Bidirectional Generative Adversarial Networks

Author: Alireza Moradzadeh
ECE 598
Instructor: Prof. Pramod Viswanath

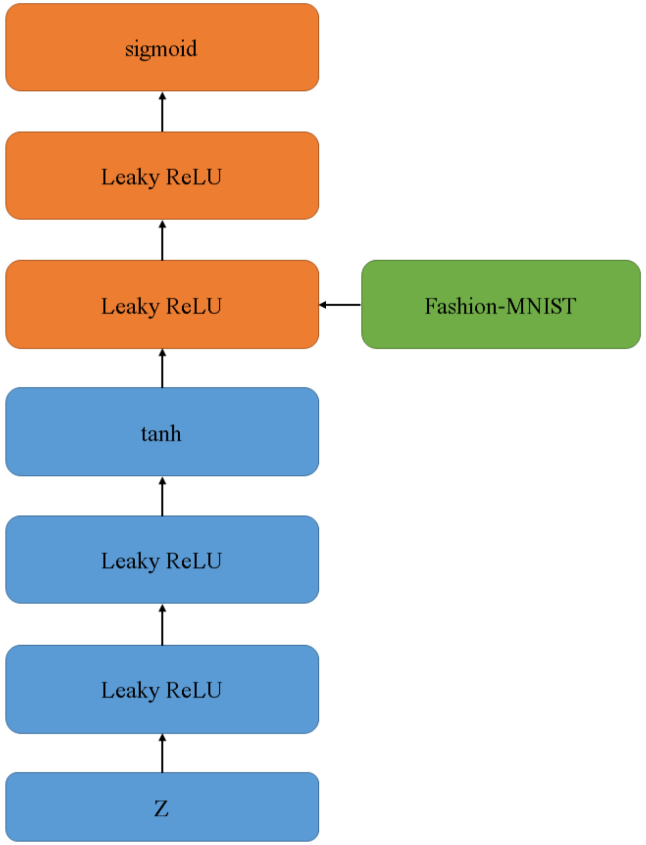
Outline

- Introduction to GAN and BiGAN
- GAN results on MNIST and Fashion-MNIST
- Future Work
- Conclusion

Generative Adversarial Neural Network (GAN)

- Theory

$$\min \max V(D, G) = E_{x \sim p_{data(x)}}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

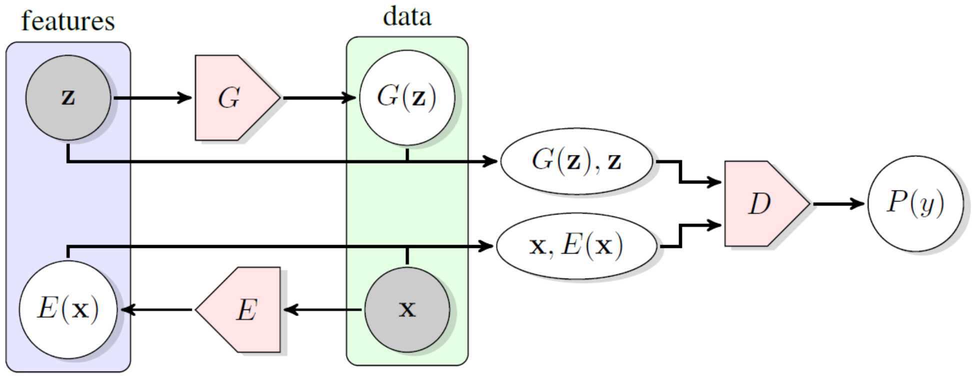


Bidirectional Generative Adversarial Networks

- BiGAN:

1. Generator network (***G***)
2. Encoder network (***E***)
3. Objective: find a mapping from latent space to data space and one from latent space to data space
4. Discriminator: (x, E(x)) vs (G(z), z)
5. Related to autoencoder with ℓ_0 loss function

- Theory



$$\min_{E,G} \max_{D} V(D,G,E) = E_{x \sim p_{data(x)}}\left[\log D\big(x, E(x)\big)\right] + E_{z \sim p_z(z)}[\log(1 - D(G(z), z)]$$

Donahue, J., Krähenbühl, P. and Darrell, T., 2016. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*.

