Generative Adversarial Network
Machine Learning (COCSC17)

Faculty: DR. Poonam Rani
Submitted by:
Vidhan Vyas (2021UCS1723)
Akshay R. Nair (2021UCS1678)
Bihaan (2021UCS1708)
Chandan Kumar (2020UCO1707)
Ansh (2021UCS1688)
Batch: CSE III

Introduction

A Generative Adversarial Network (GAN) is a machine learning framework that generates data mimicking the REAL dataset. Ian Goodfellow and his team introduced GANs in 2014. A GAN consists of two neural networks: a *generator* and a *discriminator*. These networks compete against each other constantly to produce optimized output.

The *generator* makes fake data by changing random noise into a sample that looks like real data. The *discriminator* checks the samples telling apart real data (from the dataset) and fake data (from the generator). The aim for a *generator* is to create data so similar to the real dataset, so as to confuse the *discriminator* in identifying the real or fake out of the lot.

Both networks train at the same time: the *discriminator* learns to sort real and fake data , while the *generator* learns to make data that tricks the *discriminator*. This process keeps repeating until the *generator* is able to create extremely realistic data. This makes GANs really good at tasks like test-to-image generator, enhancing image quality. GANs have changed how we make generative models by giving us a way to produce high-quality realistic outputs in many different areas.

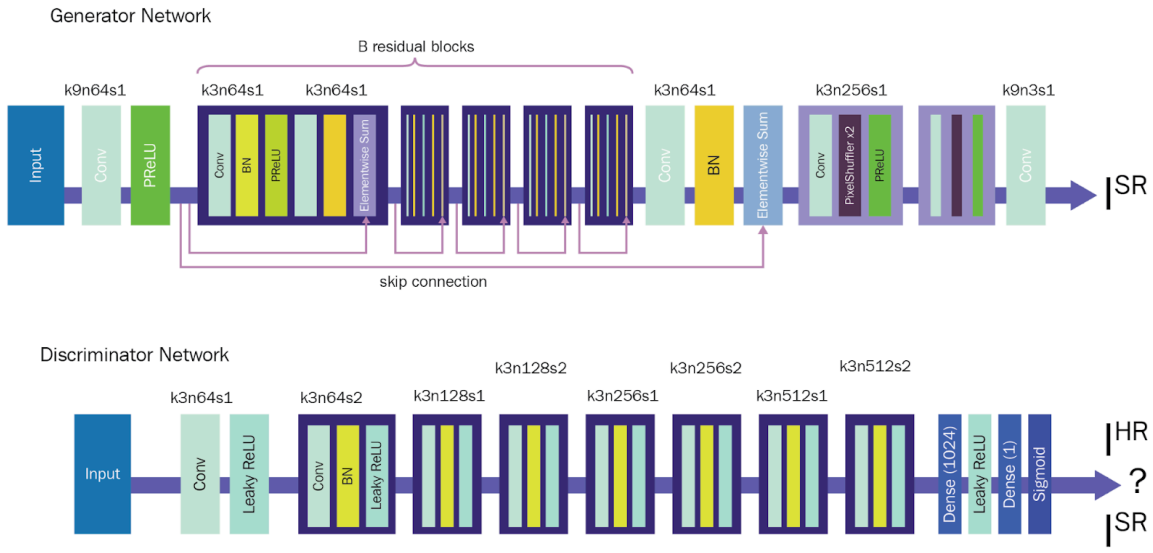
Super-Resolution

Super-Resolution (SR) refers to the process of enhancing the image resolution, i.e increasing pixel density while preserving the original detailing of the image. It is crucial in various applications of healthcare, satellite image processing and high-definition displays.

The challenge lies in accurately reconstructing high-frequency details (e.g., textures, edges) that are lost in the low-resolution image.

Generative Adversarial Network is effective for Super-Resolution because of the components of GAN .The generator learns to produce images that are indistinguishable from the actual high-resolution images, while the discriminator improves by better identifying the generated fakes.

Architecture of GAN used for SR (SRGAN)



Architecture of Generator and Discriminator Network with corresponding kernel size (k), number of feature maps (n) and stride (s) indicated for each evolutional layer.

