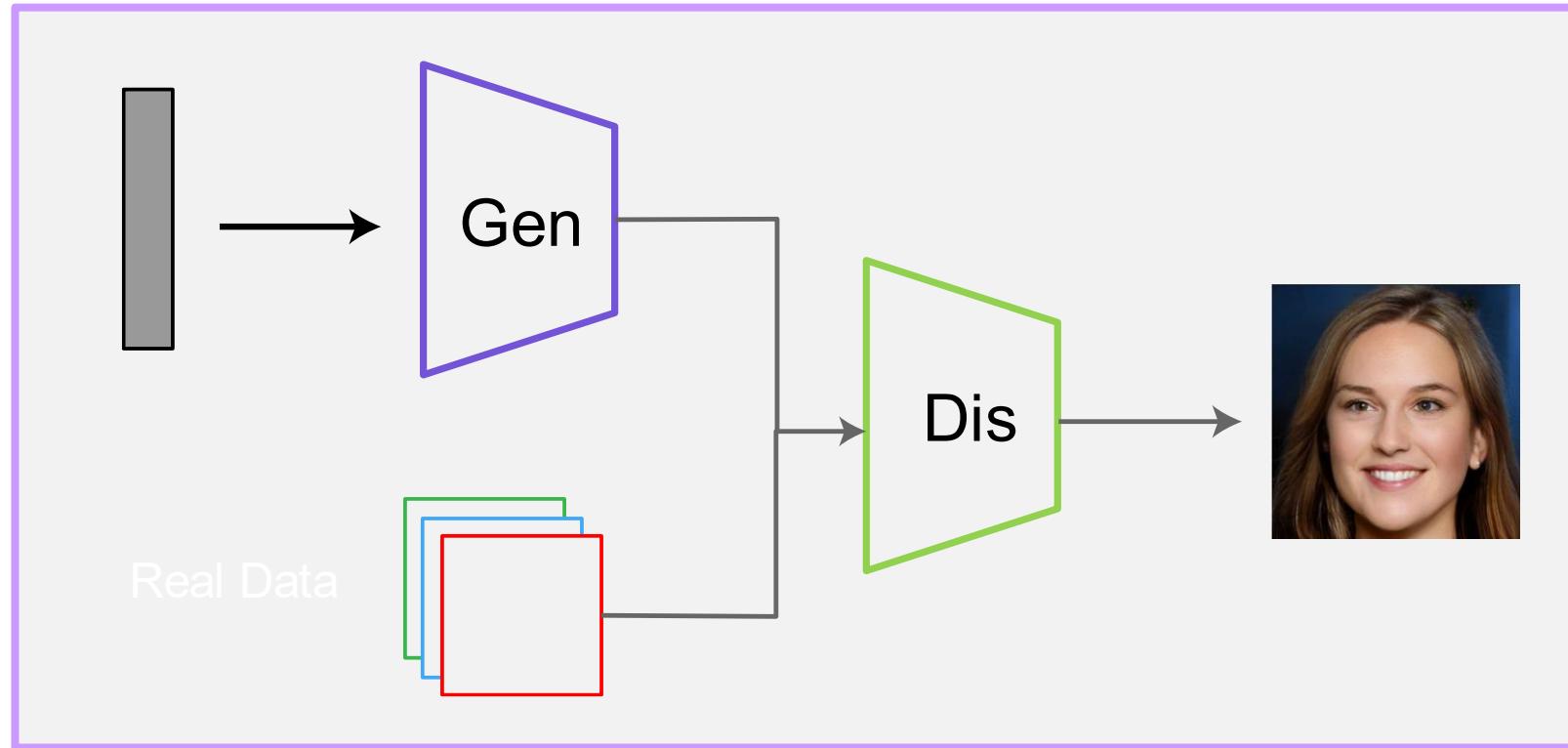
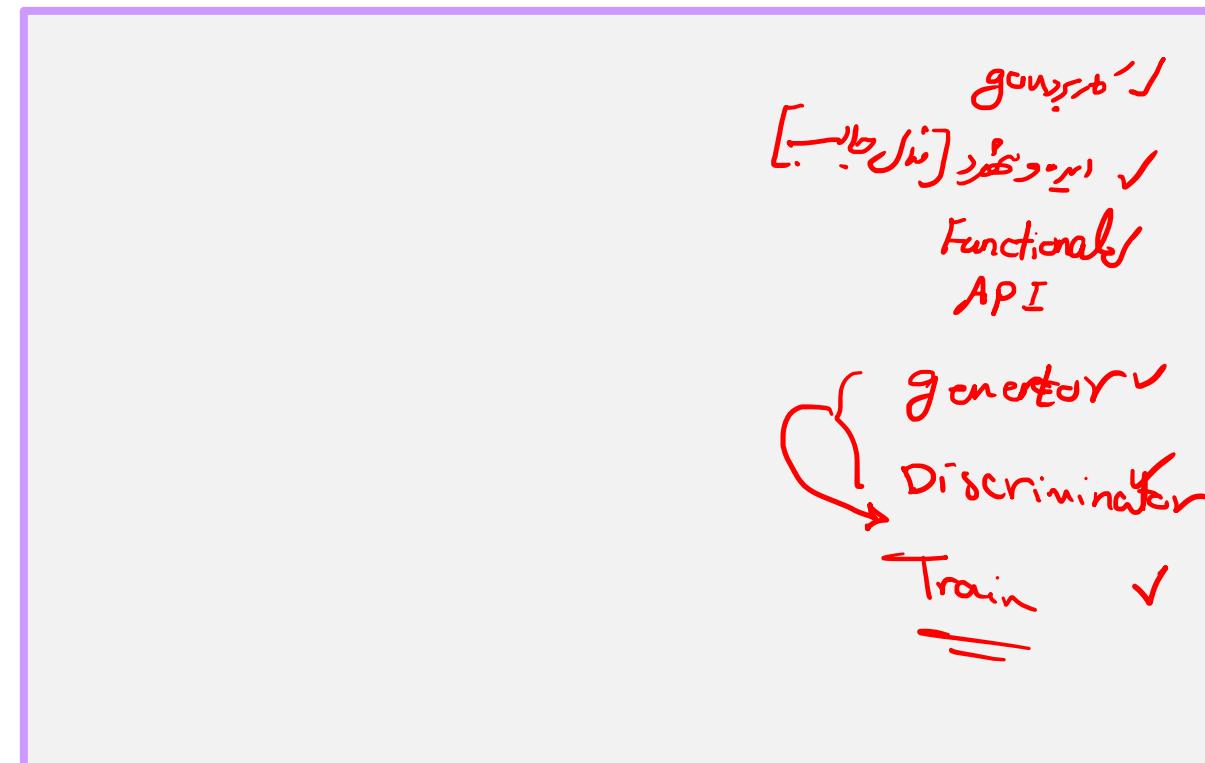


کارگاه شبکه GAN (مدل های مولد تخصصی)



آنچه امروز خواهیم گفت :



مدل های مولد - Generative Model





Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, 2015.



Progressive Growing of GANs for Improved Quality, Stability, and Variation, 2017

پیشرفت از 2014 تا 2017



2014



2015



2016



2017

This Person Does not Exist



Towards the Automatic Anime Characters Creation with Generative Adversarial Networks

