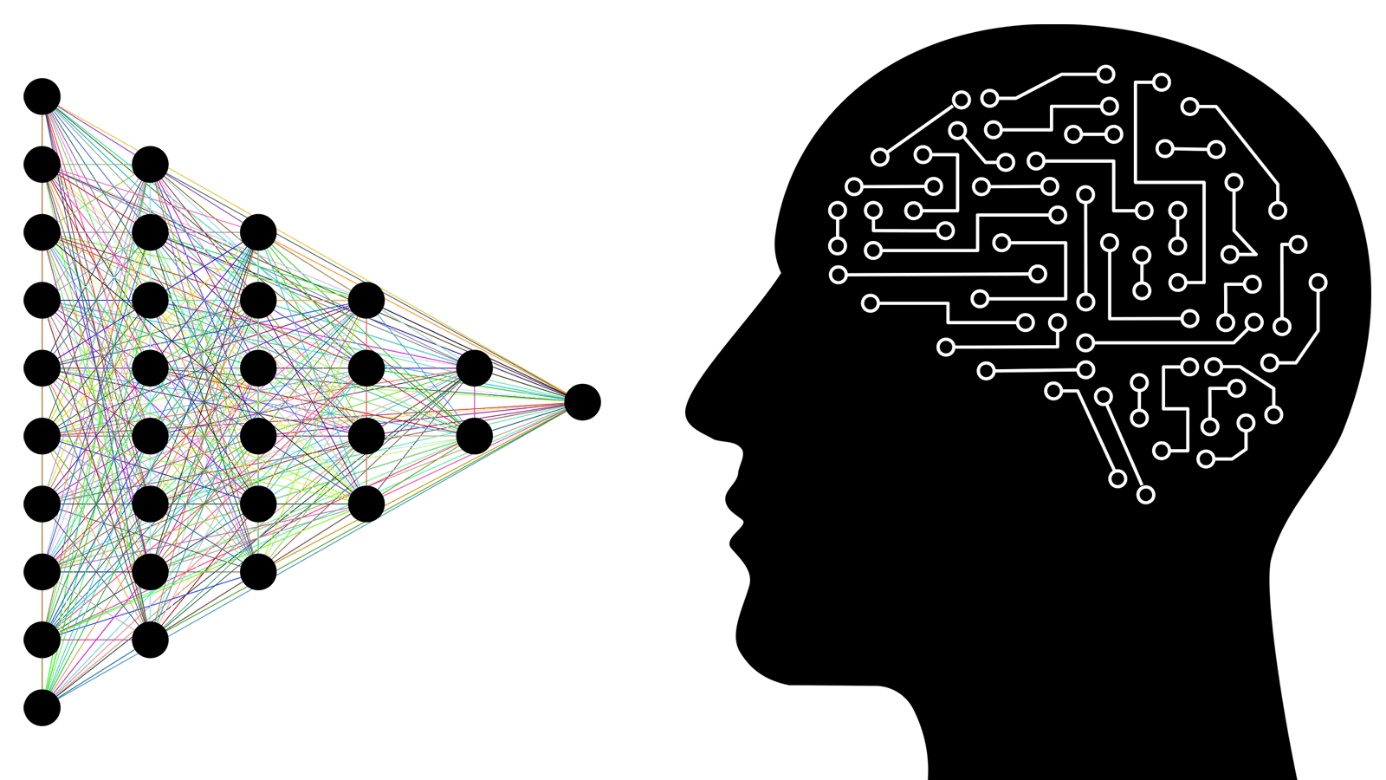
**Rapport**

**TP Deep Learning**



*Partie 1 : réseaux de neurones*

Une image contenant Graphique, conception

Description générée automatiquement

Sommaire

# **Partie 1 : Perceptron**

1. **data\_train**
   * **Taille** : torch.Size([63000, 784])
   * **Description** : Contient les images d'entraînement, chaque image étant aplatie en un vecteur de 784 pixels (28x28 pixels). La taille totale est (nombre\_d\_exemples\_d\_entraînement, nombre\_de\_pixels\_par\_image), donc (63000, 784).
2. **label\_train**
   * **Taille** : torch.Size([63000, 10])
   * **Description** : Contient les étiquettes associées aux images d'entraînement, représentées comme des vecteurs one-hot de 10 classes. La taille est (nombre\_d\_exemples\_d\_entraînement, nombre\_de\_classes), donc (63000, 10).
3. **data\_test**
   * **Taille** : torch.Size([7000, 784])
   * **Description** : Contient les images de test, chaque image étant également aplatie en un vecteur de 784 pixels. La taille est (nombre\_d\_exemples\_de\_test, nombre\_de\_pixels\_par\_image), donc (7000, 784).
4. **label\_test**
   * **Taille** : torch.Size([7000, 10])
   * **Description** : Contient les étiquettes associées aux images de test, représentées comme des vecteurs one-hot de 10 classes. La taille est (nombre\_d\_exemples\_de\_test, nombre\_de\_classes), donc (7000, 10).
5. **w**
   * **Taille** : torch.Size([784, 10])
   * **Description** : Matrice des poids du modèle. Chaque entrée représente la connexion entre un pixel d'entrée et une classe de sortie. La taille est (nombre\_de\_pixels\_par\_image, nombre\_de\_classes), donc (784, 10).
6. **b**
   * **Taille** : torch.Size([1, 10])
   * **Description** : Vecteur des biais du modèle, chaque biais étant associé à une classe de sortie. La taille est (1, nombre\_de\_classes), donc (1, 10).
7. **y**
   * **Taille** : torch.Size([5, 10])
   * **Description** : Résultat du produit matriciel entre x et w, plus b. La taille est (batch\_size, nombre\_de\_classes). Si batch\_size est 5, alors y.shape sera (5, 10) lors de l'entraînement.
8. **t**
   * **Taille** : torch.Size([5, 10])
   * **Description** : Étiquette vraie pour les exemples courants dans le lot. Représentée comme un vecteur one-hot. La taille est (batch\_size, nombre\_de\_classes). Si batch\_size est 5, alors t.shape sera (5, 10).
9. **grad**
   * **Taille** : torch.Size([5, 10])
   * **Description** : Gradient de la fonction de perte par rapport aux sorties du modèle (t-y). La taille est (batch\_size, nombre\_de\_classes), donc (5, 10) pendant l'entraînement si batch\_size est 5.
10. **acc**
    * **Taille** : tensor([5970.])
    * **Description** : Nombre total d'exemples correctement classifiés pendant l'évaluation du modèle. C'est une valeur scalaire accumulée pendant les tests.
11. **nb\_data\_test**
    * **Taille** : 7000
    * **Description** : Nombre total d'exemples dans l'ensemble de test.
12. **acc/nb\_data\_test**
    * **Taille** : tensor([0.8533])
    * **Description** : Précision du modèle sur l'ensemble de test, calculée comme la proportion d'exemples correctement classifiés par rapport au nombre total d'exemples de test. C'est un scalaire qui représente la précision du modèle.

# **Partie 2 : Shallow Network**

sdfsd