CS446: Machine Learning	**Spring 2018**
Machine Problem 11	
Handed Out: Jan. 16, 2018	*Due: Apr. 26, 2018*

Part 1: Setup

- Remove connect to a EWS machine.

```
ssh (netid)@remlnx.ews.illinois.edu
```

- Load python module, this will also load pip and virtualenv

```
module load python/3.4.3
```

- Reuse the virtual environment from mp0.

```
source ~/cs446sp_2018/bin/activate
```

- Copy mp11 into your svn directory, and change directory to mp11.

```
svn cp https://subversion.ews.illinois.edu/svn/sp18-cs446/_shared/mp11 .
cd mp11
```

- Install the requirements through pip.

```
pip install -r requirements.txt
```

- Prevent svn from checking in the data directory.

```
svn propset svn:ignore MNIST_data .
```

Part 2: Exercise

In this exercise we will implement a GAN to

1. learn the distribution of the MNIST dataset
2. generate images that resemble the original ones

.

In `main_tf.py`, an example pipeline for training is provided for you. Feel free to modify this.

Part 2.1 Implementation

- **Reading in data.** We will use an in-built function from Tensorflow to get the dataset (check `input_data.py`).

- **GAN implementation.** In `models/gan.py`, we will implement the GAN architecture. The model primarily consists of a generator and a discriminator. In our implementation, the discriminator is a binary classifier, with a normalized output close to 1 implying that the image is 'true'. The generator's goal is to fool the discriminator.

  There are several variants of GANs, and new ones keep coming up every single week. We will implement the original GAN from Goodfellow et al, which is presented in the paper titled "**Generative Adversarial Networks**".

  Play around with different architectures, hyper-parameters, etc., to ensure that the images are good, and that there is no mode collapse (the easiest way to check this is to see if your GAN produces all the different digits).

  Do not change the names of the methods, variables, etc. in `models/gan.py` that have already been provided to you.

Part 3: Writing Tests

In `test.py` we have provided basic test-cases. Feel free to write more. To test the code, run

```
nose2
```

Part 4: Submit

Submitting the code is equivalent to committing the code. This can be done with the follow command:

```
svn commit -m "Some meaningful comment here."
```

Lastly, double check on your browser that you can see your code at

```
https://subversion.ews.illinois.edu/svn/sp2018-cs446/(netid)/mp11/
```

Note: The assignment will be autograded. It is important that you do not use additional libraries, or change the provided functions input and output.