HeClaMoSTC

Herramienta de Clasificación de Movimientos para Síndrome del Túnel Carpiano

Manual de Usuario para Descarga e Instalación

Autora:
Karen Nicolle Arango Valencia

Universidad:
Pontificia Universidad Javeriana - Cali

Fecha: Noviembre 2025

Índice

1. Introducción

1.1. ¿Qué hace HeClaMoSTC?

HeClaMoSTC (Herramienta de Clasificación de Movimientos para Síndrome del Túnel Carpiano) es un sistema inteligente de clasificación automática diseñado para detectar movimientos de riesgo asociados al Síndrome del Túnel Carpiano (STC) mediante el análisis de señales electromiográficas (EMG) de superficie.

El sistema analiza señales EMG de 12 canales y clasifica automáticamente cada movimiento en dos categorías: **SEGURO** y **RIESGO**, en función de patrones aprendidos a partir de la base de datos NinaPro DB2 (Ejercicio B).

Clasificación de movimientos

- **Movimientos de RIESGO** (asociados a sobrecarga biomecánica en el túnel carpiano):
  - Movimiento 13: Flexión de muñeca.
  - Movimiento 14: Extensión de muñeca.
  - Movimiento 15: Desviación radial.
  - Movimiento 16: Desviación ulnar.
- **Movimientos SEGUROS**:
  - Movimientos 1–12: Diversos agarres y gestos de mano.
  - Movimiento 17: Extensión de muñeca con mano cerrada (considerado seguro en este contexto).

1.2. Características principales

HeClaMoSTC ofrece las siguientes características:

- **Interfaz web intuitiva**: no requiere conocimientos de programación; toda la interacción se realiza a través de un navegador web.
- **Múltiples modelos**:
  - Machine Learning (ML): Ensemble KNN y Random Forest.
  - Deep Learning (DL): CNN + LSTM + Attention y BiLSTM + Attention.

- **Sistema Dual**: combina dos modelos especializados para mejorar la detección:
  - *Especialista SAFE*: optimizado para detectar movimientos seguros.
  - *Especialista RISK*: optimizado para detectar movimientos de riesgo.
- **Pipeline automático**:
  - Filtrado (Butterworth 20–450 Hz + notch 50 Hz).
  - Normalización Z-score.
  - Segmentación en ventanas.
  - Extracción de características (para ML).
  - Clasificación y agregación de resultados.
- **Visualización**:
  - Gráficas de señales EMG.
  - Estadísticas por ventana.
  - Probabilidades de clasificación.
  - Niveles de confianza y detalle del Sistema Dual.
- **Compatibilidad con archivos propios**: permite cargar archivos `.mat` con señales EMG personalizadas.

2. Requisitos del sistema

2.1. Requisitos de hardware

Mínimo:

- Procesador dual-core a 2.0 GHz o superior.
- 4 GB de RAM.
- 1 GB de espacio libre en disco..

Para entrenamiento de modelos:

- GPU NVIDIA con soporte CUDA (opcional, se usa Google Colab).
- 16 GB de RAM.
- 10 GB de espacio libre en Google Drive.

2.2. Requisitos de software

Sistema operativo: El sistema fue desarrollado en Windows 11. En caso de usar macOS o Linux se deben revisar la compatibilidad de paquetes y comandos.

Python:

- Versión 3.8, 3.9, 3.10 o 3.11.
- `pip` instalado.

Conexión a Internet:

- Necesaria para instalación de dependencias.
- Necesaria para entrenamiento en Google Colab.
- Opcional para el uso local de la aplicación (una vez instalada).

2.3. Dependencias de Python

Las siguientes librerías se instalan desde `requirements.txt`:

Cuadro 1: Dependencias principales de HeClaMoSTC

Librería	**Versión**	**Propósito**
Flask	3.0.0	Servidor web backend
Flask-CORS	4.0.0	Manejo de CORS
TensorFlow	2.15.0	Modelos de Deep Learning
Keras	2.15.0	API de alto nivel para DL
scikit-learn	1.3.2	Modelos de Machine Learning
NumPy	1.24.3	Cálculo numérico
SciPy	1.11.4	Procesamiento de señales
pandas	2.1.4	Manejo de datos
imbalanced-learn	0.11.0	Técnicas de balanceo de clases
PyWavelets	1.5.0	Análisis wavelet
matplotlib	3.8.2	Gráficas (notebooks)
seaborn	0.13.0	Visualización avanzada

