Relatório de Treinamento de Rede Neural para Classificação de Ataques em IoT

Introdução

O objetivo deste projeto é treinar uma rede neural para classificação de dados em um conjunto de dados de ataques IoT. A rede neural será desenvolvida utilizando a biblioteca TensorFlow com a interface Keras. O modelo será treinado para prever a classe correta para cada entrada de dados.

Conjunto de Dados

O conjunto de dados CicIoT2023 que foi desenvolvido pelo Instituto Canadense de Segurança Cibernética (CIC) é utilizado nesse projeto contendo 33 dados de ataques que são divididos em sete catégorias, sendo elas DDoS, DoS, Recon, Web-based, Brute Force, Spoofing e Mirai.

Pré-processamento dos Dados

Antes do treinamento do modelo, os dados foram pré-processados da seguinte forma:

Como os dados originais são compostos por 33 rotulos distintos. Foi modificado os rotulos conforme as classes de ataques, resultando em 8 classes para o treinamento do modelo. As classes estão listadas a seguir:

DDoS, DoS, Mirai, Benign, Spoofing, Recon,Web, BruteForce.

Normalização das características para garantir que todas tenham a mesma escala.

Divisão do conjunto de dados em conjuntos de treinamento, validação e teste na proporção X:Y:Z.

Foram criados bases de dados utilizando técnicas de smote e sampling para dados desbalanceados.

Arquitetura do Modelo

Foi utilizado três arquiteturas de rede neural para o treinamento com os dados de IoT Figuras 1,2,3.

```python
def modeloDense(X_train, metricas):
    model = tf.keras.models.Sequential([
        tf.keras.layers.Dense(500, activation="relu", input_shape=(X_train.shape[-1],)),
        tf.keras.layers.Dense(units=500, activation='relu', kernel_initializer="random_uniform", bias_initializer="random_uniform"),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(units=500, activation='relu', kernel_initializer="random_uniform", bias_initializer="random_uniform"),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(8,activation='softmax')

    ])
    model.compile(loss='categorical_crossentropy',optimizer=Adam(1e-3), metrics=metricas)
    return model
```

Figura 1: Rede densa 4 camadas.

```python
def modeloDense2(X_train, metricas):
    model = tf.keras.models.Sequential([
        tf.keras.layers.Dense(500, activation="relu", input_shape=(X_train.shape[-1],)),
        tf.keras.layers.Dense(units=500, activation='relu', kernel_initializer="random_uniform", bias_initializer="random_uniform"),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.Dense(units=500, activation='relu', kernel_initializer="random_uniform", bias_initializer="random_uniform"),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.Dense(units=500, activation='relu', kernel_initializer="random_uniform", bias_initializer="random_uniform"),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.Dense(units=500, activation='relu', kernel_initializer="random_uniform", bias_initializer="random_uniform"),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.Dense(8,activation='softmax')

    ])
    model.compile(loss='categorical_crossentropy',optimizer=Adam(1e-3), metrics=metricas)
    return model
```

Figura 2: Rede densa 6 camadas.

```python
def modeloDenseFinal(X_train, metricas):
    model = tf.keras.models.Sequential([
        tf.keras.layers.Dense(500, activation="relu", input_shape=(X_train.shape[-1],)),
        tf.keras.layers.Dense(units=500, activation='relu', kernel_initializer="random_uniform", bias_initializer="random_uniform"),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.Dense(units=500, activation='relu', kernel_initializer="random_uniform", bias_initializer="random_uniform"),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.Dense(units=500, activation='relu', kernel_initializer="random_uniform", bias_initializer="random_uniform"),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.Dense(units=500, activation='relu', kernel_initializer="random_uniform", bias_initializer="random_uniform"),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.Dense(units=500, activation='relu', kernel_initializer="random_uniform", bias_initializer="random_uniform"),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.Dense(units=500, activation='relu', kernel_initializer="random_uniform", bias_initializer="random_uniform"),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.Dense(units=500, activation='relu', kernel_initializer="random_uniform", bias_initializer="random_uniform"),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.Dense(8,activation='softmax')

    ])
    model.compile(loss='categorical_crossentropy',optimizer=Adam(1e-3), metrics=metricas)
    return model
```

Figura 3: Rede densa 9 camdas.

Treinamento do Modelo

Modelo 1 foi treinado utilizando o algoritmo de otimização Adam com os seguintes parâmetros:

Taxa de Aprendizado: 1e-3

Função de Perda: categorical_crossentropy

Métricas de Avaliação:False Negatives, False Positives, True Negatives, True Positives, Precision, Recall

O treinamento foi realizado por 200 épocas com um tamanho de lote de 300.