Optimal Encoder and Generator

$$x = G\big(E(x)\big), \qquad z = E\big(G(z)\big)$$

Ideal Case

$$E = G^{-1}$$

Optimal D for any E and G is the Randon-Nikodym derivative

$f_{EG}\colon \dfrac{\mathrm{d}P_{EX}}{\mathrm{d}(P_{EX}+P_{GZ})}$

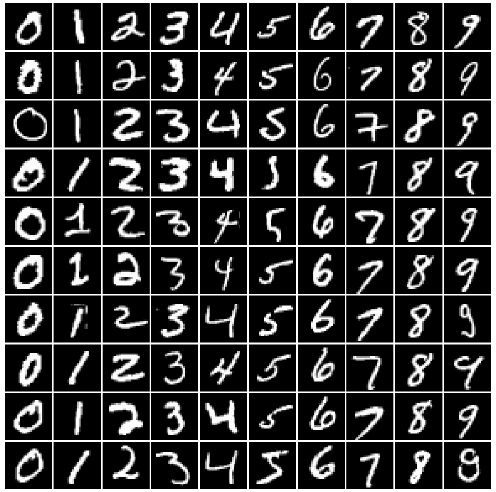
Implementation

- Tensorflow by Google

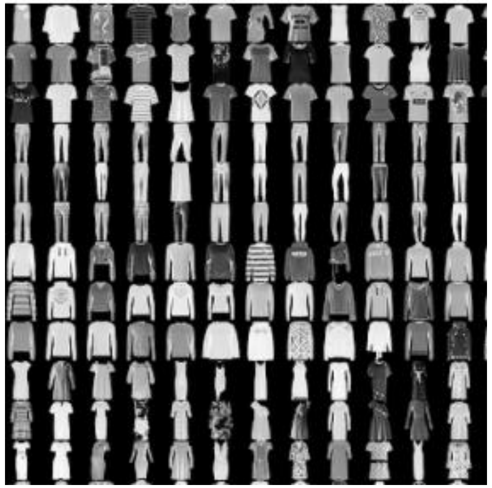
TensorFlow™

Two different datasets:

1. MNIST



2. Fashion-MNIST



GAN Architecture

- Generator

```
def generator(z, out_dim, n_units=128, reuse=False, alpha=0.01):
    with tf.variable_scope('generator', reuse=reuse):
        h1 = tf.layers.dense(z, n_units, activation=None)
        h1 = tf.maximum(alpha * h1, h1)
        logits = tf.layers.dense(h1, out_dim, activation=None)
        out = tf.tanh(logits)

        return out
```

