Proyecto MIABS

Guía completa para una primera red neuronal simulada biológica

✅ Checklist final

📚 Roadmap de aprendizaje de conceptos

Objetivo

> Tiener una pequeña “micro-red cerebral” simulada.
>
> Le mandas dos tipos de estímulos: **Patrón A** y **Patrón B** (secuencias de spikes).
>
> Al inicio, la red responde de forma caótica, sin preferencia.
>
> Después de entrenar con una regla biológica (**STDP**), la red **aprende por sí sola** a activar más una neurona de salida cuando ve el Patrón A que cuando ve el B.
>
> Tú podrás:
>
> - Ver los **picos de actividad (spikes)** como en neurociencia real.
> - Ver cómo los **pesos sinápticos cambian en el tiempo**.
> - Publicar tu modelo como proyecto open source, con gráficos y explicación.

Eso es, en esencia, lo que vas a hacer: **una mini-red neuronal de disparo (Spiking Neural Network)** con comportamiento biológicamente inspirado y un pipeline limpio para compartir.

Ejemplo visual final (Evento)

Para demostrar visual y comprensivamente la red neuronal biológica simulada se puede aplicar un decodificador de eventos final, un pequeño script que

diga: *"Si la tasa de disparo de la neurona de salida supera X umbral en los últimos 50ms, ejecuta la función* `abrir_puerta()` *"*.

Dos ejemplos muy llamativos a mostrar (con foco especial en el segundo) luego de la presentación técnica son:

1. **El Experimento de Pavlov Virtual (Condicionamiento Clásico)**

Dado que usas **STDP** (Plasticidad Dependiente del Tiempo de Disparo), estás simulando aprendizaje biológico real (regla de Hebb: "neuronas que disparan juntas, permanecen juntas").

- **La Analogía:** El perro de Pavlov.
- **La Visualización:**
  - Dibuja un pequeño "ente" o robot virtual en pantalla.
  - **Estímulo A (Olor a comida):** Genera spikes en la entrada.
  - **Estímulo B (Campana):** Genera otros spikes.
  - **Salida:** Salivar (o encender una luz verde).
  - **El Demo:** Al principio, la campana no hace nada. Muestras en tiempo real cómo, tras presentar "Campana + Comida" varias veces (entrenamiento STDP), el robot empieza a reaccionar *solo* con la campana.
- **Por qué funciona:** Es el ejemplo de neurociencia más famoso del mundo. Cualquier jurado entenderá que tu red "aprendió por asociación" sin que le explicaras matemáticamente los pesos.

2. **"Codificación Neuromórfica de Estímulos Auditivos: Análisis Comparativo en Tiempo Real entre Mecanorreceptores Biológicos y Redes Neuronales de Pulsos (SNN)"**

Este proyecto propone una validación experimental de una Red Neuronal de Pulsos (SNN) mediante una comparativa biológica directa. En lugar de limitarse a simulaciones teóricas, se desarrollará un entorno de pruebas híbrido donde un mismo estímulo auditivo (patrones rítmicos musicales) será procesado simultáneamente por dos sistemas:

1. **Sistema Biológico (In Vivo/Ex Vivo):** Neuronas sensoriales reales (mecanorreceptores) de la pata de un insecto (ej. *Blaberus discoidalis* o *Acheta domesticus*), captando las vibraciones sonoras.

2. **Sistema Artificial (In Silico):** Una SNN simulada en Python con reglas de aprendizaje STDP (Spike-Timing-Dependent Plasticity).

El objetivo es demostrar visualmente cómo la arquitectura SNN propuesta es capaz de replicar la eficiencia de codificación temporal y filtrado de ruido de un sistema nervioso biológico real.

Metodología Experimental

El experimento se estructura en un pipeline de tres etapas:

- **A. Estímulo Compartido:** Se utilizará una pista de audio con frecuencias graves predominantes y rítmica marcada. Este audio servirá como entrada de datos para la SNN y, simultáneamente, generará vibraciones físicas (ondas de presión) a través de un transductor (altavoz) dirigido al espécimen biológico.
- **B. Adquisición de Señal Biológica:** Mediante un amplificador de biopotenciales (*SpikerBox*), se registrarán los potenciales de acción (spikes) generados por los pelos sensoriales de la pata del insecto al reaccionar a la vibración del aire producida por la música. Esta señal analógica se digitalizará en tiempo real.
- **C. Procesamiento SNN:** El mismo audio será convertido a trenes de pulsos y alimentará la red neuronal artificial, la cual deberá detectar patrones y características temporales mediante aprendizaje no supervisado.