Modelo 2 foi treinado utilizando o algoritmo de otimização Adam com os seguintes parâmetros:
Taxa de Aprendizado: 1e-3
Função de Perda: categorical_crossentropy
Métricas de Avaliação:False Negatives, False Positives, True Negatives, True Positives, Precision, Recall
O treinamento foi realizado por 200 épocas com um tamanho de lote de 300.

Modelo 3 foi treinado utilizando o algoritmo de otimização Adam com os seguintes parâmetros:
Taxa de Aprendizado: 1e-3
Função de Perda: categorical_crossentropy
Métricas de Avaliação:False Negatives, False Positives, True Negatives, True Positives, Precision, Recall
O treinamento foi realizado por 200 épocas com um tamanho de lote de 300.

Avaliação do Modelo

Após o treinamento, os modelo foram avaliados utilizando o conjunto de teste separado. As seguintes métricas foram calculadas para avaliar o desempenho do modelo:
Precisão
Recall
F1-Score
Matriz de Confusão

Resultados

Os resultados da avaliação dos modelos estão listados abaixo:

Modelo 1

O Modelo 1 apresentou uma acurácia de 84% com os dados normais, 70% com os dados de sampling e 79% com os dados de SMOTE. No entanto, algumas classes tiveram um desempenho de classificação significativamente inferior a outras. As figuras abaixo ilustram esses resultados.

Dados de treinamento normais

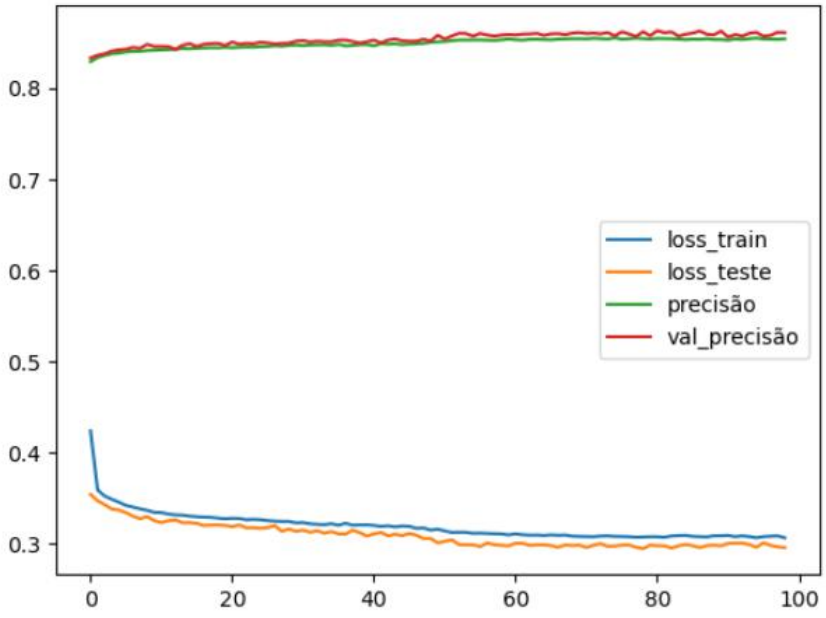

Figura 4: Tabela loss e precisão.

