

The background of the slide is a dense, colorful field of protein ribbon diagrams. The ribbons are rendered in various colors including orange, red, yellow, green, cyan, blue, and purple, creating a vibrant and complex visual texture. The ribbons are intertwined and looped, representing the three-dimensional structure of proteins.

Inteligencia Artificial y Biomedicina: Evolución, actualidad y perspectivas futuras.

Nahuel Costa Cortez

2020 - 2023



Universidad de Oviedo



Doctorado en Inteligencia Artificial

2019 - 2020



Universidad de Oviedo

Máster en Ingeniería Informática

2015 - 2019



Universidad de Oviedo

Grado en Ingeniería Informática



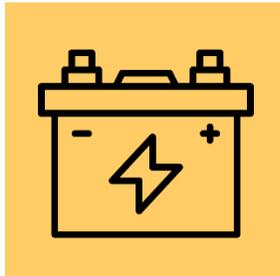
**Profesor en
Universidad de Oviedo**

Investigador en IA



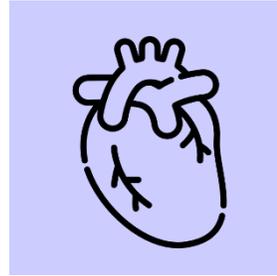
nahuelcosta.com

IA aplicada a monitorización de sistemas



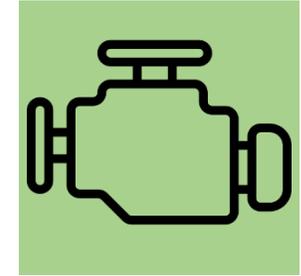
Baterías de Litio

Monitorización de datos
de celdas



Marcapasos

Monitorización de datos
intracardiacos



Motores de aviación

Monitorización de
sensores de aviones

Presentación

Nahuel Costa
Inteligencia Artificial y Biomedicina

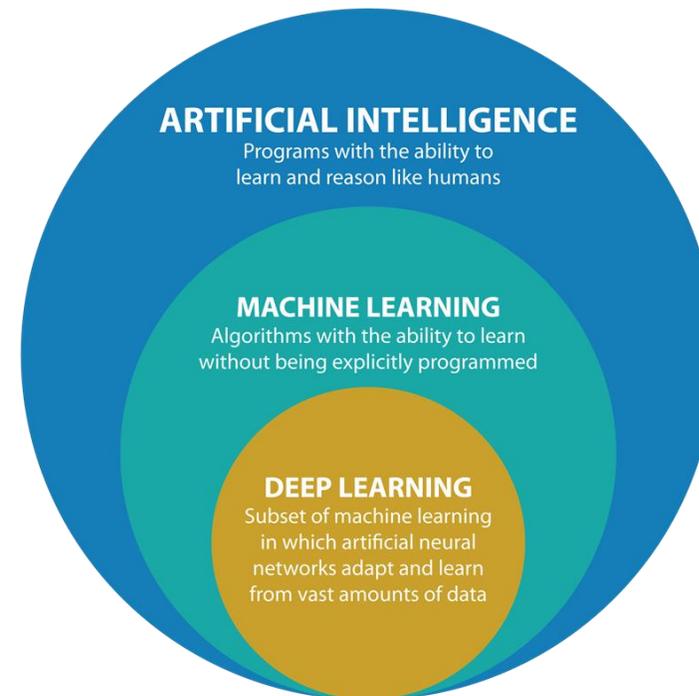


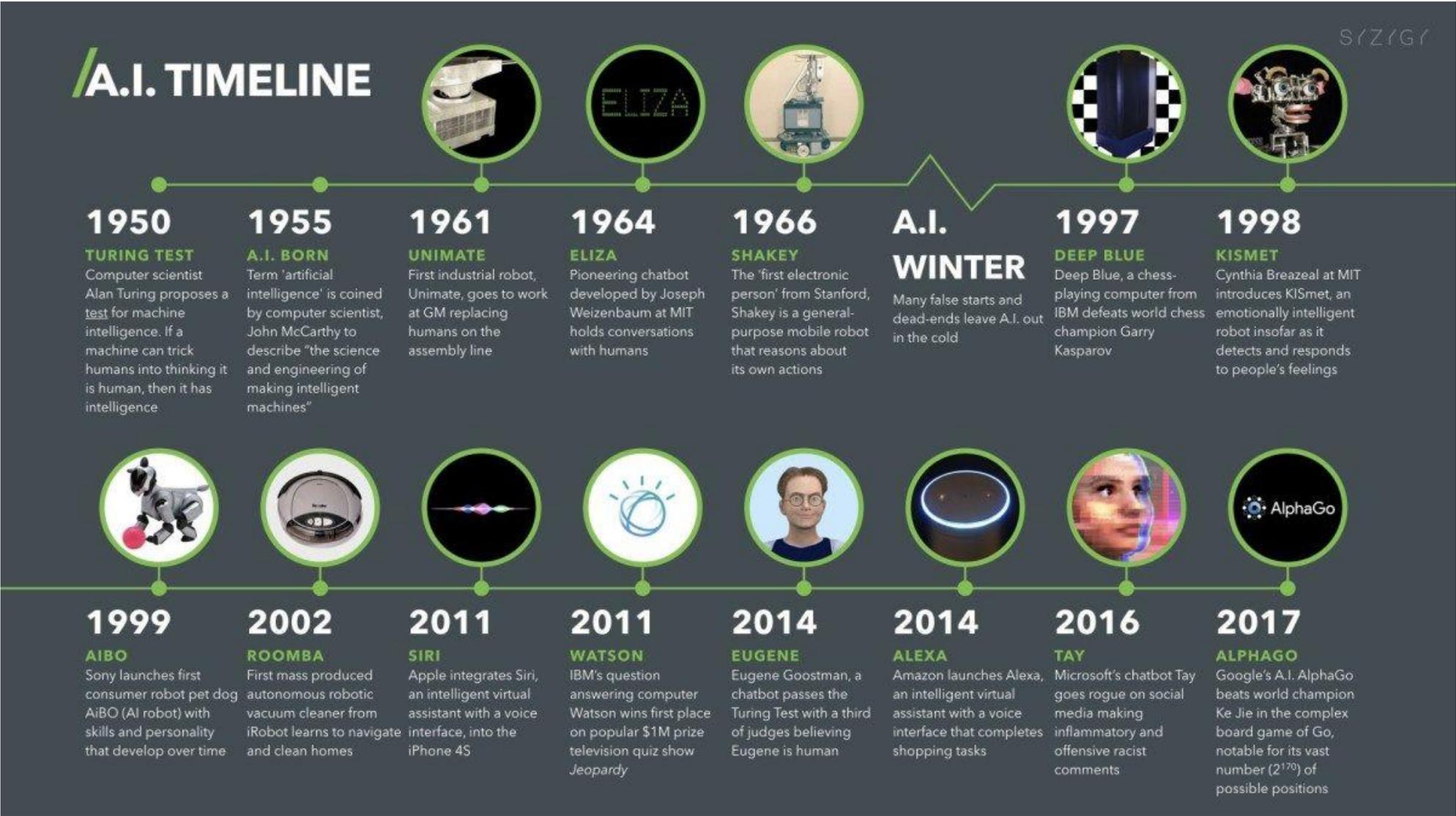
Medtronic





“Capacidad de las máquinas y los sistemas de software para realizar tareas que, en teoría, requerirían inteligencia humana para ser realizadas”





Evolución

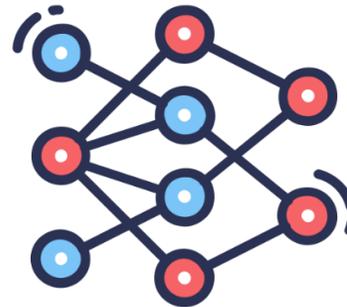
2012



IMAGENET

- Reconocimiento de objetos
- Desbloqueo móvil
- Sistemas de recomendación

2015



EVOLUCION MODELOS

- Autocorrector móvil
- Asistentes virtuales
- Detección de spam

2016



MODELOS GENERATIVOS

- Filtros en redes sociales
- Aplicaciones de diseño

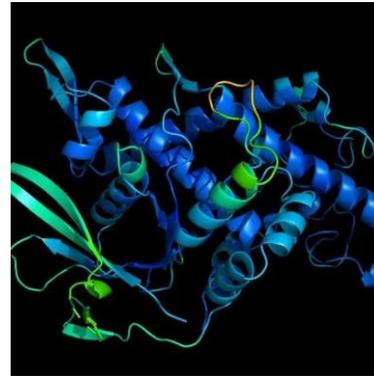
Evolución

2017



Transformers

Alphafold



Plegado de proteínas

Github Copilot



Predicción de código

ChatGPT



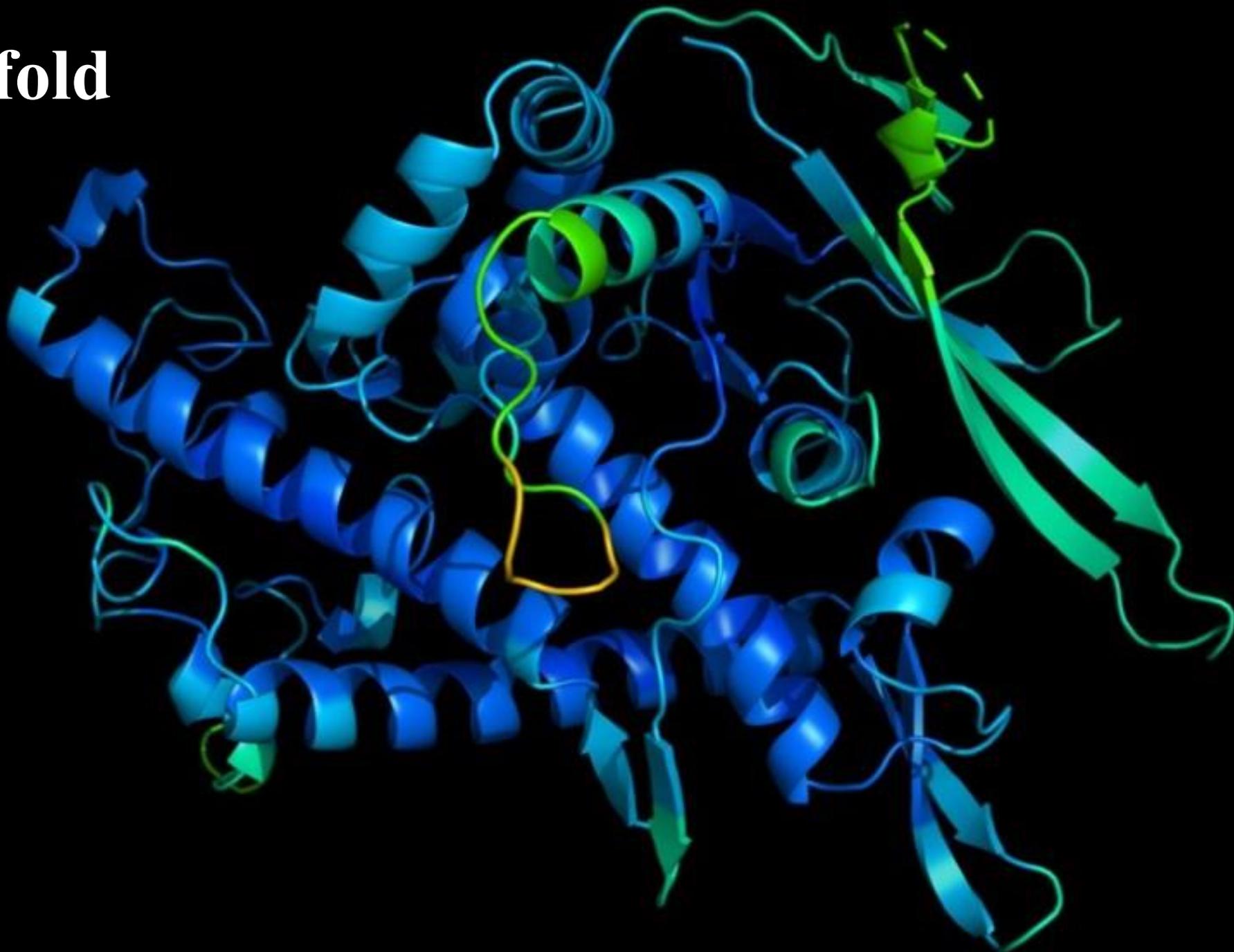
Predicción de texto

Dalle2, Midjourney...