Generator Network:

The Generator's job is to start with random noise and try to turn it into something that looks just like real data. It's like an artist who begins with a blank canvas and adds details little by little, aiming to create something indistinguishable from the real thing. Each time the Generator creates a sample, the Discriminator either "accepts" it as realistic or "rejects"

it, giving feedback on how convincing the result is. With each round of feedback, the Generator learns what works and what doesn't, refining its craft to become better at mimicking real data.

Residual Blocks**:** These are helpful in learning identity mappings, which is crucial for maintaining the content of the input image while enhancing resolution.

Upsampling Layers: These layers increase the spatial resolution of the image. Various techniques like pixel shuffle or transposed convolution are used for upsampling.

Discriminator Network:

The Discriminator's role is like a skilled critic who examines each piece of data to decide if it's the real thing or a clever imitation. It carefully studies both genuine and generated samples, learning to spot the subtle differences that make real data authentic. With each evaluation, the Discriminator sharpens its ability to tell real from fake, giving the Generator tips on how to improve its work. This ongoing feedback helps the Generator get better at its job, while the Discriminator becomes an even sharper judge. Together, they push each other to improve, making the whole model stronger.

Convolutional Layers: These layers extract features from both the generated and real images to classify them.

Loss in SRGAN

To further improve the performance of the generator and discriminator network to enhance the quality of the super-resolved image, the loss functions used:

- Perceptual Loss: Helps in ensuring that the generated images have similar high-level features as the original images, which are perceptually important to humans.

- Adversarial loss: Derived from the discriminator, this loss encourages the generator to produce more realistic images.
- Content Loss: Measures the difference in the generated image and the original image, often using features extracted from pre-trained networks.

Advantages of using GAN

High-Frequency Detailing : GANs are particularly skilled at producing high-frequency details—essentially the finer textures, edges, and intricate features that make images appear sharp and realistic. Traditional image generation methods may often produce blurred or less detailed outputs, especially at high resolutions, but GANs thrive in creating images with crisp and natural-looking detail. This capability is essential for applications where quality visuals are critical, such as super-resolution imaging, facial generation, and medical imaging.

Natural-Looking Images : GANs, thanks to their adversarial training setup, can generate images that closely resemble real ones. By training the Generator to “fool” the Discriminator into thinking generated data is real, GANs effectively create outputs that blend seamlessly with actual data. This trait is particularly valuable in creative industries, like content generation, where realistic visuals are necessary. Applications such as style transfer, image synthesis, and even artistic creations benefit from this natural-looking quality.

Better Visual Quality : Compared to many conventional methods, GANs often outperform in perceptual quality. GAN-generated images typically avoid issues like pixelation, visible interpolation artifacts, or lack of detail. This high visual fidelity is crucial for applications in fields like entertainment, virtual reality, and high-definition displays. By training on a dataset of real images, GANs learn to capture the nuances of realistic visuals, creating smoother transitions, better color consistency, and overall more polished results.

Disadvantages of using GAN

Training Instability: Training GANs can be complex and unpredictable. The adversarial training process, where the Generator and Discriminator "compete" against each other, can lead to instability. If the Generator learns too quickly, it might start consistently fooling the Discriminator early on, making further training ineffective. On the other hand, if the Discriminator becomes too good too fast, the Generator might struggle to improve, leading to stalled or unproductive training cycles. This need for balance between the two networks requires careful monitoring and tuning.

Mode Collapse: Mode collapse is a common problem in GANs where the Generator begins producing a limited variety of outputs, essentially "collapsing" into one or a few types of samples. Instead of creating diverse images, the Generator might focus on a narrow range of data that successfully fools the Discriminator, but lacks variation. This issue limits GANs' ability to produce diverse outputs, which can be problematic in applications needing a wide range of realistic images, like data augmentation or open-domain image generation.

Sensitivity to Hyperparameters: GANs are highly sensitive to the configuration of hyperparameters, such as learning rates, batch size, and network architecture choices. Fine-tuning these parameters is essential for stable training and high-quality outputs, but it can be time-consuming and technically demanding. Adjustments to one parameter can affect others, so achieving optimal performance often requires extensive experimentation. This sensitivity makes GANs challenging for less experienced practitioners and increases computational costs.

Applications of GAN

Image Generation and Editing

Generating Realistic Images: GANs can create high-quality, photorealistic images from scratch. This ability has led to applications in content creation, where GANs can produce

images for various purposes, such as stock photos, design inspiration, or even avatars in video games.

