Google Deepdream + Docker + Video

1. Installation

Prérequis:

- Docker
- git

Installation

0. Move to a directory of your choice, where the source code will downloaded

```
cd /path/to/my/directory
```

1. Clone repo

```
git clone https://github.com/Bapuch/DeepDreamVideo-Docker.git
cd DeepDreamVideo-Docker
```

2. Build the docker container

```
docker build -t deepdream .
```

2. Utilisation

> Il faut place la video d'origine dans le folder `data` (sous-dosser optionnel `videos`) pour qu'elle soit ensuite accessible dans docker de meme pour l'image guide, elle doit se trouver dans le folder `data` (sous-dosser optionnel `guide_pictures`)

Par defaut toutes les frames converties se trouveront dans un sous dossier de `data/output_frames` qui portera le nom de la video

Deepdream mode 0 à 3

N.B.: pour les modes 1 et 3 les hyper params ne sont pas utiles

with default params

```
docker run -v $PWD/data:/data deepdream -e /data/videos/my_video.mp4
```

With custom params

```
docker run -v $PWD/data:/data deepdream -e /data/videos/my_video.mp4 -itr 6
--blend 0.85 --layers inception_4c/output

docker run -v $PWD/data:/data deepdream -e /data/videos/my_video.mp4 -itr 6
--blend 0.85 --gi /data/guide_pictures/my_picture.jpg
```

Single Picture

Pour tester l'effet des différents layers il est possible de tester une seule image à travers une liste de layers

Example le modèle par défaut et la liste de layers par défaut

```
docker run -v $PWD/data:/data deepdream -sp /data/single_pictures/hulk.jpg
```

Example avec un modèle téléchargé manuellement

```
docker run -v $PWD/data:/data -v $PWD/models:/models deepdream -p
models/places205CNN -m places205CNN_iter_300000_upgraded.caffemodel -sp
/data/single_pictures/hulk.jpg --layers conv1 pool1 norm1 conv2 pool2 norm2
conv3 conv4 conv5 pool5
```

Dans cet exemple, on obtient les fichiers suivants:

```
data/
├ single_pictures/
│ ├ hulk/
│ │ ├ dream_hulk_L-conv1_i-10_o-4_os-1.4_ss-1.5_j-32_(60sec).jpg
│ │ ├ dream_hulk_L-conv2_i-10_o-4_os-1.4_ss-1.5_j-32_(189sec).jpg
│ │ ├ dream_hulk_L-conv3_i-10_o-4_os-1.4_ss-1.5_j-32_(279sec).jpg
│ │ ├ dream_hulk_L-conv4_i-10_o-4_os-1.4_ss-1.5_j-32_(345sec).jpg
│ │ ├ dream_hulk_L-conv5_i-10_o-4_os-1.4_ss-1.5_j-32_(390sec).jpg
│ │ ├ dream_hulk_L-norm1_i-10_o-4_os-1.4_ss-1.5_j-32_(61sec).jpg
│ │ ├ dream_hulk_L-norm2_i-10_o-4_os-1.4_ss-1.5_j-32_(190sec).jpg
│ │ ├ dream_hulk_L-pool1_i-10_o-4_os-1.4_ss-1.5_j-32_(61sec).jpg
│ │ ├ dream_hulk_L-pool2_i-10_o-4_os-1.4_ss-1.5_j-32_(190sec).jpg
│ │ └ dream_hulk_L-pool5_i-10_o-4_os-1.4_ss-1.5_j-32_(389sec).jpg
└ └ hulk.jpg
```

On peut aussi définir tous les autres hyper paramètres du modèle

Obtenir plus de modèles: mode 4 et 5

Une série de modèle prêts à être téléchargé est disponible. Pour en obtenir il faut lancer le mode 4

```
docker run -it deepdream --mode 4
```

la liste apparait dans le terminale. Entrez le numero du modèle à télécharger:

```
Found 31 candidate models in /deepdream/caffe/models
0 - bvlc_googlenet
1 - bvlc_reference_caffenet
2 - finetune_flickr_style
3 - bvlc_reference_rcnn_ilsvrc13
4 - bvlc_alexnet
5 - hed_pretrained_bsds
6 - VGG16_SalObjSub
7 - VGG_ILSVRC_19_layers
8 - GoogleNet_SOD_finetune
9 - VGG_CNN_F
10 - fcn-8s-pascal
11 - VGG_VOC2012ext
12 - VGG_CNN_S
13 - VGG_CNN_M_128
14 - naacl15_pool_vgg_fc7_mean_fac2
15 - VGG_CNN_M_1024
16 - yearbook_cleaned
17 - VGG_CNN_M_2048
18 - gender_net
19 - nin_imagenet
20 - VGG_ILSVRC_16_layers_fc_reduced
21 - VGG_CNN_M
22 - googlenet_finetune_web_car_iter_10000
23 - KevinNet_CIFAR10_48
24 - VGG16_SOD_finetune
25 - s2s_vgg_pstream_allvocab_fac2_iter_16000
26 - GoogleNet_SOS
27 - AlexNet_SalObjSub
28 - cifar10_nin
29 - EmotiW_VGG_S
30 - VGG_ILSVRC_16_layers
Enter the number of model (or press q to quit) :
>
```