3. Descarga e instalación

3.1. Descargar desde GitHub

Paso 1: acceder al repositorio

Abre el navegador y visita:

https://github.com/proyectoSTC/HeClaMoSTC

Paso 2: descargar el código

Opción A: descarga directa (usuarios sin Git)

a) Haz clic en el botón verde `Code`.

b) Selecciona `Download ZIP`.

c) Guarda el archivo e inmediatamente después descomprímelo.

d) Mueve la carpeta a una ruta conocida, por ejemplo:

- Windows: `C:\Users\TuUsuario\Documentos\HeClaMoSTC`
- macOS: `/Users/TuUsuario/Documentos/HeClaMoSTC`
- Linux: `/home/TuUsuario/Documentos/HeClaMoSTC`

Opción B: usando Git (usuarios avanzados)

```
git clone https://github.com/proyectoSTC/HeClaMoSTC.git
cd HeClaMoSTC
```

3.2. Instalación en Windows

Paso 1: verificar Python

Abre `CMD` y ejecuta:

```
python --version
```

Deberías ver algo como `Python 3.10.x`. Si no está instalado:

- Descarga desde https://www.python.org/downloads/.
- Marca la casilla `Add Python to PATH`.
- Completa la instalación y reinicia CMD.

Paso 2: ir a la carpeta del proyecto

```
cd C:\Users\TuUsuario\Documentos\HeClaMoSTC
```

Paso 3: crear entorno virtual (recomendado)

```
python -m venv venv
venv\Scripts\activate
```

Paso 4: instalar dependencias

```
pip install -r requirements.txt
```

Si hay errores, actualiza `pip` y repite:

```
python -m pip install --upgrade pip
pip install -r requirements.txt
```

Paso 5: verificación rápida

```
python -c "import flask, tensorflow, sklearn; print('Instalación exitosa')"
```

3.3. Estructura de carpetas

Tras la instalación, la estructura recomendada es:

```
HeClaMoSTC/

 frontend/              (Interfaz web)
    index.html
    app.js

 models/                (Modelos entrenados)
    ensemble_knn.pkl
```

```
    random_forest.pkl
    cnn_lstm_attention.keras
    bilstm_attention.keras
    scaler.pkl
    metadata.json

signals/                (Señales de prueba)
    *.mat

notebooks/              (Notebooks de Colab/Jupyter)
    HeClaMoSTC.ipynb
    EMG_Spectral_Analysis.ipynb

venv/                   (Entorno virtual)
server.py               (Servidor backend Flask)
requirements.txt        (Dependencias)
README.md               (Documentación)
```

Importante:

- Copiar `index.html` y `app.js` a la carpeta `frontend/`.
- Copiar todos los modelos (`.pkl`, `.keras`) junto con `scaler.pkl` y `metadata.json` a `models/`.
- Copiar señales de prueba a `signals/`. Para descargar las señales utilizadas en el proyecto hacerlo a traves del link: https://drive.google.com/drive/folders/1VizpksUwSnVtW46p8OzCbKDbdEurR-s?usp=sharing. (Contiene la carpeta Train, para usar el Colab y la carpeta Test para probar la App)

Sin `scaler.pkl` ni `metadata.json` el sistema no podrá funcionar correctamente.

4. Uso de la aplicación web

4.1. Iniciar el servidor

Paso 1: activar entorno virtual

Windows:

```
venv\Scripts\activate
```

Paso 2: ejecutar el servidor

```
python server.py
```

Deberías ver algo similar a:

```
======================================================================
SERVIDOR CLASIFICADOR STC
======================================================================

 Rutas: C:\Users\Karen\HeClaMoSTC\models
 Window: 500ms, Overlap: 25.0%

 Sistema Dual DISPONIBLE:
   SAFE: RandomForest (precision 0.952)
   RISK: Ensemble_KNN (recall 0.897)

 http://localhost:5000
======================================================================

 * Serving Flask app 'server'
 * Debug mode: on
 * Running on http://0.0.0.0:5000
```

Paso 3: abrir el navegador

Visita:

```
http://localhost:5000
```

Notas:

- Mantener la terminal abierta mientras se usa la aplicación.
- Para detener el servidor: `Ctrl + C`.
- Si el puerto 5000 está ocupado, se puede cambiar en `server.py`.

