



Golden  
Poppy

Imagine you're learning to identify a "Golden Poppy" flower. You don't know much about flowers initially.

- **Layer 1 (Basic Features):** You start by noticing very basic features:
  - **Shape:** Petals are rounded.
  - **Color:** A vibrant golden yellow.
  - **Stem:** Thin and green.
- **Layer 2 (Combining Basic Features):** You then learn to combine these basic features:
  - **Petal Arrangement:** The rounded petals are arranged in a cup-like shape.
  - **Flower Center:** The center of the flower has many tiny yellow stamens.
  - **Overall Structure:** The flower has a delicate, almost translucent quality.
- **Layer 3 (Abstract Concepts):** Finally, you learn to recognize the "Golden Poppy" as a whole by combining these higher-level features:
  - **Golden Poppy Shape:** The specific cup-like shape, the delicate texture, and the way the petals curve.
  - **Golden Poppy Color:** The specific shade of golden yellow, the way it seems to glow in the sunlight.



You've learned hierarchically.



You started with simple features, combined them into more complex features, and finally arrived at the abstract concept of a "Golden Poppy".

# How this relates to DBNs

- A DBN works similarly. Imagine a DBN trained on images of flowers:
- **Layer 1 (RBM 1):** The first RBM might learn to recognize simple features like edges, corners, and basic colors in the image. These are like the basic shapes and colors in our flower example.
- **Layer 2 (RBM 2):** The second RBM takes from the first RBM (which represent the basic features) as input. It learns to combine these basic features into more complex features, such as the shape of a petal, the texture of a leaf, or the pattern of light and shadow. This is like combining the basic features to recognize petal arrangement or overall structure.
- **Layer 3 (RBM 3, and so on):** Subsequent RBMs continue this process, learning even more abstract features, eventually leading to the recognition of the flower type itself. This is like recognizing the "Golden Poppy" as a whole.



# **Restricted Boltzmann Machine (RBM)**



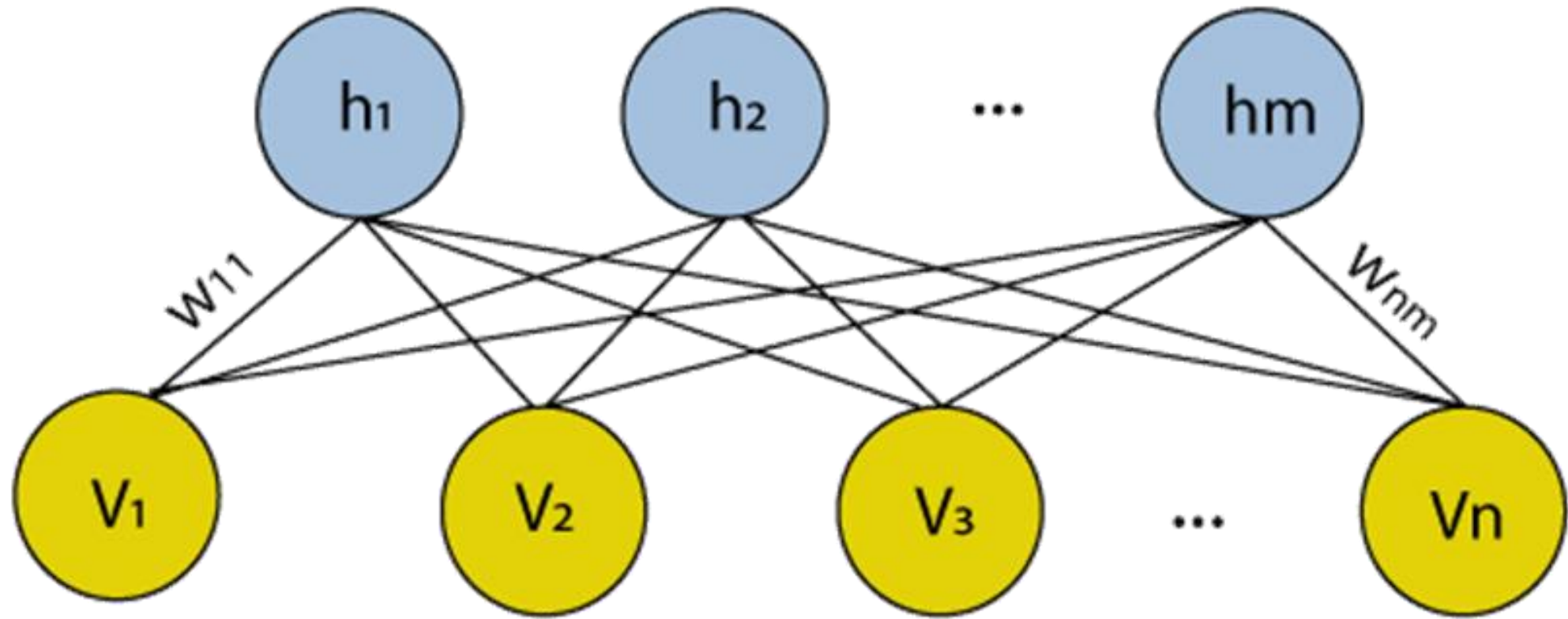
# Restricted Boltzmann Machine (RBM)