Deepfakes: GANs are used to create deepfakes, where a person's face or voice can be convincingly recreated or swapped onto someone else. While deepfakes have been a source of controversy due to privacy and misinformation risks, they also demonstrate GANs' impressive ability to mimic real-world visuals and audio.

Photo Editing and Restoration: GANs can enhance images by filling in missing parts (image inpainting), colorizing black-and-white images, and increasing image resolution (super-resolution). These capabilities are especially useful for restoring old photographs and improving the quality of low-resolution images.

Data Augmentation for Training Models

Improving Model Performance: GANs are commonly used to generate synthetic data for training machine learning models, especially in cases where labeled data is scarce or costly to obtain. By generating variations of images or data points, GANs help diversify training datasets, which can improve model robustness and generalization.

Balancing Datasets: GANs can be used to create examples for underrepresented classes in a dataset, addressing issues of class imbalance. For instance, in medical datasets, it can be challenging to gather examples of rare conditions; GANs can generate synthetic images or data points to help balance the dataset and improve classification accuracy.

Art and Creativity

Digital Art Generation: GANs have opened new frontiers in digital art by autonomously creating artworks that mimic various artistic styles. Artists and designers can use GANs to generate abstract art or reinterpret classical styles, blending human creativity with machine capabilities.

Interactive Creative Tools: GANs are being integrated into tools that allow users to create art with minimal effort. For instance, tools like Artbreeder allow users to blend images or create unique artwork by adjusting features, leveraging GAN technology to assist in art and creative design.

Virtual and Augmented Reality: GANs are used to generate realistic textures and landscapes for VR and AR applications. By creating high-quality virtual environments, GANs make digital experiences more immersive, opening up possibilities in entertainment, education, and virtual tourism.

Medical Imaging and Healthcare

Data Augmentation in Medical Imaging: Medical images such as MRI or CT scans are often limited in availability. GANs can augment datasets by generating synthetic medical images, which help train diagnostic models, especially in fields like radiology or pathology where large amounts of data are essential for accurate diagnosis.

Anomaly Detection: GANs can be trained on healthy patient data to identify patterns associated with normal conditions. Then, by comparing new images to these learned patterns, GANs help detect anomalies or abnormalities, such as tumors or other medical conditions, aiding in early diagnosis.

Image Reconstruction and Enhancement: GANs can improve the quality of medical images by reconstructing details lost due to noise or low resolution. This is particularly useful for enhancing MRI or ultrasound images, where clarity is essential for effective diagnosis and treatment planning.

Video Prediction and Synthesis

Predicting Future Frames: GANs can be trained to predict the next frames in a video sequence, which is useful for applications like autonomous driving, where the vehicle

needs to anticipate the movement of objects in real-time. This predictive ability enables better decision-making by helping models understand potential scenarios.

Synthetic Video Creation: GANs can generate realistic video sequences for applications such as video games or animated films, where synthetic environments are created for storytelling or gameplay. GANs help produce realistic movements and interactions, enhancing user experience and immersion.

Motion Transfer and Style Adaptation: In video editing, GANs can transfer motions from one video to another, allowing actions to be replicated or animated in a different style. This application is useful in animation, special effects, and even personalized virtual assistants.

Mathematics Behind GANs

Loss Functions: Binary Cross-Entropy and Minimax Game

- Binary Cross-Entropy Loss: The discriminator network is trained to distinguish real data from fake data created by the generator. To achieve this, the binary cross-entropy loss function is used, as it measures how well the discriminator classifies inputs as real or fake.
    - The discriminator's objective is to maximize the probability of assigning the correct labels (real/fake) to its inputs.
    - The generator's objective is to "fool" the discriminator into classifying its generated samples as real.
- Minimax Game: The overall GAN setup is a minimax (or adversarial) game between the generator GGG and discriminator DDD. The goal is to reach a Nash equilibrium where both networks perform optimally against each other.
    - The generator minimizes log(1−D(G(z)))\log(1 - D(G(z)))log(1−D(G(z))), effectively trying to maximize D(G(z))D(G(z))D(G(z)), so the discriminator believes the generated data is real.

