```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt


# Load the MNIST dataset
(x_train, _), (_, _) = tf.keras.datasets.mnist.load_data()
x_train = x_train.astype("float32") / 255.0
x_train = np.expand_dims(x_train, axis=-1)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
# Define the generator
def build_generator(noise_dim):
    model = tf.keras.Sequential([
        tf.keras.layers.Dense(256, activation="relu", input_dim=noise_dim),
        tf.keras.layers.Dense(512, activation="relu"),
        tf.keras.layers.Dense(28 * 28, activation="sigmoid"),
        tf.keras.layers.Reshape((28, 28, 1))
    ])
    return model


# Define the discriminator
def build_discriminator():
    model = tf.keras.Sequential([
        tf.keras.layers.Flatten(input_shape=(28, 28, 1)),
        tf.keras.layers.Dense(512, activation="relu"),
        tf.keras.layers.Dense(256, activation="relu"),
        tf.keras.layers.Dense(1, activation="sigmoid")
    ])
    return model


# Compile GAN
def compile_gan(generator, discriminator):
    discriminator.compile(optimizer="adam", loss="binary_crossentropy", metrics=["accuracy"])
    discriminator.trainable = False
    gan_input = tf.keras.Input(shape=(generator.input_shape[1],))
    gan_output = discriminator(generator(gan_input))
    gan = tf.keras.Model(gan_input, gan_output)
    gan.compile(optimizer="adam", loss="binary_crossentropy")
    return gan


# Train GAN
def train_gan(generator, discriminator, gan, noise_dim, epochs=10, batch_size=128):
    for epoch in range(epochs):
        idx = np.random.randint(0, x_train.shape[0], batch_size)
        real_images = x_train[idx]
        noise = np.random.normal(0, 1, (batch_size, noise_dim))
        fake_images = generator.predict(noise)

        # Train discriminator
        d_loss_real = discriminator.train_on_batch(real_images, np.ones((batch_size, 1)))
        d_loss_fake = discriminator.train_on_batch(fake_images, np.zeros((batch_size, 1)))
        d_loss = 0.5 * np.add(d_loss_real, d_loss_fake)

        # Train generator
        noise = np.random.normal(0, 1, (batch_size, noise_dim))
        g_loss = gan.train_on_batch(noise, np.ones((batch_size, 1)))

        print(f"Epoch {epoch+1}, D Loss: {d_loss[0]}, D Accuracy: {d_loss[1]}, G Loss: {g_loss}")



# Evaluate performance with different noise dimensions
noise_dims = [100, 200, 500, 1024]
results = {}

for noise_dim in noise_dims:
    print(f"\nTraining with noise dimension: {noise_dim}")
    generator = build_generator(noise_dim)
    discriminator = build_discriminator()
    gan = compile_gan(generator, discriminator)
    train_gan(generator, discriminator, gan, noise_dim)

    # Generate samples
    noise = np.random.normal(0, 1, (10, noise_dim))
    generated_images = generator.predict(noise)
```

```
        # Save performance results
        results[noise_dim] = generated_images

    # Display generated images
    plt.figure(figsize=(10, 10))
    for i, noise_dim in enumerate(noise_dims):
        for j in range(10):
            plt.subplot(len(noise_dims), 10, i * 10 + j + 1)
            plt.imshow(results[noise_dim][j].squeeze(), cmap='gray')
            plt.axis("off")
    plt.show()
```

```
Training with noise dimension: 100
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` arg
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
/usr/local/lib/python3.11/dist-packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not pass an `input_shape`/`input_
  super().__init__(**kwargs)
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
/usr/local/lib/python3.11/dist-packages/keras/src/backend/tensorflow/trainer.py:82: UserWarning: The model does not have any trainab
  warnings.warn("The model does not have any trainable weights.")
