



Extracción de características de EEG para detección de estrés

Integrantes:

- Leidy Tairo
- Ingrid Licota
- Sofía Nieto
- Marco Ichillumpa
- Josue Taipe

Introducción

El **estrés** es un problema creciente en Perú, agravado por la **pandemia**, que afecta la salud mental, el bienestar social y la productividad laboral, haciendo urgente la implementación de **métodos accesibles y precisos para su detección y manejo**.



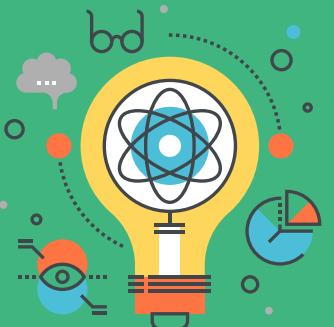
Puntos clave

Impacto Local

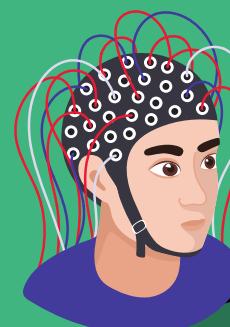


60%

Necesidad de Innovación



Propuesta Tecnológica



Más del **60%** de los peruanos experimenta estrés de manera frecuente, principalmente por factores económicos, laborales y de salud.

Los métodos tradicionales son limitados en precisión y accesibilidad; un enfoque basado en EEG ofrece una alternativa no invasiva y económica.

El modelo utiliza señales EEG y aprendizaje automático, adaptado al contexto peruano, para detectar el estrés de manera temprana, precisa y accesible.

PROBLEMÁTICA

Exámenes, tareas acumulativas y proyectos generan **presión**.



La constancia de carga conduce a un **estrés**



Para cumplir sus plazos se **sacrifican horas de sueño** generando agotamiento mental y físico



El decaimiento de las capacidades cognitivas como el nivel de atención o la retención de información resultan en **fatiga mental**.



PROBLEMÁTICA

FACTORES QUE CONTRIBUYEN AL NIVEL DE FATIGA DIARIO [1]

TABLE III
MULTIPLE REGRESSION FIT FOR STANDARDIZED DAILY FATIGUE
LEVEL AS THE OUTCOME VARIABLE

Factor	Value	Standard Error	t-value	p-value
Days from semester start	7.023×10^{-5}	20.817×10^{-5}	0.33	0.736
Stress	0.521	0.018	28.24	< 0.001***
Sleepiness	0.211	0.019	11.06	< 0.001***
Sleep quality	-0.059	0.020	-2.97	0.003**
Mood	0.036	0.023	1.60	0.109
Alertness	-0.137	0.023	-5.90	< 0.001***
Sleep duration	-0.057	0.019	-3.05	0.002**

[1] O. Komarov, L.-W. Ko, and T.-P. Jung, "Associations among emotional state, sleep quality, and Resting-State EEG Spectra: a longitudinal study in graduate students," IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 28, no. 4, pp. 795–804, Feb. 2020, doi: 10.1109/tnsre.2020.2972812.

ESTRÉS  PRIVACIÓN DEL SUEÑO

CANSANCIO EXCESIVO

SOMNOLENCIA

POCA
CONCENTRACIÓN

BAJO RENDIMIENTO
ACADÉMICO

CRISIS MENTAL

ABANDONO DEL
ESTUDIO

ESTOS FACTORES A SU VEZ GENERAN MÁS ESTRÉS,
POTENCIANDO EL CICLO

PROBLEMÁTICA

EL ESTRÉS ACADÉMICO, ASOCIADO A FACTORES COMO LAS ALTAS DEMANDAS COGNITIVAS, IRREGULARIDADES EN EL SUEÑO Y EMOCIONES NEGATIVAS, ESTÁ GENERANDO CONSECUENCIAS SIGNIFICATIVAS EN ESTUDIANTES UNIVERSITARIOS, COMO LA DISMINUCIÓN DEL RENDIMIENTO ACADÉMICO EN CONJUNTO DE PROBLEMAS DE SALUD FÍSICA Y MENTAL.

Propuesta de solución



Extracción de características de señales EEG adquiridas de estudiantes universitarios para clasificarlas en estados de 'Estresado' y 'No estresado', utilizando técnicas de aprendizaje automático (ML) basadas en energías relativas de subbandas de frecuencia.

