Meets Specifications

Excellent work with the project! ☺

Here are some resources if you wish to explore GANs more.

- This is a very good resource which covers an application of GANs for image completion and it properly explains the project as well. http://bamos.github.io/2016/08/09/deep-completion/
- Ian Goodfellow's Tutorial on GANs for NIPS 2016 https://arxiv.org/pdf/1701.00160.pdf along with this https://channel9.msdn.com/Events/Neural-Information-Processing-Systems-Conference/Neural-Information-Processing-Systems-Conference-NIPS-2016/Generative-Adversarial-Networks
- Here is an awesome list of implementations in Tensorflow on different GAN related algorithms. https://github.com/wiseodd/generative-models/tree/master/GAN

I hope the review helped you. If you feel there's something more that you would have preferred from this review please leave a comment. That would immensely help me to improve feedback for any future reviews I conduct including for further projects. Would appreciate your input too. Thanks!

Congratulations on finishing the project! 😄

Required Files and Tests

The project submission contains the project notebook, called “dlnd_face_generation.ipynb”.

All the unit tests in project have passed.

Data Loading and Processing

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

Good work. You correctly implemented the transformation and resized the image tensors!

Question for you - What other kind of transformations can you apply for this project? Check out the documentation - https://pytorch.org/docs/stable/torchvision/transforms.html and see which one could help with this dataset. Could cropping more help? Or playing around with the brightness? Play around if you'd like to!

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.

Nicely done! Normalizing the images is an important part of preprocessing the dataset for neural networks. Scaling them like this is a part of it. Here is a good resource if you'd like to read more about it - https://becominghuman.ai/image-data-pre-processing-for-neural-networks-498289068258

Build the Adversarial Networks

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

Excellent work on implementing multiple conv layers and appropriately applying the necessary activation functions and using batch normalization.

Think about how filter size affects your model learning features for a CNN. As you increase the layers, your model learns more features, so does a larger filter size make sense or a smaller one to learn more features better?

Also, GANs are difficult to optimize. If the margin by which the discriminator wins is too big, then the generator can’t learn well as the discriminator error would be too small. If the generator wins by too much, it won't learn well

because the discriminator is too weak to teach it. There is a lot of research in GANs as you might know to solve all kinds of such problems. Some recommended ways would be -

- Use a smaller model for the discriminator relative to generator.
- Using dropout in discriminator so that it is less prone to learning the data distribution. [**Note**: I recommend using dropout *after* you have passed the project. Spend some time thinking how you will set the keep_prob value to 1.0 during inference when you add dropout since currently the code isn't set for it. Hence, after passing the project.]
- Think about whether you need a dropout at all since you are using batch normalization?
- I believe this has been referred previously, but do try to read on this - https://arxiv.org/pdf/1606.03498v1.pdf

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

Nicely done!

One small suggestion -

- Strides can play an important factor as well for deconvolution (conv2d_transpose). In your training you would have noticed some checkerboard patterns in your generated images. While that might be a side-effect of working with relatively small images and restrictions on epochs, strides play a part too. I recommend reading the following for more information - https://distill.pub/2016/deconv-checkerboard/

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.

Good work! You correctly initialized the weights of both the layers.

I would encourage you to try out some other types of initializations as well and experiment with them. Like, look up the Xavier Initialization and see if you can implement that and observe your results.

Also, refer to this resource - https://stackoverflow.com/questions/49433936/how-to-initialize-weights-in-pytorch . There is a slightly different way to initialize the weights as well that you can check out if you'd like to.

Optimization Strategy

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

You correctly implemented the loss! Good work especially for using smoothing here!

There are optimizers for updating the weights of the discriminator and generator. These optimizers should have appropriate hyperparameters.

Very well done!

- Do you think different learning rates for generator and discriminator optimizers here would help? Try it out ☺

Training and Results

Real training images should be scaled appropriately. The training loop should alternate between training the discriminator and generator networks.

Nicely done!

- Try to run the optimizer for the generator twice. How do you think that will help?

Here is a good resource for some tips and tricks on training GANs - https://github.com/soumith/ganhacksDo check it out!

There is not an exact answer here, but the models should be deep enough to recognize facial features and the optimizers should have parameters that help wth model convergence.

You selected a good set of hyperparameters.

Some suggestions -

- Try smaller batch sizes
- If you check your loss progression, your loss values jump around a lot and are spiky (as you can also see from your graph). That's usually an indication of high learning rate. So, try with a smaller learning rate and see if you can improve upon that.
- Also, regarding your loss plots. Usually, a sign of increasing loss after decreasing is a sign of overfitting. Do you think that's what's happening here? Because loss values shouldn't really be increasing after a period of time when the model is trained properly. But since you have the discriminator and generator essentially "clashing" here, do you think that makes sense? Or should you train for much fewer epochs to avoid this problem?
- You selected a good value for `beta1`. Here's a good post explaining the importance of beta values and which value might be empirically better. Do check it out! http://sebastianruder.com/optimizing-gradient-descent/index.html#adam Try to tune your values based on this.

Question for you to think about - In your graph, why is there a drop in your generator loss at two instances and at the same time your discriminator loss increases at those same times? What do you think that implies?

The project generates realistic faces. It should be obvious that generated sample images look like faces.

Nicely done!

You are getting some good results when I run your model. Which is awesome.

As you might remember from P2, there's a lot to experiment when it comes to CNNs, so I encourage you to keep expanding on this model of yours 😄

The question about model improvement is answered.

Excellent thought process here! And I like your approach towards experimentation. Very thorough. Keep that up!

You point out how to have more realistic/detailed faces through increasing model depth and epochs. I recommend checking out the "history of GANs" here -

4.5 years of GAN progress on face generation. https://arxiv.org/abs/1406.2661, https://arxiv.org/abs/1511.06434, https://arxiv.org/abs/1606.07536, https://arxiv.org/abs/1710.10196, https://arxiv.org/abs/1812.04948

to see how face generation has improved over the years. That should help you learn more about this!