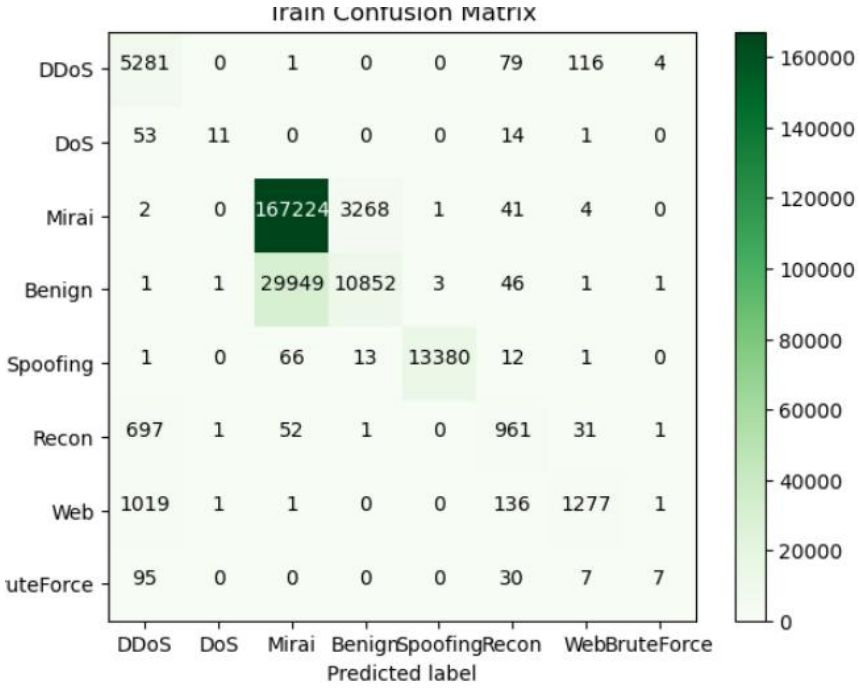

Figura 5: Matriz de confusão.

```
Accuracy: 0.8476985665296386

Micro Precision: 0.8476985665296386

Micro Recall: 0.8476985665296386

Micro F1_score: 0.8476985665296386

Macro Precision: 0.7820159336843588

Macro Recall: 0.5584824958624678

Macro F1_score: 0.5939478841865935

Weighted Precision: 0.8392009607494245

Weighted Recall: 0.8476985665296386

Weighted F1_score: 0.8175943768687111


Classification Report

              precision    recall  f1-score   support

       DDoS       0.74      0.96      0.84      5481
        DoS       0.79      0.14      0.24        79
      Mirai       0.85      0.98      0.91    170540
     Benign       0.77      0.27      0.39     40854
   Spoofing       1.00      0.99      1.00     13473
      Recon       0.73      0.55      0.63      1744
        Web       0.89      0.52      0.66      2435
 BruteForce       0.50      0.05      0.09       139

   accuracy                           0.85    234745
  macro avg       0.78      0.56      0.59    234745
weighted avg       0.84      0.85      0.82    234745
```

Figura 6: Métricas da rede.

Dados de treinamento sampling

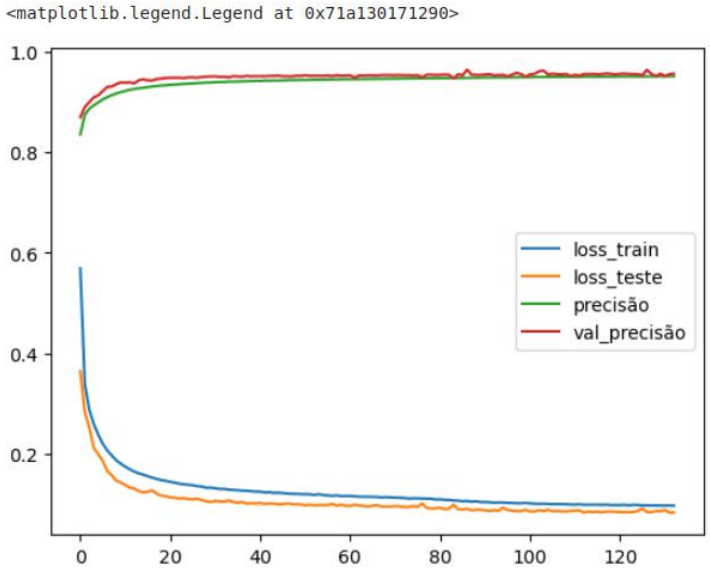

Figura 7: Tabela loss e precisão.