A Restricted Boltzmann Machine is a two-layer probabilistic neural network.

Its concept was first introduced in 1986 as the Harmonium model, but the term RBM was not used until the mid-2000s.

Its first layer (visible layer) interacts with the raw data, and the second (hidden layer) learns high-level features from the first one.





Restricted Boltzmann Machine  
Architecture



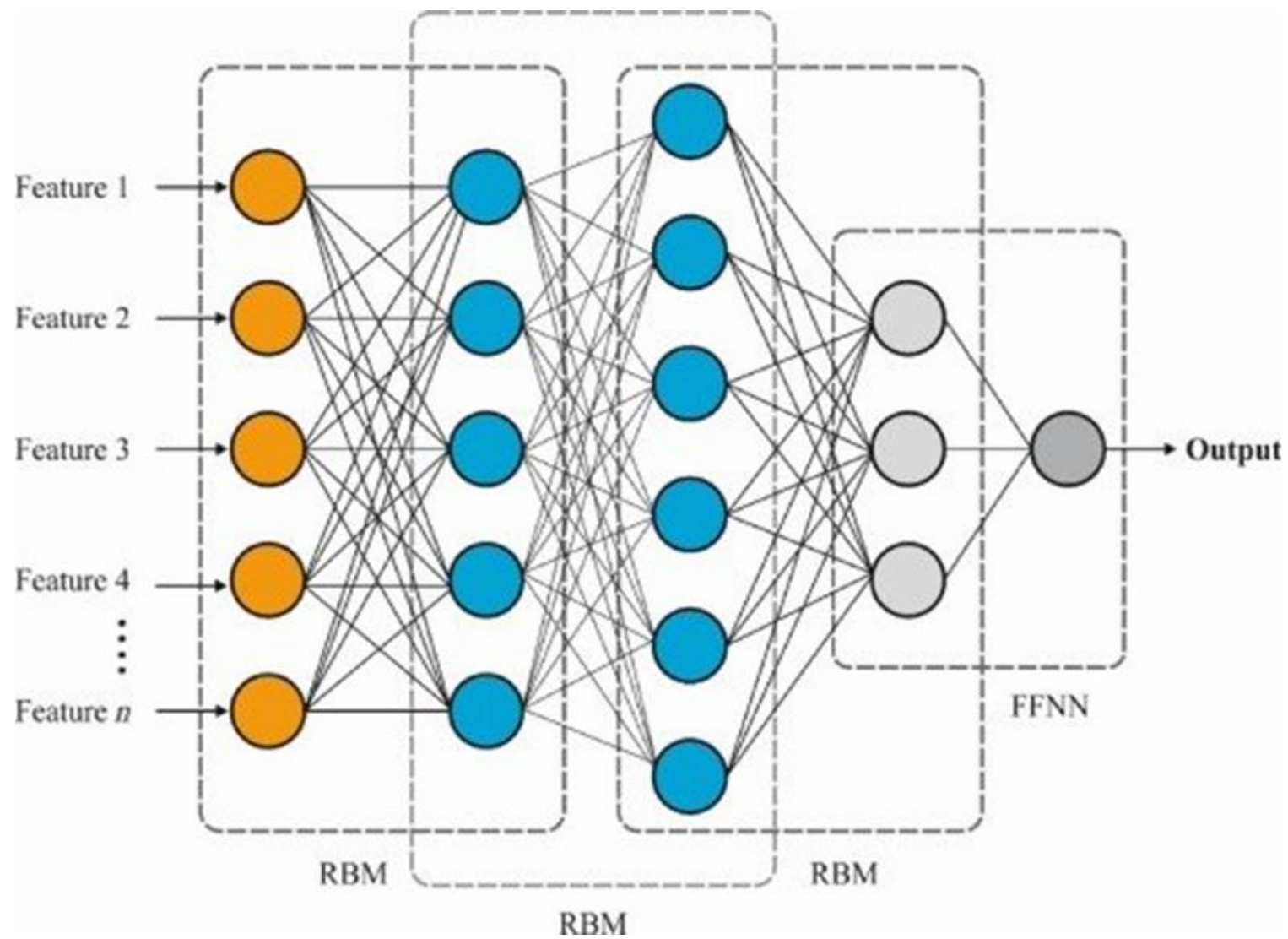
# Deep Belief Network



# Deep Belief Network

A Deep Belief Network extends the RBM functionality by creating overlapping stacks of the model. The stacks overlap since the hidden layer of one model is the visible layer of the next one.

These RBMs are trained independently, and the overall architecture forms the deep belief network.



