Advanced Machine Learning

Bilel GUETARNI, PhD

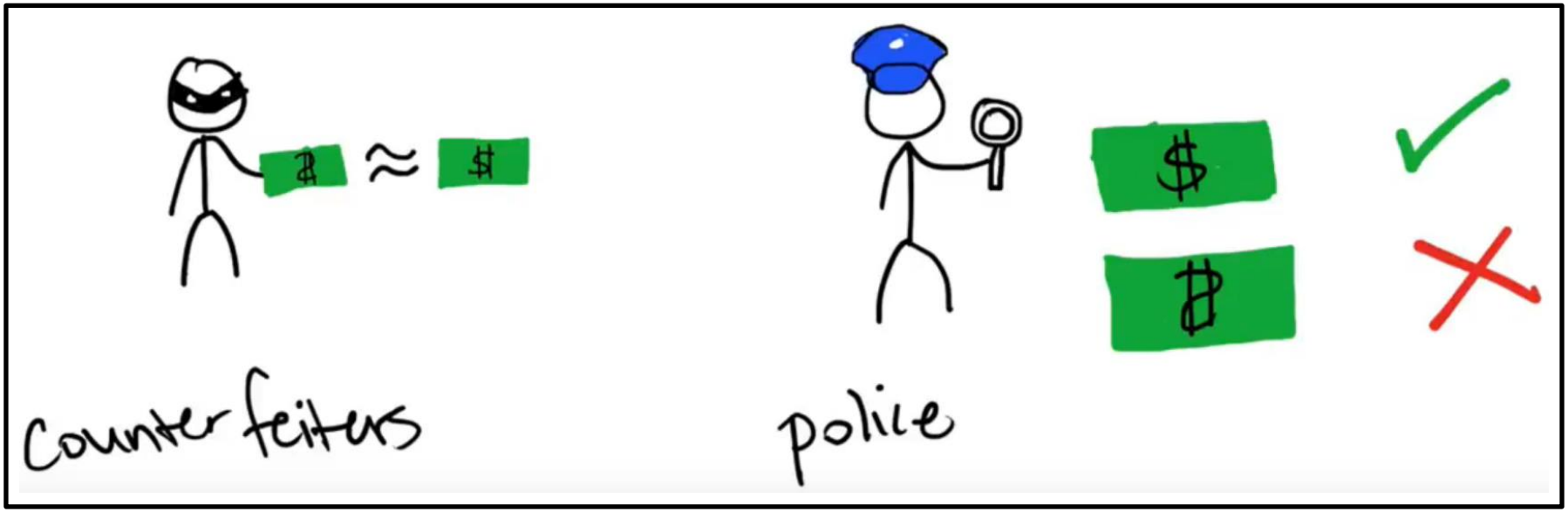
bilel.guetarni@junia.com

JUNIA ISEN

PowerPoint slides from **Halim Benhabiles, PhD** and **Farès BOUGOURZI, PhD**

Generative Adversarial Networks (GANs)