Training Techniques: Gradient Descent and Backpropagation

- Gradient Descent: GANs are typically trained using stochastic gradient descent (SGD) or its variants (e.g., Adam optimizer). Gradient descent helps in updating the weights of the generator and discriminator networks to reduce their respective losses.
    - For the generator GGG, the gradients are calculated to improve its ability to produce realistic data, essentially minimizing log(1−D(G(z)))\log(1 - D(G(z)))log(1−D(G(z))).
    - For the discriminator DDD, the gradients help improve its ability to distinguish between real and generated data, maximizing logD(x)\log D(x)logD(x) for real data and log(1−D(G(z)))\log(1 - D(G(z)))log(1−D(G(z))) for fake data.
- Backpropagation: Backpropagation is used to compute gradients for both networks during training.
    - For the discriminator, backpropagation calculates the gradient of the loss with respect to the discriminator's parameters, updating them to better classify real versus fake images.
    - For the generator, backpropagation is more challenging because the gradients flow back from the discriminator’s output. The generator's loss depends on how well it fools the discriminator, so the generator’s parameters are updated to produce data that the discriminator misclassifies as real.

Future Directions

Improved Stability and Training Techniques

- Advanced Architectures: Future research is likely to explore alternative GAN architectures and techniques to address training instability, such as spectral normalization and hybrid loss functions.
- Alternative Loss Functions: Techniques like Wasserstein loss in WGANs have improved stability, but researchers are exploring other loss functions and optimization methods to further enhance GAN training.

Reducing Mode Collapse

- Diversity-Promoting Mechanisms: Innovations like Minibatch Discrimination and unrolled GANs aim to prevent mode collapse, where GANs generate limited output diversity. New methods may incorporate adaptive mechanisms that balance diversity and output quality more effectively.
- Self-Supervision and Semi-Supervised Learning: Using self-supervised and semi-supervised learning approaches can help GANs learn broader data distributions, which is essential for generating diverse outputs and reducing mode collapse.

Expanding GAN Applications

- Cross-Disciplinary Applications: GANs are expanding into areas like genomics, physics simulations, and material science, where they can generate molecular structures or simulate scientific data.
- Robustness in Medical Imaging: GANs are expected to play an even bigger role in medical imaging, enhancing diagnosis, data augmentation, and anomaly detection. Improved GAN techniques could lead to highly reliable medical applications.

- 3D Object and Scene Generation: For virtual and augmented reality applications, future GANs could enable high-quality, realistic 3D object and scene generation, enhancing VR/AR experiences and gaming.

Ethical and Regulatory Development

- Deepfake Detection and Mitigation: With the rise of deepfakes, GAN research is likely to emphasize detection techniques and ethical guidelines, especially for the responsible use of GANs in media and entertainment.
- Data Privacy and Fairness: GANs trained on sensitive data need to ensure privacy protection and fairness, reducing potential biases. Methods like differential privacy in GANs are being developed to address this.

Integration with Other ML Models

- GANs and Reinforcement Learning (RL): Integrating GANs with reinforcement learning could improve video prediction, data synthesis, and simulation environments. This integration may lead to GANs that generate more realistic and contextually accurate scenarios.

- Quantum GANs (QGANs): Quantum computing is an emerging field where Quantum GANs (QGANs) could leverage quantum capabilities to increase the efficiency of training and potentially solve complex, high-dimensional problems faster.

