



Le réseau de neurones qui écrivait des romans

Atelier Pycon FR 2023



onepoint.
Au-delà de l'évidence

Vos accompagnatrices

Bérengère MATHIEU

- Master en informatique, spécialité Analyse d'Image
- Doctorat dans le domaine de l'image et de l'intelligence artificielle
- 5 ans d'expérience dans le milieu professionnel
- Enseignante/Formatrice depuis 10 ans
- Contact : b.mathieu@groupeonepoint.com

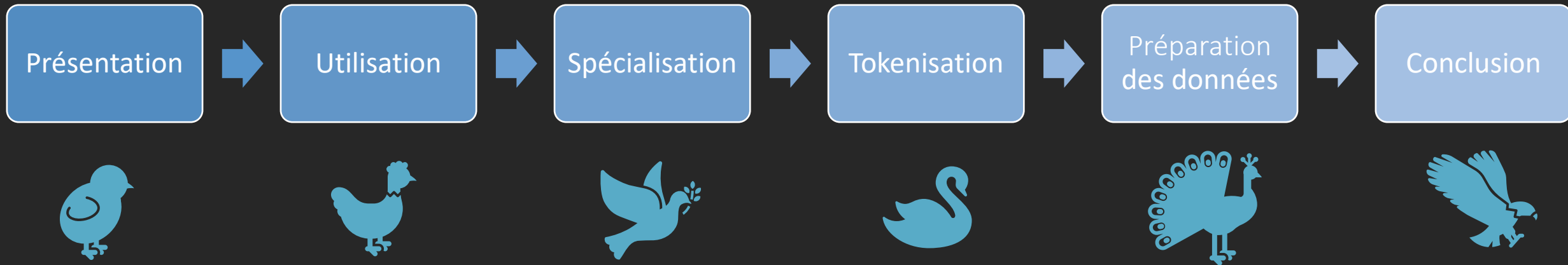
Cécile HANNOTTE

- Diplôme d'ingénieur en Mathématiques et Informatique à l'INSA de Rouen
- En parallèle Master en Science des données obtenu
- 3 ans d'expérience dans le milieu professionnel
- Contact : c.hannotte@groupeonepoint.com

Au programme

Objectif : Utiliser un algorithme d'intelligence artificielle (Réseau de neurones) pour générer du texte à la façon George Sand.

Comprendre la construction du modèle de RN de l'entraînement jusqu'à son utilisation.



Qui est George Sand ?

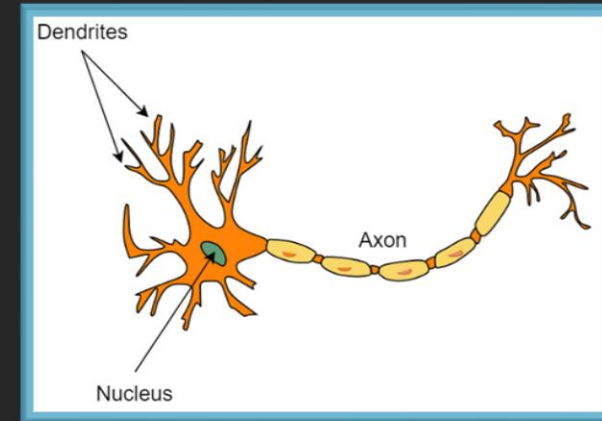
- Amantine Aurore Lucile Dupin de Francueil
- Écrivaine, journaliste, militante
- Passionnée par les sciences naturelles et avant-gardiste sur les questions écologiques



Qu'est-ce qu'un **réseau de neurones** ?

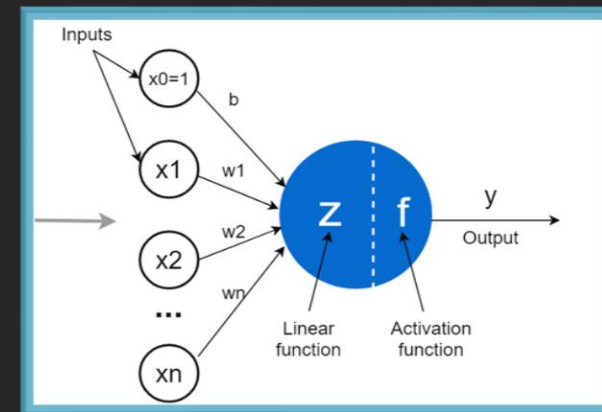
- **En biologie ?**

- RN : Ensembles de neurones interconnectés
- But neurone : Capte un stimuli d'entrée et le transmet de neurones en neurones, jusqu'au cerveau.
- Si stimuli > seuil alors réponse du cerveau (douleur, rougeur)



- **En informatique ?**

- RN informatique = Modélisation simpliste du neurone biologique
- Stimuli d'entrée = Données en entrée
- Seuil = Fonction d'activation
- Réponse = La sortie du réseau



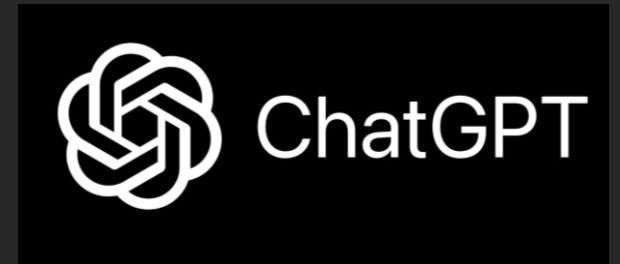
Exemples d'applications utilisant des réseaux de neurones



Image



Son



Texte

Qu'est-ce qu'un réseau de neurones **générateur** ?

Un RN dont le but est de compléter ou générer la suite d'une séquence (texte, chiffres, matrice de chiffres...) de la façon la plus proche de ses données d'entraînement

The top management of the firm are having breakfast with important clients tomorrow, they won't be _____ before 11 a.m..

- (A) joining*
- (B) available*
- (C) availed*
- (D) joins*

Partant pour l'aventure ?

Votez pour l'image qui représente le mieux votre état d'esprit



Essayons-en un !

Installation du projet et appel à l'API

Les fondations pour un projet solide

Mettre en place des outils de qualité de code

- Utiliser un IDE : éviter les Notebooks
- Documenter : fonctions, processus de traitement des données, utilisation du code, etc.
- Respect des standards de Python : [Pylint](#)
- Ecrire des tests : unitaires, de non-régression pour les codes de haut niveau

Mise en place de l'atelier 1/3

(Détail des instructions sur le README.md)

1. Récupérer le code sur Github : <https://github.com/channotte/text-gen>
2. Cloner le projet et ouvrir le dossier récupéré. Partager vos astuces et ressources sur <https://mensuel.framapad.org/p/5zjchxuzss-9xx2?lang=fr>

Mise en place de l'atelier 2/3

Présentation d'Hugging Face



Hugging Face

<https://huggingface.co/>

Catalogue de réseaux de neurones entraînés (appelés modèles).

RN Entraîné = Donne une réponse à une tâche spécifique selon ses données passées d'entraînement

Ex : On obtient un RN entraîné pour dire si une image donnée est un chat ou un chien

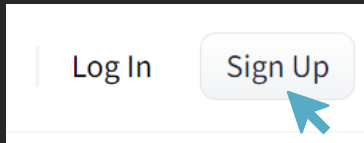
Nous allons utiliser le modèle GPT-2 entraîné sur des données textuelles en français et des œuvres de George Sand

Mise en place de l'atelier 2/3

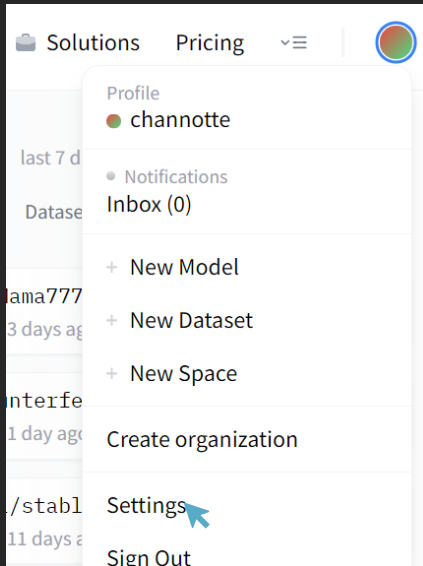
Utiliser Hugging Face – Hors Jupyter notebook

1. Se créer un compte Hugging Face (<https://huggingface.co/>) et générer un token d'accès (voir 1-4)
2. Renseigner le token et son pseudo dans le fichier aurore/credentials.ini (5)

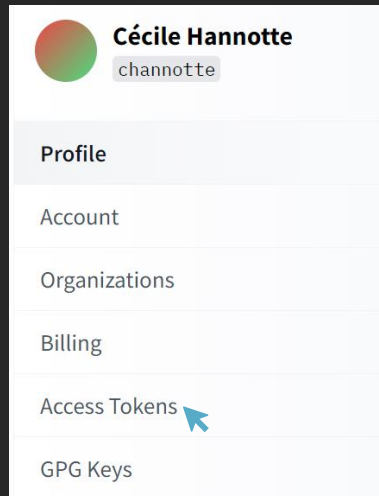
1



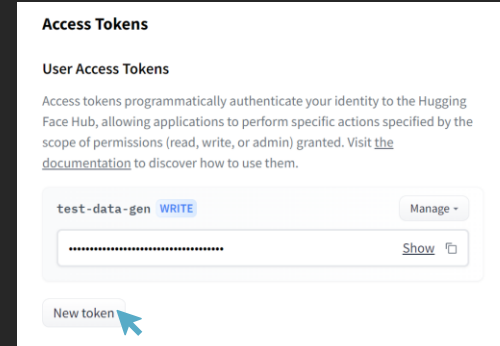
2



3



4



5

aurore/credentials.ini

```
[hugging_face]  
TOKEN=XXXX  
HUGGING_FACE_PSEUDO=XXXX
```


Mise en place de l'atelier 3/3

Lancer le code

- Ouvrir docker et construire le container docker en lançant dans une console:

docker build -t text-gen .

- Lancer le docker pour vérifier que tout se passe bien

Commande Linux : **docker run -v \$PWD/aurore:/aurore text-gen python aurore/1_prepare_dataset_solution.py**

Commande Windows : **docker run -v <chemin complet dossier text-gen>/aurore:/aurore text-gen python aurore/1_prepare_dataset_solution.py**

Nous allons utiliser un **volume (-v)** pour pouvoir modifier directement le code et les données contenues dans le docker, sans avoir à le reconstruire à chaque fois.

