I) Description du système

1) Scripts présents

Le système se compose de :
2 fichiers pythons non exécutables de support (“libraries”)

- **`fcn_zhang.py`** : fonctions permettant de générer la structure du réseau, de charger les données d’apprentissage, d’entraîner un réseau
- **`dataprocessing.py`** : fonctions permettant de générer les jeux de données d’apprentissage et de prétraiter les images pour l’évaluation

4 fichiers pythons exécutables (“outils”) s’appuyant sur les précédents :

- **`fcn_zhang_porthos.py`** : permet de lancer un apprentissage sur Porthos
- **`process_image.py`** : permet de traiter une image
- **`prepare_remove_falseneg.py`** et **`gen_extraneg.py`** : permettent de générer des jeux de données négatifs à partir d’images traitées

Sur Porthos, le système est installé dans **`src/zhang`**

2) Description de la structure et le prétraitement des données d’apprentissage

a) Structure des donnés

Les données d’apprentissage sont des séries de fichiers .npz (archive NumPy compressée, un format qui peut contenir plusieurs arrays NumPy).

Chaque fichier npz contient 4 arrays:

- **“x”** : il s’agit de l’image 224x224 RGB, sous forme d’un array de flottants de dimension 224x224x3.
- **“y_match”** : image 224x224 contenant les zones de texte lisible.
- **“y_dontcare”** : image 224x224 contenant les zones de texte non lisible.
- **“y_fusion”** : image 224x224 contenant l’union des deux précédentes (donc toutes les zones de texte, lisible ou non)

Les valeurs de l’image x sont au départ codées entre 0 et 255, mais une valeur constante (mais différente pour chaque canal R/G/B) est soustraite conformément aux instructions du réseau VGG (voir ici dans “description”)

Les valeurs des arrays y_* sont des flottants entre 0 et 1. En principe la valeur est soit 0 (pas de texte) soit 1 (zone de texte) mais il peut y avoir des valeurs intermédiaires au bord des zones de texte à cause de l’antialiasing (ce qui n’est pas gênant).

L’entrainement se fait par défaut sur y_fusion. Il y a peu de raisons de le changer, mais cela peut se faire en modifiant la valeur par défaut du parametre y_ds dans l’instanciation de la classe DataSetLoader, dans `fcn_zhang.py`

b) Localisation des données

Le système d'apprentissage a besoin de connaître le répertoire où se trouve les fichiers (par défaut **./traindata/**) et un fichier texte listant les fichiers en question (par défaut **./trainsamples.txt**). (Les défauts sont changeables dans le fichier `fcn_zhang_porthos.py`)

c*) Génération des données*

Les données sont issues :

- En majorité du dataset robust reading competition, task 4 : les images du jeu de données ont été traitées (redimensionnées à plusieurs échelles, découpées en zones de 224x224, etc.) de facon à générer les fichiers .npz Aucun outil autonome pour ce faire n'est fourni pour le moment, mais la fonction correspondante est gen_all_data dans dataprocessing.py
- En minorité de données négatives (chemins de cable, etc.) issues des données EDF: les instructions pour générer les données sont données en II.6.

II) Procédures et howtos

1) Entrainer un réseau (sur Porthos)

1) Se connecter sur porthos : **`ssh i93332@porthos.hpc.edf.fr`**
2) Si nécessaire arrêter l'entraînement en cours (cf II.3)
3) S'assurer que les répertoires **`output/`** et **`output/weights/`** existent et sont vides.
   S'ils n'existent pas, il faut les créer avec : **`mkdir -p output/weights`**
   S'ils ne sont pas vide, ils contiennent les résultats d'un précédent run, il faut supprimer et/ou renommer output (par exemple **`mv output old-output`**) et les recréer vide avec **`mkdir -p output/weights`**
4) Lancer l'entrainement :

```
sbatch -N 1 --exclusive --partition=cg --mem=200G
--wckey=P11QW0:MOVIDO --qos=cg_0028c_168h
--reservation=ag06347s --time=168:00:00 ./zhang_net_sscript.sh
```

2) Obtenir une session shell sur un noeud graphique

A utiliser à fins d'expérimentation, de test et de débugage : les entraînement eux-mêmes doivent être lancés avec sbatch comme indiqué en I.1.

1) Se connecter sur porthos : **`ssh i93332@porthos.hpc.edf.fr`**
2) Demander un noeud : **`srun -N 1 --exclusive --partition=cg --mem=200G --wckey=P11QW0:MOVIDO -t 2:00:00 neos --resolution=1024x768 --qos=cg_0028c_168h`**
   (cette ligne demande un noeud pour 2h, une autre durée peut être spécifiée avec le tag -t)
3) Le serveur va afficher un bout de XML qui contient entre autre l'IP du noeud graphique réservé, par exemple
   `<ipaddress>10.114.116.201</ipaddress>`
4) Se connecter en SSH au noeud dont on a récupéré l'adresse : **`ssh i93332@<ip du noeud>`**

3) Arrêter un entraînement en cours sur Porthos

1) Se connecter sur porthos : **`ssh i93332@porthos.hpc.edf.fr`**
2) Trouver le job slurm correspondant à celui qu'on a lancé: **`squeue | grep i93332`**
3) Il devrait afficher une ligne avec un numéro de job : faire ensuite **`scancel <numero de job>`**

4) Récupérer les données d'entrainement sur Porthos

Un run génère systématiquement :

- Des logs dans `output/log.txt`

- Des fichiers de poids après les différentes époques dans `output/weights/` (les époques "manquantes" correspondent à des époques où l'erreur de validation n'a pas baissé)
- Si l'entrainement est arrivé jusqu'au bout : un fichier `output/final.gz` qui contient des objets python représentant des statistiques sur l'entrainement et le contenu des jeux de données de validation et de test.

Il faut donc simplement copier les fichiers correspondant à coup de scp.

5) Traiter une image avec un réseau entraîné

Il faut utiliser le script **`process_image.py`** qui prend :

- Un paramètre obligatoire : le nom de l'image à traiter
- Trois paramètres optionnels :
  - Le fichier de poids à utiliser : par défaut `default-weights-textdetect.h5` est utilisé
  - Le répertoire de sortie : par défaut le répertoire courant
  - Le paramètre "overlap factor" qui impacte la facon dont l'image est découpée en imagettes de 224x224 et détermine le nombre de fois où le réseau est appliqué à chaque pixel. Le défaut de 2 devrait être bon dans le cas général ; augmenter ce nombre devrait augmenter légèrement la qualité et la précision de la reconnaissance au prix d'un temps de traitement plus élevé.

Le script génère trois fichiers :

- `<nom de l'image>-heat.png` : la heatmap brute
- `<nom de l'image>-blend.png` : la heatmap superposée à l'image initiale
- `<nom de l'image>-heatmap.npz` : une archive NumPy compressée contenant la heatmap sous forme d'array NumPy

6) Générer des données négatives à partir des données EDF pour désapprendre les chemins de cable, etc.

Sur bigware :

1) Sélectionner une image à utiliser et la copier dans le répertoire des scripts
2) La traiter avec le réseau comme decrit en II.5 :
   `python process_image.py <nom de l'image>`
3) Créer (s'il n'existe pas) le répertoire **remove_falsepos/**
4) Lancer le script de prétraitement :
   `python prepare_remove_falseneg.py <nom de l'image>`

   Ce script va tourner un certain temps et générer un dossier **remove_falsepos/<nom de l'image>/** avec trois sous-dossiers :
   - **positive/** : contient toutes les vignettes 224x224 incluses dans l'image contenant une détection de texte positive (avec un seuil assez bas)
   - **positive-blend/** : les mêmes mais à partir de l'image "blend"
   - **negative/** : les images négatives (pas de détection)