# Deep Belief Network Architecture

# Benefits of DBNs

Deep belief networks use probabilistic modeling and a supervised learning approach to offer certain benefits over conventional neural networks.

- Ability to handle large data using hidden units to extract underlying correlations.
- Faster training and better results.
- Achieving global minima due to better weights initialization.

# DBN Applications

Image Classification

Text Classification

Image Generation

Speech Recognition

# Present State of Deep Belief Networks

---



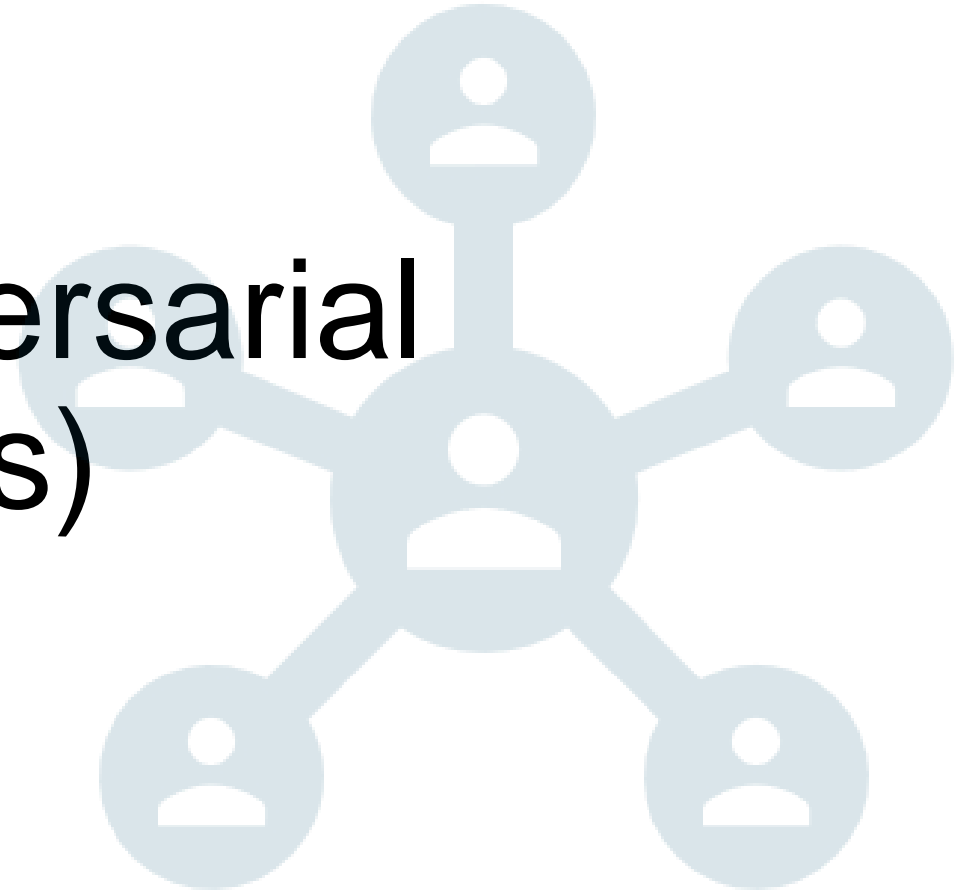
Deep belief networks did not gain much popularity despite demonstrating hybrid capabilities. Much of this was because the probabilistic model required a large amount of data to understand the underlying patterns properly like gradient descent.



