se basa en una arquitectura LSTM (Long Short-Term Memory), adecuada para la modelación de dependencias temporales en series de tiempo. La red implementada consta de una capa recurrente con 64 unidades latentes, seguida de una cabeza densa compuesta por una capa intermedia de 64 neuronas, una función de activación ReLU y una capa de salida con ocho neuronas correspondientes a las clases de gestos. Se utilizó la función de pérdida de entropía cruzada (CrossEntropyLoss) y el optimizador Adam con una tasa de aprendizaje de 0.01. La red se entrenó durante 30 épocas con particiones estratificadas para entrenamiento, validación y prueba, con el objetivo de evaluar la capacidad de generalización del modelo.

Además del entrenamiento, se desarrolló una interfaz interactiva en Google Colab utilizando ipywidgets, la cual permite cargar archivos .txt con registros EMG, aplicar el mismo proceso de normalización y obtener predicciones en tiempo real. Esta herramienta facilita la demostración del modelo, permitiendo observar tanto la clase predicha como las tres clases más probables junto con sus probabilidades estimadas.

2. SELECCIÓN DEL DATASET

Para el desarrollo del presente proyecto se seleccionó el conjunto de datos público “EMG Data for Gestures”[1] , disponible en la plataforma Kaggle. Este dataset fue obtenido mediante el brazalete Myo Armband de la compañía Thalmic Labs, el cual cuenta con ocho sensores de electromiografía superficial (sEMG) puestos alrededor del antebrazo. Los datos representan la actividad eléctrica de los músculos durante la ejecución de diferentes gestos de la mano, tales como flexión, extensión, desviación radial y cubital, puño cerrado, palma extendida y mano relajada. Cada gesto corresponde a una etiqueta numérica específica que se asocia a las lecturas simultáneas de los ocho canales.

La elección de este conjunto de datos se basó en diversos criterios técnicos y metodológicos. En primer lugar, se trata de un dataset real y multicanal, lo que permite aplicar arquitecturas profundas diseñadas para secuencias temporales, como las redes LSTM, y evaluar su capacidad de modelar dependencias dinámicas. En segundo lugar, los archivos están organizados por sesiones y sujetos, almacenados en formato de texto plano (.txt), lo cual facilita la lectura y manipulación mediante bibliotecas de análisis de datos en Python. Finalmente, la naturaleza del problema que de clasificación de gestos musculares, constituye un escenario representativo de tareas de reconocimiento de patrones fisiológicos.

Cada archivo del conjunto contiene diez columnas: una columna temporal (time), ocho canales de señal EMG (ch1 a ch8) y una columna de etiqueta (label). Las señales fueron muestreadas a alta frecuencia, permitiendo capturar los cambios rápidos de la actividad muscular. En la mayoría de los casos, las etiquetas permanecen constantes durante la ejecución de un gesto, lo cual posibilita segmentar los datos en ventanas temporales homogéneas para su tratamiento como

muestras supervisadas. Cabe destacar que, a diferencia de datasets simplificados, este conjunto de datos presenta ruido fisiológico y variabilidad intersujeto, lo que incrementa la complejidad.

Antes del procesamiento, se realizó una inspección de consistencia para verificar que todos los archivos cumplieran con el mismo formato de columnas y se descartaron aquellos con estructuras anómalas o etiquetas no válidas. Después, se contabilizaron los registros disponibles por clase, confirmando un número equilibrado de ejemplos por gesto, lo cual permitió aplicar una estrategia de estratificación durante la partición de los conjuntos de entrenamiento, validación y prueba.

3. PREPARACIÓN DE DATOS

El proceso de preparación de datos dado que los archivos del conjunto de datos se encontraban en formato de texto plano y contenían registros secuenciales con ruido relacionado a la adquisición fisiológica, se diseñó un pipeline de preprocesamiento que abarcó desde la lectura y limpieza de los archivos hasta la generación de las instancias finales de entrenamiento.

