The Describer 🙄

Le concept 😊

Mise au point d'une **intelligence artificielle** capable de **décrire une image et son contexte**.

Création d'un écosystème d'aide aux personnes malvoyantes composé :

- D'un site web (https://the-describer.netlify.app) ;
- D'une application mobile ;
- D'une extension chrome ;
- D'un bot discord ;

Hiérarchie du projet 🤔

Exercices

- `./exos` > Exercices réalisés en début de projet afin de maîtriser les notions liées au deep learnin (ex: tenseurs, réseaux de neurones, gradient, fonctions d'erreur, etc.) ;

Projet

- `./projet` > Le contenu du projet
- `./projet/ai` > Le code permettant d'entrainer le réseau de neurones, de le tester et de faire des prédictions ;
- `./projet/backend/api` > L'API de prédictions qui permet de récupérer la description associée à une image (fichier ou url), réalisée en flask ;
- `./projet/bot_discord` > Le bot discord en python ;
- `./projet/chrome_extension` > L'extension chrome en javascript ;
- `./projet/mobile` > L'application mobile en flutter ;
- `./projet/web` > Le site web en vuejs + element ui ;

Faire tourner le projet 😁

Exercices

- `./exos` > Installer les dépendances python et lancer les programmes ;

Projet > Le réseau de neurones

- `./projet/ai` > Il y a **4 modes de lancement** : `install`, `train`, `test` et `predict`. Afin de choisir le mode, il suffit de mettre à jour la liste `todo` du fichier `main.py`. On pourra ensuite lancer la programme via la commande `python main.py`. NB : Certaines variables situées dans le main permettent de **modifier facilement et rapidement les paramètres du réseau de neurones**. Ces derniers sont d'ailleurs pratiquement utililes lors de la phase d'apprentissage du réseau

> `install` :
>
> - Prérequis : Téléchargement de la **base de données COCO** (datasets d'images et de libellés) dans le répertoire `./projet/ai/datadir`. Vous pouvez utiliser le **makefile** (`./projet/ai/Makefile` avec la commande `make`), qui permets de télécharger les datasets, ainsi que les librairies python nécessaires de manière automatique.
> - NB : Attention a bien installer la bonne version de `pytorch`. Privilégiez l'installation du mode `CUDA`, qui vous permettra d'augmenter grandement les temps de calculs, en utilisant le **GPU** de votre ordinateur.
> - Fonction : Charge un **vocabulaire** des différents mots prédictibles à partir des différents labels du dataset, applique des **transformations**, **redimentionne** l'ensemble des images et les **sauvegarde** dans le répertoire `./projet/ai/datadir/`

> `train`
>
> - Prérequis : `install`
> - Fonction : **Entraine** le réseau de neurones à partir des images et des labels du jeu de données COCO en fonction des paramètres spécifiés dans le main (totalEpochs, step, etc.). Une fois le réseau entraîné, **exporte les paramètres du réseau** (encodeur + décodeur) dans le répertoire `./projet/ai/models_dir`. Ces derniers ont une extension `.ckpt`.
> - NB : Si des fichiers d'extension `.ckpt` sont présentes dans le répertoire `./projet/ai/models_dir`, on charge le réseau en début de programme avec les valeurs des fichiers d'extension `.ckpt` les plus récentes. Ainsi, si on run 2 fois le programme de manière consécutive avec 20 epochs, cela reviendra à entrainer le même réseau une première fois avec 20 epochs, puis de reprendre l'avancement par la suite et de refaire 20 epochs, soit 40 épochs au total. Pour ne pas **reprendre l'état en cours d'entrainement du réseau**, il suffit donc de supprimer les fichiers du répertoire `./projet/ai/models_dir`.

> `test`
>
> - Prérequis : `train`

> - Fonction : Lance des tests d'accuracy (**précision**) pour les datasets d'apprentissage et de test à partir réseau généré par les fichiers d'extension `.ckpt` les plus récents du répertoire `./projet/ai/models_dir` .

> `predict`
>
> - Prérequis : `train`
> - Fonction : Effectue la **prédiction de l'image** indiqué par la variable `input_image_to_test_path` , et l'affiche. Comme précédemment, on se base sur le réseau généré par les fichiers d'extension `.ckpt` les plus récents du répertoire `./projet/ai/models_dir` .

Projet > L'API

- `./projet/backend/api`

> - Prérequis : Entrainement du réseau de neurones via les instructions précisées pour le répertoire `./projet/ai` . On récupère ensuite les fichier `.ckpt` les plus récents (en fonction de la date de génération précisée dans le nom de fichier), contenant les informations du réseau de neurones (encodeur et décodeur). En particulier, on **copie colle** ces deux fichiers dans le répertoire `./projet/backend/api/models_dir` , et on les renomme respectivement `encoder.ckpt` et `decoder.ckpt` .
> - Installation des **dépendances** du fichier `./projet/backend/api/requirements.txt` ;
> - Lancer en mode "test" : `flask run` ;
> - Lancer en mode "production" sur un serveur : `gunicorn app:app` ;

Projet > Le bot discord

- `./projet/bot_discord` > Le bot discord en python ;

> - Pour la production, on héberge le bot sur un serveur, et on le rends directement téléchargeable sur le site internet (https://the-describer.netlify.app).

Projet > L'extension de naviguateur

- `./projet/chrome_extension` > L'extension chrome en javascript ;

> - Pour la production, on génère un zip, qui est directement téléchargeable sur le site internet (https://the-describer.netlify.app).
> - TODO : Rendre l'extension officielle et la rendre téléchargeable via le **manageur d'extensions chrome**.

Projet > L'application mobile

- `./projet/mobile` > L'application mobile en flutter ;

> - Installer flutter, android studio et utiliser un émulateur ou bien un téléphone portable ;
> - Pour la production, on génère un APK, puis on le zippe. Celui-ci est directement téléchargeable sur le site internet (https://the-describer.netlify.app).
> - TODO : Rendre l'extension officielle et la rendre téléchargeable sur **l'Apple Store** et sur **Google Play**.

Projet > Le site web

- `./projet/web` > Le site web en vuejs + element ui ;

> - Installer `npm`, puis télécharger les ressources nécessaires au projet avec la commande `npm install`.
> - Lancer en mode "test" : `npm run serve` ;
> - Lancer en mode "production" : `npm run build` ;