4.2. Interfaz principal

La interfaz se organiza en varias secciones:

- **Encabezado**: título y breve descripción del sistema.
- **Modo de clasificación**:
  - **Modelo independiente**: permite seleccionar un único modelo (ML o DL).
  - **Sistema Dual**: combina automáticamente dos modelos ML especialistas.
- **Selección de modelo**:
  - Tipo de modelo: ML o DL.
  - Modelo específico: `ensemble_knn`, `random_forest`, `cnn_lstm_attention` o `bilstm_attentio`
- **Señales de prueba**:
  - Lista de archivos `.mat` disponibles.
  - Botón para cargar archivos propios.
  - Botón para actualizar la lista.
- **Botón de clasificación**:
  - Se habilita tras elegir modelo y señales.
- **Área de resultados**:
  - Tarjetas de resultados por señal.
  - Gráficas y estadísticas.

4.3. Clasificación paso a paso

Modo independiente

1. Elegir **Modo de Clasificación**: "Modelo Independiente".
2. Elegir **tipo de modelo**: ML o DL.
3. Seleccionar el modelo concreto:
   - ML: `ensemble_knn` o `random_forest`.
   - DL: `cnn_lstm_attention` o `bilstm_attention`.
4. Seleccionar una o varias señales en la lista.
5. Pulsar **"CLASIFICAR SEÑALES"**.
6. Esperar a que se muestren las tarjetas de resultados.

Sistema Dual (recomendado)

1. Elegir **Modo de Clasificación**: "Sistema Dual".
2. Verificar la información del sistema:
   - Modelo especialista SAFE.
   - Modelo especialista RISK.
   - Fecha de los metadatos (`metadata.json`).
3. Seleccionar las señales a clasificar.
4. Pulsar **"CLASIFICAR SEÑALES"**.
5. Revisar:
   - Clasificación final.
   - Nivel de confianza (ALTA / MEDIA).
   - Contribución de cada modelo.

4.4. Carga de señales propias

1. Preparar los archivos `.mat` con la estructura adecuada (ver siguiente sección).
2. En la interfaz, hacer clic en **"Cargar .mat desde tu PC"**.
3. Seleccionar uno o varios archivos.
4. Confirmar la subida.
5. Verificar que aparecen en la lista de señales.
6. Clasificar normalmente.

Los archivos subidos se almacenan en la carpeta `signals/`. Se pueden eliminar manualmente desde el sistema de archivos y luego actualizar la lista desde la interfaz.

4.5. Formato de archivos `.mat`

Los archivos `.mat` deben incluir:

- `emg` (obligatoria):
  - Matriz de tamaño $[n\_muestras \times 12]$.
  - Tipo numérico (`float32`, `float64`, `int16`, etc.).
  - Cada columna es un canal EMG.
- `subject` (opcional): ID del sujeto.
- `stimulus` / `restimulus` (opcional):
  - Etiqueta de movimiento (1–17).
  - 13–16: movimientos RISK.
  - Otros: movimientos SAFE.
- `repetition` / `rerepetition` (opcional): número de repetición (1–6).

Requisitos:

- Frecuencia de muestreo: 2000 Hz.
- 12 canales EMG.

- Duración mínima recomendada: 1 segundo (2000 muestras).

Ejemplo en MATLAB:

```
emg = randn(10000, 12);
subject = 1;
stimulus = 13;
repetition = 1;
save('mi_senal.mat', 'emg', 'subject', 'stimulus', 'repetition');
```

Ejemplo en Python:

```
import numpy as np
from scipy.io import savemat

emg = np.random.randn(10000, 12).astype(np.float32)
subject = 1
stimulus = 13
repetition = 1

savemat('mi_senal.mat', {
    'emg': emg,
    'subject': subject,
    'stimulus': stimulus,
    'repetition': repetition
})
```

5. Entrenamiento de modelos en Google Colab

5.1. Acceder al notebook

1. Abrir https://colab.research.google.com/.
2. Iniciar sesión con una cuenta de Google.
3. Cargar el notebook:
   - Desde GitHub: usar la pestaña GitHub e introducir la URL del repositorio `HeClaMoSTC`.
   - O subir el archivo `HeClaMoSTC.ipynb`.

5.2. Configurar GPU y montar Google Drive

1. En Colab: `Entorno de ejecución` → `Cambiar tipo de entorno de ejecución`.
2. Seleccionar "GPU" en *Acelerador de hardware*.
3. Guardar cambios.

