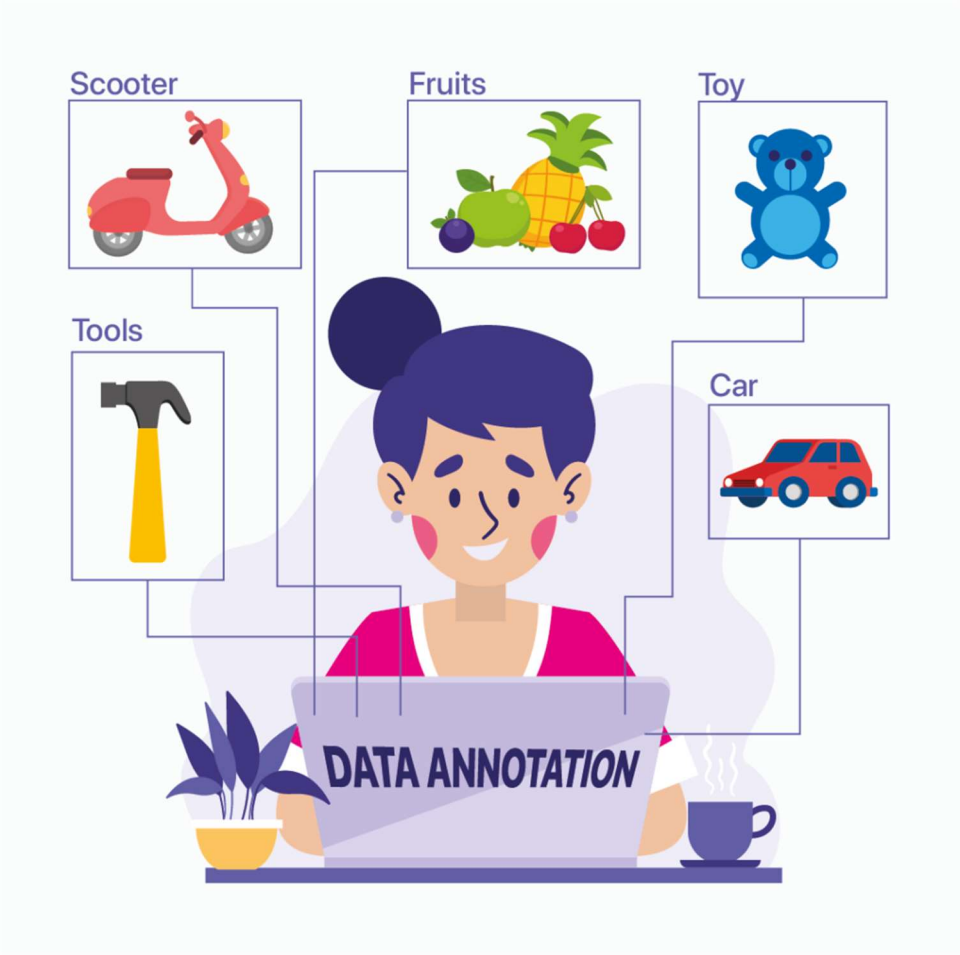
# Rapport de projet

# Annotation automatique des images




**Équipe des développeurs :**

Axel Arcidiaco

Jean Paul Sossah

Adrien Jacquenet

Tetyana Tarasenko

# Table des matières

# 1. Contexte

Le but de ce projet est de développer un modèle de deep learning capable de comprendre le contenu d'une image et de générer une légende décrivant l'image de manière précise et cohérente. Cette tâche est complexe car elle nécessite la compréhension de deux domaines distincts : la vision par ordinateur et le traitement du langage naturel. Le modèle doit être capable de combiner ces deux domaines pour produire des légendes précises.