Metodología

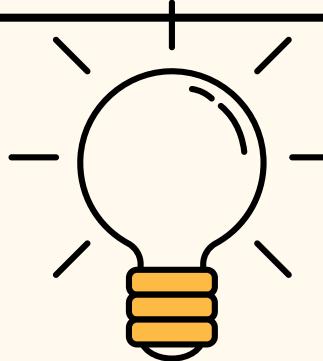
Paper de referencia:

ECS Transactions, 107 (1) 1857-1865 (2022)
10.1149/10701.1857ecst ©The Electrochemical Society

Analysis and classification of stress among students using EEG as biomarker

V.G.Rajendran^{a*}, S.Jayalalitha^b, K.Adalarasu^c, G.Usha^d

1 SELECCIÓN DE PARTICIPANTES



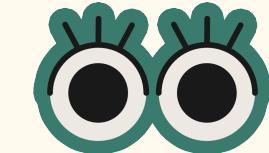
- Formulario con preguntas en torno a la evaluación del estrés del participante
- Elección de estudiantes entre 19 años a 25 años de edad



2 ADQUISICIÓN DE SEÑALES

- Se plantean 2 escenarios diferentes: Antes y después del examen.
- Se utiliza el Ultracortex junto con su software OpenBCI para adquirir datos en tiempo real.
- Se registrarán los datos en las siguientes actividades:

Fase de intervalo controlado



- 10 segundos con los ojos abiertos
- 10 segundos con los ojos cerrados

Fase de estudio

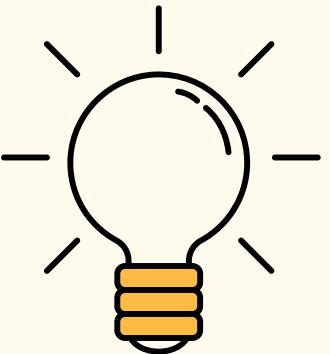


- 160 segundos para estudiar

Fase de evaluación



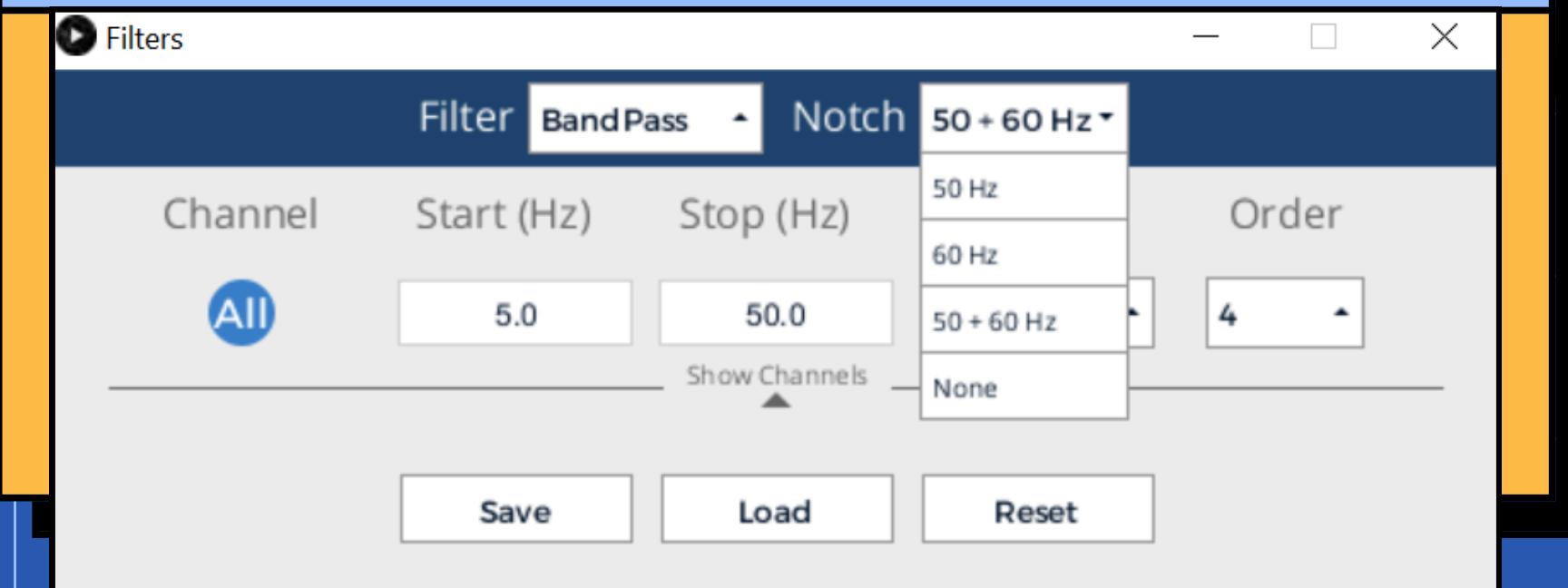
- 3 minutos de aritmética mental



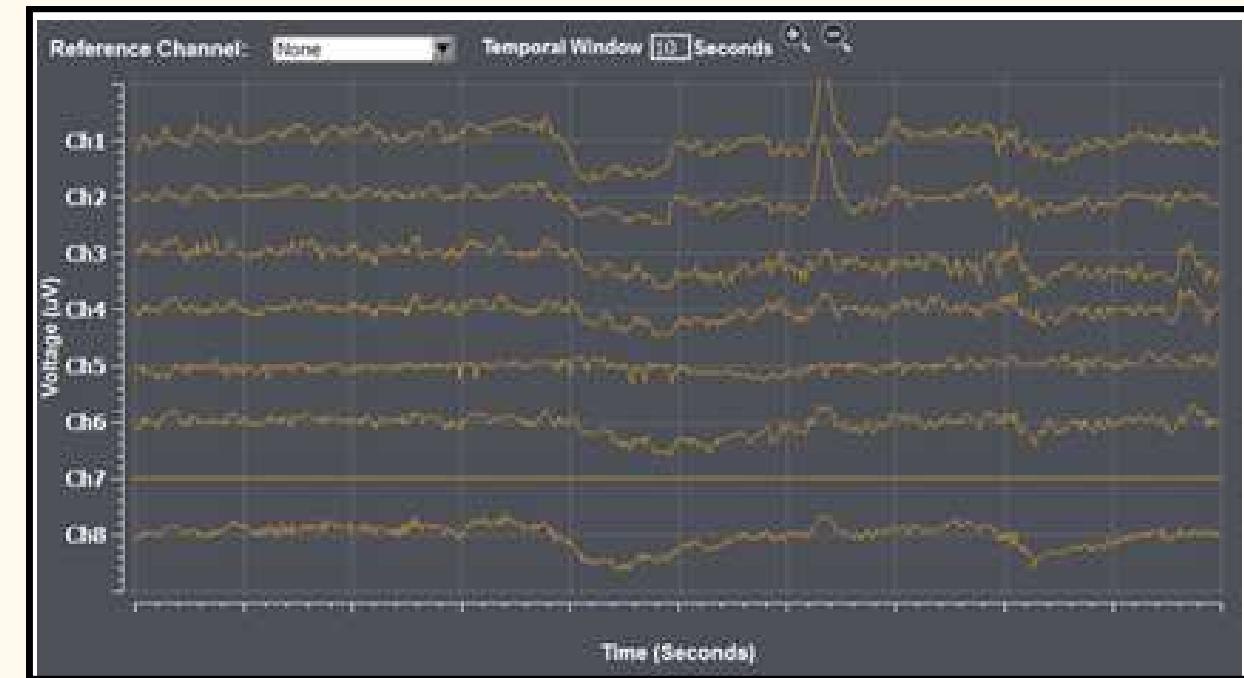
Metodología

Mediante el uso del software OpenBCI enlazado al Ultracortex, se filtran las señales EEG con un pasabanda (0.1 Hz a 40 Hz) para eliminar ruido de baja y alta frecuencia.

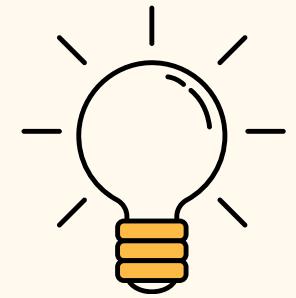
3 VISUALIZACIÓN Y FILTRADO DE LA SEÑAL



Se separan las señales EEG en componentes independientes para identificar artefactos/componentes no neuronales, como actividad ocular (EOG) o muscular (EMG).



4 APLICACIÓN DE ANÁLISIS DE COMPONENTES INDEPENDIENTES (ICA)



Metodología

Extracción de características:

- Las señales EEG se descompusieron en seis niveles utilizando Wavelet Packet Decomposition con la función Daubechies 4 (db4) como wavelet madre.
- Se analizaron las bandas theta (4-8 Hz), alpha (8-16 Hz) y beta (16-32 Hz), y se extrajeron sus coeficientes detallados (D6, D5 y D4 respectivamente).

