

[← Back to Deep Learning](#)

# Generate Faces

## REVIEW

## CODE REVIEW

## HISTORY

### Meets Specifications

Congratulations on completing the GAN project. You are getting good results. As you might have experienced GANs are advanced and complex topic. In this project a few more changes will get you better results. You have done an awesome job so far. I have mentioned some suggestions to make it improve.

Here some links to know GAN's better:

<https://arxiv.org/pdf/1511.06434.pdf>

<https://blog.openai.com/generative-models/>

[https://www.youtube.com/watch?v=YpdP\\_0-IEOw](https://www.youtube.com/watch?v=YpdP_0-IEOw)

<https://medium.com/@devnag/generative-adversarial-networks-gans-in-50-lines-of-code-pytorch-e81b79659e3f>

<https://deephunt.in/the-gan-zoo-79597dc8c347>

<http://guimperarnau.com/blog/2017/03/Fantastic-GANs-and-where-to-find-them>

Image Completion with Deep Learning in TensorFlow

<http://bamos.github.io/2016/08/09/deep-completion/>

Wasserstein GAN implementation in TensorFlow and Pytorch

<https://wiseodd.github.io/techblog/2017/02/04/wasserstein-gan/>

Stability of GAN

<http://www.araya.org/archives/1183>

Generative Adversarial Networks (GANs) - Computerphile

<https://www.youtube.com/watch?v=Sw9r8CL98N0>

GAN - intro Ian Goodfellow

[https://www.youtube.com/watch?v=YpdP\\_0-IEOw&t=250s](https://www.youtube.com/watch?v=YpdP_0-IEOw&t=250s)

### Required Files and Tests



The project submission contains the project notebook, called "dln\_d\_face\_generation.ipynb".



All the unit tests in project have passed.

## Data Loading and Processing



The function `get_data_loader` should transform image data into resized, Tensor image types and return a `DataLoader` that batches all the training data into an appropriate size.

Nice job resizing the images and creating a `DataLoader`.



Pre-process the images by creating a `scale` function that scales images into a given pixel range. This function should be used later, in the training loop.

Images are scaled correctly. We are scaling the images because we want both set of images ( generated images and real images) in the same scale. The real images are in the range of 0.0 to 1.0 while the generated images are in the range of -1.0 to 1.0 because of the tanh activation in the last layer.

## Build the Adversarial Networks



The Discriminator class is implemented correctly; it outputs one value that will determine whether an image is real or fake.

### Good

- Used a sequence of convolutional layers.
- Used batch normalization starting from second layer.
- Used strided convolution instead of max pooling to downsample making the network to learn its own pooling function.
- Leaky relu and batch norm promote healthy gradient flow.

### Suggestion:

- Use dropout layer to generalize it better. Add dropouts after leaky relu layer following each conv2d layer.



The Generator class is implemented correctly; it outputs an image of the same shape as the processed training data.

## Good:

- Used a series of strided covolutional transpose layers.
- Relu activation paired with batch norm help in smooth flow of gradients.
- Used tanh at the output to return in the range -1 to 1.

## Suggestion:

- Make the generator more powerfull. You can do this by making the generator slightly bigger compared to discriminator. Make it at least one layer bigger. You already have `g_conv_dim` value double the size of discriminator, that also help generator.

Tips and Tricks to make GANs work.

<https://github.com/soumith/ganhacks>



This function should initialize the weights of any convolutional or linear layer with weights taken from a normal distribution with a mean = 0 and standard deviation = 0.02.

Nice job initializing the weights as per the DCGAN paper.

## Optimization Strategy



The loss functions take in the outputs from a discriminator and return the real or fake loss.

Nice job calculating the losses.

## Suggestion

- Try label smoothing on calculating real\_loss. This will prevent discriminator from being too strong by encouraging it to estimate soft probabilities.

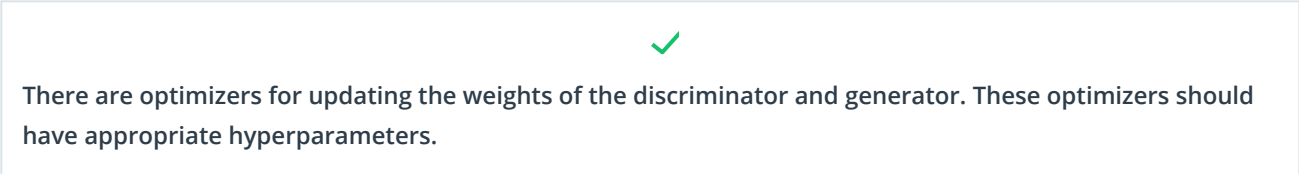
Example:

```
labels = torch.ones(batchsize)*0.9
```