On the other hand, the advances in conventional deep networks, such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Artificial Neural Networks (ANNs), have provided ground-breaking results. These algorithms have become more efficient with time and are the foundations of most modern CV and NLP applications.



# Generative Adversarial Networks (GANs)



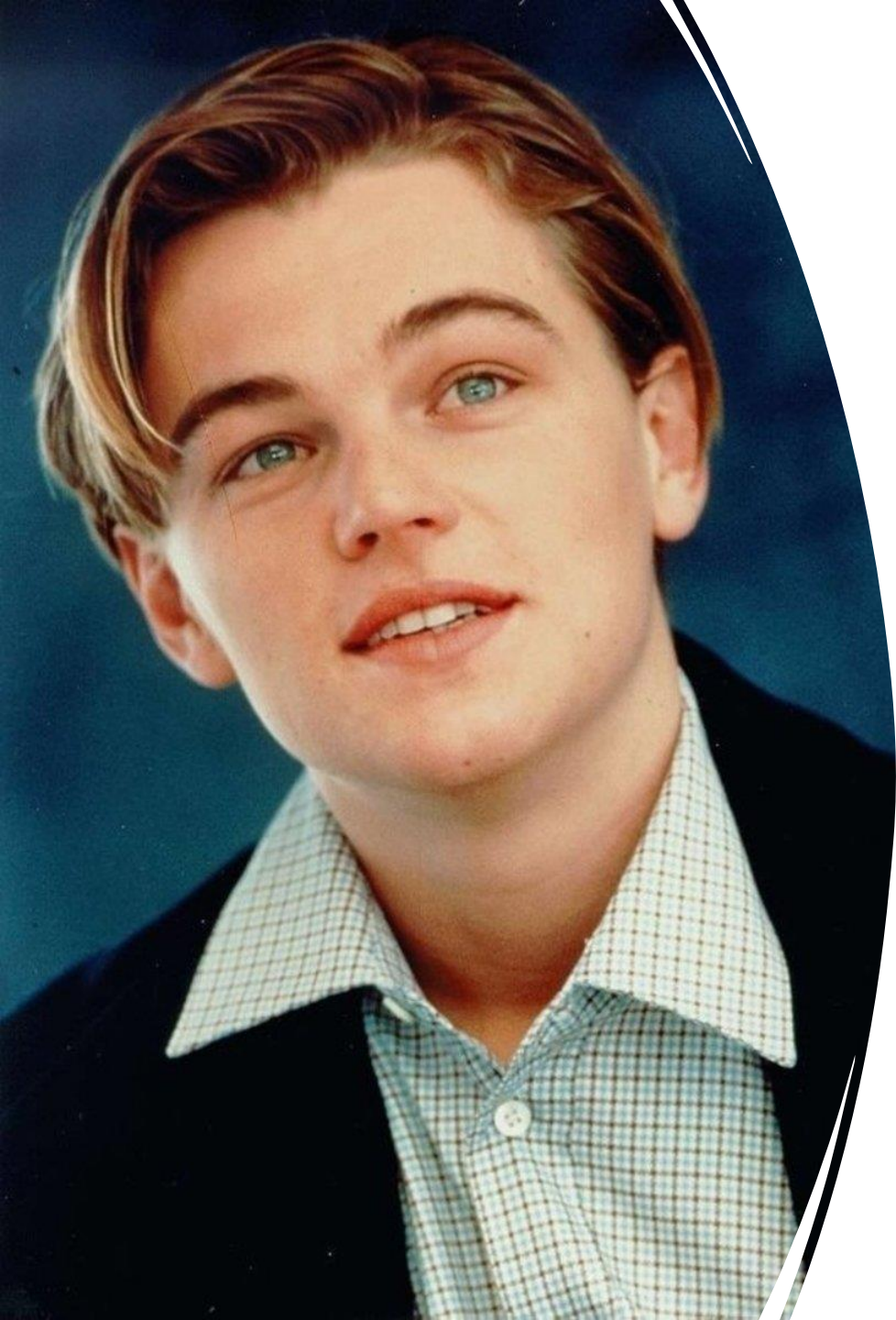
# Generative Adversarial Networks (GANs)

Imagine two players: a forger and a detective.

