TP n°3 IA : Réseaux de nuerons Perceptron

Ishak AYAD

1 Introduction

Dans ce TP on souhaite implémenter un réseau de nuerons simple de type **Perceptron** [1], au départ le réseau de nuerons est destiner a différencier deux types de motifs : la lettre "A" et la lettre "C". Puis une extension sera réalisé pour que ce réseau distingue les 26 lettres de l'alphabet.

Les motifs sont codé sur des grilles de 4×5 représentant 20 pixels pouvant être soit blancs ou noirs.

1.1 Définition

Le perceptron peut être vu comme le type de réseau de neurones le plus simple. C'est un classifieur linéaire. Ce type de réseau neuronal ne contient aucun cycle (il s'agit d'un réseau de neurones à propagation avant).

Dans sa version simplifiée, le perceptron est mono-couche et n'a qu'une seule sortie (booléenne) à laquelle toutes les entrées (booléennes) sont connectées. Plus généralement, les entrées peuvent être des nombres réels.

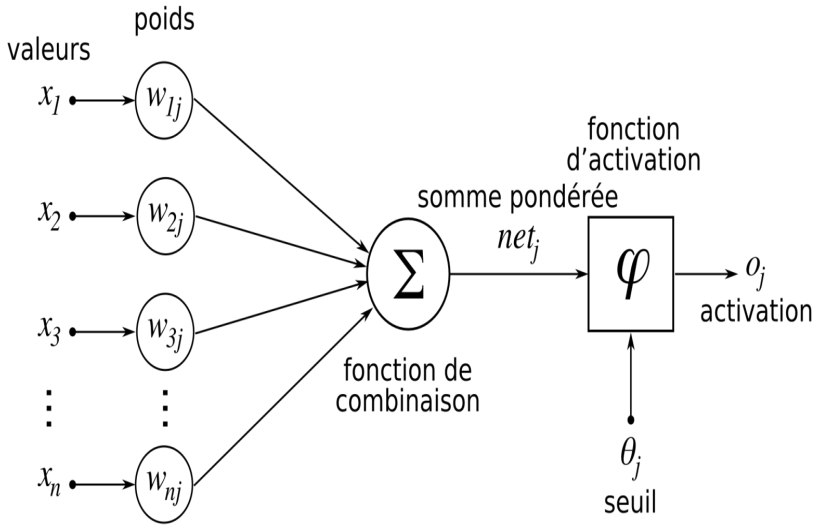
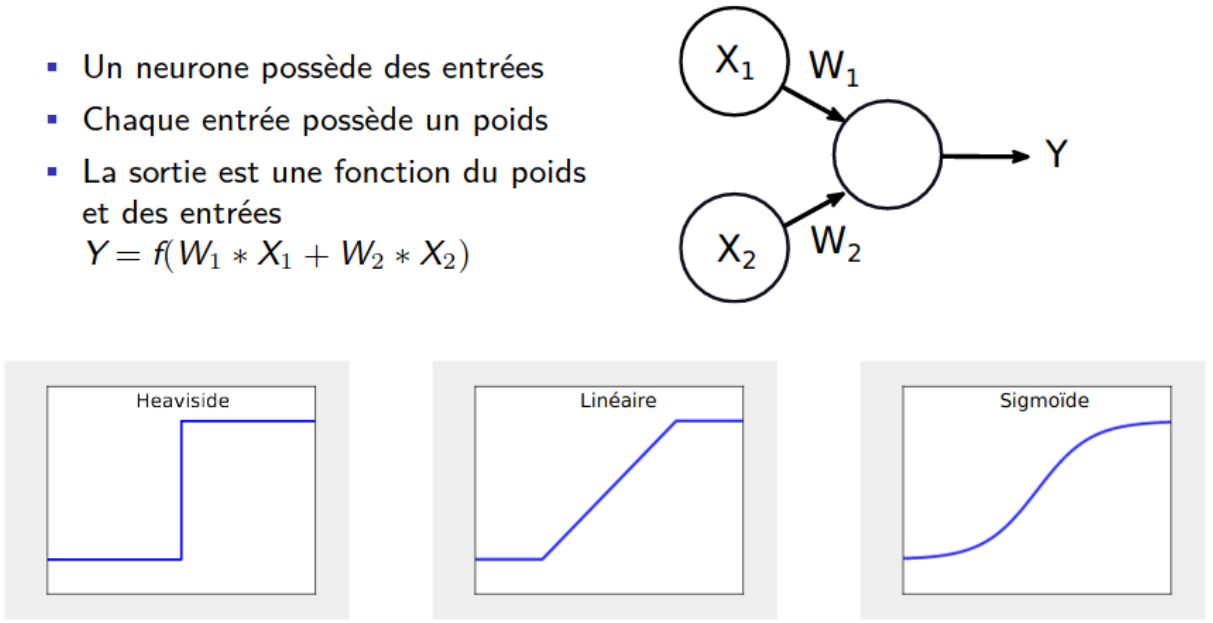