More info:

<http://www.inference.vc/instance-noise-a-trick-for-stabilising-gan-training/>

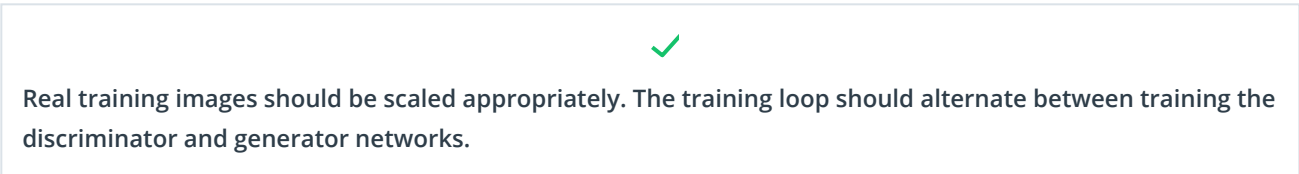
<http://www.inference.vc/instance-noise-druck-for-stabilising-gain-training/>



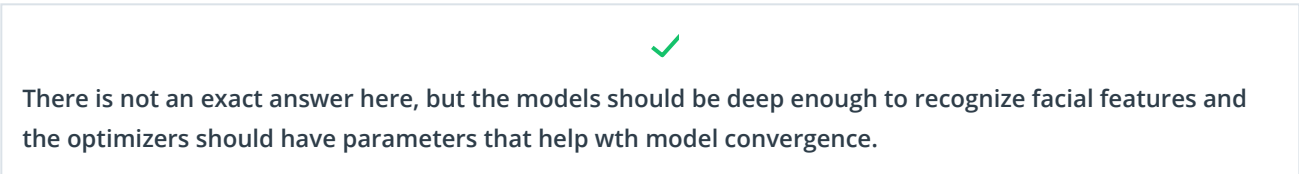
Nice choice of Adam, best optimizer for this scenario. Check this article to know more about optimization in GAN.

<https://towardsdatascience.com/understanding-and-optimizing-gans-going-back-to-first-principles-e5df8835ae18>

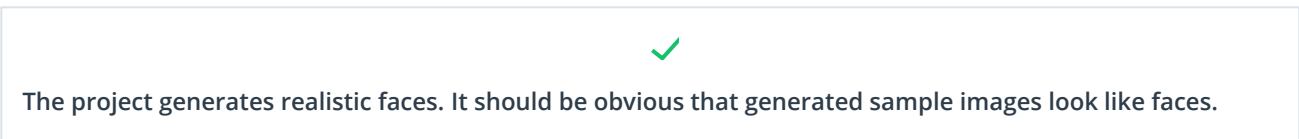
## Training and Results



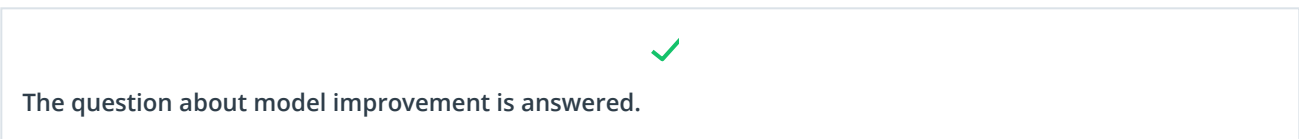
Nice job putting together all the components and make it work.



The hyper parameters chosen are good enough for a stable training, the generator and discriminator doesn't overpower each other.



The generated images look like faces. Well done! Try the suggestions mentioned above (especially label smoothing and bigger generator) for better results.



Good observations on experimenting with different number of epochs. Experimenting with `z_size` with values between 64-256 can create varied faces.

The results most reflecting the dataset. The racial bias is a huge concern in the AI community, check out the news articles below:

<https://www.forbes.com/sites/parmyolson/2018/02/26/artificial-intelligence-ai-bias-google/#6303f72b1a01>

<https://www.nature.com/articles/d41586-018-05707-8>

<https://www.independent.co.uk/life-style/gadgets-and-tech/amazon-ai-sexist-recruitment-tool-algorithm>

<https://www.independent.co.uk/life-style/gadgets-and-tech/amazon-ai-sexist-recruitment-tool-algorithm>

<https://www.independent.co.uk/life-style/gadgets-and-tech/amazon-far-sexist-recruitment-tool-algorithm-a8579161.html>

<https://www.theverge.com/2018/7/26/17615634/amazon-rekognition-aclu-mug-shot-congress-facial-recognition>

 [DOWNLOAD PROJECT](#)

[RETURN TO PATH](#)

Rate this review