The Forger (Generator): The forger's job is to create fake things, like paintings, that look as real as possible.

The Detective (Discriminator): The detective's job is to tell the difference between real things and the forger's fakes.





# Catch Me If You Can (2002)

---

Leonardo DiCaprio



# Frank Abagnale Jr.

- Frank Abagnale Jr. is a fascinating and controversial figure. Here's a deeper dive into his life story:
- **Early Life and Cons:**
- **Teenage Runaway:** Abagnale's story begins in the 1960s. As a teenager, he ran away from home after his parents' divorce.
- **Impersonation:** To survive, he quickly learned to impersonate various professionals. His most famous impersonations included:
  - A Pan American World Airways pilot: He forged credentials and "deadheaded" (flew for free) on numerous flights around the world.
  - A doctor: He worked briefly as a pediatrician in a hospital.
  - A lawyer: He passed the Louisiana bar exam without attending law school.
- **Check Forgery:** He became a master of check forgery, creating sophisticated counterfeit checks that were difficult to detect. This was his primary method of obtaining money

# Capture and Aftermath:

- **International Pursuit:** Abagnale's activities caught the attention of law enforcement agencies worldwide. He was pursued by the FBI, particularly by agent Carl Hanratty (a fictionalized version of real FBI agents).
- **Capture and Imprisonment:** He was eventually captured in France and extradited to the United States. He served time in federal prison.
- **Rehabilitation:** After his release, Abagnale began working with the FBI, using his expertise to help them detect and prevent fraud.
- **Security Consultant:** He founded Abagnale & Associates, a security consulting firm that advises businesses on fraud prevention.

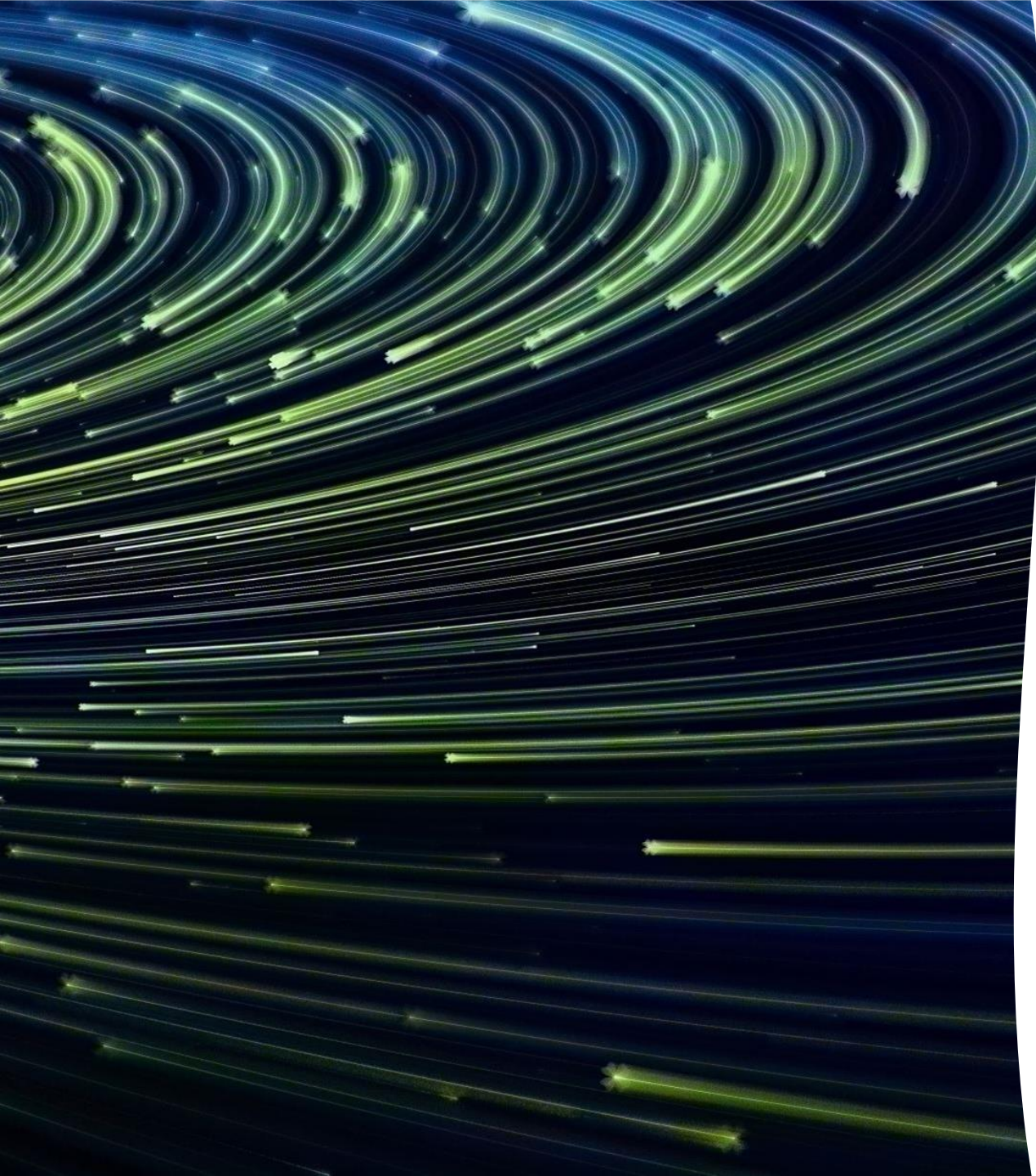
# Generative Adversarial Networks (GANs)

