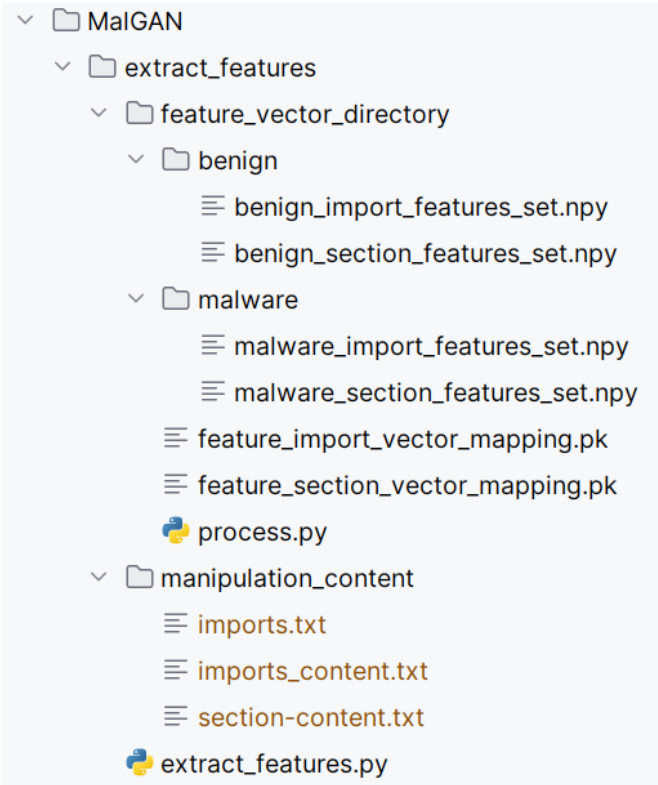
GRAPES--help文档

一.MalGAN训练

1.数据准备



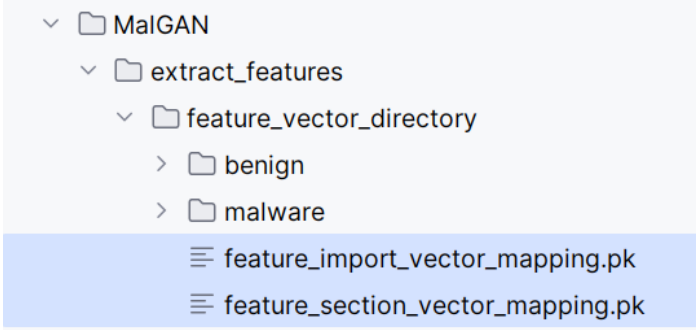
运行**MalGAN/extract_features/extract_features.py**文件

输入：

- -m:所有的PE恶意软件样本所在的目录（malware_rl/envs/utils/samples/benign）
- -b:所有的PE良性软件样本所在的目录 (malware_rl/envs/utils/samples/malware)
- -o:输出内容的路径（MalGAN/extract_features/feature_vector_directory）

