Member-only story

Mastering GANs: Creating Synthetic Fashion Designs

Ritesh Gupta · Follow
Published in Stackademic · 5 min read · Nov 7, 2024

A Step-by-Step Guide to Building Generative Adversarial Networks for Art and Fashion

GANs or generative adversarial networks changed the whole scenario for us to generate and produce data. Starting from creating pictures resembling real-life images, art designs, or even simple prints, GAN has several applications in creative domains related to art, fashion, and entertainment. In this post, we'll try to dive into the very meaning of GAN, and how to train a generator to create synthetic images in forms of art or even fashions.

Photo by cottonbro studio: https://www.pexels.com/photo/woman-in-black-dress-with-black-leather-gloves-6153116/

What is a GAN?

This includes the Generative Adversarial Network, the machine learning model used for generating novel data that potentially may denote real-world data but consisting of two networks, neural ones.

Generator: It is a network that generates new data-noise that resembles the real data.

Discriminator: This network verifies the samples that the generator produces with the intention of determining whether it is real or fake.

The GANs are defined in a game-theoretic framework where the generator and discriminator are adversaries.

The generator wants to dupe the discriminator by generating an appearance of real data.

The purpose of a discriminator is to give the probability whether it receives a real or generated one. The generator and discriminator make improvements over time; the generator increases the realism of the produced data, while the discriminator is perfecting distinguishing between real and fake data.

How GANs Work: The Training Process

Training a GAN encompasses improvement on iterations of the capabilities of both networks. A number of steps are involved in the process.

1. Initializing the Networks

- This generator and discriminator are initialized as neural networks, often using deep learning frameworks like TensorFlow or PyTorch.
- The generator begins with noise that is fed as input, and the discriminator starts with randomly initialized weights that output nothing meaningful yet.

2. Generator's Role

- The generator takes the random noise, a vector of random numbers, and transforms it into the synthetic image. Those images are by far not realistic originally.

3. Discriminator's Role

- The discriminator takes both the real images copied from the training set and the fake images produced by the generator.
- It tries to classify each image as real or fake. In the early training, the discriminator is really bad at distinguishing between both.

4. Training Loop

- The generator and discriminator undergo a training loop where they are constantly competing with each other. The generator attempts to create better images to fool the discriminator, while the discriminator tries to become more accurate in distinguishing fake from real.
- **Loss Functions**: The generator's loss function encourages it to create more realistic images, while the discriminator's loss function penalizes it for incorrectly classifying real or fake images.

The training process continues until the generator creates images that are nearly indistinguishable from real images.

5. Improvement Over Time

- Over several epochs (iterations through the dataset), the generator improves its ability to produce realistic images, while the discriminator gets better at telling the difference between real and fake.

Example Project: Fashion Design Generator 👗🎨

Let's consider an example of training a GAN to generate synthetic **fashion designs**. Here's how we can approach the project:

1. Dataset Collection

- To generate fashion designs, the first step is to gather a dataset of fashion images. A popular dataset for such tasks is **Fashion MNIST**, which contains 60,000 grayscale images of 10 different fashion categories such as shirts, shoes, and dresses.
- Alternatively, you could use datasets like **DeepFashion** for more detailed fashion items.

2. Preprocessing the Data

- The images need to be preprocessed to be in the right format for training the GAN. This typically involves resizing images to a standard size (e.g., 64x64 pixels) and normalizing pixel values (scaling them to a range of -1 to 1).