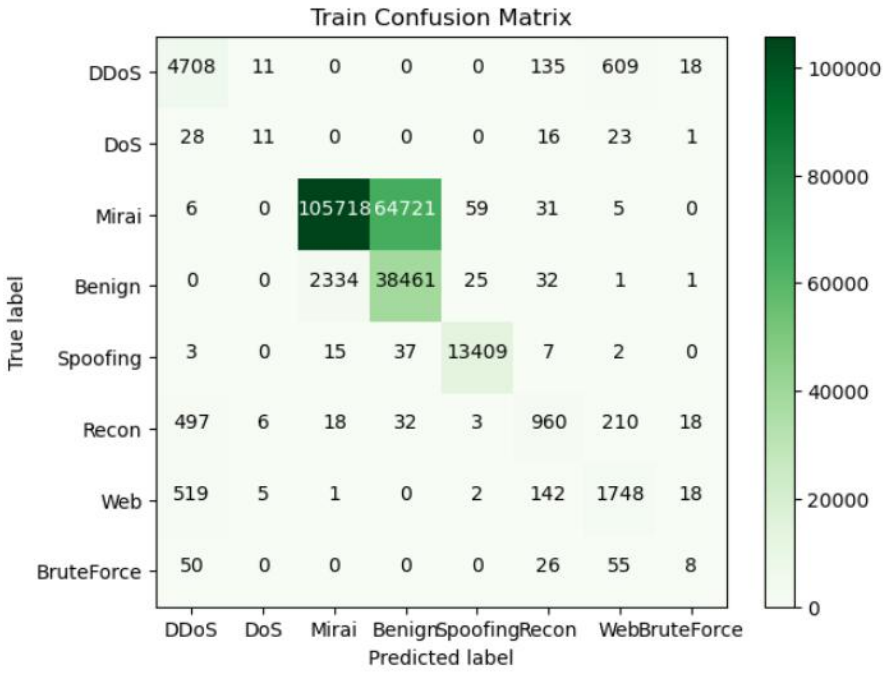

Figura 8: Matriz de confusão.

```
Accuracy: 0.7029883490596179

Micro Precision: 0.7029883490596179

Micro Recall: 0.7029883490596179

Micro F1_score: 0.7029883490596179

Macro Precision: 0.6228792346247424

Macro Recall: 0.6100827261549604

Macro F1_score: 0.5879931743933391

Weighted Precision: 0.86364274857046

Weighted Recall: 0.7029883490596179

Weighted F1_score: 0.7325879381357774

Classification Report

              precision    recall  f1-score   support

       DDoS       0.81      0.86      0.83      5481
        DoS       0.33      0.14      0.20        79
      Mirai       0.98      0.62      0.76    170540
     Benign       0.37      0.94      0.53     40854
   Spoofing       0.99      1.00      0.99     13473
      Recon       0.71      0.55      0.62      1744
        Web       0.66      0.72      0.69      2435
 BruteForce       0.12      0.06      0.08       139

   accuracy                           0.70    234745
  macro avg       0.62      0.61      0.59    234745
weighted avg       0.86      0.70      0.73    234745
```

Figura 9: Métricas da rede.

Dados de treinamento smote

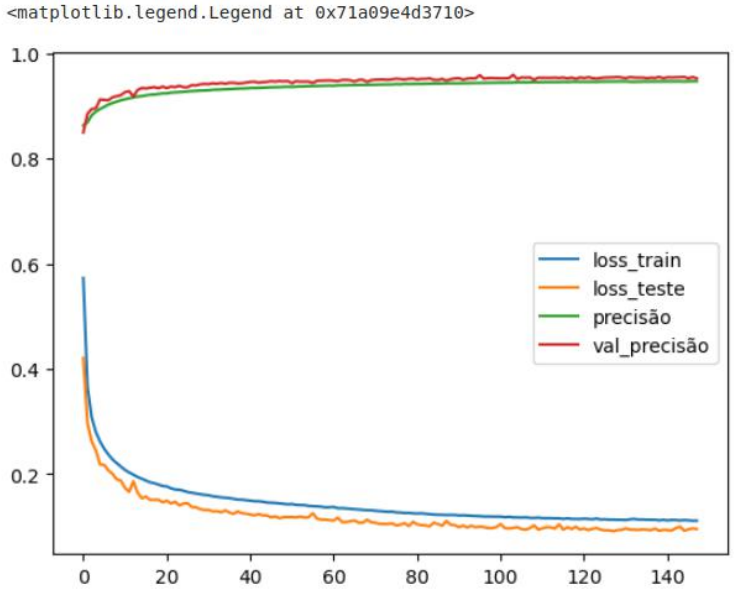

Figura 10: Tabela loss e precisão.

