GAN Application

Random Number Generator

https://youtu.be/3N_otcqpFlM

GANs

- Generative Adversarial Networks

- Two Models
  - Generator
  - Discriminator

GANs Example Code

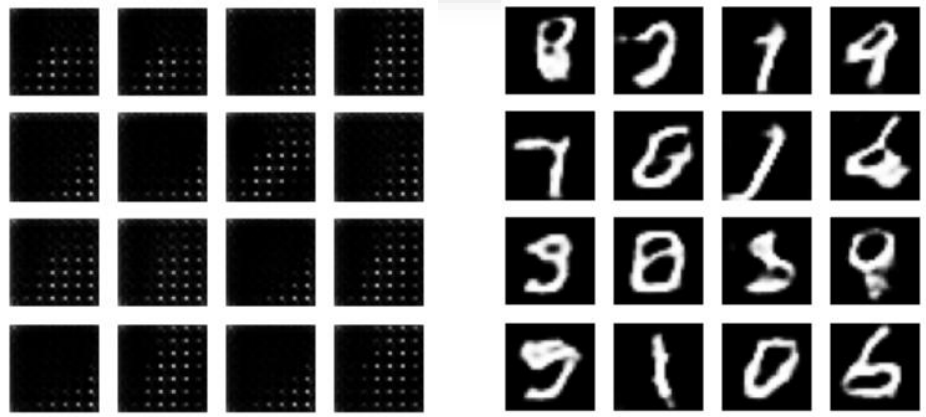
https://www.tensorflow.org/tutorials/generative/dcgan?

- Using MNIST Data

```python
def discriminator_loss(real_output, fake_output):
    real_loss = cross_entropy(tf.ones_like(real_output), real_output)
    fake_loss = cross_entropy(tf.zeros_like(fake_output), fake_output)
    total_loss = real_loss + fake_loss
    return total_loss
```

```python
def generator_loss(fake_output):
    return cross_entropy(tf.ones_like(fake_output), fake_output)
```

```python
@tf.function
def train_step(images):
    noise = tf.random.normal([BATCH_SIZE, noise_dim])

    with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
      generated_images = generator(noise, training=True)

      real_output = discriminator(images, training=True)
      fake_output = discriminator(generated_images, training=True)

      gen_loss = generator_loss(fake_output)
      disc_loss = discriminator_loss(real_output, fake_output)

    gradients_of_generator = gen_tape.gradient(gen_loss, generator.trainable_variables)
    gradients_of_discriminator = disc_tape.gradient(disc_loss, discriminator.trainable_variables)

    generator_optimizer.apply_gradients(zip(gradients_of_generator, generator.trainable_variables))
    discriminator_optimizer.apply_gradients(zip(gradients_of_discriminator, discriminator.trainable_variables))
```