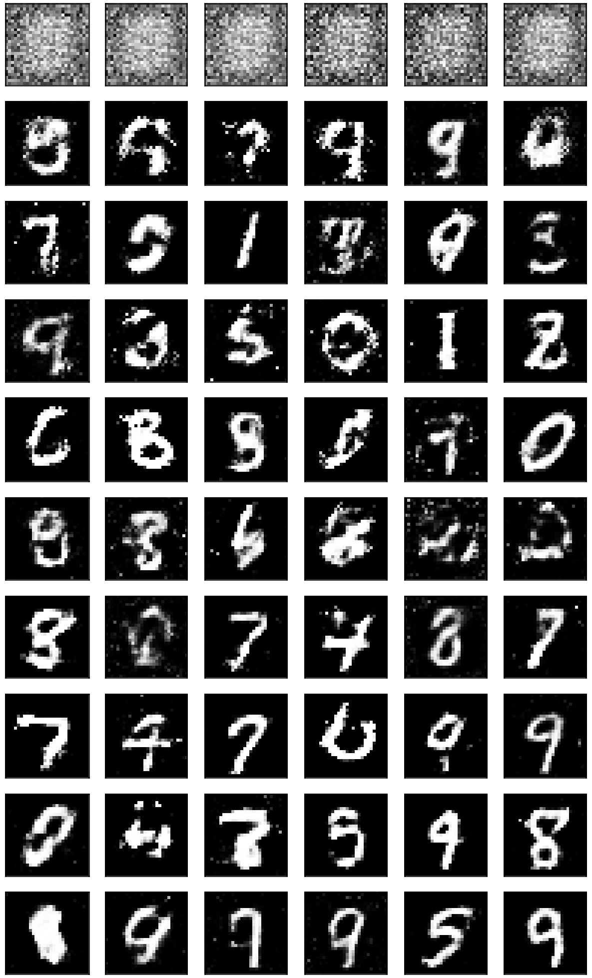
- Discriminator

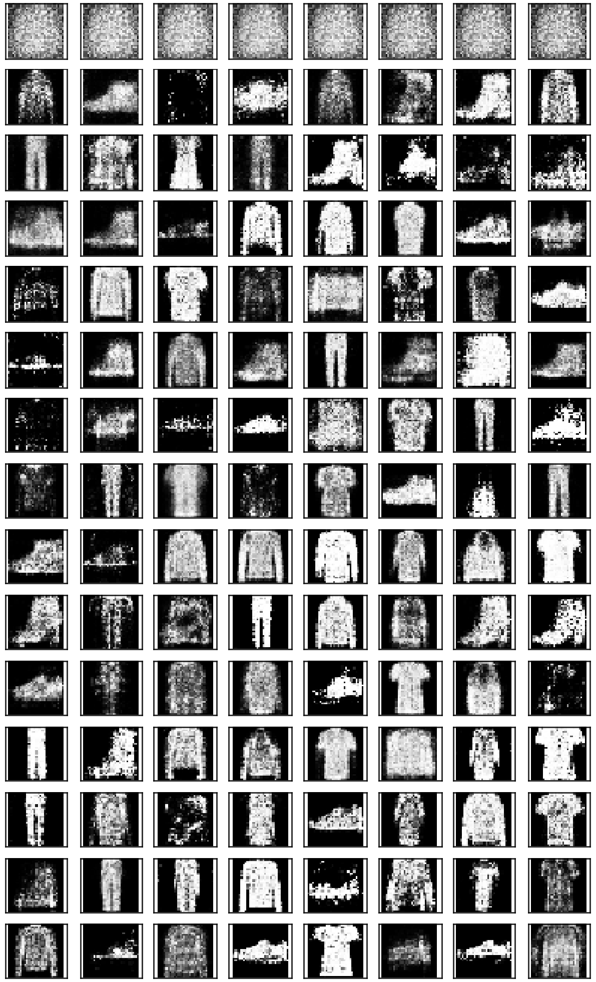
```
def discriminator(x, n_units=128, reuse=False, alpha=0.01):
    with tf.variable_scope('discriminator', reuse=reuse):
        h1 = tf.layers.dense(x, n_units, activation=None)
        h1 = tf.maximum(alpha * h1, h1)
        logits = tf.layers.dense(h1, 1, activation=None)
        out = tf.sigmoid(logits)

        return out, logits
```

