UDACITY

‹ Return to "Deep Learning" in the classroom

DISCUSS ON STUDENT HUB

Generate Faces

REVIEW

CODE REVIEW 1

HISTORY

Meets Specifications

Your work on this Project shows your hard work and great passion for learning ! 👏
This also shows your commitment towards your goals 👍
Keep learning and i am sure you will graduate and come out with flying colors 🎉

💡 Great article series for learning more about GAN's :

https://medium.com/@jonathan_hui/gan-gan-series-2d279f906e7b
💡 Since GAN's are new and there are various research going on , so you can keep up with by reading more latest research papers about it on
https://arxiv.org/ (in fact this is true for any deep learning topic).

Be safe during this time of period where world is going through suffering , try to remove all distractions from nearby and focus on your goals !
This is the best time to study and learn new skills.

I hope that this review is useful to you and it helps in your learning journey.
Happy learning !

Required Files and Tests

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

Great, your submission included all the necessary files required for review 👍

All the unit tests in project have passed.

Good, all your unit tests passed.
This generally means methods are implemented correctly.

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.

- method transforms image data into tensor type.
- images are resized to 32.
- batch size = 128 is selected.

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.

Correct implementation of transformation formula for scaling in range [-1, 1]. 👍

Build the Adversarial Networks

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

💡 You can also create your helper method to build a sequential wrapper for conv layers like shown below :

helper function for defining convolutional layers

```
def conv(in_channels, out_channels, kernel_size, stride=2, padding=1, batch_norm=True):
    layers = []
    conv_layer = nn.Conv2d(in_channels, out_channels, kernel_size, stride, padding, bias=False)

    # append conv layer
    layers.append(conv_layer)

    # if batch_norm is true, then append it to the conv layers lists
    if batch_norm:
        layers.append(nn.BatchNorm2d(out_channels))

    # return Sequential wrapper container
    return nn.Sequential(*layers)
```

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

💡 Same advice here as above , helper method :

```
def deconv(in_channels, out_channels, kernel_size, stride=2, padding=1, batch_norm=True):

    layers = []
    transpose_conv_layer = nn.ConvTranspose2d(in_channels, out_channels, kernel_size, stride, padding, bias=False)

    # append transposed convolutional layer
    layers.append(transpose_conv_layer)

    # if batch_norm is true, then append it to the conv layers lists
    if batch_norm:
        layers.append(nn.BatchNorm2d(out_channels))

    return nn.Sequential(*layers)
```

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.

- Weights are initialized properly with a mean = 0 and standard deviation = 0.02.

Optimization Strategy

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

- Correct implementation of `real` loss and `fake` loss.

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

- Good work on using `Adam` as optimizers for both discriminator and generator 👍
- 💡 There are some info on using different optimizers here, check this out : https://towardsdatascience.com/understanding-and-optimizing-gans-going-back-to-first-principles-e5df8835ae18

Training and Results

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

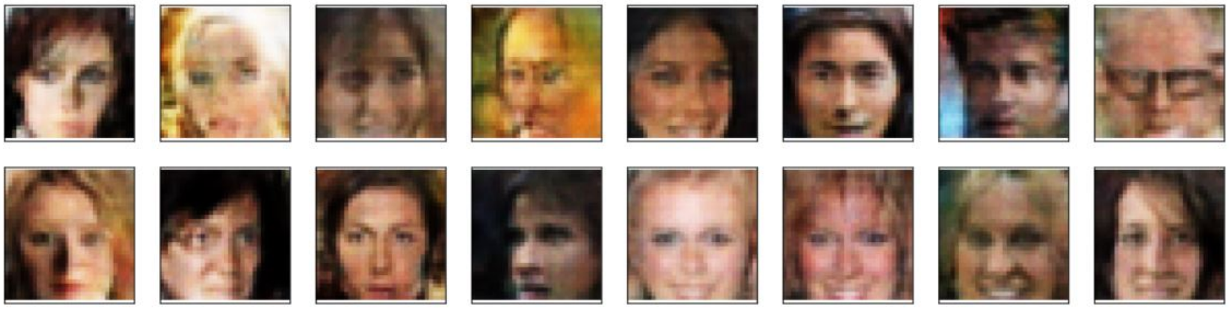
- The training loop consists of alternating both loops.

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.

- Good work.

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

- Generated samples looks like human faces.



The question about model improvement is answered.

Good reasoning.

- Your point about bias is correct.
- We can try to make more deep descriminator networks because generally changing generators do not affect much for GAN's.
- We can try to use different techniques for batch normalization like virtual batch normalization.
- We can try to use some different cost functions which avoids overfitting too much.

DOWNLOAD PROJECT

1 CODE REVIEW COMMENTS ›

RETURN TO PATH

Rate this review