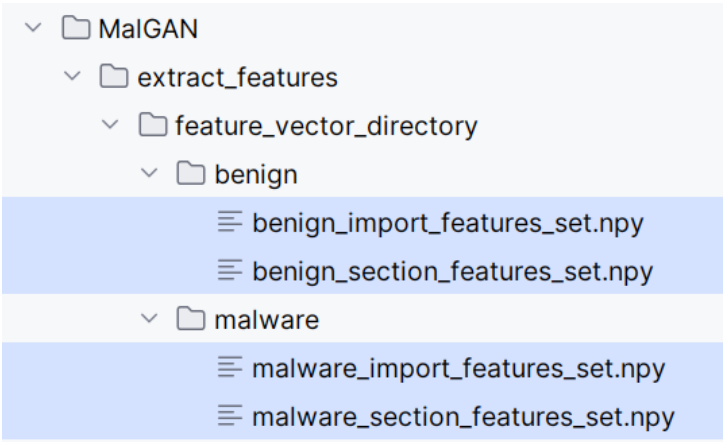
输出：

- feature_vector_directory中关于所有输入的PE恶意和良性的文件的节和导入表的feature_mapping

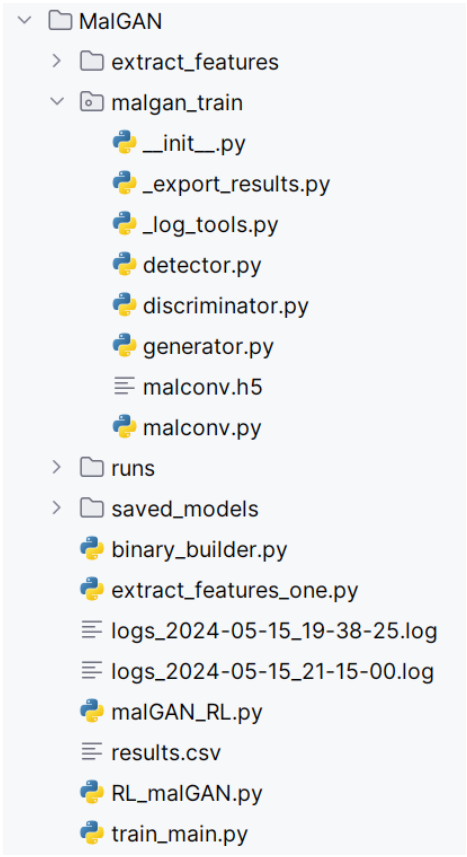


- benign、malware文件夹中每个样本关于节和导入表的特征向量（相对于上述的feature_mapping）

下图是将所有的pk文件转换为npy



2.训练malGAN



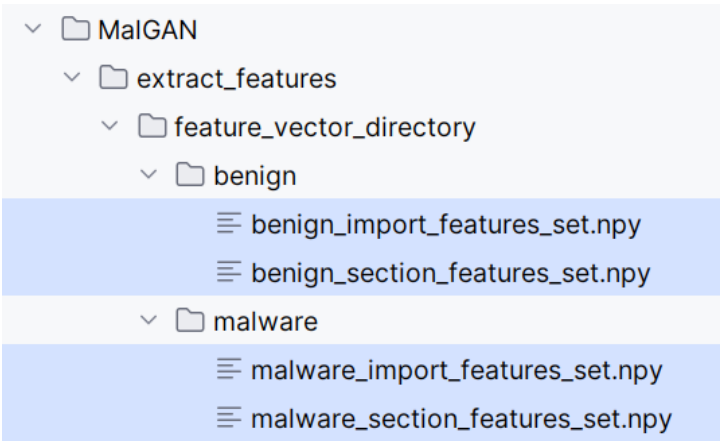
1.MalGAN/malgan_train目录解读

- generator.py:malGAN的生成器的网络结构
- discriminator.py:malGAN的鉴别器的网络结构
- detector.py:黑盒检测器（五个机器学习模型:DecisionTree,LogisticRegression,MultiLayerPerceptron,RandomForest,SVM）
- malconv.py(malconv.h5):malconv黑盒检测器
- _log_tools.py:日志配置
- _init_.py:malGAN训练代码
- _export_results.py:导出测试结果

2.运行MalGAN/train_main.py，即可开始训练malGAN

要求从控制台输入变量：

- Z：噪声向量的维度
- batch_size:dataloader的批次
- num_epoch:训练的轮数
- --gen-hidden-sizes：malGAN生成器的隐藏层的维度
- --discrim-hidden-sizes：malGAN判别器的隐藏层的维度
- --activation：激活函数的类型（ReLU, ELU,LeakyReLU, tanh,sigmoid）
- --detector：黑盒检测器的类型（DecisionTree,LogisticRegression,MultiLayerPerceptron,RandomForest,SVM,MalConv
- 特征向量的path



利用恶意和良性的关于节的特征向量训练malGAN（MalGAN/train_main.py），控制台输入：

```
cd MalGAN
```

```
python train_main.py --gen-hidden-sizes 256 512 256 --discrim-hidden-sizes 256 512 256 --activation ReLU --detector MalConv --print-results 10 30 10 MalGAN/extract_features/feature_vector_directory/malware/malware_section_features_set.npy MalGAN/extract_features/feature_vector_directory/benign/benign_section_features_set.npy
```

3.查看malGAN训练结果

MalGAN/saved_models：保存malGAN的模型参数

MalGAN/results.csv：保存评估训练malGAN的性能指标

至此，会根据训练malGAN的数据类型（节或者导入表），训练出对应的malGAN

二.PE恶意软件通过GRAPES开始逃逸

1.运行ppo.py

当智能体利用PE恶意软件样本的特征向量判断出下一步的action后，会进入malware_rl/envs/controls/modifier.py文件，得到修改的对抗性的PE恶意软件样本

2.分析malware_rl/envs/controls/modifier.py文件

```
class ModifyBinary:
    Depth
    def __init__(self, bytez, action):
        self.bytez = bytez
        self.trusted_path = module_path + "/trusted/all_benign/benign/"
        self.good_str_path = module_path + "/good_strings/"
        if (action in ['add_section_benign_data', 'rename_section']):
            features_type='section'
            features_vector=extract_features_one.features_mapping_section_index(bytez)
            adversarial_features=RL_malGAN.test_malGAN(features_vector,features_type)
            self.adversarial_sections=malGAN_RL.section_added_extractor(adversarial_features,bytez)
        if(action in ['add_imports']):
            features_type = 'import'
            features_vector=extract_features_one.features_mapping_import_index(bytez)
            adversarial_features=RL_malGAN.test_malGAN(features_vector,features_type)
            self.adversarial_imports=malGAN_RL.import_added_extractor(adversarial_features,bytez)
```

1. 判断如果**action是加入节或者导入表**，那么会利**用第一部分训练的malGAN**来帮助PE恶意软件样本更好的逃逸
2. 如果是**加入节**：那么**feature_type是section**（如果是加入导入表，那么feature_type是import）
3. 进入**MalGAN/extract_features_one.py**，得到**PE恶意软件样本中关于节的特征向量（one-hot编码）featue_vector**

4. 接着将**featue_vector**传入**MalGAN/RL_malGAN.py**，根据feature_type访问对应的训练好的关于节的malGAN，得到**对抗性的关于节的特征向量adversarial_features**
5. 接着将**adversarial_features**传入**MalGAN/malGAN_RL.py**，得到adversarial_features在featue_vector上增加的节的内容**self.adversarial_sections**。

这样就代替了原来从良性的节中随机选择的内容加入到PE恶意软件中，这样就会使得对抗性的PE恶意软件逃逸的可能性被增大。