Epoch 1, D Loss: 0.7414190769195557, D Accuracy: 0.474609375, G Loss: 0.4601284861564636
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
Epoch 2, D Loss: 0.8319919109344482, D Accuracy: 0.4124348759651184, G Loss: 0.38725996017456055
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
Epoch 3, D Loss: 0.9194599390029907, D Accuracy: 0.39531248807907104, G Loss: 0.326714426279068
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
Epoch 4, D Loss: 1.0134928226470947, D Accuracy: 0.38818359375, G Loss: 0.2760734558105469
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
Epoch 5, D Loss: 1.122746229171753, D Accuracy: 0.3826389014720917, G Loss: 0.23500053584575653
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
Epoch 6, D Loss: 1.2439346313476562, D Accuracy: 0.3736683130264282, G Loss: 0.2021654099225998
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
Epoch 7, D Loss: 1.3753150701522827, D Accuracy: 0.3708791136741638, G Loss: 0.17621636390686035
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
Epoch 8, D Loss: 1.5095851421356201, D Accuracy: 0.3738769590854645, G Loss: 0.15563121438026428
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
Epoch 9, D Loss: 1.645202398300171, D Accuracy: 0.3721788227558136, G Loss: 0.13905708491802216
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
Epoch 10, D Loss: 1.7795333862304688, D Accuracy: 0.367228627204895, G Loss: 0.12558674812316895
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step

Training with noise dimension: 200
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
Epoch 1, D Loss: 0.8669067621231079, D Accuracy: 0.25, G Loss: 1.7759363651275635
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
Epoch 2, D Loss: 0.7315696477890015, D Accuracy: 0.4166666865348816, G Loss: 1.4756134748458862
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
Epoch 3, D Loss: 0.7528221011161804, D Accuracy: 0.42330729961395264, G Loss: 1.2279915809631348
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
Epoch 4, D Loss: 0.8055453300476074, D Accuracy: 0.35888671875, G Loss: 1.0359220504760742
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
Epoch 5, D Loss: 0.8742988109588623, D Accuracy: 0.28285589814186096, G Loss: 0.8891278505325317
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step
Epoch 6, D Loss: 0.9466871023178101, D Accuracy: 0.23345762491226196, G Loss: 0.7754231095314026
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
Epoch 7, D Loss: 1.0199613571166992, D Accuracy: 0.1987680196762085, G Loss: 0.686280369758606
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
Epoch 8, D Loss: 1.0926547050476074, D Accuracy: 0.17306315898895264, G Loss: 0.6147134304046631
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
Epoch 9, D Loss: 1.1636055707931519, D Accuracy: 0.1532500982284546, G Loss: 0.5563347339630127
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
Epoch 10, D Loss: 1.233201265335083, D Accuracy: 0.1375102698802948, G Loss: 0.5077811479568481
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step

Training with noise dimension: 500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
Epoch 1, D Loss: 0.7351897954940796, D Accuracy: 0.697265625, G Loss: 0.35634422302246094
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
Epoch 2, D Loss: 0.9095140695571899, D Accuracy: 0.5514322519302368, G Loss: 0.2671689987182617
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
Epoch 3, D Loss: 1.0686841011047363, D Accuracy: 0.5199218988418579, G Loss: 0.20829981565475464
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
Epoch 4, D Loss: 1.2320646047592163, D Accuracy: 0.50537109375, G Loss: 0.16767537593841553
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
Epoch 5, D Loss: 1.3981640338897705, D Accuracy: 0.49726563692092896, G Loss: 0.1388486921787262
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
Epoch 6, D Loss: 1.561867594718933, D Accuracy: 0.4914180636405945, G Loss: 0.11787635087966919
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
Epoch 7, D Loss: 1.7199187278747559, D Accuracy: 0.48735833168029785, G Loss: 0.10206397622823715
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
Epoch 8, D Loss: 1.872570276260376, D Accuracy: 0.4863932430744171, G Loss: 0.08982905000448227
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
Epoch 9, D Loss: 2.0186004638671875, D Accuracy: 0.4847707152366638, G Loss: 0.08014722913503647
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
Epoch 10, D Loss: 2.1557717323303223, D Accuracy: 0.48309004306793213, G Loss: 0.07231682538986206
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step

Training with noise dimension: 1024
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
Epoch 1, D Loss: 0.7448089122772217, D Accuracy: 0.404296875, G Loss: 0.5116912722587585
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
Epoch 2, D Loss: 0.8705601096153259, D Accuracy: 0.3098958134651184, G Loss: 0.36592480540275574
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
Epoch 3, D Loss: 1.0424602031707764, D Accuracy: 0.2750000059604645, G Loss: 0.2775971591472626
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
Epoch 4, D Loss: 1.2158540487289429, D Accuracy: 0.2657645046710968, G Loss: 0.22092756628990173
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
Epoch 5, D Loss: 1.3876053094863892, D Accuracy: 0.26141494512557983, G Loss: 0.1818096935749054
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
Epoch 6, D Loss: 1.5568225383758545, D Accuracy: 0.2579604685306549, G Loss: 0.1538056880235672
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
```