3. Building the GAN Model

- The generator network could be a **convolutional neural network** (CNN) that takes a random noise vector and outputs an image of a fashion item.
- The discriminator would also be a CNN that takes an image as input and outputs a probability of whether the image is real or fake.

```
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential

# Generator Model
generator = Sequential()
generator.add(layers.Dense(128, input_dim=100))
generator.add(layers.LeakyReLU(0.2))
generator.add(layers.BatchNormalization(momentum=0.8))
generator.add(layers.Dense(256))
generator.add(layers.LeakyReLU(0.2))
generator.add(layers.BatchNormalization(momentum=0.8))
generator.add(layers.Dense(512))
generator.add(layers.LeakyReLU(0.2))
generator.add(layers.BatchNormalization(momentum=0.8))
generator.add(layers.Dense(1024))
generator.add(layers.LeakyReLU(0.2))
generator.add(layers.BatchNormalization(momentum=0.8))
generator.add(layers.Dense(64*64*3, activation='tanh'))
generator.add(layers.Reshape((64, 64, 3)))

# Discriminator Model
discriminator = Sequential()
discriminator.add(layers.Conv2D(64, kernel_size=3, strides=2, padding='same', in
discriminator.add(layers.LeakyReLU(0.2))
discriminator.add(layers.Dropout(0.25))
discriminator.add(layers.Conv2D(128, kernel_size=3, strides=2, padding='same'))
discriminator.add(layers.LeakyReLU(0.2))
discriminator.add(layers.Dropout(0.25))
discriminator.add(layers.Flatten())
discriminator.add(layers.Dense(1, activation='sigmoid'))
```

4. Training the GAN

- After building the generator and discriminator, the next step is to train them. This involves alternating between training the discriminator and generator:
- Train the discriminator with both real and fake images.
- Train the generator by using the discriminator's feedback to improve the quality of the synthetic images.

```
# Compile the discriminator with binary cross-entropy loss and an Adam optimizer
discriminator.compile(loss='binary_crossentropy', optimizer='adam', metrics=['ac

# Combined model (generator + discriminator)
discriminator.trainable = False
combined = Sequential([generator, discriminator])
combined.compile(loss='binary_crossentropy', optimizer='adam')
```

5. Challenges in Training GANs

- **Mode Collapse**: The generator might start producing the same image repeatedly, which reduces diversity.
- **Vanishing Gradient**: The discriminator becomes too strong, making it hard for the generator to improve.
- **Training Stability**: GANs are notoriously difficult to train due to the adversarial nature of the process. Tuning the learning rates and using techniques like **gradient penalty** can help stabilize training.

Example Output: Fashion Design Images 👘👠

After training the GAN, you'll be able to generate synthetic images of fashion items that resemble real clothing designs. Here are some examples of synthetic fashion designs generated by a GAN:

- 🧥 A stylish leather jacket.
- 👟 Trendy sneakers with unique patterns.
- 👗 Elegant dresses with intricate designs.

These images might not be perfect at first, but as the GAN trains further, they will improve in quality and become more realistic.

Applications of GANs in Fashion and Art 🎨👗

1. **Fashion Design**: GANs can help designers create new fashion items, visualize designs, and explore trends.
2. **Art Generation**: GANs can generate unique artwork, such as paintings or illustrations, blending different artistic styles.
3. **Virtual Try-Ons**: GANs can be used to create virtual fitting rooms where customers can try on clothes without physically wearing them.

Conclusion

Building a **Generative Adversarial Network** (GAN) to generate synthetic images, such as fashion designs or artwork, is a fascinating project that combines creativity with cutting-edge machine learning technology. By understanding how GANs work, and applying the training process, you can create realistic images that open up new possibilities in various industries.

GANs are a powerful tool for generating creative content, but training them can be challenging. By following this guide and experimenting with different datasets, architectures, and techniques, you can create high-quality synthetic images that push the boundaries of AI and design. 💥💡

Happy GAN training! 🧑‍💻

Stackademic 🎓

Thank you for reading until the end. Before you go:

- Please consider **clapping** and **following** the writer! 👏
- Follow us X | LinkedIn | YouTube | Discord | Newsletter | Podcast
- Create a free AI-powered blog on Differ.
- More content at Stackademic.com

Gans | Artificial Intelligence | Data Science | Machine Learning | Python

Written by Ritesh Gupta

3.6K Followers · 28 Following

Follow

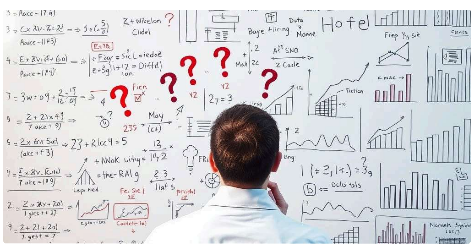
Data Scientist, I write Article on Machine Learning| Deep Learning| NLP | Open CV | AI Lover ❤️

No responses yet

What are your thoughts?