FIGURE 1 – Schéma des calculs

1. Un algorithme d'apprentissage supervisé de classifieurs binaires. Il s'agit d'un neurone formel muni d'une règle d'apprentissage qui permet de déterminer automatiquement les poids synaptiques de manière à séparer un problème d'apprentissage supervisé

Dans la figure 1 le mécanisme du perceptron est illustré Les entrées i_1 à i_n sont multipliées avec les poids W_1 à W_n, puis sommées, avant qu'une fonction d'activation soit appliquée.

1.2 Perceptron simple



1.2.1 Règle de Hebb

La règle de Hebb, établie par Donald Hebb, est une règle d'apprentissage des réseaux de neurones artificiels dans le contexte de l'étude d'assemblées de neurones.

Cette règle suggère que lorsque deux neurones sont excités conjointement, ils créent ou renforcent un lien les unissant.

Dans le cas d'un neurone artificiel seul utilisant la fonction signe comme fonction d'activation cela signifie que :

$$W_i' = W_i + \alpha(\mathrm{Y}.X_i)$$

Où W_i' est représente le poids i corrigé et α représente le pas d'apprentissage.

Cette règle n'est malheureusement pas applicable dans certains cas bien que la solution existe.

1.2.2 Règle d'apprentissage du perceptron (loi de Widrow-Hoff)

Le perceptron de Frank Rosenblatt est très proche de la règle de Hebb, la grande différence étant qu'il tient compte de l'erreur observée en sortie.

Cette fonction est recommandée lorsque la tangente hyperbolique (tanh) est utilisée comme fonction d'activation.

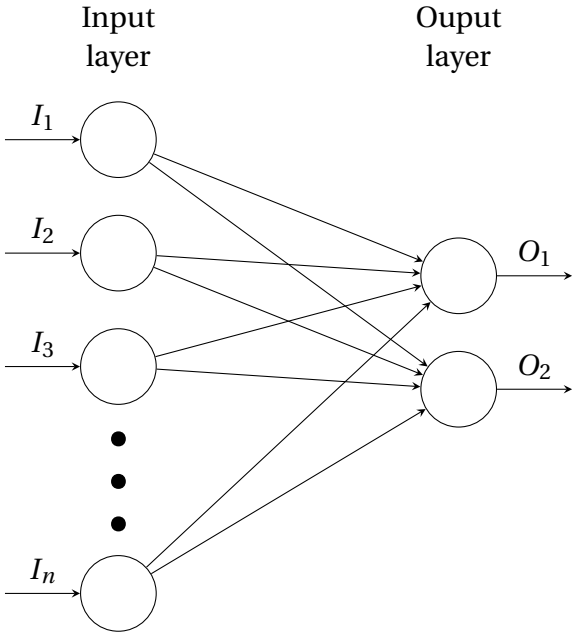
$$W_i^{'} = W_i + \alpha(Y_t\text{-Y})X_i$$

— $W_i^{'}$ = le poids i corrigé
— W_i = le poids i actuel
— α = le taux d'apprentissage
— Y_t = sortie attendue
— Y = sortie observée
— X_i = l'entrée du poids i pour la sortie attendue Y_t

2 Questions préliminaires

— Le schéma du réseau de nuerons :

Le Perceptron est formé de n = 20 nuerons d'entrées représentant les pixels du motif en entrée et deux nuerons en sorties définissants les deux motifs a reconnaître.