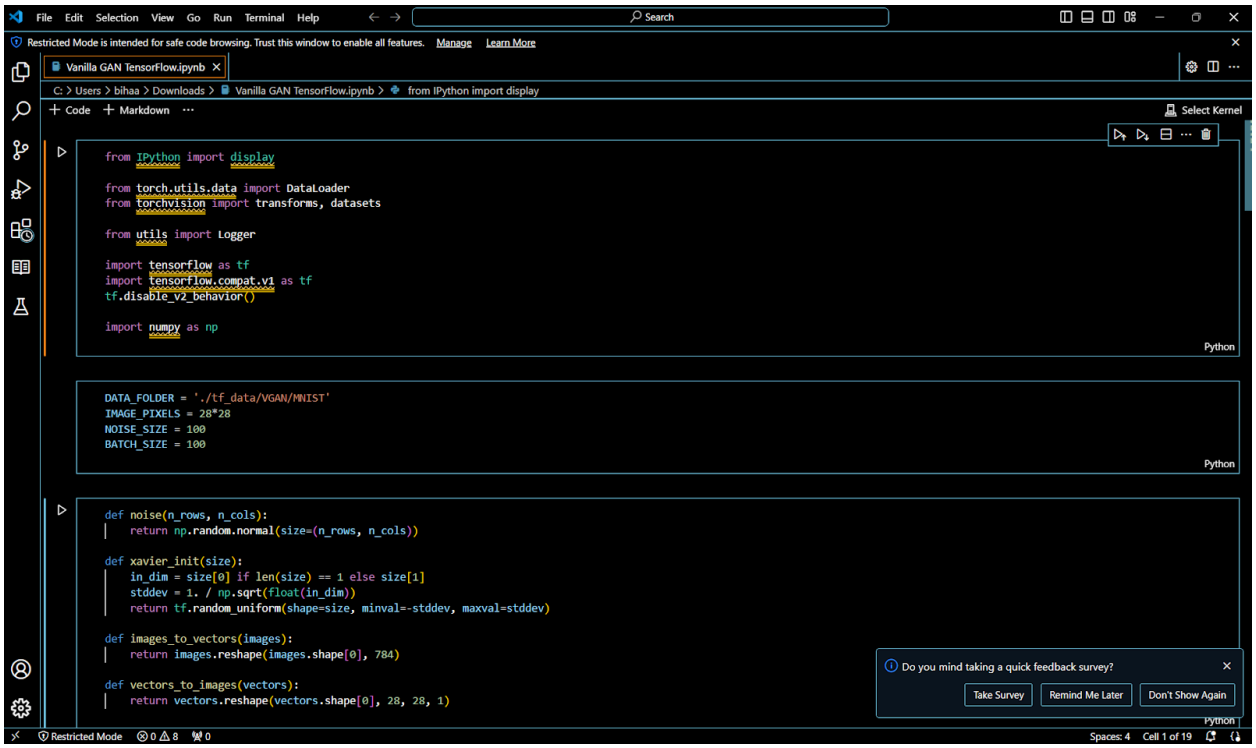
Enhanced Interpretability and Transparency

- Explainable GANs: Future research may focus on making GANs more interpretable, providing insights into why they generate specific outputs, especially in critical applications like medical imaging and security.

- Transparency in Output Generation: Providing users with the ability to trace the GAN's output back to its source data or generation pathway could improve trust and accountability in GAN-generated media.

Implementation:

Code Used -



Load Data

```python
def mnist_data():
    compose = transforms.Compose(
        [transforms.ToTensor(),
        transforms.Normalize((.5,), (.5,))
        ])
    out_dir = '{}/dataset'.format(DATA_FOLDER)
    return datasets.MNIST(root=out_dir, train=True, transform=compose, download=True)
```

Python

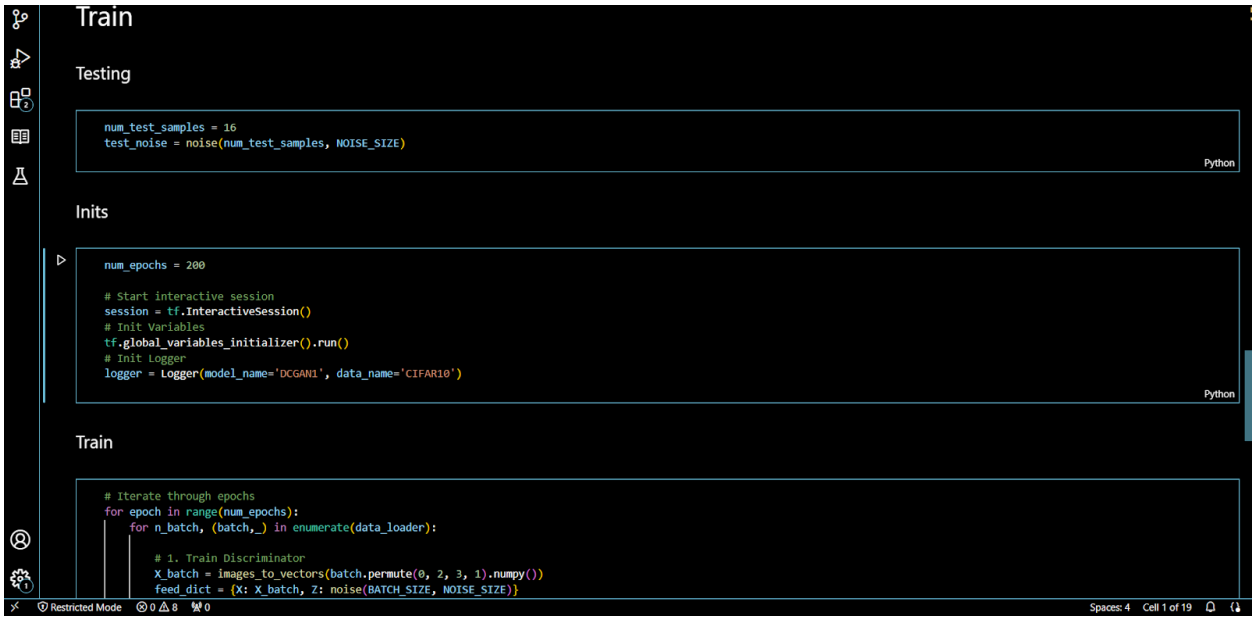
```python
# Load data
data = mnist_data()
# Create loader with data, so that we can iterate over it
data_loader = DataLoader(data, batch_size=BATCH_SIZE, shuffle=True)
# Num batches
num_batches = len(data_loader)
```