GAN Results on MNIST and Fashion-MNIST

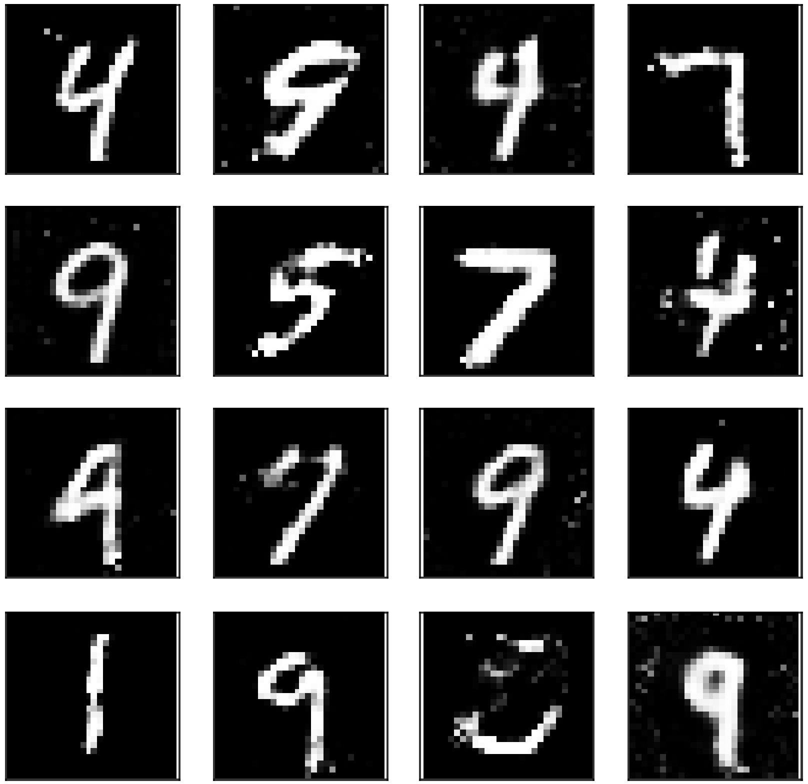
- During Training

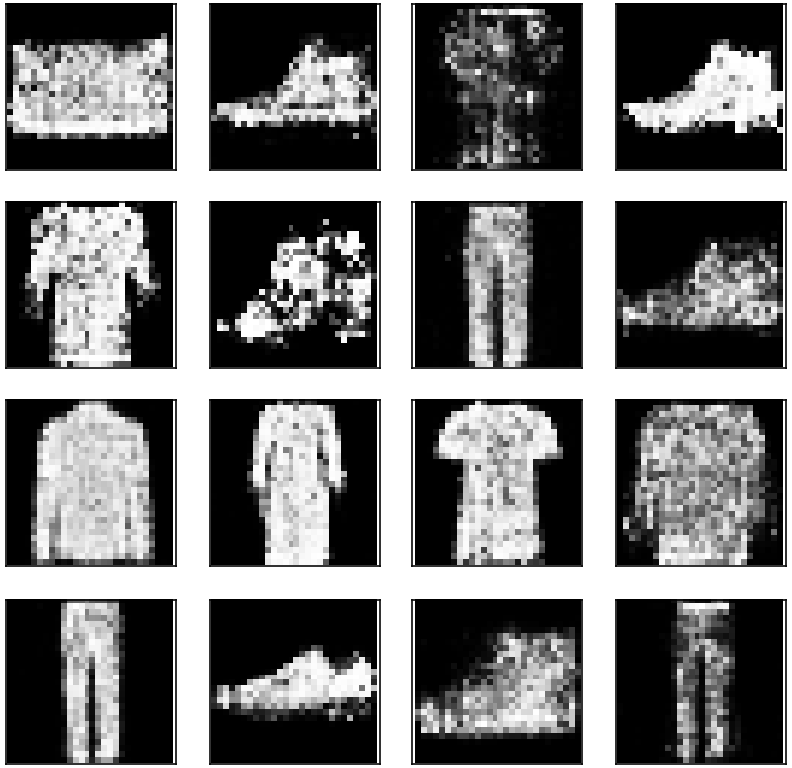




GAN Results on MNIST and Fashion-MNIST

- After Training





BiGAN Architecture

- Encoder

```
def encoder(x, out_dim, is_training, n_units=128, reuse=False, alpha=0.01):
    with tf.variable_scope('encoder', reuse=reuse):
        h1 = tf.layers.dense(x, n_units, activation=None)
        h1 = tf.layers.batch_normalization(h1, training=is_training)
        h1 = tf.maximum(alpha * h1, h1)
        logits = tf.layers.dense(h1, out_dim, activation=None)
        out = tf.tanh(logits)

        return out
```

- Generator and Discriminator same as GAN

Future Work

- Solve Issue with BiGAN
- BiGAN for Fashion-MNIST and MNIST
- Comparison between GAN and BiGAN
- Calculating accuracy of BiGAN *i.e.* x ? $G\big(E(x)\big)$ and z? $E(G(z))$

Conclusion

- GAN network implemented in TF
- GAN is trained on Fashion-MNIST and MNIST
- BiGAN Network is theoretically explained
- BiGAN implemented but still problem with training

Thank You
Any Question?