Generación de imágenes

Alphafold

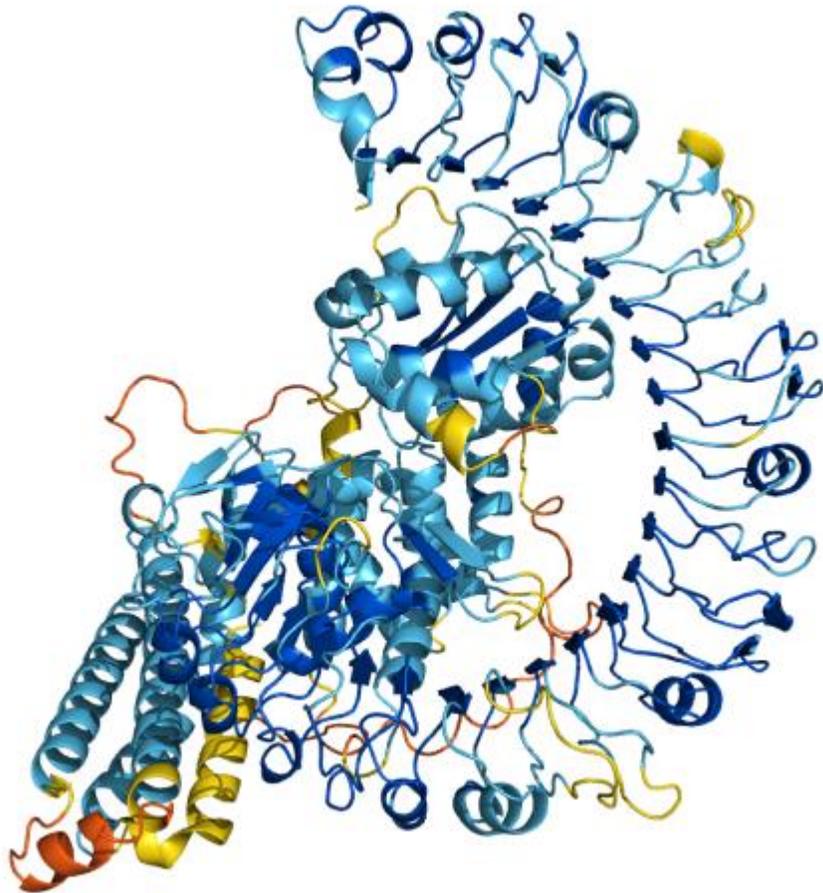


DeepMind



PLEGADO DE PROTEÍNAS

Proteínas - Secuencias de aminoácidos



PLEGADO DE PROTEÍNAS

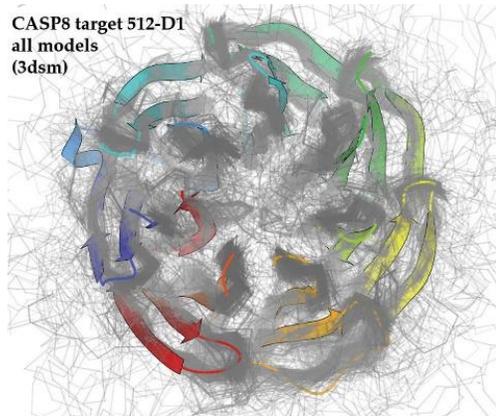
Proteínas - Secuencias de aminoácidos



Mal plegado



Enfermedades: Alzheimer, Parkinson...

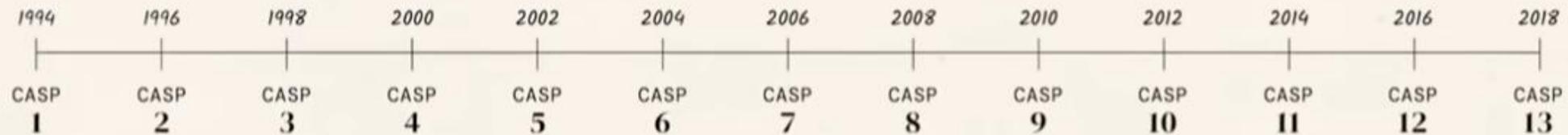


COMPETICIÓN CASP

Métodos tradicionales -
Computacionalmente muy costosos



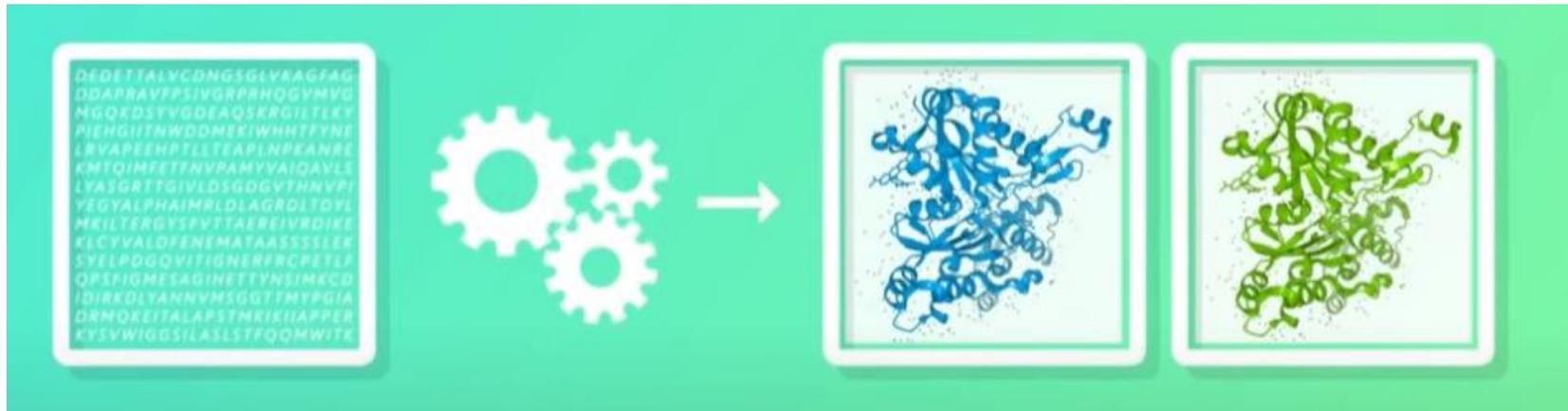
Critical Assessment of Structure Prediction



COMPETICIÓN CASP

SECUENCIA DE AMINOÁCIDOS

ESTRUCTURA 3D PROTEINA

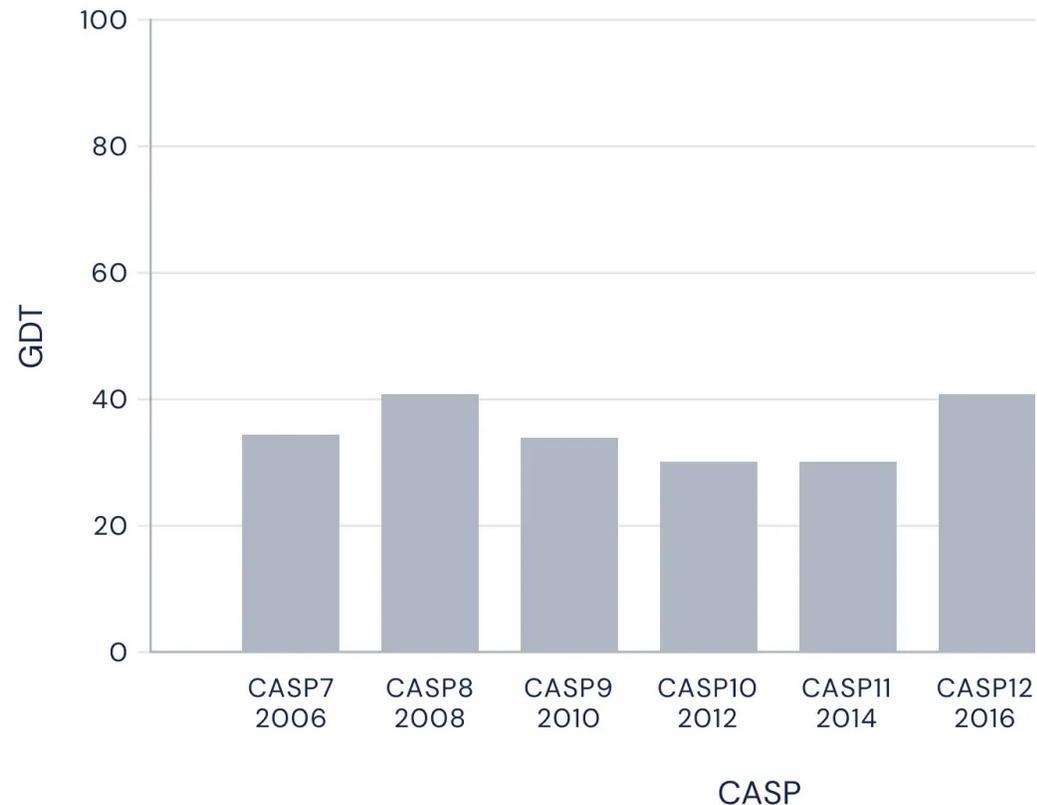


Métrica - Global Distance Test (GDT)

Indica el porcentaje de aminoácidos que se localizan en la posición correcta



Median Free-Modelling Accuracy

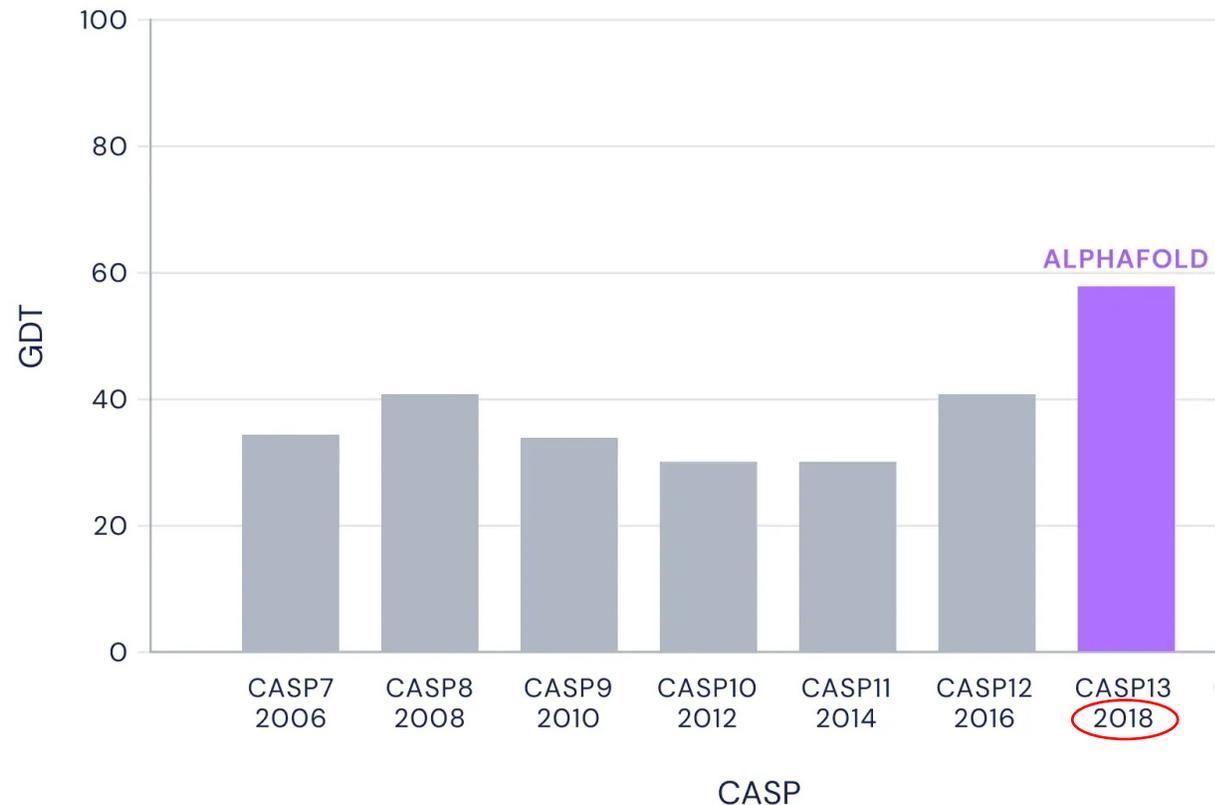


Métrica - Global Distance Test (GDT)

Indica el porcentaje de aminoácidos que se localizan en la posición correcta



Median Free-Modelling Accuracy



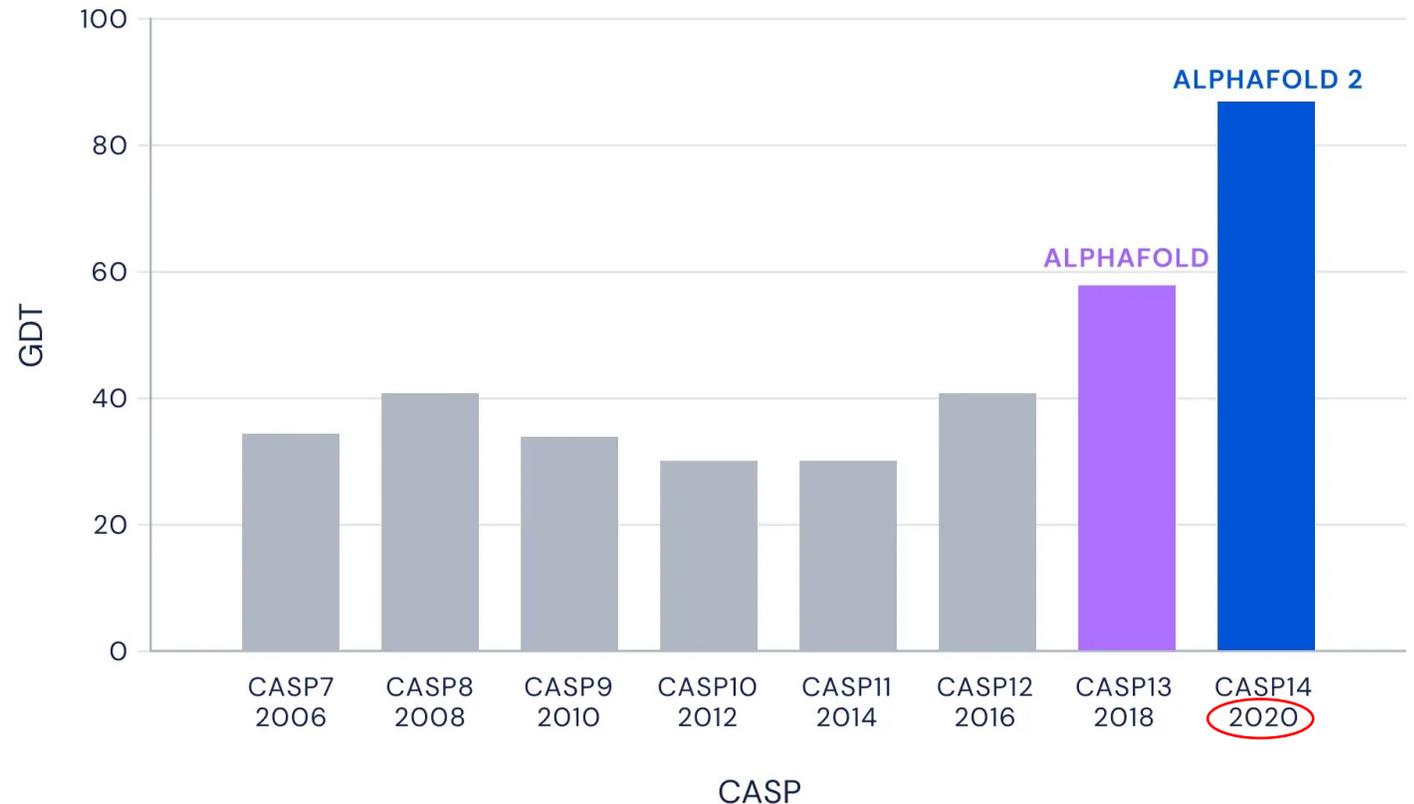
Métrica - Global Distance Test (GDT)

Indica el porcentaje de aminoácidos que se localizan en la posición correcta



¡Problema resuelto!

Median Free-Modelling Accuracy



ALPHAFOLD 1

Convertir una secuencia unidimensional a una estructura tridimensional

ALPHAFOLD 1

Convertir una secuencia unidimensional a una estructura tridimensional

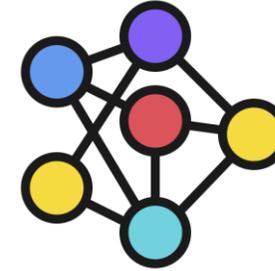


Convertir una imagen a otra imagen

AlphaFold

Tomar una imagen y modificar su estilo

Content image



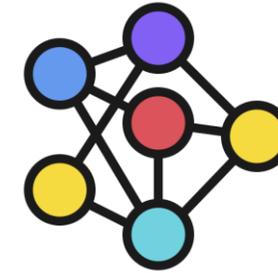
Output image



AlphaFold

Tomar una imagen y modificar su estilo

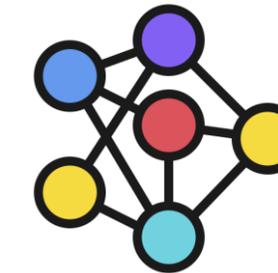
Content image



Output image



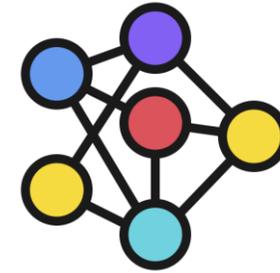
Tomar una imagen y generar un mapa de segmentación



AlphaFold

Tomar una imagen y modificar su estilo

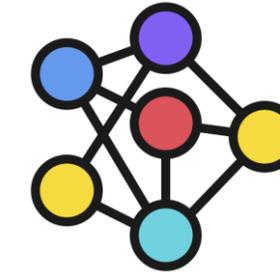
Content image



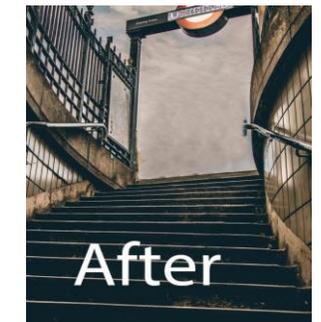
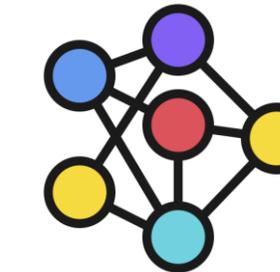
Output image



Tomar una imagen y generar un mapa de segmentación

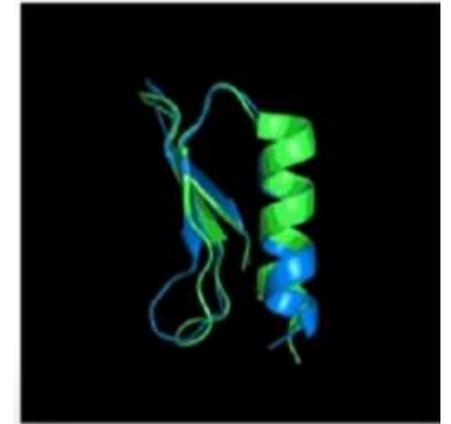
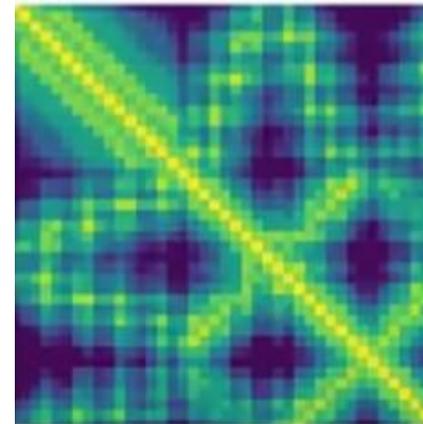
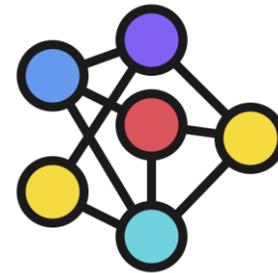
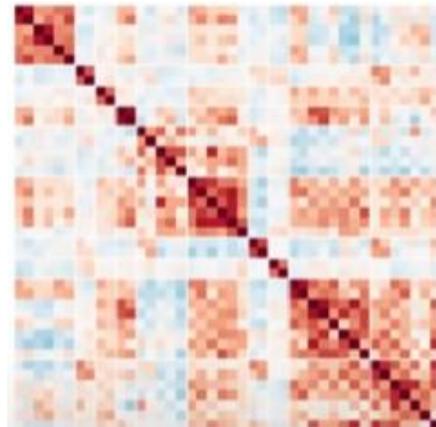


Tomar una imagen y borrar partes

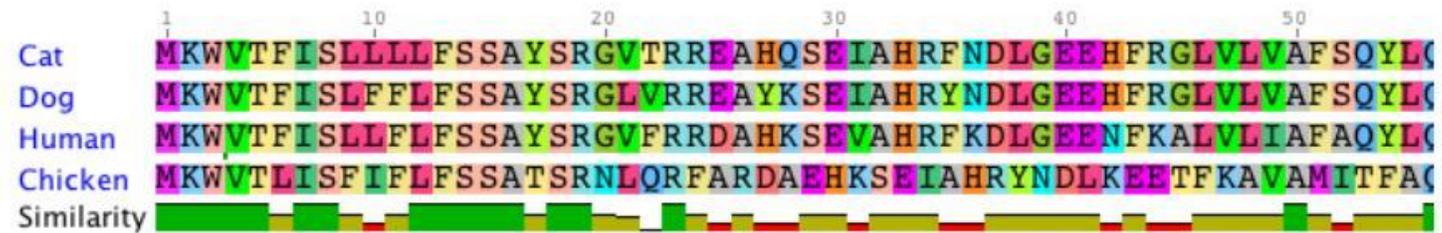


ALPHAFOLD 1

SQETRKKCTEMKKKFKN
CEVRCDESNHCVEVRCS
DTKYTLC



Multiple sequence alignment



Multiple Sequence Alignment (MSA)