Python

Initialize Graph

```python
## Discriminator

# Input
X = tf.placeholder(tf.float32, shape=(None, IMAGE_PIXELS))

# Layer 1 Variables
D_W1 = tf.Variable(xavier_init([784, 1024]))
D_B1 = tf.Variable(xavier_init([1024]))

# Layer 2 Variables
D_W2 = tf.Variable(xavier_init([1024, 512]))
D_B2 = tf.Variable(xavier_init([512]))

# Layer 3 Variables
D_W3 = tf.Variable(xavier_init([512, 256]))
D_B3 = tf.Variable(xavier_init([256]))

# Out Layer Variables
D_W4 = tf.Variable(xavier_init([256, 1]))
D_B4 = tf.Variable(xavier_init([1]))

# Store Variables in list
D_var_list = [D_W1, D_B1, D_W2, D_B2, D_W3, D_B3, D_W4, D_B4]
```

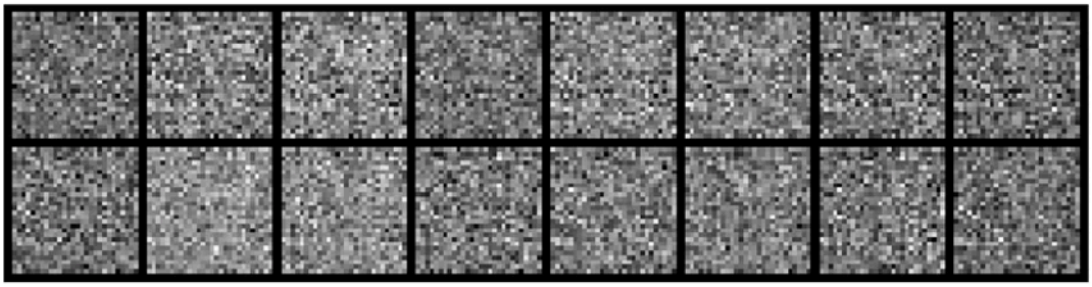
Python


Train
Testing
num_test_samples = 16
test_noise = noise(num_test_samples, NOISE_SIZE)
Python
Inits
num_epochs = 200
# Start interactive session
session = tf.InteractiveSession()
# Init Variables
tf.global_variables_initializer().run()
# Init Logger
logger = Logger(model_name='DCGAN1', data_name='CIFAR10')
Python
Train
# Iterate through epochs
for epoch in range(num_epochs):
for n_batch, (batch,_) in enumerate(data_loader):
# 1. Train Discriminator
X_batch = images_to_vectors(batch.permute(0, 2, 3, 1).numpy())
feed_dict = {X: X_batch, Z: noise(BATCH_SIZE, NOISE_SIZE)}
Restricted Mode
Spaces: 4
Cell 1 of 19