Respond

More from Ritesh Gupta and Stackademic



Ritesh Gupta

Can You Handle These 25 Toughest Data Science Interview Questions?

The role of a Data Scientist demands a unique blend of skills, including statistics, machine…

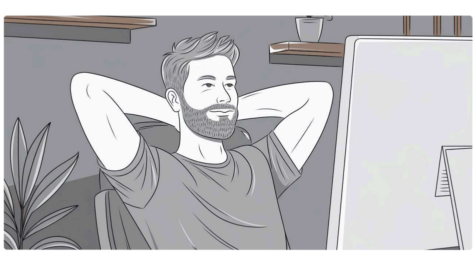
Sep 25 · 211 · 7

In Stackademic by Abdur Rahman

Python is No More The King of Data Science

5 Reasons Why Python is Losing Its Crown

Oct 23 · 8.4K · 34



In Stackademic by Abdur Rahman

20 Python Scripts To Automate Your Daily Tasks

A must-have collection for every developer

Oct 7 · 4.2K · 55

In Python in Plain English by Ritesh Gupta

7 GitHub Repos to Transform You into a Pro ML/AI Engineer

Hands-On Guides, Tools, and Frameworks to Fast-Track Your AI Journey

Nov 5 · 143 · 1

See all from Ritesh Gupta

See all from Stackademic

Recommended from Medium

Thomas Reid

A quick test of DeepSeek's new R1-Lite-Preview model

Does this Chinese LLM really rival OpenAI's o1-preview?

6d ago · 33 · 1

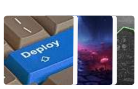
In Level Up Coding by Salvatore Raieli

Context vs. Prior Knowledge: How to Modify LLM Behavior

Unveiling the Mechanism Behind Controlling Sensitivity in Language Models

Nov 21 · 333 · 3

Lists



Predictive Modeling w/ Python

20 stories · 1691 saves

Practical Guides to Machine Learning

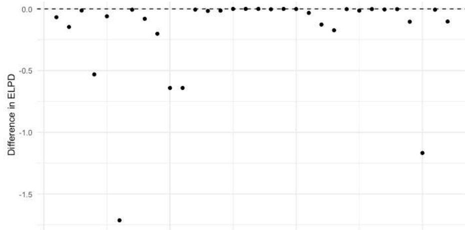
10 stories · 2057 saves

Natural Language Processing

1839 stories · 1461 saves

ChatGPT prompts

50 stories · 2295 saves



William Christiansen, Ph.D.

Leave One Out (LOO) Cross Validation for Bayesian Modeling

LOO cross-validation is a powerful technique for evaluating the predictive performance of...

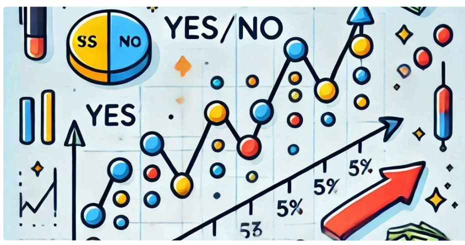
Nov 22 10

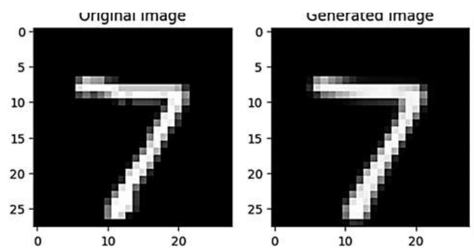
In AI Advances by Muhammad Saad Uddin

Are the Rumors True? Have We Reached the Peak of LLM...

Separating Fact from Fiction

6d ago 531 2





In InsiderFinance Wire by Leo Mercanti

Logistic Regression (AI Meets Finance—Algorithms Series)

The Role of Logistic Regression in Predictive Financial Modeling

Nov 18 167

om pramod

Autoencoders Explained

Part 4: Variational Autoencoders

Jun 16 403

See more recommendations