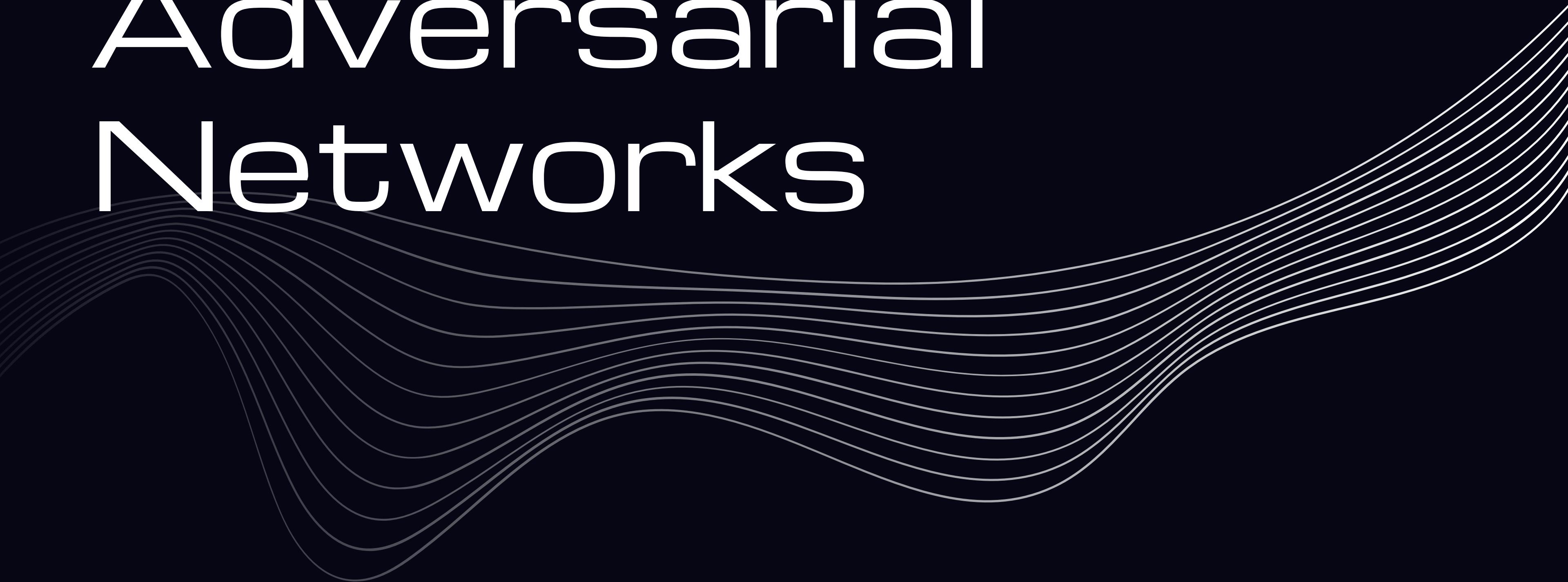


Generative Adversarial Networks

01



02

“Derin öğrenmede ilginç gelişmeler olmakta... Bana göre bunlardan en önemlisi çekişmeli eğitim veya diğer bir deyişle GAN yani Çekişmeli Üretici Ağlardır. Önerilmekte olan bu yapı ve çeşitleri, geçtiğimiz on yılın en ilginç fikridir.”

YANN LECUN.



Generative Adversarial Networks nedir?

03

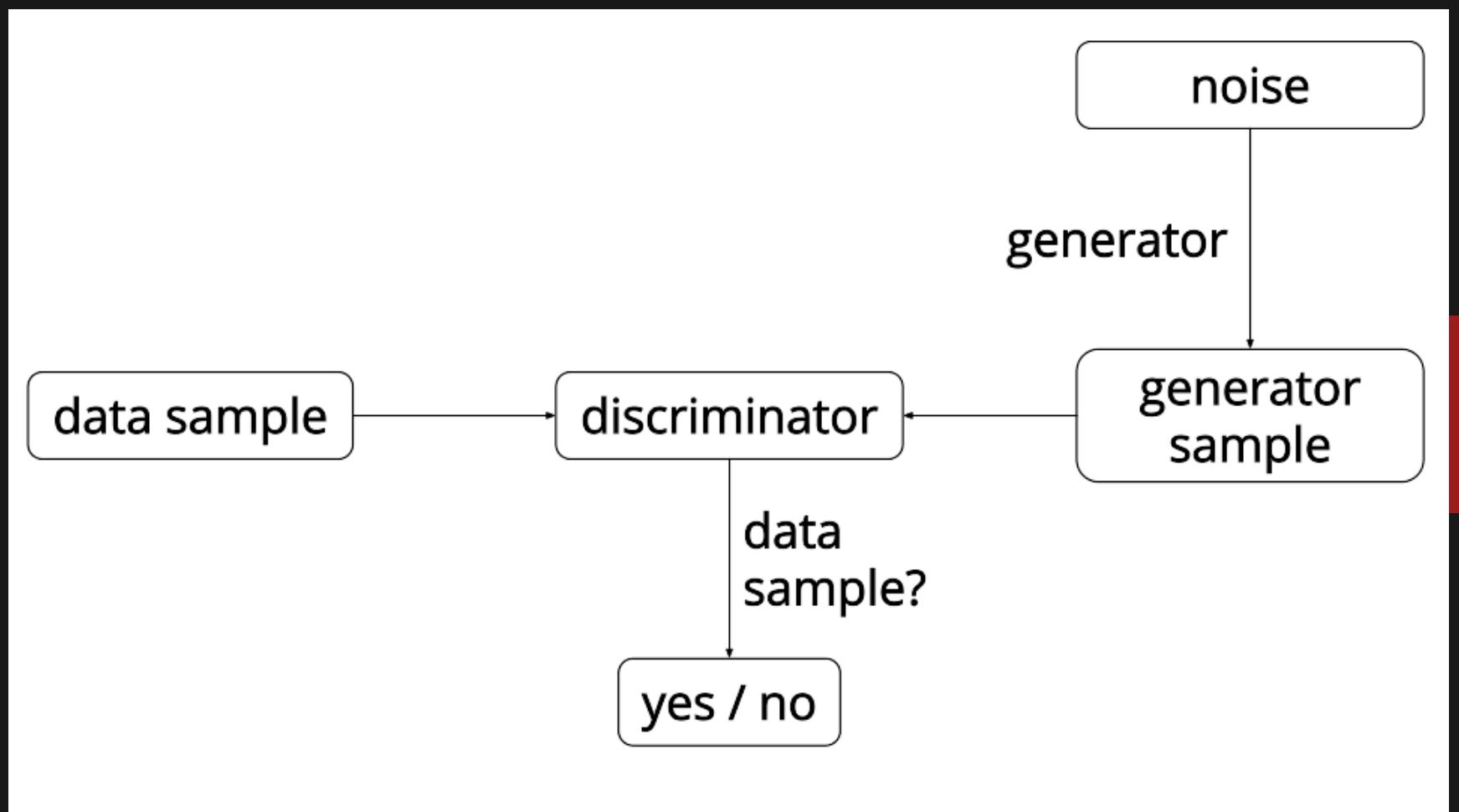
GAN, belirli bir veri dağılımını taklit etmeyi öğrenebilen makine öğrenimi sistemleridir.

GAN'lar , biri veri üretmek için eğitilmiş, diğerinin sahte verileri gerçek verilerden ayırmak için eğitilmiş iki sinir ağından oluşur (bu nedenle modelin "rakip" doğasıdır).

Discriminative model , girdi verilerini (x) istenen bazı çıktı sınıfı etiketiyle (y) eşleyen bir ilevi öğrenir. Olasılıksal terimlerle , $P(y|x)$ koşullu dağılımını doğrudan öğrenirler.

Generative model, giriş verilerinin ortak olasılığını öğrenmeye çalışır ve aynı anda etiketler, yani $P(x, y)$. Bu, Bayes kuralı aracılığıyla sınıflandırma için $P(y | x)$ 'e dönüştürülebilir, ancak üretkenlik yeteneği, muhtemelen yeni (x, y) örnekler oluşturmak gibi başka bir şey için de kullanılabilir.

04



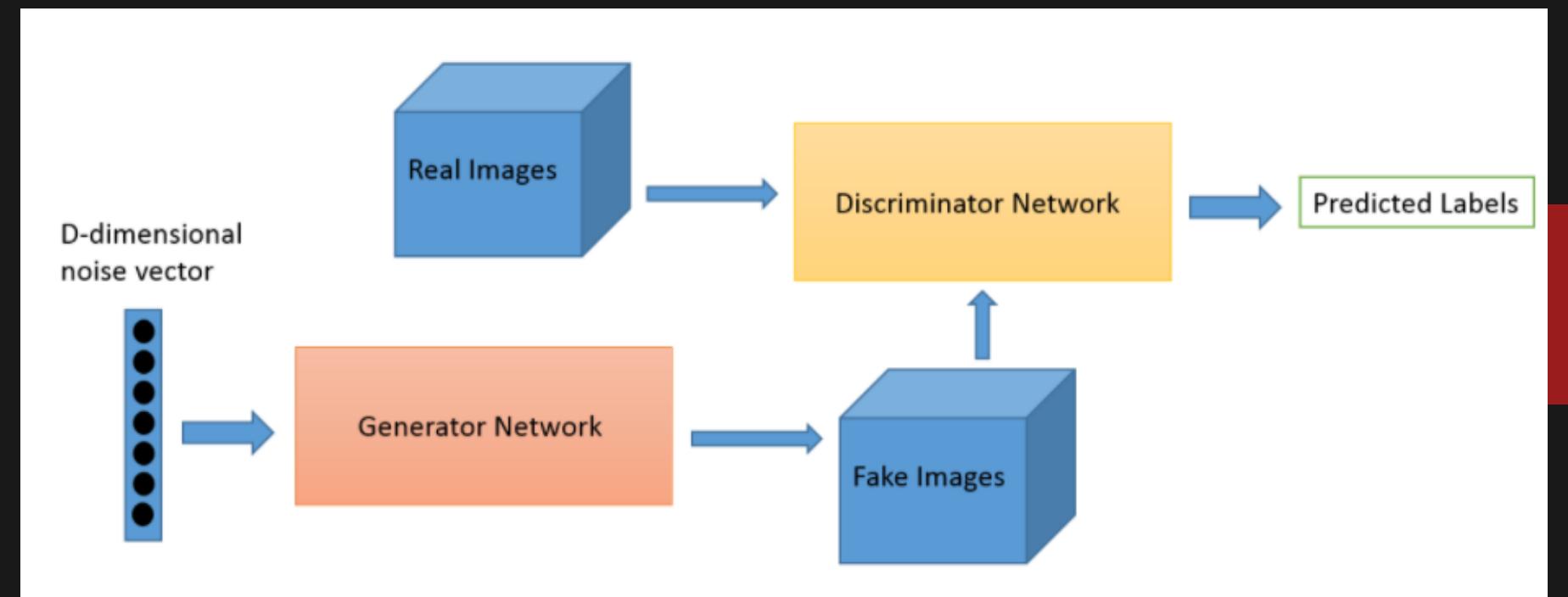
05

Bir GAN'ın arkasındaki ana fikir, iki rakip sinir ağının modeline sahip olmaktadır. Gürültüyü girdi olarak alır ve örnekler üretir (ve buna jenerator denir). Diğer model (ayırıcı olarak adlandırılır), hem jeneratorden hem de eğitim verilerinden örnekler alır ve iki kaynak arasında ayrımlı yapabilmelidir. Bu iki ağ, üreticinin giderek daha gerçekçi örnekler üretmeyi öğrendiği sürekli bir oyun oynar ve discriminator, üretilen verileri gerçek verilerden ayırt etmede daha iyi ve daha iyi olmayı öğrenir. Bu iki ağ aynı anda eğitilir ve umut, rekabetin üretilen örnekleri gerçek verilerden ayırt edilemez hale getirmesidir.

06

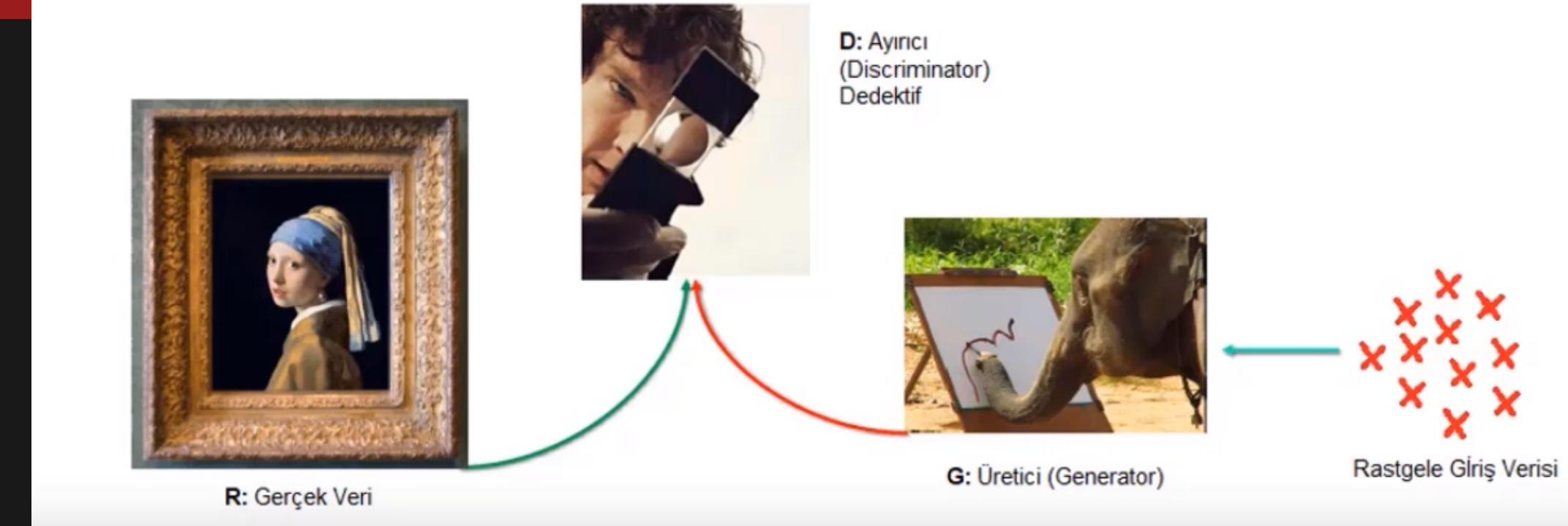
GAN'lar Nasıl Çalışır?

Diyelim ki Mona Lisa'yı taklit etmekten daha sıradan bir şey yapmaya çalışıyoruz. MNIST veri setinde bulunanlar gibi gerçek dünyadan alınan elle yazılmış rakamlar üreteceğiz. Gerçek MNIST veri kümesinden bir örnek gösterildiğinde ayırcının amacı, gerçek olanları tanıtmaktır. Bu arada, jeneratör ayırcıya aktardığı yeni, sentetik görüntüler yaratıyor. Sahte olsalar bile, onlar da gerçek kabul edilecekleri umuduyla bunu yapar. Jeneratörün amacı, elle yazılan elverişli rakamlar oluşturmaktır: yakalanmadan yalan söylemek. Ayırcısının amacı, jeneratörden gelen görüntülerin sahte olarak belirlemektir.



07

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

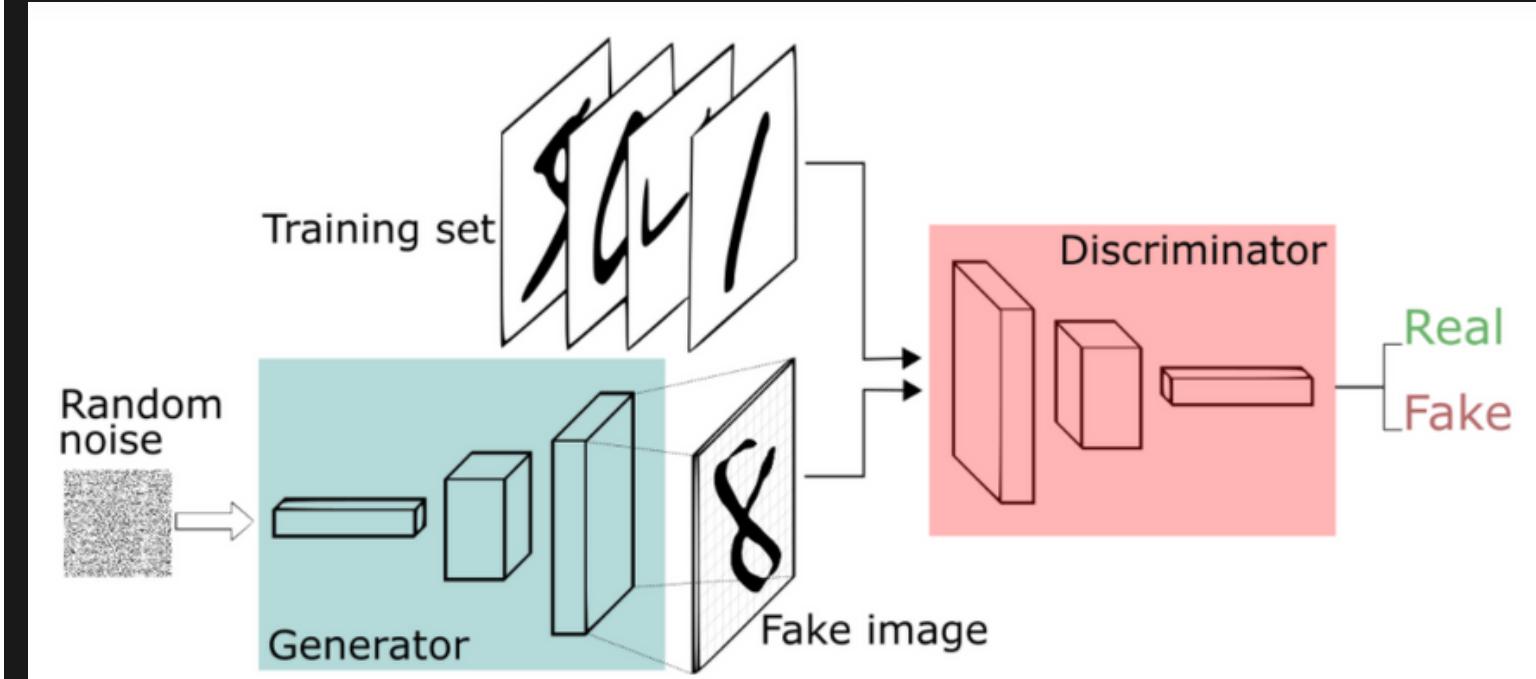


Model iki girişlidir. Birinci girişte rastgele gürültü üretilir. Bunu üretici ağ (Generator) yapıyor. Ters evrişim yaparak gürültüden görüntü elde etmeyi sağlar.

Bu elde edilen veri ile gerçek veriyi ayırcı(discriminator) ağa iletiyoruz. Bu da düz evrişim yapar. Ve ikili sınıflandırma yapar (real-fake).

Giderek discriminator gerçekle sahteyi ayırt edememeye başlar. Yani üreticinin performansı artar ayırt edicinin performansı azalır. Bir yandan geri yayılım ile üretici beslenmeye devam eder.

MNIST için ayırcı ağ, kendisine beslenen görüntülerini kategorize edebilen standart bir evrişimli ağdır, görüntülerini gerçek veya sahte olarak etiketleyen iki terimli bir sınıflandırıcıdır. Jeneratör, bir anlamda ters evrişimli bir ağdır: Standart bir evrişimli sınıflandırıcı bir görüntüyü alıp bir olasılık oluşturmak için altörneklerken, üretici bir rasgele gürültü vektörü alır ve bunu bir görüntüye yukarı örnekler. Birincisi, maksimum paylaşım gibi alt örnekleme teknikleriyle verileri atar ve ikincisi yeni veriler üretir. Her iki ağ da sıfır-zum oyununda farklı ve zıt bir amaç işlevi veya kayıp işlevini optimize etmeye çalışıyor. Bu aslında bir aktör-eleştirmen modelidir. Ayırcı davranışını değiştirdikçe, jeneratör de değişir ve bunun tersi de geçerlidir. Kayıpları birbirlerini itiyor.



09

GAN Eğitiminde İpuçları

Birkaç satır Ayırıcıyı eğittiğinizde, jenerator değerlerini sabit tutun; ve jeneratoru eğittiğinizde, ayırıcıyı sabit tutun. Her biri statik bir düşmana karşı antrenman yapmalıdır. Örneğin, bu, jeneratorre öğrenmesi gereken gradyan hakkında daha iyi bir okuma sağlar. gövde metni ekle

Aynı şekilde, eğitime başlamadan önce ayırmayı MNIST'e karşı önceden eğitmek, daha net bir gradyan oluşturacaktır.

GAN'ın her iki tarafı da diğerine üstün gelebilir. Ayırıcı çok iyi ise, 0 veya 1'e o kadar yakın değerler döndürür ki, jenerator gradyanı okumakta zorlanacaktır. Jenerator çok iyi ise, yanlış negatiflere yol açan ayırmacı zayıflıkları ısrarla istismar edecektir. Bu, ağların ilgili öğrenme oranları ile hafifletilebilir. İki sinir ağı benzer bir "beceri seviyesine" sahip olmalıdır.

GAN'ların eğitilmesi uzun zaman alır. Tek bir GPU'da bir GAN saatler sürebilir ve tek bir CPU'da bir günden fazla sürebilir. Ayarlaması ve dolayısıyla kullanımı zor olsa da, GAN'lar birçok ilginç araştırma ve yazmayı teşvik etti .

10

Kullanım Alanları

<https://deephunt.in/the-gan-zoo-79597dc8c347>

Yazıdan Görüntü Sentezleme:

this small bird has a pink breast and crown, and black primaries and secondaries.



the flower has petals that are bright pinkish purple with white stigma



this magnificent fellow is almost all black with a red crest, and white cheek patch.



this white and yellow flower have thin white petals and a round yellow stamen



Text descriptions (content) Images (style)

The bird has a **yellow breast** with **grey features** and a **small beak**.

This is a large **white** bird with **black wings** and a **red head**.

A small bird with a **black head and wings** and features **grey wings**.

This bird has a **white breast**, brown and white coloring on its head and wings, and a thin pointy beak.

A small bird with **white base** and **black stripes** throughout its belly, head, and feathers.

A small sized bird that has a cream belly and a short pointed bill.

This bird is **completely red**.

This bird is **completely white**.



Eskizden Resim Üretme:

User edits



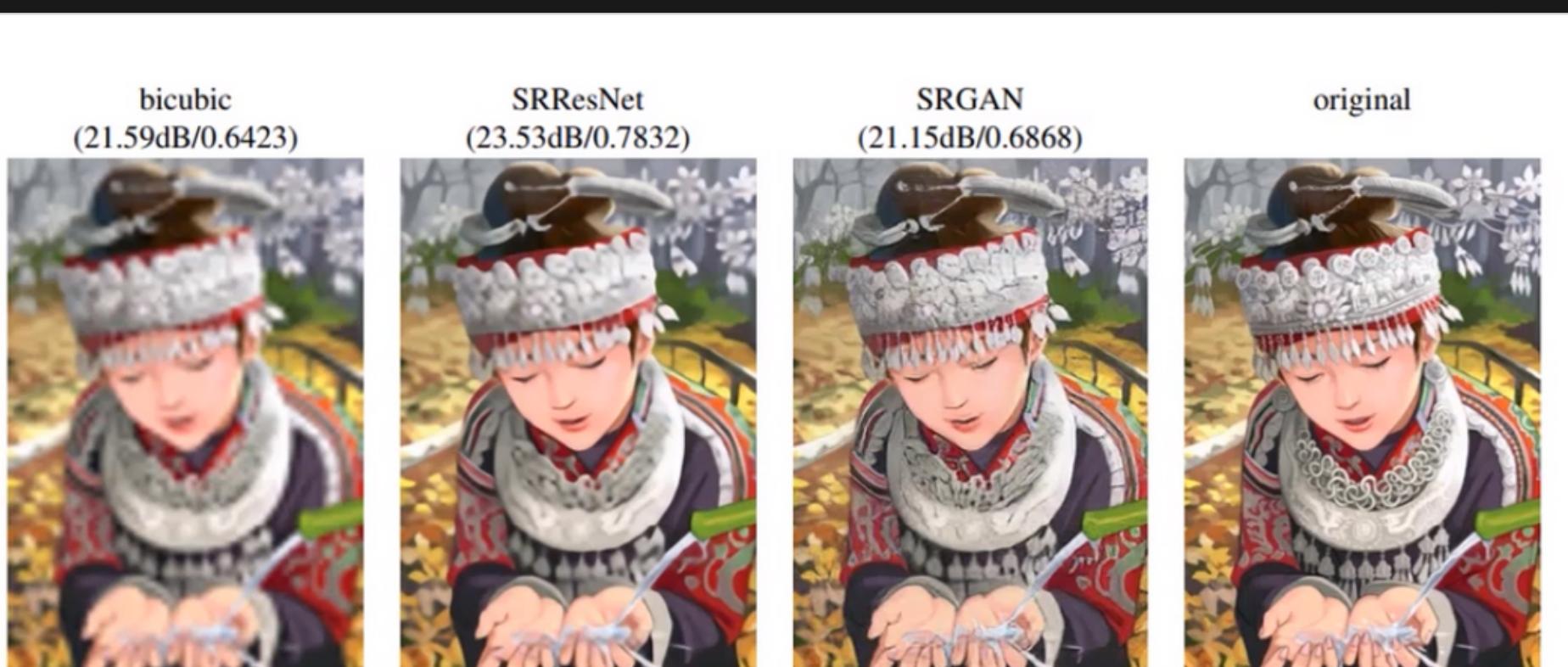
Generated images



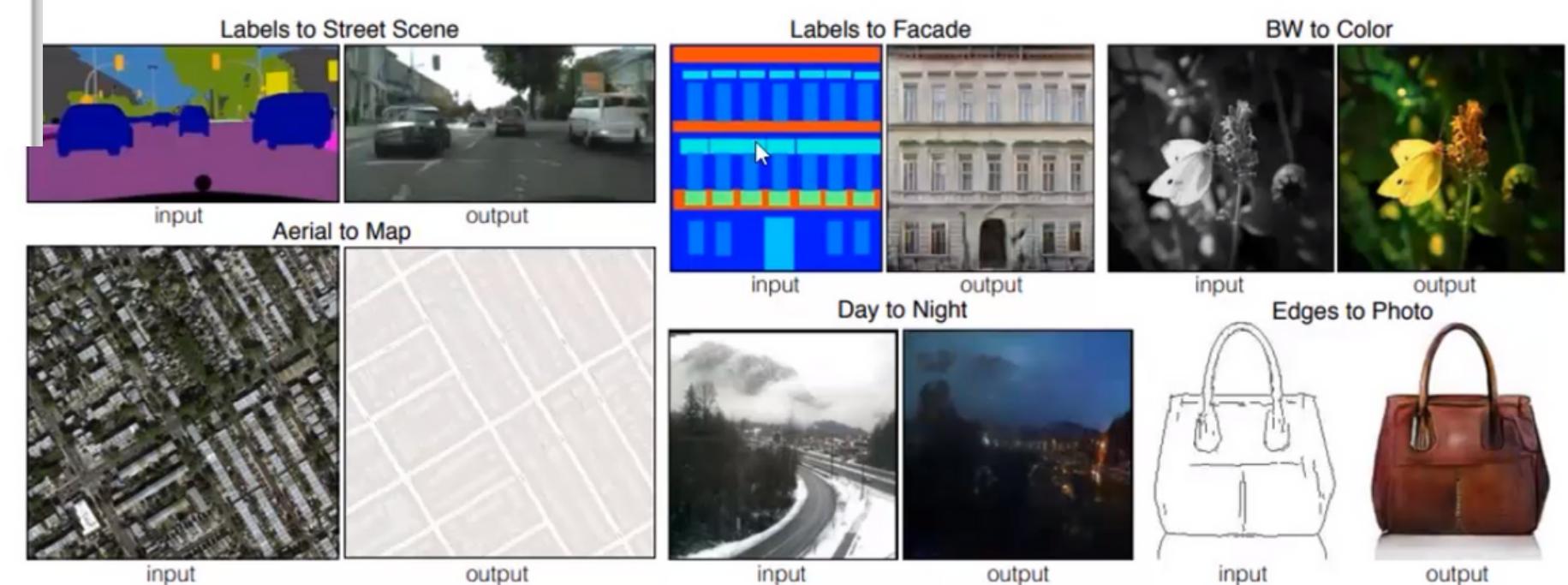
Color
Sketch

11

Çözünürlük Arttırma:



Görüntüden Görüntüye Dönüşümme:



12



Bir yapay zeka programı tarafından yaratılan bir sanat eseri açık artırma bin dolara (2,4 milyon TL) satıldı.

Satış rakamı, portre için tahmin edilen 7-10 bin dolar aralığını kat kat aştı.

New York'taki Christie's müzayedede evinde satılan Edmond Belamy'nin Portresi, Paris merkezli Obvious (Aşikar) adlı sanat kolektifi tarafından geliştirilen yapay zeka programının imzasını taşıyor.

Yapay zeka algoritması 14. ve 20. yüzyıllar arasında yapılan 15 bin portreyi bir veri tabanına sahip.



NEW
NEW
NEW
NEW

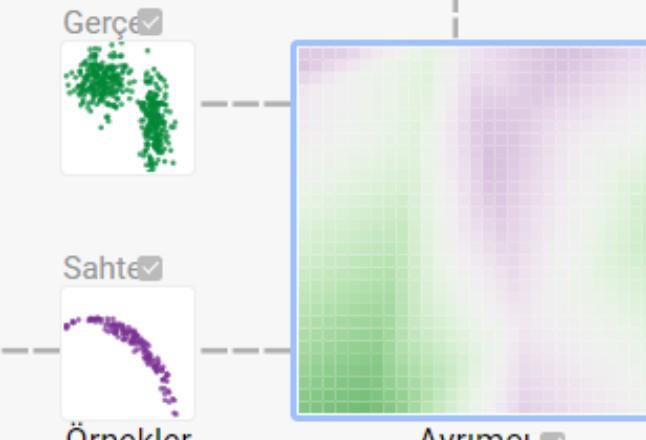
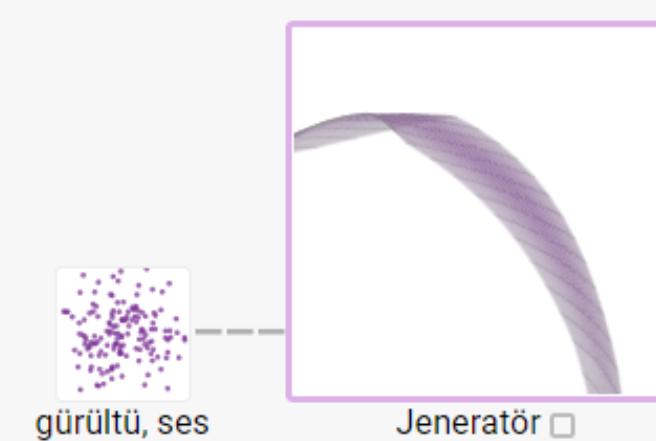
GAN Laboratuvarı

Veri Dağılımı

 Önceden eğitilmiş modeli kullanın

Dönem
000.758

MODELE GENEL BAKIŞ GRAFIĞI



Gradyanlar Dönem başına 2 güncelleme

Optimizer: SGD

Öğrenme oranı:

0.1

Log loss

Ayrımcı kaybı

Jeneratör kaybı

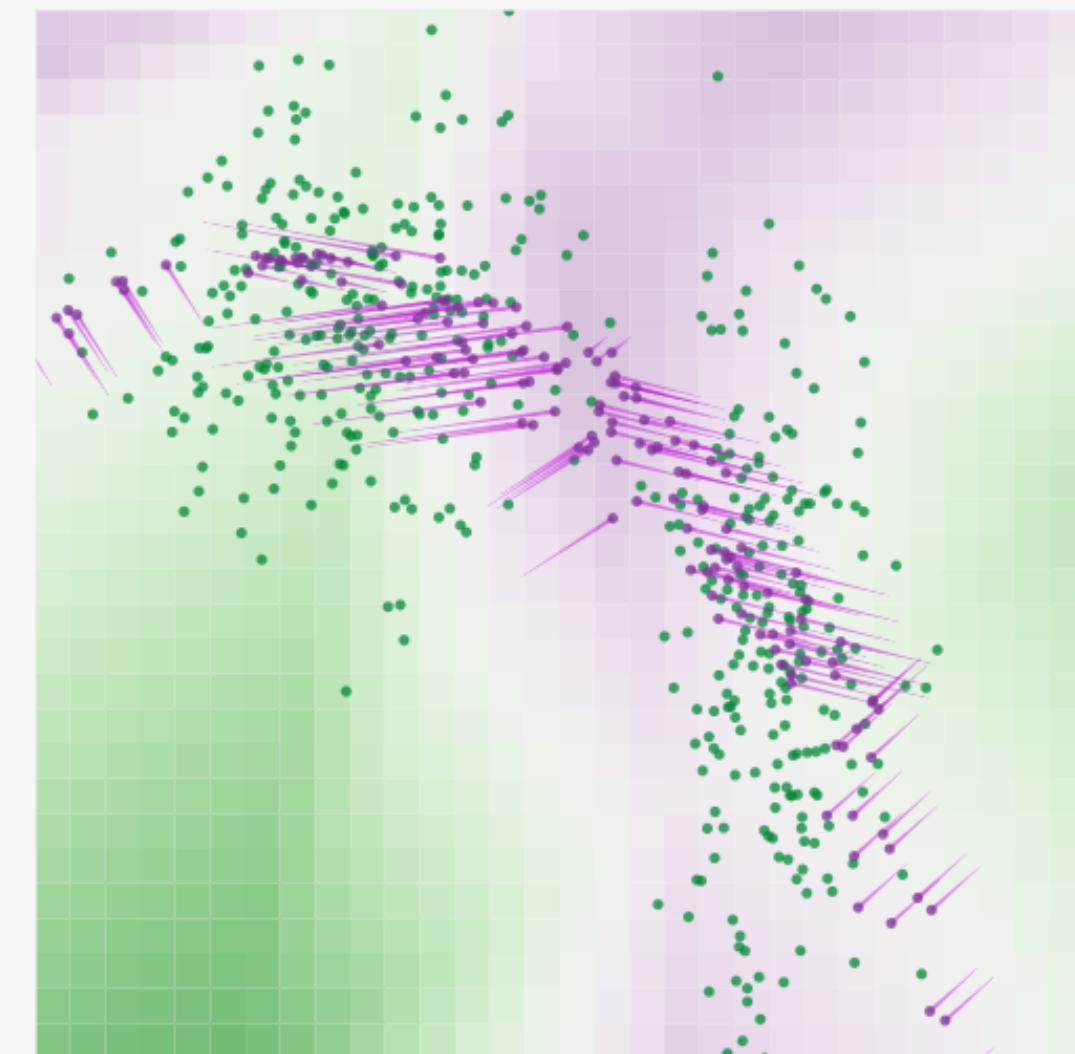
1 gizli katman 8 nöron (lar)