RNG

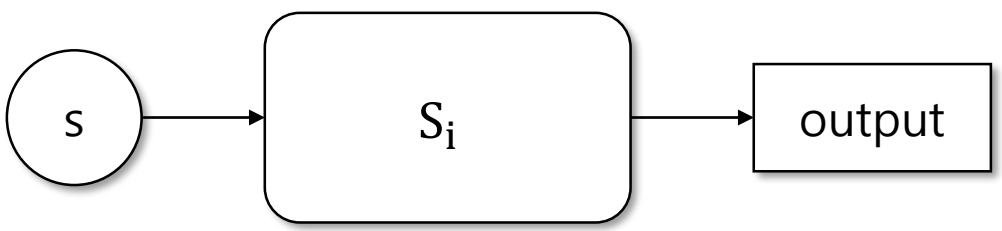
- Random Number Generator

- Two RNGs
  - TRNG(True RNG)
  - PRNG(Pseudo RNG)
    - Seed s

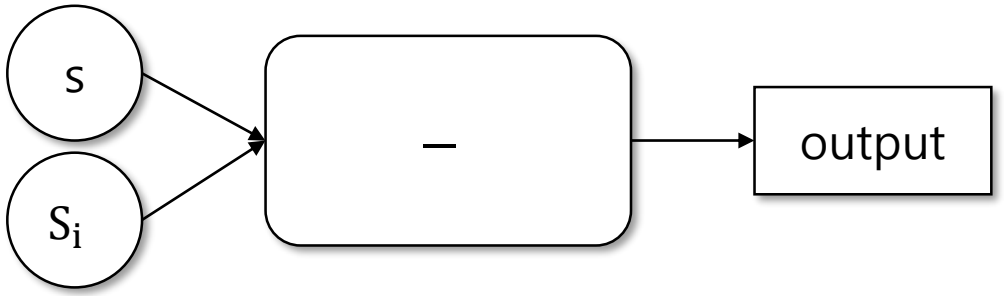
Design

$$prng(s) : \mathbb{B} \rightarrow \mathbb{B}^n$$

$$prng^{\nabla}(s, S_i) : X \rightarrow \mathbb{B}$$

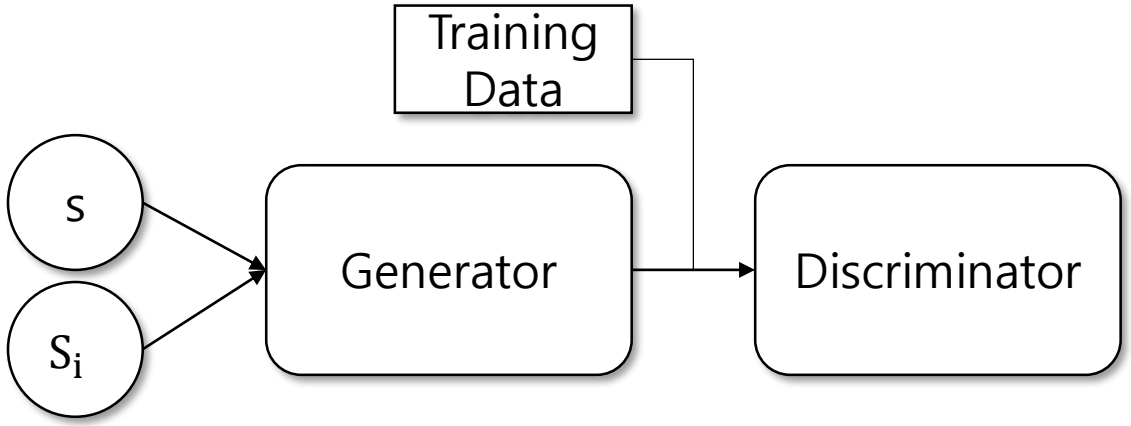


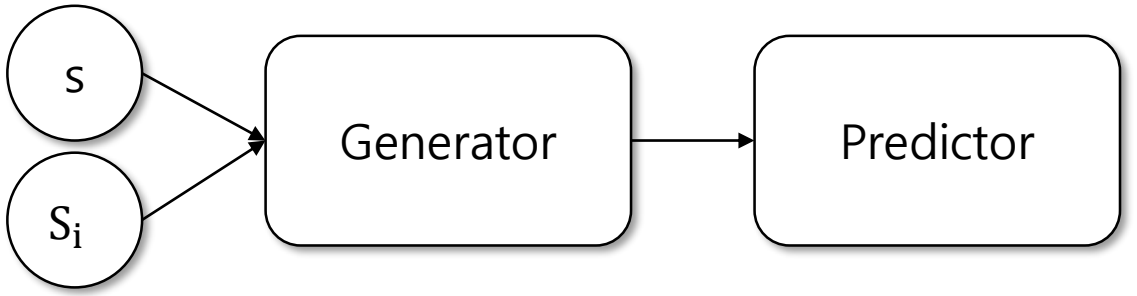
$$G^{\nabla}(s, \boldsymbol{o}_t) : \mathbb{B}^{t+1} \rightarrow \mathbb{B}^n$$