Aminoácidos que coevolucionan

Qué interacción se produce dentro de la estructura tridimensional

Predecir plegado

Multiple Sequence Alignment (MSA)

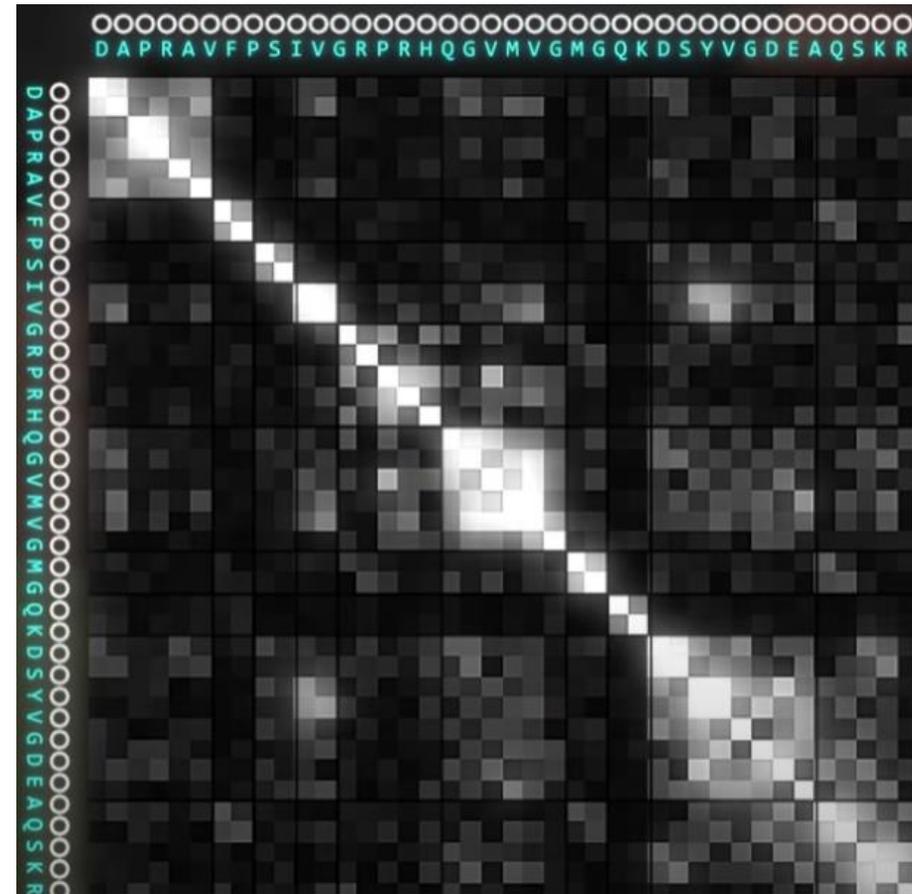
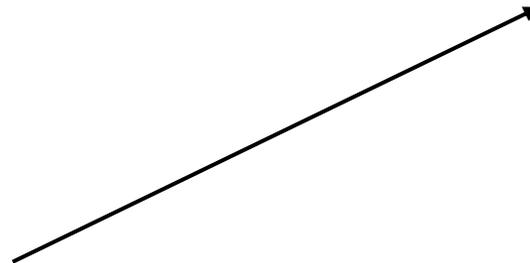


Aminoácidos que coevolucionan

Qué interacción se produce dentro de la estructura tridimensional

Predecir plegado

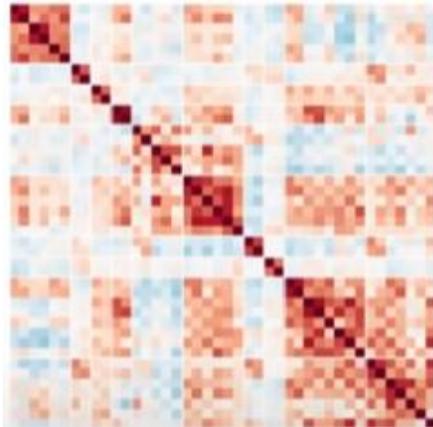
Interacción de aminoácidos



AlphaFold

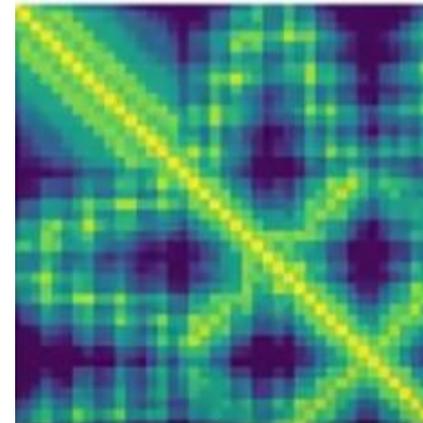
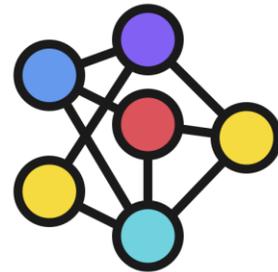
Nahuel Costa
Inteligencia Artificial y Biomedicina

```
SQETRRKKCTEMKKKFKN  
CEVRCDESNHCVEVRC  
DTKYTLC
```



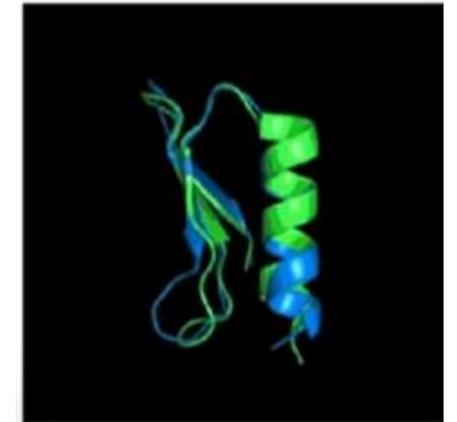
INPUT

Información de la secuencia
de aminoácidos



OUTPUT

Matriz de distancia de la
proteína



OUTPUT FINAL
Estructura 3D de
la proteína

ALPHAFOLD 2

Convertir una secuencia unidimensional a una estructura tridimensional



Modelarlo como un problema de texto



Transformers

ALPHAFOLD 2

Convertir una secuencia unidimensional a una estructura tridimensional



Modelarlo como un problema de texto



Transformers



Permiten atender a las partes más importantes de una secuencia

ALPHAFOLD 2

Convertir una secuencia unidimensional a una estructura tridimensional



Modelarlo como un problema de texto



Transformers

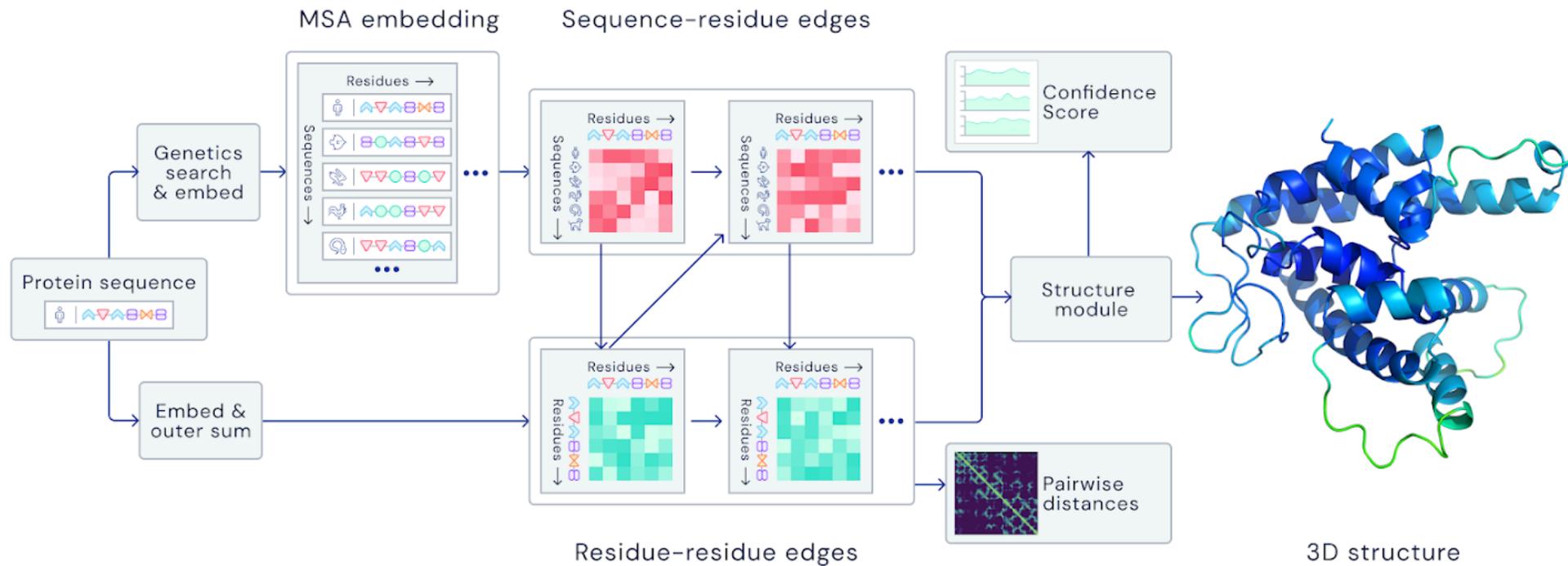


Permiten atender a las partes más importantes de una secuencia



Pueden ser entrenados en paralelo

AlphaFold



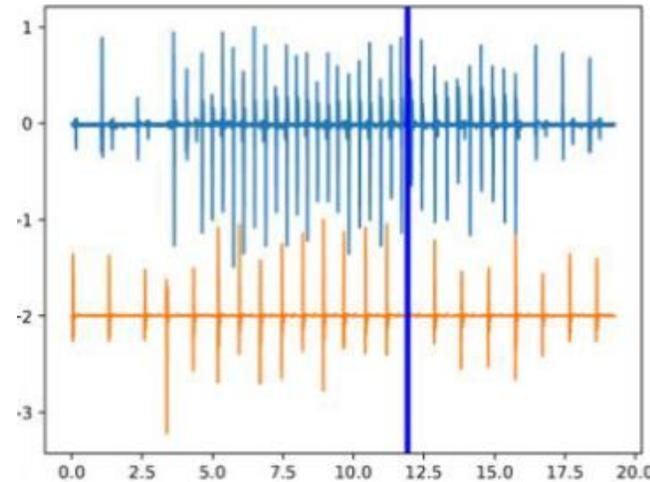
Encuentra secuencias similares a la entrada y pasa esa información a otra red neuronal que produce la estructura 3D final



Fibrilación Auricular

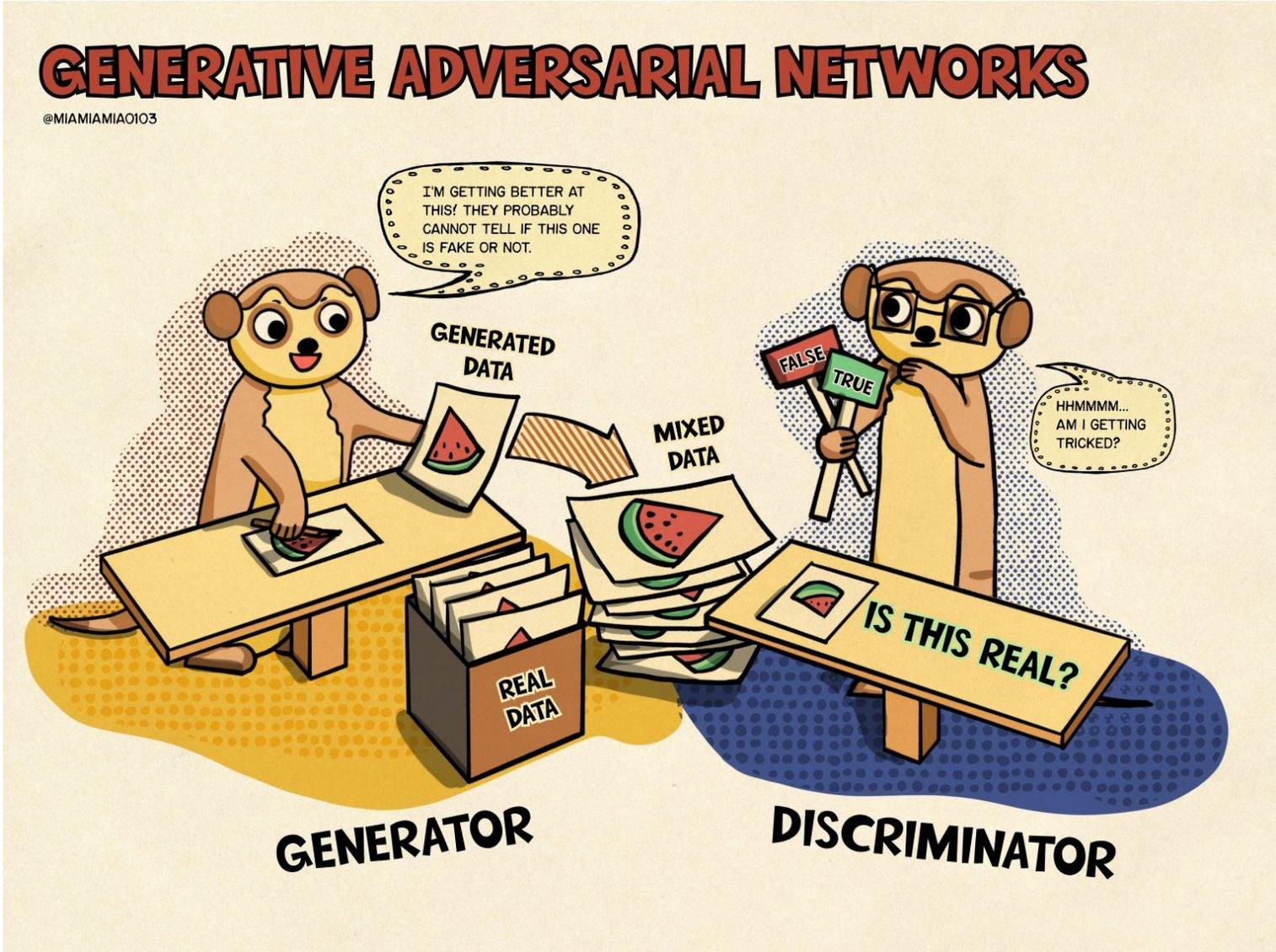
Fibrilación Auricular

Nahuel Costa
Inteligencia Artificial y Biomedicina

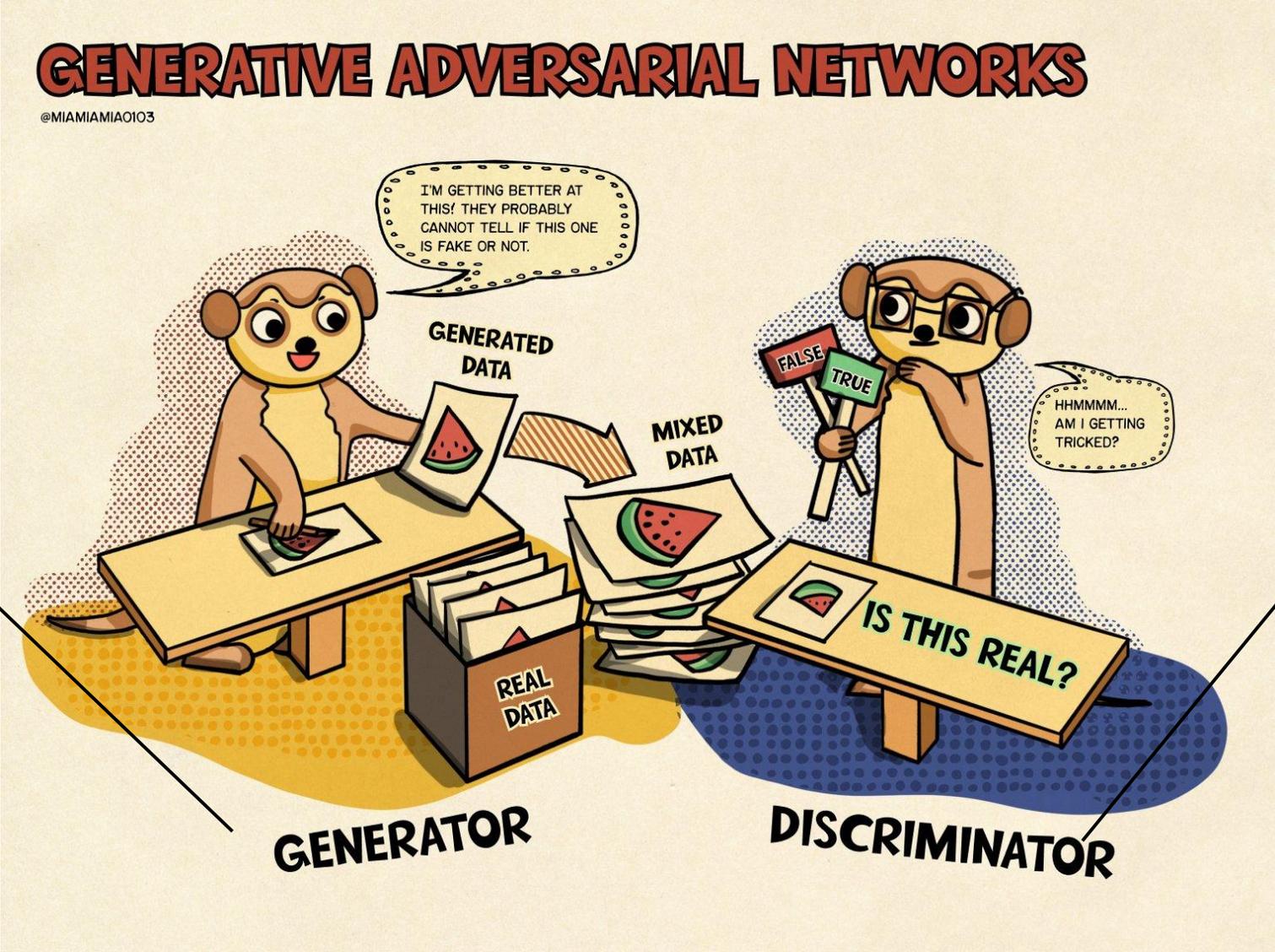


DATOS INCOMPLETOS
REGISTROS LIMITADOS

GANs

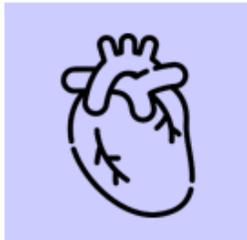


GANs

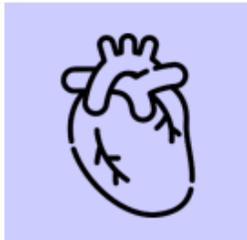


Generar datos
[thisdoesnotexist](#)