Normalización de datos:

- Los valores de energía de las bandas fueron normalizados en el rango de 0 a 1 para asegurar consistencia en los datos.

5

NORMALIZACIÓN Y ALINEACIÓN DE LA SEÑAL

Sub-band Energy

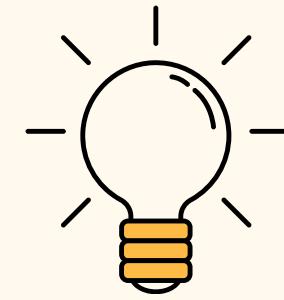
Relative wavelet packet energy

$$E(D_k) = \sum_{j=1}^N |D_{kj}|^2$$
$$p_k = \frac{E_k}{E_T}$$

EEG Band	Frequency Range	Wavelet Packet Decomposition Coefficient
Total bandwidth of EEG signal	0-32 Hz	A(3,1)
Theta band	4-8 Hz	D(6,1)
Alpha band	8-16 Hz	D(5,1)
Beta band	16-32 Hz	D(4,1)
Delta band	0-4 Hz	A(6,1)

- Tabla 1: Rangos de frecuencia y sus coeficientes aproximados (A) y detallados (D) extraídos de la descomposición en paquetes de wavelets

Metodología



6 CLASIFICACIÓN

Los datos normalizados de las energías relativas (theta, alpha y beta) fueron utilizados para la clasificación en dos estados:

- Estrés: Señales antes del examen.
- No estrés: Señales después del examen.

Se usa el asistente de Edge Impulse para entrenar el modelo con los datos etiquetados.

Implementa el modelo en un dispositivo que soporte el procesamiento de las señales EEG en tiempo real.

CLASIFICACIÓN MEDIANTE EDGE IMPULSE

The screenshot shows the Edge Impulse web interface. At the top, it displays 'DATA ACQUISITION (ANOMALY DETECTION)' with tabs for 'Training data', 'Test data', and 'Export data'. It shows '10m 45s' of data collected and a 'TRAIN / TEST SPLIT' of '79% / 21%'. Below this is a table titled 'Collected data' listing 18 samples, each 3 seconds long and labeled 'Mode 1' or 'Mode 2'. To the right, there's a 'Record new data' section with fields for 'Device' (set to 'My device'), 'Label' ('Mode 1'), 'Sample length (ms.)' (9000), 'Sensor' ('Built-in accelerometer'), and 'Frequency' (100Hz). A 'Start sampling' button is present. At the bottom, a 'RAW DATA' section shows a graph for 'Mode 1.2nebrb8o.s22' with three traces: accX (red), accY (green), and accZ (blue), plotted against time from 0 to 2808 ms.