Imagine two players: a forger and a detective.

The Forger (Generator): The forger's job is to create fake things, like paintings, that look as real as possible.

The Detective (Discriminator): The detective's job is to tell the difference between real things and the forger's fakes.





They play a game where the forger tries to get better at creating fakes, and the detective tries to get better at spotting them.

This is exactly how a GAN works



# A GAN consists of two neural networks:

**Generator (The Forger):** This network takes random noise as input and tries to transform it into something that looks like real data (e.g., an image, a sound clip, text).

**Discriminator (The Detective):** This network takes two types of input: real data from your dataset and fake data generated by the generator. Its job is to classify each input as either "real" or "fake."

# The Training Process: Adversarial Training



The two networks are trained simultaneously in an adversarial manner:



**Generator Training:** The generator tries to fool the discriminator. It generates fake data and tries to make it look so real that the discriminator classifies it as "real." The generator's loss is based on how often it can trick the discriminator.



**Discriminator Training:** The discriminator tries to catch the generator's fakes. It's trained on both real data (labeled as "real") and fake data from the generator (labeled as "fake").

The discriminator's loss is based on how accurately it can distinguish between real and fake data.



This process is repeated many times. As the training progresses:

The generator gets better at creating realistic fake data.

The discriminator gets better at spotting fake data.

Ideally, this leads to a point where the generator is producing fakes that are almost indistinguishable from real data, and the discriminator is struggling to tell the difference.

Imagine you want  
to train a GAN to  
generate realistic  
pictures of cats.



**Generator:** Takes random noise as input and tries to create a cat image. At the beginning, its attempts will be terrible – maybe just random colors.

**Discriminator:** Takes two types of input: real cat pictures from a dataset and the fake cat pictures generated by the generator. It tries to classify each image as "real" or "fake."

**Training:** The generator gets feedback: "Your cat picture was classified as fake. Try harder!" It adjusts its weights to create better cat pictures.

