Simulateur du MIND 1024

Guide d'une application qui reconnaît des dessins similaires

1. Installation

Il suffit de télécharger le fichier *'Application.zip'*. Une fois téléchargé, double-cliquez dessus pour soutirer le dossier avec tous les fichiers nécessaires :

Dossier *'Hopfield'*

- Fichier *'index.html'*
- Dossier *'scripts'*
  - Fichier *'canvas.js'*
  - Fichier *'controls.js'*
  - Fichier *'hopfield.js'*
  - Fichier *'main.js'*
- Dosser *'styles'*
  - Fichier *'main.css'*

2. Exécution du code

L'application est implémentée à l'aide du *Javascript.* On ouvre le fichier *'index.html'* avec un navigateur, (*Chrome,* par exemple), pour exécuter le code en local.

3. Modification des paramètres

Après avoir exécuté le code, plusieurs carrés s'affichent, ainsi que deux boutons. Les carrés se trouvant au-dessus des boutons correspondent aux dessins d'entrée, (ce que l'on va apprendre au réseau). Le carré en bas sert d'un nouvel état, c'est-à-dire, le dessin que le simulateur va tenter de correspondre à un dessin « mémorisé ».

- Pour modifier la taille de tous les carrés, il suffit de changer la valeur du variable *'CANVAS_SIZE'*.
```
4   var CANVAS_SIZE = 300;
```
- Le variable *'PICTURE_SIZE'* permet de régler le nombre de pixels que l'on peut dessiner dans les carrés.
```
3   var PICTURE_SIZE = 5;
```
Si on fixe sa valeur à *5*, par exemple, cela implique qu'un carré va avoir *5 x 5 = 25* pixels. Le nombre de neurones du réseau dépend sur le nombre de pixels dans le dessin. Le MIND 1024 possède 1024 neurones et du coup, afin de bien simuler cette machine, on peut avoir jusqu'à *32 x 32 = 1024* pixels.
- Le variable *'DRAWING_COUNT'* modifie le nombre de dessins d'entrée.
```
3   var DRAWING_COUNT = 3;
```
Faites attention : plus ce nombre augmente, plus le simulateur ne va pas être capable de bien correspondre le nouveau dessin à un dessin « dans sa mémoire ».

4. Les boutons

Il y a deux boutons : un à droite, étiqueté *'Apprentissage',* et l'autre à gauche, étiqueté *'Exécuter'*.

- Celui à droite commence la génération du réseau de *Hopfield.* Ce réseau sera de taille $(PICTURE\_SIZE)^2$ et sera construit à partir des dessins d'entrée fournis. Le but est de créer un réseau qui représente une combinaison de tous les dessins, (une « mémorisation » de tous les états). Il devient difficile de générer une mémorisation exacte au fur et à mesure que le nombre de dessins augmente.

- Une fois que le réseau aura appris tous les dessins d'entrée et un nouveau dessin sera fourni, (l'état cible), le bouton à gauche serait utilisé pour commencer l'algorithme qui cherche à trouver le dessin mémorisé correspondant le mieux au nouveau dessin.

Pour apprendre en quoi consiste un réseau de Hopfield et pour comprendre les algorithmes utilisés par cette application, je vous conseille de jeter un coup d'œil sur les slides de ma présentation : 'MIND_1024_Présentation.pdf'.

5. Utilités éventuelles du simulateur

Un réseau de *Hopfield* est souvent utilisé pour reconnaître des motifs. Ici, il s'agit de petits dessins. Il faut garder à l'esprit qu'il est plus probable que le simulateur produise le résultat souhaité si le nouveau dessin fourni est déjà assez similaire à un dessin mémorisé. Ci-dessous, nous parlerons de deux simulations ou « jeux », que l'application nous permet d'implémenter :

A. Reconnaissance des images de taille jusqu'à *32 x 32* pixels :

- Il est possible de fournir des dessins d'assez grand taille au réseau. On pourrait donc imaginer que l'application soit utilisée pour la reconnaissance des petites images. On peut fournir des dessins carrés de taille *32 x 32* pixels, mais utilisant que des pixels noirs et blancs. (Les neurones du réseau sont des états binaires).
- Alors, après avoir choisi quelques images, il faudrait les convertir en images carrées de la bonne taille et en noir et blanc.
- Cette conversion peut être implémentée automatiquement, mais vous devriez ajouter une fonction dans le code pour la faire puisque cela n'existe pas en ce moment.

B. Reconnaissance des codes QR (comme dans *'MIND_1024_Présentation.pdf'*) :

- L'idée serait de générer des codes QR de la bonne taille, ($\leq$ 1024 pixels). Par exemple, des phrases courtes peuvent être représentées par un code QR de taille *29 x 29* pixels.
- Tout simplement, on inventerait quelques phrases et obtiendrait des codes QR qui les représentent. En revanche, lors de l'étape d'apprentissage, on rencontrait un problème : l'implémentation ne permet pas de fournir automatiquement des dessins et donc on devrait dessiner tous les pixels à la main !
- Alors, je vous conseille d'ajouter une fonction pour fournir immédiatement les dessins d'entrée à partir des codes QR générés.
- Puis, on pourrait apprendre les codes au réseau.
- Il faudrait alors inventer une nouvelle phrase avant l'exécution du code. Comme souligné avant, cette phrase devrait être similaire à une phrase déjà mémorisée.
- Il serait intéressant de varier la similarité des phrases pour découvrir les capacités du réseau.

6. Conclusion

Nous avons décrit les fonctionnalités de l'application et avons mis en avant quelques « jeux » qui montrent son utilité. Maintenant, c'est à vous d'imaginer d'autres utilités éventuelles du simulateur du MIND 1024 !