RECONOCIMIENTO DE EMOCIONES & CREACIÓN DE EMOCIONES SINTÉTICAS & VISUALIZACIÓN

ALBERTO PADILLA & MARCO A. HERNANI

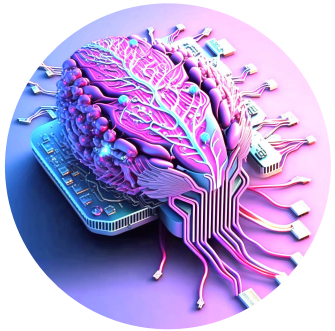
Assembler Institute of Technology

BCMI Brain-like Computing & Machine Intelligence

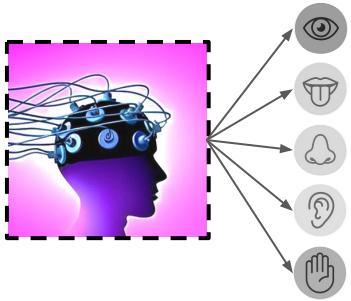
ÍNDICE

Assembler
Institute of Technology


COMENZAMOS

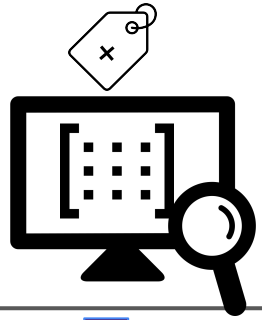


1. Objetivo nº1: Reconocimiento de Emociones

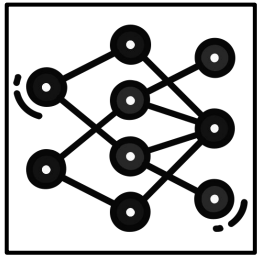


ETL

# mean_0_a	mean_1_a	mean_2_a	mean_3_a	mean_4_a
4.62E+00	3.03E+01	-3.56E+02	1.56E+01	2.63E+01
2.88E+01	3.31E+01	3.20E+01	2.58E+01	2.28E+01
8.90E+00	2.94E+01	-4.16E+02	1.67E+01	2.37E+01
1.49E+01	3.16E+01	-1.43E+02	1.98E+01	2.43E+01
2.83E+01	3.13E+01	4.52E+01	2.73E+01	2.45E+01
3.10E+01	3.09E+01	2.96E+01	2.85E+01	2.40E+01
1.08E+01	2.10E+01	4.47E+01	4.87E+00	2.81E+01
1.78E+01	2.78E+01	-1.02E+02	1.69E+01	2.69E+01
1.15E+01	2.97E+01	3.49E+01	1.02E+01	2.69E+01
8.91E+00	2.92E+01	-3.14E+02	6.51E+00	3.09E+01
5.21E+00	2.84E+01	1.85E+01	3.66E+00	2.26E+01

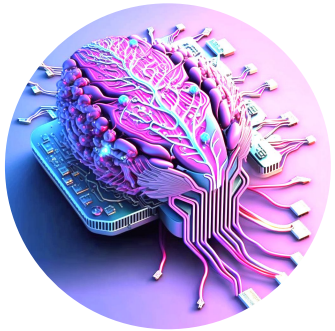


CNN

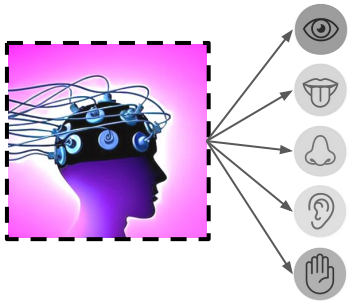


% PRECISIÓN

Assembler
Institute of Technology

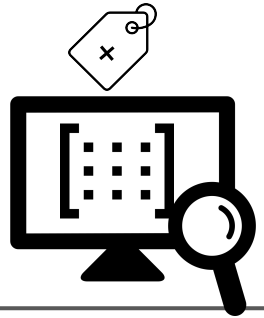


1. Objetivo nº2: Creación de Señales Sintéticas

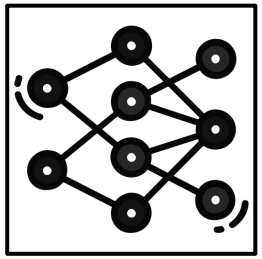


ETL

# mean_0_a	mean_1_a	mean_2_a	mean_3_a	mean_4_a
4.62E+00	3.03E+01	-3.56E+02	1.56E+01	2.63E+01
2.88E+01	3.31E+01	3.20E+01	2.58E+01	2.28E+01
8.90E+00	2.94E+01	-4.16E+02	1.67E+01	2.37E+01
1.49E+01	3.16E+01	-1.43E+02	1.98E+01	2.43E+01
2.83E+01	3.13E+01	4.52E+01	2.73E+01	2.45E+01
3.10E+01	3.09E+01	2.96E+01	2.85E+01	2.40E+01
1.08E+01	2.10E+01	4.47E+01	4.87E+00	2.81E+01
1.78E+01	2.78E+01	-1.02E+02	1.69E+01	2.69E+01
1.15E+01	2.97E+01	3.49E+01	1.02E+01	2.69E+01
8.91E+00	2.92E+01	-3.14E+02	6.51E+00	3.09E+01
5.21E+00	2.84E+01	1.85E+01	3.66E+00	2.26E+01