Utilisation

Utilisation d'un RN et chargement d'un réseau

Appel RN via API Flask



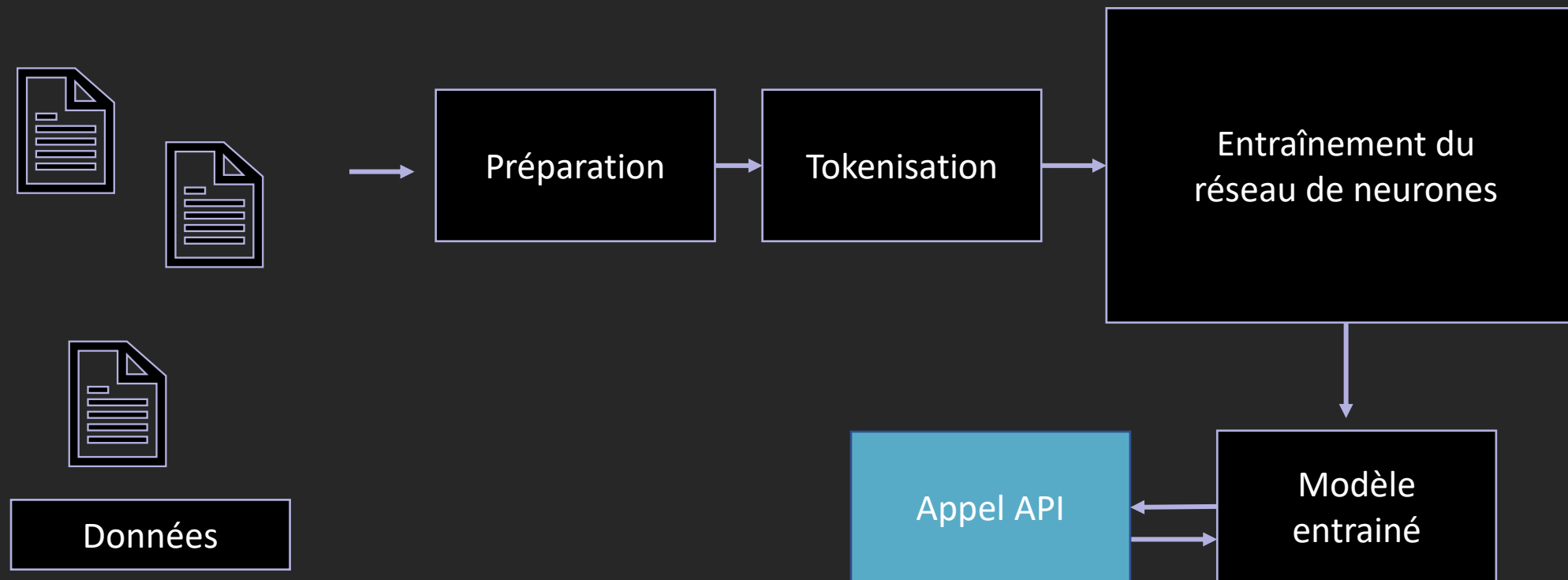
Flask permet de développer rapidement une **application web**

1. Lancez le fichier 5_flask.py :

- **Commande Linux** : `docker run -v $PWD/aurore:/aurore aurore python aurore/5_flask.py`
- **Commande Windows** : `docker run -p 127.0.0.1:80:80 -v $PWD/aurore:/aurore aurore python aurore/5_flask.py`
- Se rendre sur le lien généré : <http://172.17.0.3/generate> et rentrez du texte
- Appuyez sur "générer" et le réseau effectuera sa prédiction (attendre un peu) et l'affichera
- Observer dans le code comment nous avons appelé le réseau de neurones et quel réseau nous avons utilisé

[Présentation rapide](#)

Les étapes d'un RN : Appel à un réseau pré-entraîné (via API ou code directement)

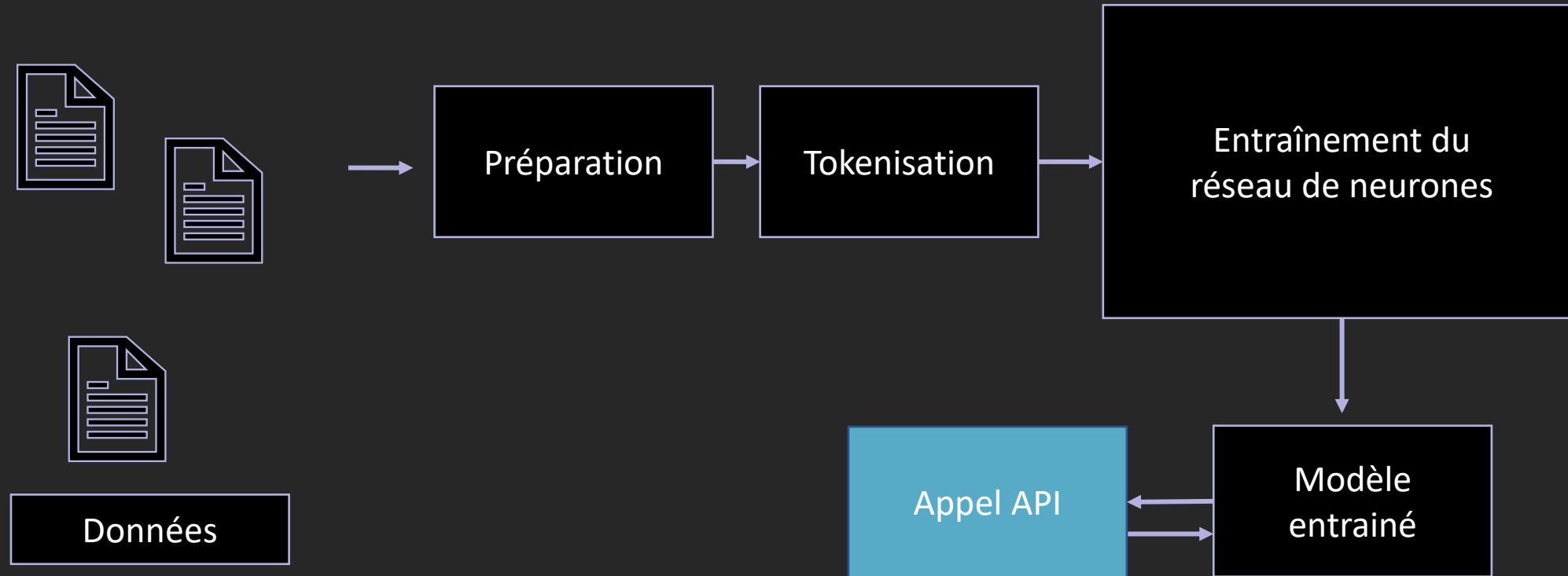


Vous êtes ici :

Objectif : Requêter un modèle pré- entraîné

1. Aller dans le fichier `4_generate.py`
2. Modifier les variables config et model afin d'appeler le modèle comme dans l'API Flask
3. S'inspirer de la fonction `get_text()` et créer un prompt de deux phrases :
 1. Créer une liste *prompts* comprenant deux débuts de prompt de votre choix (ex : "Cette femme était assise", "Le soir elle se rappelait"...)
 2. Pipeline de Hugging face permet de créer des prompt pour appeler le réseau. Utiliser la variable pipe qui prend en entrée prompts pour ressortir les deux textes (output0 pour le premier, output1 pour le deuxième)
 3. Afficher les deux output (en print tout simplement)

Comment avons-nous obtenu un modèle (informations sur le reseau entrainé) ?

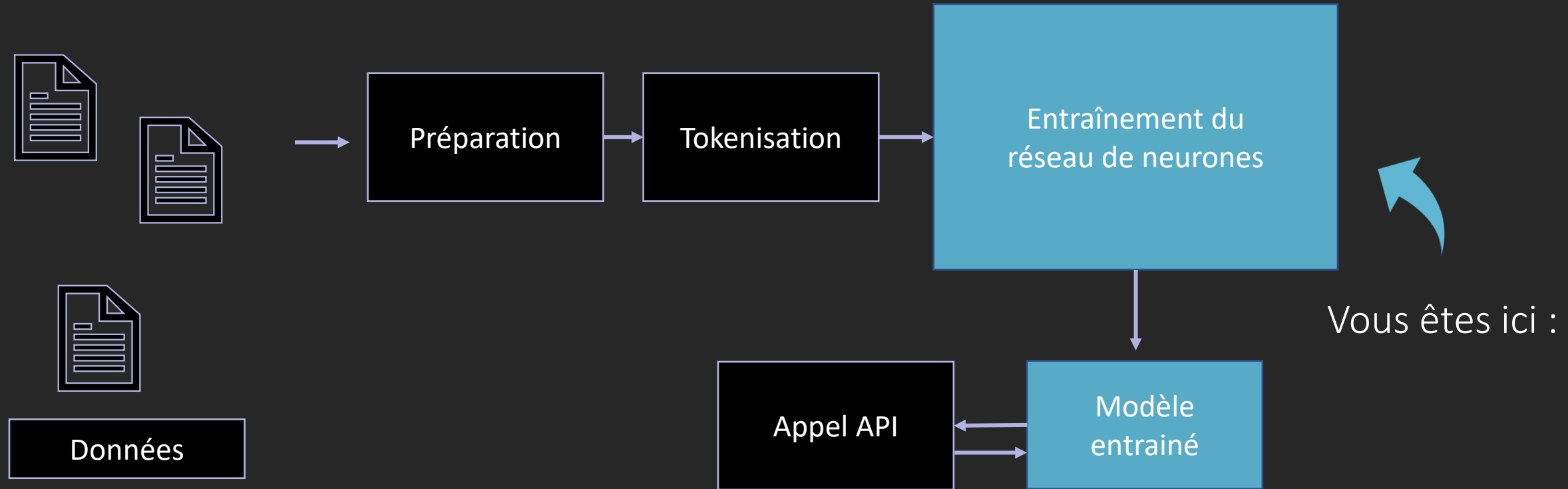


Vous êtes ici :



Entraînement et Spécialisation

*Sélectionner le bon candidat
pour résoudre notre problématique*

Comment avons-nous obtenu un modèle (informations sur le réseau entraîné) ?



Comment choisir un réseau de neurones

- Veille scientifique et technique sur la thématique cible (ici la génération de texte)
- S'assurer que vos données sont proches des données utilisées. (texte d'appel d'API en français, plutôt littéraire, car entraîné en français avec des œuvres de George Sand)
- Est-ce que du code est disponible ? Si oui est-il de qualité ? 
- Est-ce qu'un modèle pré-entraîné est disponible ? 

Les catalogues de modèles



Hugging Face

<https://huggingface.co/>

Model Zoo

<https://modelzoo.co/>

Mettent à disposition des réseaux de neurones déjà entraînés, classés par thématiques et par frameworks.

GPT-2 : le choix pour ce codelab

- Conçu par OpenAI (ChatGPT, c'est eux aussi !)
- GPT : Generative Pre-trained Transformer
- Principe : entraîner un réseau sur un corpus très large (des centaines de milliards de données) avec une tâche où la vérité terrain peut être générée automatiquement
- Ce réseau peut ensuite être adapté sur des tâches plus spécifique (spécialisation) comme les œuvres de George Sand. Pour adapter un RN déjà entraîné, on nécessite de beaucoup moins de données qu'à son entraînement initial
- **A vous de jouer !** Sur Hugging Face, recherchez « gpt2 » pour trouver les différents modèles GPT2. Quel modèle est le plus adapté ?