— Codage du réseau de nuerons :

Le réseau de nuera est présenté sous la forme d'une structure de données contenant une matrice de poids, un tableau d'entrée et un pointeur vers une fonction qui désigne la fonction d'activation. Plus une structure de données pour stocker les données d'entraînement/apprentissage du perceptron, voir la section [3].

— Initialisation des poids :

L'initialisation des poids s'effectue de manière aléatoire, on génère des nombres réels entre [0,1] pour l'ensemble des poids des nuerons.

— La propagation de l'information : **Famille des réseaux à propagation avant ou feed-forward**

Lorsqu'un réseau propage l'information au travers de ses couches successives, de la couche d'entrée à la couche de sortie, et sans avoir un retour de l'information en arrière, alors il s'agit d'un réseau à propagation avant ou feed-forward en anglais.

— Variation des poids pendant l'apprentissage :

Les poids varient en fonction de la règle d'apprentissage du perceptron (**loi de Widrow-Hoff**). La loi de Widrow-Hoff est la règle d'apprentissage du perceptron la plus connue. Elle est locale, c'est à dire que chaque cellule de sortie apprend indépendamment des autres. Elle peut s'écrire de la manière suivante : $W_i^{'} = W_i + \alpha(Y_t\text{-Y})X_i$

— Le temps que l'apprentissage prendra :

Le temps d'apprentissage dépend de l'erreur local et global pendant l'apprentissage, car on quitte la boucle d'apprentissage quand l'erreur global est nulle.

En-outre la complexité de l'apprentissage pour une itération est de rang $\mathcal{O}$(m+n), ou m est la taille des entrées et n la taille des sorties du perceptron.

3 Implémentation du perceptron

— **Structure de données :**

```c
//Pointeur vers la fonction d'activation
typedef float (*outActivation)(float x);

typedef struct {
    //La taille des entrees
    int output_layer_size;
    //La taille des sorties
    int input_layer_size;
    //Les sorties
    float* output;
    //Matrices des poids
    float* weight;
    //La fonction d'activation
    outActivation funcActivation;
}perceptron;

```

Listing 1 – Structure de données utilisé pour implémenter le perceptron

4 Déroulement

4.1 Motifs d'entrée pour l'apprentissage

Les motifs d'entrée utilisés pour l'apprentissage sont stocké dans un fichier textuel, les 20 premiers bites sont réservés pour les entrées des neurones et le reste des bits pour les sorties désirées.

On stocke ses valeurs dans une structure de données pour avoir un bon passage entre les fonctions du système.

```
1
2 //Structure de donnees pour stocker les entree d'apprentissage
3 typedef struct{
4     int size_input, size_output;
5     int line;
6     float** input;
7     float** output;
8 }DATA;
9
```

Listing 2 – Structure de données utilisé pour stocker les données d'apprentissage

4.2 Affichage et lecture des motifs

L'affichage et la lecture des motifs en entrer pour tester le réseau de neurones est fait en utilisant la librairie OpenGL[2], on affiche une matrice vide de 4×5 case et le remplissage se fait en cliquant sur les cases, pour afficher le résultat l'utilisateur doit taper [Entrer] et le résultat sera affiché sur la console.