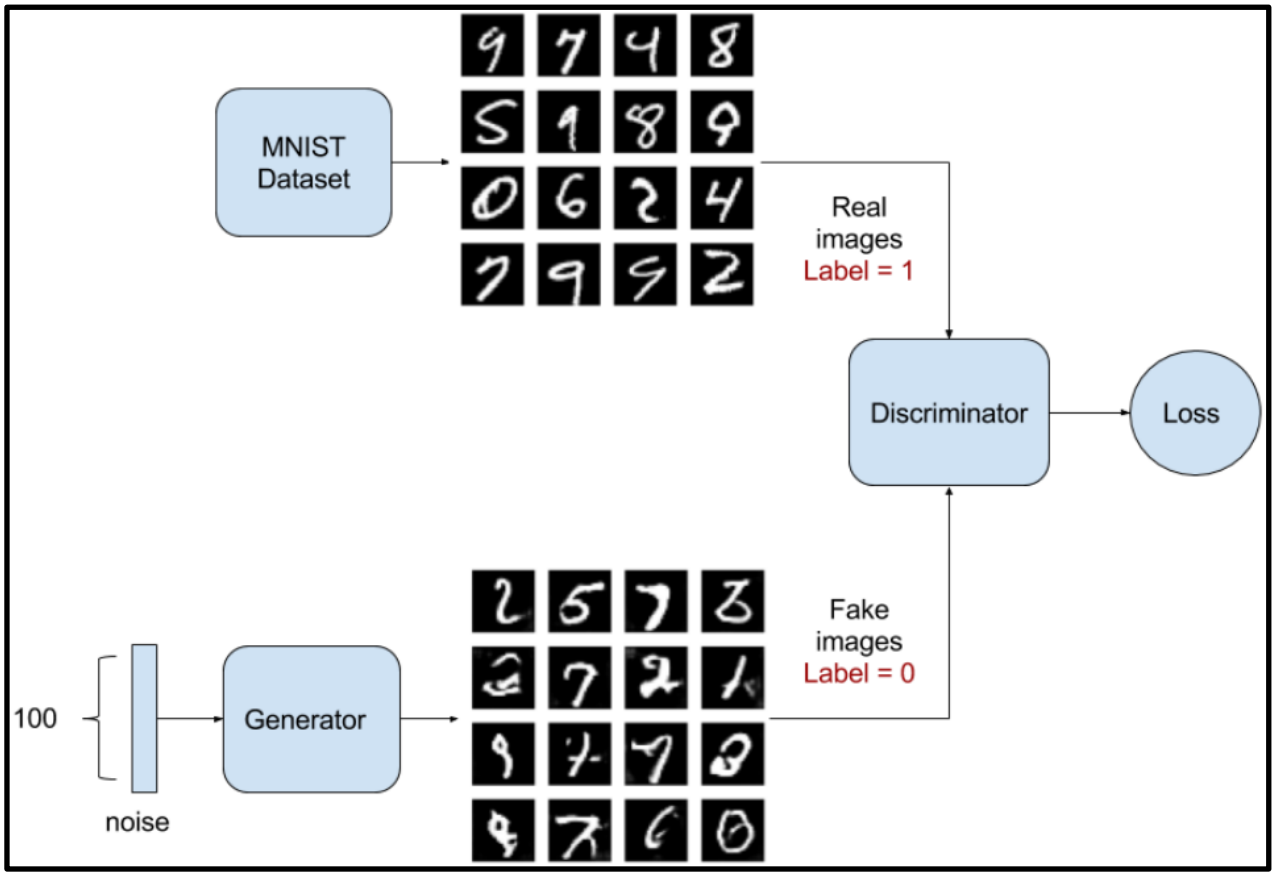
- Intuition behind these models



Extracted from
https://www.youtube.com/watch?v=ZRgwcMqxhPw

Generative Adversarial Networks (GANs)

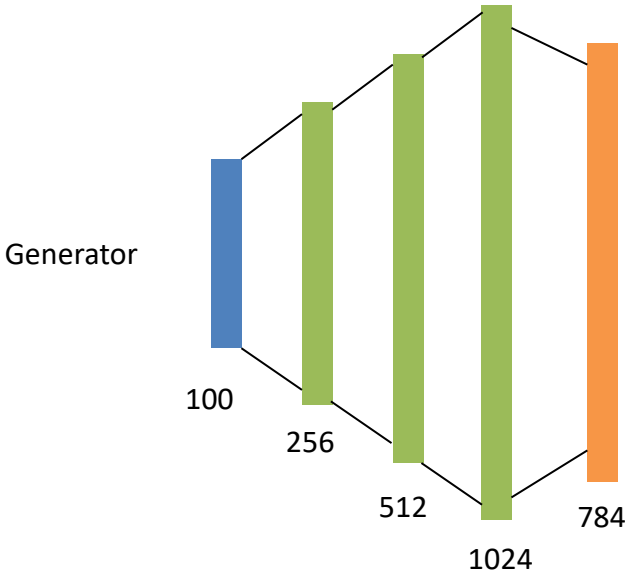
- GAN model



Extracted from

https://towardsdatascience.com/gan-by-example-using-keras-on-tensorflow-backend-1a6d515a60d0

Building a GAN using Keras - Tensorflow



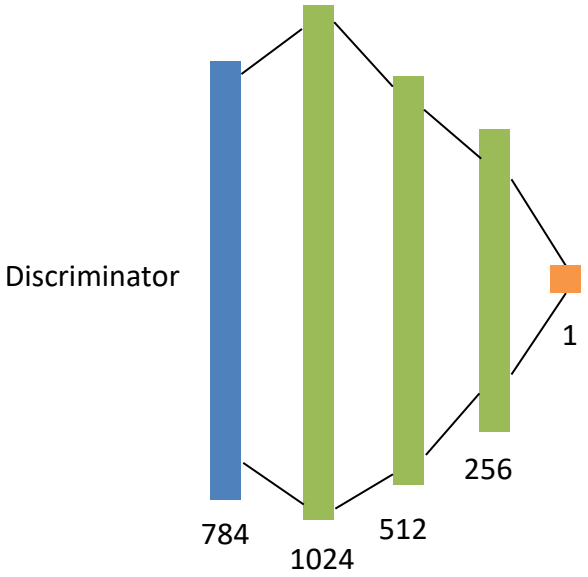

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 256)               25856
_________________________________________________________________
dense_1 (Dense)              (None, 512)               131584
_________________________________________________________________
dense_2 (Dense)              (None, 1024)              525312
_________________________________________________________________
dense_3 (Dense)              (None, 784)               803600
=================================================================
Total params: 1,486,352
Trainable params: 1,486,352
Non-trainable params: 0
_________________________________________________________________
```

Building a GAN using Keras - Tensorflow

```python
# create a generator network
# generate an 784 vector output from 100 vector input

def create_generator():
    generator=Sequential()
    generator.add(Dense(256,activation=LeakyReLU(0.2), input_dim=100))
    generator.add(Dense(512, activation=LeakyReLU(0.2)))
    generator.add(Dense(1024, activation=LeakyReLU(0.2)))

    generator.add(Dense(784, activation='tanh'))

    generator.compile(loss='binary_crossentropy', optimizer='adam')
    return generator

g=create_generator()
g.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 256)               25856
_________________________________________________________________
dense_1 (Dense)              (None, 512)               131584
_________________________________________________________________
dense_2 (Dense)              (None, 1024)              525312
_________________________________________________________________
dense_3 (Dense)              (None, 784)               803600
=================================================================
Total params: 1,486,352
Trainable params: 1,486,352
Non-trainable params: 0
_________________________________________________________________
```

Building a GAN using Keras - Tensorflow




```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 1024)              803840
_________________________________________________________________
dropout (Dropout)            (None, 1024)              0
_________________________________________________________________
dense_5 (Dense)              (None, 512)               524800
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_6 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 257
=================================================================
Total params: 1,460,225
Trainable params: 1,460,225
Non-trainable params: 0
_________________________________________________________________
```

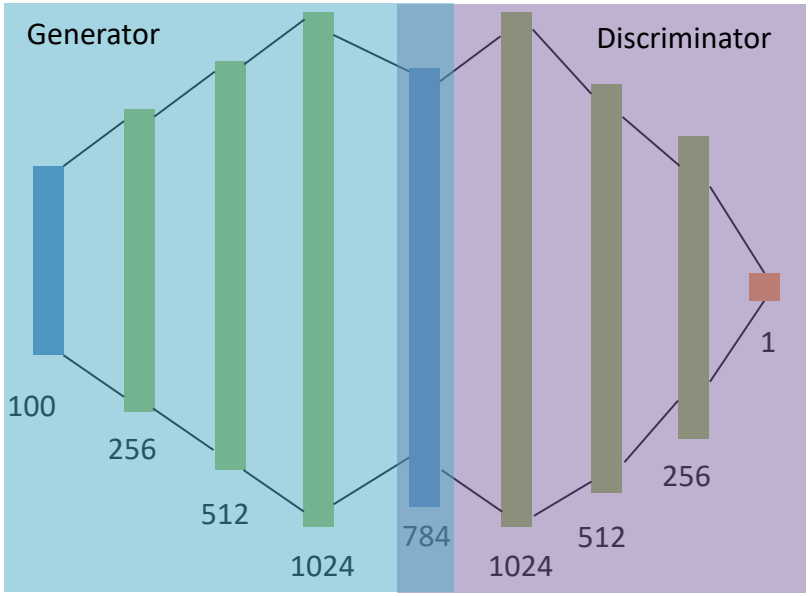
Building a GAN using Keras - Tensorflow

```python
# create a descriminator network
# take an image shape and classify it
def create_discriminator():
    discriminator=Sequential()
    discriminator.add(Dense(1024,activation=LeakyReLU(0.2), input_dim=784))
    discriminator.add(Dropout(0.3))


    discriminator.add(Dense(512, activation=LeakyReLU(0.2)))
    discriminator.add(Dropout(0.3))

    discriminator.add(Dense(256, activation=LeakyReLU(0.2)))

    discriminator.add(Dense(1, activation='sigmoid'))

    discriminator.compile(loss='binary_crossentropy', optimizer='adam')
    return discriminator

d =create_discriminator()
d.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 1024)              803840
_________________________________________________________________
dropout (Dropout)            (None, 1024)              0
_________________________________________________________________
dense_5 (Dense)              (None, 512)               524800
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_6 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 257
=================================================================
Total params: 1,460,225
Trainable params: 1,460,225
Non-trainable params: 0
_________________________________________________________________
```

Building a GAN using Keras - Tensorflow

GAN



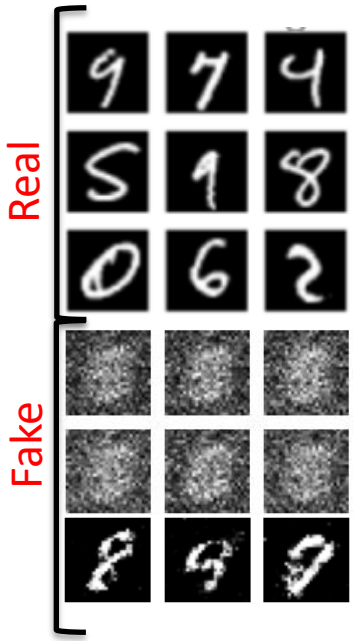

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 100)]             0
_________________________________________________________________
sequential (Sequential)      (None, 784)               1486352
_________________________________________________________________
sequential_1 (Sequential)    (None, 1)                 1460225
=================================================================
Total params: 2,946,577
Trainable params: 1,486,352
Non-trainable params: 1,460,225
_________________________________________________________________
```

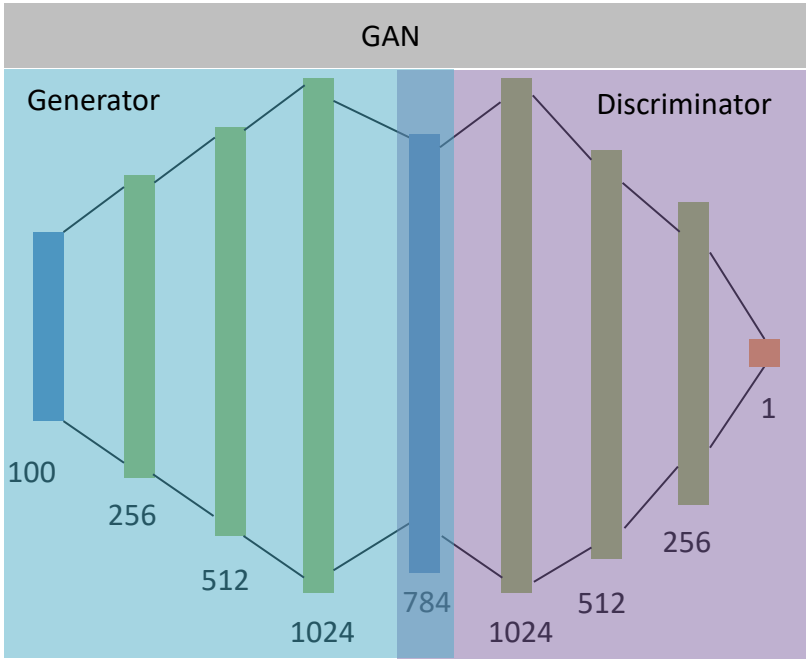
Building a GAN using Keras - Tensorflow

```python
from tensorflow.keras import Input, Model

def create_gan(discriminator, generator):
    discriminator.trainable=False
    gan_input = Input(shape=(100,))
    x = generator(gan_input)
    gan_output= discriminator(x)
    gan= Model(inputs=gan_input, outputs=gan_output)
    gan.compile(loss='binary_crossentropy', optimizer='adam')
    return gan
gan = create_gan(d,g)
gan.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 100)]             0
_________________________________________________________________
sequential (Sequential)      (None, 784)               1486352
_________________________________________________________________
sequential_1 (Sequential)    (None, 1)                 1460225
=================================================================
Total params: 2,946,577
Trainable params: 1,486,352
Non-trainable params: 1,460,225
_________________________________________________________________
```

