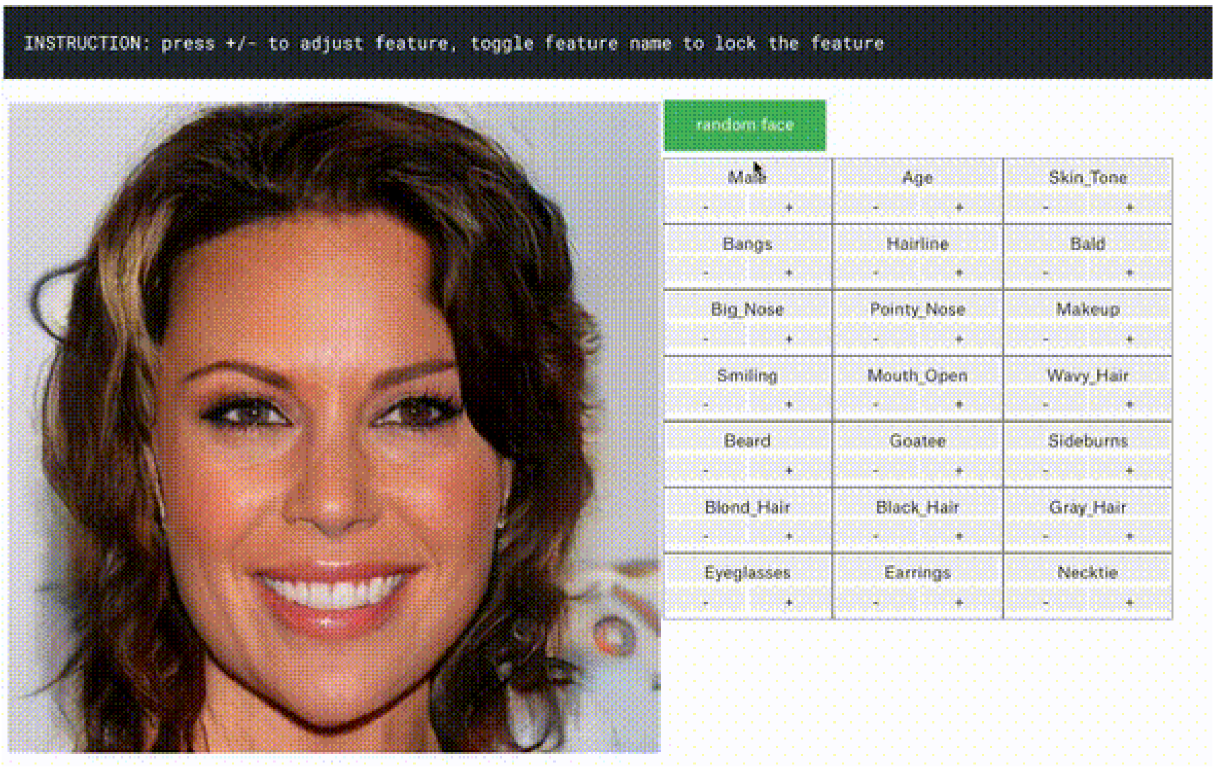
TL-GAN: transparent latent-space GAN

This is the repository of my three-week project: "**Draw as you can tell: controlled image synthesis and edit using TL-GAN**"

Resource lists:

- **Blog post** expaining the motivation, architecture and results is posted on this Medium link
- **Slides presentation** explaining the core ideas of this project are available at this Google Drive link
- A video presentation of this project will be available soon on YouTube
- An **interactive demo** can be found in this Kaggle notebook: https://www.kaggle.com/summitkwan/tl-gan-demo, have fun playing with this model!



A high quaility video of the above GIF on YouTube

Core ideas

- This project provides a novel method to control the generation process of a unsupervisedly-trained generative model like GAN (generative adversarial network).

- GANs can generate random photo-realistic images from random noise vectors in the latent space (see stunning examples of the Nvidia's PG-GAN), but we can no control over the features of the generated images.
- Knowing that the images are determined by the noise vector in the latent space, if we can understand the latent space, we can control our generation process.
- For a already well-trained GAN generator, I made its latent space *transparent* by discovering feature axes in it. When a vector moves along a feature axis in the latent space, the corresponding image morphs along that feature, which enables controlled synthesis and edit.
- This is achieved by leveraging a coupled feature extractor network (a CNN here in this demo, but can be any other CV techniques), which enables us to find correlation between noise vectors and image features.
- Advantages of this method over conditional GAN and AC-GAN:
  - Efficiency: To add a new controller of the generator, you do not have to re-train the GAN model, thus it only takes <1h to add 40 knobs with out methods.
  - Flexibility: You could use different feature extractors trained on different dataset and add knobs to the well-trained GAN

1. Instructions on the online demo

1.1 Why hosting the model on Kaggle I host the demo as a Kaggle notebook instead of a more convenient web app due to cost considerations.

Kaggle generously provides kernels with GPUs for Free! Alternatively, a web app with a backend running on an AWS GPU instance costs ~$600 per month. Thanks to Kaggle that makes it possible for everyone to play with the model without downloading code/data to your local machine!

1.2 To use the demo Open this link from your web browser: https://www.kaggle.com/summitkwan/tl-gan-demo

1. Make sure you have a Kaggle account. If not, please register one (this can be done in seconds by linking to your Google or Facebook account). To have a Kaggle account is actually very rewarding, since allows you to participate numerous data science challenges and join the knowledgeable and friendly community.
2. Fork the current notebook
3. run the notebook by pressing the double right arrow button at the bottom left of the web page. If something does not work right, try to restart the kernel by pressing the circular-arrow button on the bottom right and rerun the notebook

4. Go to the bottom of the notebook and play with the image interactively
5. You are all set, play with the model:
    - Press the "-/+" to control every feature
    - Toggle the name of feature to lock one particular feature. e.g. lock "Male" when playing with "Beard"

2. Instructions on running the code on your machine

Tested on Nvidia K80 GPU with CUDA 9.0, with Anaconda Python 3.6

2.1 Set up the code and environment

1. Clone this repository
2. `cd` to the root directory of the project (the folder containing the `README.md`)
3. Install dependencies by running `pip install -r requirements.txt` in terminal. You can use virtual environment in order not to modify your current python environment.

2.2 Use the trained model on your machine

1. Manually download the pre-trained pg-GAN model (provided by Nvidia), the trained feature extractor network, and the discovered feature axis from my personal dropbox link

2. Decompress the downloaded files and put it in project directory as the following format

```
root(d):
  asset_model(d):
    karras2018iclr-celebahq-1024x1024.pkl   # pretrained GAN from Nvidia
    cnn_face_attr_celeba(d):
      model_20180927_032934.h5              # trained feature extractor network
  asset_results(d):
    pg_gan_celeba_feature_direction_40(d):
      feature_direction_20181002_044444.pkl # feature axes
```

3. Run the interactive demo by first enter interactive python shell from terminal (make sure you are at the project root directory), and then run the commands in python `python exec(open('./src/tl_gan/script_generation_interactive.py').read())`

    Alternatively, you can run the interactive demo from the Jupyter Notebook at `./src/notebooks/tl_gan_ipywidgets_gui.ipynb`

4. A interactive GUI interface will pop up and play with the model

2.3 Instructions on training the model on your own

1. Download celebA dataset `python ./src/ingestion/process_celeba.py celebA`
2. to be continued…

3. Project structure

- **src** : Put all source code for production within structured directory
- **data** : Include example a small amount of data in the Github repository so tests can be run to validate installatio
- **static** : Any images or content to include in the README or web framework if part of the pipeline
- to be continueed