En primer lugar, se implementó un procedimiento automatizado para la lectura masiva de archivos .txt mediante la biblioteca glob, permitiendo recorrer de manera recursiva la estructura de directorios y concatenar los datos en un único DataFrame de pandas. Durante esta etapa se validó que cada archivo contuviera exactamente diez columnas y que las etiquetas de clase (label) fueran distintas de cero, eliminando filas vacías, no numéricas o con formatos inconsistentes. Esta limpieza garantizó la homogeneidad estructural de los registros y evitó que los errores de formato afectaran las fases posteriores de normalización y segmentación.

Después se aplicó una transformación de estandarización por canal utilizando la clase StandardScaler de scikit-learn. Este proceso normalizó cada canal de señal para que su media fuera cero y su desviación estándar uno, preservando la forma de las señales pero reduciendo la dispersión entre canales y sesiones.

Una vez normalizadas las señales, se procedió a la segmentación temporal mediante ventanas deslizantes (sliding windows). Se definió una longitud de ventana de 200 muestras y un desplazamiento de 100 muestras entre ventanas consecutivas. Cada ventana se consideró válida únicamente si todas las etiquetas en su interior correspondían a una misma clase, lo cual asegura la consistencia de las instancias. Este método permitió convertir las secuencias continuas de cada archivo en múltiples muestras independientes con estructura tridimensional (n_samples, time_steps, features), donde features corresponde a los ocho canales EMG.

El conjunto resultante de instancias fue posteriormente dividido en subconjuntos de entrenamiento, validación y prueba mediante la función train_test_split con estratificación, de modo que la distribución de clases se mantuviera balanceada en cada partición. Se asignó un 80% de los datos al conjunto de entrenamiento y el 20% restante se subdividió equitativamente entre validación y prueba (10% cada uno). Esta división permitió ajustar los hiperparámetros del modelo sobre la base de validación y evaluar su capacidad de generalización sobre un conjunto completamente independiente.

Finalmente, se encapsularon las matrices de entrada y etiquetas en un objeto personalizado de la clase Dataset de PyTorch, denominado EMGDataset. Este contenedor facilita la indexación y la creación de DataLoaders que alimentan los lotes (batches) de datos durante el entrenamiento y la inferencia. Cada lote se conformó con 64 ventanas temporales, seleccionadas de forma aleatoria en el conjunto de entrenamiento, permitiendo un aprendizaje eficiente mediante mini-lotes y aceleración en GPU.

4. IMPLEMENTACIÓN DEL MODELO, EJECUCIÓN Y ANÁLISIS DEL ENTRENAMIENTO

La primera versión del modelo propuesto se basó en una arquitectura Long Short-Term Memory (LSTM), una red neuronal recurrente diseñada para modelar dependencias temporales en secuencias de datos. La elección de esta arquitectura se fundamenta en la naturaleza temporal de las señales electromiográficas (EMG), las cuales presentan correlaciones a lo largo del tiempo debido a la contracción muscular sostenida y la dinámica fisiológica inherente a cada gesto. A diferencia de las redes convolucionales o de perceptrones multicapa, las redes LSTM son capaces de mantener un estado de memoria interna, permitiendo capturar patrones secuenciales a largo plazo sin sufrir el problema del desvanecimiento del gradiente.

La arquitectura implementada está compuesta por una capa LSTM unidireccional con 64 unidades latentes (latentDim), seguida de una cabeza densa conformada por una capa completamente conectada de 64 neuronas con activación ReLU, una capa Dropout con una tasa de 0.3, y una capa de salida lineal con ocho neuronas correspondientes a las clases de gestos. La función de pérdida utilizada fue la entropía cruzada categórica (CrossEntropyLoss), mientras que el optimizador Adam

fue seleccionado por su robustez. El modelo fue implementado en PyTorch, aprovechando su estructura modular para definir la clase EMGLSTMClassifier, que encapsula la inicialización y el proceso de propagación hacia adelante (forward pass).

