**Generative Adversarial Networks (GANs):**

Generative Adversarial Networks (GANs) are a class of artificial intelligence algorithms used in unsupervised machine learning, implemented by a system of two neural networks contesting with each other in a zero-sum game framework. GANs have been widely applied in generating realistic synthetic data, such as images, audio, and text.

**Generator Model:**

The make\_generator\_model function defines the architecture of the generator neural network. This generator takes random noise vectors as input and produces synthetic images as output. Here's a breakdown of its architecture:

**Input Layer:**

A dense layer takes a 100-dimensional random noise vector as input.

**Batch Normalization and Activation**:

Batch normalization is applied followed by a LeakyReLU activation function.

**Reshape Layer:**

Reshapes the output to a 3D tensor.

**Convolutional Transpose Layers:**

Transpose convolutional layers are used to upscale the tensor gradually to generate an image. Each layer is followed by batch normalization and **LeakyReLU** activation.

**Output Layer:**

The final convolutional transpose layer outputs a 28x28x1 tensor representing the generated image with pixel values ranging from -1 to 1 (thanks to the tanh activation function).

**Generating Images:**

Generator Creation: The make\_generator\_model function is called to create an instance of the generator model.

**Random Noise Vector:**

A random noise vector of shape [1, 100] is generated using TensorFlow's tf.random.normal function.

**Image Generation:**

The generator model processes the noise vector to generate an image. The training argument is set to False to ensure that batch normalization layers run in inference mode.

**Displaying the Image:**

The generated image is displayed using Matplotlib. Since the image is grayscale, it is displayed using the 'black' colormap.

This code demonstrates the basic functionality of a GAN by generating a synthetic image using a pre-trained generator model. Keep in mind that training a GAN from scratch requires a substantial dataset and computational resources. Additionally, tuning hyperparameters and optimizing the architecture are crucial for generating high-quality images.