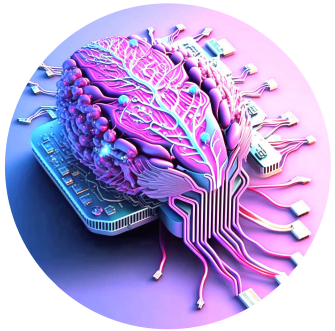
GAN



SEÑAL ARTIFICIAL

VISUALIZACIÓN

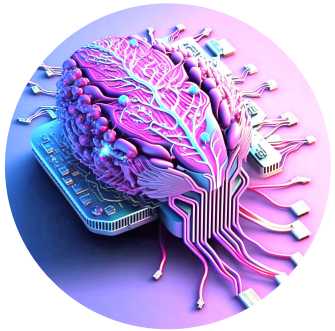
Assembler Institute of Technology



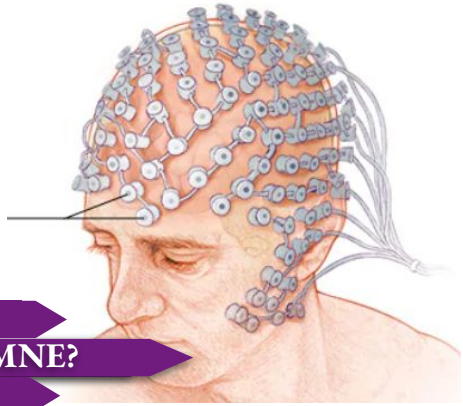
1. Motivación

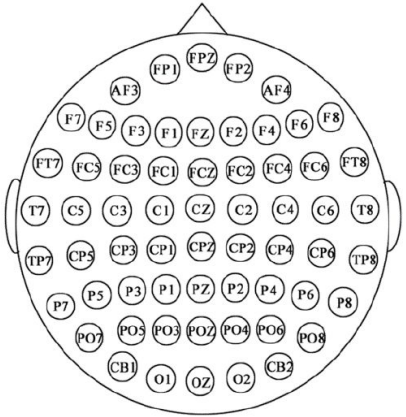
- Transformar los pensamientos a formato digital.
- Conocer el estado de un paciente sin necesidad de hablar.
- Ayudar la comunicación con extremidades artificiales.
- Realidad Virtual en los videojuegos.
- Estímulos artificiales para enfermedades neurodegenerativas como el Alzheimer y la demencia.

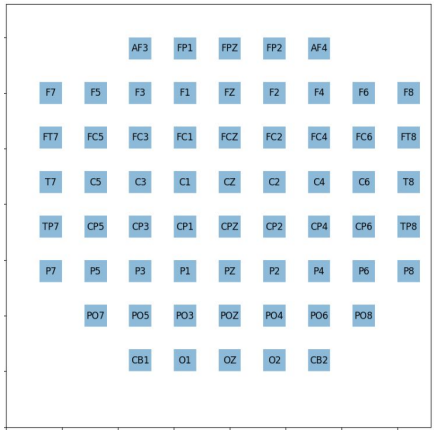
Assembler
Institute of Technology



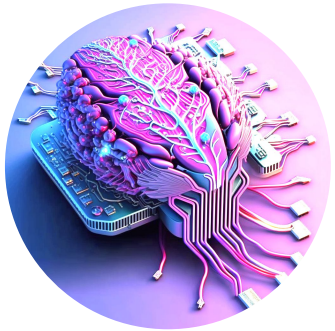
2. ¿Qué tipos de datos son EEG?



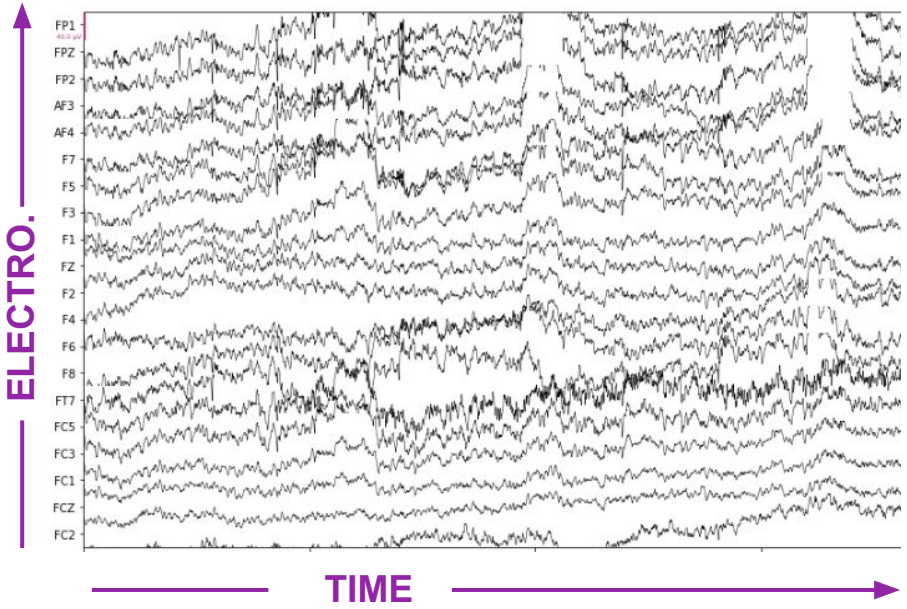




Assembler
Institute of Technology

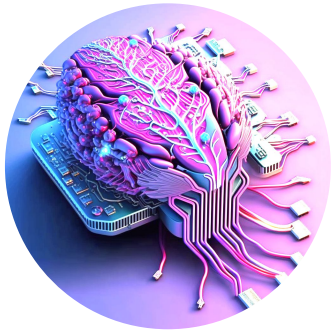


2. ¿Qué tipos de datos son EEG?

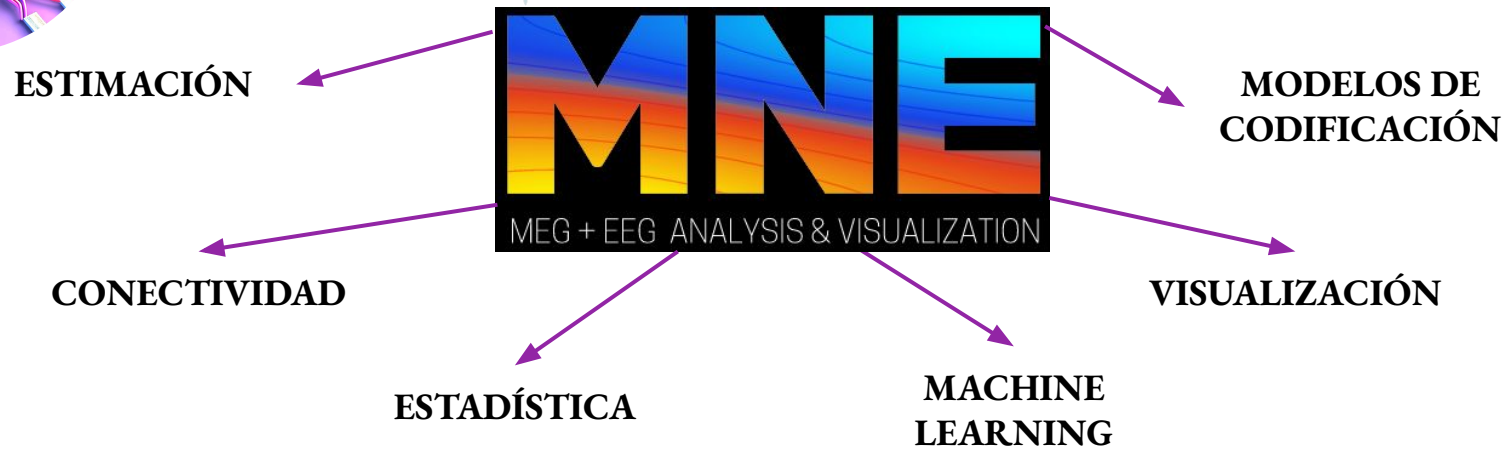



```
<RawCNT | 6_3_20180802.cnt, 66 x 3117520 (3117.5 s), ~68 kB, data nc

<Info | 8 non-empty values
 bads: []
 ch_names: FP1, FPZ, FP2, AF3, AF4, F7, F5, F3, F1, FZ, F2, F4, F6,
 chs: 66 EEG
 custom_ref_applied: False
 highpass: 0.0 Hz
 lowpass: 500.0 Hz
 meas_date: 2018-02-08 03:46:51 UTC
 nchan: 66
 projs: []
 sfreq: 1000.0 Hz
 subject_info: 5 items (dict)
```