Verificar GPU:

```
!nvidia-smi
```

Montar Google Drive:

```
from google.colab import drive
drive.mount('/content/drive')
```

5.3. Preparar datos en Google Drive

Estructura recomendada:

```
Mi unidad/
 DB2_E1_only/
     train/
        S1_E1_A1.mat
        ...
```

```
test/
    S1_E1_A1.mat
    ...
```

Cada archivo `.mat` debe contener al menos:

- `emg`: $[n\_muestras \times 12]$.
- `restimulus`: vector con el movimiento.
- `rerepetition`: vector con la repetición.

Verificación rápida:

```
from pathlib import Path

data_dir = Path('/content/drive/MyDrive/DB2_E1_only/train')
mat_files = list(data_dir.glob('*.mat'))

print(f"Archivos encontrados: {len(mat_files)}")
for f in mat_files[:5]:
    print(" -", f.name)
```

5.4. Configurar el notebook

Rutas principales

```
class Config:
    BASE_DIR = Path('/content/drive/MyDrive')
    DATA_DIR = BASE_DIR / 'DB2_E1_only'
    TRAIN_DIR = DATA_DIR / 'train'
    SAVE_DIR = BASE_DIR / 'New_ML_DL_models_stc_optimized'
```

Selección de sujetos y modelos

```
USE_ALL_SUBJECTS = True
SELECTED_SUBJECTS = [1,2,3,4,5,6,7,8,9,10,
                     11,12,13,14,15,16,17,18,19,20]

SELECTED_MODELS = ['1', '2', '3', '4']
```

```
# '1': Ensemble KNN
# '2': Random Forest
# '3': CNN+LSTM+Attention
# '4': BiLSTM+Attention
```

Técnicas de balanceo

```
ML_BALANCE_TECHNIQUE = 'adasyn'
DL_BALANCE_TECHNIQUE = 'augment_only'
```

5.5. Ejecutar el entrenamiento

Se puede usar `Entorno de ejecución` → `Ejecutar todas` o ir celda por celda.

El notebook:

- Carga y preprocesa los datos.
- Entrena los modelos ML y DL.
- Genera métricas (accuracy, precision, recall, F1, AUC).
- Guarda los modelos y artefactos.

5.6. Descarga de modelos entrenados

Los modelos se guardan en:

```
Mi unidad/New_ML_DL_models_stc_optimized/run_YYYYMMDD_HHMMSS/
```

Dentro de `artifacts/` se encuentran:

- `ensemble_knn.pkl`
- `random_forest.pkl`
- `cnn_lstm_attention.keras`
- `bilstm_attention.keras`
- `scaler.pkl`
- `metadata.json`

- `results_summary.csv`

Estos archivos deben copiarse a la carpeta `models/` del proyecto local.

5.7. Análisis espectral de señales (opcional)

El notebook `EMG_Spectral_Analysis.ipynb` permite:

- Analizar espectros de frecuencia por canal.
- Comparar movimientos SAFE vs RISK.
- Calcular energía por bandas de frecuencia.
- Generar mapas de calor y medidas de consistencia entre repeticiones.

Se configura de forma similar, ajustando únicamente la ruta de datos.

6. Interpretación de resultados

6.1. Tarjetas de resultados

Cada señal genera una tarjeta con:

- Color de fondo:
  - Rojo: movimiento de RIESGO detectado.
  - Verde: movimiento SEGURO.
- Mensaje principal: alerta o confirmación de seguridad.
- Nivel de confianza (en Sistema Dual).
- Probabilidades de clasificación (modo independiente).
- Estadísticas:
  - Total de ventanas.
  - Ventanas clasificadas como RISK.
  - Ventanas clasificadas como SAFE.
  - Porcentaje de riesgo.
- Gráfica de un canal EMG.
- Metadata (si está disponible): sujeto, movimiento, repetición.

6.2. Métricas de evaluación

Las métricas usadas incluyen:

- **Accuracy**.
- **Precision**.
- **Recall** (sensibilidad para la clase RISK).
- **F1-Score**.
- **AUC** (área bajo la curva ROC).

Para el problema de detección de movimientos de riesgo:

- El **recall** de la clase RISK es crítico (evitar falsos negativos).
- La **precision** también es importante (evitar demasiados falsos positivos).

6.3. Cómo actuar según los resultados

Si la señal se clasifica como RIESGO:

- No es un diagnóstico médico, sino una alerta preventiva.
- Revisar la calidad de la señal y el contexto del movimiento.
- Considerar acciones ergonómicas (cambios de postura, reducción de repetición, pausas activas).
- Si hay síntomas compatibles con STC (dolor, hormigueo, debilidad), acudir a valoración médica.

