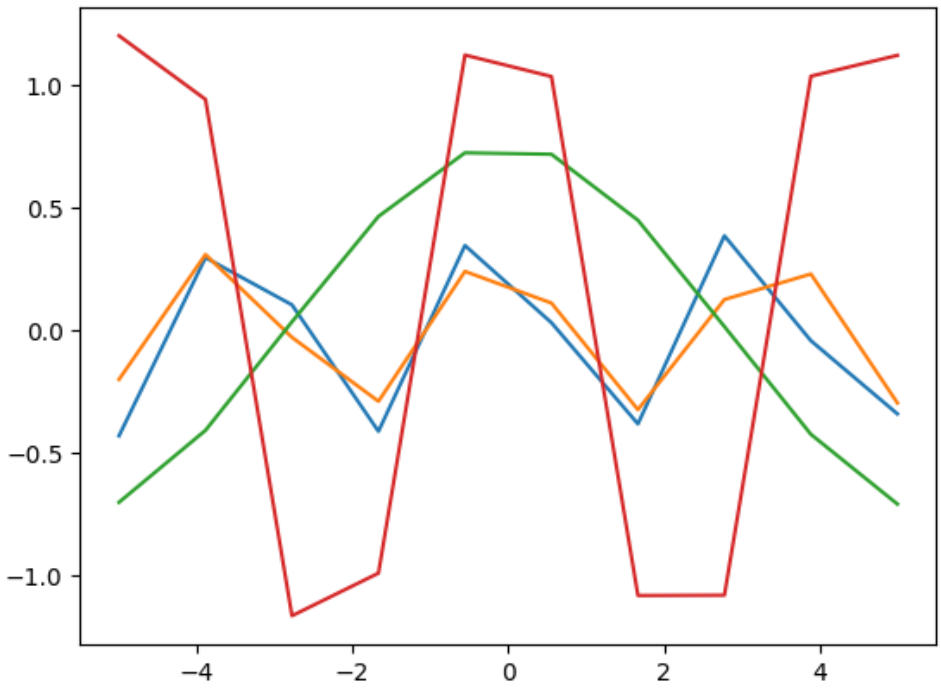
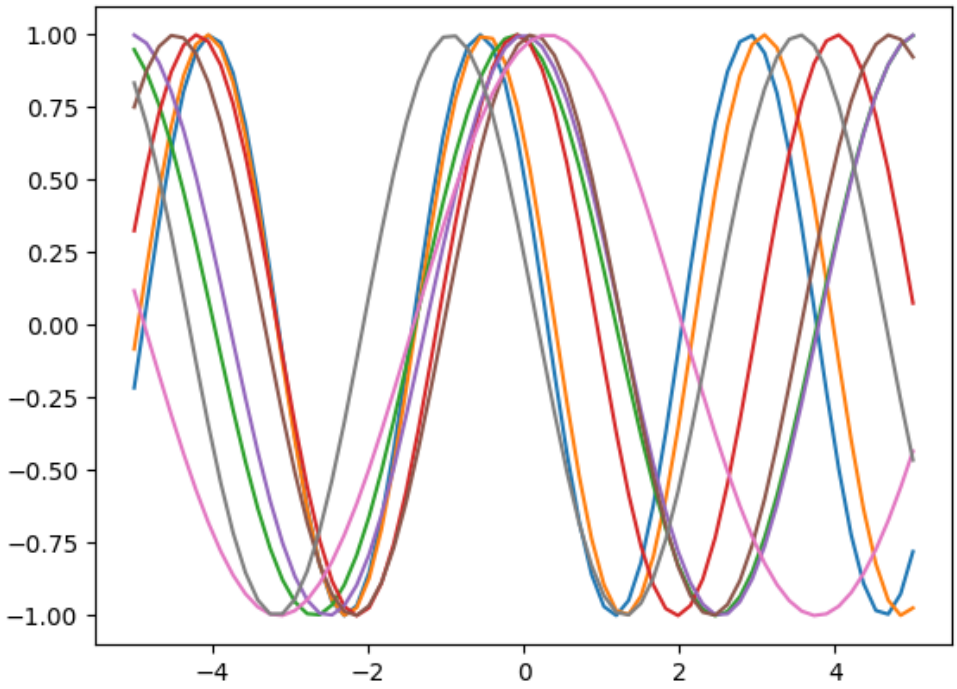
```
#     GAN

#https://www.codemotion.com/magazine/ai-ml/deep-learning/how-to-build-a-gan-in-python/

import matplotlib.pyplot as plt
import numpy as np
from numpy.random import randint, uniform

X_MIN = -5.0
X_MAX = 5.0
SAMPLE_LEN=10
X_COORDS = np.linspace(X_MIN , X_MAX, SAMPLE_LEN)
fig, axis = plt.subplots(1, 1)
for i in range(4):
    axis.plot(X_COORDS, uniform(0.1,2.0)*np.sin(uniform(0.2,2.0)*X_COORDS + uniform(2)))
```




```
import numpy as np
from numpy.random import uniform

import matplotlib.pyplot as plt

SAMPLE_LEN = 64       # number N of points where a curve is sampled
SAMPLE_SIZE = 32768   # number of curves in the training set
X_MIN = -5.0          # least ordinate where to sample
X_MAX = 5.0           # last ordinate where to sample

# The set of coordinates over which curves are sampled
X_COORDS = np.linspace(X_MIN , X_MAX, SAMPLE_LEN)

# The training set
SAMPLE = np.zeros((SAMPLE_SIZE, SAMPLE_LEN))
for i in range(0, SAMPLE_SIZE):
    b = uniform(0.5, 2.0)
    c = uniform(np.math.pi)
    SAMPLE[i] = np.array([np.sin(b*x + c) for x in X_COORDS])
```

```
# We plot the first 8 curves
fig, axis = plt.subplots(1, 1)
for i in range(8):
    axis.plot(X_COORDS, SAMPLE[i])
```




```
from keras.models import Sequential
from keras.layers import Dense, Dropout, LeakyReLU

DROPOUT = Dropout(0.4)        # Empirical hyperparameter
discriminator = Sequential()
discriminator.add(Dense(SAMPLE_LEN, activation="relu"))
discriminator.add(DROPOUT)
discriminator.add(Dense(SAMPLE_LEN, activation="relu"))
discriminator.add(DROPOUT)
discriminator.add(Dense(1, activation = "sigmoid"))
discriminator.compile(optimizer = "adam", loss = "binary_crossentropy", metrics = ["accuracy"])


LEAKY_RELU = LeakyReLU(0.2)   # Empirical hyperparameter
generator = Sequential()
generator.add(Dense(SAMPLE_LEN))
generator.add(LEAKY_RELU)
generator.add(Dense(512))
generator.add(LEAKY_RELU)
generator.add(Dense(SAMPLE_LEN, activation = "tanh"))
generator.compile(optimizer = "adam", loss = "mse", metrics = ["accuracy"])


gan = Sequential()
gan.add(generator)
gan.add(discriminator)
gan.compile(optimizer = "adam", loss = "binary_crossentropy", metrics = ["accuracy"])
```

```
EPOCHS = 3
BATCH=10

NOISE = uniform(X_MIN, X_MAX, size = (SAMPLE_SIZE, SAMPLE_LEN))
ONES = np.ones((SAMPLE_SIZE))
ZEROS = np.zeros((SAMPLE_SIZE))
print("epoch | dis. loss | dis. acc | gen. loss | gen. acc")
print("------+-----------+----------+-----------+----------")

fig = plt.figure(figsize = (8, 12))
ax_index = 1
for e in range(EPOCHS):
    for k in range(SAMPLE_SIZE//BATCH):
        # Addestra il discriminatore a riconoscere le sinusoidi vere da quelle prodotte dal genera
        n = randint(0, SAMPLE_SIZE, size = BATCH)
        # Ora prepara un batch di training record per il discriminatore
        p = generator.predict(NOISE[n])
        x = np.concatenate((SAMPLE[n], p))
        y = np.concatenate((ONES[n], ZEROS[n]))
        d_result = discriminator.train_on_batch(x, y)
        discriminator.trainable = False
        g_result = gan.train_on_batch(NOISE[n], ONES[n])
        discriminator.trainable = True
    print(f" {e:04n} |  {d_result[0]:.5f}  |  {d_result[1]:.5f} |  {g_result[0]:.5f}  |  {d_result
    # At 3, 13, 23, ... plots the last generator prediction
    if e % 10 == 3:
        ax = fig.add_subplot(8, 1, ax_index)
        plt.plot(X_COORDS, p[-1])
        ax.xaxis.set_visible(False)
        plt.ylabel(f"Epoch: {e}")
        ax_index += 1

# Plots a curve generated by the GAN
y = generator.predict(uniform(X_MIN, X_MAX, size = (1, SAMPLE_LEN)))[0]
ax = fig.add_subplot(8, 1, ax_index)
plt.plot(X_COORDS, y)
```

```
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 56ms/step
 0002 |  0.95137  |  0.50000 |  0.25160  |  0.50000
1/1 [==============================] - 0s 32ms/step
[<matplotlib.lines.Line2D at 0x7aa57904e5f0>]
```