TF_Hub_generative_image_module

April 24, 2019


```
In [0]: # Copyright 2018 The TensorFlow Hub Authors. All Rights Reserved.
        #
        # Licensed under the Apache License, Version 2.0 (the "License");
        # you may not use this file except in compliance with the License.
        # You may obtain a copy of the License at
        #
        #     http://www.apache.org/licenses/LICENSE-2.0
        #
        # Unless required by applicable law or agreed to in writing, software
        # distributed under the License is distributed on an "AS IS" BASIS,
        # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
        # See the License for the specific language governing permissions and
        # limitations under the License.
        # ==============================================================================
```

1 TF-Hub generative image model

Run in Google Colab

View source on GitHub

This Colab demonstrates use of a TF-Hub module based on a generative adversarial network (GAN). The module maps from N-dimensional vectors, called latent space, to RGB images.

Two examples are provided: * **Mapping** from latent space to images, and * Given a target image, **using gradient descent to find** a latent vector that generates an image similar to the target image.

1.1 Optional prerequisites

- Familiarity with low level Tensorflow concepts.
- Generative Adversarial Network on Wikipedia.
- Paper on Progressive GANs: Progressive Growing of GANs for Improved Quality, Stability, and Variation.

```
In [0]: # Install the latest Tensorflow version.
        !pip -q install --quiet "tensorflow>=1.7"
```

```
# Install TF-Hub.
!pip -q install tensorflow-hub
# Install imageio for creating animations.
!pip -q install imageio
!pip -q install scikit-image
```

In [3]:
```
#@title Imports and function definitions

import imageio
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
import tensorflow_hub as hub
import time

try:
  from google.colab import files
except ImportError:
  pass

from IPython import display
from skimage import transform

# We could retrieve this value from module.get_input_shapes() if we didn't know
# beforehand which module we will be using.
latent_dim = 512


# Interpolates between two vectors that are non-zero and don't both lie on a
# line going through origin. First normalizes v2 to have the same norm as v1.
# Then interpolates between the two vectors on the hypersphere.
def interpolate_hypersphere(v1, v2, num_steps):
  v1_norm = tf.norm(v1)
  v2_norm = tf.norm(v2)
  v2_normalized = v2 * (v1_norm / v2_norm)

  vectors = []
  for step in range(num_steps):
    interpolated = v1 + (v2_normalized - v1) * step / (num_steps - 1)
    interpolated_norm = tf.norm(interpolated)
    interpolated_normalized = interpolated * (v1_norm / interpolated_norm)
    vectors.append(interpolated_normalized)
  return tf.stack(vectors)


# Given a set of images, show an animation.
def animate(images):
  converted_images = np.clip(images * 255, 0, 255).astype(np.uint8)
```

```
    imageio.mimsave('./animation.gif', converted_images)
    with open('./animation.gif','rb') as f:
        display.display(display.Image(data=f.read(), height=300))


# Simple way to display an image.
def display_image(image):
  plt.figure()
  plt.axis("off")
  plt.imshow(image)


# Display multiple images in the same figure.
def display_images(images, captions=None):
  num_horizontally = 5
  f, axes = plt.subplots(
      len(images) // num_horizontally, num_horizontally, figsize=(20, 20))
  for i in range(len(images)):
    axes[i // num_horizontally, i % num_horizontally].axis("off")
    if captions is not None:
      axes[i // num_horizontally, i % num_horizontally].text(0, -3, captions[i])
    axes[i // num_horizontally, i % num_horizontally].imshow(images[i])
  f.tight_layout()

tf.logging.set_verbosity(tf.logging.ERROR)
```

```
WARNING: Logging before flag parsing goes to stderr.
W0424 17:05:25.867466 140172453291904 __init__.py:56] Some hub symbols are not available becaus
```

1.2 Latent space interpolation

1.2.1 Random vectors

Latent space interpolation between two randomly initialized vectors. We will use a TF-Hub module progan-128 that contains a pre-trained Progressive GAN.

```
In [4]: def interpolate_between_vectors():
          with tf.Graph().as_default():
            module = hub.Module("https://tfhub.dev/google/progan-128/1")

            # Change the seed to get different random vectors.
            v1 = tf.random_normal([latent_dim], seed=3)
            v2 = tf.random_normal([latent_dim], seed=1)

            # Creates a tensor with 50 steps of interpolation between v1 and v2.
            vectors = interpolate_hypersphere(v1, v2, 25)

            # Uses module to generate images from the latent space.
```