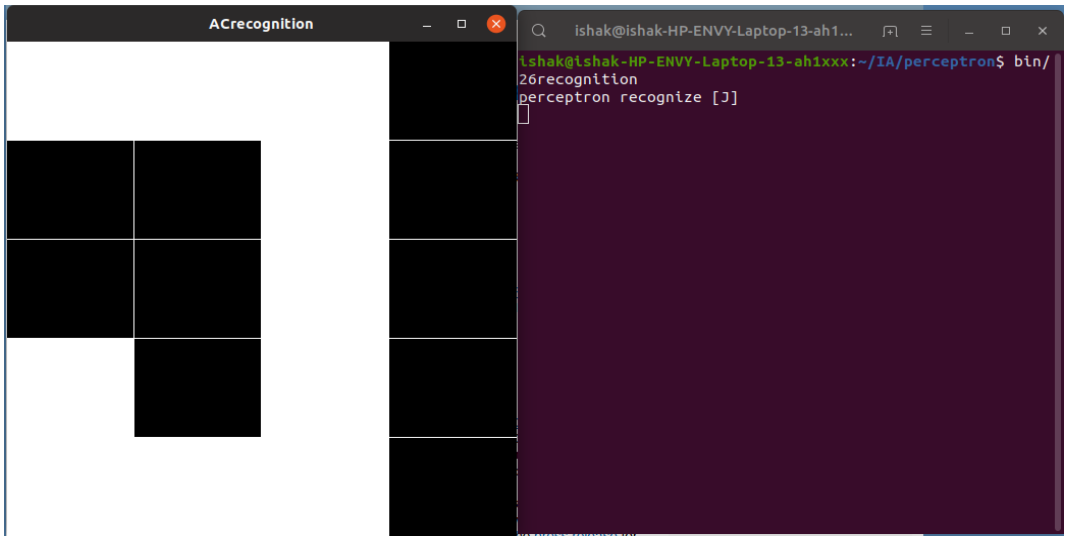


FIGURE 2 – Interface homme machine pour la reconnaissance des lettres

Pour compiler le programme veiller taper ouvrez le terminal :

— $ make make test [pour compiler le programme]
— $ bin/ACrecognition [pour lancer la reconnaissance des deux motifs A et C]
— $ bin/26recognition [pour lancer la reconnaissance des 26 lettres de l'alphabet]

```
1 $ sudo apt-get update
2 $ sudo apt-get install libglu1-mesa-dev freeglut3-dev mesa-common-dev
3
```

Listing 3 – Installation OpenGL sous Ubuntu

2. OpenGL (Open Graphics Library) est un ensemble normalisé de fonctions de calcul d'images 2D ou 3D lancé par Silicon Graphics en 1992.