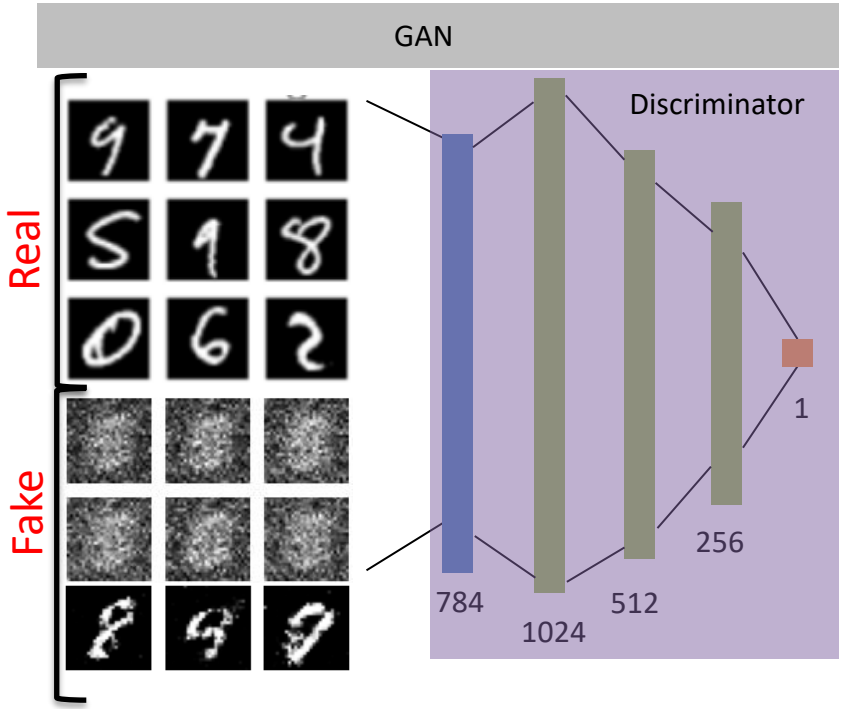
Building a GAN using Keras - Tensorflow





How to train our GAN?

Building a GAN using Keras - Tensorflow



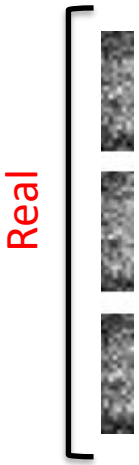
First, train discriminator only with two labels data

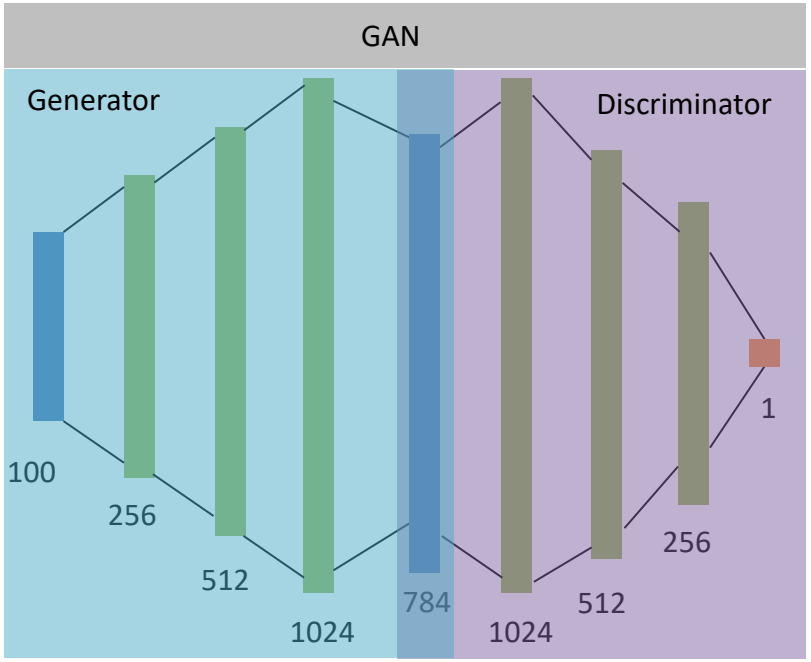
Building a GAN using Keras - Tensorflow

```python
import tensorflow

from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Flatten, Reshape
from tensorflow.keras.layers import Conv2D, Conv2DTranspose, UpSampling2D
from tensorflow.keras.layers import LeakyReLU, Dropout
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.optimizers import Adam, RMSprop

import numpy as np
import time
import matplotlib.pyplot as plt
#import pandas as pd


# load mnist data
def load_data():
    (x_train, y_train), (x_test, y_test) = mnist.load_data()
    # normalise to range [-1,1]
    x_train = (x_train.astype(np.float32) - 127.5)/127.5

    # convert shape of x_train from (60000, 28, 28) to (60000, 784)
    # 784 columns per row
    x_train = x_train.reshape(60000, 784)
    return (x_train, y_train, x_test, y_test)

(X_train, y_train,X_test, y_test)=load_data()
print(X_train.shape)
```

```
(60000, 784)
```