Si la señal se clasifica como SEGURO:

- El movimiento se considera de bajo riesgo según el modelo.
- Se recomienda mantener buenas prácticas ergonómicas.

En el Sistema Dual:

- Confianza *ALTA*: ambos modelos coinciden (SAFE o RISK).
- Confianza *MEDIA*: desacuerdo entre modelos; el sistema adopta la opción más conservadora (prioriza RISK).

6.4. Aviso legal

HeClaMoSTC es una herramienta de investigación y apoyo, no un dispositivo médico regulado. No sustituye:

- Diagnóstico médico.
- Evaluación clínica.
- Pruebas de laboratorio o estudios neurofisiológicos clínicos.

7. Solución de problemas

7.1. Problemas frecuentes en la aplicación

Problema	**Causa posible**	**Solución**
El servidor no inicia	Puerto 5000 ocupado	Cambiar el puerto en `server.py` (por ejemplo, 5001).
`ModuleNotFoundError`	Dependencias no instaladas	Ejecutar `pip install -r requirements.txt`.
Sistema Dual no disponible	Falta `metadata.json`	Copiar `metadata.json` a la carpeta `models/`.
Modelos DL no cargan	Formato incorrecto (.h5 en lugar de .keras)	Re-entrenar con el notebook actualizado que guarda en formato `.keras`.
Señales no aparecen en lista	Ruta configurada incorrectamente	Verificar y ajustar la ruta de `signals/` en `server.py`.
Error al clasificar	Archivo `.mat` corrupto o sin variable `emg`	Revisar el archivo, comprobar estructura y dimensiones.
Todas las predicciones iguales	Problema con `scaler.pkl`	Asegurarse de usar el `scaler.pkl` generado junto con los modelos.
Interfaz no carga	Faltan archivos `index.html` o `app.js`	Copiarlos a la carpeta `frontend/`.
Botón de clasificar deshabilitado	Faltan selección de modelo o señales	Escoger modo, modelo (si aplica) y al menos una señal.

7.2. Problemas con Google Colab

Error al montar Google Drive:

- Verificar que se ha dado permiso correctamente.
- Probar en ventana de incógnito.
- Volver a ejecutar el montaje.

Out of Memory:

- Reducir `BATCH_SIZE`.
- Usar menos sujetos (`USE_ALL_SUBJECTS = False`).
- Entrenar modelos por separado.

Entrenamiento muy lento:

- Verificar que la GPU esté activa con `!nvidia-smi`.
- Reducir número de épocas.
- Entrenar primero solo los modelos ML.

Runtime desconectado o sesión reiniciada:

- Mantener actividad mínima en la pestaña.
- Aprovechar que los modelos se guardan en disco al finalizar cada bloque.

Apéndice A: Estructura técnica del pipeline

El pipeline principal de HeClaMoSTC aplica las siguientes etapas:

1. Carga del archivo `.mat` y extracción de `emg`.
2. Filtrado pasa-banda Butterworth (20–450 Hz).
3. Filtro notch a 50 Hz.
4. Normalización Z-score por canal.
5. Segmentación en ventanas de 500 ms con 25 % de solapamiento.
6. Extracción de características (para modelos ML).
7. Preparación de secuencias (para modelos DL).
8. Clasificación por ventana.
9. Agregación de resultados por señal.
10. Decisión final (modo independiente o Sistema Dual).

Apéndice B: Especificaciones de modelos

Modelo Ensemble Subspace KNN

- Tipo: K-Nearest Neighbors en ensamble (bagging).
- Vecinos: $k = 7$.
- Número de clasificadores: 30.
- Ventajas: robustez al ruido, buen desempeño en datasets medianos.

Modelo Random Forest

- Tipo: ensamble de árboles de decisión (CART).
- Número de árboles: 200.
- Máxima profundidad: 30.
- Aporta rapidez en inferencia y capacidad de manejar datos desbalanceados con técnicas de sobremuestreo.

Modelo CNN + LSTM + Attention

Arquitectura híbrida que combina:

- Capas convolucionales 1D para capturar patrones locales.
- Capas LSTM bidireccionales para dependencias temporales.
- Mecanismo de atención para ponderar regiones temporales relevantes.
- Capa de salida sigmoide para probabilidad binaria RISK/SAFE.

Modelo BiLSTM + Attention

Similar al anterior pero con énfasis en el modelado secuencial usando BiLSTM y atención, sin convoluciones iniciales, lo que le da otra forma de capturar la dinámica temporal.