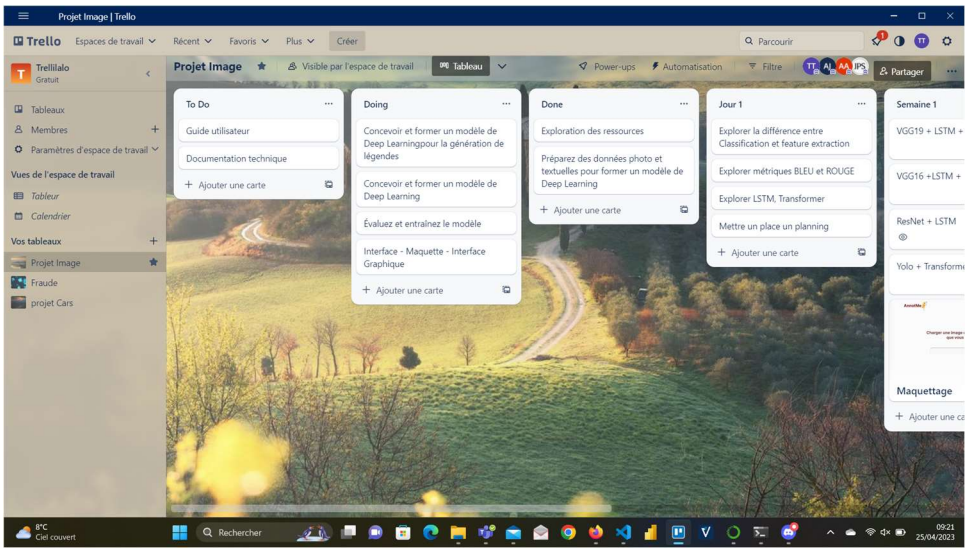
Le projet se divise en plusieurs étapes : la collecte et la préparation des données, l'entraînement du modèle, l'évaluation des performances du modèle et l'amélioration du modèle. Nous avons utilisé un ensemble de données de référence pour entraîner notre modèle et avons évalué ses performances à l'aide de métriques standard telles que BLEU et ROUGE.

Nous avons utilisé un réseau avec une couche LSTM pour la génération de légende et avons également exploré différents types d'extraction de features pour les images, tels que VGG, ResNet et Inception V3, pour déterminer laquelle est la plus efficace pour notre architecture.

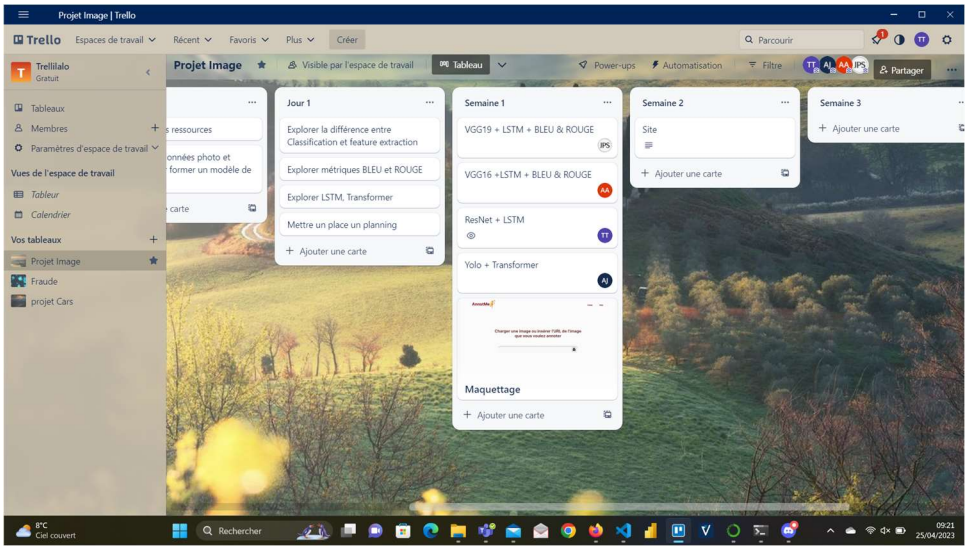
L'objectif final de notre projet est de créer une application conviviale qui permet aux utilisateurs de générer automatiquement des légendes pour leurs images en utilisant notre modèle d'apprentissage en profondeur. Cette application peut être utilisée dans divers domaines tels que la publicité, les réseaux sociaux, la reconnaissance d'images et bien plus encore.

# 2. Planification

Pour la planification de ce projet, nous avons choisi d'utiliser la méthode agile en utilisant l'outil **Trello**. Nous avons commencé par décomposer le projet en plusieurs tâches, puis nous avons créé des cartes Trello pour chacune de ces tâches. Nous avons ensuite organisé ces cartes en différentes colonnes représentant les différentes étapes du projet, telles que la planification, le développement, les tests et le déploiement.



Nous avons utilisé les fonctionnalités de Trello pour attribuer des tâches aux différents membres de l'équipe, ajouter des commentaires et des pièces jointes pour clarifier les exigences et les attentes, et suivre l'état d'avancement des différentes tâches.




En utilisant cette approche, nous avons pu suivre de près les progrès du projet et nous assurer que toutes les tâches étaient effectuées dans les délais impartis. Les réunions régulières de l'équipe nous ont permis de discuter des obstacles éventuels et de collaborer pour trouver des solutions efficaces.

