StyleGAN Ideas Locas CDO

Por supuesto, el primer paso es hacer un git clone del repositorio de github

```
git clone https://github.com/NVlabs/stylegan
```

Este artículo explica todo muy claramente. Cómo se puede hacer uso de stylegan para generar personajes de anime.

```
https://www.gwern.net/Faces#karras-et-al-2018-figures
```

Características de la máquina usada para la PoC

Se hizo uso de Azure Cloud. De toda la oferta de instancias ofrecidas, en concreto nos servía esta:

Standard NV12 Promo con Linux Ubuntu 18.04

	NV12
Cores	12 (E5-2690v3)
GPU	2 x M60 GPU (1/2 Physical Card)
Memory	112 GB
Disk	380 GB SSD

1. Setup del entorno

```
sudo apt update
sudo ubuntu-drivers
sudo ubuntu-drivers autoinstall
sudo reboot
```

Si funciona bien, instalará muchos paquetes deb, como "nvidia-driver-390", etc. Luego reinicie su máquina. Si se reinicia con éxito, abra una terminal y use el comando "nvidia-smi". Si le muestra correctamente el estado de la GPU de Nvidia como se muestra a continuación, entonces ya está correctamente instalado

1.1. Asegúrate de que nvidia esté habilitada

Si compra una máquina preinstalada en Ubuntu de un proveedor, el controlador nvidia ya está instalado, pero no está activado de forma predeterminada.
Use el siguiente comando para asegurarse de que nvidia esté habilitado.

```
sudo prime-select nvidia
sudo reboot
```

Después de eso, `nvidia-smi` emitirá el estado de la GPU.

1.2. Procedemos a instalar Tensorflow

Recomendamos primero instalar anaconda, que para Linux se realiza de la siguiente manera.

```
wget    https://repo.anaconda.com/archive/Anaconda3-5.2.0-Linux-x86_64.sh
bash Anaconda3-5.2.0-Linux-x86_64.sh
```

Anaconda añadirá automáticamente las rutas al fichero .bashrc (en el fichero de arranque) si escribes "*yes*" justo en este punto de la instalación:

En caso de no funcionar, puedes añadirlas manualmente editando el fichero:

```
$ gedit ~/.bashrc
```

Y luego añadimos la ruta:

```
export PATH="/home/$USER/anaconda3/bin:$PATH"
```

```
conda create -n myenv python=3.6 anaconda
```

Activamos el entorno myenv

```
source activate myenv
```

Y ahora ejecutamos lo siguiente en un terminal

```
conda                          install                          \
 tensorflow-gpu==1.10.0                                         \
 cudatoolkit==9.0                                               \
 cudnn=7.1.2                                                    \
 h5py
```

1.3. Instalación de NCCL

Hay que instalar NCCL para poder hacer que las dos GPUs trabajen de manera compartida.

Seguimos las instrucciones indicadas en el siguiente enlace

```
https://docs.nvidia.com/deeplearning/sdk/nccl-install-guide/index.html#down
```

Elegimos la versión que nos interés de la NVIDIA Collective Communications Library (NCCL) Download Page. En nuestro caso se eligió NCCL v2.4.7, for

CUDA 9.0, May 15, 2019, Local installer for Ubuntu 16.04. No se han encontrado incompatibilidades con Ubuntu 18.04.

2. Creando nuestro propio dataset

2.1. Grabación de los sujetos

Se ha procedido a la grabación de un vídeo de 9 personas con la cámara de un Samsung S9+ durante 1 minuto 30 segundos para cada usuario. La intención es poder tener como mínimo rostros de las personas con una resolución de 512x512 para proceder al entrenamiento.

2.2. OPCIÓN 1: Extracción de frames, detección de rostros, resizing y conversión a png

Nos hemos construido una serie de scripts muy simples que nos permite generar el dataset.