Reconocer datos



Atrial Fibrillation



Atrial Fibrillation

Class 1

Class 2

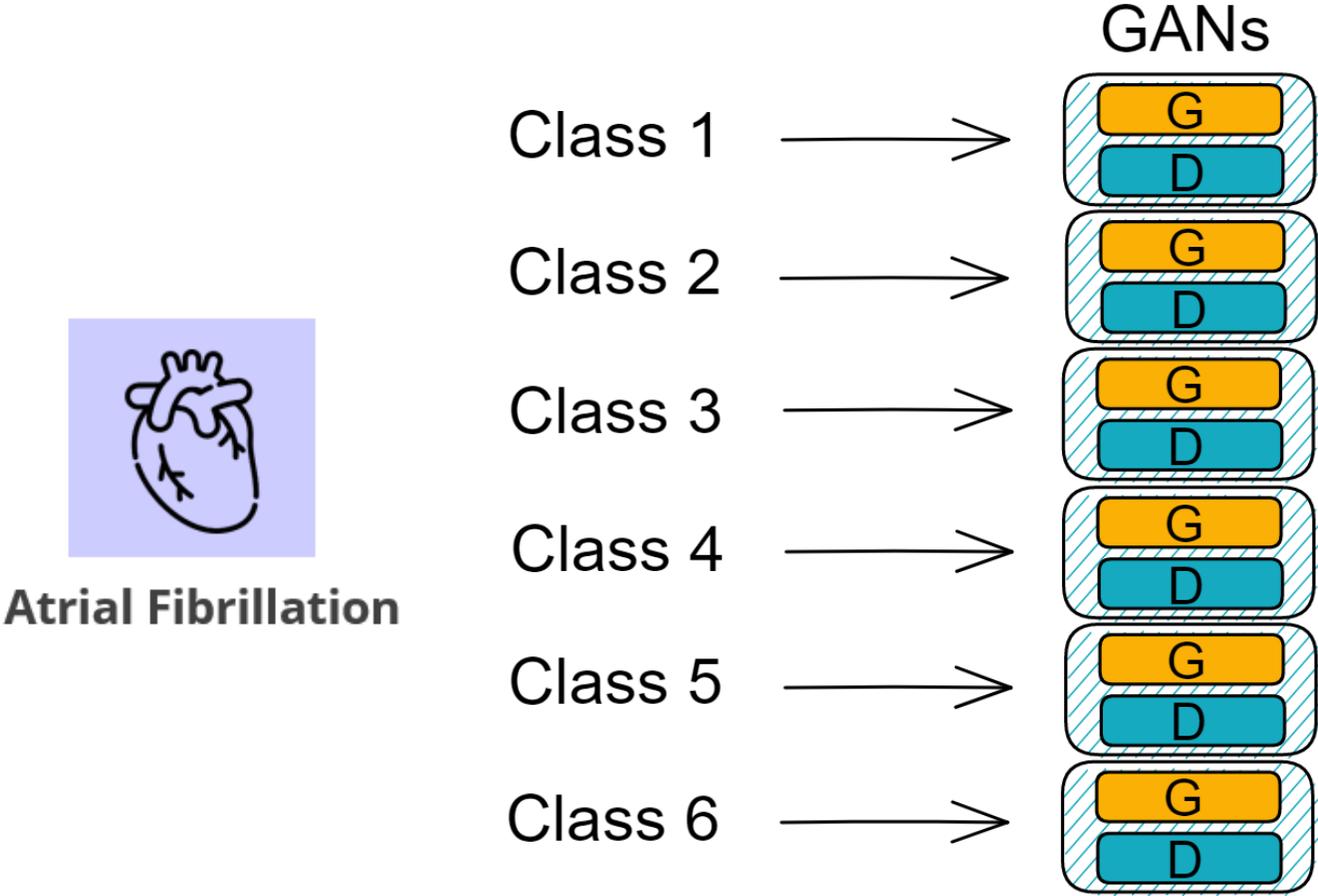
Class 3

Class 4

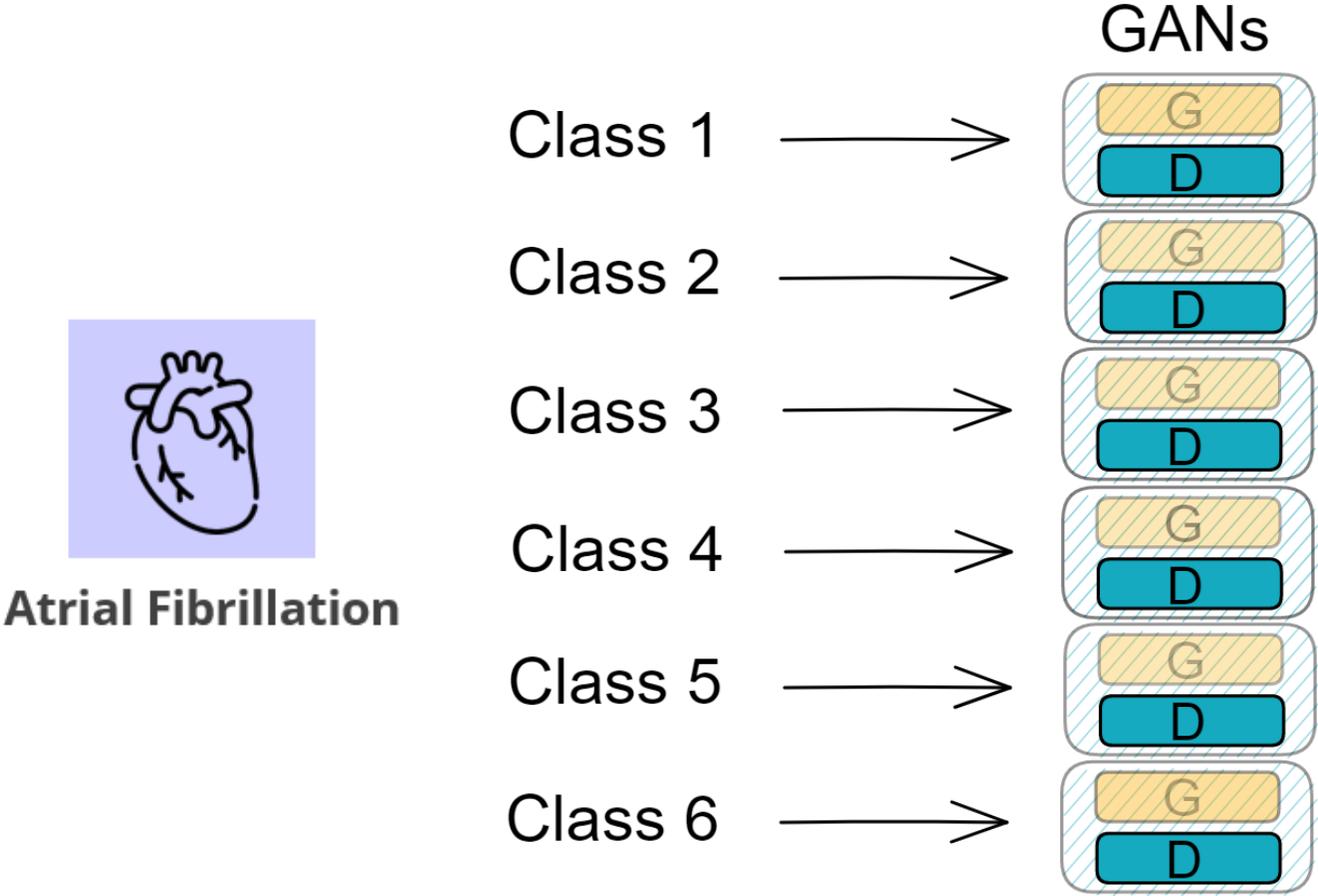
Class 5

Class 6

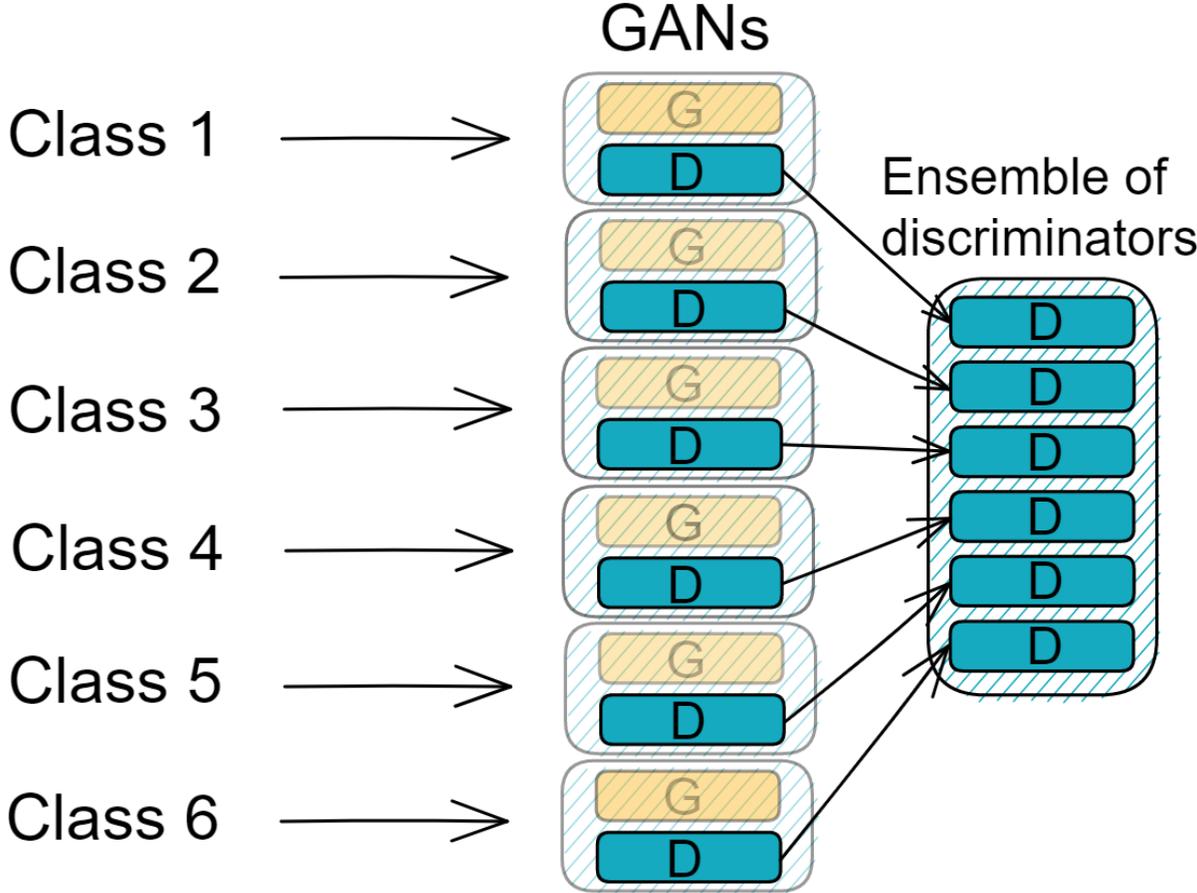
Fibrilación Auricular

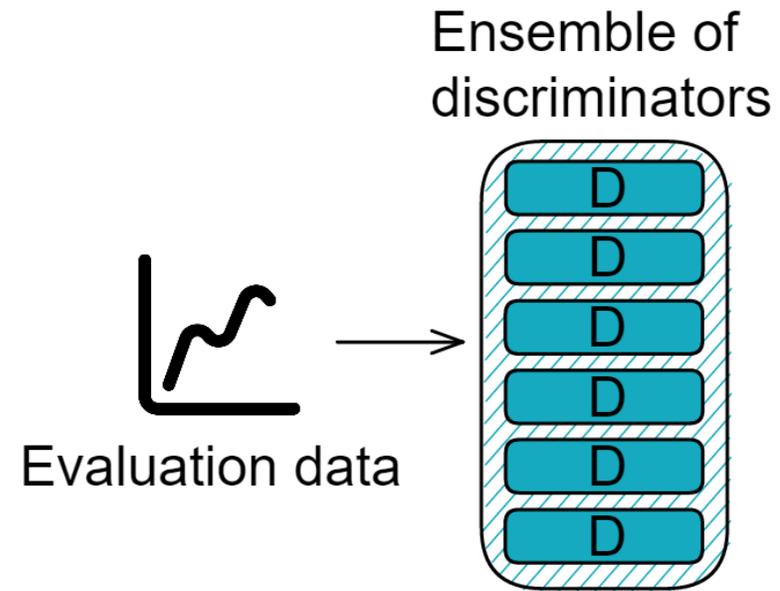


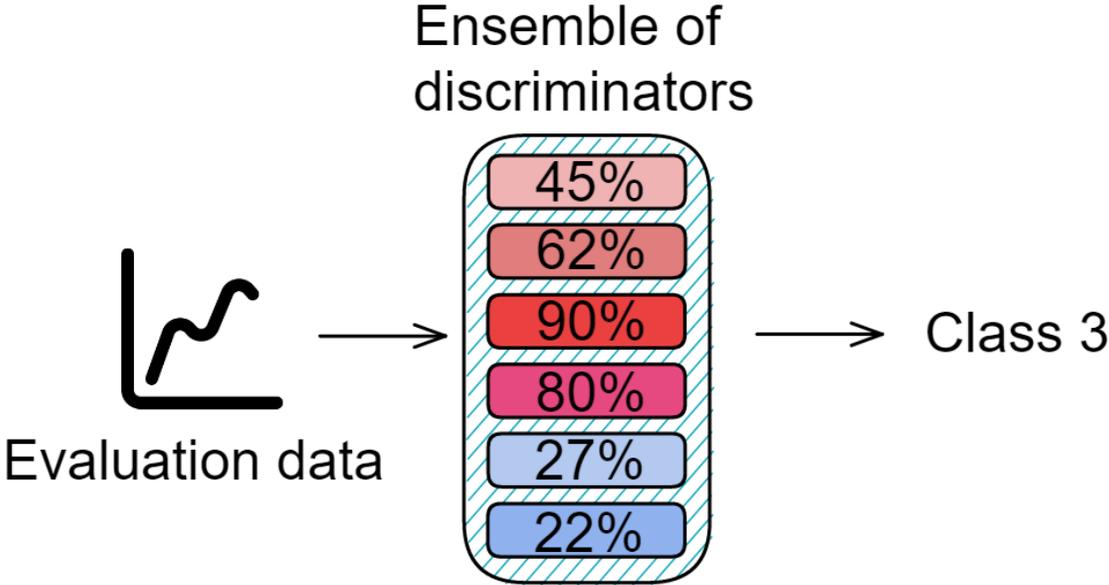
Fibrilación Auricular

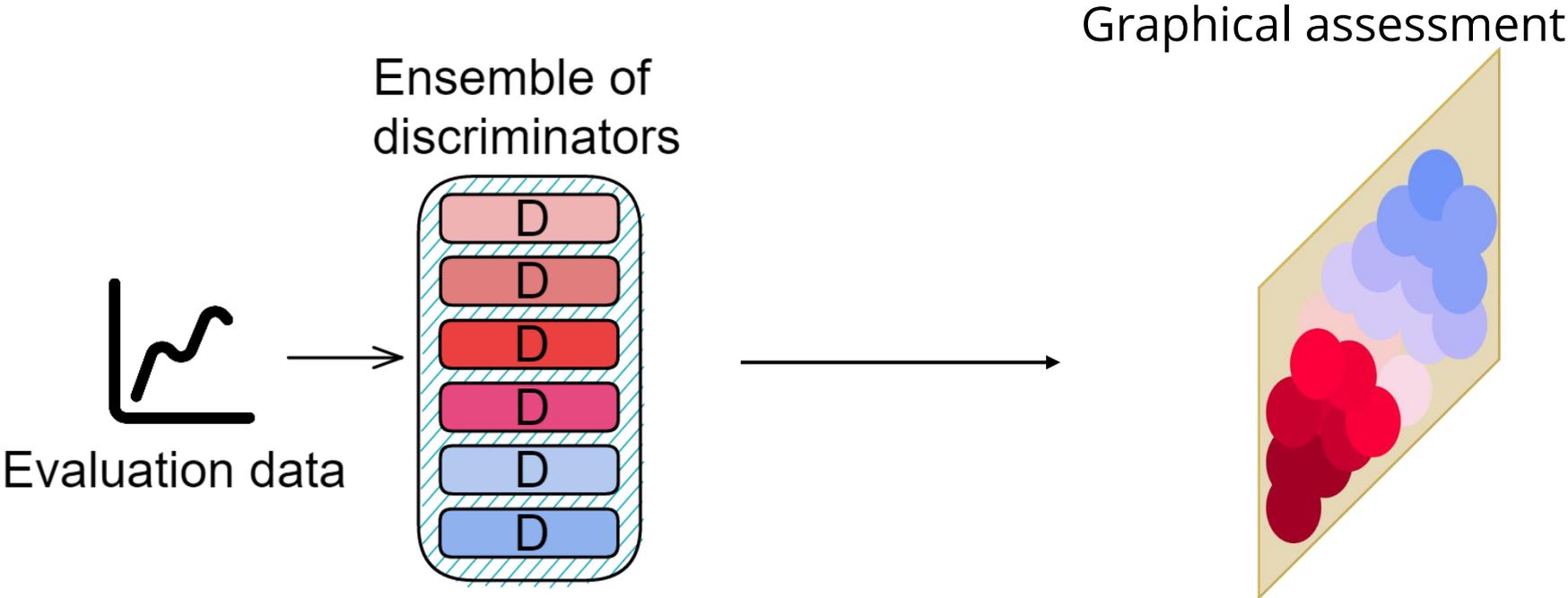


Fibrilación Auricular











Atrial Fibrillation

Class 1

Class 2

Class 3

Class 4

Class 5

Class 6

α = velocidad de progresión hacia arritmia permanente

β = tiempo medio entre episodios de arritmia

Fibrilación Auricular



Atrial Fibrillation

Class 1

Class 2

Class 3

Class 4

Class 5

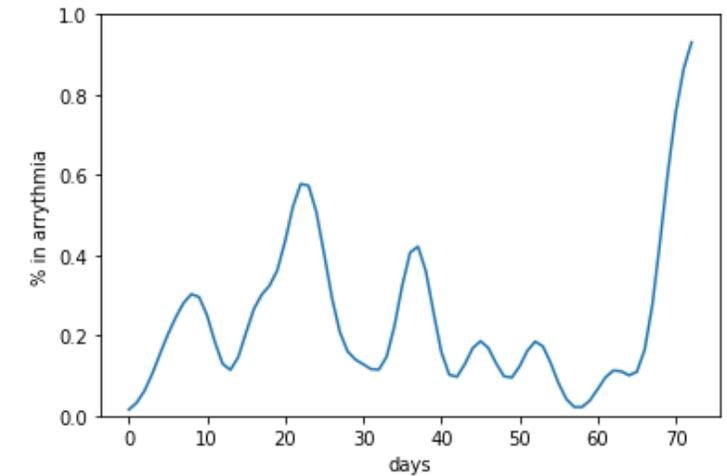
Class 6

Muchos episodios,
progresión rápida
hacia permanente

α = velocidad de progresión hacia arritmia permanente

β = tiempo medio entre episodios de arritmia

α alta, β baja



Fibrilación Auricular



Atrial Fibrillation