NOTA: IMAGEN REFERENCIAL

Prototipo

APLICACIÓN DE ICA

```
# Crear objeto Raw de MNE
ch_names = ['Fp1', 'Fp2', 'C3', 'C4', 'P3', 'P4', 'O1', 'O2'] # Nombres válidos
ch_types = ['eeg'] * 8
info = create_info(ch_names=ch_names, sfreq=Fs, ch_types=ch_types)
raw = RawArray(signals_filtered_notch, info)

montage = make_standard_montage('standard_1020')
raw.set_montage(montage)

# Filtrado adicional
raw.filter(l_freq=1.0, h_freq=None)

# Realizar ICA
ica = ICA(n_components=8, random_state=97, max_iter='auto', method='picard')
ica.fit(raw)

# Ploteo de fuentes y componentes ICA
ica.plot_sources(raw, title=f"Fuentes ICA - {nombres[idx]}")
ica.plot_components(title=f"Componentes ICA - {nombres[idx]}", picks=range(8))

# Detectar automáticamente artefactos musculares
muscle_idx_auto, scores = ica.find_bads_muscle(raw)
ica.plot_scores(scores, exclude=muscle_idx_auto, title=f"Scores de Artefactos Musculares - {nombres[idx]}")

# Mostrar propiedades y excluir componentes
ica.plot_properties(raw, picks=muscle_idx_auto, log_scale=True)
ica.exclude = muscle_idx_auto

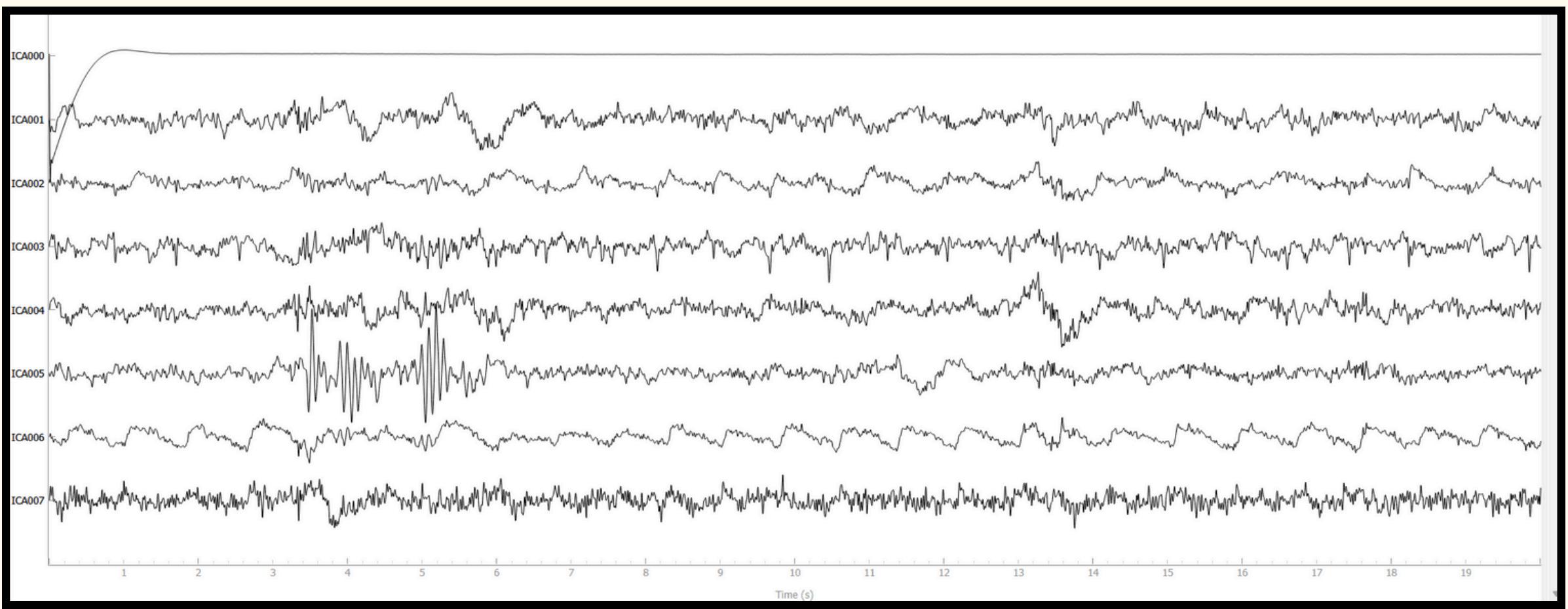
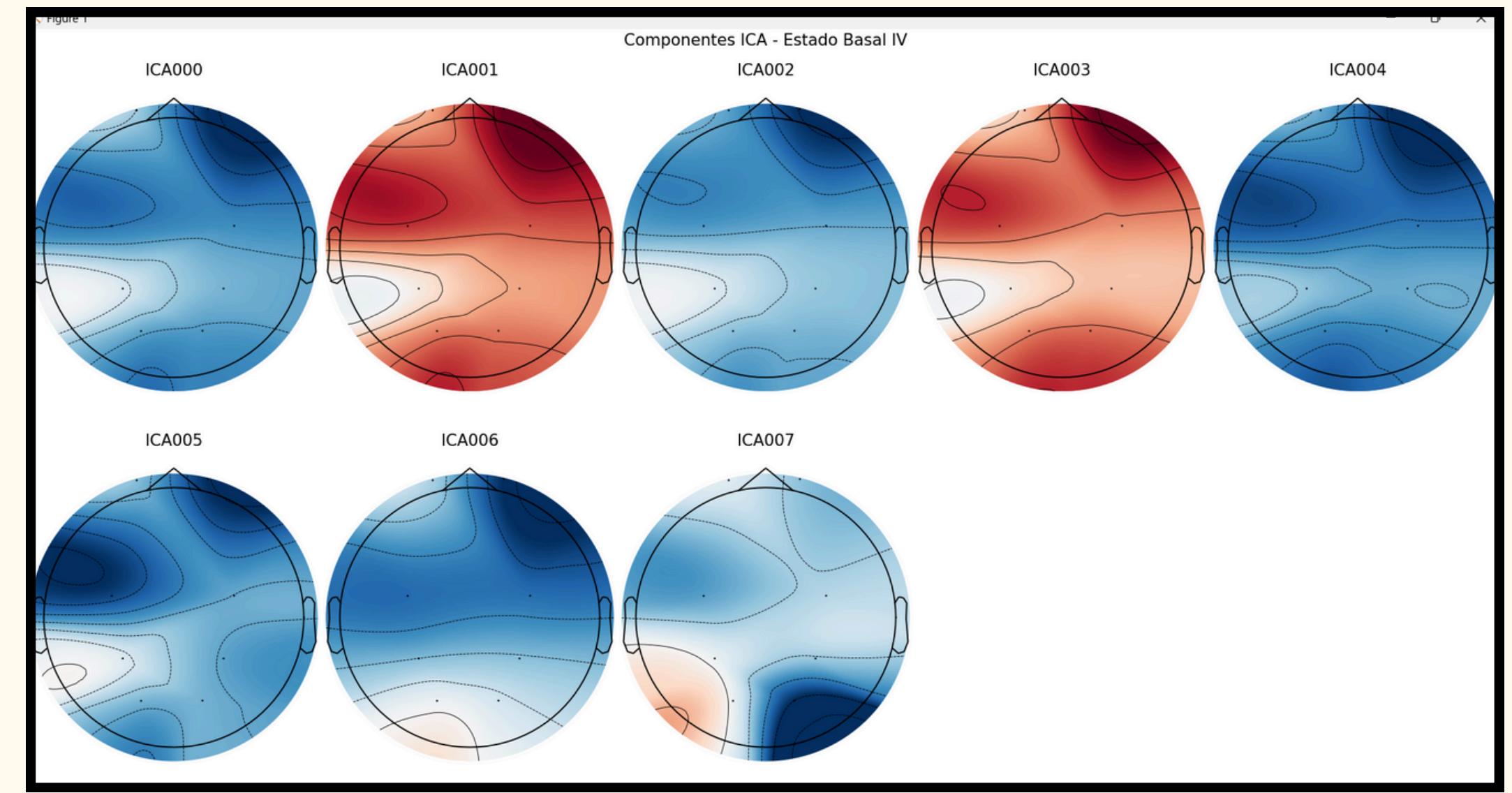
# Aplicar ICA para corregir los datos
raw_corrected = ica.apply(raw.copy())

# Ploteo antes y después de la corrección
ica.plot_overlay(raw, exclude=ica.exclude, title=f"Señales Antes y Despues de ICA - {nombres[idx]}")

# Calcular PSD de un canal corregido
signal_corrected = raw_corrected.get_data()[0, :]
```

