|  |  |
| --- | --- |
| **Scénario: L’utilisateur veut entraîner un nouveau réseau, à partir d’un dataset** | |
| L’utilisateur lance le script en ligne de commande | Le programme retourne une erreur (Missing arguments) |
| L’utilisateur lance le script en ligne de commande, avec l’argument --new | Le programme retourne une erreur (Missing Dataset) |
| L’utilisateur lance le script en ligne de commande, avec les arguments --new --data <path> | Le programme ouvre le chemin fourni par l’utilisateur, et vérifie que les données sont valables. Les données ne sont pas valables, le script retourne une erreur (Invalid Dataset). |
| L’utilisateur lance le script en ligne de commande, avec les arguments --new --data <path> | Le programme ouvre le chemin fourni par l’utilisateur, et vérifie que les données sont valables. Les données sont valables, le script lance le processus de génération de réseaux. |
|  | Le programme tombe sur une image invalide. Le programme lance une erreur, mais continue l’entraînement. |
|  | Une fois l’entraînement terminé, le meilleur réseau est retenu et sauvegardé dans un dossier dédié dans le dossier actif. |

# Use case : Entraînement d’un réseau

|  |  |
| --- | --- |
| **Scénario: L’utilisateur veut entraîner un nouveau réseau, à partir d’un dataset, en le limitant à X images** | |
| L’utilisateur lance le script en ligne de commande | Le programme retourne une erreur (Missing arguments) |
| L’utilisateur lance le script en ligne de commande, avec l’argument --new | Le programme retourne une erreur (Missing Dataset) |
| L’utilisateur lance le script en ligne de commande, avec les arguments --new --data <path> -L <number> | Le programme ouvre le chemin fourni par l’utilisateur, et vérifie que les données sont valables. Les données ne sont pas valables, le script retourne une erreur (Invalid Dataset). |
| L’utilisateur lance le script en ligne de commande, avec les arguments --new --data <path> -L <number> | Le programme ouvre le chemin fourni par l’utilisateur, et vérifie que les données sont valables. Les données sont valables, le script lance le processus de génération de réseaux. |
|  | Le programme tombe sur une image invalide. Le programme lance une erreur, mais continue l’entraînement. |
|  | Le programme atteint le nombre limite d’images d’entraînement (argument *L)* |
|  | Une fois l’entraînement terminé, le meilleur réseau est retenu et sauvegardé dans un dossier dédié dans le dossier actif. |

|  |  |
| --- | --- |
| **Use Case : L’utilisateur veut utiliser le réseau pour reconnaître des chiffres manuscrits** | |
| L’utilisateur lance le script en ligne de commande, avec l’argument --recognize  --data<path> --network <path> | Le programme charge le réseau neuronal qui se trouve au chemin spécifié dans l’argument *network s’il existe*, charge les données (images) se trouvant au chemin spécifié dans l’argument *data s’il est valide*, les analyse, et affiche les résultats dans l’invite de commande. |
| Un fichier (Data ou réseau neuronal) est corrompu/Invalide | L’exécution du programme s’interrompt, une erreur est lancée et affichée dans l’invite de commande |

# Use case : Utilisation d’un réseau

|  |  |
| --- | --- |
| **Use Case : L’utilisateur veut utiliser le réseau pour reconnaître des chiffres manuscrits qu’il a écrit lui même** | |
| L’utilisateur écrit ses chiffres sur un éditeur d’images (paint, photoshop, …), et les enregistre dans un dossier spécifique, dans un format spécifique (bitmap, 28 pixels de côté). | Le programme récupère l’image, la convertit en format lisible par le réseau neuronal, et l’ajoute aux données d’entrainement & de test |
| L’utilisateur lance le script en ligne de commande, avec l’argument --recognize  --data<path> --network <path> | Le programme charge le réseau neuronal qui se trouve au chemin spécifié dans l’argument *network s’il existe*, charge les données (images) se trouvant au chemin spécifié dans l’argument *data s’il est valide*, les analyse, et affiche les résultats dans l’invite de commande. |