El entrenamiento se realizó durante 30 épocas, con un tamaño de lote de 64 muestras y una tasa de aprendizaje inicial de 0.01. El conjunto de entrenamiento se utilizó para la actualización de los pesos, mientras que el conjunto de validación sirvió para monitorear la pérdida (loss) y la exactitud (accuracy) después de cada época. Este seguimiento permitió evaluar la convergencia del modelo y detectar posibles signos de sobreajuste. El cálculo de métricas se automatizó mediante funciones auxiliares, las cuales permitieron medir la exactitud global y generar las predicciones sobre los conjuntos de validación y prueba.

Durante la ejecución, se observó un descenso progresivo de la función de pérdida de entrenamiento, acompañado por un incremento sostenido en la precisión de validación, estabilizándose a partir de la época veinte. En la etapa final del entrenamiento, la exactitud en el conjunto de prueba alcanzó un 68.05 %, lo que demuestra una capacidad de generalización moderada considerando la complejidad de las señales EMG.

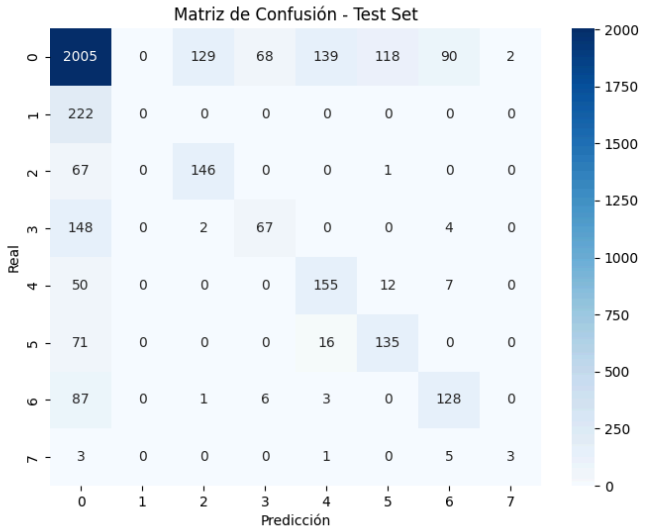

Figura 1. Matriz de confusión a partir de las predicciones sobre el conjunto de prueba

Para el análisis cualitativo del desempeño, se generó una matriz de confusión (ver figura 1) a partir de las predicciones sobre el conjunto de prueba. En esta matriz se evidenció que la mayoría de las clases fueron correctamente clasificadas, con errores concentrados principalmente entre gestos de morfología electromiográfica similar, como las desviaciones radial y cubital o las posturas de flexión y extensión.

	precision	recall	f1-score	support
0	0.7557	0.7860	0.7706	2551
1	0.0000	0.0000	0.0000	222
2	0.5252	0.6822	0.5935	214
3	0.4752	0.3032	0.3702	221
4	0.4936	0.6920	0.5762	224
5	0.5075	0.6081	0.5533	222
6	0.5470	0.5689	0.5577	225
7	0.6000	0.2500	0.3529	12
accuracy			0.6782	3891
macro avg	0.4880	0.4863	0.4718	3891
weighted avg	0.6422	0.6782	0.6569	3891

Figura 2. Reporte de clasificación

Además, el reporte de clasificación (ver figura 2) mostró valores de precisión y recall equilibrados entre clases, lo que sugiere que el modelo no presenta un sesgo fuerte hacia categorías específicas.

Finalmente, se almacenaron tanto el modelo entrenado (EMGLSTMModel.pt) como el objeto de normalización (scaler.save) para su uso en tareas de inferencia. Estos elementos fueron integrados en un cuaderno de Google Colab diseñado con ipywidgets, el cual permite cargar un archivo .txt, aplicar la misma normalización utilizada durante el entrenamiento y ejecutar el modelo en modo de evaluación (eval mode). La interfaz muestra la clase predicha, la etiqueta real y un botón para consultar las curvas de aprendizaje, facilitando la interpretación de los resultados.

