Language: Python3.8
Framework: PyTorch
Libraries: Numpy, Pandas, PyTorch, Matplotlib, Sklearn
Environment: Jupyter Notebook

Task:
Apply **GAN** to augment data for “No Malware” class.

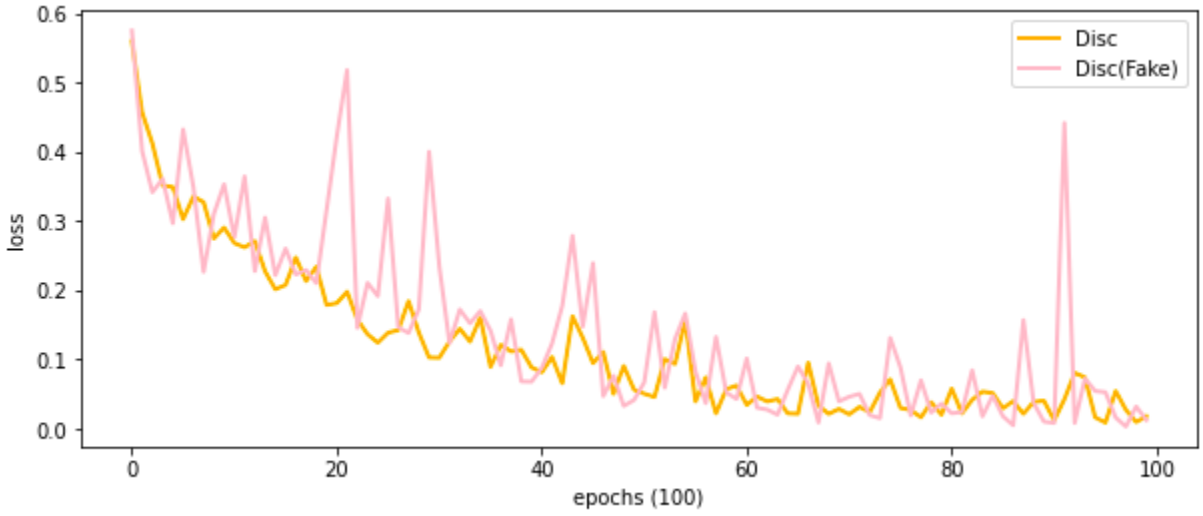
Outcome:
234 samples were generated for the “No Malware” class to balance the dataset.
MLP AUC: 98.27%

Models:

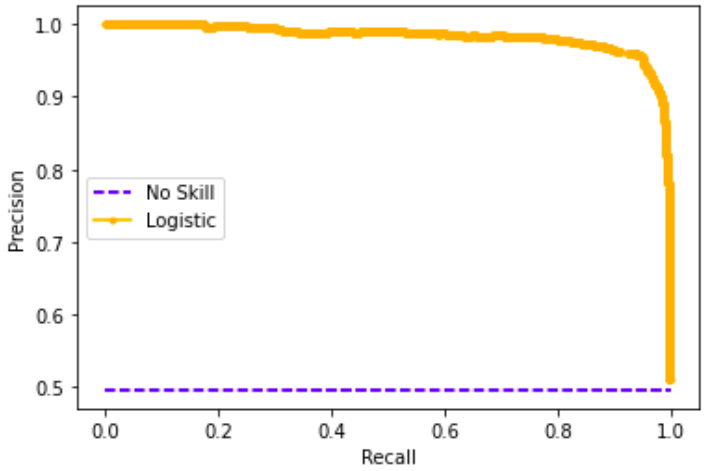
```
class Discriminator(nn.Module):
  def __init__(self, data_dim):
    super().__init__()
    self.disc = nn.Sequential(
        nn.Linear(data_dim, 128),
        nn.BatchNorm1d(128),
        nn.ReLU(),
        nn.Linear(128, 128),
        nn.BatchNorm1d(128),
        nn.ReLU(),
        nn.Linear(128, 1),
        nn.Sigmoid()
    )

  def forward(self, x):
    return self.disc(x)


class Generator(nn.Module):
  def __init__(self, z_dim, data_dim):
    super().__init__()
    self.gen = nn.Sequential(
        nn.Linear(z_dim, 128),
        nn.BatchNorm1d(128),
        nn.ReLU(),
        nn.Linear(128, 128),
        nn.BatchNorm1d(128),
        nn.ReLU(),
        nn.Linear(128, data_dim),
        nn.BatchNorm1d(data_dim)
    )

  def forward(self, x):
```

```
    return self.gen(x)
```

Training Loss:



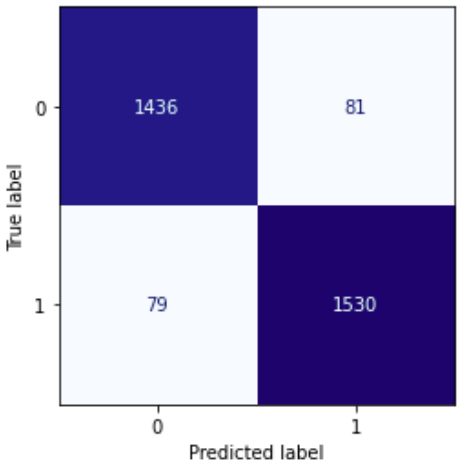
MLP on encoded balanced data:




```
F1: 0.9507692307692307
AUC: 0.9827364792091238
Accuracy 0.9510254055708601
```

Confusion Matrix:

Without Augmentation



With Augmentation