4.3 Analyse expérimentale

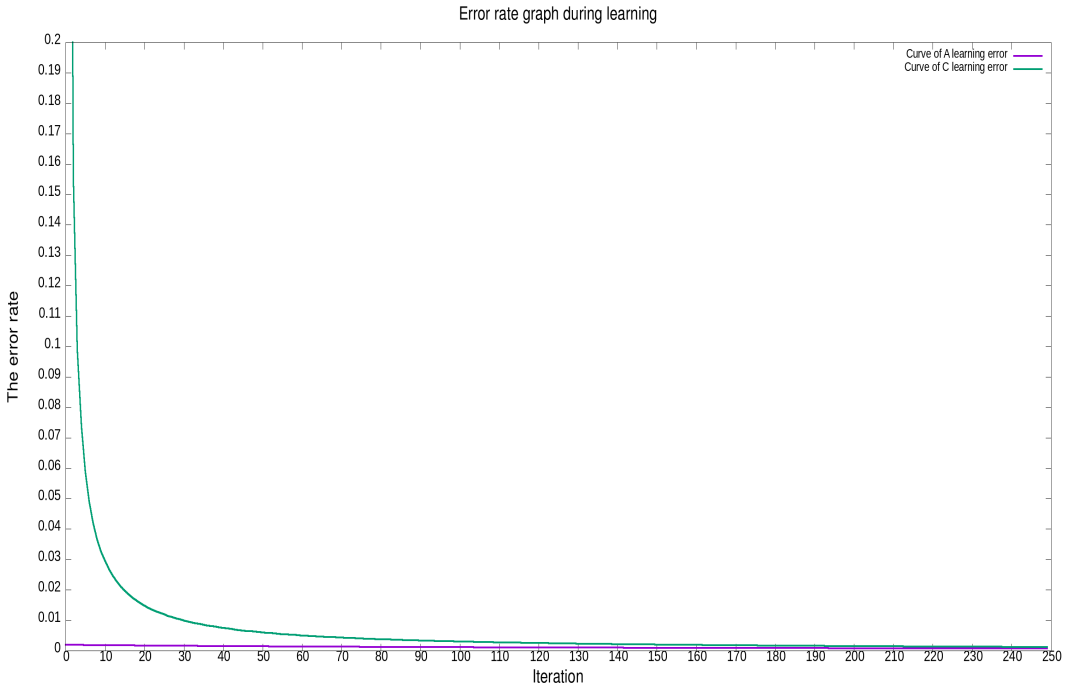


FIGURE 3 – Analyse expérimentale, réseau de nuerons apprend A en premier

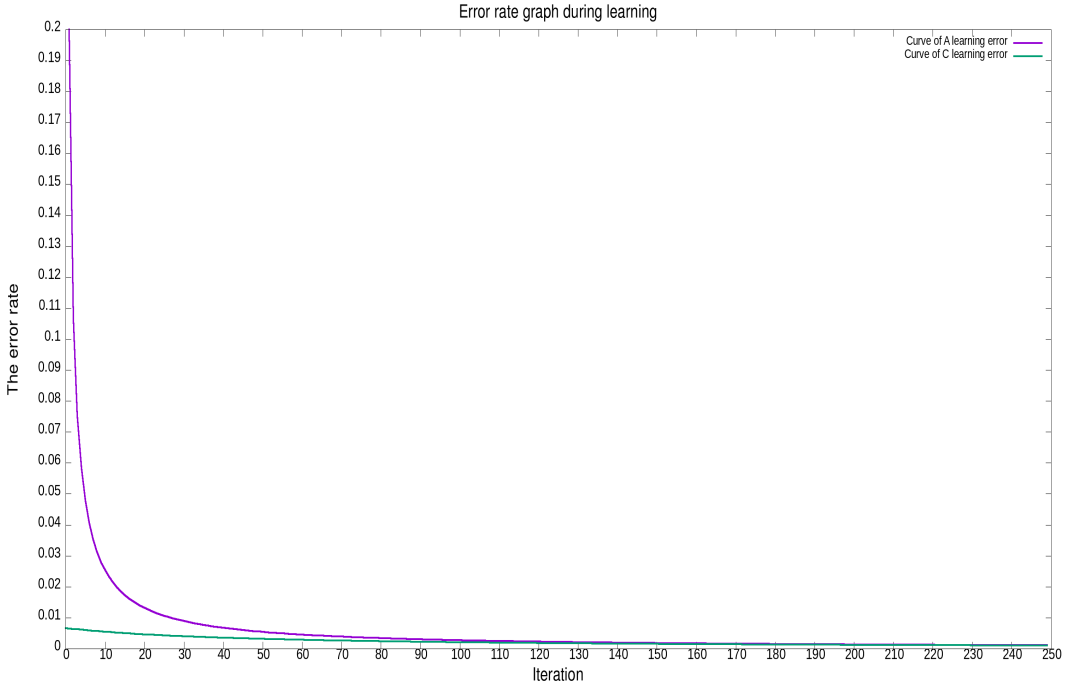


FIGURE 4 – Analyse expérimentale, réseau de nuerons apprend C en premier

Dans les deux cas "apprentissage pour A en premier" et "Apprentissage pour A en premier" la remarque est la même, l'erreur pour le premier apprentissage reste trop basse car l'initialisation des poids se fait aléatoirement entre [0,1] ce qui donne une chance au perceptron de se rapprocher du résultat attendu, mais pour l'apprentissage du deuxième motif l'erreur commence très haute et décroître jusqu'à affoler le zéro.