Entregable Visual: El "Ecualizador Neuromórfico"

Como resultado final, se presentará una interfaz gráfica de usuario (GUI) en pantalla dividida que muestre en tiempo real:

1. **Canal Biológico:** Visualización de los spikes crudos del insecto, evidenciando el ruido natural y la respuesta estocástica ante el ritmo.
2. **Canal Artificial:** Visualización de los disparos de la SNN y la evolución de los pesos sinápticos.
3. **Correlación:** Un gráfico dinámico (tipo ecualizador) que se ilumina cuando ambos sistemas coinciden temporalmente, demostrando la precisión de la SNN para imitar la respuesta biológica ante el estímulo.

Justificación de Innovación (Por qué esto es relevante)

Este enfoque supera las tesis tradicionales de simulación al integrar **Bio-Ingeniería** y **Hardware**. Demuestra no solo la capacidad del estudiante para

diseñar algoritmos complejos (SNN/STDP), sino también para manejar interfaces de adquisición de datos reales y enfrentar el desafío de la "computación ruidosa" propia de los sistemas biológicos, validando la robustez del modelo propuesto en un entorno físico.

Experimento al completo: https://gemini.google.com/share/d0ead2645da7

---

1. Ruta mental del proyecto (visión general)

Objetivo global:

Diseñar, implementar y documentar una **Spiking Neural Network biológicamente inspirada** que aprenda una tarea sencilla mediante **plasticidad sináptica (STDP)**, y publicar el código + documentación como proyecto open source.

Bloques del trabajo:

1. Entender los conceptos clave (neurona biológica, spikes, SNN, STDP).
2. Diseñar en papel tu modelo: arquitectura, reglas, tarea.
3. Implementar el modelo paso a paso en Python con una librería de simulación (p.ej. Brian2).
4. Diseñar experimentos simples y producir gráficos interpretables.
5. Escribir documentación y empaquetar el código para open source.
6. Bonus: explorar cómo usar IBM Quantum para optimizar hiperparámetros (a nivel PoC).

---

2. Conceptos que debes comprender antes de tocar el modelo

2.1 Neurociencia básica (nivel mínimo necesario)

- **Neurona biológica**
  - Cuerpo celular (soma), dendritas, axón.
  - Potencial de membrana, potencial de acción (spike).
  - Refractoriedad (periodo tras un spike donde la neurona no dispara).
- **Sinapsis**

    - Conexiones excitatorias vs inhibitorias.
    - Fuerza sináptica (peso).
- **Plasticidad**
    - Hebb: *"neurons that fire together, wire together"*.
    - **STDP (Spike-Timing Dependent Plasticity)**:
        - Si la neurona presináptica dispara justo antes que la postsináptica → peso aumenta.
        - Si dispara justo después → peso disminuye.

> Objetivo: poder explicar con tus propias palabras qué es un spike y qué es STDP.

2.2 Redes de neuronas de disparo (Spiking Neural Networks — SNN)

- Diferencias frente a redes neuronales artificiales clásicas:
    - Trabajan con **eventos en el tiempo (spikes)**, no con valores continuos estáticos.
    - El **tiempo** es parte central del cálculo.
    - Pueden ser mucho más cercanas a la dinámica cerebral real.
- Modelos de neurona que debes conocer (al menos conceptualmente):
    - **Integrate-and-Fire / Leaky Integrate-and-Fire (LIF)**.
    - Izhikevich (saber que existe, no necesitas implementarlo al inicio).
- Codificación de información:
    - **Rate coding**: información en la frecuencia de disparos.
    - **Temporal coding**: información en la sincronía y el timing de los spikes.

2.3 Simulación computacional de redes neuronales biológicas

- Simulación en **tiempo discreto** (pasos de Δt).
- Ecuaciones diferenciales simples para describir el potencial de membrana.
- Concepto de:

- **Poblaciones neuronales** (grupos de neuronas con el mismo tipo de ecuación).
- **Conectividad probabilística** (p.ej. conectar cada neurona con probabilidad p).

> No necesitas ser experto en ecuaciones diferenciales, pero sí entender que el modelo está basado en cambios continuos aproximados en pasos discretos.

2.4 Conceptos mínimos para el bonus cuántico

- Qubit, superposición, medición.
- Circuito cuántico con parámetros (ángulos de rotación, por ejemplo).
- Idea de un **bucle clásico–cuántico**:
  - El circuito propone parámetros.
  - Tú evalúas el rendimiento del modelo clásico (tu SNN).
  - Ajustas los parámetros para mejorar.

No necesitas profundizar en algoritmos cuánticos avanzados; basta con ver la cuántica como **motor de búsqueda de hiperparámetros**.