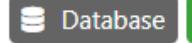
Resultados

APLICACIÓN DE ICA



Base de datos:

<https://physionet.org/content/eegmat/1.0.0/>

 Database  Open Access

EEG During Mental Arithmetic Tasks

Igor Zyma , Ivan Seleznov , Anton Popov , Mariia Chernykh , Oleksii Shpenkov 

Published: Dec. 17, 2018. Version: 1.0.0

Descripción

- Sistema utilizado: Neurocom EEG de 23 canales (XAI-MEDICA, Ucrania).
- Grabaciones de EEG de sujetos antes y durante la realización de tareas de aritmética mental
- Filtros aplicados:
 - Filtro paso alto: Frecuencia de corte de 30 Hz.
 - Filtro notch: Eliminación del ruido de línea de alimentación a 50 Hz.
- Segmentos EEG procesados:
 - Duración: 60 segundos.
- Libre de artefactos gracias al uso de Análisis de Componentes Independientes (ICA) para eliminar artefactos oculares, musculares y cardíacos.

- Criterios de inclusión:
 - Agudeza visual normal o corregida y visión en color normal.
 - Ausencia de manifestaciones clínicas de deterioro mental o cognitivo, y de discapacidades de aprendizaje verbal o no verbal.
- Criterios de exclusión:
 - Uso de medicamentos psicoactivos.
 - Adicción a drogas o alcohol.
 - Quejas psiquiátricas o neurológicas.

