**Partie 1**

Répondez par **Oui** ou **Non** (si **Non**, veuillez justifier votre réponse) :

1. **L'apprentissage non supervisé** utilise des étiquettes pour l'entraînement.

Non, C’est l’apprentissage supervisé qui en utilise

1. **Max-pooling** conserve la valeur maximale d'une région spécifique d'une image.

Oui

1. Un mot très fréquent dans tous les documents aura une valeur **TF-IDF** élevée.

Non, il le TF-IDF permet de montrer l’importance d’un mot dans un corpus de texte. Si le mot est très présent dans tous les texte, il aura un TF-IDF faible.

1. Le **redimensionnement** d'une image ne change pas sa résolution. Non, la résolution d’une image correspond au nombre de pixel de l’image donc à sa taille.
2. Le **perceptron** utilise une fonction d'activation de type seuil.

Oui mais ce n’est pas la seule

1. **L'écart-type** mesure la dispersion d'un ensemble de données.

Oui

1. Un **perceptron** à une seule couche peut approximer toute fonction booléenne.

Non, il ne peut apprendre que des problèmes linéaires séparables

1. **TF-IDF** est insensible à l’ordre des mots dans un document, contrairement aux modèles basés sur des n-grams.

Oui

1. Une **couche de pooling** introduit nécessairement des paramètres à apprendre.

Non, elle n’introduit aucun paramètre à apprendre

1. Le problème **XOR** a initialement démontré les limites du **perceptron**.

Oui car le perceptron ne peut résoudre que des problèmes linéairement séparable.

**Partie 2**

Choisissez **la** ou **les** **réponses** **correctes** : Complétez les affirmations suivantes :

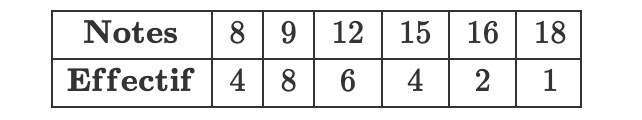
1. Le **Perceptron**, est un algorithme d'apprentissage **supervisé**
2. **Le Perceptron** utilise des étiquettes pour l'entraînement.
3. Le **POLLING** réduit le nombre de paramètres dans un **réseau de neurones convolutionnels**
4. La **QUERY** est pour GraphQL, ce que **GET** est pour les APIs REST.
5. **La normalisation** des images aide à accélérer l'apprentissage des réseaux.

**Partie 3**

Réaliser les exercices suivantes :

* **Exercice 1 :**

Les notes obtenues à un devoir par un groupe de 25 élèves.

****

**Déterminer l’écart-type des notes obtenues.**

**‘’’ python**

import numpy as np

# Données

notes = np.array([8, 9, 12, 15, 16, 18])

effectif = np.array([4, 8, 6, 4, 2, 1])

# Calcul de la moyenne pondérée (moyenne des notes)

moyenne = np.sum(notes \* effectif) / np.sum(effectif)

# Calcul de l'écart-type

variance = np.sum(effectif \* (notes - moyenne)\*\*2) / np.sum(effectif) ecart\_type = np.sqrt(variance)