1 gizli katman 8 nöron (lar)

Dönem başına 1 güncelleme

Gradyanlar

KATMANLI DAĞILIMLAR



Her nokta bir 2B veri örneğidir: [gerçek örnekler](#); [sahte örnekler](#).

Izgaralarının arka plan renkleri, [ayırıcı](#) edicinin sınırlandırmalarını temsil eder.

İçinde Numuneler [vesil bölgelere](#) gerçek olması muhtemeldir: bu [mor bölgelerde](#)

METRIKLER

Discriminator's Loss

Generator's Loss



KL Divergence (by grid)

JS Divergence (by grid)



Gereken kütüphanlerin yüklenmesi

```
[6] %matplotlib inline
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
from tqdm import tqdm
print('Tensorflow version:', tf.__version__)

[26] def show(images, n_cols=None):
    n_cols = n_cols or len(images)
    n_rows = (len(images) - 1) // n_cols + 1
    if images.shape[-1] == 1:
        images = np.squeeze(images, axis=-1)
    plt.figure(figsize=(n_cols, n_rows))
    for index, image in enumerate(images):
        plt.subplot(n_rows, n_cols, index + 1)
        plt.imshow(image, cmap="binary")
        plt.axis("off")
```

14

Kodla GAN Örneği

15

Verilerin yüklenmesi ve önişlemlerin gerçekleştirilemesi

Bu çalışmada Fashion MNIST veri kümesini kullanacağız ve bu veri kümesinde çok çeşitli ayakkabı, bot, sandalet, tişört, kaban, bandana vb. tekstil ürününden etiketli örnekler yer almaktadır.

```
[25] #Fashion MNIST veri kümesinin keras yoluyla indiriyoruz.  
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.fashion_mnist.load_data()  
x_train = x_train.astype(np.float32) / 255.0  
x_test = x_test.astype(np.float32) / 255.0
```

```
# Veri kümelerinden 10x10 piksel büyülüklü 25 tane örnek ekranaya yazdırıp neye benzediğine bakalım.  
plt.figure(figsize=(10,10))  
for i in range(25):  
    plt.subplot(5,5,i+1)  
    plt.xticks([])  
    plt.yticks([])  
    plt.grid(False)  
    plt.imshow(x_train[i], cmap=plt.cm.binary)  
plt.show()
```



```
[23] batch_size = 32
    # Bu veri kümesi, bir arabelleği buffer_size elemanları ile doldurur,
    #ardından seçilen elemanları yeni elemanlarla değiştirerek rastgele örnekler.
dataset = tf.data.Dataset.from_tensor_slices(x_train).shuffle(1000)
    # Bu veri kümesinin ardışık öğelerini toplu olarak birleştirir.
dataset = dataset.batch(batch_size, drop_remainder=True).prefetch(1)
```

Eğitim verilerinin oluşturulması

```
[22] # ÜRETİCİ KATMANINDAKİ EVRİŞİMLİ SİNİR AĞI
num_features = 100 # öznitelik sayısı

# giriş değerini verdigimiz features sayısına göre başlatıyoruz
# Conv2DTranspose versiyonunu kullanıyoruz.
generator = keras.models.Sequential([
    keras.layers.Dense(7 * 7 * 128, input_shape=[num_features]),
    keras.layers.Reshape([7, 7, 128]),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2DTranspose(64, (5,5), (2,2), padding="same", activation="selu"),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2DTranspose(1, (5,5), (2,2), padding="same", activation="tanh"),
```



```
[27] noise = tf.random.normal(shape=[1, num_features])
generated_images = generator(noise, training=False)
show(generated_images, 1)
```