En fin de compte, l'utilisation de Trello et de la méthode agile nous a permis de planifier efficacement le projet, d'adapter notre approche en fonction des changements de situation et de maintenir un haut niveau de communication et de collaboration tout au long du processus de développement.

## 3. Guide utilisateur


**Page d'accueil :**
Sur la page d'accueil, vous pouvez voir le nom et le logo de l'application dans le coin supérieur gauche. En haut à droite se trouvent deux options de menu : "**Accueil**" pour retourner à la page d'accueil et "**Aide**" pour obtenir le guide utilisateur qui donne ainsi une explication détaillée de l'utilisation de l'application.


Au centre de la page, vous trouverez un champ de téléchargement de photo qui vous permet de télécharger une photo à partir de votre ordinateur et le bouton “**Envoyer**”. En bas de la page se trouve le champ de copyright.

**Chargement de photo :**
Dès que vous cliquez sur le bouton "**Envoyer**", la photo est chargée et affichée sur une nouvelle page qui vous permet soit de la modifier, soit de l'annoter. La photo chargée est sauvegardée dans un dossier temporaire qui se supprime automatiquement.

**Page de modification de photo :**
Si vous souhaitez modifier la photo que vous avez téléchargée, vous pouvez cliquer sur le bouton "**Modifier**". Cela renvoie à la page d'accueil et une fonction en back-end supprime la photo du dossier temporaire. Si vous souhaitez annoter la photo, vous pouvez cliquer sur le bouton "**Annoter**".

**Page de visualisation de la photo annotée :**
La photo chargée est automatiquement analysée par notre modèle d'IA pour générer une description de la photo. Cette description est affichée sur la page de l'annotation. Si vous changez d'avis ou si vous souhaitez annuler les modifications, vous pouvez cliquer sur le bouton "**Accueil**" pour revenir à la page d'accueil pour choisir une autre image.


# 4. Documentation technique

Durant ce projet, nous avons travaillé sur le développement d'un système d'annotation automatique d'images à partir de légendes en langage naturel. Pour atteindre notre objectif, nous avons mis en place un **pipeline** de traitement de données en utilisant plusieurs techniques de préparation de données pour les images et les légendes.

## Données

Nous disposons d'un **Flickr8k_dataset** de 8091 images et 5 captions par images.

Elles ont été prétraitées puis séparées en données d'entraînement, de validation et de test.

## Pré-traitement des données

### Données image

1. Resizing des images au format (224,224,3) pour correspondre à l'entrée du modèle de feature extraction.

2. Transformation du type PIL en tableau Numpy.
3. Nous avons aussi créé un pipeline de data augmentation, optionnel, capable de tripler le nombre d'image en faisant une symétrie verticale et en ajoutant du flou à l'image initiale.

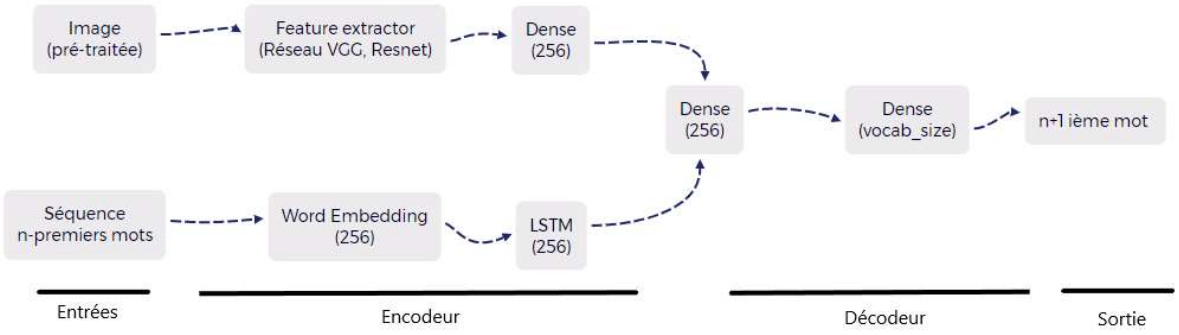
### Données textuelles

Ce processus de prétraitement des données est essentiel pour améliorer les performances des modèles de génération de légendes d'images, en fournissant un vocabulaire cohérent et en éliminant le bruit dans les descriptions.

1. La première étape consiste à extraire les légendes cible du modèle d'un fichier txt à l'aide d'expressions régulières.
2. Ensuite nous avons nettoyer les légendes en supprimant la **ponctuation**, les **stop-words** et avons effectué une **lemmatisation** dessus.
3. Une étape essentielle pour formater les données textuelles pour les adapter à l'entrée du modèle est la transformation de données textuelles en données numérique. On utilise pour cela la classe **Tokenizer** de Tensorflow qui fait cette transformation et créant un dictionnaire du vocabulaire existant dans les données.
4. **Tokenizer** nous permet également de créer une séquence. Car notre la couche **LSTM** prend en entrée une séquence de texte, c'est à dire une liste de mot qui correspond à tout ou partie de la légende (du 1$^{er}$ au n$^{ième}$ mot), et prend en sortie le n+1$^{ième}$ mot. **Tokenizer** créé cette séquence est le n+1$^{ième}$ mot.

Ces données prétraitées seront stockées dans un dataframe, prêt à être *splitté* et utilisé pour l'entraînement. On stock également dans le dataframe les données brutes pour pouvoir évaluer le modèle.

## Création du modèle




### Entrées

Notre modèle prend deux entrées distinctes :

1. L'image prétraité au format Numpy de shape (224,224,3)
2. Une séquence de n premiers mot de la caption.

### Sortie

La sortie du modèle génère ou prend en cible le n+1$^{ième}$ mot de la légende.

### Composition

#### *Encodeur*

**Branche 1 :** Dédiée à l'image, elle extrait les caractéristiques de l'image à l'aide d'un modèle de classification d'image pré-entraîné. Nous avons fait des essais avec **VGG16**, **VGG19**, **Resnet50V2** et **InceptionV3**. Puis on passe la sortie du modèle dans une couche dense.

**Branche 2 :** Dédiée au traitement du texte, réduit la dimensionnalité des mots et les plongeant dans un espace vectoriel réduit en fonction de la sémantique des mots. Ceci est effectué par la couche **Embedding**.

La couche LSTM prend en compte n premiers mots d'une légende et est redirigé vers une couche dense commune avec la branche 1.

#### *Décodeur*

La partie décodeur permet de décompresser les informations avec une couche dense de la taille du vocabulaire de notre texte de référence.

Le modèle est entraîné en utilisant la fonction de perte de **cross-entropy** catégorique et l'optimiseur **Adam**.

## Entraînement

Nous avons entraîné plusieurs modèles avec des extracteurs de caractéristiques différents : VGG16, VGG19 et ResNet50V2.

Les courbes d'apprentissage ont toute la forme ci-dessous. Synonyme que le modèle apprend bien.

## Évaluation

### BLEU

La métrique ***BLEU (Bilingual Evaluation Understudy)*** est une mesure couramment utilisée pour évaluer la qualité des traductions automatiques par rapport à une traduction de référence. Bien que développé pour la traduction, il peut être utilisé pour évaluer le texte généré pour une suite de tâches de traitement du langage naturel.

### ROUGE

***ROUGE (Recall-Oriented Understudy for Gisting Evaluation)*** est en fait un ensemble de mesures :

**ROUGE-N** : mesure le nombre de « n-grammes » correspondants entre notre texte généré par le modèle et une « référence ». (Équivalent à Bleu)

**Recall** : Nombre de gram qui match entre le modèle et la référence, diviser par le nombre de gram de la référence.

**Precision** : Nombre de gram qui match entre le modèle et la référence, divisé par le nombre de gram du modèle.

**F1-score :** Cela nous donne une mesure fiable des performances de notre modèle qui repose non seulement sur le fait que le modèle capture autant de mots que possible (rappel), mais le fait sans produire de mots non pertinents (précision).

### Tableau récapitulatif

| | BLEU | ROUGE F1-score |
|---|---|---|
| VGG16 | 1 : 0 ; 2 : 0 ; 3 :0 | 1 : 0,48 ; 2 : 0 ; l : 0,048 |
| VGG19 | 1 : 0 ; 2 : 0 ; 3 :0 | 1 : 0,046 ; 2 : 0 ; l : 0,46 |
| ResNet50V2 | 1 : 0 ; 2 : 0 ; 3 :0 | 1 : 0 ; 2 : 0 ; 3 : 0 |
| InceptionV3 | 1 : 0 ; 2 : 0 ; 2 :0 | 1 : 0,050 ; 2 : 0 ; 0,050 |

Les résultats ne sont pas satisfaisants.

## Traduire les légendes en français

Pour traduire les descriptions d'images générées par nos modèles d'annotation d'image automatique, nous avons utilisé la bibliothèque Python "Translate".

Translate est un outil de traduction simple mais puissant qui prend en charge l'API Microsoft Translation, l'API Translated MyMemory, LibreTranslate, les API gratuites et professionnelles de DeepL, et l'API de Google Translate qui est l'API utilisée par défaut pour générer les traductions.

# 5. Difficultés rencontrées

## Démarche technique

Nous avons appris qu'il serait préférable de ne pas partir de zéro quand on a un nouveau projet à mener. Nous avons exploré les ressources et tenter d'améliorer le tutoriel dont nous disposions dans les ressources. En effet, il traitait exactement du même sujet. Cependant, il n'est pas toujours facile de comprendre le code fait par autrui. Cela peut demander plus de temps de travail. Des efforts de compréhension ont été aussi nécessaires. L'organisation des fichiers dans le tutoriel a pu nous induire en erreur sur une autre façon de procéder.

## Puissance de calcul

Par moment, nos machines n'avaient pas suffisamment de puissance pour certaines opérations. Les tentatives dans Colab et Kaggle ont conduit à des plantages et des beugs. Il a fallu parfois tout relancer et patienter pour enfin avoir de mauvais résultats.

## Évaluation avec le rouge score

Nous avons constaté que le ROUGE SCORE ne s'implémentait pas de la même manière que le BLEU. Il nous a fallu un peu plus de temps pour mieux le comprendre et l'ajouter aux procédés d'évaluation.

# 6. Perspectives d'évolutions

Nous pouvons essayer d'implémenter les éléments suivants pour améliorer notre modèle :

- Data augmentation des images (symétrie, modification des contrastes, etc).
- Data augmentation des légendes (avec des synonymes par exemple)
- Essayer un autre modèle de word embedding comme avec gensim.
- Essayer d'autre modèle de feature extraction.
- Essayer un autre type de modèle de génération de légende. Avec des transformers par exemple.
- Tester d'autres paramètres d'entraînement (optimizer, learning rate, epochs)
- Modifier l'architecture de l'encodeur décodeur.
- Mieux sélectionner les légendes d'entraînement
- Améliorer le tokenizer. Beaucoup de mots inconnus persistaient.

# 7. Conclusion

Ce projet nous a permis de mettre en pratique les réseaux de neurones, premièrement sur la partie images avec le modèle de feature extraction que nous avons décortiquée afin de pouvoir extraire les caractéristiques des images. Mais aussi sur la partie NLP avec la génération de texte avec une couche LSTM est sa structure particulière.

Le prétraitement des images et texte a également été instructive avec la mise en application de connaissances diverses (expression régulière, tokenizer, transformation d'image).

Nous avons aussi renforcé nos connaissances Flask sur la partie application.

## 8. Bilan de groupe

Bonne ambiance communicative dans le groupe. L'entraide nous a aidé beaucoup.

## 9. Bilans personnels

**Axel Arcidiaco** : C'était un projet intéressant et très instructif. J'ai principalement travaillé sur la partie prétraitement et construction du modèle. J'ai beaucoup appris sur la construction d'un feature extractor et d'encodeur décodeur.

**Jean Paul Sossah** : Pour être franc, je n'ai pas été très emballé par ce projet qui se situait entre Computer Vision et le NLP. J'ai eu du mal à me structurer dans la démarche technique et je me suis cantonné à la première ressource. Cependant, j'ai bien aimé travailler sur le Back-end et le Front-end de l'application et pris les devants en y testant le modèle que j'avais entrainé. Cela m'a permis de ne pas rester sans rien faire et me mettre à la tâche là où je me sentais on peut plus compétent.

**Adrien Jacquenet** : Bilan positif, je me suis concentré sur la partie prétraitement et construction du modèle. J'ai beaucoup appris sur la construction d'un feature extractor et d'encodeur décodeur. J'ai également approfondi l'architecture projet en utilisant des classes et en séparant le pipeline dans plieurs fichiers. Avant ce projet, je ne connaissais pas les métriques BLEU et ROUGE. De plus il me reste beaucoup d'idées pour améliorer le modèle.

**Tetyana Tarasenko** : Ce projet a été très éducatif pour moi, qui m'a permis d'approfondir mes connaissances dans le fonctionnement des modèles de computer vision. J'ai pu entrainer différents modèles de Keras, notament : VGG16, ResNet50V2 et Inception_V3, comprendre les métriques BLEU. Qui restent encore à approfondir pour comprendre comment améliorer le modèle de prédiction.

## 10. Glossaire

**LSTM** : Long Short-Term Memory (Mémoire à Long Terme et Court Terme) est un type de réseau de neurones récurrents (RNN) qui est capable de capturer les relations à long terme dans les données séquentielles. Les LSTM sont particulièrement utiles pour la modélisation de séquences de données, telles que la reconnaissance de la parole, la traduction automatique, la génération de texte et la prédiction de séries chronologiques.

**Feature extraction** : l'extraction de caractéristiques - est un processus dans lequel des informations pertinentes sont extraites à partir de données brutes (dans notre exemple – photos) afin d'être utilisées pour l'analyse ou l'apprentissage automatique. Cela peut inclure l'extraction de mots-clés, la création de vecteurs de mots, l'extraction de n-grammes, la réduction de dimension, l'extraction de motifs, etc.

**Stop words** : des mots très courants qui sont souvent supprimés des textes avant le traitement afin de réduire la dimensionnalité du texte et d'améliorer l'efficacité des algorithmes de traitement du langage naturel. Ce sont généralement des mots qui n'ont pas de signification importante en soi et

qui sont fréquemment utilisés dans la langue en question, tels que "le", "la", "de", "un", "une", "et", etc.

**Lemmatisation** : est un processus de traitement linguistique qui consiste à regrouper les différentes formes flexionnelles d'un mot en une seule forme canonique, appelée lemme. Le lemme représente la forme de base d'un mot, qui est commune à toutes ses variations grammaticales. Par exemple, les formes "joue", "joues", "jouent", "jouait", "jouerez" ont pour lemme "jouer".

**Stemming** : est une technique de traitement du langage naturel qui consiste à réduire les mots à leur racine (ou "stem" en anglais), en enlevant les préfixes et suffixes. Cela permet de regrouper les différentes formes d'un même mot pour en faciliter l'analyse et le traitement automatique.

**Tokenization** : est le processus de division d'un texte en unités plus petites appelées tokens. Un token est une séquence de caractères consécutifs ayant une signification propre. Les tokens peuvent être des mots, des phrases, des symboles de ponctuation ou tout autre élément du texte que l'on souhaite prendre en compte dans l'analyse.

**BLEU** : est une mesure de la similarité entre deux textes en utilisant la précision des n-grammes communs. Utilisé pour évaluer la qualité de la traduction automatique. BLEU calcule la précision de chaque n-gramme dans le texte généré automatiquement par rapport au texte de référence. Les n-grammes dans le texte généré sont ensuite comparés avec ceux du texte de référence pour calculer la similarité. Plus le score BLEU est élevé, plus le texte généré automatiquement est similaire au texte de référence.

**ROUGE** : Recall-Oriented Understudy for Gisting Evaluation est une métrique d'évaluation de la qualité de la traduction automatique et de la génération de texte. Il mesure la similarité entre un texte généré automatiquement et un texte de référence (généralement un texte rédigé par un humain). Il calcule le rappel (recall) des n-grammes communs entre les deux textes. Les n-grammes sont des séquences de n mots consécutifs dans un texte. ROUGE peut être utilisé avec des n-grammes de différentes tailles, de 1 à 4 mots généralement. Plus le score ROUGE est élevé, plus le texte généré automatiquement est similaire au texte de référence.

**Optimiseur Adam** : (Adaptive Moment Estimation) est un algorithme d'optimisation du gradient stochastique (SGD) basé sur l'estimation adaptative du premier et du deuxième moment des gradients. Cet algorithme utilise une combinaison de moment de premier ordre (la moyenne des gradients) et de moment de deuxième ordre (la moyenne des carrés des gradients) pour mettre à jour les poids du réseau de neurones.

**Cross-entropy** : est une mesure de la différence entre deux distributions de probabilités, qui mesure la différence entre la distribution de probabilités prédite par un modèle et la distribution de probabilités réelle.

**Embedding :** est une technique qui consiste à projeter des données dans un espace vectoriel continu de dimensions réduites. Cela permet de représenter les données sous forme de vecteurs denses et de conserver les relations sémantiques entre les éléments.

**Pipeline** : séquence d'étapes de prétraitement, de transformation et de modélisation des données, qui sont enchaînées dans un ordre spécifique pour former un processus de bout en bout. Les pipelines d'apprentissage automatique sont utilisés pour automatiser et standardiser les tâches courantes, telles que la normalisation les données, la sélection des fonctionnalités, l'entraînement du modèle et l'évaluation des performances, afin de faciliter le développement et le déploiement de modèles de machine learning efficaces et robustes.