4.4 Entrées bruitées

Le bruitage des données est modélisé par la probabilité d'échange des valeurs des pixels du motif d'entrée en fonction du niveau de bruit correspondant.

On appelle se bruit **Gaussian noise** le bruit gaussien, du nom de Carl Friedrich Gauss, est un bruit statistique ayant une fonction de densité de probabilité égale à celle de la distribution normale, également appelée distribution gaussienne En d'autres termes, les valeurs que le bruit peut prendre sont réparties en Gauss.

La fonction de densité de probabilité p d'une variable aléatoire gaussienne z est donnée par :

$$\mathscr{P}_G = \frac{1}{\sigma\sqrt{2\pi}} \times \exp(-\frac{(z-\mu)^2}{2\sigma^2})$$

où z représente le niveau de gris, μ la valeur moyenne et σ l'écart type.

dans notre cas z ∈ (1,0).

5 Questions compréhension

— En regardant la matrice des poids, l'apprentissage effectue :

Ce que nous faisons, c'est de rendre la perte aussi proche que possible de zéro la prochaine fois que nous utiliserons de nouveau le réseau pour une prédiction.

— La robuste du réseau de neurones :

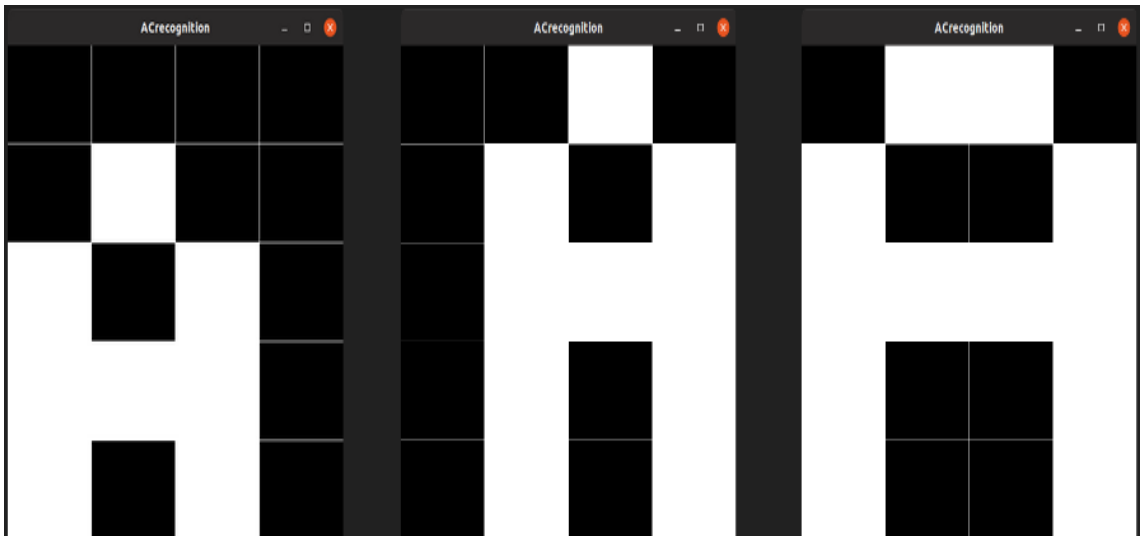
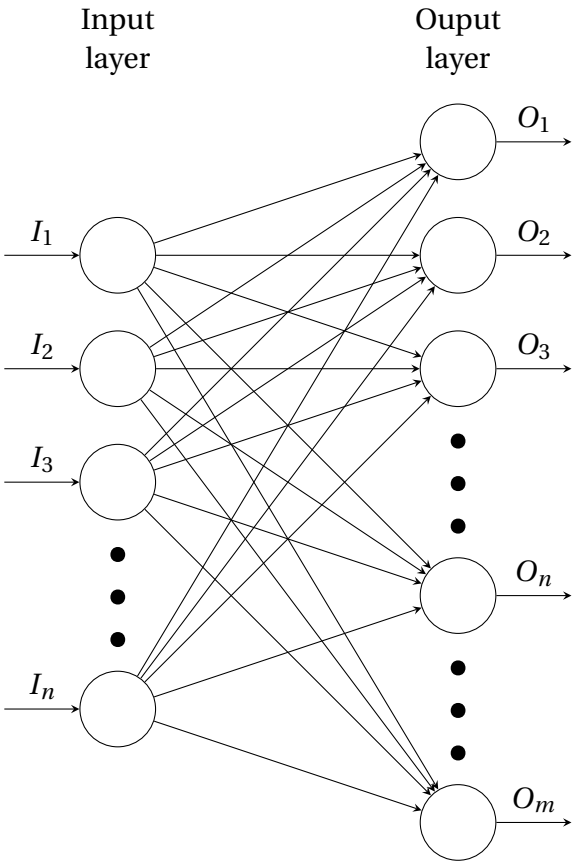


FIGURE 5 – Test de robuste du réseau de neurones

Dans la figure [5] j'ai effectué un test de robuste sur le réseau de neurones et le résultat était le même dans les trois cas "le réseau de neurones reconnaît la lettre A", on peut observer que les deux motifs donnés en entrée pour le réseau de neurones sont différents que des troisièmes motifs sur la figure ie : le motif donnée en entrée pour l'apprentissage.

— Réseau de nuerons pour l'apprentissage et la reconnaissance des 26 lettres de l'alphabet :

Le Perceptron est formé de n = 20 nuerons d'entrées représentant les pixels du motif en entrée et m = 26 nuerons en sorties définissants les lettres d'alphabet a reconnaître en ordre (A, B, C, ..., Y, Z).



Le motif reconnu sera celui qui a la valeur maximale dans le tableau de sortie, et puis on fait une petite conversion à partir de l'index du motif dans le tableau vers ASCII.

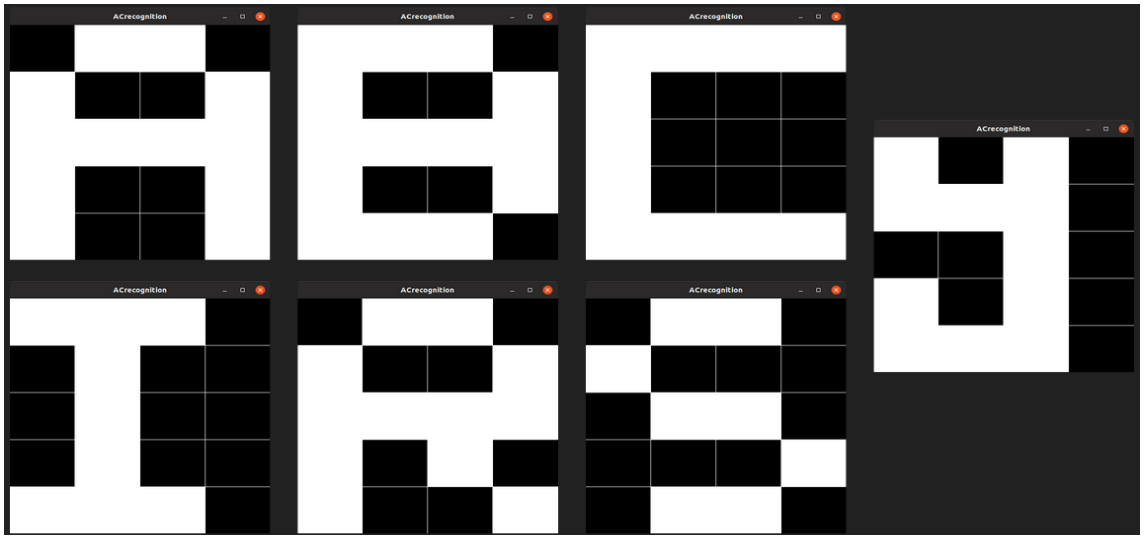


FIGURE 6 – Quelques motifs pour tester la reconnaissance des 26 lettres