4.1. TRADUCCIÓN DE COLAB A PYTHON

Tras la finalización de la primera versión del modelo LSTM en Google Colab, se realizó una migración completa del proyecto a un entorno local con el propósito de mejorar la reproducibilidad y facilitar la ejecución del modelo en diferentes equipos sin dependencia directa de servicios en la nube. Esta adaptación no modificó la lógica interna del pipeline, la arquitectura ni los hiperparámetros del modelo, sino únicamente la estructura de los paths de acceso a datos, modelos y objetos serializados. De este modo, las rutas empleadas en Colab (/content/drive/MyDrive/...) fueron reemplazadas por referencias relativas dentro de carpetas locales (Dataset/, Modelos/, Scalers/). Esta reorganización permite que el código pueda ejecutarse de forma autónoma en cualquier entorno de desarrollo compatible con Python 3.10+ y las librerías listadas en el archivo requirements.txt.

En el entorno local, el proyecto se dividió en dos scripts principales:

1. modelo.py, encargado del entrenamiento, evaluación y guardado del modelo, y
2. interfaz.py, responsable de la carga del modelo entrenado y del despliegue de una interfaz gráfica para predicciones.

Ambos scripts conservan la misma estructura modular previamente desarrollada en los notebooks de Colab, garantizando compatibilidad y coherencia entre versiones. La clase EMGLSTMClassifier se mantiene idéntica, al igual que los procesos de normalización, segmentación por ventanas y cálculo de métricas.

La interfaz gráfica fue hecha con la librería Tkinter, la cual sustituye la versión basada en ipywidgets de Colab. Desde una ventana principal, el usuario puede seleccionar un archivo .txt de señales EMG, ejecutar la predicción del gesto dominante y consultar los resultados de clasificación junto con la clase predicha y la real. Además, puedes consultar las gráficas de curva de aprendizaje con un botón. El script principal guarda estas curvas en un archivo comprimido (training_curves.npz) que almacena los valores de loss de entrenamiento, accuracy de validación y exactitud final de prueba. Desde la interfaz, el botón "Mostrar curvas de aprendizaje" permite cargar y graficar estos datos de manera dinámica, desplegando dos paneles: la evolución de la pérdida por época y la variación de la exactitud en validación junto con la línea de referencia del accuracy en el conjunto de prueba.

Finalmente, se verificó la consistencia de los resultados entre las ejecuciones en Colab y en el entorno local. Las métricas finales fueron reproducibles, manteniendo un accuracy en el conjunto de prueba del 67.82% y un comportamiento análogo en la matriz de confusión y el reporte de clasificación.

5. CONCLUSIÓN DE LA PRIMERA VERSIÓN DEL MODELO

Los resultados obtenidos en la primera versión del modelo LSTM permiten evaluar de manera crítica el desempeño inicial del sistema de clasificación de gestos a partir de señales electromiográficas (EMG). Tras el proceso de entrenamiento de 30 épocas, la red alcanzó una precisión promedio del 67.82% en el conjunto de prueba. Aunque el modelo logra distinguir correctamente las clases más frecuentes y de mayor amplitud electromiográfica, tiene dificultades en la identificación de gestos menos frecuentes, lo cual se refleja en la dispersión de las métricas de precisión y recall entre categorías como se ve en la tabla de la figura 2.

El análisis detallado del reporte de clasificación muestra que las clases con mayor número de ejemplos, como la clase 0 (relacionada con la posición de reposo muscular), alcanzaron valores de precisión y recall superiores al 75%, con un F1-score promedio de 0.77. A diferencia de los gestos de baja frecuencia, como la clase 7 que esos obtuvieron valores más bajos (precisión de 0.60 y recall de 0.25), mientras que algunas clases como la 1 no lograron ser predichas correctamente. Este comportamiento probablemente se debe a que hay bastante desequilibrio en la distribución de clases (donde la clase 0 concentra más de la mitad de las muestras totales) y a la variabilidad en la morfología de las señales EMG para gestos de menor intensidad o corta duración. Esto hace que la red empiece a favorecer patrones dominantes durante la optimización.