DCGAN için üretici (generator) ağının oluşturulması



18

DCGAN için ayırıcı (discriminator) ağının oluşturulması

```
▶ discriminator = keras.models.Sequential([  
    keras.layers.Conv2D(64, (5,5), (2,2), padding="same", input_shape=[28, 28, 1]),  
    keras.layers.LeakyReLU(0.2),  
    keras.layers.Dropout(0.3),  
    keras.layers.Conv2D(128, (5,5), (2,2), padding="same"),  
    keras.layers.LeakyReLU(0.2),  
    keras.layers.Dropout(0.3),  
    keras.layers.Flatten(),  
    keras.layers.Dense(1, activation='sigmoid')  
])  
  
[28] # Üretilen görsel için ayırt edici %50nin altında bir değer üretti ilk adım için  
decision = discriminator(generated_images)  
print(decision)  

```

19

DCGAN için
ayırıcı
(discriminator)
ağının
oluşturulması

```
[29] discriminator.compile(loss="binary_crossentropy", optimizer="rmsprop")
discriminator.trainable = False
gan = keras.models.Sequential([generator, discriminator])
gan.compile(loss="binary_crossentropy", optimizer="rmsprop")
```

```
[30] from IPython import display
     from tqdm import tqdm
     seed = tf.random.normal(shape=[batch_size, 100])
```

```
▶ from tqdm import tqdm
def train_dcgan(gan, dataset, batch_size, num_features, epochs=5):
    generator, discriminator = gan.layers
    for epoch in tqdm(range(epochs)):
        print("Epoch {}/{}".format(epoch + 1, epochs))
        for X_batch in dataset:
            noise = tf.random.normal(shape=[batch_size, num_features])
            generated_images = generator(noise)
            X_fake_and_real = tf.concat([generated_images, X_batch], axis=0)
            y1 = tf.constant([[0.]] * batch_size + [[1.]] * batch_size)
            discriminator.trainable = True
            discriminator.train_on_batch(X_fake_and_real, y1)
            noise = tf.random.normal(shape=[batch_size, num_features])
            y2 = tf.constant([[1.]] * batch_size)
            discriminator.trainable = False
            gan.train_on_batch(noise, y2)
            # Üretilen görüntüleri ekrana yazdırıp doyaya kaydedelim
            display.clear_output(wait=True)
            generate_and_save_images(generator, epoch + 1, seed)

            display.clear_output(wait=True)
            generate_and_save_images(generator, epochs, seed)
```

DCGAN için ayırıcı (discriminator) ağının oluşturulması

```
[32] ## Kaynak: https://www.tensorflow.org/tutorials/generative/dcgan#create\_a\_gif
def generate_and_save_images(model, epoch, test_input):
    # 'Eğitim' False seçeneğine ayarlandı.
    # Böylece tüm katmanlar çıkarım modunda (batchnorm) çalışır.
    predictions = model(test_input, training=False)

    fig = plt.figure(figsize=(10,10))

    for i in range(25):
        plt.subplot(5, 5, i+1)
        plt.imshow(predictions[i, :, :, 0] * 127.5 + 127.5, cmap='binary')
        plt.axis('off')

    plt.savefig('image_at_epoch_{:04d}.png'.format(epoch))
    plt.show()
```

```
[33] # Eğitim için yeniden boyutlandırmanın yapılması
x_train_dcgan = x_train.reshape(-1, 28, 28, 1) * 2. - 1.
```

```
[34] #Batch size boyutunun ve shuffle özelliklerinin belirlenmesi
batch_size = 32
dataset = tf.data.Dataset.from_tensor_slices(x_train_dcgan)
dataset = dataset.shuffle(1000)
dataset = dataset.batch(batch_size, drop_remainder=True).prefetch(1)
```

DCGAN'ın eğitilmesi

```
# Eğitim gan modeli, tanımlanan dataset için belirlenen batch_size boyutu ve öznitelik sayısı ile  
# 10 epoch olarak gerçekleşecek, siz de epoch sayısını değiştirmerek modelin gelişimini gözlemlerebilrisiniz.  
%%time  
train_dcgan(gan, dataset, batch_size, num_features, epochs=10)
```



CPU times: user 6min 9s, sys: 18.1 s, total: 6min 27s
Wall time: 7min 1s

23

DCGAN ile sentetik görüntülerin oluşturulması

```
noise = tf.random.normal(shape=[batch_size, num_features])  
generated_images = generator(noise)  
show(generated_images, 8)
```



KODLAR:

<https://colab.research.google.com/drive/1QJmchKBYUSTt3cleUAezMeAA0WENTBEO?usp=sharing>
Presentations are communication tool
https://colab.research.google.com/github/ayyucekizrak/GAN_UreticiCekismeliAglar_ile_SentetikVeriUretme/blob/master/DCGAN_ile_Keras_Kullanarak_Sentetik_Goruntu__Olusturulmasi.ipynbs.

<https://medium.com/@devnag/generative-adversarial-networks-gans-in-50-lines-of-code-pytorch-e81b79659e3f>

[https://medium.com/@muhammedbuyukkinaci/generative-adversarial-networks-gan-nedir-türkçe-5819fe9c1fa7](https://medium.com/@muhammedbuyukkinaci/generative-adversarial-networks-gan-nedir-t%C3%BCrk%C3%A7e-5819fe9c1fa7)

<https://www.tensorflow.org/tutorials/generative/dcgan>

<https://www.btkakademi.gov.tr/portal/course/keras-ile-derin-ogrenmeye-giris-10599#/about>

<https://www.bbc.com/turkce/haberler-dunya-45987201>

<https://stackabuse.com/introduction-to-gans-with-python-and-tensorflow/>

<https://www.google.com/url?q=https://aylien.com/blog/introduction-generative-adversarial-networks-code-tensorflow&usg=AOvVaw3jiVe3A9QQuV9rhCzeG15i>

https://www.google.com/url?q=https://wiki.pathmind.com/generative-adversarial-network-gan&usg=AOvVaw2LuNSJXFMC2_KjAEQIKfAY

<https://realpython.com/generative-adversarial-networks/>