5) Aller dans **remove_falsepos/<nom de l'image>/positive-blend/**, parcourir les images à la main et **supprimer tous les fichiers correspondant à des vrais positifs** (des images positives contenant véritablement du texte)
   /!\ Ne pas toucher à remove_falsepos/<nom de l'image>/positive/ et remove_falsepos/<nom de l'image>/positive/ /!
6) Créer, s'il n'en existe pas déjà un, un dossier destiné à accueillir le jeu de données généré : par exemple
   `mkdir negative_dataset`
7) Utiliser le script **gen_extraneg.py** pour générer le dataset : la syntaxe est :
   `python gen_extraneg.py remove_falsepos/<nom de l'image> --use-trueneg=[true|false] <outdir> <fileslist>`
   Les paramètres sont :
   - Le premier est le chemin du dossier généré à l'étape 4, contenant les sous-dossiers positive/, negative/, etc.
   - Le second détermine si le jeu de données contiendra uniquement les faux positifs (--use-trueneg=false) ou bien les faux positifs et les vrais négatifs (--use-trueneg=true) : j'avais essayé la second approche mais je recommande plutôt la première maintenant.
   - Le troisième est le nom du dossier crée à l'étape précédente
   - Le quatrième est le nom d'un fichier texte (qui sera crée s'il n'existe pas) ou sera ajoutée la liste des fichiers générés.

Exemple complet pour une image appelée IMG_4242.JPG :

1) **`cp ..... ./IMG_4242.JPG`**
2) **`python process_image.py IMG_4242.JPG`**
3) **`mkdir remove_falsepos`** (si besoin)
4) **`python prepare_remove_falseneg.py IMG_4242.JPG`**
5) Aller dans **`./IMG_4242/positive-blend/`** et supprimer toutes les images contenant des vrais positifs
6) **`mkdir negative_dataset`** (si besoin)
7) **`python gen_extraneg.py remove_falsepos/IMG_4242 --use-trueneg=false negative_dataset negativedata.txt`**

Note : Le répertoire **remove_falsepos/<nom de l'image>/** peut être supprimé après ces opérations

Note 2 : Le répertoire de sortie et le fichier listant les données (ici **`negative_dataset`** et **`negativedata.txt`**) peuvent être réutilisés pour les données issues de plusieurs images, de facon a constituer un jeu de données négatives unique issu de plusieurs images.

7) Modifier les données d'apprentissage utilisées sur Porthos

Comme vu en I.2.b, le script d'apprentissage utilise un répertoire contenant les données, et une liste de fichiers présents dans ce répertoire qui définit les données qui seront effectivement utlilisées pour l'apprentissage. Par défaut, ce dossier et ce fichier sont

respectivement **traindata/** et **trainsamples.txt** dans le dossier où est installé le système (`src/zhang` sur porthos[1]).

Pour modifier les données ou en ajouter il faut donc

- D'une part ajouter tous les fichiers à utiliser au dossier **traindata/** : le plus simple est de tout copier par SCP depuis le répertoire contenant les fichiers .npz (par exemple `negative_dataset` dans l'exemple de II.6 :

scp *.npz i93332@porthos.hpc.edf.fr:src/zhang/traindata/

  (depuis bigware, dans `negative_dataset/`)
- D'autre part modifier le fichier `trainsamples.txt` pour qu'ils contiennent les données à utiliser.
  Le plus simple est de recréer le fichier à partir des listings des jeux de données à utiliser: par exemple les fichiers du robust reading competition (fichier `trainsamples-rrc_t4.txt` en pièce jointe) et ceux générés à partir des données EDF (fichier `negativedata.txt` dans l'exemple précédent) sur Bigware :
  cat trainsamples-rrc_t4.txt negativedata.txt > newtraindata.txt
  Puis de copier la liste sur Porthos :
  scp newtraindata.txt i93332@porthos.hpc.edf.fr:src/zhang/trainsamples.txt

On peut alors relancer l'entrainement avec la procédure II.1 sur le nouveau jeu de données.

[1] Le jeu de données étant volumineux, `src/zhang/traindata` est en réalité un lien symbolique pointant vers un répertoire du scratch (espace de stockage de masse), mais cela ne change rien du point de vue des scripts.