---

3. Diseño del modelo en papel (antes de programar)

3.1 Definir la tarea de aprendizaje

Ejemplo de tarea sencilla:

> La red recibe secuencias de spikes que representan 2 patrones distintos (A y B).
>
> Se espera que al final del entrenamiento, una neurona de salida dispara más ante A que ante B.

Decide:

- Número de clases (empieza con 2: A y B).
- Qué define un patrón (por ejemplo, distinta intensidad o combinación de entradas).

3.2 Definir la arquitectura

Piensa en capas/poblaciones:

- **Población de entrada**
  - Grupo de neuronas que reciben estímulos externos (spikes Poisson con distintas tasas o patrones).
- **Población excitatoria**
  - Grupo de neuronas LIF que intermedian entre entrada y salida.
  - Aquí se formará parte de la representación interna.
- **Población inhibitoria** (opcional pero interesante)
  - Pocas neuronas que inhiben a las excitatorias para crear competencia (solo algunas se activan).
- **Población de salida**
  - Poquitas neuronas (ej. 2) que representen cada clase: neurona 0 → A, neurona 1 → B.

Define en papel:

- Tamaño de cada población.
- Cómo se conectan (entrada → excitatoria, excitatoria → salida, etc.).
- Qué conexiones tienen plasticidad (por ejemplo, entrada → excitatoria y excitatoria → salida).

3.3 Definir el modelo de neurona

Elige un modelo concreto:

- **LIF (Leaky Integrate-and-Fire)** para:
  - Población excitatoria.
  - Población de salida.

Conceptos a fijar:

- Potencial de reposo.
- Umbral de disparo.
- Potencial tras el reset.

- Constante de tiempo de la membrana.
- Periodo refractario.

3.4 Definir la regla de plasticidad (STDP)

En papel, especifica:

- Variables que intervienen:
  - Peso sináptico `w`.
  - Traces de actividad presináptica y postsináptica (algo como `pre` y `post`).
- Dinámica cualitativa:
  - Cuando hay spike pre → se actualiza una traza.
  - Cuando hay spike post → se actualiza otra.
  - El peso se incrementa o decrementa según el orden y la coincidencia de spikes.
- Rango de pesos permitido (p.ej. [0, 1]).

3.5 Protocolo de entrenamiento

Define:

- Duración total de la simulación (ej. varios segundos simulados).
- Cómo se presenta cada patrón:
  - Ventana de tiempo donde se activa el patrón A.
  - Otra ventana para el patrón B.
  - Intervalos de reposo o ruido entre patrones.
- Número de “épocas” o repeticiones.
- Qué conexiones son plásticas durante el entrenamiento (y si luego las “congelas” para evaluar).

3.6 Protocolo de evaluación

Decide cómo medir si tu micro-red “aprendió”:

- Métricas posibles:

    - Tasa de disparo de neuronas de salida para A vs B.
    - Porcentaje de presentaciones donde la neurona de salida correcta dispara más que la incorrecta.
    - Cambios de peso promedio en las sinapsis asociadas a cada patrón.
- Visualizaciones que vas a generar:
    - Raster plot de spikes.
    - Evolución de pesos sinápticos.
    - Potencial de membrana de algunas neuronas.

---

4. Entorno de desarrollo y herramientas

4.1 Lenguaje y entorno

- **Lenguaje:** Python.
- **Entorno:** conda o venv.
- **IDE / editor:** VS Code o Jupyter Notebook (para explorar y luego mover a scripts limpios).

4.2 Librerías principales

- **Simulación neurobiológica:**
    - Brian2 → ideal para definir ecuaciones neuronales, poblaciones y STDP sin pelearte con demasiados detalles de bajo nivel.
- **Visualización:**
    - Librería de gráficos (por ejemplo, para raster plots, curvas de potencial, etc.).
- **Organización y análisis de datos:**
    - Opcional, una librería tipo pandas para organizar métricas de los experimentos.
- **Bonus cuántico:**
    - Qiskit para jugar con IBM Quantum y circuitos parametrizados.

---

5. Implementación por fases (sin entrar en código)

Fase 1 — Prototipo ultra simple

Meta: simular una sola neurona LIF con una entrada constante y ver cómo dispara spikes.

Pasos conceptuales:

1. Definir la neurona LIF (parámetros de membrana).
2. Aplicar una corriente constante o estímulo simple.
3. Simular un intervalo de tiempo.
4. Graficar:
    - Potencial de membrana vs tiempo.
    - Tiempos de spikes.

Esta fase te hace entender la dinámica básica.

Fase 2 — Población de neuronas excitatorias (sin plasticidad)

Meta: ver una pequeña red excitatoria respondiendo a una entrada estocástica.

