Résumé du Pont Python–BizHawk et Intégration avec un Réseau de Neurones

1 Objectif général

L'objectif est de créer une intelligence artificielle capable de jouer automatiquement à **Super Mario World** sur un émulateur SNES. Le réseau de neurones est écrit en Python et s'entraîne avec l'optimiseur **Adam**. Cependant, l'émulateur BizHawk utilise le **Lua** pour contrôler les entrées. Il faut donc établir un pont entre Python et BizHawk.

2 Principe du Pont Python–BizHawk

L'idée générale est de :

1. Lancer un script **Lua dans BizHawk** qui :
   - écoute des commandes venant de Python via un socket (TCP par exemple),
   - lit ces commandes (boutons à presser),
   - applique les commandes via `joypad.set`,
   - avance l'émulation image par image via `emu.frameadvance()`.
2. Depuis Python :
   - générer des actions via le réseau de neurones,
   - envoyer les actions à BizHawk sous forme de données (JSON ou tableau),
   - éventuellement recevoir des informations (RAM, image, reward).

3 Principe du contrôle BizHawk sans inclure de code Lua

Le fichier ne doit pas contenir de script Lua. Cette section décrit donc seulement le principe général, sans aucun extrait de code.

BizHawk utilise des scripts Lua pour recevoir et appliquer des commandes de contrôle. Le rôle de Lua est uniquement :

- de recevoir des instructions envoyées par Python (état des boutons),
- de les convertir en actions dans l'émulateur via `joypad.set`,
- puis de faire avancer l'émulation image par image.

Dans ce document, aucun code Lua n'est fourni conformément à votre demande.

4 Structure minimale du client Python

```
import socket, json

def send_input(buttons):
    s = json.dumps(buttons) + "\n"
    sock.sendall(s.encode())
```

```
sock = socket.create_connection(("127.0.0.1", 5555))

# Exemple : Mario avance
send_input({"Right":1, "A":0})
```

5 Ce qui est déjà fait en Python

- des couches de neurones (classe `Layer`),
- propagation avant (`forward`),
- rétropropagation manuelle,
- optimisation avec Adam entièrement codé à la main,
- architecture `Neural_Network` supportant différentes activations,
- entraînement mini-batch (SGD et Adam).

Votre réseau est donc fonctionnel et peut produire des actions à chaque frame.

6 À faire

Pour connecter le réseau de neurones à BizHawk, il reste :

1. Définir l'observation envoyée au réseau

- Capturer l'écran depuis BizHawk (via Lua ou un bridge existant), ou
- lire des valeurs RAM pertinentes (position de Mario, vitesse, etc.).

2. Convertir cette observation en entrée compatible avec votre NN

- normalisation des pixels,
- ou simple vecteur d'état si RAM.

3. Transformer la sortie du NN en actions bouton

- typiquement : sortie `softmax` ou seuils pour `A`, `B`, `Left`, etc.,
- puis construire un dictionnaire JSON pour Lua.

4. Implémenter la boucle Principal RL

- envoyer observation dans `forward()`,
- envoyer l'action au script Lua,
- récupérer reward + nouvel état,
- faire une mise à jour (Adam).

5. Définir une fonction de reward

- progression horizontale de Mario,
- éviter les morts,
- éventuellement : vitesse, pièces, niveau atteint.

7 Conclusion

Il ne manque que le **pont avec BizHawk** et la **boucle d'entraînement RL** qui utilise ce pont.