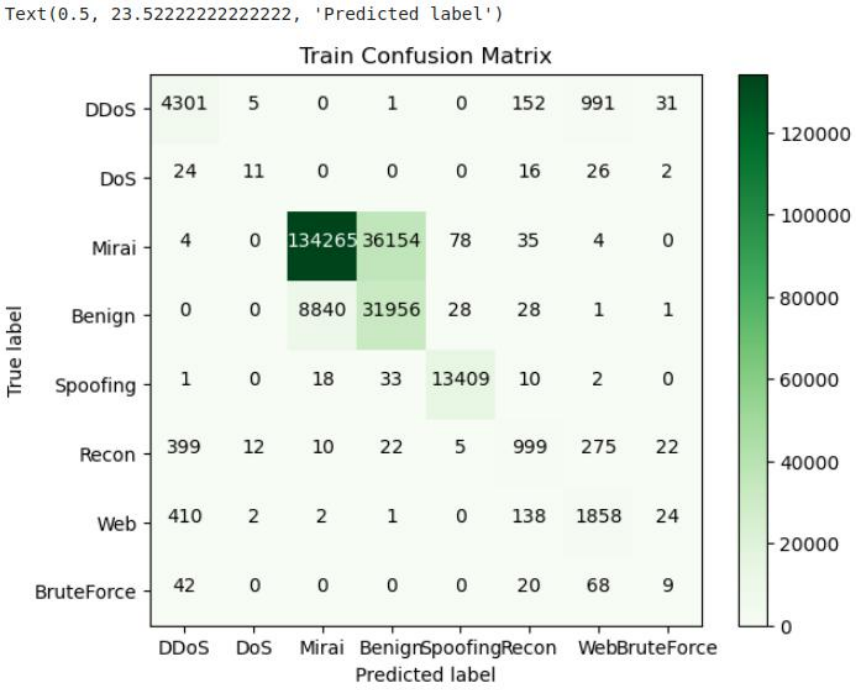

Figura 11: Matriz de confusão.

```
Accuracy: 0.795791177660866

Micro Precision: 0.795791177660866

Micro Recall: 0.795791177660866

Micro F1_score: 0.795791177660866

Macro Precision: 0.6234081928662608

Macro Recall: 0.6111628417532717

Macro F1_score: 0.6019800214101656

Weighted Precision: 0.8508306155475608

Weighted Recall: 0.795791177660866

Weighted F1_score: 0.8114666460219401


Classification Report

              precision    recall  f1-score   support

       DDoS       0.83      0.78      0.81      5481
        DoS       0.37      0.14      0.20        79
      Mirai       0.94      0.79      0.86    170540
     Benign       0.47      0.78      0.59     40854
   Spoofing       0.99      1.00      0.99     13473
      Recon       0.71      0.57      0.64      1744
        Web       0.58      0.76      0.66      2435
 BruteForce       0.10      0.06      0.08       139

   accuracy                           0.80    234745
  macro avg       0.62      0.61      0.60    234745
weighted avg       0.85      0.80      0.81    234745
```

Figura 12: Métricas da rede.

Modelo 2

O Modelo 2 apresentou uma acurácia de 84% com os dados normais, 72% com os dados de sampling e 76% com os dados de SMOTE. No entanto, algumas classes tiveram um desempenho de classificação significativamente inferior a outras. As figuras abaixo ilustram esses resultados.

Dados de treinamento normais

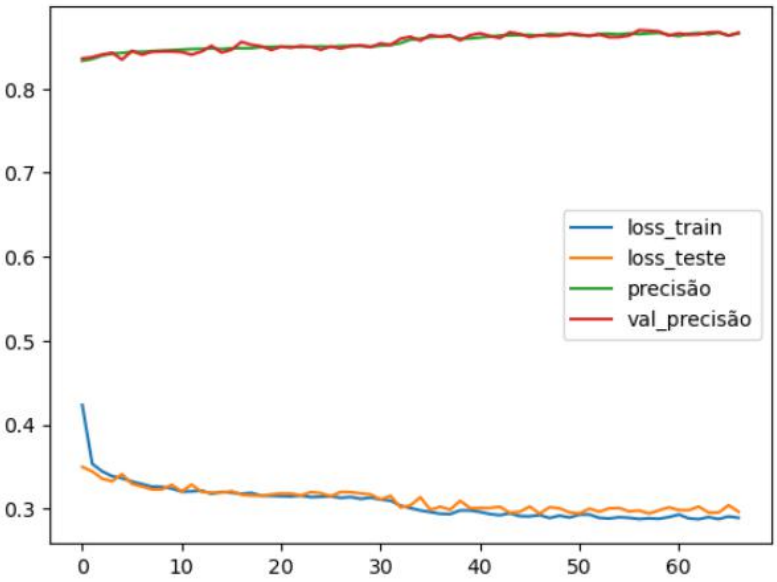

Figura 13: Tabela loss e precisão.

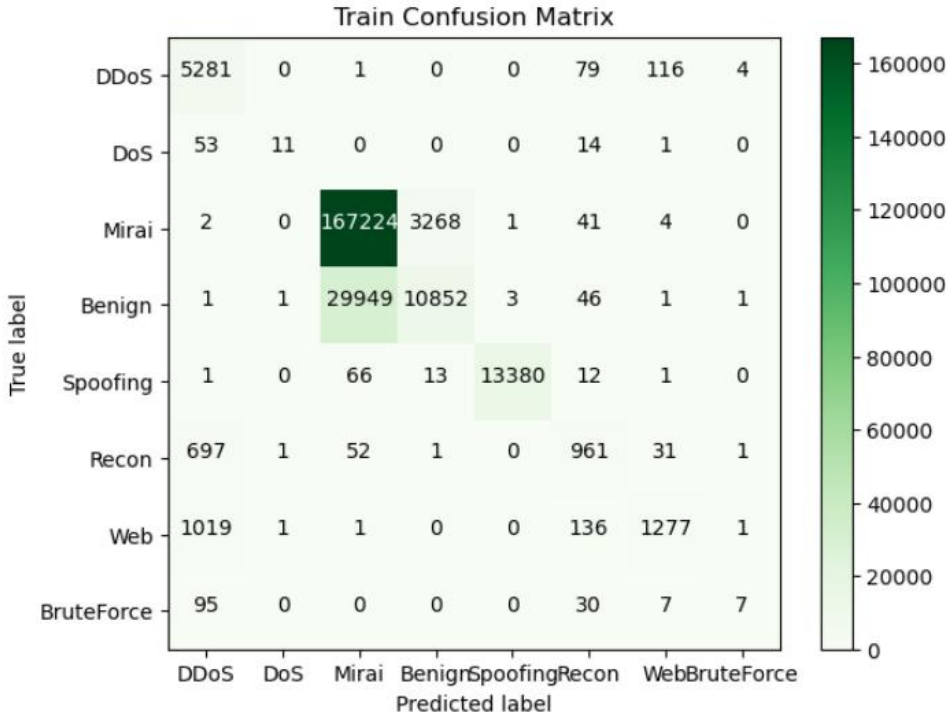

Figura 14: Matriz de confusão.

```
Accuracy: 0.8476985665296386

Micro Precision: 0.8476985665296386

Micro Recall: 0.8476985665296386

Micro F1_score: 0.8476985665296386

Macro Precision: 0.7820159336843588

Macro Recall: 0.5584824958624678

Macro F1_score: 0.5939478841865935

Weighted Precision: 0.8392009607494245

Weighted Recall: 0.8476985665296386

Weighted F1_score: 0.8175943768687111


Classification Report

              precision    recall  f1-score   support

       DDoS       0.74      0.96      0.84      5481
        DoS       0.79      0.14      0.24        79
      Mirai       0.85      0.98      0.91    170540
     Benign       0.77      0.27      0.39     40854
   Spoofing       1.00      0.99      1.00     13473
      Recon       0.73      0.55      0.63      1744
        Web       0.89      0.52      0.66      2435
 BruteForce       0.50      0.05      0.09       139

   accuracy                           0.85    234745
  macro avg       0.78      0.56      0.59    234745
weighted avg       0.84      0.85      0.82    234745
```

Figura 15: Métricas da rede.

Dados de treinamento sampling

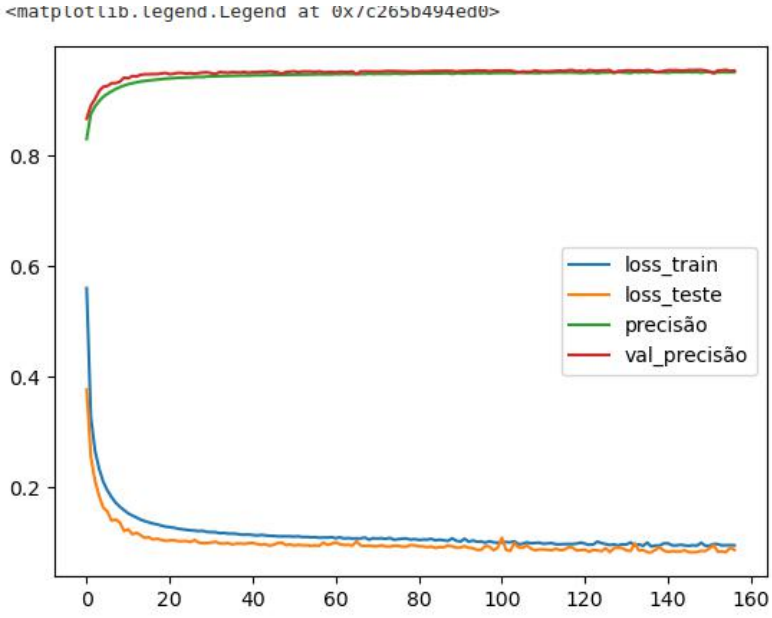

Figura 15: Tabela loss e precisão.

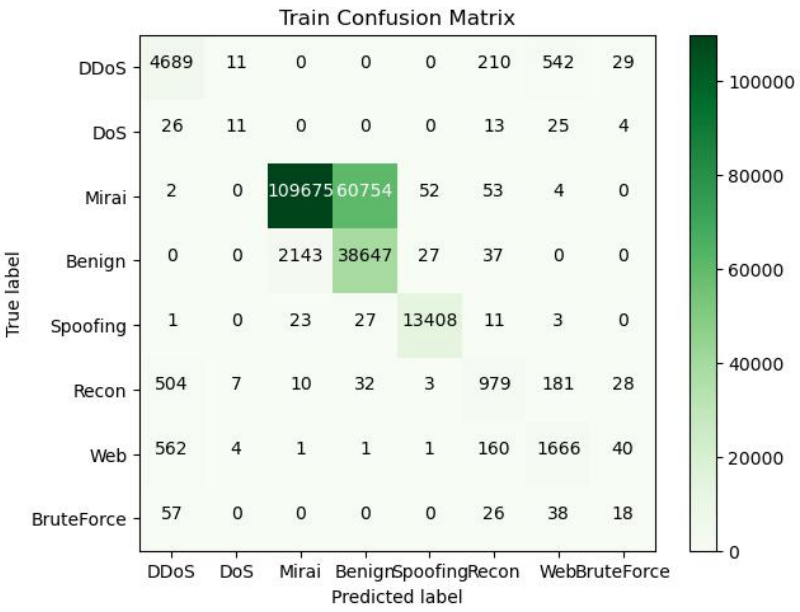

Figura 16: Matriz de confusão.

```
Accuracy: 0.7203263115295321

Micro Precision: 0.7203263115295321

Micro Recall: 0.7203263115295321

Micro F1_score: 0.7203263115295321

Macro Precision: 0.6231644573482702

Macro Recall: 0.619254751029219

Macro F1_score: 0.5966067947841684

Weighted Precision: 0.8678737399701103

Weighted Recall: 0.7203263115295321

Weighted F1_score: 0.7483053326199293

Classification Report

              precision    recall  f1-score   support

       DDoS       0.80      0.86      0.83      5481
        DoS       0.33      0.14      0.20        79
      Mirai       0.98      0.64      0.78    170540
     Benign       0.39      0.95      0.55     40854
   Spoofing       0.99      1.00      0.99     13473
      Recon       0.66      0.56      0.61      1744
        Web       0.68      0.68      0.68      2435
 BruteForce       0.15      0.13      0.14       139

   accuracy                           0.72    234745
  macro avg       0.62      0.62      0.60    234745
weighted avg       0.87      0.72      0.75    234745
```

Figura 17: Métricas da rede.

Dados de treinamento smote

```
<matplotlib.legend.Legend at 0x74e7eddd9790>
```

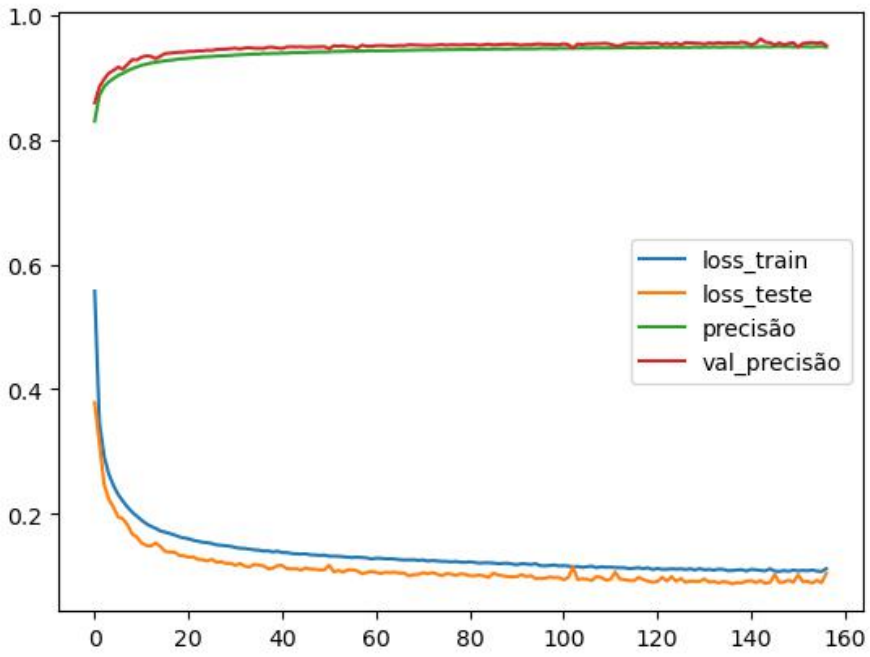

Figura 18: Tabela loss e precisão.

```
Text(0.5, 23.52222222222222, 'Predicted label')
```