Par example, en choisissant 22, on voit ensuite les layers du modèle `googlenet_finetune_web_car_iter_10000.caffemodel`. Il faudra en préciser au moins un avec l'argument `--layers` pour utiliser ce modèle

```
data                                                conv1
pool1
norm1                                               conv2_1x1
```

```
conv2_3x3
norm2                                                pool2
pool2_pool2_0_split_0
pool2_pool2_0_split_1                                pool2_pool2_0_split_2
pool2_pool2_0_split_3
inception_3a_1x1                                     inception_3a_3x3_reduce
inception_3a_3x3
inception_3a_5x5_reduce                              inception_3a_5x5
inception_3a_pool
inception_3a_pool_proj                               inception_3a_output
inception_3a_output_inception_3a_output_0_split_0
inception_3a_output_inception_3a_output_0_split_1
inception_3a_output_inception_3a_output_0_split_2
inception_3a_output_inception_3a_output_0_split_3
inception_3b_1x1                                     inception_3b_3x3_reduce
inception_3b_3x3
inception_3b_5x5_reduce                              inception_3b_5x5
inception_3b_pool
inception_3b_pool_proj                               inception_3b_output
pool3

...
```

Pour revoir les layers du modèle: `--mode 5`

```
docker run deepdream --mode 5 -m
googlenet_finetune_web_car_iter_10000.caffemodel
```

La commande pour utiliser `googlenet_finetune_web_car_iter_10000.caffemodel` avec le layer `inception_3a_5x5`

```
docker run -v $PWD/data:/data deepdream -e /data/videos/my_video.mp4 -itr 6
--blend 0.85 -m googlenet_finetune_web_car_iter_10000.caffemodel --layers
inception_3a_5x5
```

Utiliser un modèle télécharger manuellement

Il faudra:

1. placer le modèle dans un dossier particulier (exemple: `models`)
2. creer un sous dossier avec le nom du modèle (exemple: `models\my_model\`) et y placer les fichiers téléchargés Pour chaque commande
3. attaché le volume `models` a docker avec `-v $PWD/models:/models`
4. préciser le chemin du dossier du modele avec `-p /models/my_model`
5. préciser le nom complet du modèle avec `-m my_model.caffemodel`
6. Obtenir la liste des layers du modèle avec `--mode 5`

    - `docker run -v $PWD/models:/models deepdream --mode 5 -m my_model.caffemodel -p /models/my_model`
7. Préciser les layers à utiliser pour faire tourner deepdream:

```
docker run -v $PWD/data:/data -v $PWD/models:/models deepdream -e /data/videos/my_video.mp4 -m my_model.caffemodel -p /models/my_model --layers layers1 layers2 [...]
```

Autre commandes

- Accèder au bash du container (avec le volume `data` attaché optionnellement)

```
sudo docker run -it --entrypoint bash -v $PWD/data:/data deepdream
```

3. Liens utiles

- Visualizing every layer of GoogLeNet with Python
- Deep dream Data sets

4. Parametres

Optionnels - run

--mode

Action(s) à faire

- **Valeur par défaut**: 0
- Choix: de 0 à 5

> - 0: (default) run all (create frames, dream and recreate the video)
> - 1: extract frames only
> - 2: run deepdream only (make sure frames are already where they should be)
> - 3: make the video from already existing processed frames
> - 4: download a new model
> - 5: show layers (requires --model-name and --model-path if different from default)

-sp, --single-picture

Chemin de l'image

- pas besoin de préciser --input
- par defaut **output_dir**=`data/single_picture/<picture_name>/`
- une image sera générer par layer

    - les images seront nommé avec les differentes parametres utilisés

-i, --input

Chemin du dossiers où seront extrait les frames de la video

- **Valeur par défaut**: "./data/input_frames"
- requis pour les mode 0 et 1 si '--exctract' n'est pas donné

-o, --output

Chemin où sera placée la deepdream video

- **Valeur par défaut**: "./data/input_frames"

-e, --exctract

Chemin de la vidéo d'origine

- requis pour les mode 0 et 3 si '--input' n'est pas donné

-it, --image-type

Extension des frames

- **Valeur par défaut**: "jpg"
- Choix: 'jpg' ou 'png'

