1. **Generator Architecture**:
   * The generator is responsible for generating fake images that resemble real images from the dataset.
   * It starts with a dense layer that takes random noise vectors of size **latent\_dim** as input.
   * The dense layer is reshaped to a 3D tensor and then passed through a series of convolutional layers.
   * Each convolutional layer is followed by batch normalization and leaky ReLU activation.
   * The final layer uses a convolutional transpose layer to upsample the generated features to the desired image size.
   * The output layer uses the hyperbolic tangent activation function to ensure pixel values are in the range [-1, 1].
2. **Discriminator Architecture**:
   * The discriminator is a binary classifier that distinguishes between real and fake images.
   * It takes images of size **(SIZE, SIZE, 3)** as input.
   * It consists of several convolutional layers followed by batch normalization and leaky ReLU activation.
   * The output layer is a dense layer with a sigmoid activation function, which outputs a probability score indicating the likelihood of the input image being real.
3. **Training Steps**:
   * The training steps involve optimizing the generator and discriminator simultaneously using adversarial training.
   * In each training step, the generator generates fake images from random noise, and the discriminator classifies both real and fake images.
   * The loss functions for both the generator and discriminator are computed based on the discriminator's classification results.
   * The gradients of the loss functions are used to update the weights of both the generator and discriminator using the RMSprop optimizer.

Why is latent\_dim set to 100?

The latent\_dim represents the dimensionality of the latent space, which is the space of random noise vectors that the generator uses as input to generate fake images. The choice of 100 for latent\_dim is somewhat arbitrary and is a common practice in many GAN implementations. It's often chosen as a compromise between having enough dimensions to capture the variability in the data and keeping the computational complexity manageable.

Why is RMSprop optimizer used and not Adam?

Both RMSprop and Adam are popular optimization algorithms used in deep learning. RMSprop tends to work well for GANs because it adapts the learning rate for each parameter individually based on the magnitude of the gradients, which can help stabilize training. The choice between RMSprop and Adam can depend on the specific problem and empirical performance.

What is clipvalue?

clipvalue is a parameter used in the RMSprop optimizer that bounds the magnitude of the gradients during training. Gradients with magnitudes larger than clipvalue are clipped to be no larger than clipvalue. This can help prevent gradient explosion during training.

Why is tanh function used for the generator and not any other?

The tanh activation function is commonly used in the output layer of the generator because it squashes the output values to the range [-1, 1]. This matches the range of pixel values typically seen in images (where 0 represents black, -1 represents white, and values in between represent various shades of gray). Using tanh helps ensure that the generated images have realistic pixel values.

What is batch normalization?

Batch normalization is a technique used to normalize the activations of each layer in a neural network by adjusting and scaling the activations based on the mean and variance of the mini-batch. This can help stabilize and speed up training by reducing internal covariate shift and ensuring that the activations are centered and have a consistent scale.

Why is upsampling and downsampling used? How will it help?

Upsampling and downsampling are used in convolutional neural networks, including GANs, to change the spatial dimensions of feature maps. Downsampling (e.g., using pooling or strided convolutions) reduces the spatial dimensions while increasing the number of channels, capturing larger-scale features. Upsampling (e.g., using transposed convolutions or nearest-neighbor interpolation) increases the spatial dimensions while decreasing the number of channels, allowing the network to generate finer details.

What is padding='same'?

padding='same' is a parameter used in convolutional layers that pads the input feature map with zeros so that the output feature map has the same spatial dimensions as the input. This helps preserve spatial information and ensures that the output size matches the input size when using stride 1 convolutions.

What is kernel\_initializer?

kernel\_initializer is a parameter used to initialize the weights of the convolutional kernels in the layer. It specifies the method for initializing the weights, such as random initialization from a normal distribution ('he\_normal' or 'glorot\_uniform') or constant initialization.

Why is use\_bias=False?

Setting use\_bias=False in convolutional layers means that the layer does not use bias parameters. This can help reduce the number of parameters in the model and can sometimes improve training stability, especially when batch normalization is used.

What is from\_logits?

from\_logits is a parameter used in loss functions to specify whether the input values are raw logits (i.e., unnormalized scores) or probabilities. Setting from\_logits=True indicates that the input values are logits, and the loss function will apply the appropriate transformations (e.g., softmax) internally.

noise = np.random.normal(0,1,(batch\_size,latent\_dim)) - What does this mean?

This line generates random noise samples from a normal distribution with mean 0 and standard deviation 1. The generated noise has a shape of (batch\_size, latent\_dim), where batch\_size represents the number of samples in the batch, and latent\_dim represents the dimensionality of each noise vector.

What is GradientTape?

GradientTape is a TensorFlow API for automatic differentiation. It records operations for computing gradients during the forward pass of a computation, allowing TensorFlow to compute gradients with respect to variables (e.g., model parameters) later during the backward pass. It's commonly used in custom training loops to compute and apply gradients for optimization.