Building a GAN using Keras - Tensorflow



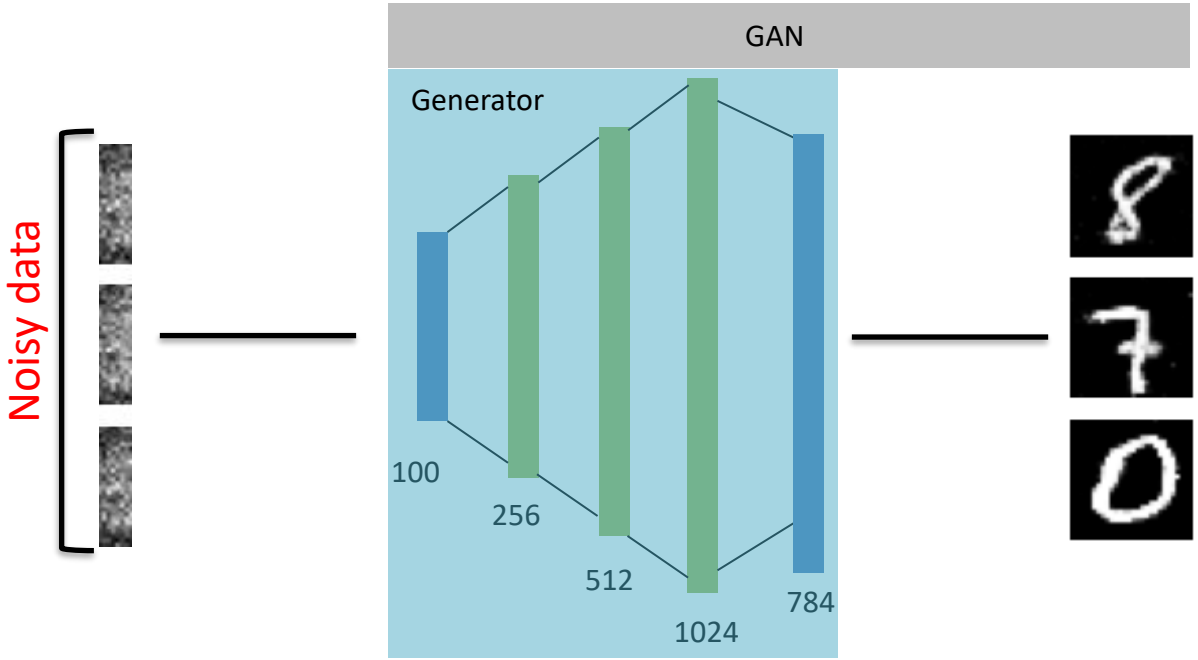


Second, train all the GAN with tricky data, one label

Building a GAN using Keras - Tensorflow

```python
def training(epochs=1, batch_size=128):

    #Loading the data
    (X_train, y_train, X_test, y_test) = load_data()
    #Used to train on batches
    batch_count = int (np.ceil(X_train.shape[0] / batch_size)) -1
    print (batch_count)

    # Creating GAN
    generator= create_generator()
    discriminator= create_discriminator()
    gan = create_gan(discriminator, generator)
    for e in range(1,epochs+1 ):
        print("Epoch %d" %e)
        for _ in (range(batch_size)):
            #generate  random noise as an input  to  initialize the  generator
            noise= np.random.normal(0,1, [batch_size, 100])

            # Generate fake MNIST images from noised input
            generated_images = generator.predict(noise)

            # Get set of  real images
            #image_batch =X_train[index* batch_size:(index+1) * batch_size]
            # Get a random set of  real images
            image_batch =X_train[np.random.randint(low=0,high=X_train.shape[0],size=batch_size)]

            #Construct different batches of  real and fake data
            X= np.concatenate([image_batch, generated_images])

            # Labels for generated and real data 0:fake and 1:real
            y_dis=np.zeros(2*batch_size)
            y_dis[:batch_size]=0.9

            #Pre train discriminator on  fake and real data  before starting the gan.
            discriminator.trainable=True
            discriminator.train_on_batch(X, y_dis)

            #Tricking the noised input of the Generator as real data, set fake data to 1
            noise= np.random.normal(0,1, [batch_size, 100])
            y_gen = np.ones(batch_size)

            # During the training of gan,
            # the weights of discriminator should be freezed.
            #We can enforce that by setting the trainable flag
            discriminator.trainable=False

            #training  the GAN by alternating the training of the Discriminator
            #and training the chained GAN model with Discriminator's weights freezed.
            gan.train_on_batch(noise, y_gen)
        if e == 1 or e % 20 == 0:
            plot_generated_images(e, generator)


training(1000,64)
```

Building a GAN using Keras - Tensorflow



Our generator is ready to generate data

Building a GAN using Keras - Tensorflow

```
def plot_generated_images(epoch, generator, examples=100, dim=(10,10), figsize=(10,10)):
    noise= np.random.normal(loc=0, scale=1, size=[examples, 100])
    generated_images = generator.predict(noise)
    generated_images = generated_images.reshape(examples,28,28)
    plt.figure(figsize=figsize)
    for i in range(generated_images.shape[0]):
            plt.subplot(dim[0], dim[1], i+1)
            plt.imshow(generated_images[i], cmap='gray')
            plt.axis('off')
            plt.tight_layout()
    plt.show()
    #plt.savefig('gan_generated_image %d.png' %epoch)
```

Lab session Keras – TensorFlow core

- Build Your MLP-GAN model and train it on the MNIST data set
  - Show the training performance (loss , accuracy)
  - Once your done generate some images

- Adapt the model to build a CNN-GAN
  - https://towardsdatascience.com/gan-by-example-using-keras-on-tensorflow-backend-1a6d515a60d0