Assembler
Institute of Technology

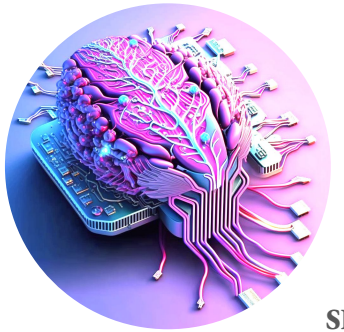


2. Librería MNE



- Paquete Python de código abierto para explorar, visualizar y analizar datos neurofisiológicos humanos: MEG, **EEG**, sEEG, ECoG, NIRS...

Assembler Institute of Technology



3. DataSets.

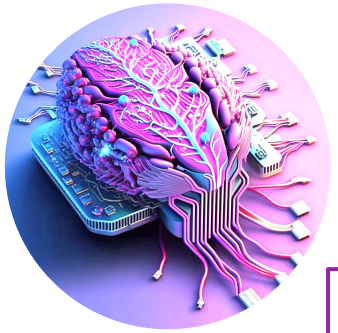
SEED: EEG de **15 sujetos**. Los datos se recogieron mientras veían fragmentos de películas. Los vídeos se seleccionaron cuidadosamente para inducir distintos tipos de emociones: positivas, negativas y neutras.

SEED_V: EEG de **20 sujetos**. Evolución del conjunto de datos original de SEED. El número de categorías de emociones cambia a cinco: feliz, triste, miedo, asco y neutro.

SEED_GER: EEG de **8 sujetos** alemanes con etiquetas emocionales positivas, negativas y neutras.

SEED_FRA: EEG de **8 sujetos** franceses con etiquetas emocionales positivas, negativas y neutras.

Assembler Institute of Technology



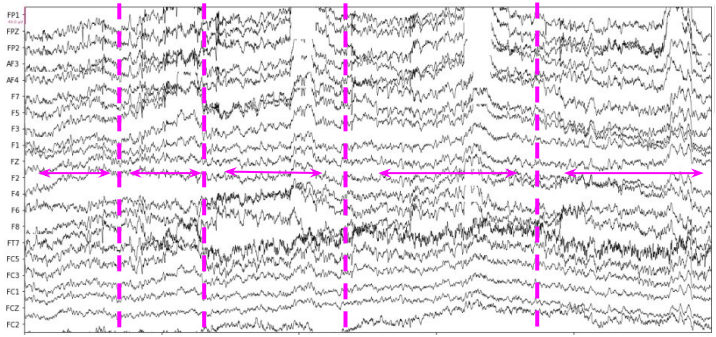
3. DataSets.

Descarga:

- 1_1_20180804.cnt
- 2_1_20180416.cnt
- 3_1_20180414.cnt
- 4_1_20180414.cnt
- 5_1_20180719.cnt
- 6_1_20180713.cnt
- 7_1_20180411.cnt

```
<RawCNT | 6_3_20180802.cnt, 66 x 3117520 (3117.5 s), ~68 kB, data not loaded>

<Info | 8 non-empty values
 bads: []
 ch_names: FP1, FPZ, FP2, AF3, AF4, F7, F5, F3, F1, FZ, F2, F4, F6, F8, ...
 chs: 66 EEG
 custom_ref_applied: False
 highpass: 0.0 Hz
 lowpass: 500.0 Hz
 meas_date: 2018-02-08 03:46:51 UTC
 nchan: 66
 projs: []
 sfreq: 1000.0 Hz
 subject_info: 5 items (dict)
```

Trial-1 Trial-2 Trial-3 … Trial-n



Datos totales: 1 sujeto - 1 experimento

```
(62, 3117520)
```

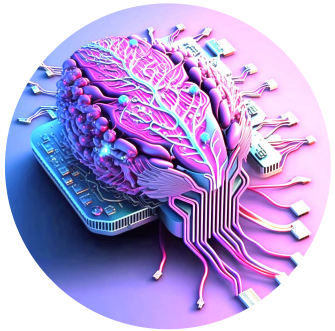
Datos cada Estímulo: 1 sujeto - 1 experimento

```
(62, 291000)
(62, 52000)
(62, 178000)
...
```

- 1_3_20180808-Trial-1.parquet
- 1_3_20180808-Trial-2.parquet
- 1_3_20180808-Trial-3.parquet

- 1_1-Trial-1_POSITIVE.parquet
- 1_1-Trial-6_POSITIVE.parquet
- 1_1-Trial-9_POSITIVE.parquet