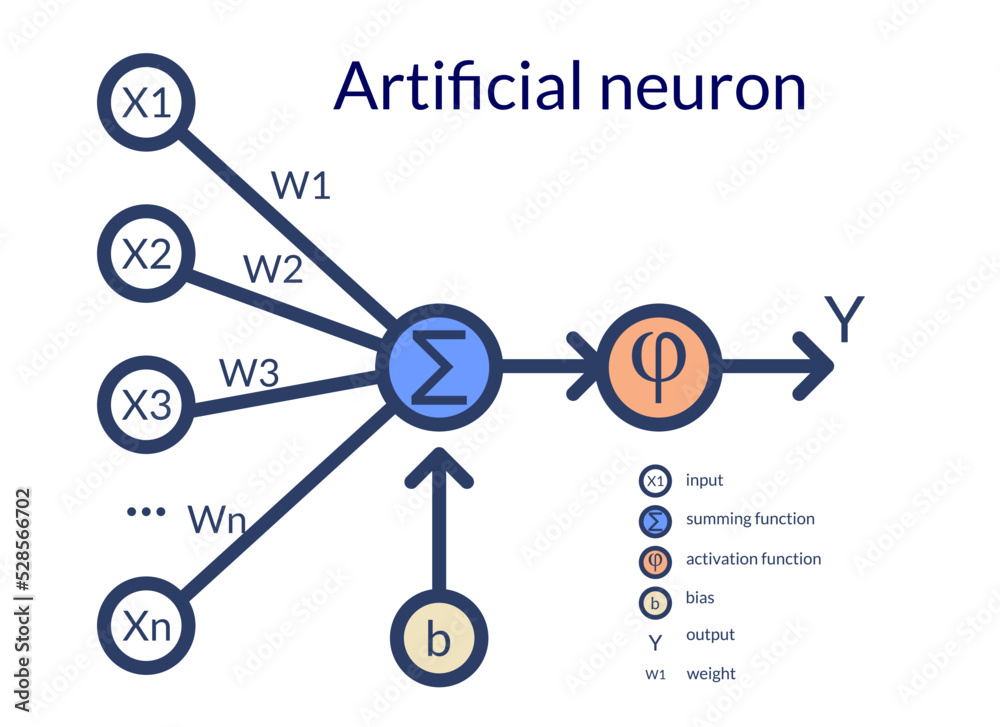
print(ecart\_type)

‘’’

* np.float64(3.0473595127585456)

L’écart type des notes obtenue est de 3,05

* **Exercice 2 :**

****

Pour concevoir un perceptron capable de prédire l'approbation (OUI/NON) d'un prêt, nous allons prendre en compte trois critères :

* Salaire du demandeur, **(X1)**
* Salaire de son conjoint **(X2)**
* Salaire de son père **(X3)**

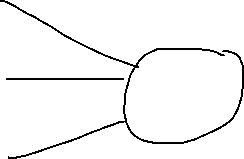
Répondez aux questions suivantes :

1. Dessinez un diagramme du perceptron.

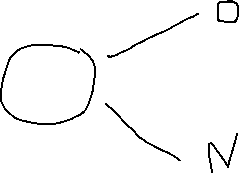


X1

X2 (X1 \* w1) +



(X2 \* w2) + Step



(X3 \* w3) +

b

X3



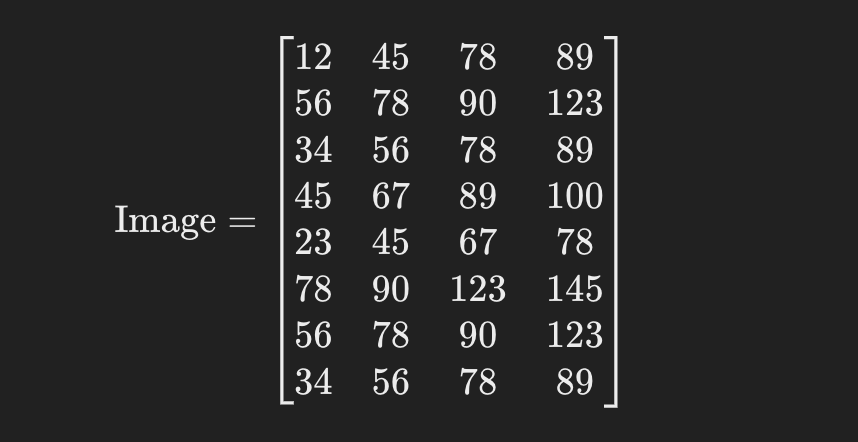
1. Quelle fonction d'activation est adaptée entre « STEP » et « SIGMOID » ?

La fonction STEP est plus adaptée car elle permet de classé les sortie en deux catégories et c’est ce que nous voulons

1. Établissez l'équation de la fonction de transfert en tenant compte du biais.

Z = (X1 \* w1) +(X2 \* w2) + (X3 \* w3) + b

* **Exercice 3:**

Considérons une image de dimensions 8×4. 