pix2pix

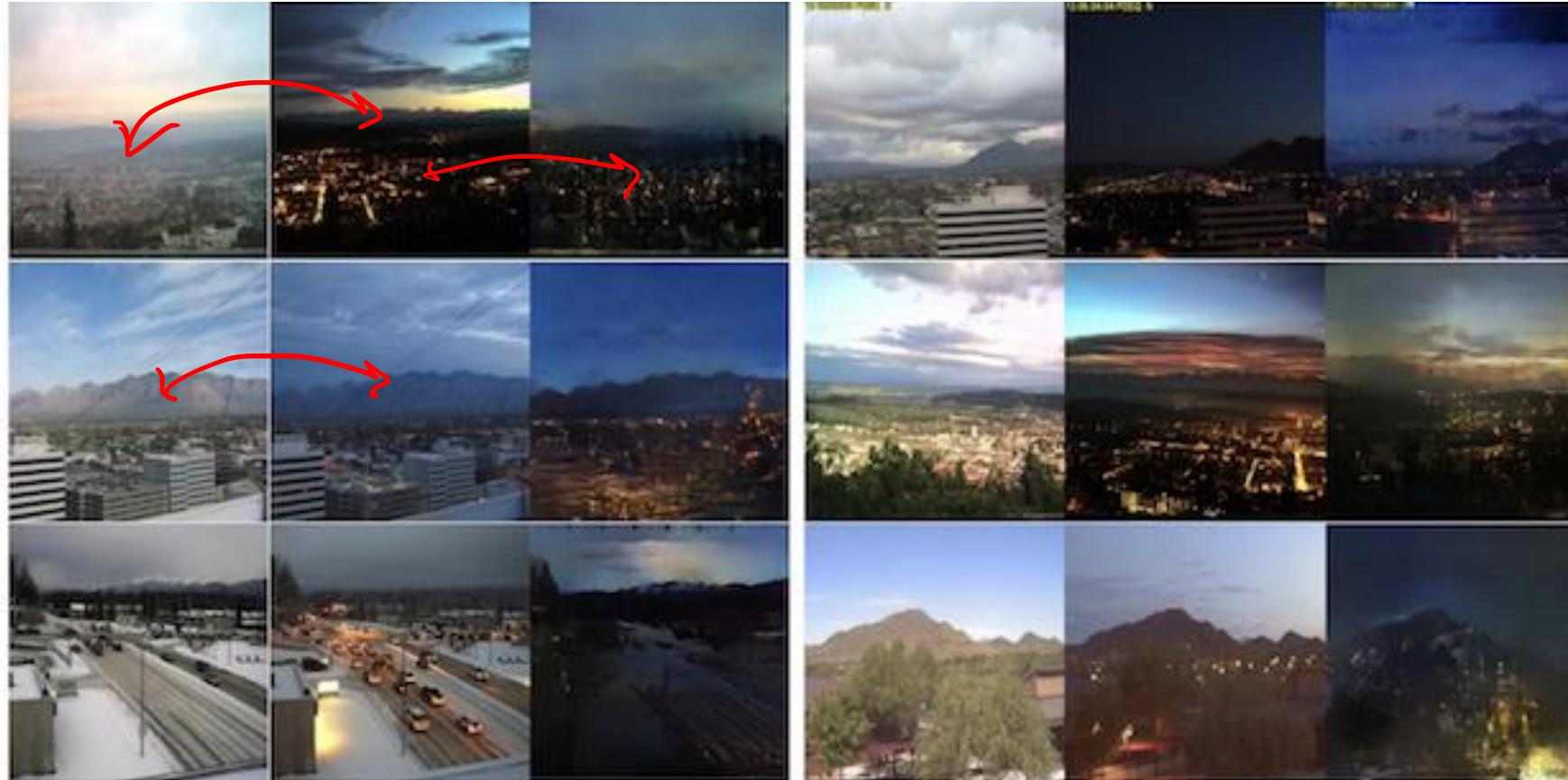


Image-to-Image Translation with **Conditional Adversarial Networks**

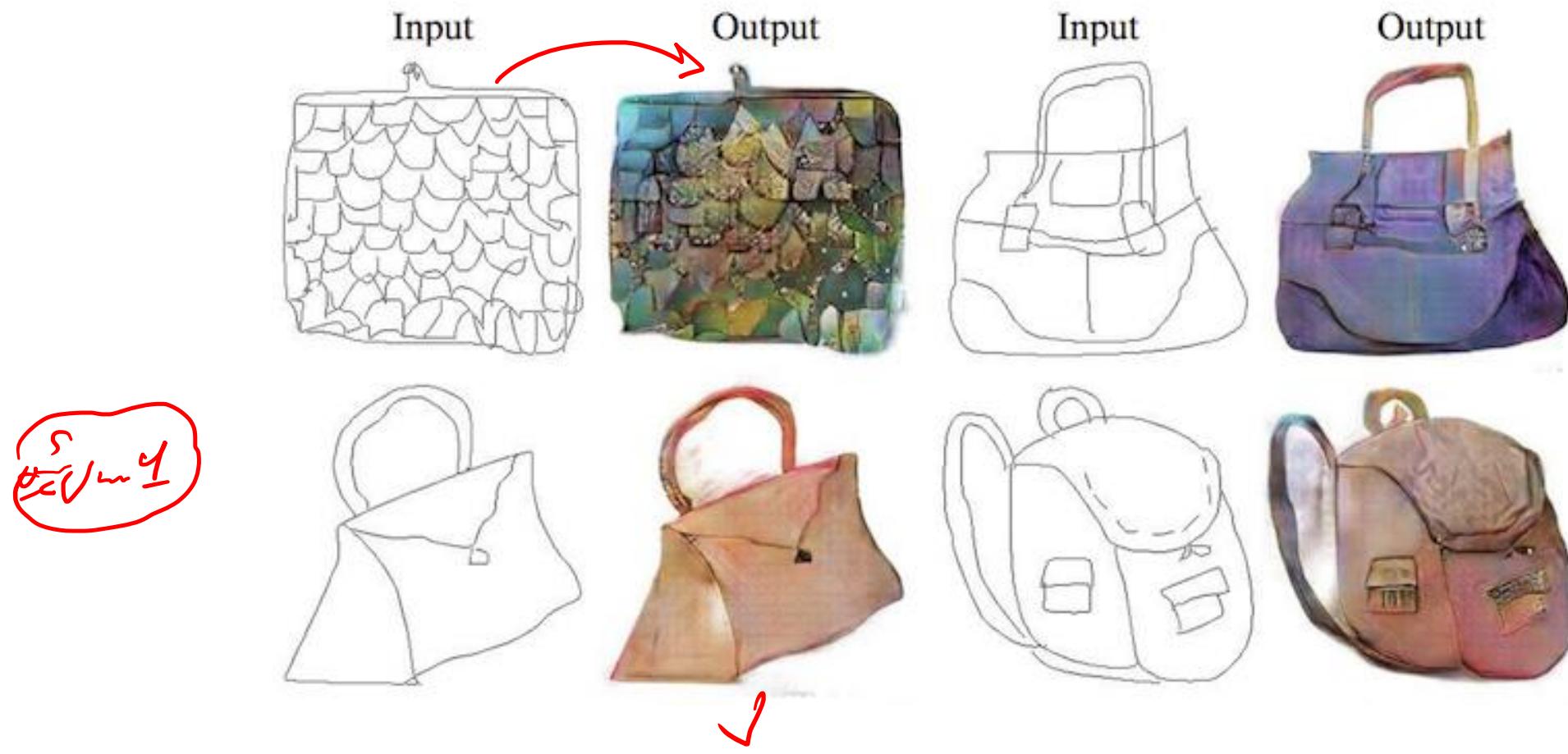
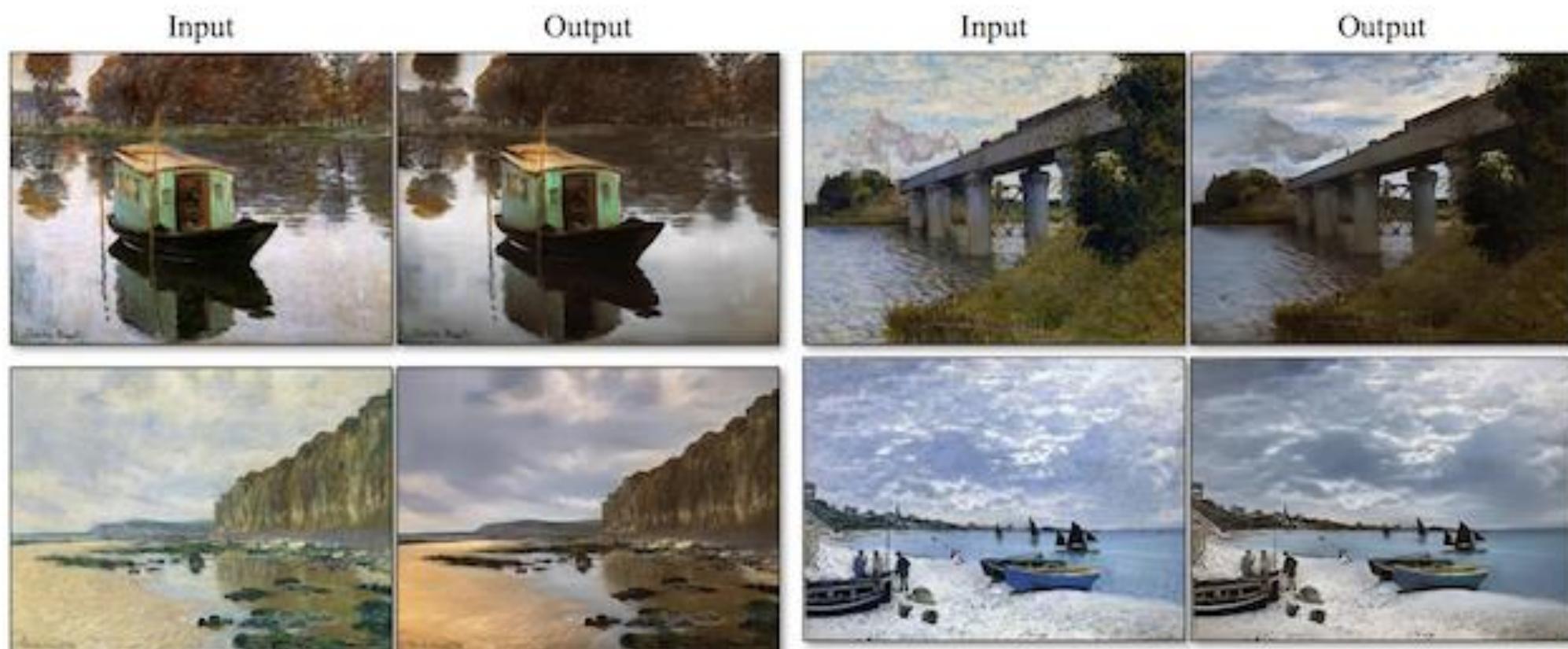
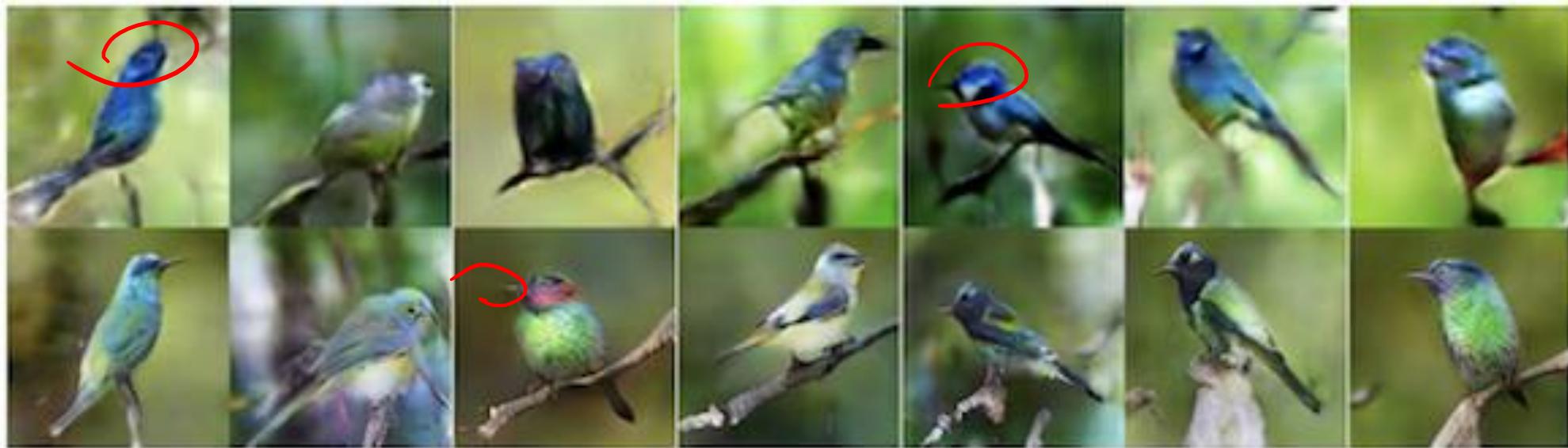


Image-to-Image Translation with **Conditional Adversarial Networks**



Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks



Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks, 2016.

Face Frontal View Generation



Global and Local Perception GAN for Photorealistic and Identity Preserving Frontal View Synthesis

Photos to Emojis

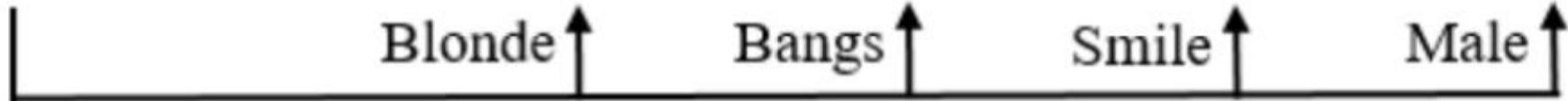
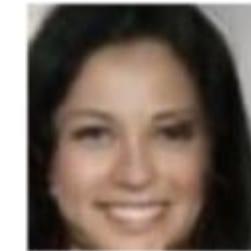


Photograph Editing

Real image



Reconstructed images



Invertible Conditional GANs For Image Editing, 2016.

ایده شبکه های GAN

هاجرای من و کره!

برگ ناله

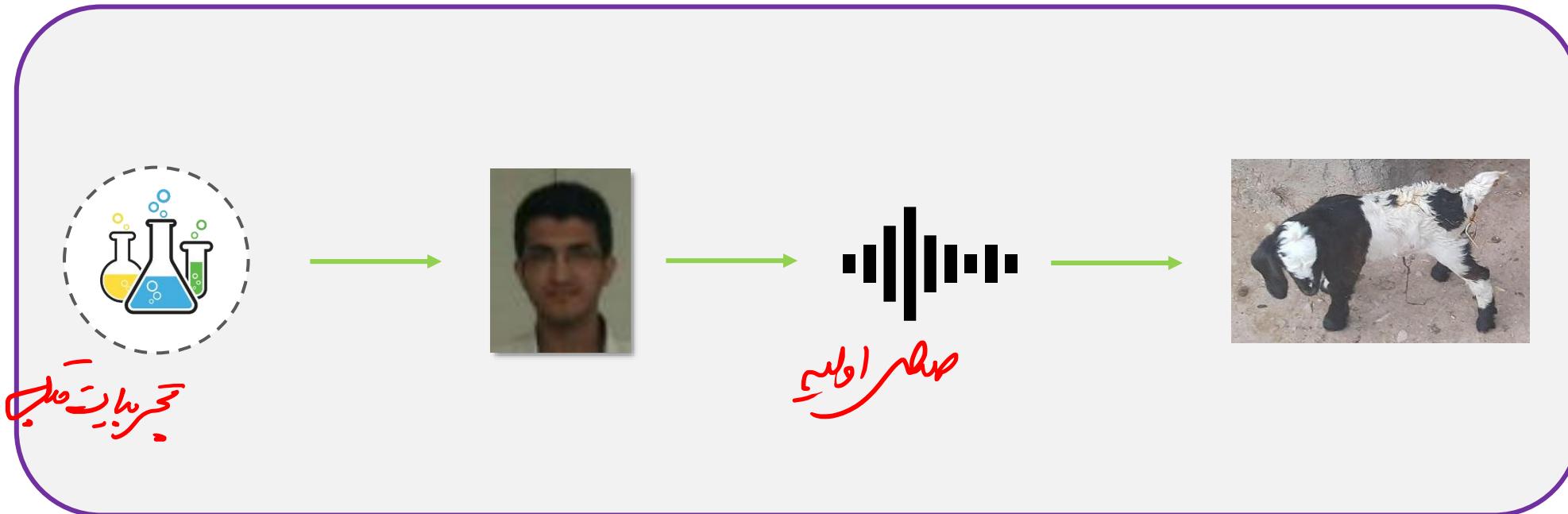


من چگونه می توانستم با این کره ارتباط برقرار کنم ؟



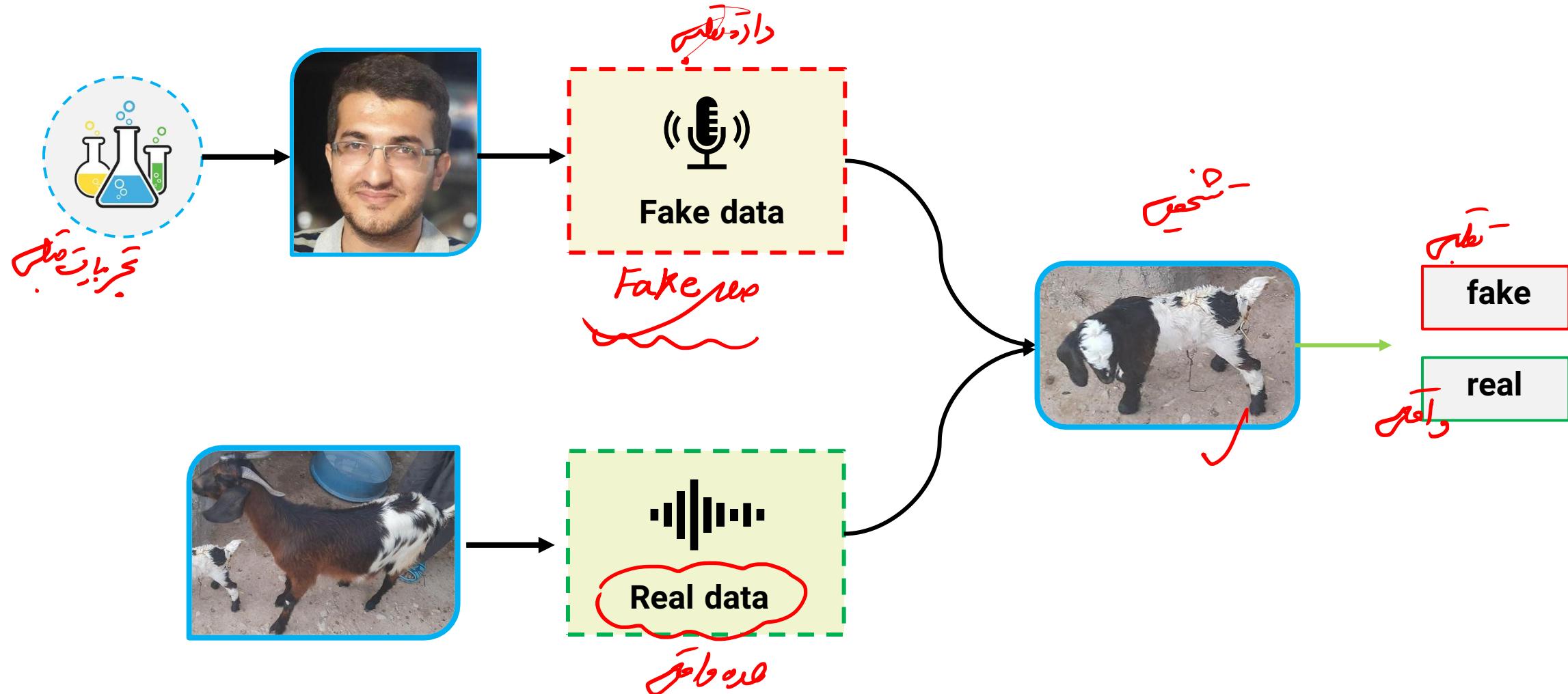
طبعتا باید صدای نزدیک به صدای که هادرش تولید میکرد تولید کنم !

اولین تلاش های من برای برقراری ارتباط



البته موفق آمیز نبود و نمی توانستم ارتباطی برقرار کنم !

دیاگرام کاہل ارتباٹ با کرہ !



حال دو حالت پیش می آید !

حالت اول - کره گول میخورد !



محبت هادرانه انتظار داشت که اتفاق نمی افتاد و میفهمید

سوتی داده !



از این پس به **صدای هادرش** دقต می کرد و دقت خود را

بالاتر می برد. (در ادبیات ما **Train** میکرد خودش را !)

کره میخورد

حالت دوم - کره گول نمیخورد !



من نیاز داشتم دوباره به **صدای هادرش** دقت کنیم و سعی

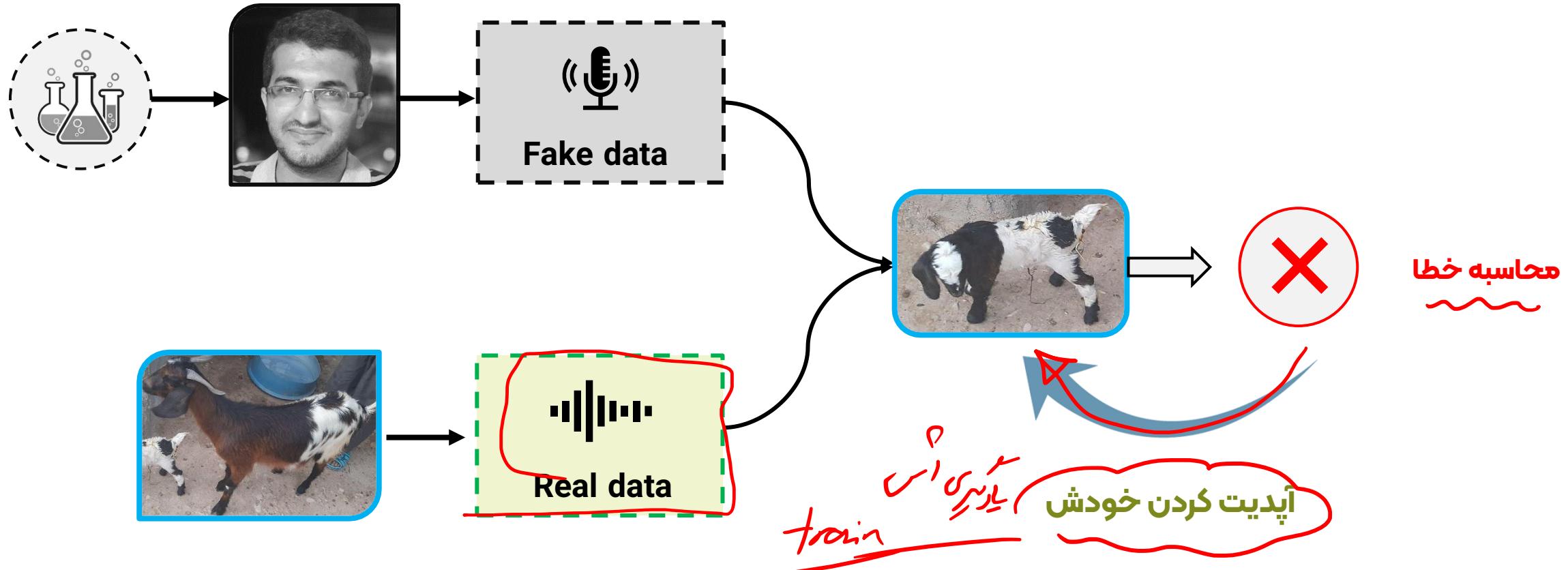
کنم صدایی شبیه به **هادرش** تولید کنم.



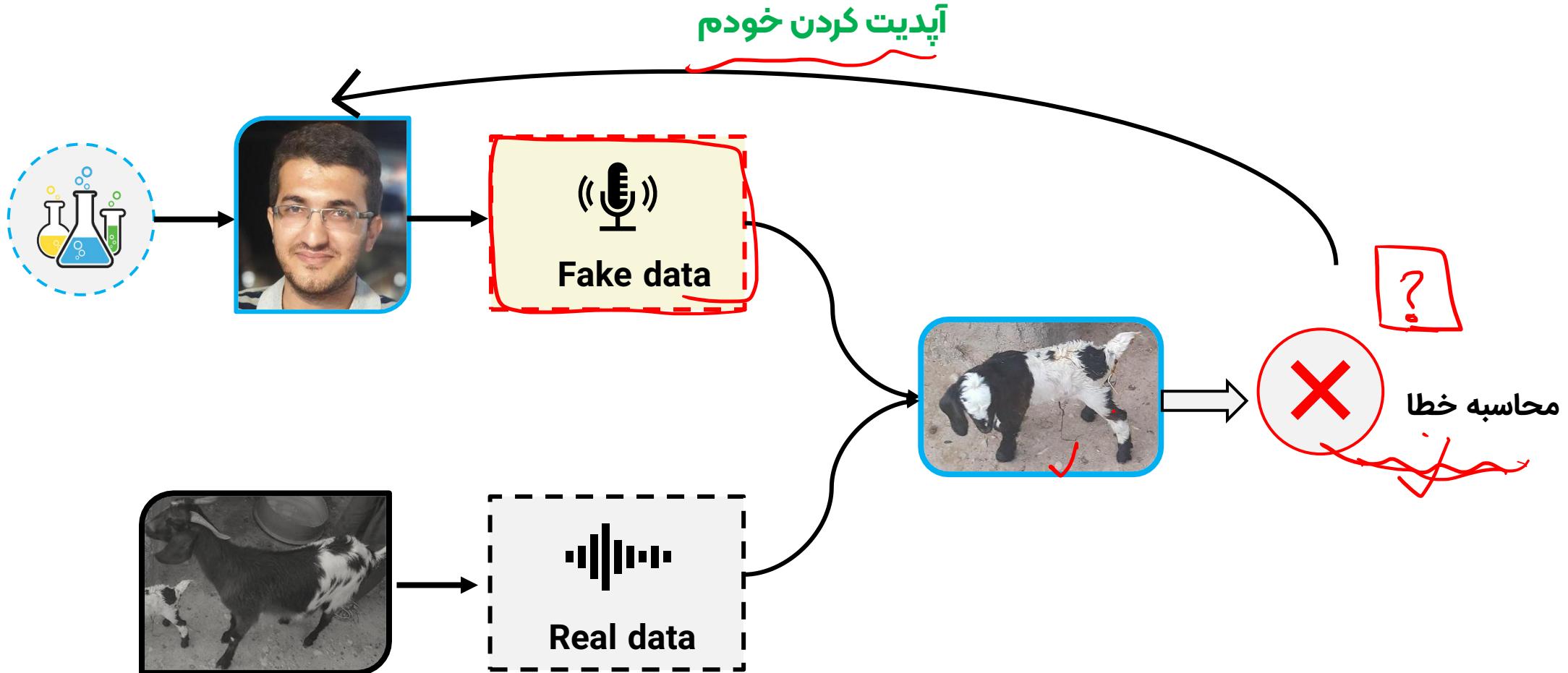
اصطلاحا خودم را **Train** کنم !



حالت اول - کره گو^ل میخورد!



حالت دوم - کره گول نمیخورد!



نکته مهم:

هدف من این بود که خطای کره را زیاد کنم و هدف کره این بود که خطای خودش را کم کند. او به خطای خودش نگاه میکرد و من هم به خطای او . هر دو ما به یک خطانگاه میکردیم اما با دو هدف متفاوت !

اداوه این داستان ...

سکانس ۳

به مرور کره فهمید که من دارم گوش می
زنم و به همین علت دوباره صدای فیک من
را تشخیص داد و دوباره من نتوانستم با او
ارتباط برقرار کنم

سکانس ۲

به مرور توانستم **مهارت بیشتری** به دست
آورم و این بچه کره را گول بزنم و با او ارتباط
برقرار کنم ولی محبت هادرانه را طبیعتاً
نداشتم!

سکانس ۱

در روزهای اول کره به هیچ عنوان **گول نمی خورد** و من اصلا هیچ ارتباطی **نمی توانستم**
با او برقرار کنم و **تلاش کردم خود را آپدیت**
کنم.

ادامه این داستان ...

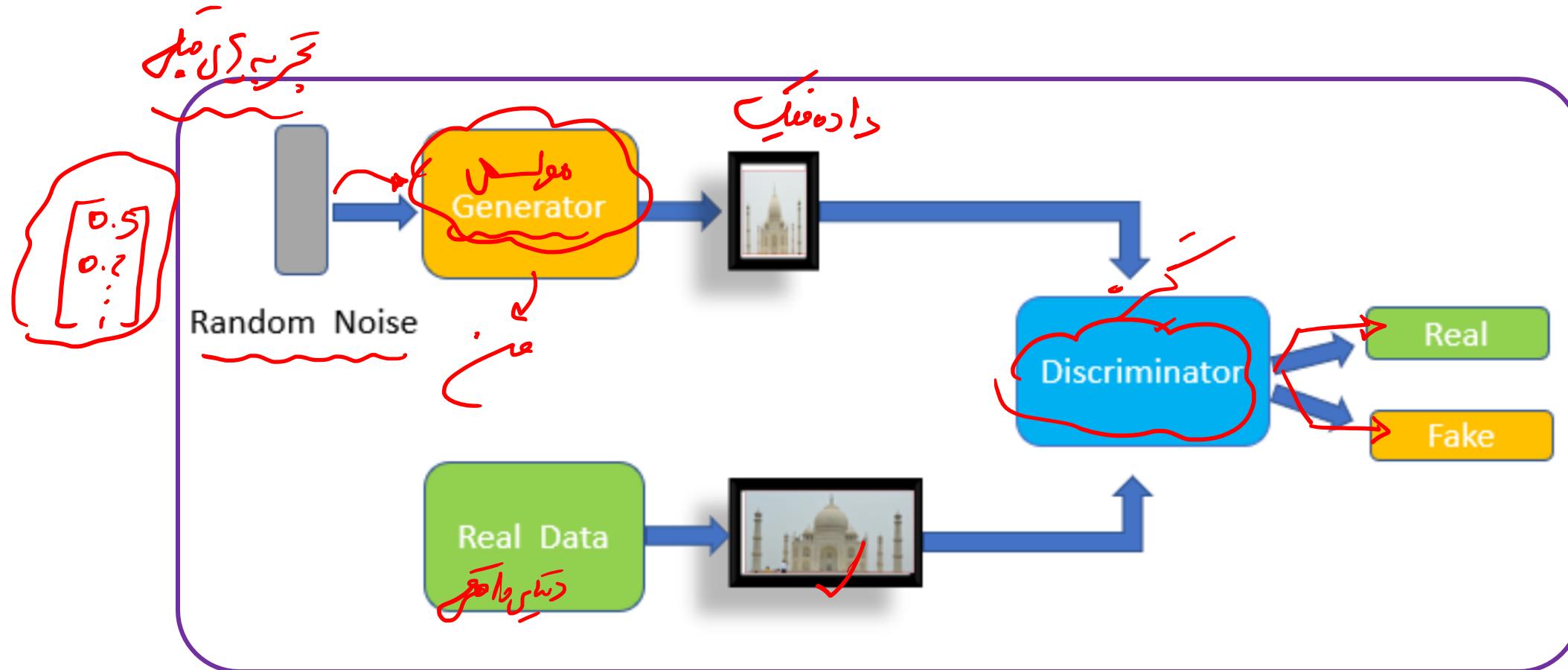
و این رقابت هر روز ادامه داشت. هر بار هن بهتر تقلید می کردم و او را گول میزدم و بار دیگر او صدای هادرش را بهتر می شناخت.

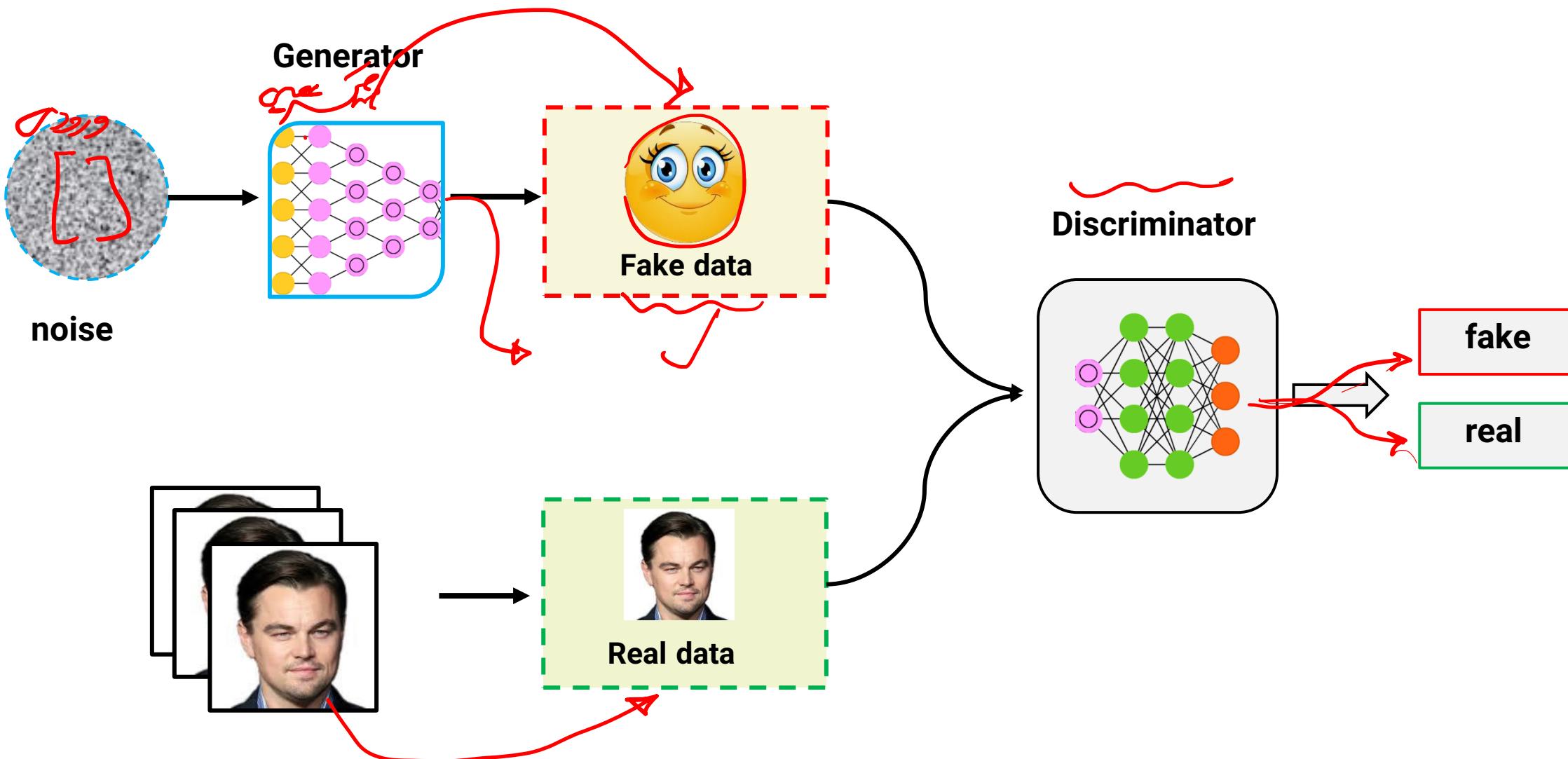
اگر این کار تا بی نهایت ادامه داشت احتمالا هن بهترین تقلید کننده و بچه گره به عنوان بهترین تشخیص دهنده در دنیا می شدیم!

این رقابت باعث شده بود هر دوی ها روز به روز پیشرفت کنیم و بهتر شویم!

معرفی شبکه های GAN

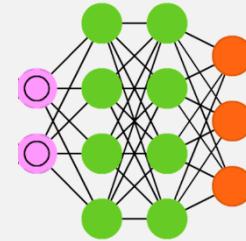
ایده شبکه های GAN





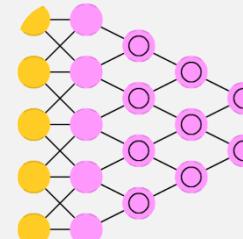
دو جزء اصلی شبکه GAN

Discriminator شبکه



تشخیص داده های تقلیبی که شبکه **Generator** تولید می کند.

Generator شبکه



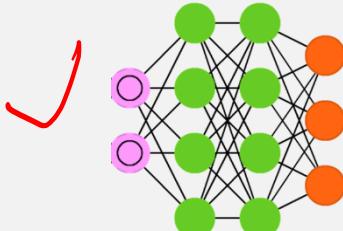
تولید داده به صورتی که **Discriminator** را به اشتباه بیندازد.

با رقابت این دو شبکه با یکدیگر هر دو ارتقا پیدا می کنند.

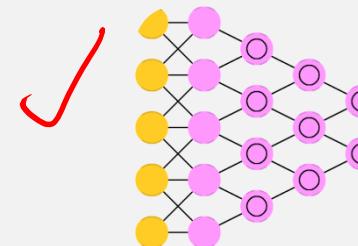
حال چند نکته :

آموزش شبکه GAN

Discriminator شبکه



Generator شبکه

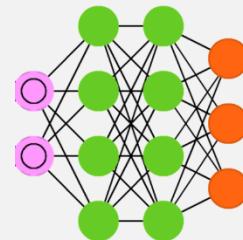


نکته مهم : در هنگام آموزش هر کدام از این شبکه ها، **وزن های شبکه دیگری ثابت است.**

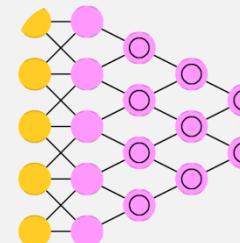


نکته : هر دوی این شبکه ها خودشان را به کمک خطای **Discriminator** آموزش می دهند !

Discriminator شبکه



Generator شبکه



آموزش شبکه Discriminator

شبکه **Discriminator** می گوید هن چقدر توانسته ام داده های حقیقی را از داده های **Fake** تشخیص دهم. طبق آن خودم را آپدیت می کنم !

اگر دقت پایینی در تشخیص داشته ام که کلا باید خودم را آپدیت کنم و بالعکس !

آموزش شبکه Generator

شبکه Generator می گوید هن چقدر توانسته ام شبکه Discriminator را گول بزنم. طبق آن می آیم و خودم را آپدیت می کنم!

اگر نتوانسته ام Discriminator را گول بزنم که کلا باید عوض کنم خودم را و اگر توانسته ام که خیلی هم عالی! (و باید Discriminator خودش را عوض کند!)

نتیجه گیری:

شبکه **Generator** به دنبال افزایش خطای **Discriminator** است و شبکه **Discriminator** به دنبال کاهش این خطأ است.



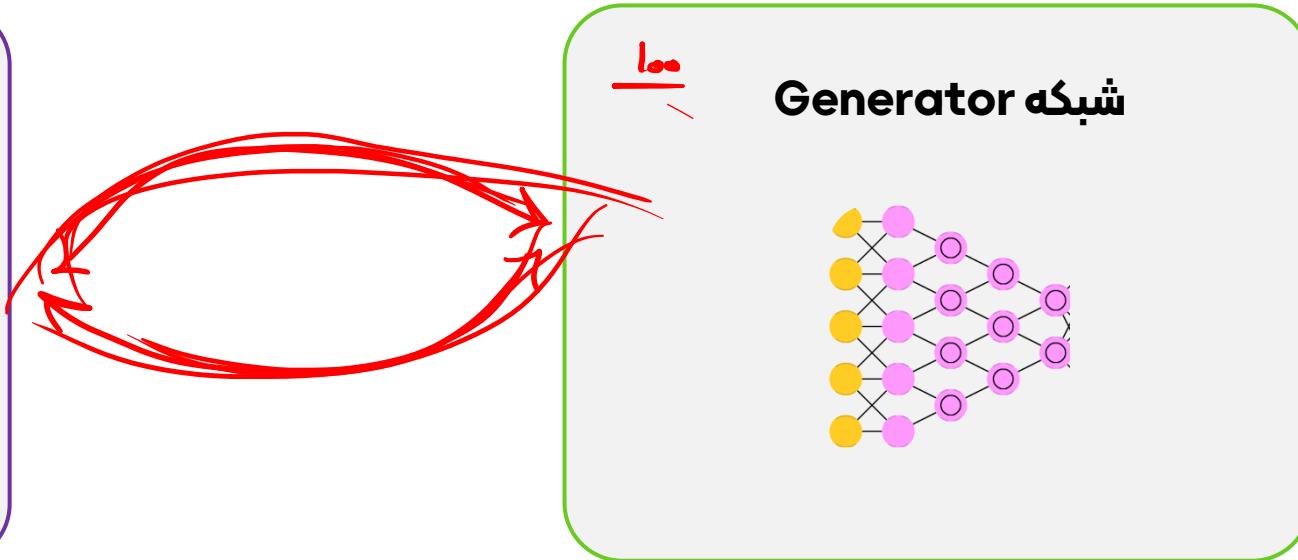
در اینجا با **بهینه سازی رقابتی** روبرو هستیم.

نکات تكميلى و کم کدنويسى ...

~~fit~~
~~epoch=1~~

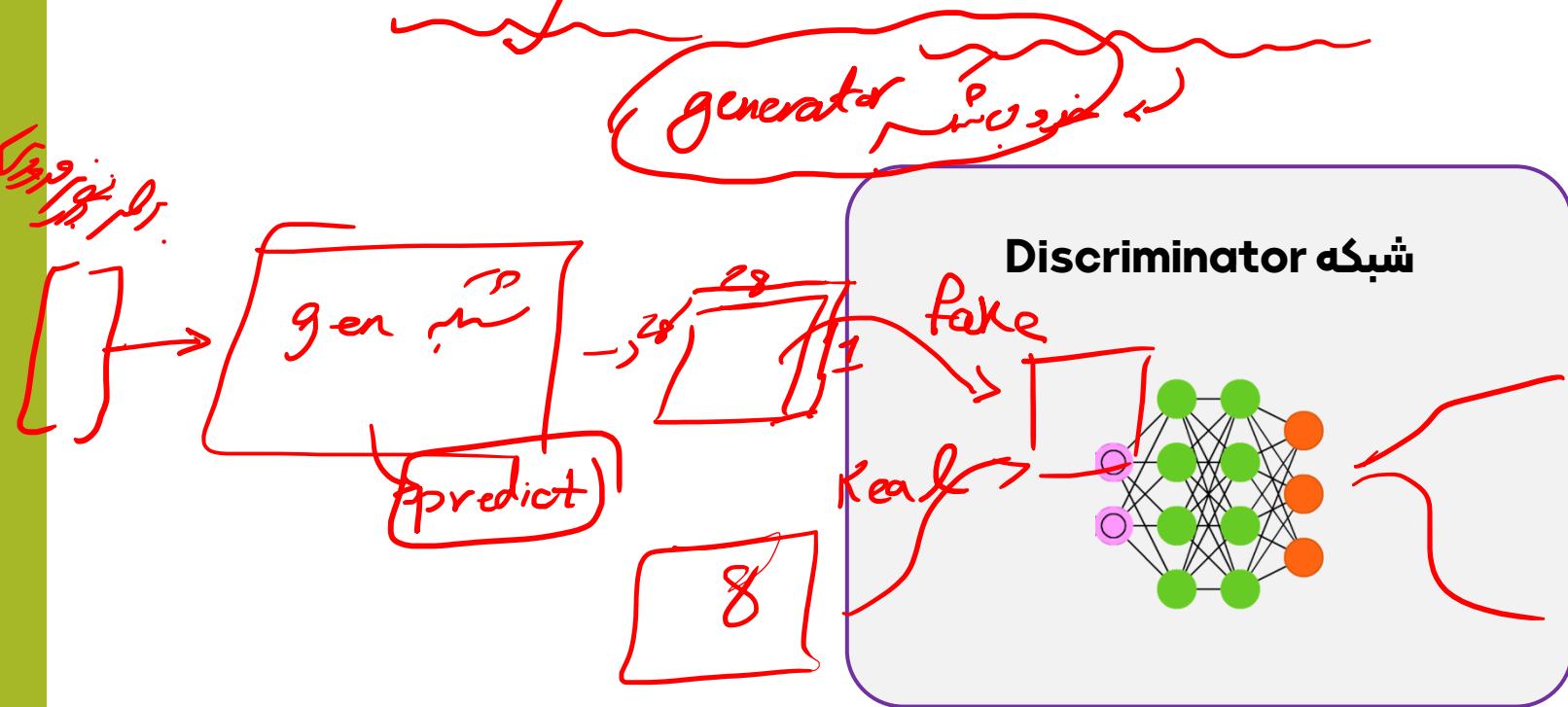
نکته ۱: در فرآیند آموزش این دو هم باید رشد کند

train_on_batch



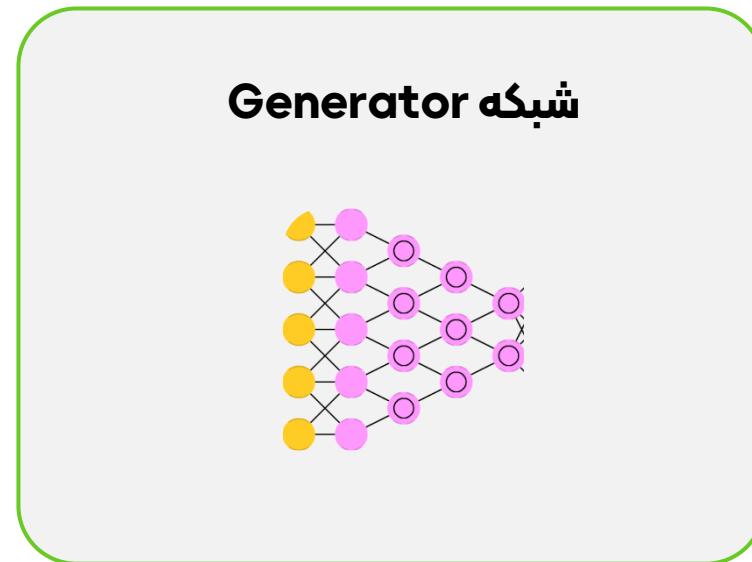
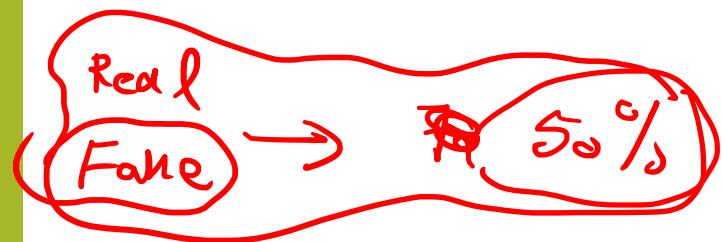
یعنی هتلابه ازای ۳۲ داده شبکه **Discriminator** را آموزش می دهیم، باید به ازای ۳۲ داده شبکه **Generator** را نیز آموزش دهیم.

نکته 2 : شبکه Discriminator هم با داده های Fake آموزش می بیند و هم با داده های Real



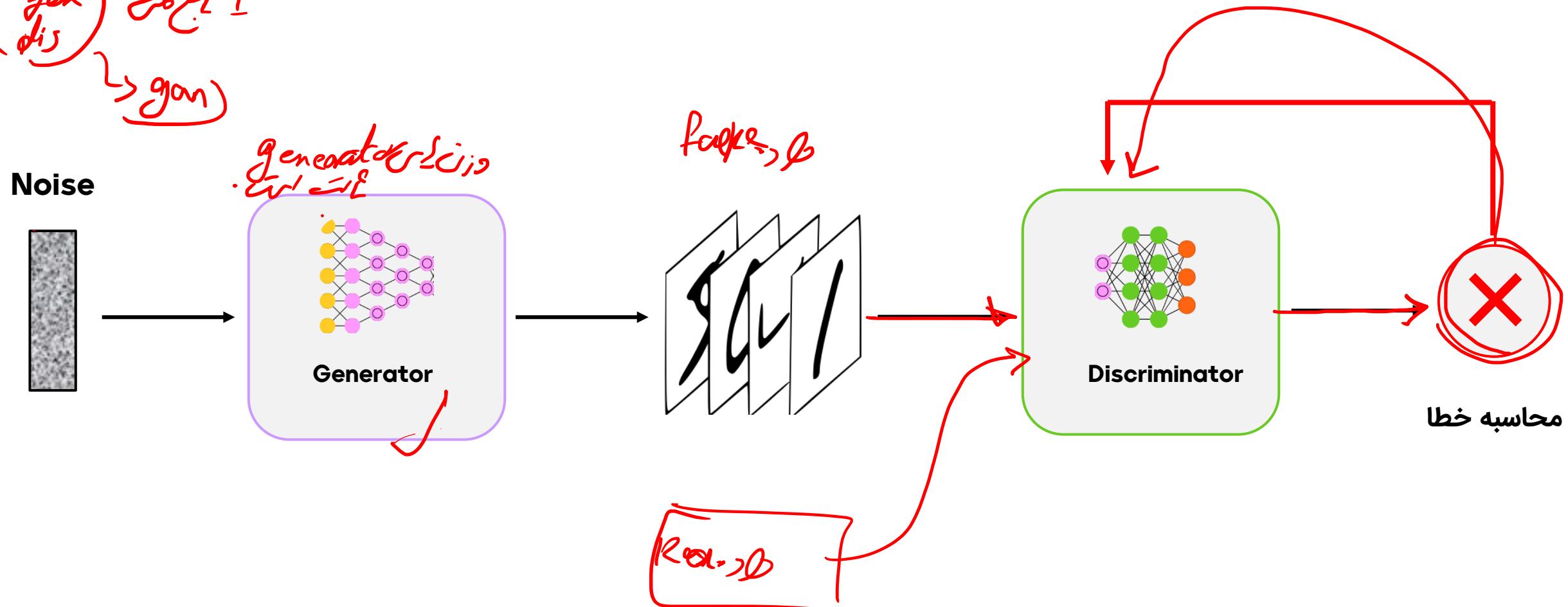
و نکته مهم آن که برای تولید داده های Fake از شبکه Generator ای استفاده می کنیم که تاکنون آموزش دیده است. اگرچه ممکن است داده های خوبی برای ها تولید نکند!

نکته ۳ : شبکه Generator فقط با داده های آمورش میبیند !



چون هی خواهد خودش را ارزیابی کند که چقدر داده خوب و قوی تولید می کند !

گام ۱: آموزش شبکه Discriminator

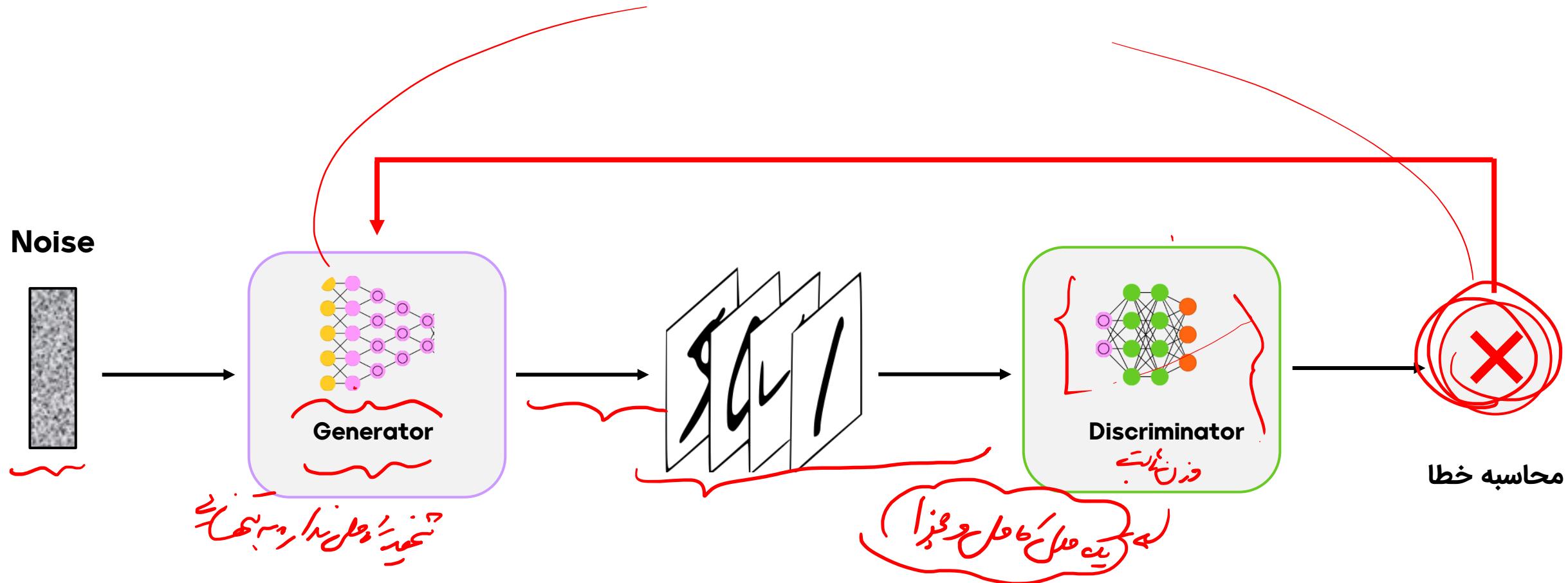


گام ۱: آموزش شبکه Discriminator

step 1: Train Discriminator:

```
W of Generator = Constant  
{ X_real, y_real = Generate_Real_Data()  
  { X_fake, y_fake = Generate_Fake_Data()  
    loss = Train_Discriminator(X_real, y_real, X_fake, y_fake)  
    Update weight of Dirscriminator
```

گام ۲: آموزش شبکه



گام 2: آموزش شبکه Generator

step 2: Train Generator:

{ W of Discriminator = Constant }

X_fake, y_fake = Generate_Fake_Data()

loss = Train_GAN(X_fake, y_fake)

Update weight of Generator

Train_generator

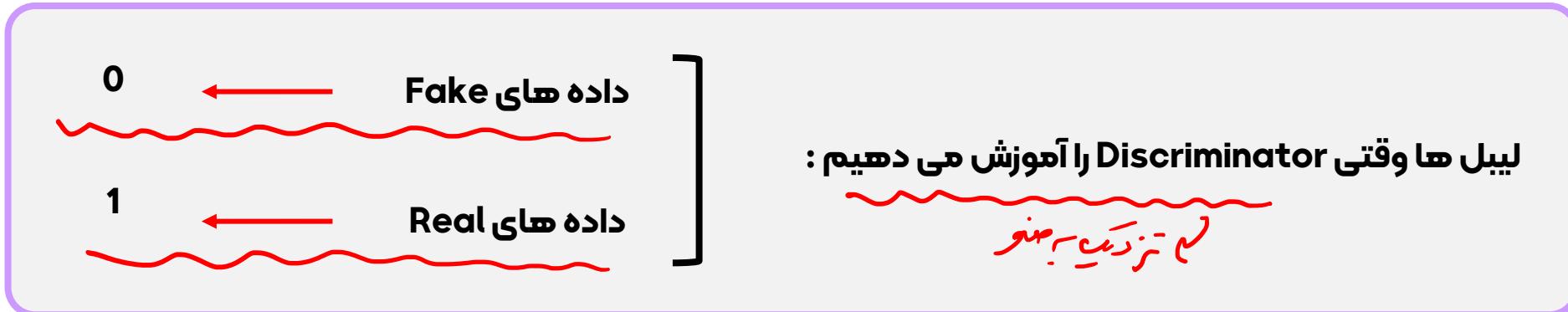
! Loss!

Dis \rightarrow دیس

0 1

Binary Cross Entropy

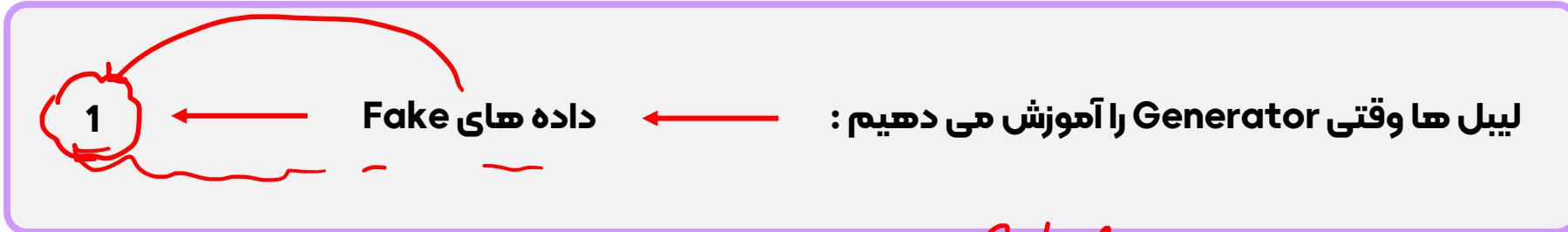
لیبل ها در Discriminator



حال مثل همیشه اگر Discriminator اشتباه کند و لیبل 0 را 1 تشخیص دهد و برعکس، میزان Loss آن

زیاد می شود و بالعکس اگر Train Discriminator دنبال همین موضوع هستیم!

لیبل ها در Generator



generator

دستوراتی که بر مبنای داده هایی که در پیش از آن در خروجی داشتند تغییر می کنند
برای مثال داده هایی که در خروجی داشتند با داده هایی که در پیش از آن در خروجی داشتند تغییر می کنند



سوال:

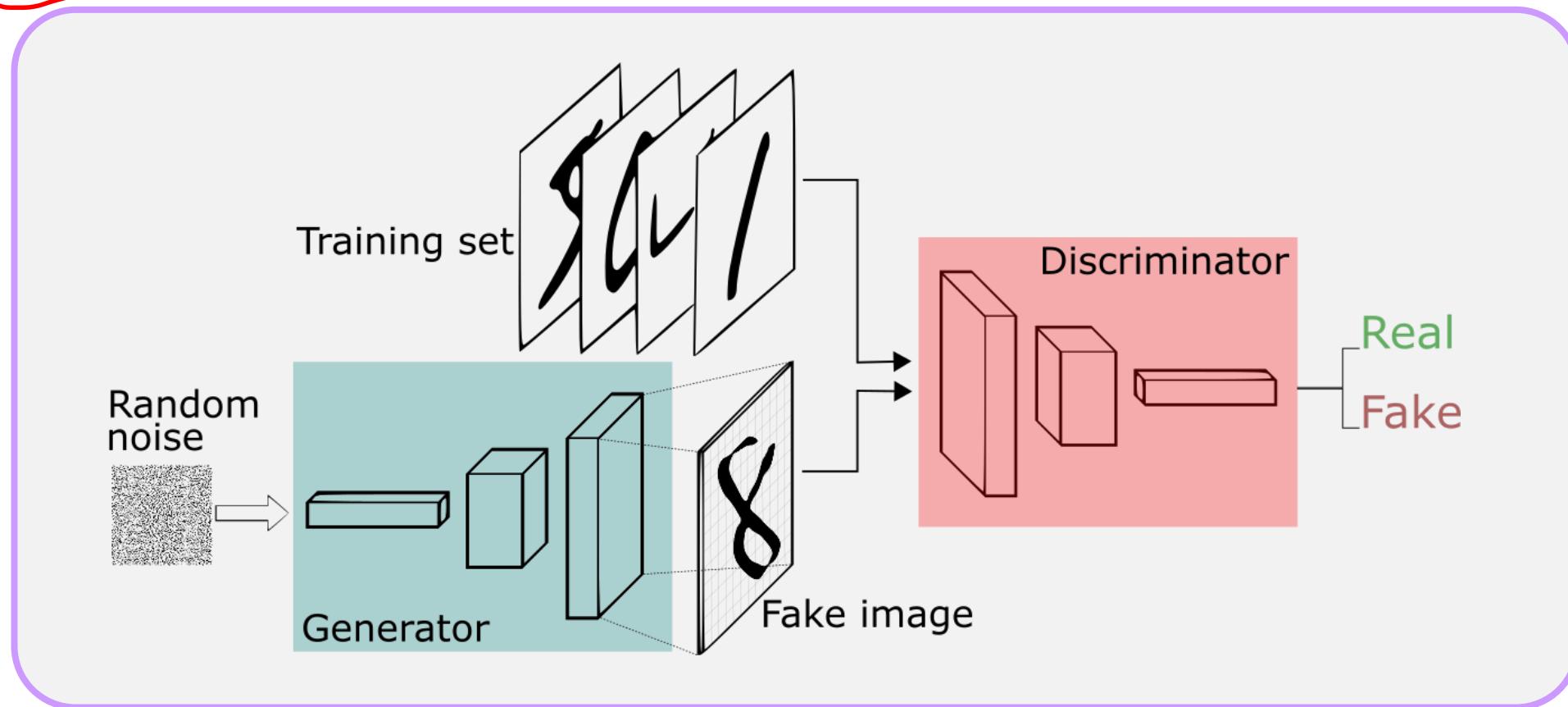


یادگیری در شبکه GAN یک یادگیری است.



حل مثال : تولید ارقام دست نویس

functional
api



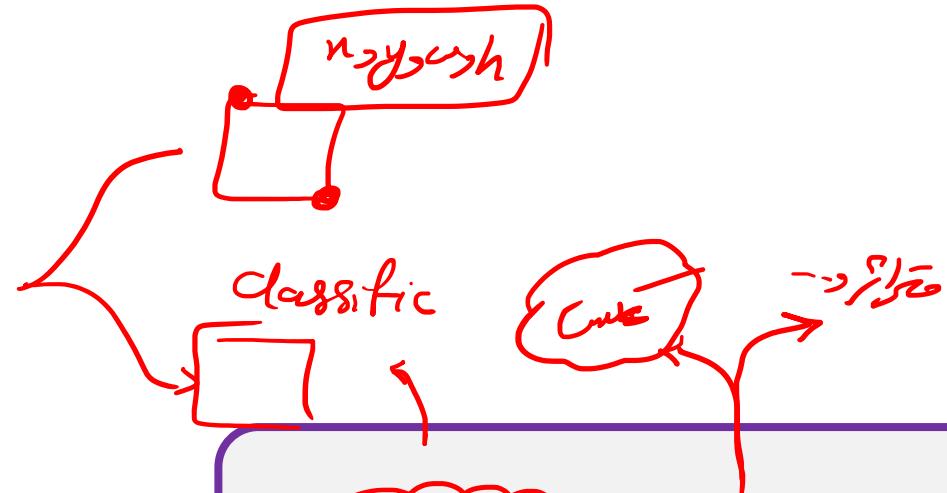
آشنایی با Functional API



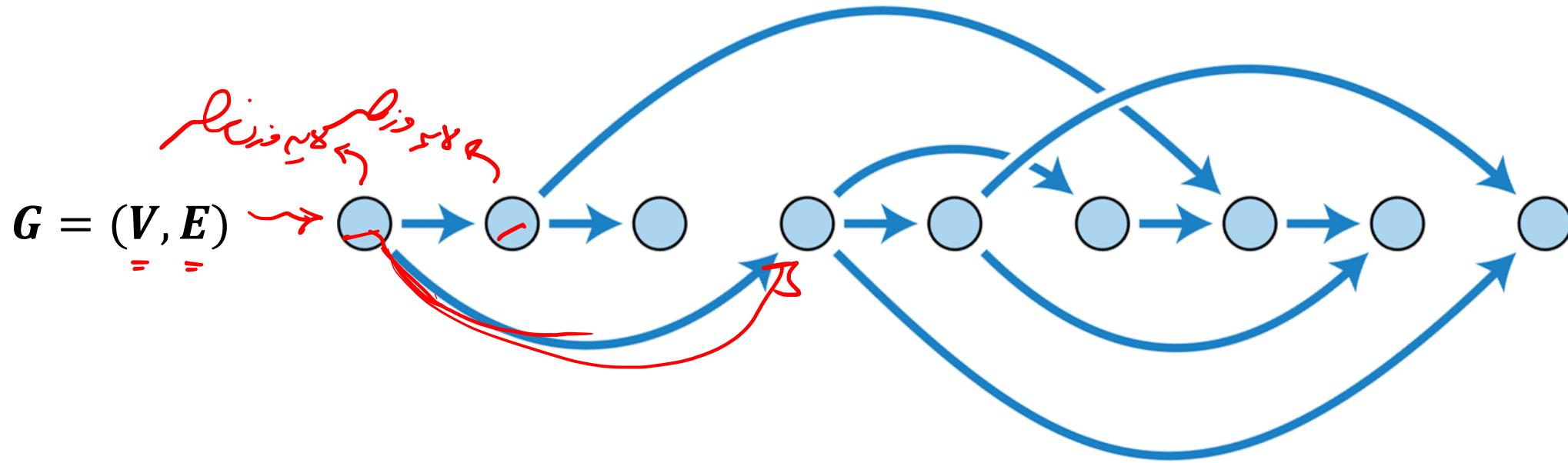
کاربرد Functional API

ساختارهای موازی مثل ResNet و Inception

ساختارهای چند ورودی و یا چند خروجی



ایدهه پشت گراف: Functional API



گراف جهت دار بدون دور

نحوه ایجاد یک مدل Functional API

کام 1 : تعریف ورودی شبکه ✓

کام 2 : تعریف Node ها و اتصالات آن ✓

کام 3 : ساخت مدل (تعریف ورودی ها و خروجی ها)

گام 1: تعریف ورودی به کمک لایه Input

```
input_layer = layers.Input(shape=(32, 32, 3))
```

گام 2: تعریف لایه ها و اتصال به یکدیگر



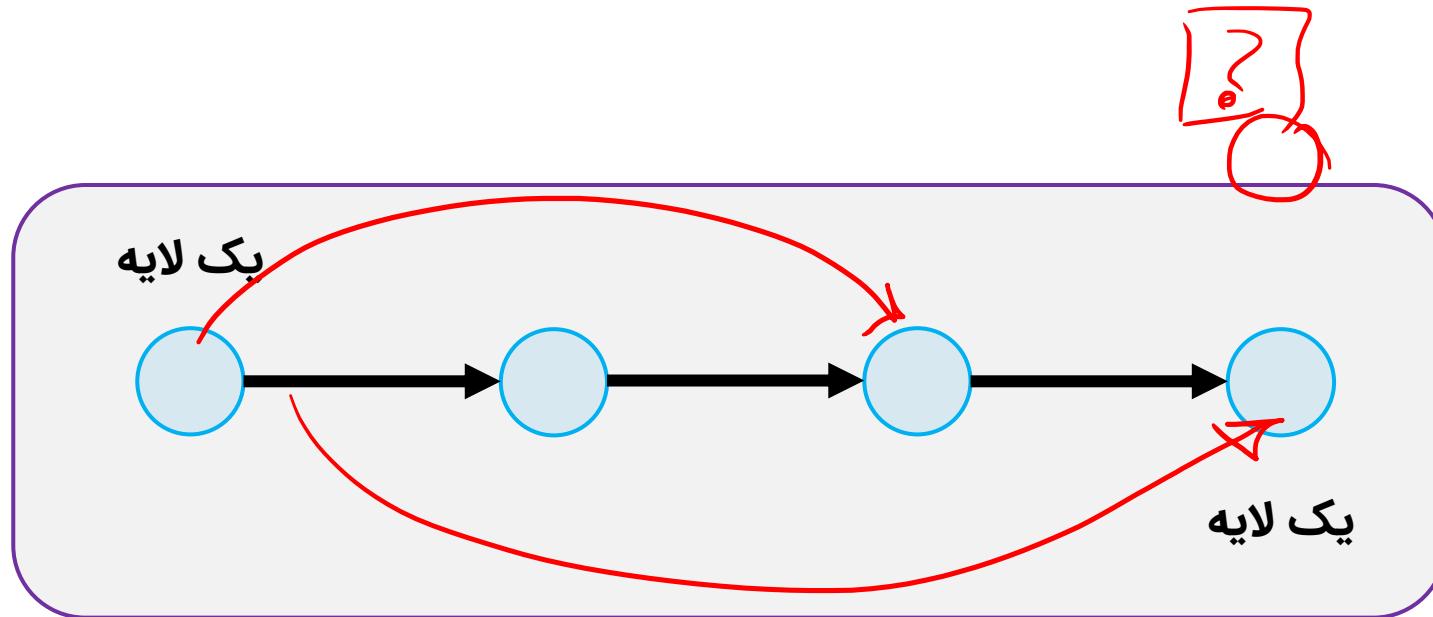
گام ۳ : ساخت مدل با تعریف ورودی ها و خروجی ها

```
model = keras.Model(inputs=inputs, outputs=outputs)
```

[input1, input2]

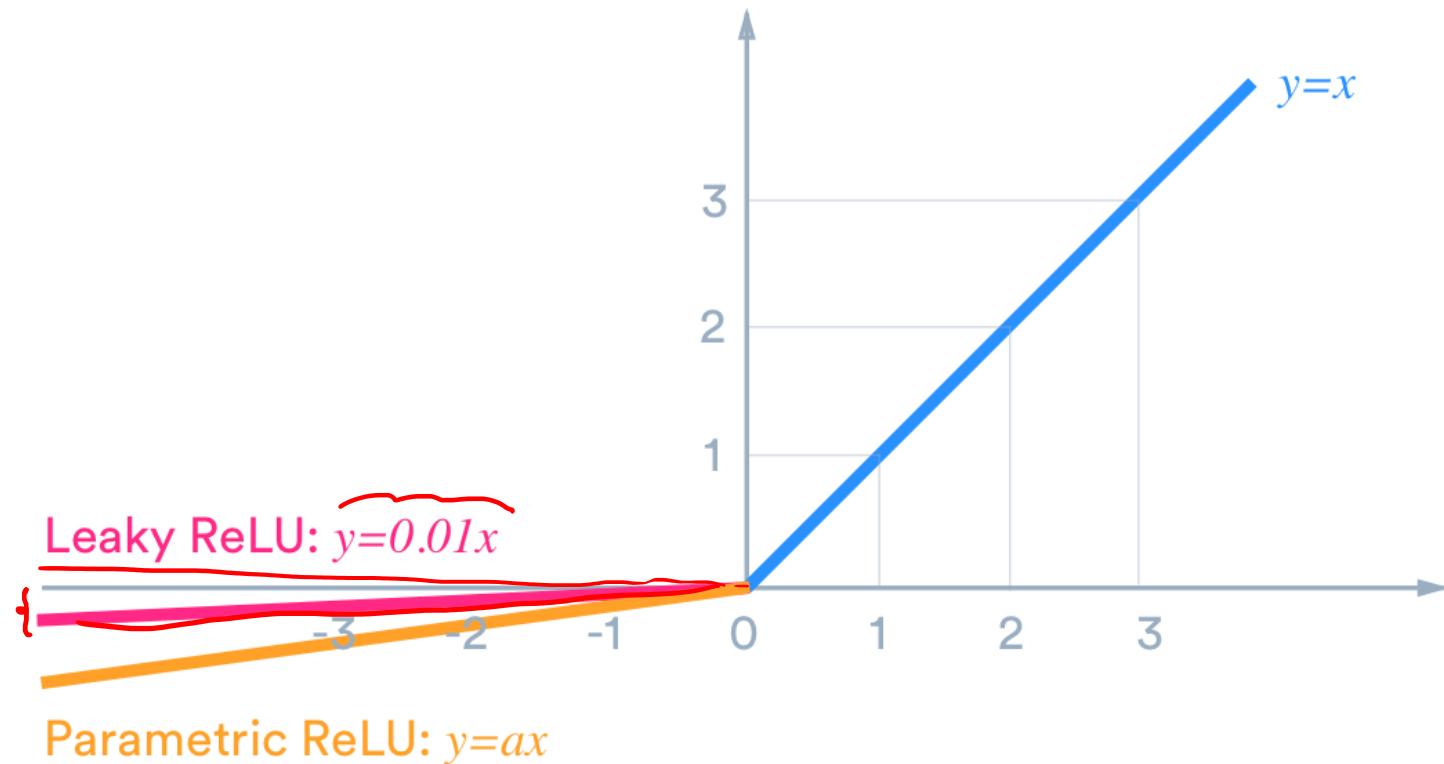
⋮

حال می توانیم به Sequential API یک بار دیگر نگاه کنیم!

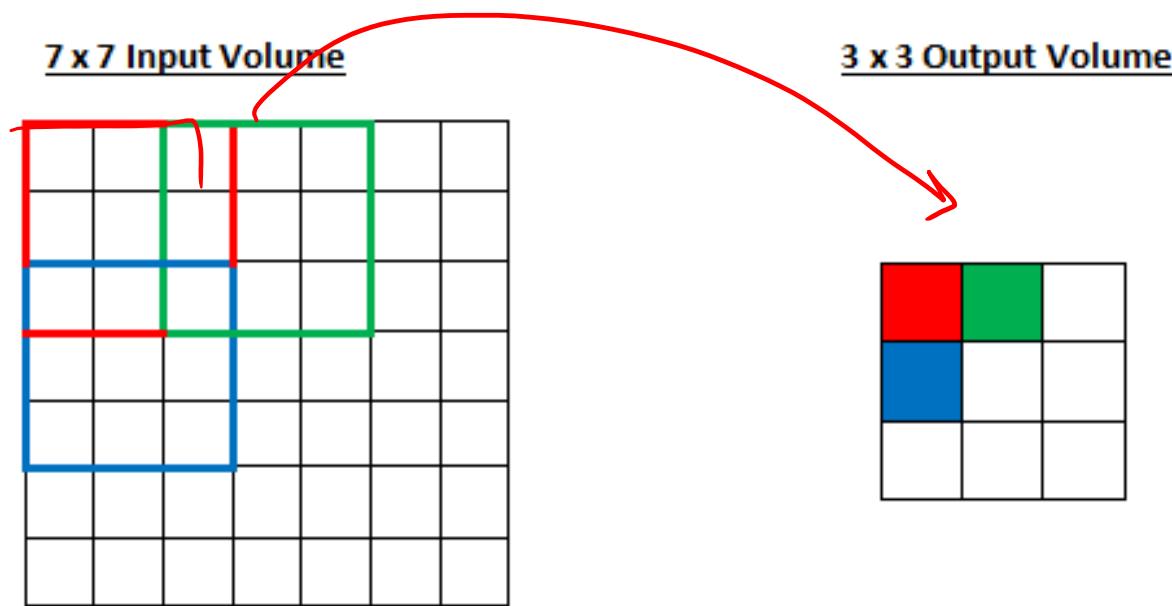


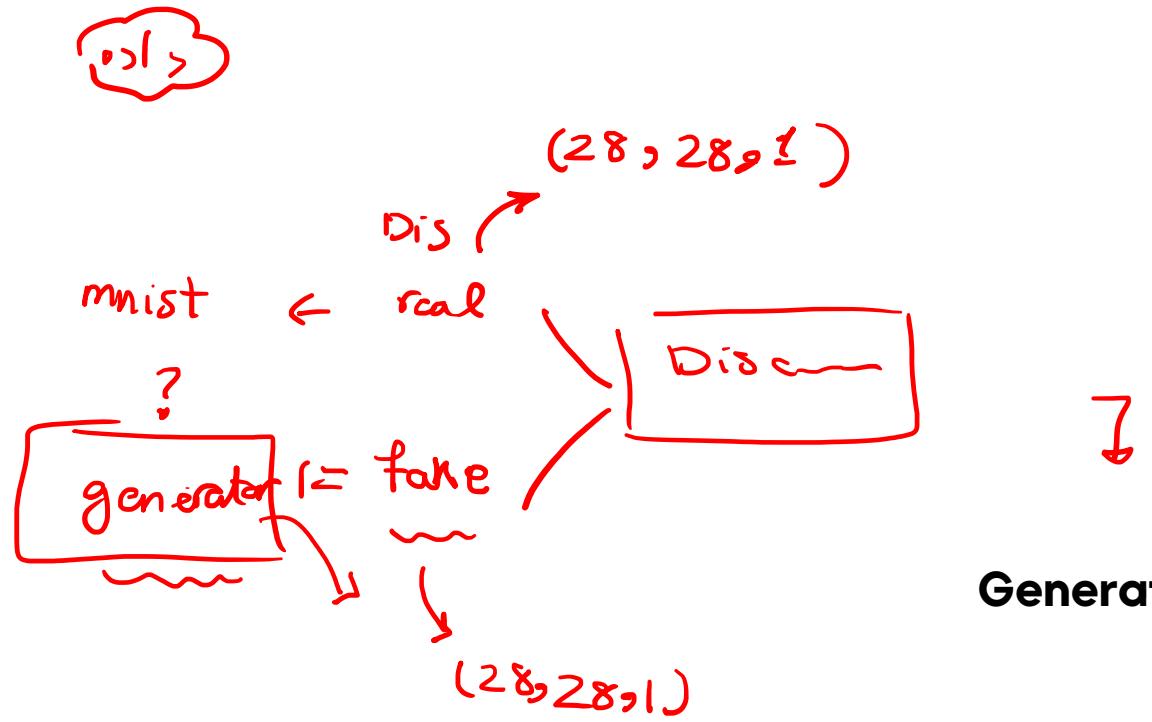
يک مثال :

تغییر ۱: استفاده از تابع فعال سازی LeakyRelu



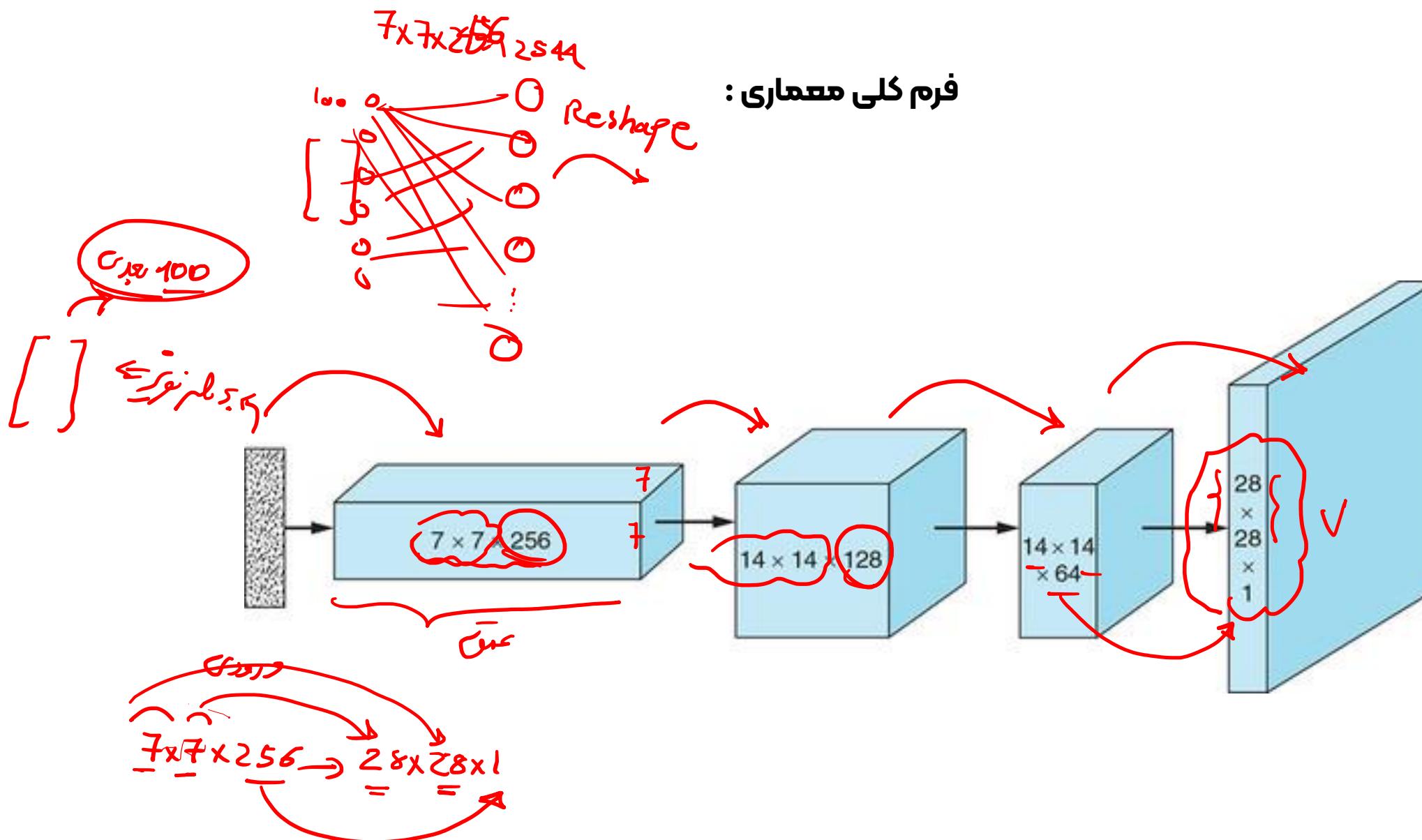
تغییر ۲ : استفاده از Conv به جای MaxPool



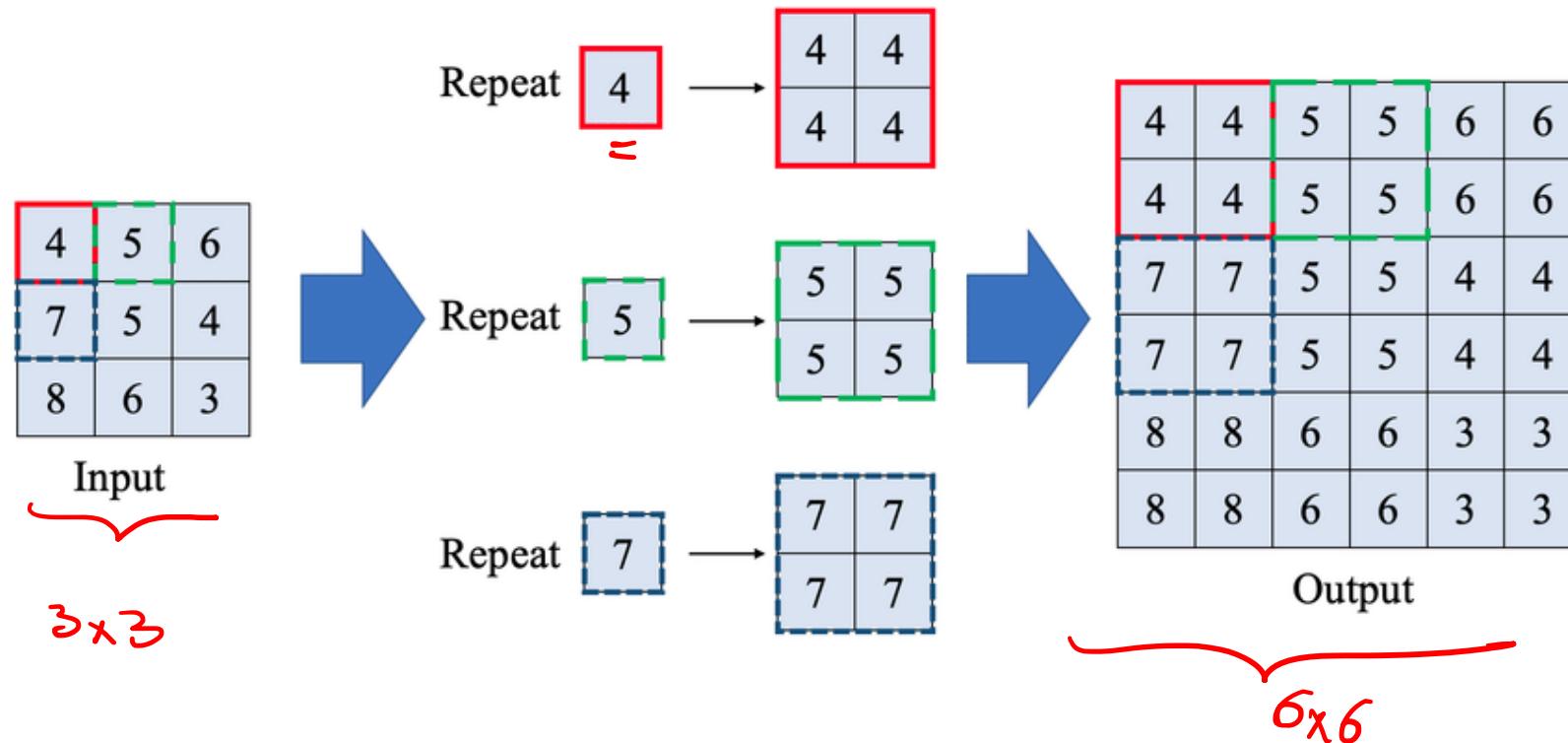


Generator معماري

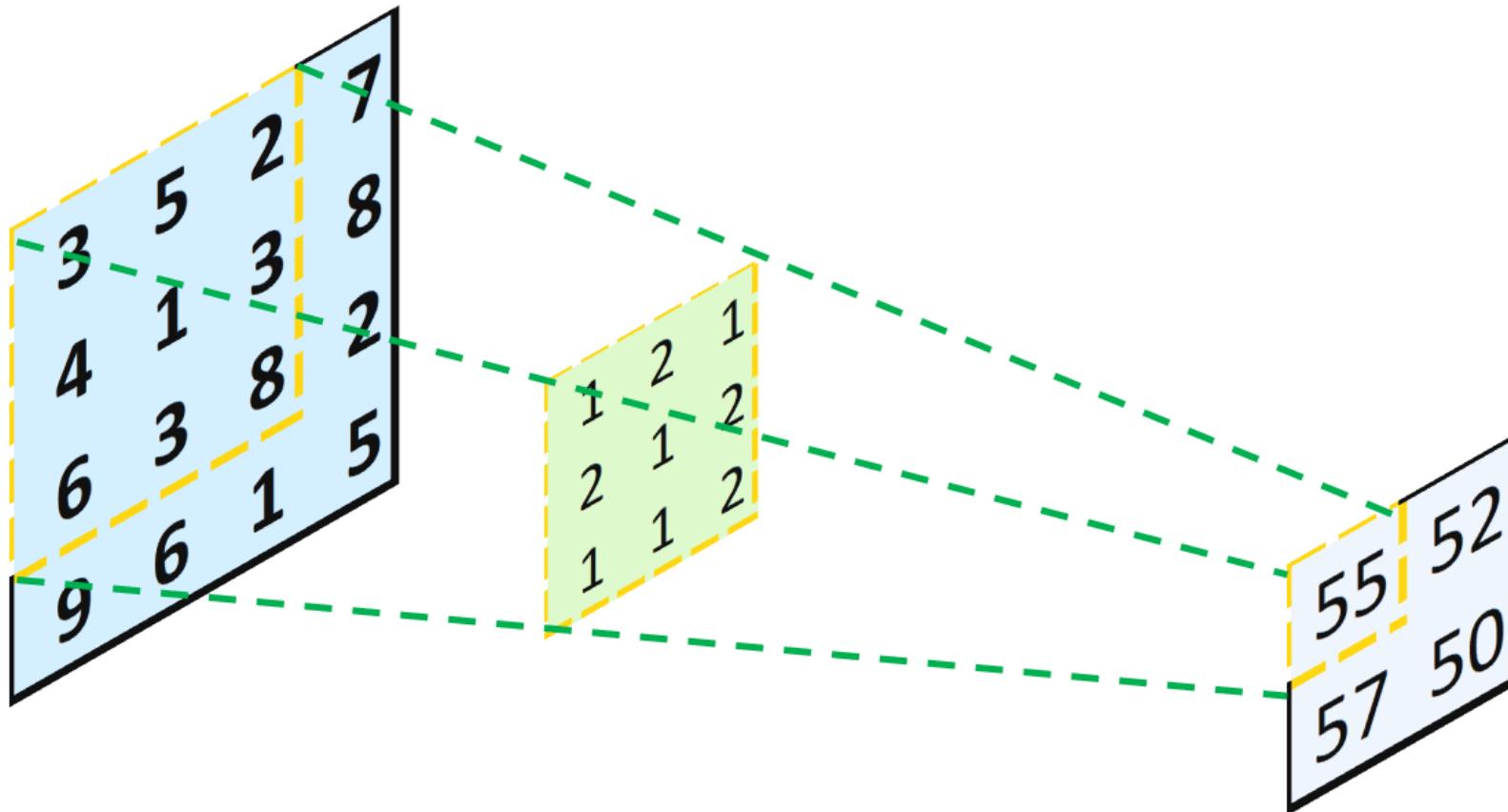
فرم کلی معماري:

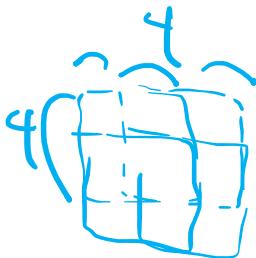
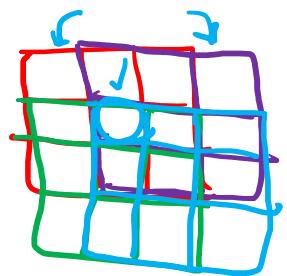


UpSampling2D

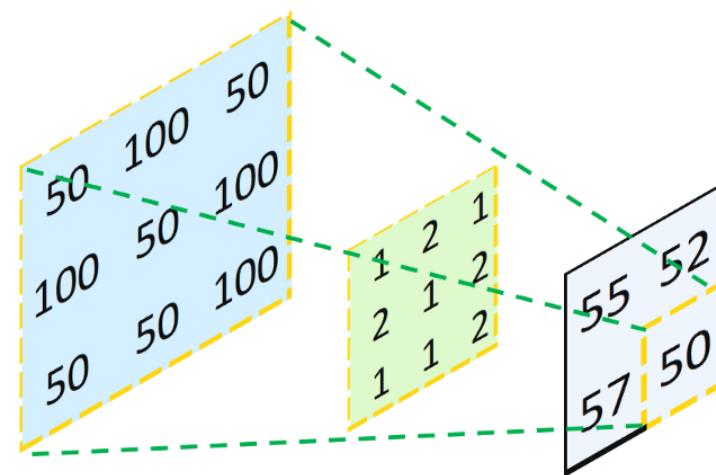
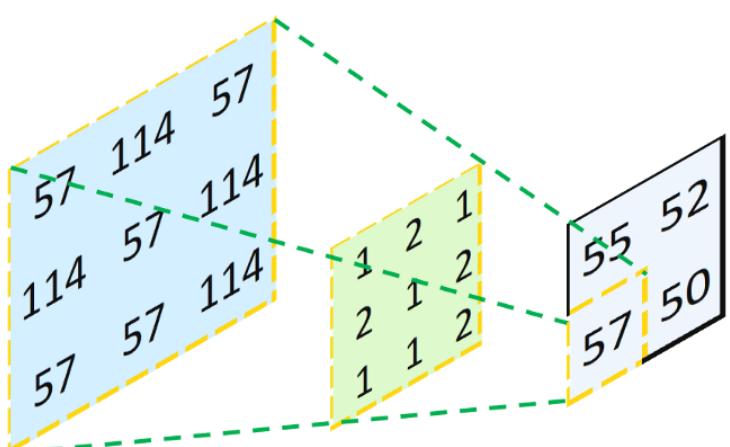
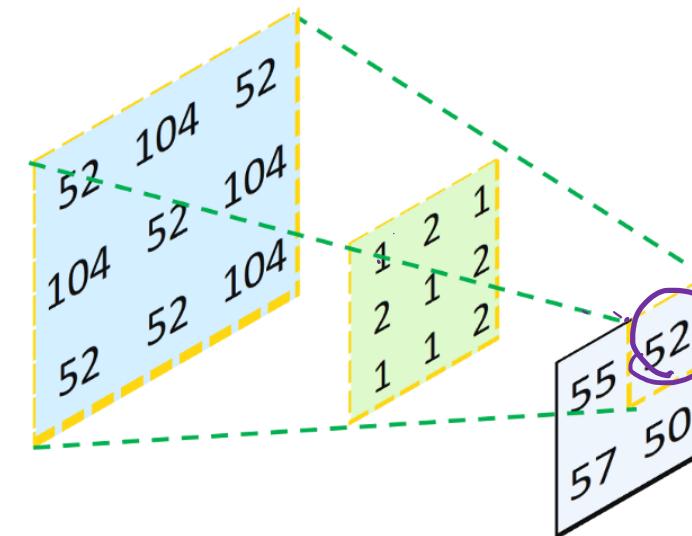
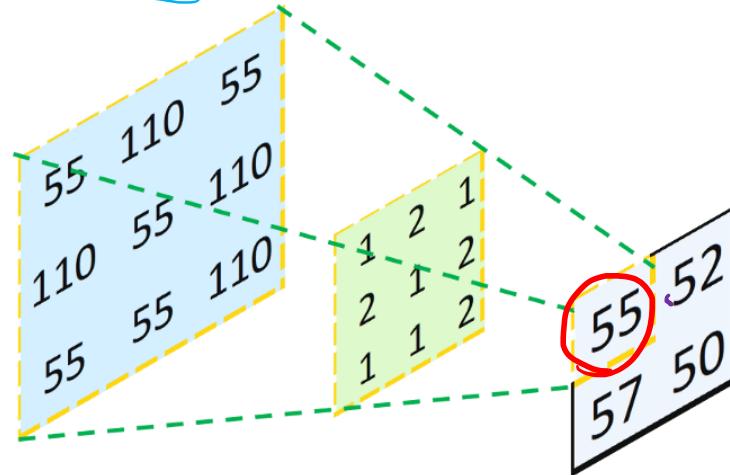
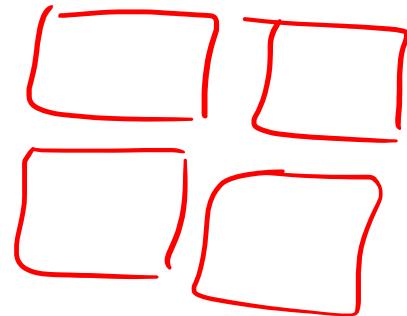


خود لایه Convolution

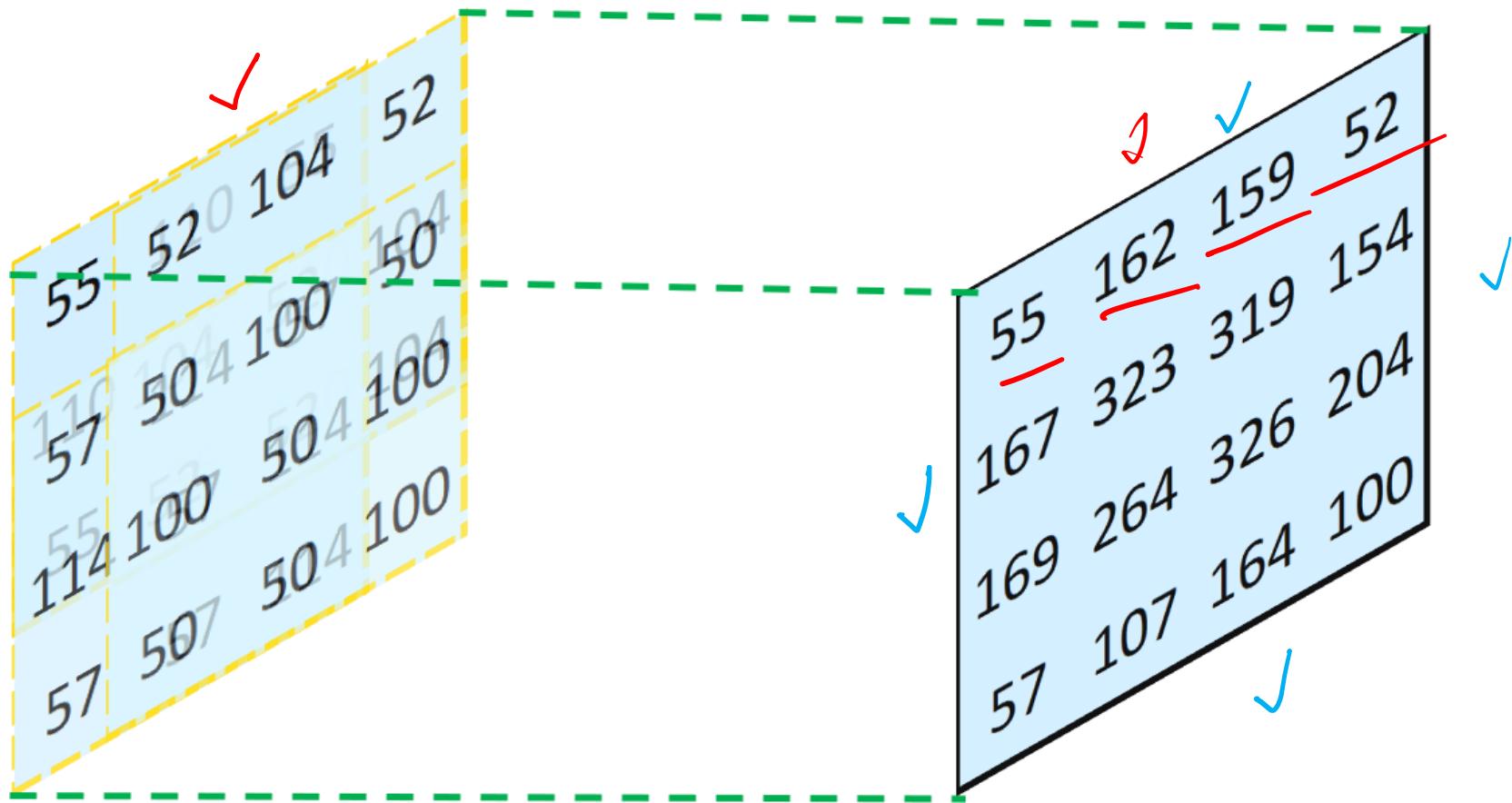




Conv2DTranspose



خروجی نهایی:



يک کد بیانیم با هم

اثر اندازه Kernel

2x2 Input Matrix

55	52
57	50

3x3 Kernel

1	2	1
2	1	2
1	1	2

4x4 Output Matrix

55	162	159	52
167	323	319	154
169	264	326	204
57	107	164	100

2x2 Input Matrix

55	52
57	50

2x2 Kernel

1	2
2	1

3x3 Output Matrix

55	162	104
167	323	152
114	157	50

stride μι

Strides = (1, 1)

55	52
57	50

1	2
2	1



55	162	104
167	323	152
114	157	50

55	162	104
167	323	152
114	157	50

3x3 Output Matrix

Strides = (2, 2)

55	52
57	50

1	2
2	1



55	110	52	104
110	55	104	52
57	114	50	100

55	110	52	104
110	55	104	52
57	114	50	100

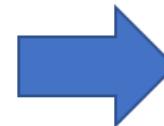
4x4 Output Matrix

Padding

Padding: “valid”

55	52
57	50

1	2	1
2	1	2
1	1	2

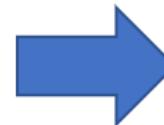


55	162	159	52
167	323	319	154
169	264	326	204
57	107	164	100

Padding: “same”

55	52
57	50

1	2
2	1



55	162	159	52
167	323	319	154
169	264	326	204
57	107	164	100

Generator معماري

```
n_nodes = 128 * 7 * 7  
model = models.Sequential([  
    layers.Dense(n_nodes, input_dim=latent_dim),  
    layers.LeakyReLU(alpha=0.2),  
    layers.Reshape((7, 7, 128)),  
    layers.Conv2DTranspose(128, (4, 4), strides=(2, 2), padding='same'),  
    layers.LeakyReLU(alpha=0.2),  
    layers.Conv2DTranspose(128, (4, 4), strides=(2, 2), padding='same'),  
    layers.LeakyReLU(alpha=0.2),  
    layers.Conv2D(1, (7, 7), activation='sigmoid', padding='same'),  
])
```

Discriminator معماري

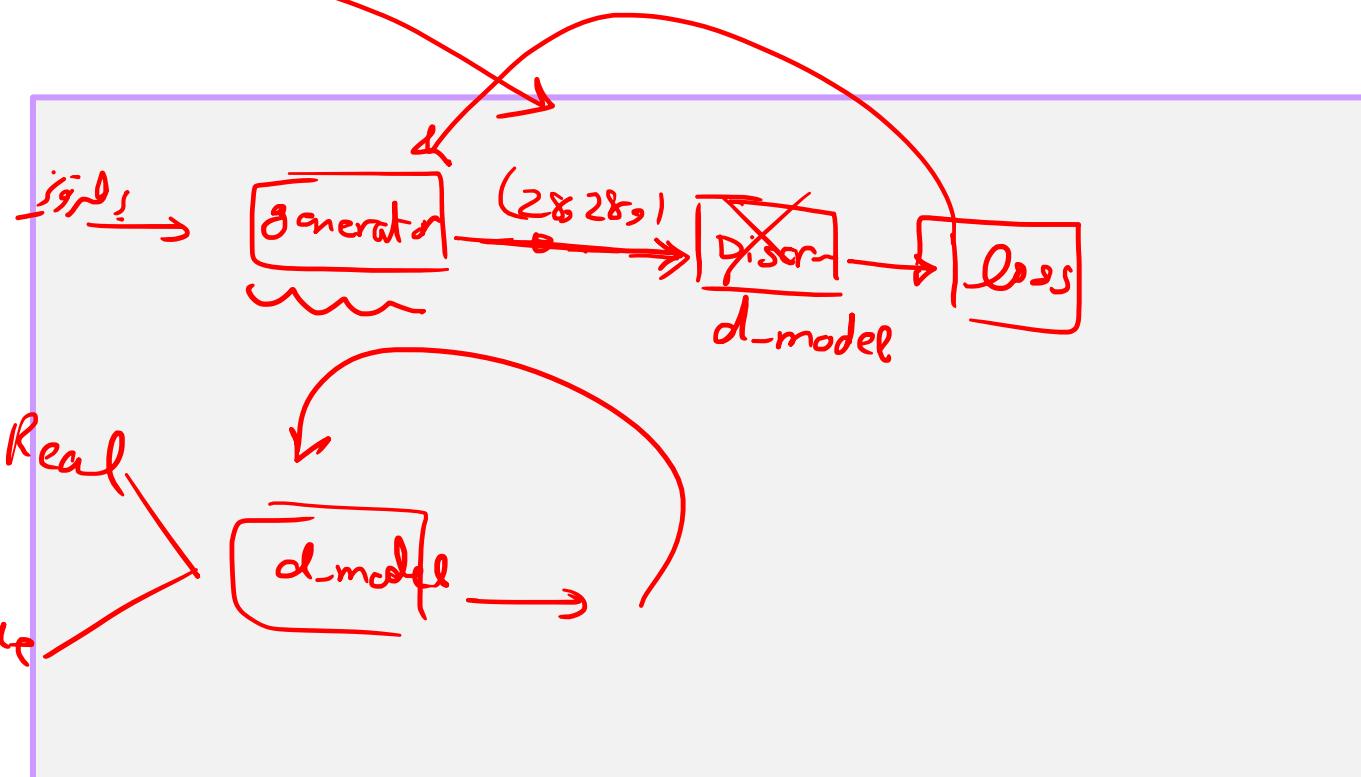
```
model = models.Sequential([
    layers.Conv2D(64, (3, 3), strides=(2, 2), padding='same', input_shape=in_shape),
    layers.LeakyReLU(alpha=0.2),
    layers.Dropout(0.4),
    { layers.Conv2D(64, (3, 3), strides=(2, 2), padding='same'),
      layers.LeakyReLU(alpha=0.2),
      layers.Dropout(0.4),
      { layers.Flatten(),
        layers.Dense(1, activation='sigmoid')
    }
])
```

(28,28,1)

معماری کامل GAN

```
def define_gan(g_model, d_model):  
    d_model.trainable = False  
  
    model = models.Sequential([  
        g_model,  
        d_model  
    ])
```

دربـ ~ False



خواندن دیتاست Real

64
64

128

Disc

64 Real
64 Foke

Gene → 128

```
def load_real_samples():
    (trainX, _), (X, y) = mnist.load_data()
    X = np.reshape(trainX, (len(trainX), 28, 28, 1))
    X = X.astype('float32')
    X = X / 255
    return X
```

float64

Real $\rightarrow 1$
Fake $\rightarrow 0$

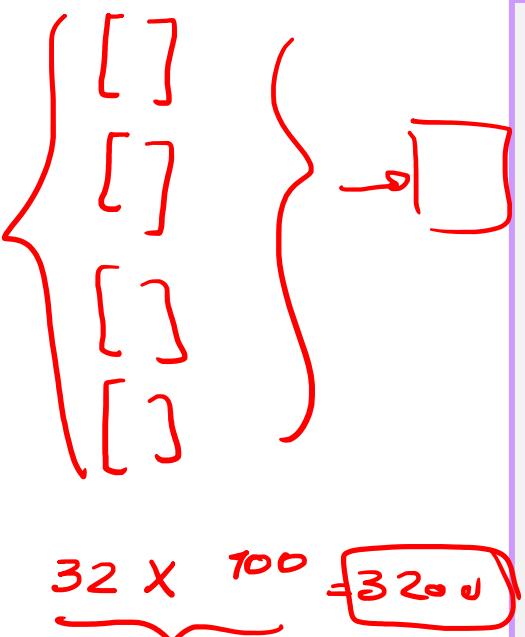
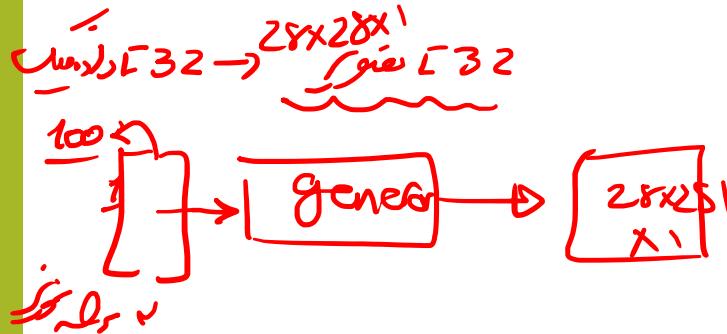
نمونه‌گیری از داده‌های Real

```
def generate_real_samples(dataset, n_samples):  
    ix = np.random.randint(0, dataset.shape[0], n_samples)  
    X = dataset[ix]  
    y = np.ones((n_samples, 1))  
    return X, y
```

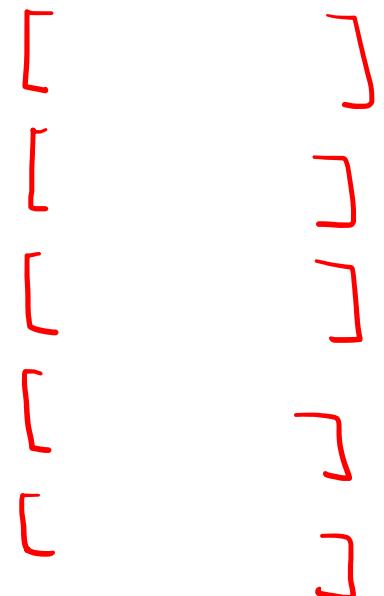
Annotations:

- dataset: 64 → 100
- n_samples: 2 → 127
- ix: 60000 → 64
- X: 60000 → 64
- y: 127 → 127

تولید بردارهای نویز ورودی به تعداد کافی!



```
def generate_latent_points(latent_dim, n_samples):  
    x_input = np.random.randn(latent_dim * n_samples)  
    x_input = x_input.reshape(n_samples, latent_dim)  
    return x_input
```



تولید داده های فیک!

```
def generate_fake_samples(g_model, latent_dim, n_samples):  
    x_input = generate_latent_points(latent_dim, n_samples)  
    x = g_model.predict(x_input)  
    y = np.zeros((n_samples, 1))  
    return x, y
```

28x28x1 (-32)

نیز

حالا برسیم به اصل ماجرا ! تابع train

بخش اول :

```
def train(g_model, d_model, gan_model, dataset, latent_dim, n_epochs=100, n_batch=256):  
    bat_per_epo = int(dataset.shape[0] / n_batch)  
    half_batch = int(n_batch / 2)  
    for i in range(n_epochs):
```

epoch = 8 (256) → 256048
for i in range(100) = 2048 / 256
100
100x8

for i = 1 to 8
 for i in bat -- 8 {
 }
 }
 }

بخش دو: discriminator شبکه Train

```
X_real, y_real = generate_real_samples(dataset, half_batch)

X_fake, y_fake = generate_fake_samples(g_model, latent_dim, half_batch)

X, y = np.vstack((X_real, X_fake)), np.vstack((y_real, y_fake))

d_loss, _ = d_model.train_on_batch(X, y)
```

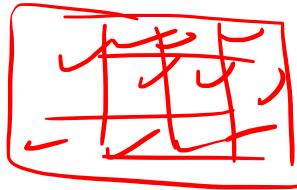
بخش دو: Train شبکه Generator

```
X_gan = generate_latent_points(latent_dim, n_batch)

y_gan = np.ones((n_batch, 1))

g_loss = gan_model.train_on_batch(X_gan, y_gan)
```

```
print(f">{i+1}, {j+1}/{bat_per_epo}, d= {d_loss:.3f}, g={g_loss:.3f}")
```



summarize performance ملخص

```
def summarize_performance(epoch, g_model, d_model, dataset, latent_dim, n_samples=100):  
    x_real, y_real = generate_real_samples(dataset, n_samples)  
    _, acc_real = d_model.evaluate(x_real, y_real, verbose=0)  
    x_fake, y_fake = generate_fake_samples(g_model, latent_dim, n_samples)  
    _, acc_fake = d_model.evaluate(x_fake, y_fake, verbose=0)  
    ✓ print(f'>Accuracy real: {acc_real*100}, fake: {acc_fake*100}')  
    ✓ save_plot(x_fake, epoch)  
    { filename = f'generator_model_{epoch + 1}.h5'  
      g_model.save(filename)
```

نهايش خروجي

```
def save_plot(examples, epoch, n=10):
    for i in range(n * n):
        plt.subplot(n, n, 1 + i)
        plt.axis('off')
        plt.imshow(examples[i, :, :, 0], cmap='gray_r')
        filename = 'generated_plot_e%03d.png' % (epoch+1)
        plt.savefig(filename)
        plt.close()
```

The
end.