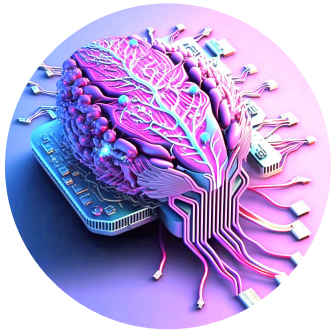
Assembler Institute of Technology



4. Incidencias DataSets

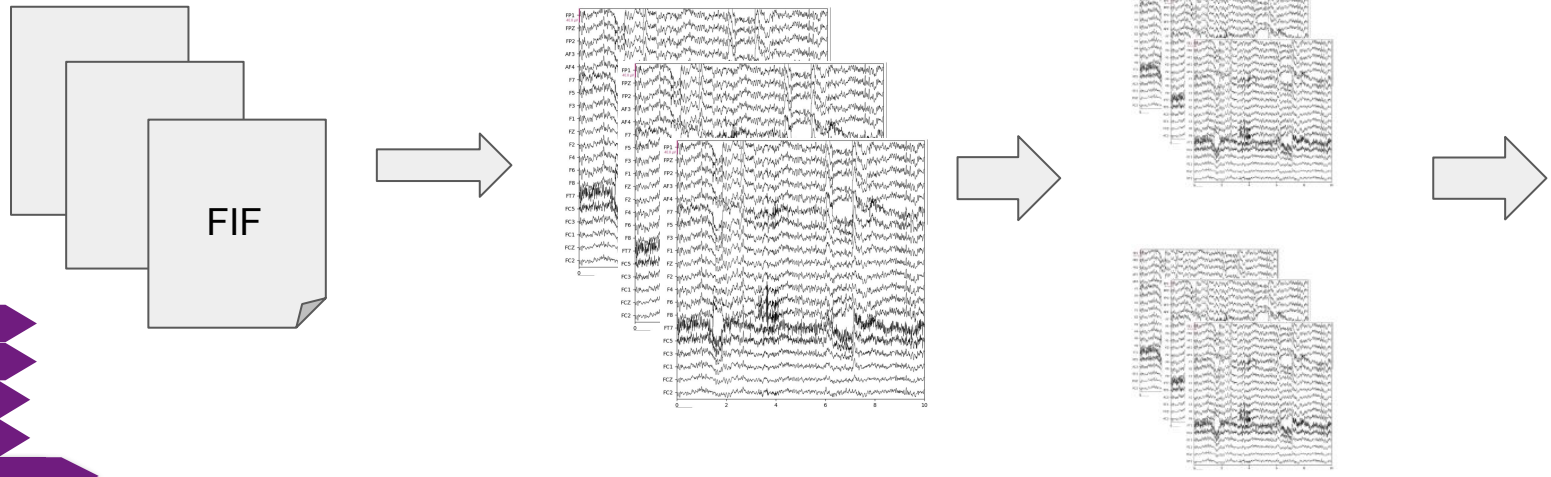
- Investigación de los experimentos realizados para conformar el datasets. Adaptación en algunos casos.
- Acceso a Datos. (Univ. Harvard X - Univ. Shangai ✓ - Otros ✓)
- Compatibilidad de DataSets
- Tamaño de los archivos (sol: csv. (±30gB) - .parquet (±3Gb) - .fif (±3Gb))
- Comparar matrices: = nº columas ≠ nº filas .

Assembler
Institute of Technology

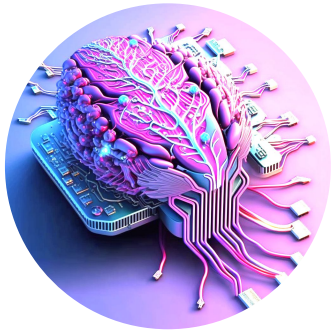


5. Proceso de ETL:

○ Trial general

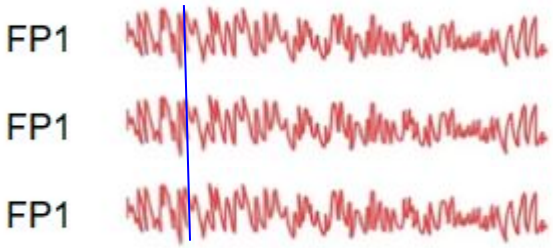


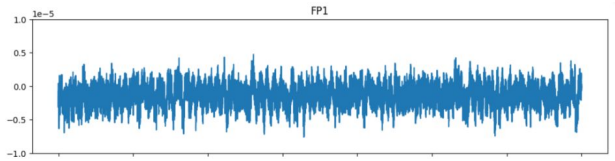
Assembler
Institute of Technology



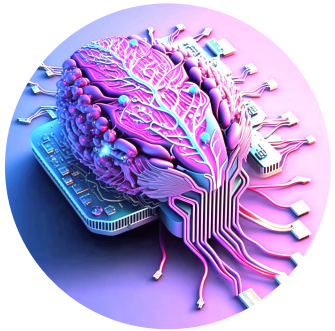
5. Proceso de ETL:

○ Trial General



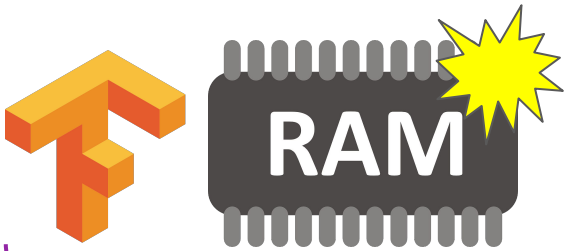


Assembler
Institute of Technology



5. Proceso de ETL:

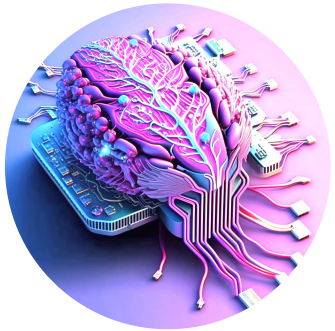
○ Interpolación




```
(62, 291000)
(62, 52000)
(62, 178000)
```

(62,50000)

Assembler
Institute of Technology

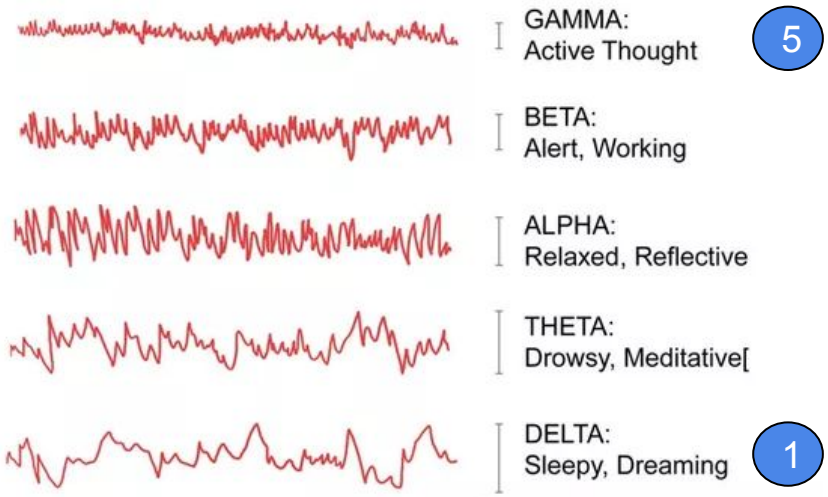


5. Proceso de ETL:

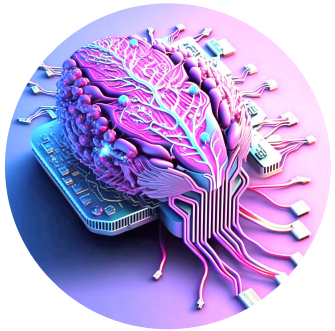
○ Bandas de Frecuencia + 4D

Frecuencia = Velocidad Onda

Bandas = Estados Mentales

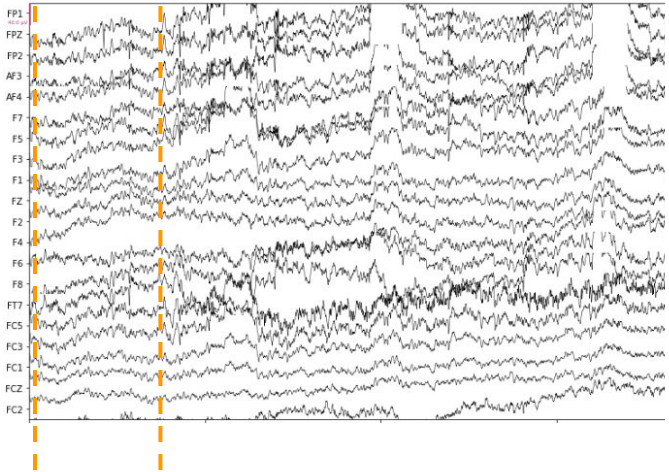


Assembler
Institute of Technology

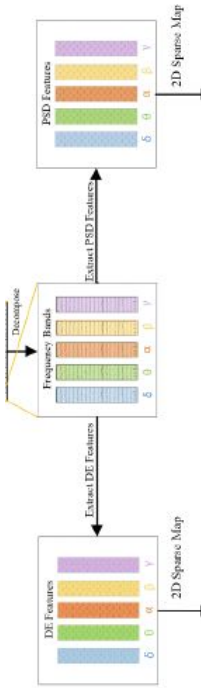


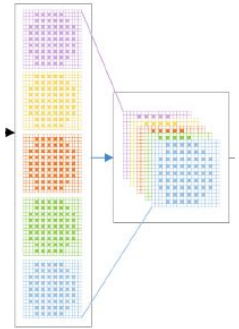
5. Proceso de ETL:

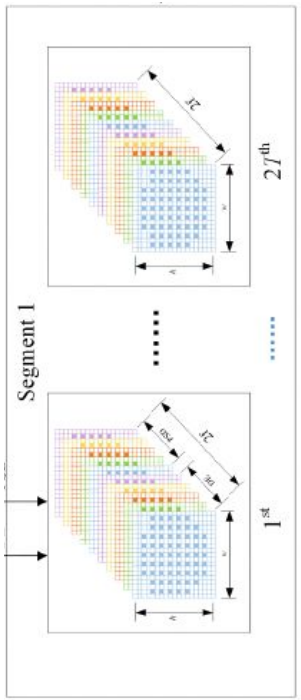
○ Bandas de Frecuencia + 4D

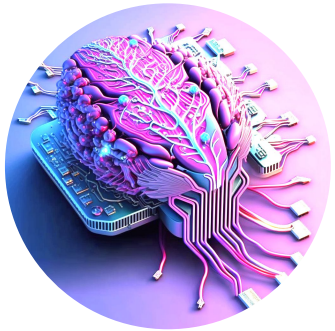


VENTANA = 2s









6. Modelo de Reconocimiento de Emociones (CNN)

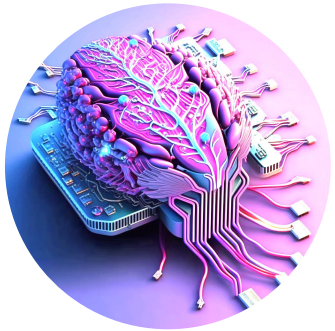
○ Resultados

epochs = 20
batch_size = 32
learning rate=0.001

ACC. TEST: 0.3724

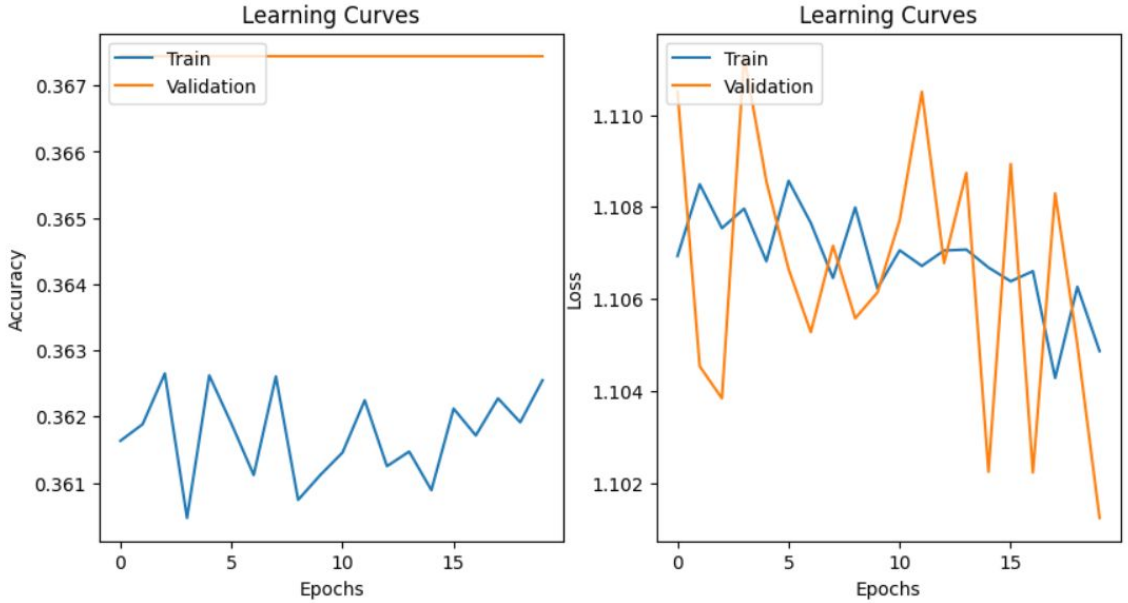
ACC. TRAIN: 0.3625

Assembler Institute of Technology

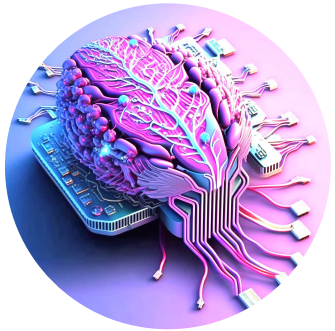


6. Modelo de Reconocimiento de Emociones (CNN)

- Visualización

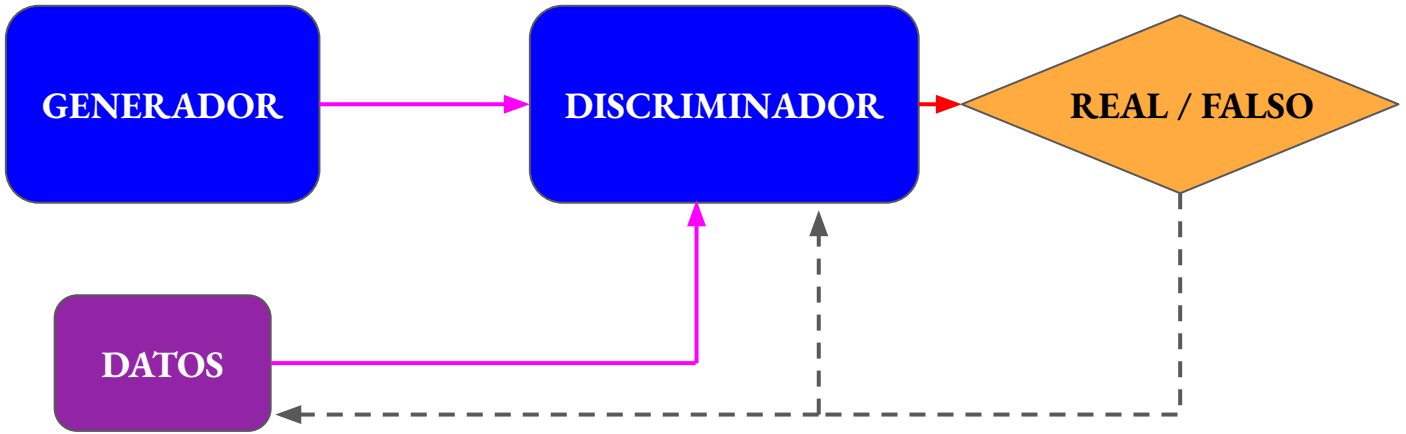


Assembler Institute of Technology

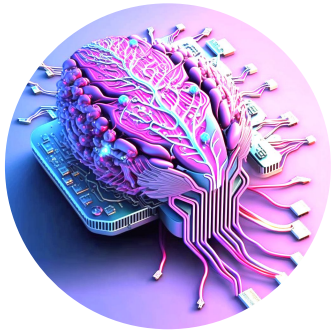


7. Modelo Generador de Señales Sintéticas (GAN)

- ¿Qué es?



Assembler Institute of Technology



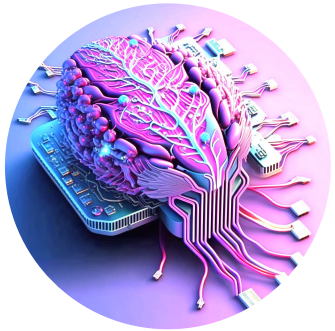
7. Modelo Generador de Señales Sintéticas (GAN)

○ Resultados

epochs = 100
batch_size = 32
learning rate
discriminator=0.0002

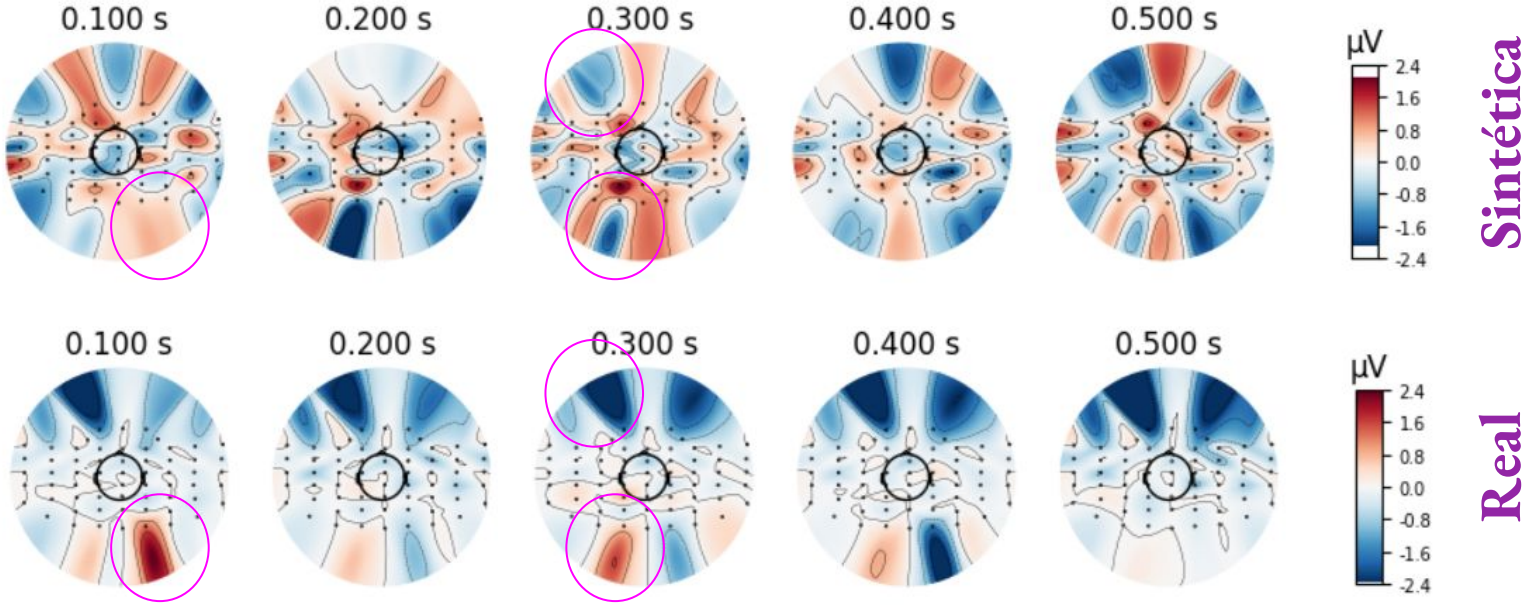
```
[D loss: 3.985933780670166, acc.: 34.375

[G loss: 2.191690444946289]
```

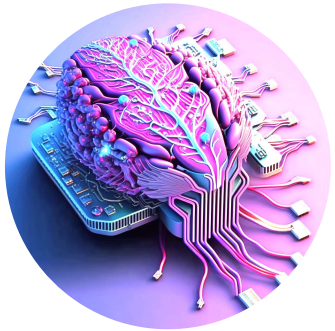
Assembler
Institute of Technology



7. Modelo Generador de Señales Sintéticas (GAN)

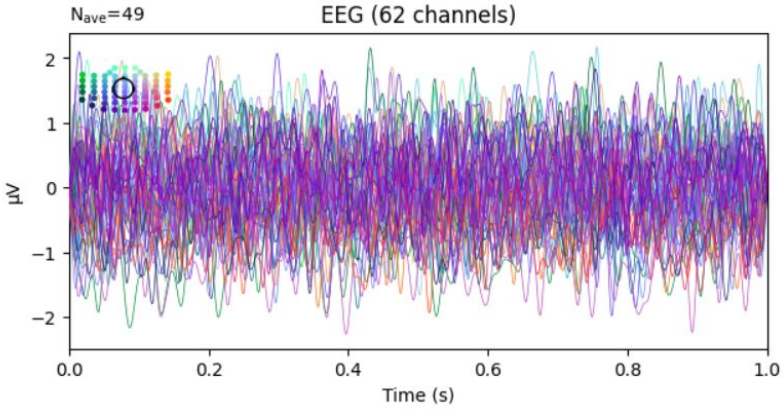
- Visualización (Positivo)



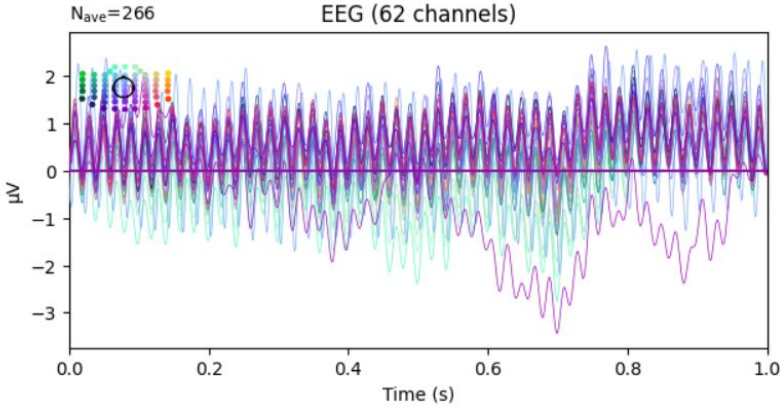


7. Modelo Generador de Señales Sintéticas (GAN)

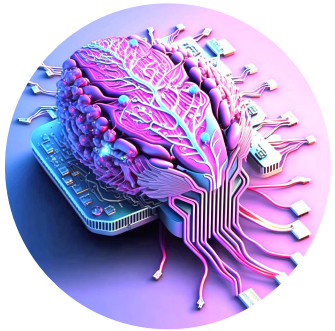
- Visualización (Positivo)



Sintética



Real

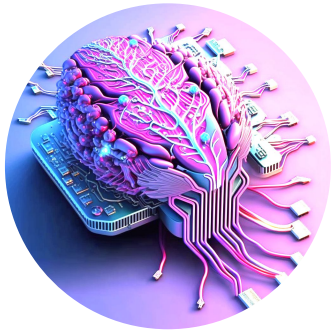


8. Conclusiones

- **Reconocimiento de Emociones (CNN):**
  - El modelo es muy sencillo.
  - Falta de un módulo de atención autoadaptativo formado por dos submódulos, el módulo de atención espacial y el módulo de atención espectral.

- **Creación de Señales Sintéticas (GAN):**
  - Discriminador tiene una alta precisión.
  - La pérdida del generador es alta y todavía se puede mejorar.
  - Gran Coste de Computación.
  - Señales reales y sintéticas muy distintas debido a la interpolación (50s).

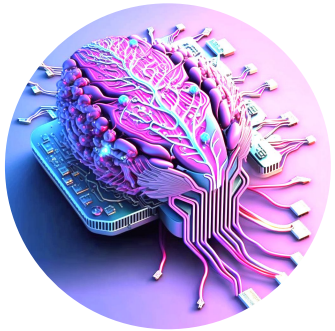
Assembler
Institute of Technology



9. Incidencias Modelos

- Gran cantidad de datos pero poco representativos.
- Alto coste computacional.
- Altos tiempos de ejecución. (7h para ventanas de 2s)
- Pérdida de datos. (interpolación)

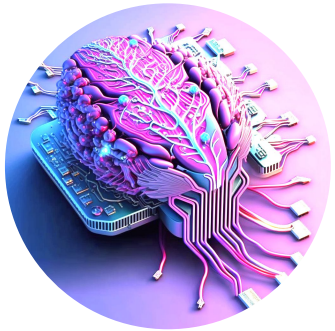
Assembler
Institute of Technology



10. Trabajos Futuros

- **1º Mejorar el modelo de reconocimiento (CNN)**, comprender y añadir el **módulo de atención** adaptativa en el modelo.
- 2º Probar un modelo de **reconocimiento de emociones con las imágenes** de la señal para cada emoción etiquetada.
- 3º Mejorar el **modelo GAN** entrenado con otros datos de entrada (segmentos 4D) o aplicando o**tros métodos** para reducir la información de forma más correcta.
- 4º **Ampliar** la base de datos.
- 5º Intentar **aplicar** esta metodología a **otros campos** donde pueda ser beneficioso reconocer y generar señales eléctricas (comprobación de redes, electrónica...)

Assembler
Institute of Technology



11. Agradecimientos

- **Assembler Institute of Technology:** Por el apoyo aportado durante la ejecución de este projecto, así como durante todo el período lectivo.

- **Universidad de Shangai Jiao Thon:** Agradecer el acceso a sus bases de datos y la ayuda ofrecida para el entendimiento de los archivos.

- **Compañeros de máster:** por estar hombro con hombro estos 6 meses, ayudarnos y buscar consuelo en los momentos bajos.

- **Lucciano Gabinelli:** Por ser el último en pie de los mentores que empezó esta edición. Por saber escuchar, ayudar y aconsejar en todo lo posible, tanto a nivel personal como intelectual.

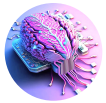
Assembler
Institute of Technology

¿Preguntas?

Assembler
Institute of Technology

Muchas Gracias por su Atención

Assembler
Institute of Technology

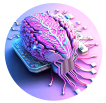


Trial General

Assembler
Institute of Technology

Interpolación

```
# Determine the common length for interpolation
common_length = 50000  # You can adjust this value depending on your needs

interpolated_eeg_data_list = []

for eeg_data in tqdm(eeg_data_list,leave=False):
    num_channels = eeg_data.shape[0]
    original_length = eeg_data.shape[1]
    new_eeg_data = np.zeros((num_channels, common_length))

    for ch_idx in range(num_channels):
        x_original = np.linspace(0, 1, original_length)
        x_new = np.linspace(0, 1, common_length)
        new_eeg_data[ch_idx] = np.interp(x_new, x_original, eeg_data[ch_idx])

    interpolated_eeg_data_list.append(new_eeg_data)
```



Segmentos 4D

```
# Initialize the 4D input with zeros
input_4d_neg = np.zeros((num_segment, height, width, num_maps))

for i, segment in tqdm(enumerate(segment_neg_list), desc="Creating 4D...", leave=False):
    for feature_type_idx, feature_data in enumerate(segment[0]):  # Agregar [0] aquí
        current_feature_data = segment[0][feature_type_idx]
        for channel_idx, channel_name in enumerate(electrode_positions):
            for band_idx, band_name in enumerate(freq_bands.keys()):
                x, y = electrode_positions_matrix[channel_name]
 # Aplanar los datos
                current_feature_data_flat = current_feature_data[band_name].flatten()

                # Aplicar Winsorizing
                current_feature_data_winsorized = mstats.winsorize(current_feature_data_flat, limits=[0.01, 0.01])

                # Remodelar los datos a la forma original
                current_feature_data_winsorized = current_feature_data_winsorized.reshape(current_feature_data[band_name].sh

                # Asignar los datos winsorizados a la matriz de entrada
                input_4d_neg[i, y, x, band_idx + feature_type_idx * len(freq_bands)] = current_feature_data_winsorized[y, x]
```

Assembler
Institute of Technology

CNN

```
model1 = Sequential()

model1.add(Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=input_shape, kernel_regularizer=l2(l2_reg)))
model1.add(BatchNormalization())
model1.add(MaxPooling2D((2, 2)))
model1.add(Dropout(0.25))

model1.add(Conv2D(64, (3, 3), activation='relu', padding='same', kernel_regularizer=l2(l2_reg)))
model1.add(BatchNormalization())
model1.add(MaxPooling2D((2, 2)))
model1.add(Dropout(0.25))

model1.add(Conv2D(128, (3, 3), activation='relu', padding='same', kernel_regularizer=l2(l2_reg)))
model1.add(BatchNormalization())
model1.add(MaxPooling2D((2, 2)))
model1.add(Dropout(0.25))

model1.add(Flatten())

model1.add(Dense(512, activation='relu', kernel_regularizer=l2(l2_reg)))
model1.add(BatchNormalization())
model1.add(Dropout(0.5))

model1.add(Dense(1024, activation='relu', kernel_regularizer=l2(l2_reg)))
model1.add(BatchNormalization())
model1.add(Dropout(0.5))

model1.add(Dense(num_classes, activation='softmax'))

model1.compile(optimizer=Adam(lr=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
```

GAN

```python
def build_generator(latent_dim):
    input_layer = Input(shape=(latent_dim,))

    x = Dense(32)(input_layer)
    x = LeakyReLU(alpha=0.2)(x)
    x = BatchNormalization()(x)

    x = Dense(64)(x)
    x = LeakyReLU(alpha=0.2)(x)
    x = BatchNormalization()(x)

    x = Dense(128)(x)
    x = LeakyReLU(alpha=0.2)(x)
    x = BatchNormalization()(x)

    x = Dense(256)(x)
    x = LeakyReLU(alpha=0.2)(x)
    x = BatchNormalization()(x)

    x = Dense(num_canales * num_puntos_por_canal, activation='tanh')(x)
    output_layer = Reshape((num_canales, num_puntos_por_canal))(x)

    return Model(input_layer, output_layer)
```

```python
# Discriminador
def build_discriminator():
    input_layer = Input(shape=(num_canales, num_puntos_por_canal))
    x = Flatten()(input_layer)
    x = Dense(32)(x)
    x = LeakyReLU(alpha=0.2)(x)
    x = Dropout(0.5)(x)

    x = Dense(64)(x)
    x = LeakyReLU(alpha=0.2)(x)
    x = Dropout(0.5)(x)

    x = Dense(128)(x)
    x = LeakyReLU(alpha=0.2)(x)
    x = Dropout(0.5)(x)
    output_layer = Dense(1, activation='sigmoid')(x)

    return Model(input_layer, output_layer)
```

```python
def build_gan(generator, discriminator):
    z = Input(shape=(latent_dim,))
    generated_eeg = generator(z)
    validity = discriminator(generated_eeg)

    return Model(z, validity)
```

Assembler
Institute of Technology