Design

- Two Models
  - Generator
  - Discriminator

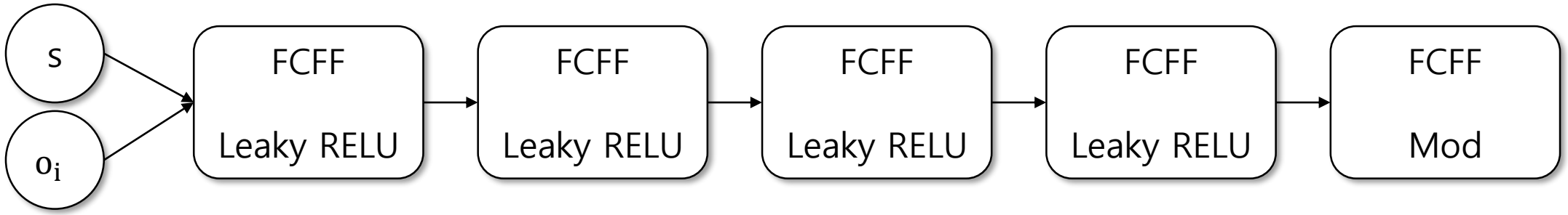




Design G

- Two Models
  - Generator
  - Discriminator

$$G^{\nabla}(s, o_1) : \mathbb{B}^2 \rightarrow \mathbb{B}^8$$



Loss = absolute difference loss

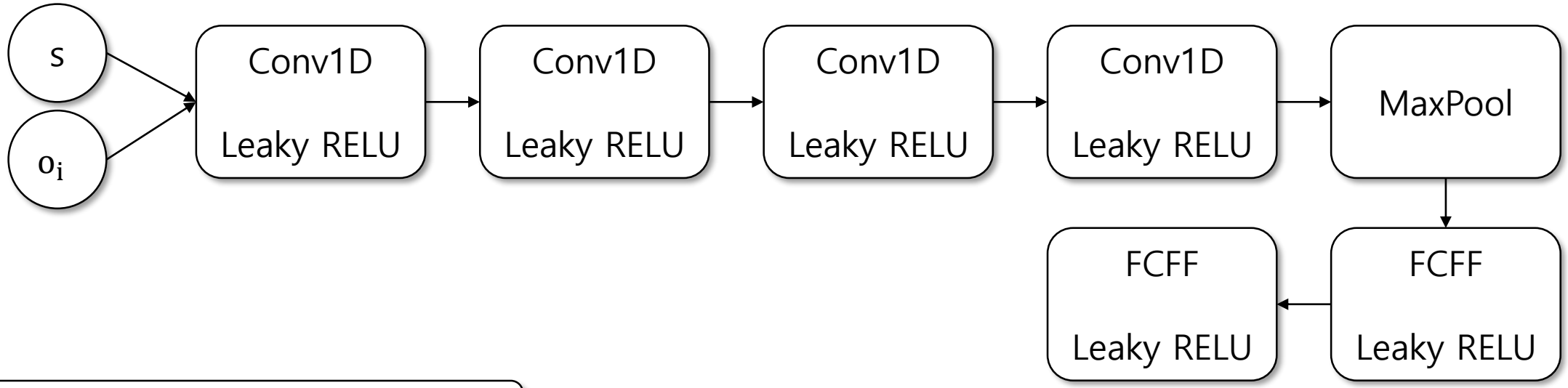
Optimizer = Adam

https://medium.com/@himanshuxd/activation-functions-sigmoid-relu-leaky-relu-and-softmax-basics-for-neural-networks-and-deep-8d9c70eed91e

Design D or P

- Two Models
  - Generator
  - Discriminator

$$D(\boldsymbol{r}) : \mathbb{B}^8 \rightarrow [0, 1] \qquad P(\boldsymbol{r}_{split}) : \mathbb{B}^7 \rightarrow \mathbb{B}$$



Loss = absolute difference loss

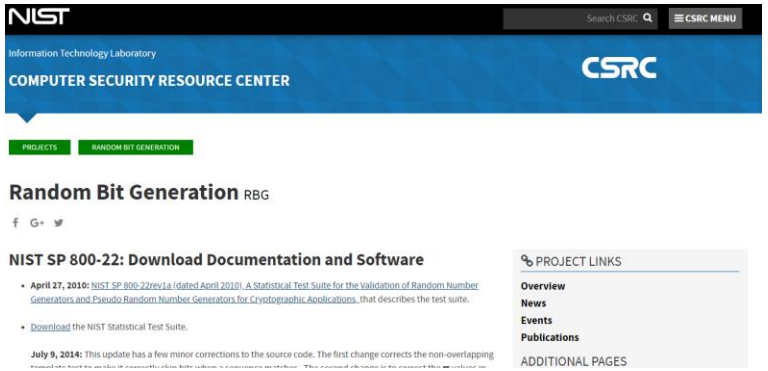
Optimizer = Adam

Experimental Procedure

1. Generate data with untrained Generator → **Data 1**
2. Generate data with trained Generator → **Data 2**
3. Compare **Data 1** and **Data 2** using NIST test suite

NIST test suite

188 tests,
Each repeated 10 times
1,000,000 input bits for each repetition



https://csrc.nist.gov/projects/random-bit-generation/documentation-and-software

Evaluation

i	T	$\langle T_I \rangle$	$\langle F_I \rangle$	$\langle F_{I\%} \rangle / \%$	$\langle F_p \rangle$	$\langle F_T \rangle$	$\langle F_\% \rangle / \%$
D_{before}	188	1800	1796	99.8	188	188	100.0
D_{after}	188	1800	61	3.5	4.3	6.9	3.9
P_{before}	188	1800	1798	99.9	188	188	100.0
P_{after}	188	1830	56	3.0	2.7	4.5	2.5

1. Best pass rate was around 98%.
2. Non-cryptographic PRNGs.