Pre-procesamiento

Lectura de EDF

```
import pyedflib
import numpy as np
import matplotlib.pyplot as plt

# Cargar el archivo EDF
file_name = "C:\\\\Users\\\\Ingrid\\\\Downloads\\\\eeg-during-mental-arithmetic-tasks-1.0.0\\\\eeg-during-mental-arithmetic-tasks-1.0.0\\\\Subject08_1.edf"
f = pyedflib.EdfReader(file_name)

# Obtener información básica del archivo
n_signals = f.signals_in_file # Número de señales en el archivo
signal_labels = f.getSignalLabels() # Nombres de las señales
sample_frequency = f.getSampleFrequency(0) # Frecuencia de muestreo de

# Leer las señales y almacenarlas
sigbufs = []
for i in range(n_signals):
    sigbufs.append(f.readSignal(i))

# Cerrar el archivo
f._close()

# Graficar las señales
plt.figure(figsize=(12, 8))
for i in range(n_signals):
    plt.plot(sigbufs[i], label=f"Señal {i + 1}: {signal_labels[i]}")

plt.title("Señales del archivo EDF")
plt.xlabel("Tiempo (muestras)")
plt.ylabel("Amplitud")
plt.legend(loc='upper right')
plt.grid()
plt.show()
```

Transformar de EDF a CSV

```
file_name = "C:\\\\Users\\\\Ingrid\\\\Downloads\\\\eeg-during-mental-arithmetic-tasks-1.0.0\\\\eeg-during-mental-arithmetic-tasks-1.0.0\\\\Subject08_1.edf"
output_csv = "Antes_examen_1.csv" # Nombre del archivo CSV de salida

# Leer el archivo EDF
f = pyedflib.EdfReader(file_name)

# Obtener información básica del archivo
n_signals = f.signals_in_file # Número de señales
signal_labels = f.getSignalLabels() # Etiquetas de las señales
sample_freqs = [f.getSampleFrequency(i) for i in range(n_signals)] # Frecuencias de muestreo
signal_data = []

# Leer y almacenar las señales
for i in range(n_signals):
    signal_data.append(f.readSignal(i))

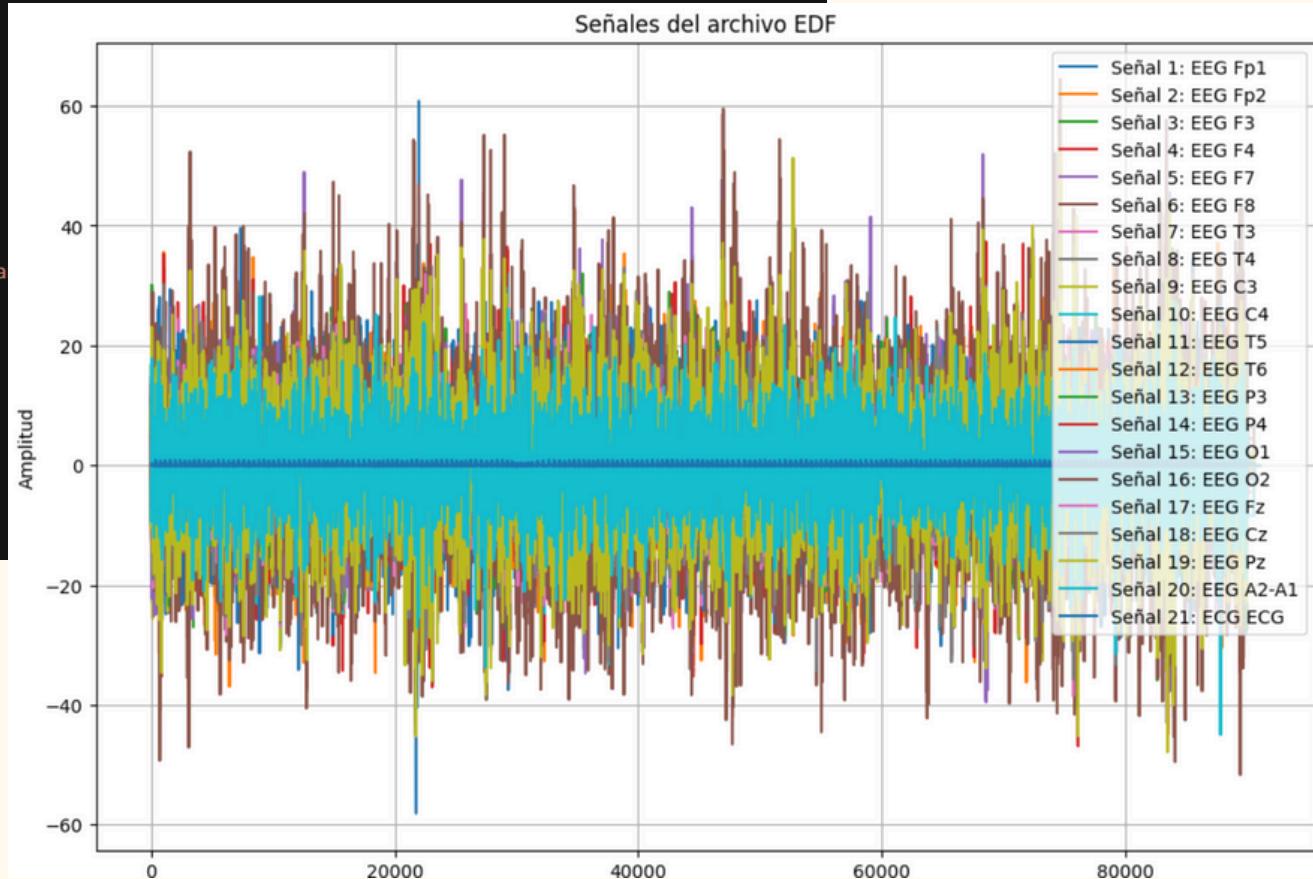
# Cerrar el archivo EDF
f._close()

# Crear un DataFrame de pandas
# Asegurar que todas las señales tengan la misma longitud
max_length = max(len(data) for data in signal_data)
for i, data in enumerate(signal_data):
    if len(data) < max_length:
        signal_data[i] = np.pad(data, (0, max_length - len(data)), 'constant')

# Convertir los datos a un DataFrame
df = pd.DataFrame(np.array(signal_data).T, columns=signal_labels)

# Guardar en un archivo CSV
df.to_csv(output_csv, index=False)

print(f"Archivo CSV guardado en: {output_csv}")
print("Ruta actual de trabajo:", os.getcwd())
```