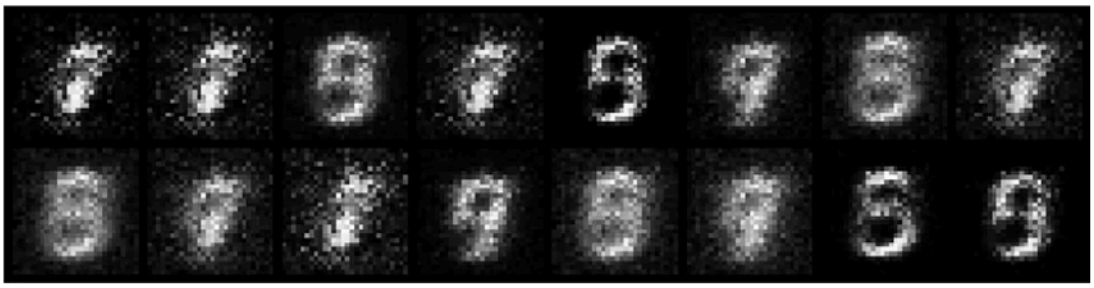

)
# 2. Train Generator
feed_dict = {Z: noise(BATCH_SIZE, NOISE_SIZE)}
_, g_error = session.run(
[G_opt, G_loss], feed_dict=feed_dict
)
if n_batch % 100 == 0:
display.clear_output(True)
# Generate images from test noise
test_images = session.run(
G_sample, feed_dict={Z: test_noise}
)
test_images = vectors_to_images(test_images)
# Log Images
logger.log_images(test_images, num_test_samples, epoch, n_batch, num_batches, format='NHWC');
# Log Status
logger.display_status(
epoch, num_epochs, n_batch, num_batches,
d_error, g_error, d_pred_real, d_pred_fake

Result:

In the beginning the images generated are pure noise:

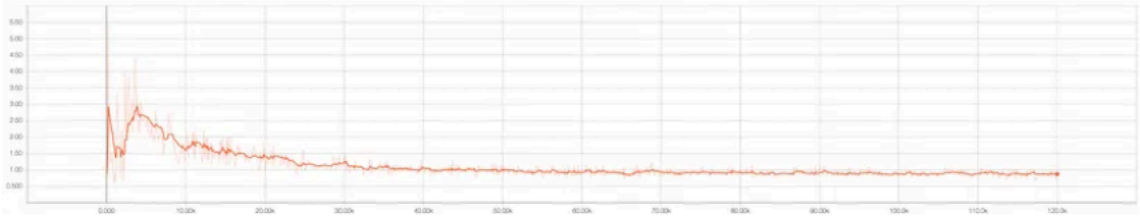


But then they improve,

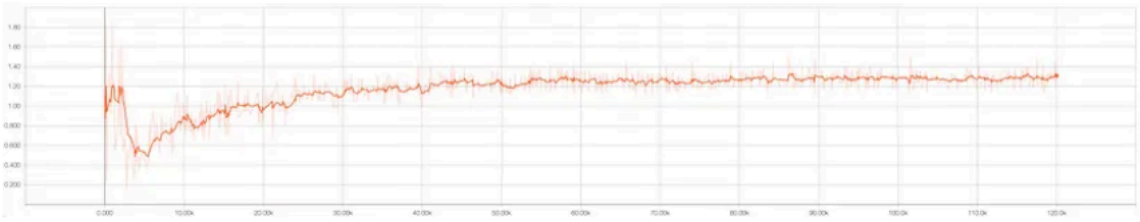


Until you get pretty good syntethic images,

It is also possible to visualize the learning process. As you can see in the next figures, the discriminator error is very high in the beginning, as it doesn't know how to classify correctly images as being either real or fake. As the discriminator becomes better and its error decreases to about .5 at step 5k, the generator error increases, proving that the discriminator outperforms the generator and it can correctly classify the fake samples. As time passes and training continues, the generator error lowers, implying that the images it generates are better and better. While the generator improves, the discriminator's error increases, because the synthetic images are becoming more realistic each time.



Generator's Error through Time



Discriminator's Error through Time

Conclusion:

We started by learning what kind of algorithms they are, and why they're so relevant nowadays. Next we explored the parts that conform to a GAN and how they work together. Finally we finished linking the theory with the practice by programming with a fully working implementation of a GAN that learned to create synthetic examples of the MNIST dataset.

References:

1. ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks

   Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, Chen Change Loy; Proceedings of the European Conference on Computer Vision (ECCV) Workshops, 2018

2. RankSRGAN: Generative Adversarial Networks With Ranker for Image Super-Resolution

   Wenlong Zhang, Yihao Liu, Chao Dong, Yu Qiao; Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 3096-3105

3. Generative Adversarial Networks and Perceptual Losses for Video Super-Resolution

   Alice Lucas; Santiago López-Tapia; Rafael Molina; Aggelos K. Katsaggelos

4. Generative Adversarial Networks

   By Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio