Cahier de charge du projet

Le but de ce projet est de créer Generative Adversarial Network pour générer un ensemble de données images (panda animal) par la suite.

l'architecture de ce modele GAN est composée de deux réseaux de neurones,. Le premier, appelé **générateur**, crée un échantillon de données : faux images panda.

Le deuxième réseau, appelé **discriminateur** tente de détecter si l'image panda est réelle ou générer par le générateur.

Entrainement de notre modèle sur un GPU SPELL et enregistrement su résultat final.

Réalisation :

Step 1 :

J'ai utilisé une dataset publique via google dataset search qui correspond à un ensemble d'image panda de taille 188MB équivalent à 1500 images, puis la mettre dans le même root que mon fichier python.

Step 2 :

J'ai redimensionné les images pour une dimension unique 128x128 en utilisant la bibliotheque PILLOW et les stocker dans une liste : numpy array. Après pour remodeler et normaliser j'ai utiliser numpy puis sauvegarder la liste des image dans un fichier binaire npy à savoir 'outputAnimaldataset.npy'.

Step 3 :

J'ai procéder à la création du modèle du GAN en utilisant Keras.

J'ai utilisé la configuration suivante pour balayer toutes les images de ma dataset EPOCHS = 1500 et BATCH_SIZE = 32 nombre de flux image pour chaque itération ainsi charger ma liste 'outputAnimaldataset.npy' .

Step 4 :

Création de la fonction descriminator, pour cela j'ai initialise un model séquentiel de keras pour pouvoir créer mes couches linéaire.

après j'ai rajouté une couche LeakyRelu qui est une fonction d'activation puis pour la derniere couche une fonction d'activation sigmoid. Le but de la fonction desciminator est une fonction de classification binaire ici qui permet de définir si les images de pandas sans réels ou fausses.

Step 5

création de la fonction generateur, j'ai initialisé avec le meme modele séquentiel.

je donne mon modele summary :

Layer (type)	Output Shape	Param #
dense_12 (Dense)	(None, 4096)	413696
reshape_6 (Reshape)	(None, 4, 4, 256)	0
up_sampling2d_26 (UpSampling	(None, 8, 8, 256)	0
conv2d_61 (Conv2D)	(None, 8, 8, 256)	590080
batch_normalization_50 (Batc	(None, 8, 8, 256)	1024
activation_31 (Activation)	(None, 8, 8, 256)	0
up_sampling2d_27 (UpSampling	(None, 16, 16, 256)	0
conv2d_62 (Conv2D)	(None, 16, 16, 256)	590080
batch_normalization_51 (Batc	(None, 16, 16, 256)	1024
activation_32 (Activation)	(None, 16, 16, 256)	0
up_sampling2d_28 (UpSampling	(None, 32, 32, 256)	0
conv2d_63 (Conv2D)	(None, 32, 32, 256)	590080
batch_normalization_52 (Batc	(None, 32, 32, 256)	1024
activation_33 (Activation)	(None, 32, 32, 256)	0
up_sampling2d_29 (UpSampling	(None, 64, 64, 256)	0
conv2d_64 (Conv2D)	(None, 64, 64, 256)	590080
batch_normalization_53 (Batc	(None, 64, 64, 256)	1024
conv2d_65 (Conv2D)	(None, 128, 128, 256)	590080
batch_normalization_54 (Batc	(None, 128, 128, 256)	1024
activation_35 (Activation)	(None, 128, 128, 256)	0

Total params: 3,369,216
Trainable params: 3,366,656
Non-trainable params: 2,560

Step 6 :

Création d'une fonction helper qui permet de sauvegarder les images apés chaque iteration, elle prend comme parametres le bruit et le compte des images.

Step 7 :

Compilation du modèle : toute les étapes sont décrites dans le code, je voulais mentionnée que j'ai utilisé optimization Adam.

Step 8 :

Entrainement du modèle, pour cela j'ai utilisé comme décrit au début un serveur avec un GPU qui s'appelle SPELL et j'ai procéder ainsi :

```
pip install spell
```

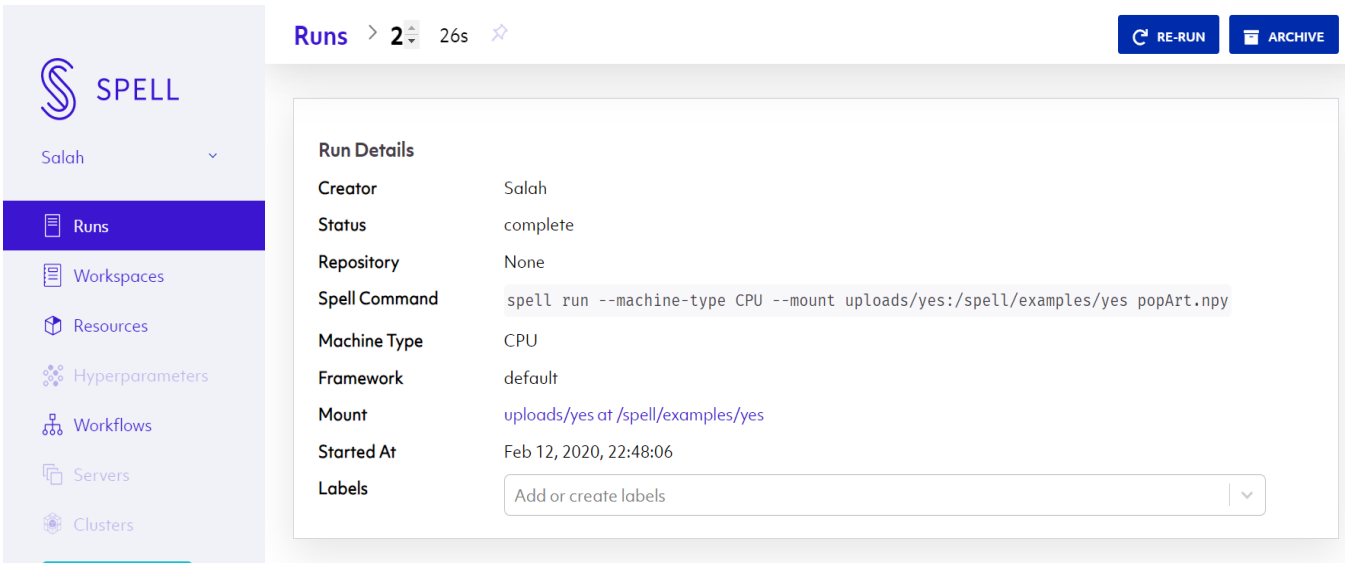
```
spell logi
```

```
OMP: Info #250: KMP_AFFINITY: pid 3200 tid 18160 thread 22 bound to OS proc set 5
OMP: Info #250: KMP_AFFINITY: pid 3200 tid 14664 thread 23 bound to OS proc set 7
OMP: Info #250: KMP_AFFINITY: pid 3200 tid 2588 thread 24 bound to OS proc set 0
OMP: Info #250: KMP_AFFINITY: pid 3200 tid 19996 thread 25 bound to OS proc set 2
OMP: Info #250: KMP_AFFINITY: pid 3200 tid 4212 thread 26 bound to OS proc set 4
OMP: Info #250: KMP_AFFINITY: pid 3200 tid 7080 thread 27 bound to OS proc set 6
OMP: Info #250: KMP_AFFINITY: pid 3200 tid 18864 thread 28 bound to OS proc set 1
OMP: Info #250: KMP_AFFINITY: pid 3200 tid 8752 thread 29 bound to OS proc set 3
OMP: Info #250: KMP_AFFINITY: pid 3200 tid 12360 thread 30 bound to OS proc set 5
OMP: Info #250: KMP_AFFINITY: pid 3200 tid 15804 thread 31 bound to OS proc set 7
OMP: Info #250: KMP_AFFINITY: pid 3200 tid 13100 thread 32 bound to OS proc set 0
OMP: Info #250: KMP_AFFINITY: pid 3200 tid 12256 thread 33 bound to OS proc set 2
OMP: Info #250: KMP_AFFINITY: pid 3200 tid 18292 thread 34 bound to OS proc set 4
OMP: Info #250: KMP_AFFINITY: pid 3200 tid 18544 thread 35 bound to OS proc set 6
[I 10:58:20.791 NotebookApp] Interrupted...
[I 10:58:20.793 NotebookApp] Shutting down 3 kernels
forrtl: error (200): program aborting due to control-C event
Image              PC                Routine            Line        Source
libifcoremd.dll    00007FFA98B53B58  Unknown               Unknown  Unknown
KERNELBASE.dll     00007FFB0FF35FA3  Unknown               Unknown  Unknown
KERNEL32.DLL       00007FFB10B67BD4  Unknown               Unknown  Unknown
ntdll.dll          00007FFB121CCED1  Unknown               Unknown  Unknown
[I 10:58:22.404 NotebookApp] Kernel shutdown: 4e4844cf-c568-4708-8d77-8e82aff74694
[I 10:58:22.406 NotebookApp] Kernel shutdown: 1d7eca95-3cf0-469e-9eba-f7fba4666ad3
[I 10:58:22.410 NotebookApp] Kernel shutdown: f310d794-7702-430f-af7a-db33c8f15da6

(base) C:\Users\SSAPROBOT-SH>spell login
Enter your spell username or email: salah
Enter your spell password:
```

Pour exécuter mon code sur le serveur :

```
Spell run python art_gan.py -t V100 -m
uploads/art_gan/'outputAnimaldataset.npy'
```



Après les logs pour le RUN

Logs

Show less verbose

```
Run created -- waiting for a CPU machine.
Run is building
Feb 12, 2020, 22:48:05: building: Machine acquired -- commencing run
Feb 12, 2020, 22:48:07: building: Retrieving cached/private environment...
Run is mounting
Feb 12, 2020, 22:48:27: mounting: Successfully added mount: uploads/yes:/spell/examples/yes
Run is running
```

Modèle descriptif de l'architecture de mon projet avec une partie du résultat :