# PARKING DETECTION SYSTEM

## Training the GAN Model:

### 1. Generator and Discriminator Classes:

- The code defines two classes: Generator and Discriminator, representing the generator and discriminator parts of the GAN architecture.
- The generator creates synthetic parking space images, and the discriminator distinguishes between real and generated images.

### 2. Hyperparameters and Data Loading:

- Hyperparameters like input size, output size, batch size, and epochs are set.
- The dataset is loaded using a custom GAN Dataset class, which reads image paths and annotations from a CSV file.

### 3. Training Loop:

- The training loop iterates through the dataset for a specified number of epochs.
- For each batch, random noise is generated (noise) and used to create synthetic images with the generator.
- The discriminator is trained to distinguish between real and generated images.
- The generator is trained to generate images that the discriminator classifies as real.

### 4. Loss Calculation and Optimization:

- Binary Cross Entropy Loss is used for both the discriminator and generator.
- Separate optimizers are defined for the generator and discriminator.

### 5. Model Saving:

The trained models are saved in separate files.

## Making Predictions:

**1. Generator Initialization and Load Pretrained Weights:**

- A new generator is initialized with the same architecture as the trained one.
- Pretrained weights for the generator are loaded from the saved file.

**2. Prediction Function:**

- A function is defined to generate predictions using the trained generator.
- It generates random noise, passes it through the generator, and saves the result.

**3. Resize and Save Predictions:**

- The predictions are initially reshaped to have spatial dimensions.
- The code uses the F.interpolate function to resize the generated image tensor to the desired size.
- The final resized image is saved.

The example usage section shows how to use the prediction function on a test image. The predicted parking place is saved, and further processing or visualization can be done. The code is organized into functions and classes, making it modular and easy to understand. Hyperparameters are clearly defined at the beginning of the script. The training loop and prediction function are separated for clarity.

**Thanks**