Objectif : Découvrir les différents modèles utilisables

1. Se placer sur le fichier `3_train.py`
2. Afficher le nombre de paramètres et de couches `model.summary()` d'un modèle de type gpt2
3. Modifiez `MODEL_NAME` par un différents modèles en français (voir différents modèles sur Hugging Face***). Comparer les summary

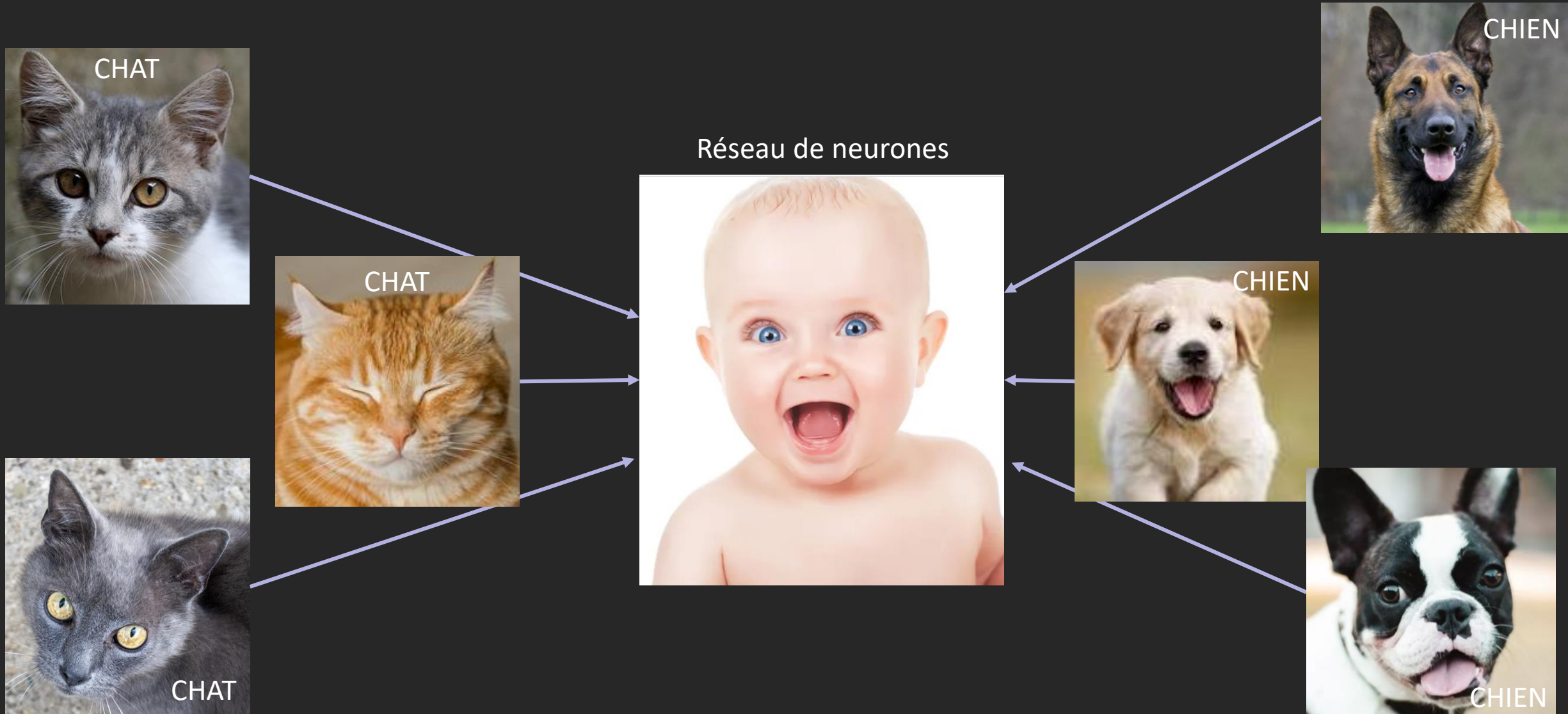
Ces params vont permettre de récupérer une version spécifique du réseau de neurones GPT-2

*** Le modèle utilisé est enregistré au format PyTorch, d'où l'argument `from_pt = True` dans `model.from_pretrained(MODEL_NAME, from_pt=True)`

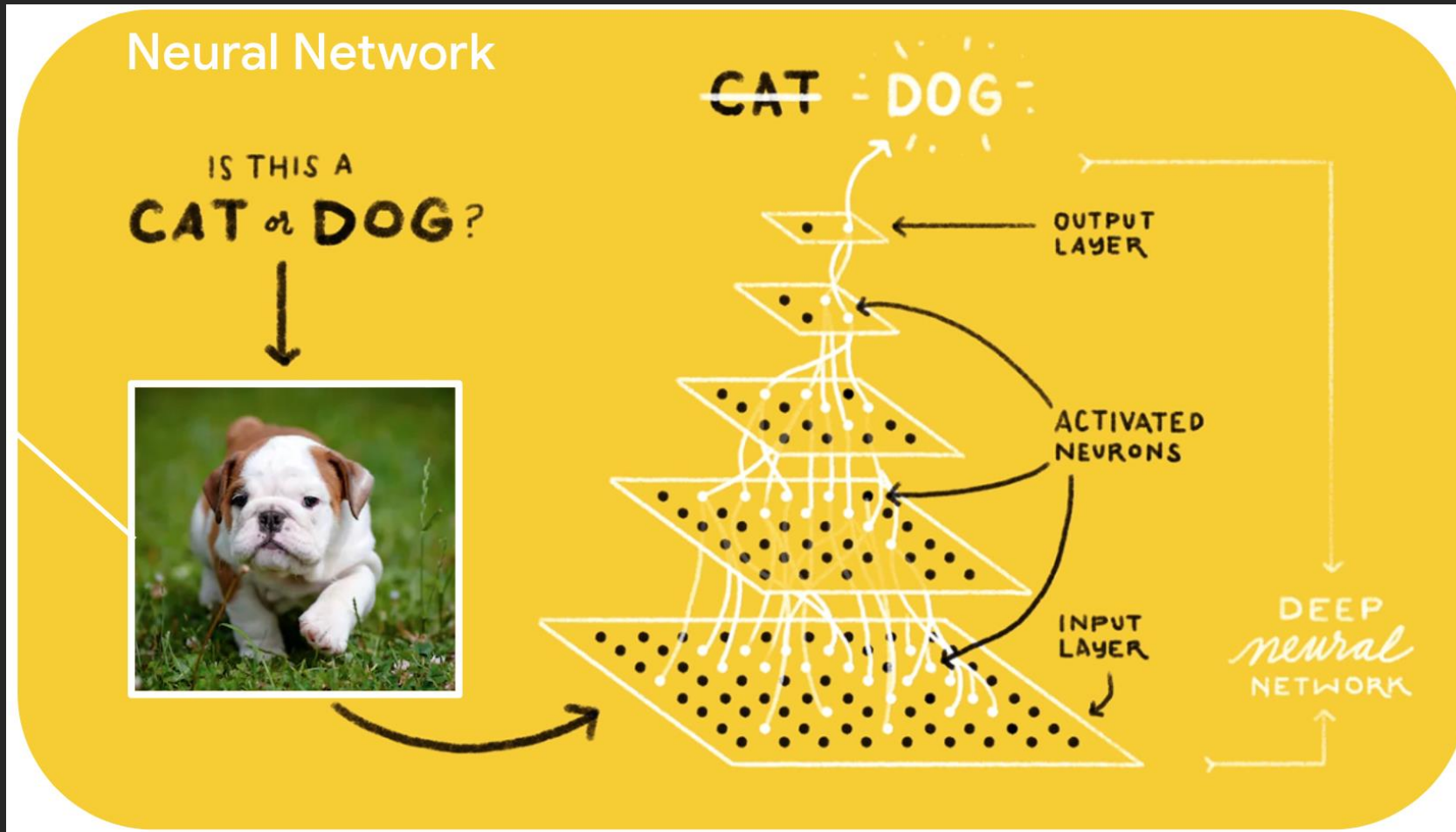
Si une erreur est levée car le modèle est du tensorflow, remplacer cet argument par `from_tf=True`

Comment un réseau s'entraîne ?

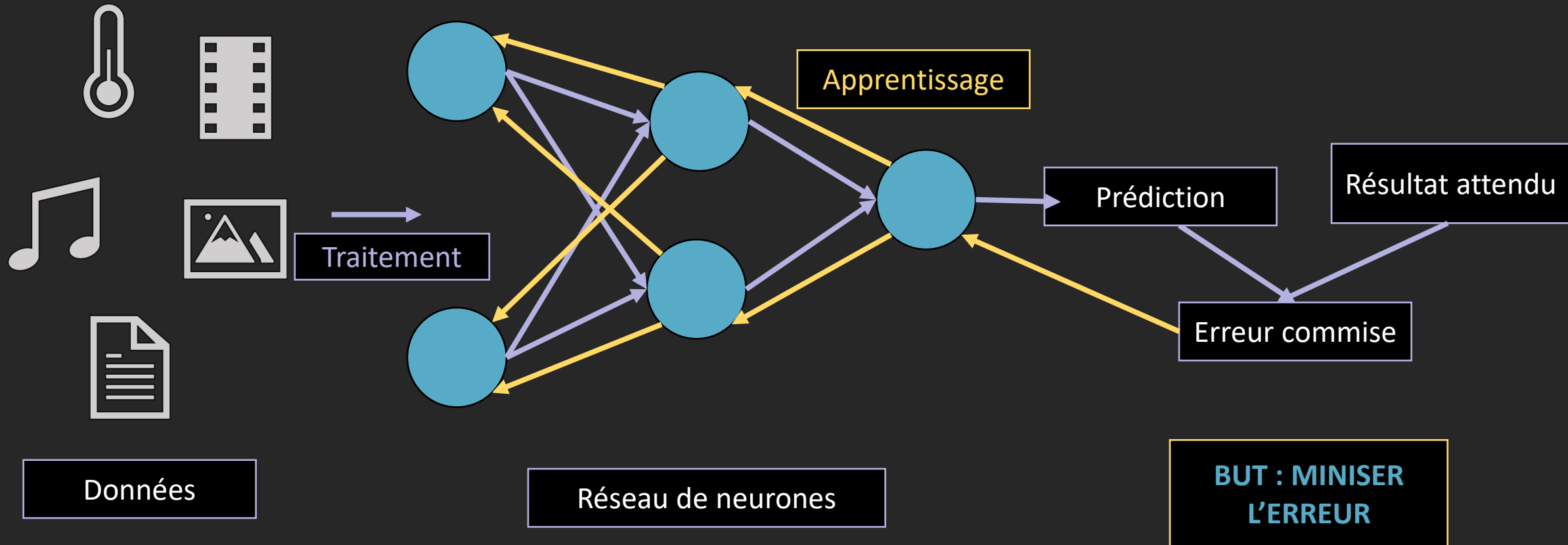
Mise en pratique d'un RN : **Chien ou chat ?**



Mise en pratique d'un RN : Chien ou chat ?



Comment un réseau s'entraîne ?

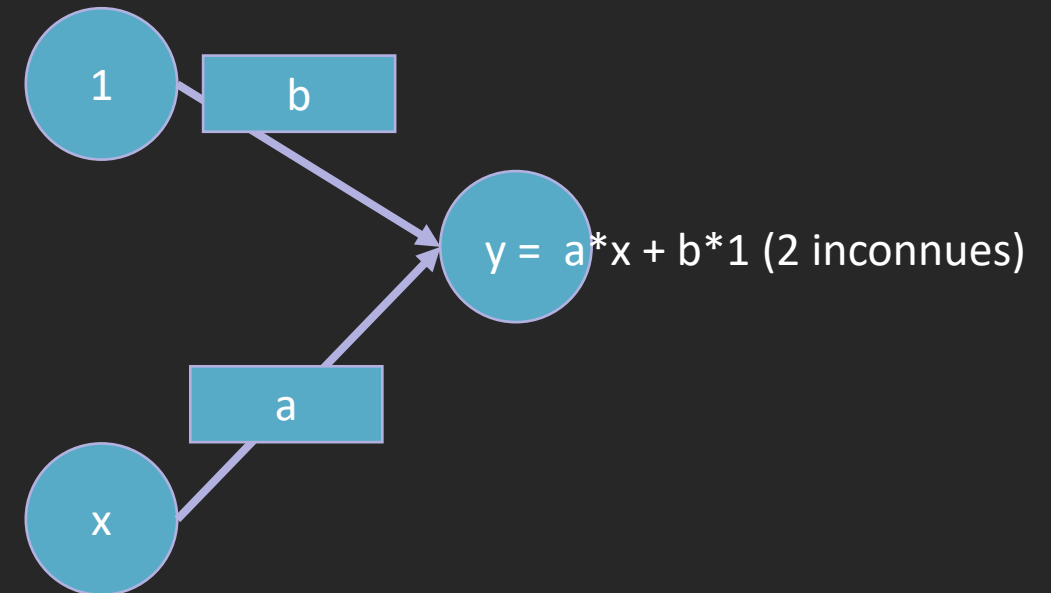
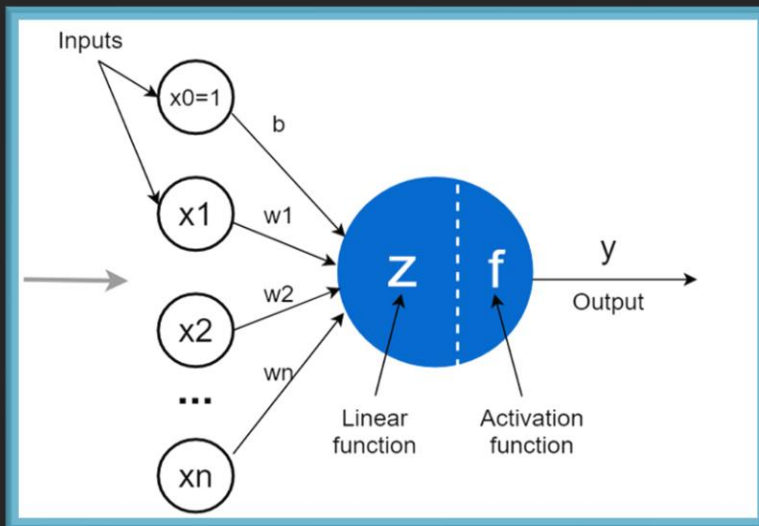


Objectif : Récupérer les données d'entraînement

1. Se placer sur le fichier `3_train.py`
2. Vérifier que vous avez bien les données dans `aurore/data`, dans le cas contraire lancez le script `1_prepare_dataset_solution.py`
3. Charger les données dans la variable `dataset` via la fonction `load_from_disk(<chemin>)` de `datasets`

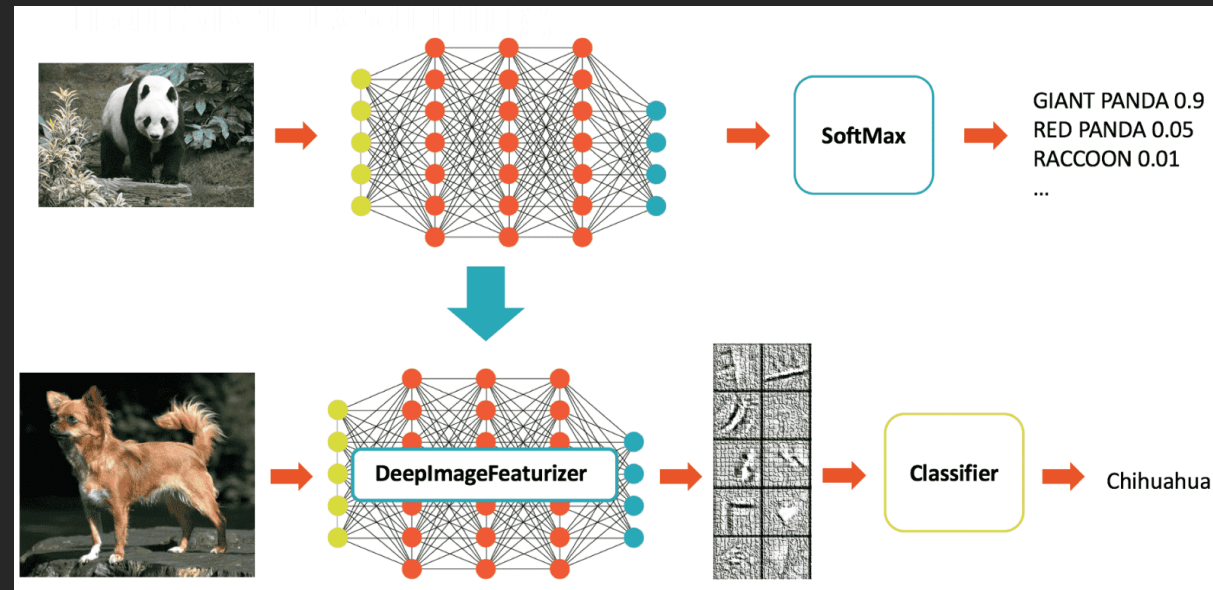
Entraîner un réseau de neurones

- Chaque neurone calcule la **somme pondérée** des entrées qu'il reçoit
- **Entraîner** un réseau de neurones revient à trouver les **bonnes valeurs** pour les **pondérations** (les poids w ici)
- Chaque pondération à trouver est un paramètre inconnu
- GPT – 1,5 à 2 Milliards d'inconnues à retrouver par entraînement



La Spécialisation (transfert learning)

- Spécialiser un modèle consiste à partir d'un réseau de neurones déjà entraîné et à [poursuivre cet apprentissage](#) sur de nouvelles données
- Repose sur le principe qu'il est plus facile d'apprendre à jouer du violon si vous maîtrisez déjà le piano et la guitare que si c'est votre premier instrument
- Entraîner RN récent de 0 : Cout important (énergie et données). En général = les entreprises se contentent de les spécialiser



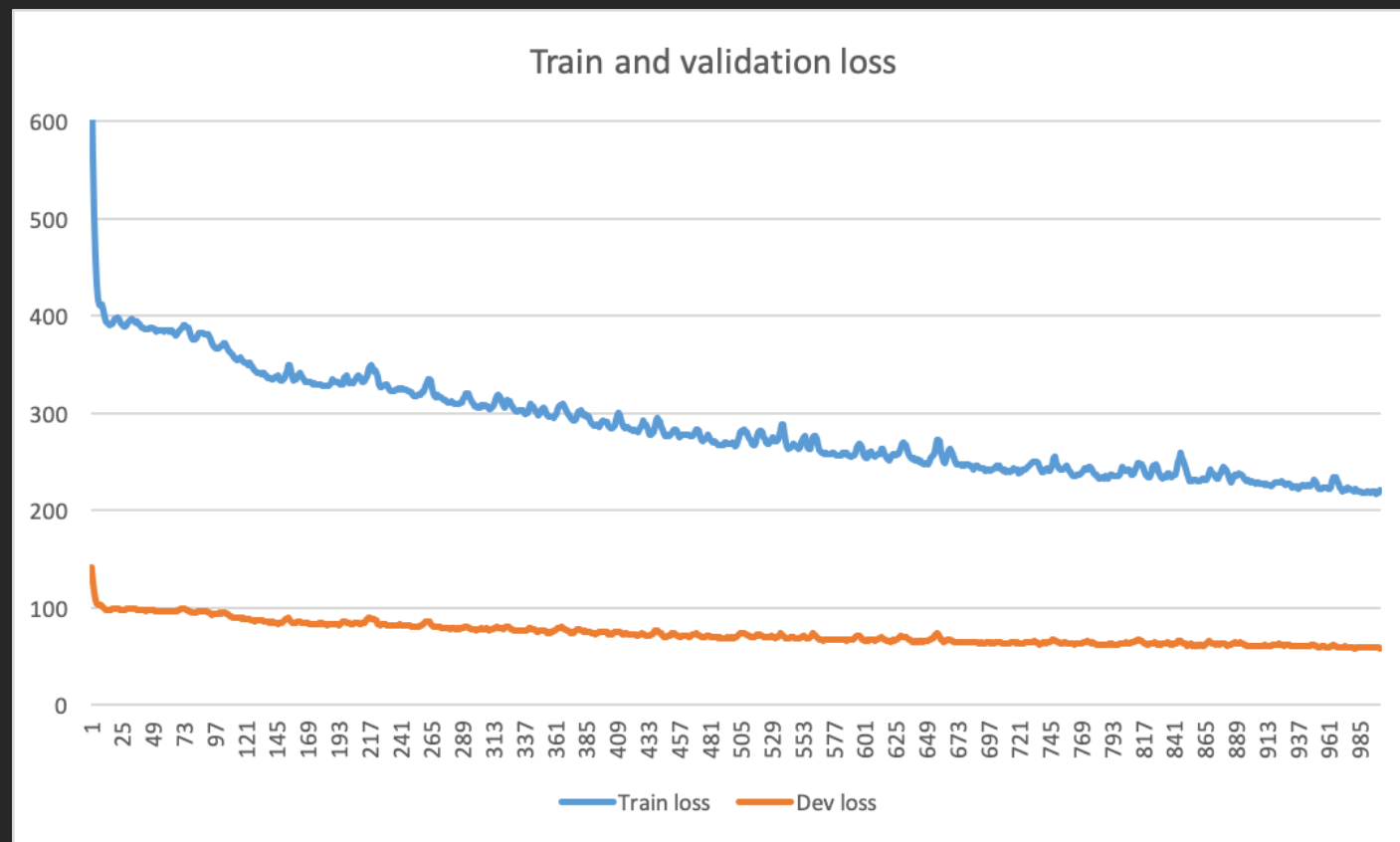
Objectif : Entraîner le réseau

1. Placez-vous sur le fichier `3_train.py`
2. Appeler `fit()` sur le modèle
 1. Fit prendra en entrées le `tf_train_dataset`, `tf_eval_dataset`, et le nombre d'époques
3. Une fois la fonction complétée, lancez le script
4. Changer nombre d'époques (max 3) et entraîner
5. Changer le batch size et entraîner

Pour des raisons de temps de calcul, nous ne pourrons entraîner que durant une seule époque.

Apprentissage par lot

- Un réseau de neurone apprend en **itérant plusieurs fois sur les données** et le résultat attendu
- Lorsque le réseau de neurones a parcouru l'ensemble des données on dit qu'il a accompli une **époque**

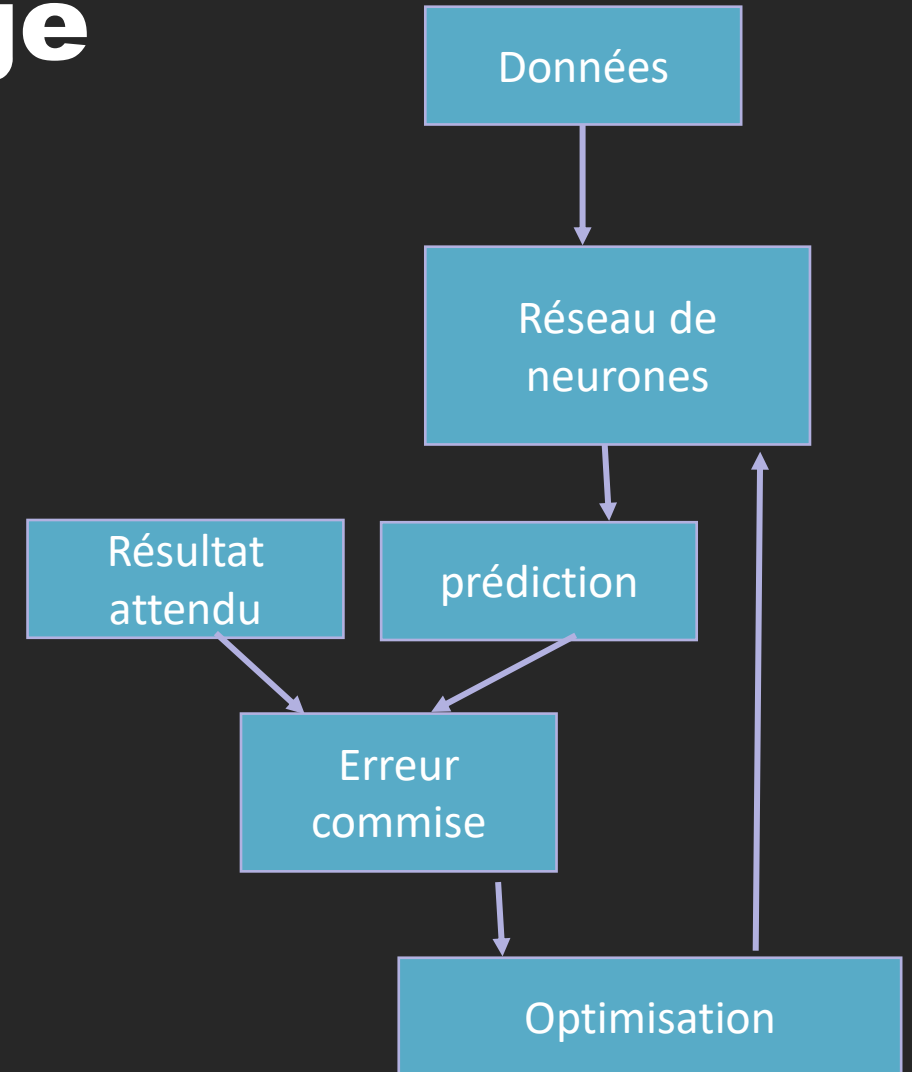


Source : [Why my network needs so many epochs to learn?](#)

Stratégie d'apprentissage

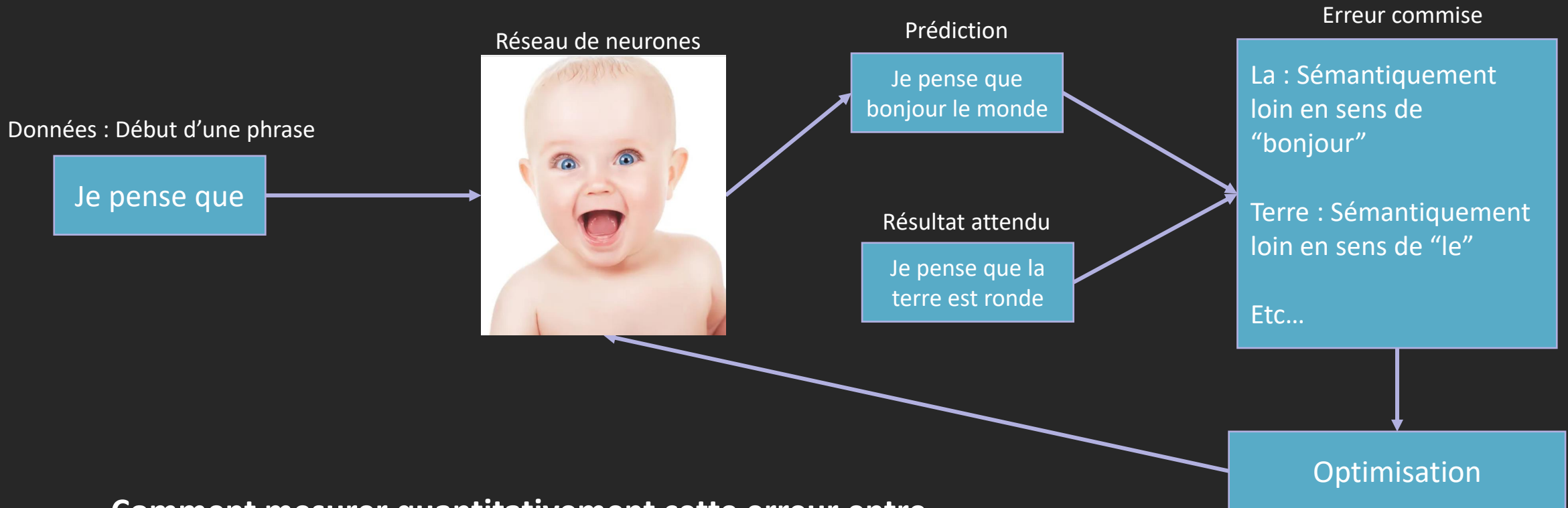
Pour apprendre vite et bien je dois déterminer quand est-ce que je calcule l'erreur commise et que je met à jour mon réseau

- **Stratégie 1** : à chaque donnée => entraîne beaucoup d'instabilité, risque de partir dans un minima local
- **Stratégie 2** : à chaque époque => lent
- **Stratégie 3** : à chaque lot (batch) de N données



Stratégie d'apprentissage

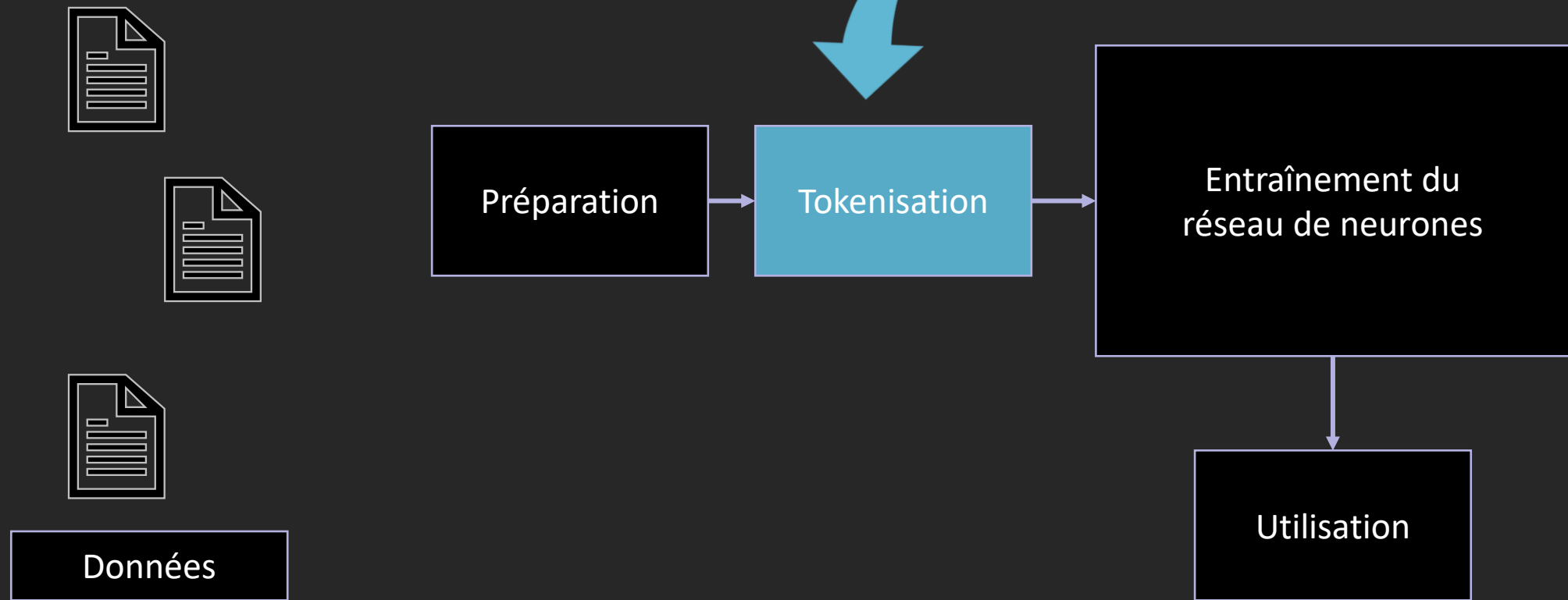
Dans notre cas de la generation de texte



Tokenisation

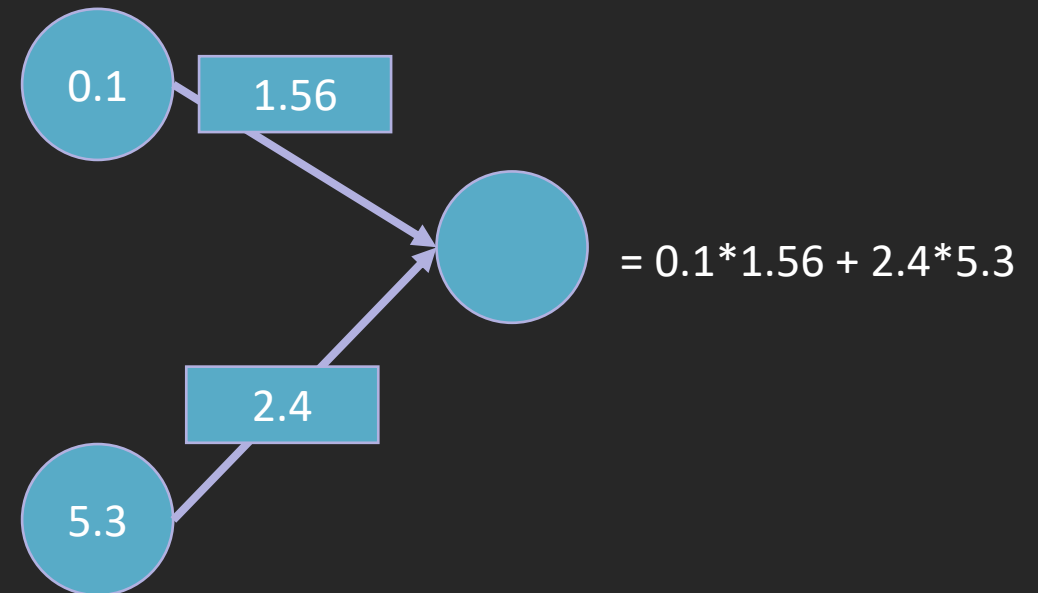
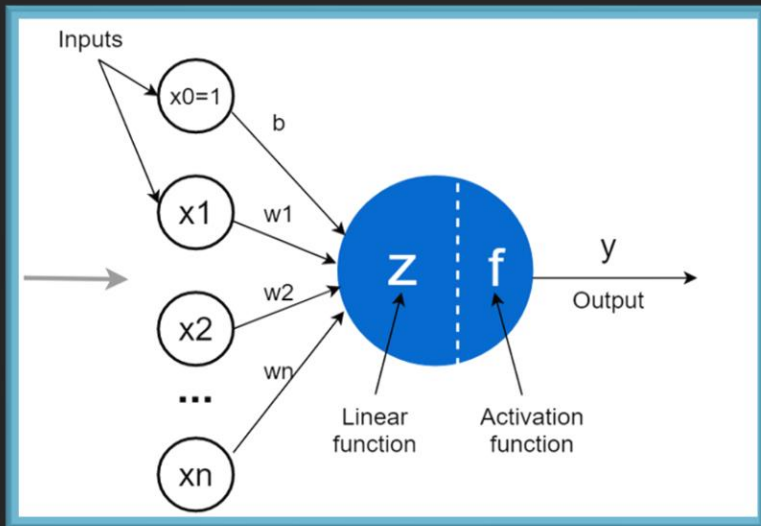
Toutes les données sont complexes.
Certaines le sont plus que d'autres

Vous êtes ici



Entraîner un réseau de neurones

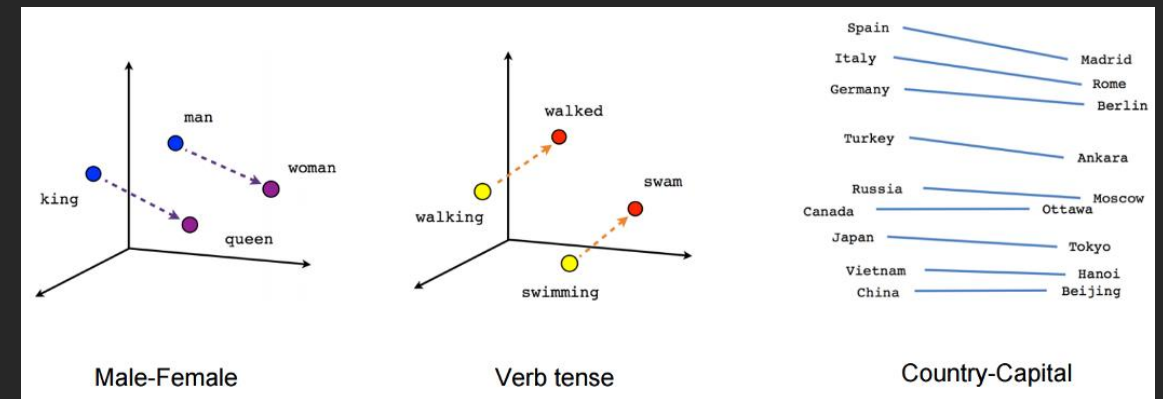
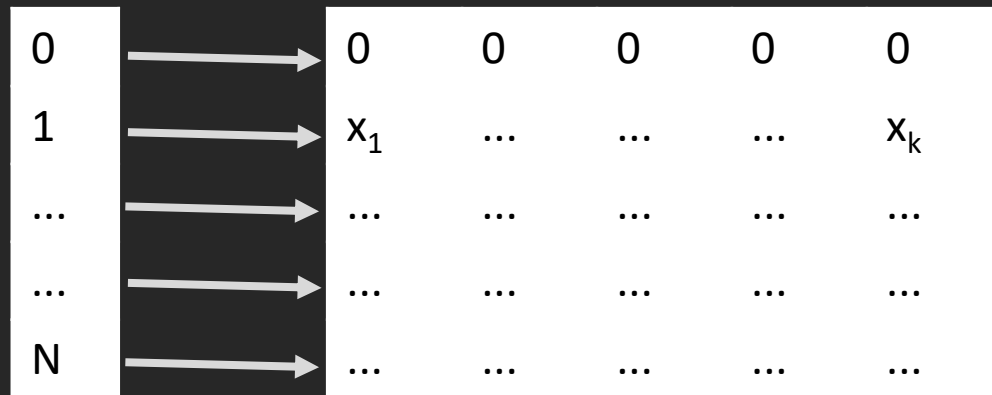
- Entraîner un réseau de neurones revient à trouver les bonnes valeurs pour les pondérations
- Nos données doivent donc être sous forme de nombres



Couche d'embedding dans les réseaux de neurones

Tokenisation : Transforme des données textuelles en nombres

Embedding : remplacer chaque mot du lexique => un vecteur numérique. Transforme des entiers positifs en vecteurs réels

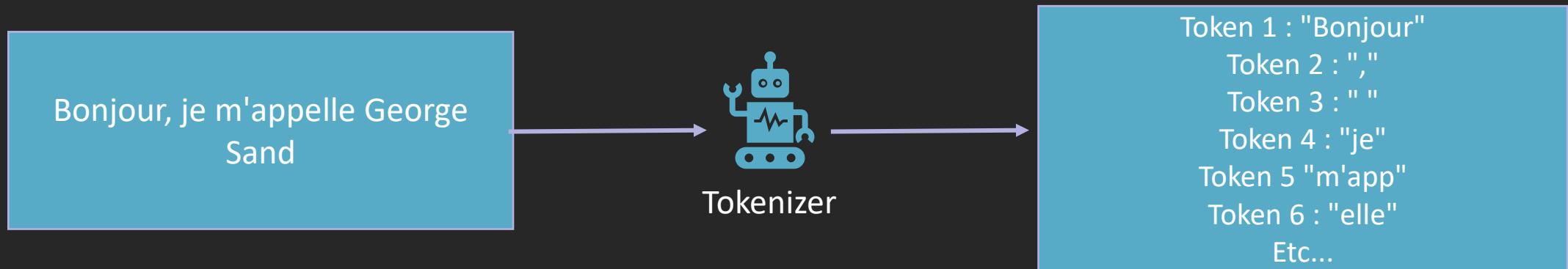


Objectif : Récupérer un tokenizer

1. Se placer dans le fichier `2_tokenize.py`
2. (ligne 33) Instancier le tokenizer pré-entraîné **benjamin/gpt2-wechsel-french** grâce à la méthode `from_pretrained` de la classe `AutoTokenizer`

Convertir du texte en nombre

- Pour les réseaux de neurones la première étape consiste à découper le texte en tokens et à associer un identifiant numérique à chaque token
- Un token peut être : une lettre, une syllabe, un groupe de lettre (un peu moins ou un peu plus qu'une syllabe), un mot, un groupe de mot



Objectif : Tester le tokenizer

1. Se placer sur le fichier [2_tokenize.py](#)
2. (ligne 44) Créer une variable contenant un texte simple, par exemple "Bonjour Madame, je m'appelle George Sand. Et vous ?"
3. (ligne 52) Analyser ce texte avec le tokenizer instancié
Tokens_analysis est un dictionnaire et la valeur associée à la clé **input_ids** donne le découpage en tokens
 1. (ligne 65) Utiliser la fonction [convert ids to tokens](#) pour afficher le texte associé à chaque token

Bien choisir ses tokens

- Tokens = comment le RN voit le texte
- Si le token font intervenir des groupes de lettres ou des mots, ils doivent être choisi en fonction du champ lexical du domaine cible (exemple avec CUTIE pour la comptabilité)
- Les tokens forment un lexique fixe : les RN ne savent pas gérer les tokens inconnus
- Pour être rigoureux il faut donc créer les tokens à partir des données d'apprentissage uniquement
- Nous pourrons ensuite vérifier leur pertinence pour les données d'évaluation

Objectif facultatif : Spécialiser le tokenizer

- Utiliser la fonction `get_training_corpus` pour récupérer un itérateur sur le jeu de données d'entraînement
- Utiliser la méthode `train_new_from_iterator` du tokenizer pour spécialiser le tokenizer avec comme paramètre :
 - `text_iterator` : le résultat de la fonction `get_training_corpus`
 - `vocab_size` : 52 000

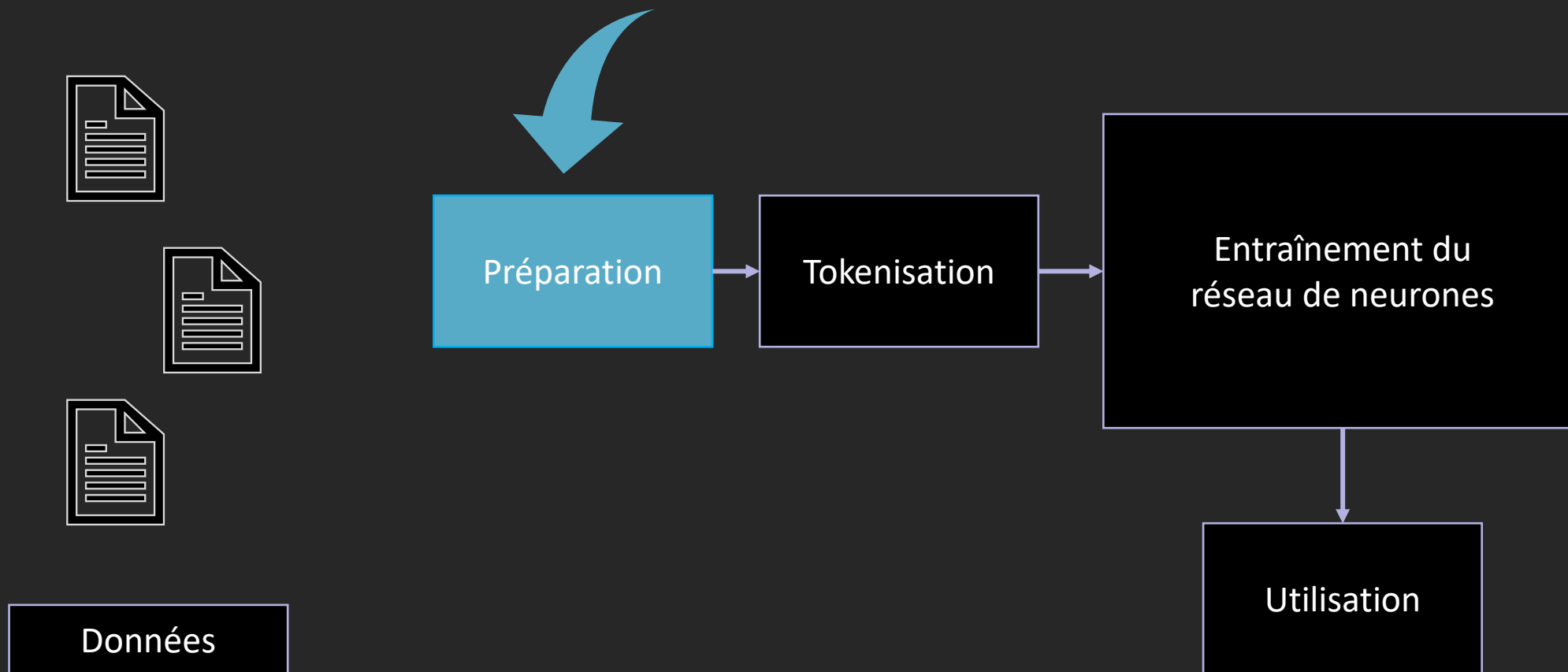
Objectif facultatif : Sauvegarder le tokenizer spécialisé en local

- Utiliser la fonction [save_pretrained](#) pour sauvegarder le tokenizer dans le répertoire aurore/tokenizer
- Observer le résultat

Préparer vos données

*La mare au diable de la data science,
ce sont les données !*

Vous êtes ici



Les fondations pour un projet solide

- Est-ce que je dispose d'assez de données ?
 - Pour une même problématique différentes solutions sont possibles
 - Pour chaque solution le nombre minimal de données nécessaire doit être évalué
- Est-ce que je dispose de la vérité terrain ?
 - Vérité terrain : la solution attendue
 - Les réseaux de neurones sont des techniques par apprentissage supervisé (supervised machine learning)
 - Sans la vérité terrain le réseau ne peut pas apprendre et le projet est bloqué

Les fondations pour un projet solide

Exercice : Placez vous dans le fichier `1_solution.py`

- Recherchez sur internet combien de données ont été nécessaires pour entrainer ce type de réseau ? (GPT-2)
- Dans le code, combien de données utilisons-nous pour le spécialiser ?

Est-ce que je dispose d'assez de données ?

- Données nécessaires pour GPT-2 : contenu d'environ 8 livres pour l'apprentissage et 1 pour le test
- Les livres de George Sand appartiennent au domaine public : projet Gutenberg.

Récupérer les données

Le côté obscur de la force : utiliser une méthode de chargement de données trop générique

```
from dataset import load_dataset
lelia_url = "https://www.gutenberg.org/files/39738/39738-0.txt"
la_petite_fadette_url = "https://www.gutenberg.org/cache/epub/34204/pg34204.txt"
gabriel_url = "https://www.gutenberg.org/cache/epub/13380/pg13380.txt"
lettre_voyageur_url = "https://www.gutenberg.org/files/37989/37989-0.txt"
la_mare_au_diable_url = "https://www.gutenberg.org/files/23582/23582-0.txt"

train_paths = [lelia_url, la_petite_fadette_url, gabriel_url, lettre_voyageur_url]
test_path = la_mare_au_diable_url

dataset = load_dataset("text", data_files={"train": train_paths, "test": test_path})
```

Inclus les 50 premières lignes qui n'ont rien à voir avec le texte de George Sand mais sont des ajouts du projet Gutenberg

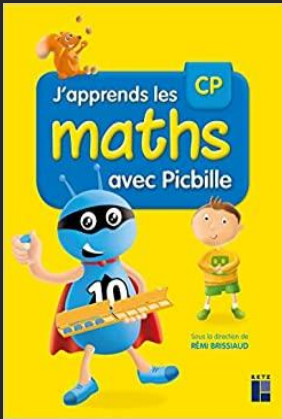
Exercice 1 : récupérer les données

Nous allons compléter le code pour préparer les données

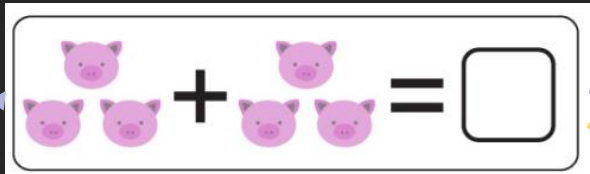
- Méthode **download_file** : pour chaque fichier
 1. Utilisez la fonction get_file de la bibliothèque Keras pour récupérer chacun des fichiers
 2. Ouvrir chaque fichier et récupérer la liste des lignes contenues dans ce fichier avec la méthode `readlines()`
 3. Retirer les 50 premières lignes
 4. Retirer les 100 dernières lignes

Les données d'apprentissage, d'évaluation et de test

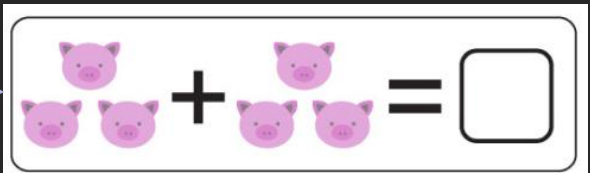
Objectif : Contrôle de maths. Quelle est la meilleure manière d'apprendre ?



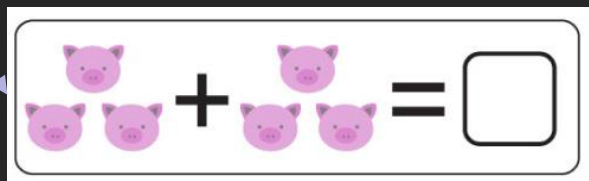
100 exercices de maths



80 exercices de maths
+ corrigés donnés (Vérité terrain) au RN
Données d'apprentissage



10 exercices de maths
+ corrigés non donnés au RN
Données de validation



EXAMEN : 10 exercices de maths
Données de test

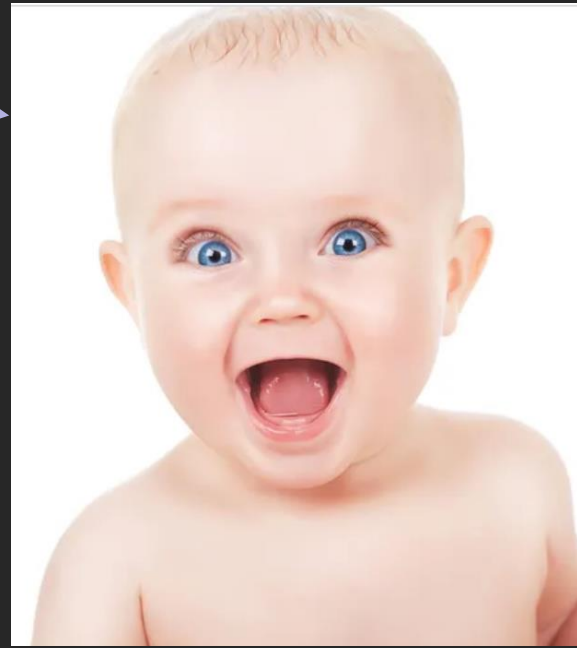
$3+3 = ?$

$3+3 = 3$

Apprentissage
 $3+3 = 6 !$

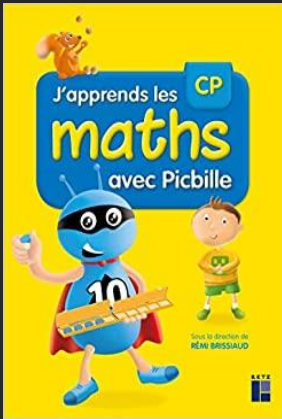
Ok !

Réseau de neurones

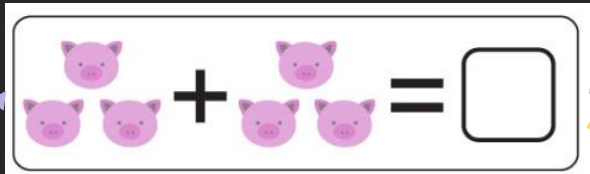


Les données d'apprentissage, d'évaluation et de test

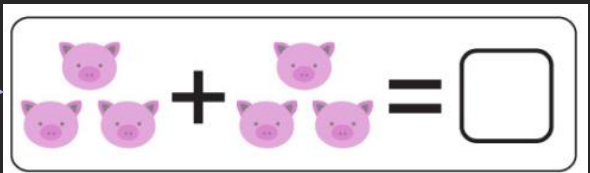
Objectif : Contrôle de maths. Quelle est la meilleure manière d'apprendre ? Modifier la Batch size (=3)



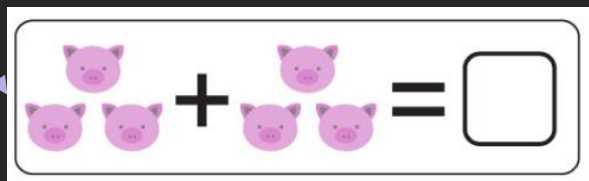
100 exercices de maths



80 exercices de maths
+ corrigés donnés au RN
Données d'apprentissage



10 exercices de maths
+ corrigés non donnés au RN
Données de validation



EXAMEN : 10 exercices de maths
Données de test

- a) $3+3 = ?$
- b) $3+2 = ?$
- c) $4+3 = ?$

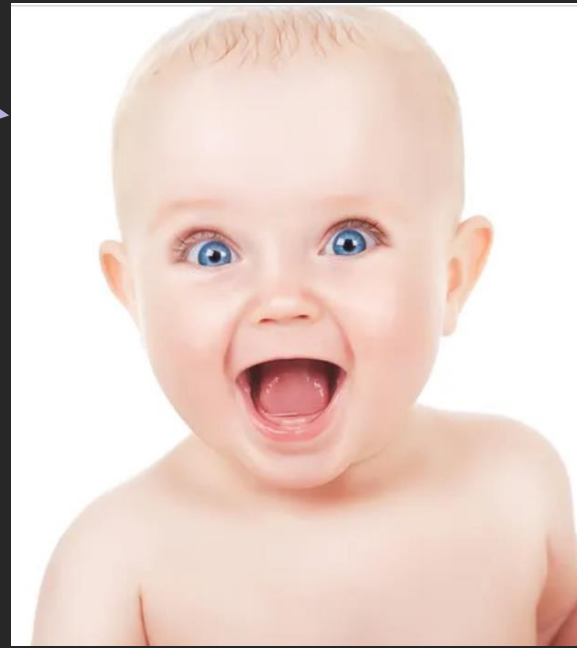
- a) 5
- b) 6
- c) 3

Apprentissage

- a) $3+3 = 6 !$
- b) $3+2 = 5 !$
- c) $4+3 = 7 !$

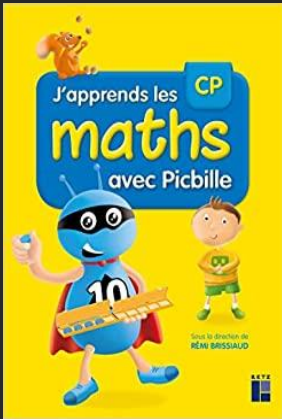
Ok !
(réajustement
des
connaissances)

Réseau de neurones



Les données d'apprentissage, d'évaluation et de test

Objectif : Contrôle de maths. Quelle est la meilleure manière d'apprendre ? Attention à la diversité des données



100 exercices de maths

80 exercices de maths
+ corrigés donnés au RN
Données d'apprentissage

10 exercices de maths
+ corrigés non donnés au RN
Données de validation

EXAMEN : 10 exercices de maths
Données de test

- a) $3+3 = ?$
- b) $4+2 = ?$
- c) $1+5 = ?$

- a) 5
- b) 6
- c) 3

Apprentissage

- a) $3+3 = 6 !$
- b) $4+2 = 6 !$
- c) $1+5 = 6 !$

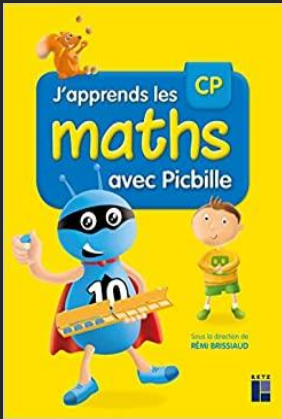
Ok ! Donc en fait c'est toujours 6 non ? Facile !

Réseau de neurones



Les données d'apprentissage, d'évaluation et de test

Objectif : Contrôle de maths. Quelle est la meilleure manière d'apprendre ? Après une 1 époque (epoch)



100 exercices de maths

80 exercices de maths
+ corrigés donnés au RN
Données d'apprentissage



10 exercices de maths
+ corrigés non donnés au RN
Données de validation

3+3 = ?

3+3 = 5

Vérité : 6

Prédiction : 5

Erreur = 6 - 5 = 1

Conclusion : Y'a encore du boulot !!!

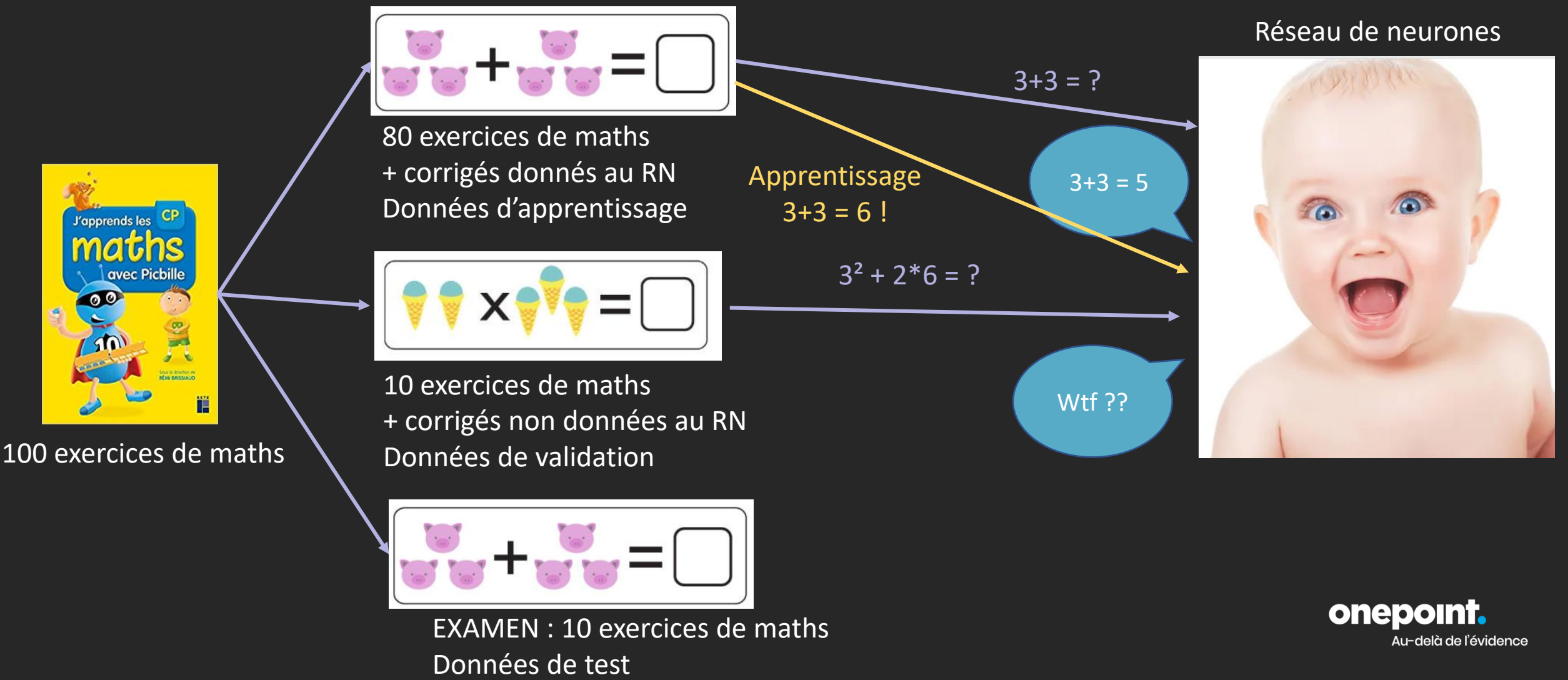
On recommence les 80 exos

EXAMEN : 10 exercices de maths
Données de test



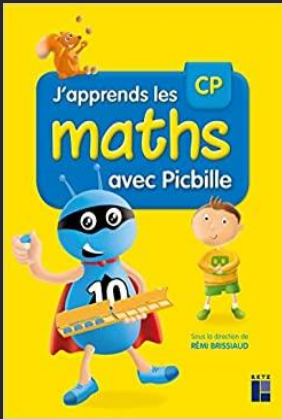
Les données d'apprentissage, d'évaluation et de test

Objectif : Contrôle de maths. Quelle est la meilleure manière d'apprendre ? Attention à l'hétérogénéité des données



Les données d'apprentissage, d'évaluation et de test

Objectif : Contrôle de maths. Quelle est la meilleure manière d'apprendre ? Attention à l'hétérogénéité des données



100 exercices de maths

80 exercices de maths
+ corrigés donnés au RN
Données d'apprentissage



10 exercices de maths
+ corrigés non donnés au RN
Données de validation

Il a l'air prêt, on arrête tout !



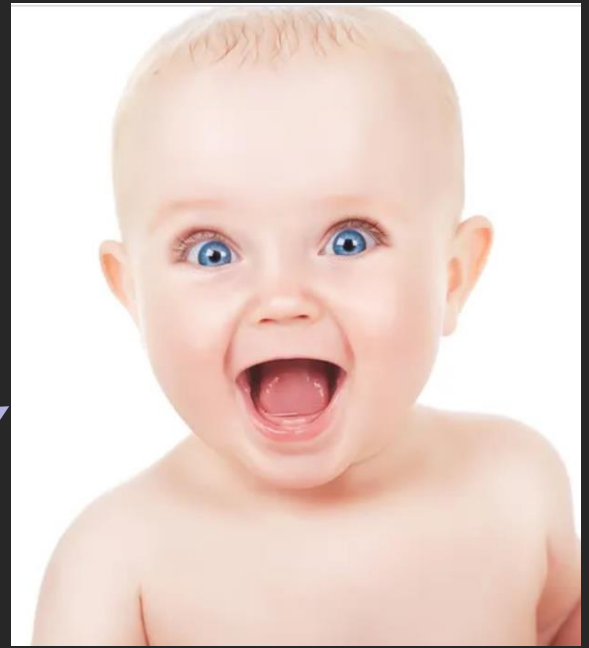
3 + 3 = ?

6

EXAMEN : 10 exercices de maths
Données de test

Note finale :
9/10

Réseau de neurones



Les fondations pour un projet solide



Mettre en place une démarche scientifique

- Le réseau est **entraîné** à réaliser une tâche **sur une partie des données**. Pour savoir si l'apprentissage est concluant nous devons **l'évaluer** sur des **données** qu'il n'a **jamais utilisées**
- Fidélité des résultats : si j'utilise des **données** d'apprentissage et d'évaluation **différentes**, est-ce que j'obtiens des **scores de réussite similaires** ?
- La **validité** de notre expérience est **limitée** : savoir générer du George Sand n'implique pas que notre réseau saura générer du Virginia Woolf.

Les fondations pour un projet solide

Exercice 2

- A votre avis, dans notre cas, en quoi consiste la vérité terrain ?

Exercice 3

- Quelle stratégie utiliseriez-vous pour sélectionner les données d'évaluation ?

Les fondations pour un projet solide

Est-ce que je dispose de la vérité terrain ?

- Cas particulier où les données contiennent déjà la vérité terrain

Evaluation

- Nous pourrions sélectionner 1/10 des phrases de chaque livre
- Ou sélectionner un livre particulier

Exercice 2 : Séparer les phrases

Nous allons compléter le code pour préparer les données

- Méthode [split_text_to_list](#) : identifier les phrases présentes dans chaque ligne
 1. Retirer les espaces en fin de ligne avec la fonction Python [rstrip](#)
 2. Recréer une chaîne de caractères avec la méthode [join](#)
 3. Identifier le début et la fin de chaque phrase en utilisant la bibliothèque spécialisée [NLTK](#)

Facultatif : calcul du nombre de mots

Quelques ajouts bien utiles :

- Afficher le nombre de phrases pour les données d'apprentissage et d'évaluation
- Regarder au hasard quelques phrases et vérifier qu'il n'y a pas trop d'erreurs

Conseils : Utiliser un réseau de neurones

- Un RN n'est jamais que l'une des briques d'un projet beaucoup plus complexe
- Il vous faudra **préparer et vérifier les données** fournies par l'utilisateur (le texte est-il en français ? Contient-il des propos injurieux ? Etc.)
- Il vous faudra également **mettre en forme le résultat** produit par le réseau de neurones
- Vous devrez gérer des problématiques d'accès, de mises à jour, etc.
- Permettre une interface de qualité pour une expérience utilisateur optimale

Conclusion

Votez pour l'image qui représente le mieux votre état d'esprit.



Un feedback ?

Si vous souhaitez nous donner votre avis sur le lab, scannez ce QR Code !



Ressources

- Notebook qui a inspiré ce lab : [Notebook Hugging Face causal language model](#)
- Article medium sur tous les concepts de GPT 2 (architecture) : [GPT2 : Architecture overview](#)
- Documentation Hugging Face sur les différents modèles GPT2 : https://huggingface.co/docs/transformers/model_doc/gpt2

GPT-2 : bibliographie

- Présentation des modèles de type transformer : [Attention is all you need](#)
- Présentation des modèles de type GPT : [Improving language understanding by generative pre-training](#)
- Polémique à la sortie de GPT-2 : [GPT-2 d'OpenAI : Un meilleur outil de traitement automatique du langage et les questions éthiques qu'il soulève](#)
- Article de recherche associé : [Language Models are unsupervised multitask learner](#)
- [Présentation sur Huggingface](#)

Documentation complémentaire

[Hugging Face: Understanding tokenizers](#)