Pasos:

1. Crear un grupo de neuronas LIF.
2. Crear un grupo de entrada (por ejemplo, neuronas con disparo Poisson con cierta tasa).
3. Conectar entrada → excitatoria con pesos iniciales (fijos).
4. Simular y graficar:
    - Raster plot de la actividad excitatoria.

Aún no hay aprendizaje, solo respuesta dinámica.

Fase 3 — Añadir STDP a una conexión

Meta: que los pesos sinápticos cambien en función de la actividad pre/post.

Pasos:

1. Marcar una conexión (por ejemplo, entrada → excitatoria) como “plástica”.
2. Definir las reglas cualitativas de STDP:

- Qué pasa con el peso si pre dispara antes que post.
- Qué pasa si post dispara antes que pre.

3. Simular con un solo tipo de patrón de entrada.
4. Observar:
   - Evolución de pesos en el tiempo.
   - Cambios en la actividad de las neuronas.

Fase 4 — Definir patrones A y B y entrenar

Meta: que la red aprenda a discriminar entre dos patrones.

Pasos:

1. Diseñar patrón A y patrón B (diferentes tasas, diferentes subconjuntos de neuronas de entrada activas, etc.).
2. Programar un ciclo donde presentas A y B intercalados durante la simulación.
3. Permitir que STDP actúe durante este entrenamiento.
4. Introducir una capa de salida que reciba de la capa excitatoria.
5. Observar si, tras el entrenamiento:
   - Alguna neurona de salida responde claramente más a A que a B.

Fase 5 — Pulir el modelo y hacer experimentos reproducibles

Meta: tener uno o dos experimentos claros, comparables.

Ideas:

- Experimento 1:
  - Entrenar la red solo con A.
  - Ver cómo las conexiones que transportan información de A se refuerzan.
- Experimento 2:
  - Entrenar con A y B.
  - Analizar si alguna neurona de salida se especializa en A y otra en B.

Para cada experimento, deja definido:

- Parámetros utilizados (número de neuronas, constantes de tiempo, intensidades de estímulo).
- Duración del entrenamiento.
- Métricas medidas.

---

6. Documentación y publicación como proyecto open source

6.1 Estructura sugerida del repositorio

- Carpeta principal con:
  - Archivos de código organizados por módulos (modelo, entrenamiento, plots, etc.).
  - Carpeta `experiments` con scripts o notebooks para correr escenarios específicos.
  - Carpeta `figures` con imágenes (raster plots, evolución de pesos, etc.).
- Archivos clave:
  - **README**:
    - Descripción del proyecto.
    - Contexto biológico y de IA.
    - Cómo instalar dependencias.
    - Cómo correr el experimento principal.
    - Ejemplos de resultados (con imágenes).
  - **LICENSE**:
    - Licencia MIT o Apache 2.0 (ideal para open source académico).

6.2 Cómo contar la historia en el README

Incluye secciones como:

- **What is this?**

SNN biológicamente inspirada que aprende a discriminar patrones temporales mediante STDP.

- **Why does it matter?**

  Está más cerca del funcionamiento real del cerebro que las redes neuronales clásicas; útil como herramienta educativa y base para investigación.

- **How it works (high level)**

  Diagrama simple de la arquitectura, patrones de entrada, plasticidad.

- **How to run the example experiment**

  Pasos resumidos (instalar dependencias, ejecutar un script, ver gráficos).

---

7. Bonus: uso conceptual de IBM Quantum para ajustar parámetros

7.1 Qué problema quieres atacar con cuántica

Ejemplo:

> Optimizar parámetros de plasticidad (p.ej. constantes de STDP, escala de refuerzo/punición) para maximizar una métrica de rendimiento: qué tan bien la red diferencia entre A y B.

7.2 Esquema de trabajo clásico–cuántico

1. Definir una **métrica de rendimiento** para tu red:
   - Por ejemplo, diferencia media de tasa de disparo de la neurona de salida correcta entre patrón A y B.
2. Elegir qué parámetros del modelo vas a optimizar (ej. 2–4 valores).
3. Crear un **circuito cuántico parametrizado** donde los parámetros del circuito representen esos hiperparámetros.
4. Bucle conceptual:
   - Asignas un conjunto de parámetros desde el circuito cuántico.
   - Corres tu simulación SNN con esos parámetros.
   - Calculas la métrica de rendimiento.

- Usas ese resultado para guiar el ajuste de parámetros cuánticos (optimización variacional).

5. Ejecutar algunas iteraciones:
    - Primeras pruebas en simulador cuántico.
    - Algunas pruebas en hardware real de IBM Quantum (limitado por tiempo).
6. Documentarlo como **prueba de concepto**:
    - Mostrar que es posible usar computación cuántica como co-ayudante en la optimización de modelos neuromórficos, aunque sea a pequeña escala.