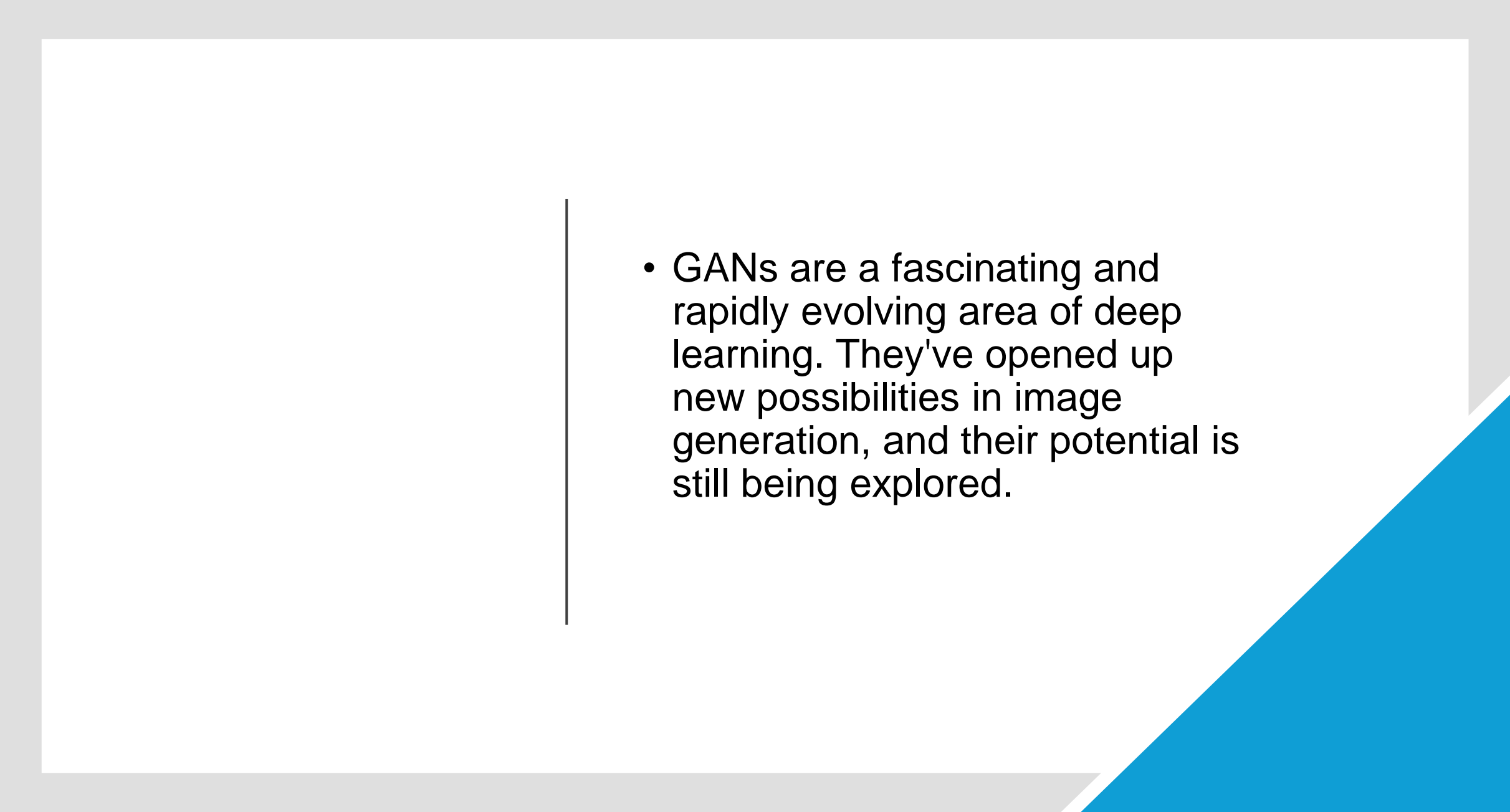
The discriminator also gets feedback: "You misclassified this real cat picture as fake!" or "You classified this fake cat picture as real!" It adjusts its weights to become a better cat picture detector.

This process continues until the generator is creating cat pictures that are so realistic that the discriminator can't reliably tell them apart from real cat pictures.

# Why are GANs useful?

GANs are incredibly powerful for generative tasks. They can be used for:

- **Image generation:** Creating realistic images of anything (people, objects, scenes).
- **Image editing:** Modifying existing images in creative ways.
- **Style transfer:** Applying the style of one image to another.
- **Text-to-image synthesis:** Generating images from text descriptions.
- **Video generation:** Creating realistic videos.

- 
- A decorative gray border surrounds the slide content. A blue triangle is located in the bottom right corner, pointing towards the center.
- GANs are a fascinating and rapidly evolving area of deep learning. They've opened up new possibilities in image generation, and their potential is still being explored.