Class 1

Class 2

Class 3

Class 4

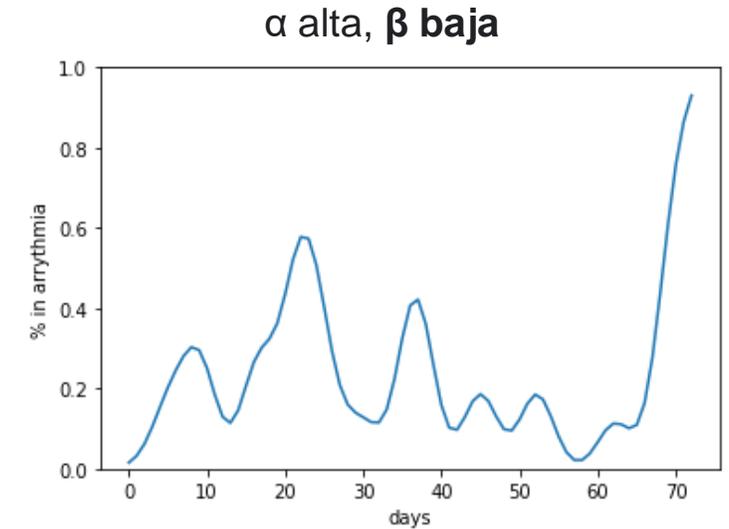
Class 5

Class 6

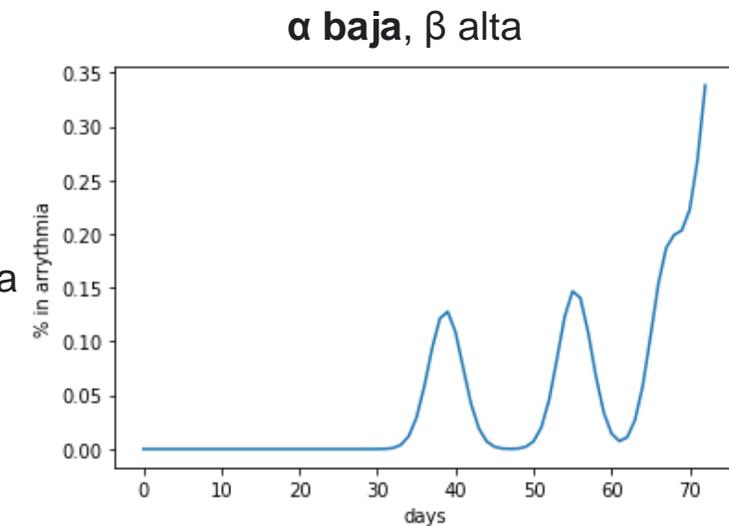
α = velocidad de progresión hacia arritmia permanente

β = tiempo medio entre episodios de arritmia

Muchos episodios,
progresión rápida
hacia permanente



Pocos episodios,
pero también
progresión rápida hacia
permanente



Fibrilación Auricular



Atrial Fibrillation

- Class 1
- Class 2
- Class 3
- Class 4
- Class 5
- Class 6

Muchos episodios,
progresión rápida
hacia permanente

Criticidad es la misma
Independientemente del
número de episodios

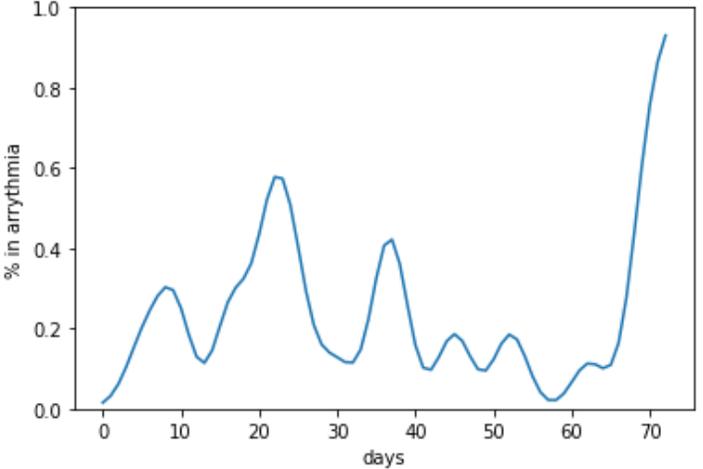


Pocos episodios,
pero también
progresión rápida hacia
permanente

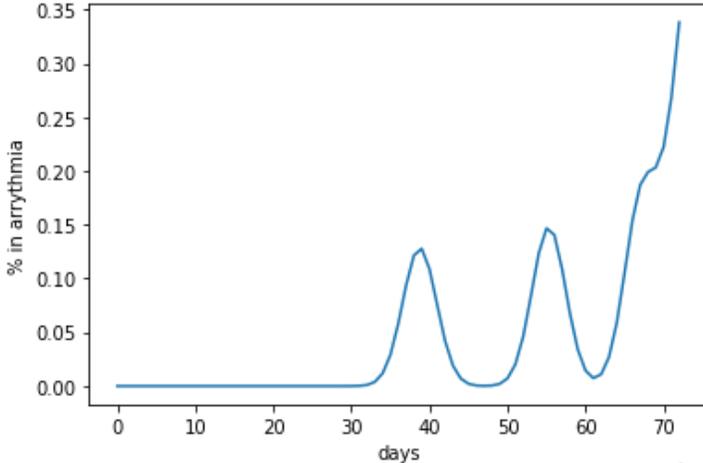
α = velocidad de progresión hacia arritmia permanente

β = tiempo medio entre episodios de arritmia

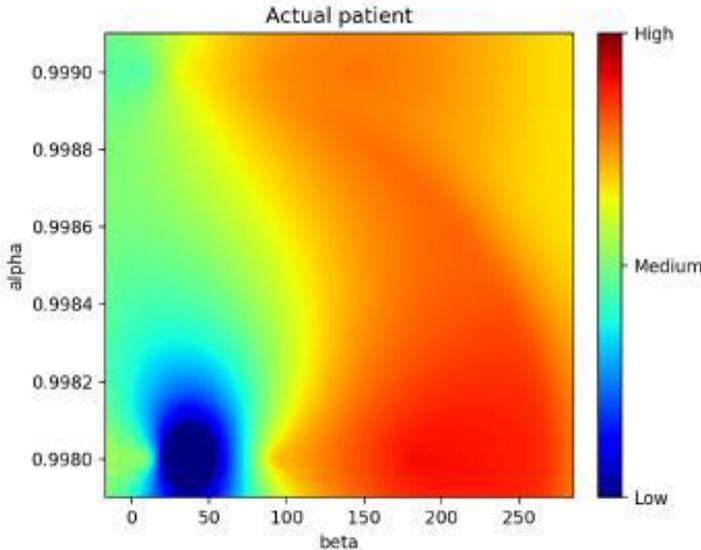
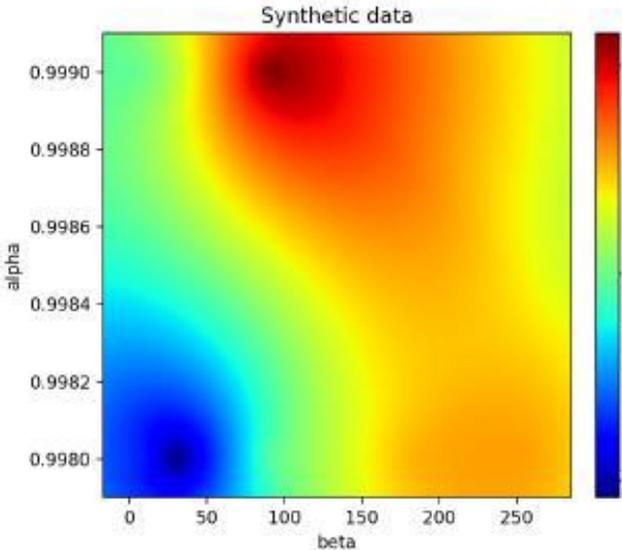
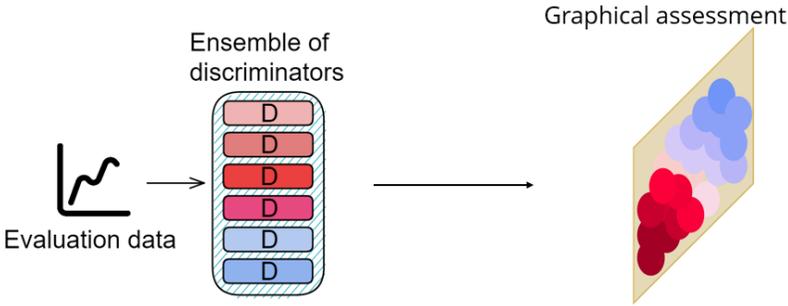
α alta, β baja