El script *face_detection.sh*:

a. Los vídeos grabados con el móvil se almacenan en el directorio `~/stylegan/resources/videos/`
b. Los frames de los vídeos para cada sujeto se extraen en los directorios correspondientes `~/stylegan/resources/frames/sujeto/` en formato JPG
c. De cada uno de esos directorios se extraen los rostros haciendo llamada a un script en Python *crop.py* que, haciendo uso de OpenCV, tira del modelo haarcascade_frontalface_default.xml. Las caras se almacenan en los directorios correspondientes `~/stylegan/resources/faces/sujeto/` en formato JPG
d. Se hace un resizing de todas las caras detectadas a 512x512 pixels
e. TODO: modificaciones en los canales RGB de las imágenes
f. Nos llevamos todas las caras al directorio `~/stylegan/datasets/custom_dataset/`
g. Transformamos todos los ficheros a formato PNG

2.3. OPCIÓN 2: Extracción de frames y detección de rostros

Se procedió a hacer uso de las rutinas de DeepFaceLab para la extracción de frames y detección de caras. Esta alternativa genera rostros alineados en resolución 256x256. Para cambiar la resolución, se modificó el fichero `~/DeepFaceLab/_internal/bin/DeepFaceLab/mainscripts/Extractor.py` en la línea 400. Aquí se cambia el argumento imagesize de 256 a 512.

2.4. Generación de los TFRecords

Otro script *custom_dataset.sh*, crea los TFRecords necesarios para comenzar el entrenamiento.
Los scripts de entrenamiento y testeo funcionan en conjuntos de datos almacenados como TFRecords de resolución múltiple. Cada conjunto de datos

está representado por un directorio que contiene los mismos datos de imagen en varias resoluciones para permitir una transmisión eficiente. Hay un archivo de .tfrecords separado para cada resolución, y si el conjunto de datos contiene etiquetas, también se almacenan en un archivo separado.

AVISO: la ejecución de estos dos scripts lleva tiempo. Con esto se han obtenido 23,480 caras.

3. Ejecución de entrenamiento

Ejecutamos en `~/stylegan`

```
python train.py
```

comentando todas las líneas de llamada a los datasets e incluyendo:

```
desc = '-custom'; dataset = EasyDict(tfrecord_dir='custom-dataset')
```

Y ya que contamos con dos gpus:

```
desc += '-2gpu'; submit_config.num_gpus = 2; sched.minibatch_base = 8; sched.minibatch_dict = {4: 256, 8: 256, 16: 128, 32: 64, 64: 32, 128: 16, 256: 8}
```

4. Testo del modelo entrenado

El resultado de un entrenamiento se va guardando en la carpeta `~/stylegan /results/XXXXX-custom-2gpu`

Aquí lo que nos interesa es el fichero *network-snapshot-XXXXXX.pkl*, de lo que tiraremos para generar los vídeos e imágenes. Podemos hacer una llamada con la url desde Drive o bajarlo a local y llamarlo en los siguientes scripts (esta última es la opción por la que yo he tirado).

4.1. video.py

Este script genera:

Un video de interpolación estándar, que es simplemente un recorrido aleatorio a través del espacio latente, modificando todas las variables animándolas.

Otro video de "mezcla de estilos"; se genera y mantiene constante una única cara de "fuente"; se genera un video de interpolación secundaria. La cara original se modificará con todo tipo de orientaciones y expresiones faciales, mientras que seguirá siendo reconocible el carácter original.

4.2. circular_interpolation.py

Las interpolaciones circulares son otro tipo interesante de interpolación, que en lugar de recorrer aleatoriamente el espacio latente, con transiciones grandes, trata de moverse alrededor de un punto fijo de alta dimensión haciendo: "Búsqueda binaria para obtener "MSE es aproximadamente igual entre marcos".

4.3. random_samples.py

Genera *n_images* imágenes aleatorias muestreando en el espacio latente de las redes generadoras.

5. Miscelánea

Para mejorar la calidad de las imágenes, existe una librería `https://github.com/nagadomi/waifu2x` que hace uso de Redes Neuronales para duplicar la resolución de las imágenes.

Intenté hacerla correr en el directorio `~/stylegan/resources/`, pero no fue posible, pues piden instalar Torch `http://torch.ch/docs/getting-started.html`

La instalación de la librería está optimizada a Ubuntu 14.04 y a una versión muy anterior de CUDNN, por lo que al instalar cmake `https://anglehit.com/how-to-install-the-latest-version-of-cmake-via-command-line/`, obtenemos un error que no es posible subsanar con la versión de `cudatoolkit==9.0` que se usa.