Durante el entrenamiento, la evolución de la pérdida (loss) y la precisión de validación nos mostró un proceso de aprendizaje estable dentro de lo que cabe, sin indicios muy contundentes de sobreajuste. La pérdida promedio de entrenamiento bajó de 1.10 a 0.64, mientras que la exactitud de validación aumentó poco a poco desde 65.4% hasta valores cercanos al 69% hacia las últimas épocas. Estos resultados sugieren que el modelo logró capturar una estructura discriminativa general de las señales, aunque su capacidad representacional puede verse limitada por la profundidad reducida de la red (una sola capa recurrente) y la tasa de dropout efectiva (inactiva en la capa LSTM al tener solo una capa).

En términos cualitativos, la matriz de confusión (figura 1) mostró un reconocimiento correcto de los gestos que son más dominantes, pero hay confusiones entre pares de movimientos electromiográficamente similares, como las desviaciones radial y cubital o las posturas de flexión y extensión.

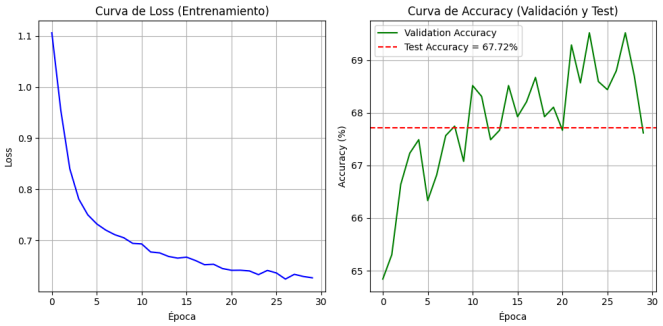

Figura 3. Curvas de pérdida y exactitud obtenidas durante el entrenamiento de la primera versión del modelo LSTM.

La Figura 3 muestra las curvas generadas por el modelo, donde puede observarse una disminución sostenida de la función de pérdida y un incremento gradual de la exactitud de validación a lo largo de las 30 épocas. Ambas curvas evidencian un proceso de aprendizaje estable, sin fluctuaciones abruptas, lo cual corrobora que el modelo alcanzó un punto de convergencia adecuado sin sobreajuste notable. Esta evidencia visual refuerza las conclusiones obtenidas mediante las métricas cuantitativas descritas en la sección anterior.

En el futuro, se considera la implementación de estrategias de regularización y ajuste arquitectónico para poder mejorar la discriminación de las clases de este dataset.

6. IMPLEMENTACIÓN, EJECUCIÓN Y ANÁLISIS DEL MODELO

   Para el modelo de mejora

7. CONCLUSIÓN DE LA SEGUNDA VERSIÓN DEL MODELO

   Para el modelo de mejora

8. COMPARATIVA ENTRE EL PRIMER MODELO Y EL SEGUNDO MODELO

   Para el modelo de mejora

   Ejemplo de tabla.

Métrica	Modelo Manual (Lineal)	Modelo Manual Mejorado (L2)	Modelo Manual Mejorado (Huber + L2 + PCA)	Modelo Framework (Random Forest)
MAE (Test)				
R^2 (Test)				
Bias (Test)				
Varianza (Test)				
MAE (Val)				
R^2 (Val)				

Figura 23. Tabla comparativa de métricas modelo manual vs modelo manual con L2 vs modelo manual con Huber + L2 + PCA vs Modelo Framework

Referencias

[1] Krilova, N., Kastalskiy, I., Kazantsev, V., Makarov, V., & Lobov, S. (2018). EMG Data for Gestures [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C5ZP5C.