1. Appliquer un redimensionnement 4×2 basé sur la méthode AVG Polling.

Je vais faire la moyenne des valeurs de chaque groupe de 4 pixel :

(12 + 45 + 56 + 78) / 4 = 47,75 ~= 48 (Ex pour le premier)

[ 48 95

51 89

59 103

56 95 ]

1. Appliquer une normalisation sur l’intervalle [0, 1].

Pour normaliser, je vais diviser tous les nombres par 255.

[ 0,188 0,373

0,2 0,349

0,231 0,404

0,220 0,373 ]

* **Exercice 4 :**

Soit les documents

* **“Hello Ynov Paris”**
* **“Hello Paris France”**
* **“Bonjour Paris Ynov”**

calculer le TF-IDF (Hello, Ynov, Paris).

TF-IDF(Hello) = 1/3 \* log(3/2) = 0,333 \* 0,176 = 0,059

TF-IDF(Ynov) = 1/3 \* log(3/2) = 0,333 \* 0,176 = 0,059

TF-IDF(Paris) = 1/3 \* log(3/3) = 0,333 \* 0 = 0

**Partie 4**

Une entreprise spécialisée dans l'analyse d'images médicales a récemment déployé un système de machine learning sur le cloud pour automatiser la détection de pathologies à partir d'IRM et de radiographies.

Ce système utilise un pipeline complet d'**image preprocessing** (redimensionnement, normalisation, augmentation de données), un réseau **deep learning** convolutionnel avec des couches de **pooling** pour la détection d'anomalies, et un module de classification basé sur des méthodes de **machine learning** supervisé.

Parallèlement, un module d'indexation et de recherche de rapports médicaux utilise **TF-IDF** pour extraire et pondérer les mots-clés des comptes rendus médicaux afin d'accélérer les recherches.

Cependant, après la mise en production, des problèmes sont survenus :

* Le modèle de classification affichait des performances incohérentes sur les nouvelles images, malgré des résultats prometteurs en phase de test.
* Le module d'extraction textuelle retournait des résultats biaisés et peu pertinents pour certains types de requêtes.
* Une enquête a révélé que les étapes de prétraitement et certaines configurations du modèle n'étaient pas optimisées ou étaient absentes lors du déploiement, et que le jeu de données initial était insuffisant.

**Question :**Parmi les mesures suivantes, lesquelles auraient pu immédiatement améliorer la performance et la fiabilité du système déployé ?

* Utilisation d'une méthode de **normalisation** cohérente pour toutes les images d'entraînement, de validation et de test.

Oui

* Ajout d'une couche de **dropout** dans le réseau de neurones pour limiter le surapprentissage.

Oui

* Application d'une technique de **validation croisée** lors de l'entraînement du modèle pour éviter un biais lié à la répartition des données.

Non

* Augmentation du dataset en utilisant des techniques d'**image preprocessing** comme la rotation et la symétrie.

Peux causer un pb de surapprentissage

* Recalibrage des poids de **TF-IDF** en fonction de la fréquence des mots dans les nouveaux documents.

Oui

**Partie 5**

Le contenu suivant présente un fichier **main.py**, deux erreurs se sont glissées dans l'implémentation.

**Identifiez-les** et apportez les **corrections** nécessaires.

**main.py—---------------------------------------------------**

***from*** *sklearn.feature\_extraction.text* ***import*** *TfidfVectorizer*

***from*** *scipy.spatial.distance* ***import*** *cosine*

***from*** *scipy.stats* ***import*** *pearsonr*

*# Corpus*

***texts*** *= [ "Le traitement du langage naturel est fascinant.", "Le traitement des langues est une branche de l'intelligence artificielle.", "L'analyse de texte est utilisée pour la traduction automatique." ]*

*# Vectorisation TF-IDF*

***vect*** *= TfidfVectorizer()*

***tfidf\_mat*** *=* ***vect****.fit\_transform(****texts****).toarray()*

*# Étape 2 : Similarité Cosine*

*cosine\_similarity = 1 - cosine(tfidf\_mat[0], tfidf\_mat[1])*

*# Étape 3 : Corrélation de Pearson*

*pearson\_corr = pearsonr(tfidf\_mat[0], tfidf\_mat[1])*

**—------------------------------------------------------------**