α baja, β alta



Mapa gráfico de las activaciones de las últimas capas de cada discriminador

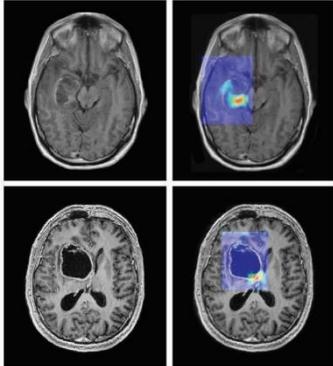




Otras aplicaciones

Otras aplicaciones

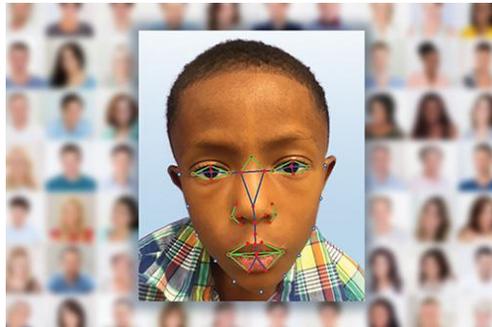
Nahuel Costa
Inteligencia Artificial y Biomedicina



Análisis de imágenes médicas



Tratamientos farmacológicos



Genética



Obstetricia y ginecología



Prótesis



Por dónde empiezo?

Recursos

Nahuel Costa
Inteligencia Artificial y Biomedicina



Andrés Torrubia



Aurelia Bustos

SOFTWARE 2.0

Recursos

Nahuel Costa
Inteligencia Artificial y Biomedicina



Andrés Torrubia



Aurelia Bustos

SOFTWARE 2.0



Jeremy Howard



Rachel Thomas

FASTAI



[DotCSV](#)

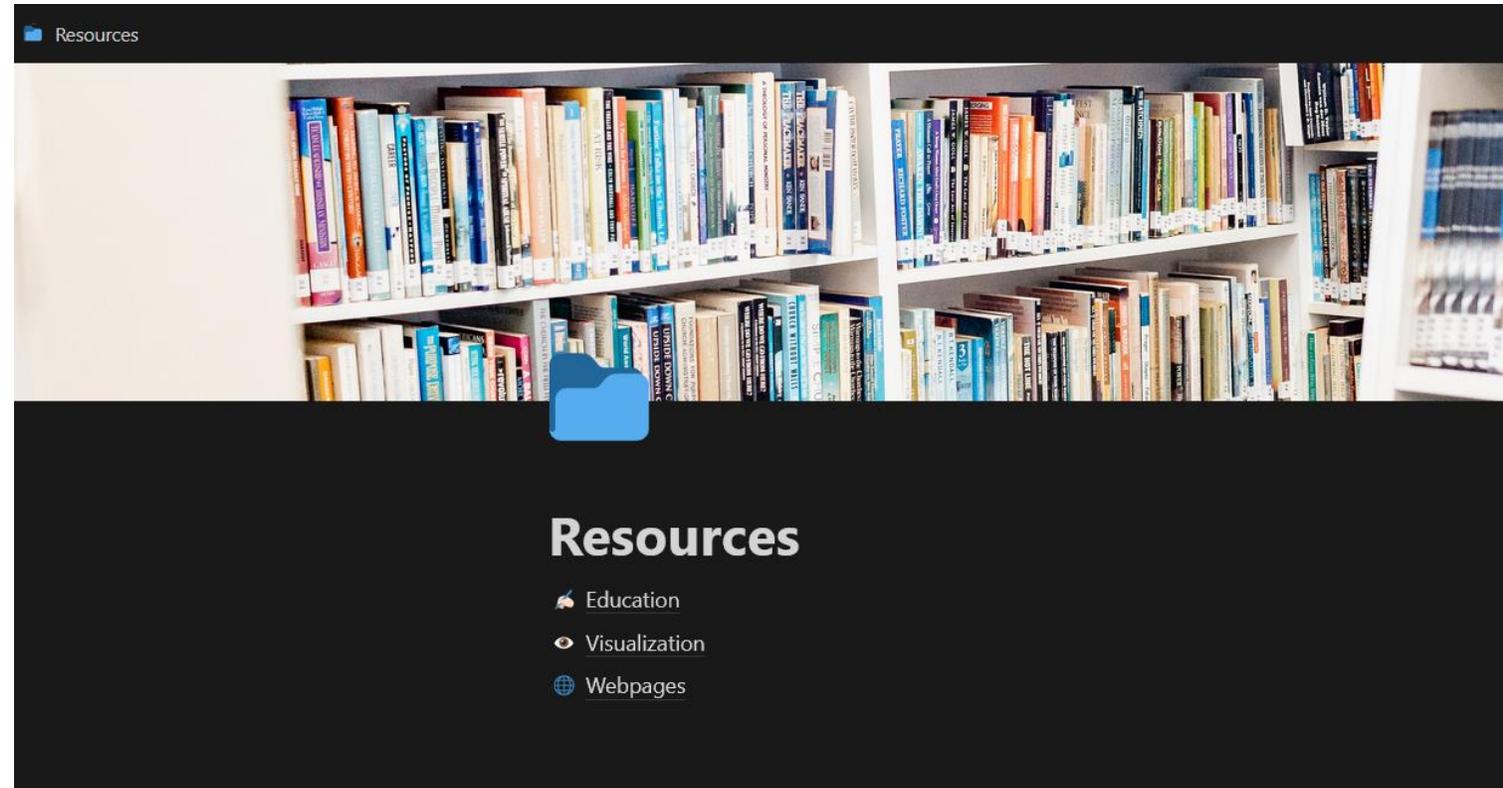
Divulgador de Inteligencia Artificial



[La hiperactina](#)

Divulgadora biomedicina

[Lista de recursos](#)



Perspectivas futuras



```
import random
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Input, Dense, LSTM, Bidirectional, Masking

# keeping the random seed constant from one experiment to the next makes it
# easier to interpret the effects of hyper-parameters values
seed = 99
random.seed(seed)
tf.random.set_seed(seed)

def create_model(timesteps, input_dim, intermediate_dim, batch_size, latent_dim, epochs,
optimizer): # the network parameters:
    timesteps = timesteps
    input_dim = input_dim
    intermediate_dim = intermediate_dim
    batch_size = batch_size
    latent_dim = latent_dim
    epochs = epochs
    if optimizer == 'adam':
        optimizer = keras.optimizers.Adam(learning_rate=0.001)
    else:
        print("unimplemented optimizer")
        exit(-1)
    masking_value = -99.

    class Sampling(keras.layers.Layer):
        """Uses (z_mean, sigma) to sample z, the vector encoding an engine trajectory."""
        def call(self, inputs):
            mu, sigma = inputs
            batch = tf.shape(mu)[0]
            dim = tf.shape(mu)[1]
            epsilon = tf.keras.backend.random_normal(shape=(batch, dim))
            return mu + tf.exp(0.5 * sigma) * epsilon

    # ----- Encoder -----
    inputs = Input(shape=(timesteps, input_dim,), name='encoder_input')
    mask = Masking(mask_value=masking_value)(inputs)

    # LSTM encoding
    h = Bidirectional(LSTM(intermediate_dim))(mask)

    # VAE Z layer
    mu = Dense(latent_dim)(h)
    sigma = Dense(latent_dim)(h)
    z = Sampling()(mu, sigma)

    # Instantiate the encoder model:
    encoder = keras.Model(inputs, [z, mu, sigma], name='encoder')
    print(encoder.summary())
    # ----- Regressor -----
    reg_latent_inputs = Input(shape=(latent_dim,), name='z_sampling_reg')
    reg_intermediate = Dense(200, activation='tanh')(reg_latent_inputs)
    reg_outputs = Dense(1, name='reg_output')(reg_intermediate)
    # Instantiate the classifier model:
    regressor = keras.Model(reg_latent_inputs, reg_outputs, name='regressor')
    print(regressor.summary())
    # -----
    return go(f, seed, [])
}
```

```
import random
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Input, Dense, LSTM, Bidirectional, Masking

# keeping the random seed constant from one experiment to the next makes it
# easier to interpret the effects of hyper-parameters values
seed = 99
random.seed(seed)
tf.random.set_seed(seed)

def create_model(timesteps, input_dim, intermediate_dim, batch_size, latent_dim, epochs,
optimizer):
    """the network parameters:
    timesteps = timesteps
    input_dim = input_dim
    intermediate_dim = intermediate_dim
    batch_size = batch_size
    latent_dim = latent_dim
    epochs = epochs
    if optimizer == 'adam':
        optimizer = keras.optimizers.Adam(learning_rate=0.001)
    else:
        print("unimplemented optimizer")
        exit(-1)
    masking_value = -99.

    class Sampling(keras.layers.Layer):
        """Uses (z_mean, sigma) to sample z, the vector encoding an engine trajetory."""
        def call(self, inputs):
            mu, sigma = inputs
            batch = tf.shape(mu)[0]
            dim = tf.shape(mu)[1]
            epsilon = tf.keras.backend.random_normal(shape=(batch, dim))
            return mu + tf.exp(0.5 * sigma) * epsilon

    # ----- Encoder -----
    inputs = Input(shape=(timesteps, input_dim,), name='encoder_input')
    mask = Masking(mask_value=masking_value)(inputs)

    # LSTM encoding
    h = Bidirectional(LSTM(intermediate_dim))(mask)

    # VAE Z layer
    mu = Dense(latent_dim)(h)
    sigma = Dense(latent_dim)(h)
    z = Sampling()(mu, sigma)

    # Instantiate the encoder model:
    encoder = keras.Model(inputs, [z, mu, sigma], name='encoder')
    print(encoder.summary())
    # -----
    # ----- Regressor -----
    reg_latent_inputs = Input(shape=(latent_dim,), name='z_sampling_reg')
    reg_intermediate = Dense(200, activation='tanh')(reg_latent_inputs)
    reg_outputs = Dense(1, name='reg_output')(reg_intermediate)
    # Instantiate the classifier model:
    regressor = keras.Model(reg_latent_inputs, reg_outputs, name='regressor')
    print(regressor.summary())
    # -----
    return go(f, seed, [])
}
```



“Procesa mis datos y aplica una red convolucional para aprender patrones que ayuden a identificar X patología”

- Creatividad para combinar modelos
- Explicabilidad
- Ética y privacidad

