BCSE209L – Machine Learning

Enhancing Malware Detection and Classification with CNN-GAN Integration

By

21BCE1660 Neelam Naga Saivenkata Suryavenu
21BCE1553 Dhashwanth Bharathi
21BCE1449 Madhavan Raman

Submitted to

Dr. R. Rajalakshmi

School of Computer Science and Engineering

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Documentation of the Code file :

Installation of Dependencies:

The code snippet includes the installation of necessary packages like opendatasets and other dependencies. Installing dependencies ensures that all required libraries, tools, and frameworks are available for executing the generative model code. It aids in setting up the development environment with essential components necessary for model training and evaluation.

Downloading Dataset:

The code fetches a malware dataset from Kaggle utilizing the opendatasets library. This dataset likely comprises a collection of malware samples labeled with their respective classes for training and testing the generative model. Accessing relevant datasets from platforms like Kaggle streamlines the process of acquiring data for machine learning projects.

Data Processing:

The code involves preprocessing image data from a designated directory using the ImageDataGenerator. ImageDataGenerator is a utility in Keras that applies real-time data augmentation and preprocessing to image data during model training. Data processing tasks may include resizing images, normalizing pixel values, and applying augmentation techniques like rotation or flipping.

Our CNN Model (Trained for 20 epochs):

The provided code demonstrates the training of a Convolutional Neural Network (CNN) model for 20 epochs on a malware dataset. The dataset consists of grayscale images of malware samples, each labeled with its respective class for classification tasks.

Model Architecture:

The CNN model architecture comprises multiple convolutional layers followed by max-pooling layers and dropout regularization to prevent overfitting. Specifically, it consists of:

Two sets of convolutional layers with ReLU activation functions, followed by max-pooling and dropout layers.

Flattening layer to convert the 2D feature maps into a vector.

Dense (fully connected) layers with ReLU activation and dropout regularization.

Output layer with softmax activation to predict the probability distribution over the 25 malware classes.

Training and Evaluation:

The model is trained using the Adam optimizer with a categorical cross-entropy loss function. During training, the model learns to minimize the loss and improve accuracy on both the training and validation sets. After 20 epochs of training, the model achieves a training accuracy of approximately 95.83% and a validation accuracy of about 95.44%.

Performance Evaluation:

The trained model is evaluated on the test set, yielding a test accuracy of around 95.84%. This indicates that the model generalizes well to unseen data, demonstrating its effectiveness in classifying malware samples.

Generating Images:

The provided code involves generating images utilizing the trained CNN model and saving one of the generated images as a PNG file named 'this.png'. This process typically involves passing random latent points into the generator component of a GAN model to produce synthetic images. These generated images serve various purposes such as visual inspection, performance evaluation of the model, or further analysis.

Defining the Discriminator:

The code defines a discriminator neural network model for adversarial training within a GAN setup. The discriminator's role is to differentiate between real and fake images generated by the generator. It typically comprises convolutional and dense layers with activation functions like LeakyReLU to learn discriminative features from images.

Defining the Generator:

The provided code defines a generator neural network model responsible for generating fake images in the GAN setup. The generator takes random noise (latent points) as input and learns to produce realistic images. It often consists of transposed convolutional layers to upsample the input noise into a meaningful image representation.

Creating Directories:

Directories are created to store datasets and generated images. Proper directory organization ensures the structured storage of datasets, model checkpoints, logs, and generated outputs. It facilitates efficient management and accessibility of data and results throughout the model development and evaluation stages.

Defining the GAN Model:
This snippet combines the generator and discriminator models to form a GAN model. In a GAN setup, the generator aims to produce images that deceive the discriminator, while the discriminator aims to distinguish real from fake images. The GAN model is trained iteratively, with both components competing against each other to enhance their performance.

Compiling Models:
The models are compiled with specific hyperparameters such as the learning rate and beta value for the Adam optimizer. This compilation step configures the models for training by specifying the loss function, optimizer, and evaluation metrics. Binary cross-entropy loss is commonly used in GANs to optimize both the generator and discriminator networks.

Conclusion :

The documentation highlights the significance of data preprocessing, model architecture, training, and evaluation in building an effective malware detection system. Leveraging techniques such as CNNs and GANs, the model demonstrates promising results in accurately classifying malware samples while also generating synthetic images for further analysis.

Moreover, the documentation underscores the importance of proper directory organization and model compilation in streamlining the development and evaluation workflow. By creating directories for storing datasets, generated images, and model checkpoints, researchers can efficiently manage and access data throughout the project lifecycle.

Overall, the detailed documentation serves as a valuable resource for researchers and practitioners in the field of cybersecurity. It provides insights into the methodologies and techniques employed in developing advanced malware detection systems and offers a foundation for further research and experimentation in this critical domain.