Prototipo:

Obtención de energía de sub-banda de la base de datos y datos recopilados: (Usar Edge impulse)

```
import numpy as np
import pywt
import matplotlib.pyplot as plt

# Simulación de datos EEG
def simulate_eeg_data(num_samples=6, num_channels=8, duration=3, fs=128):
    """
    Simula datos EEG con forma (num_samples, num_channels, duration * fs).
    """
    time_points = duration * fs
    data = np.random.randn(num_samples, num_channels, time_points)
    return data

# Filtrado de bandas (opcional, aquí se simula sin cambios en los datos)
def bandpass_filter(data, low_freq=0.1, high_freq=40, fs=1000):
    """
    Aplica un filtro paso banda. Aquí se simula sin modificar la señal.
    """
    return data
```

```
# Extracción de características basada en bandas de frecuencia
def extract_wavelet_features(data, wavelet='db4', levels=6):
    """
    Extrae características basadas en las bandas de frecuencia Delta, Theta, Alpha y Beta,
    utilizando la descomposición wavelet.
    """
    features = []
    for signal in data:
        # Descomposición wavelet en niveles especificados
        coeffs = pywt.wavedec(signal, wavelet, level=levels)

        delta_energy = np.sum(np.square(coeffs[6])) # A(6,1)
        theta_energy = np.sum(np.square(coeffs[6])) # D(6,1)
        alpha_energy = np.sum(np.square(coeffs[5])) # D(5,1)
        beta_energy = np.sum(np.square(coeffs[4])) # D(4,1)
        total_energy = np.sum(np.square(coeffs[3])) # A(3,1)

        # Cálculo de energías relativas
        relative_energies = [
            delta_energy / total_energy,
            theta_energy / total_energy,
            alpha_energy / total_energy,
            beta_energy / total_energy,
        ]
        features.append(relative_energies)

    return np.array(features)

# Flujo principal
if __name__ == "__main__":
    # Simulación de datos EEG
    eeg_data = simulate_eeg_data(num_samples=6, num_channels=8, duration=5, fs=1000)

    # Procesamiento
    filtered_data = bandpass_filter(eeg_data)

    # Extracción de características
    features = []
    for student_data in filtered_data:
        student_features = extract_wavelet_features(student_data)
        features.append(np.mean(student_features, axis=0))
    features = np.array(features)

    # Resultados parciales
    print("Características extraídas por estudiante (energías relativas):")
    for i, f in enumerate(features):
        print(f"Estudiante {i+1}: Delta={f[0]:.4f}, Theta={f[1]:.4f}, Alpha={f[2]:.4f}, Beta={f[3]:.4f}")
```

GRACIAS