-v, --verbose

Verbosité du programme, détails donnés dans le terminale

- **Valeur par défaut**: 2
- Valeurs: 1 ou 2
- 2: donnera l'avancement pour chaque iterations de chaque octave
- 1: donnera seulement l'avancement par frame (moins detaillé)

-gi, --guide_image

chemin pour l'image guide

-sf, --start_frame

Numero de la frame à partir de laquelle commencer le process

- **par défaut**: *première frame*

-ef', --end_frame

Numero de la dernière frame à processer

- **par défaut**: *dernière frame*

Optionnels hyper param

-p, --model-path

Chemin du dossier où se trouve le model `.caffemodel`

- **par défaut**: 'caffe/models/bvlc_googlenet/'

-m, --model-name

Nom du model `.caffemodel`

- **par défaut**: 'bvlc_googlenet.caffemodel'

-oct, --octaves

Nombre d'octave

- **Valeur par défaut**: 4
- valeur entière positive uniquement
- a un impact sur le computation time :
    - 1 super iteration par octave
    - chaque super-iteration contient *iteration* iterations (10 par defaut)
    - 4 * 10 = 40 iterations en tout avec les valeurs par défaut

> pour chaque octave le frame est redimensionné selon l'*octave_scale* jusqu'a revenir à la diemension d'origine de l'image si octave = 4 et octave_scale = 1.4 et iterations = 10: 1ere octave dimension (383, 612) - 10 iterations 2ème octave dimension (536, 857) - 10 iterations 3ème octave dimension (750, 1200) - 10 iterations 4ème octave dimension (1050, 1680) - 10 iterations

plus il y aura d'octave plus les formes "rêvées" seront "blended", moins reconnaisbles mais bien presentes s'il y a peu d'octave on vera peu de chose ressortir

-octs, --octavescale

valeur pour l'octave scale

- **Valeur par défaut**: 1.4
- nombre décimaux
    - eviter les valeurs inférieures à 1
- a un impact sur le computation time:
    - plus l'image est petite plus ça va vite
    - si l'*octave_scale* est élevée, les premières frames seront très petites

plus cette valeur est faible, plus on vera d'animaux apparaitre dans les plus petits details, plus elle grande plus les animaux seront surtout distinct sur les formes plus grandes

-itr, --iterations

Nombre d'itérations

- **Valeur par défaut**: 10

- valeur entière positive uniquement
- a un impact sur le computation time: autant d'itérations pour chaque octave

-j, --jitter

Nombre d'itérations

- **Valeur par défaut**: 10
- valeur entière positive uniquement
- parametre de la descente de gradient:
    - range pour la valeur du shift choisi aléatoirement à chaque pas

-s, --stepsize

Nombre d'itérations

- **Valeur par défaut**: 10
- valeur entière positive uniquement
- parametre de la descente de gradient:
    - pour la normalization

-b, --blend

Technique de blending

- **Valeur par défaut**: 10
- Exemple: `"0.5"` (constant), `"loop"` (0.5-1.0), `"random"`

The best results come from a well selected blending factor, used to blend each frame into the next, keeping consitancy between the frames and the dreamed up artefacts, but without the added dreamed artefacts overruling the original scene, or in the opposite case, switching too rapidly.

blending can be set by `--blend` and can be a *float* (default 0.5), "random" (a random float between 0.5 and 1., where 1 means disregarding all info from the old frame and starting from scratch with dreaming up artefacts), and "loop" which loops back and forth from 0.5 to 1.0, as originally done in the Fear and Loathing clip.

- every next unprocessed frame in the movie clip is blended with the previous processed frame before being "dreamed" on, moving the alpha from 0.5 to 1 and back again (so 50% previous image net created, 50% the movie frame, to taking 100% of the movie frame only). This takes care of "overfitting" on the frames and makes sure we don't iteratively build more and more "hallucinations" of the net and move away from the original movie clip.

-l, --layers

List des layers

- **Valeur par défaut**: `customloop`
    - boucle sur une liste des layers prédéfinis pour le model `bvlc_googlenet.caffemodel` (ne fonctionne donc que pour ce modèle), un layer par frame. Voir liste ci-dessous.
- Example:

- `inception_4c/output`,
- `inception_3b output inception_4a output inception_4b output inception_4c`

> layers locked to moving upwards from inception_4c/output to inception_5b/output (only the output layers, as they are most sensitive to visualizing "objects", where reduce layers are more like "edge detectors") and back again

> N.B.: si `[customloop]` est choisi, voici la liste des layers concernés layersloop = ['inception_4c/output', 'inception_4d/output', 'inception_4e/output', 'inception_5a/output', 'inception_5b/output', 'inception_5a/output', 'inception_4e/output', 'inception_4d/output', 'inception_4c/output']