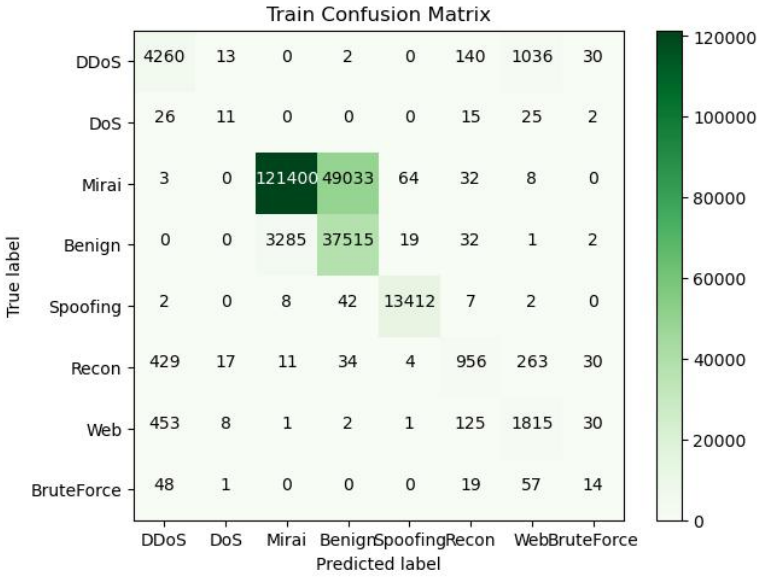

Figura19: Matriz de confusão.

```
Accuracy: 0.7641611109927794

Micro Precision: 0.7641611109927794

Micro Recall: 0.7641611109927794

Micro F1_score: 0.7641611109927795

Macro Precision: 0.6065646999611836

Macro Recall: 0.6170417745937042

Macro F1_score: 0.5939504384191276

Weighted Precision: 0.8700530360246634

Weighted Recall: 0.7641611109927794

Weighted F1_score: 0.7869624659141475


Classification Report

              precision    recall  f1-score   support

        DDoS       0.82      0.78      0.80      5481
         DoS       0.22      0.14      0.17        79
       Mirai       0.97      0.71      0.82    170540
      Benign       0.43      0.92      0.59     40854
    Spoofing       0.99      1.00      0.99     13473
       Recon       0.72      0.55      0.62      1744
         Web       0.57      0.75      0.64      2435
  BruteForce       0.13      0.10      0.11       139

    accuracy                           0.76    234745
   macro avg       0.61      0.62      0.59    234745
weighted avg       0.87      0.76      0.79    234745
```

Figura 20: Métricas da rede.

Modelo 3

O Modelo 3 apresentou uma acurácia de 84% com os dados normais, 79% com os dados de sampling e 78% com os dados de SMOTE. No entanto, algumas classes tiveram um desempenho de classificação significativamente inferior a outras. As figuras abaixo ilustram esses resultados.

Dados normais

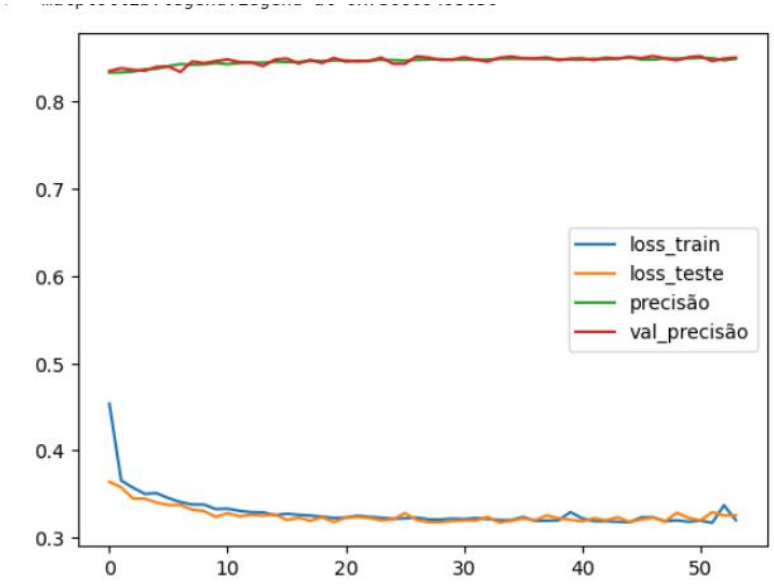

Figura 21: Tabela loss e precisão.

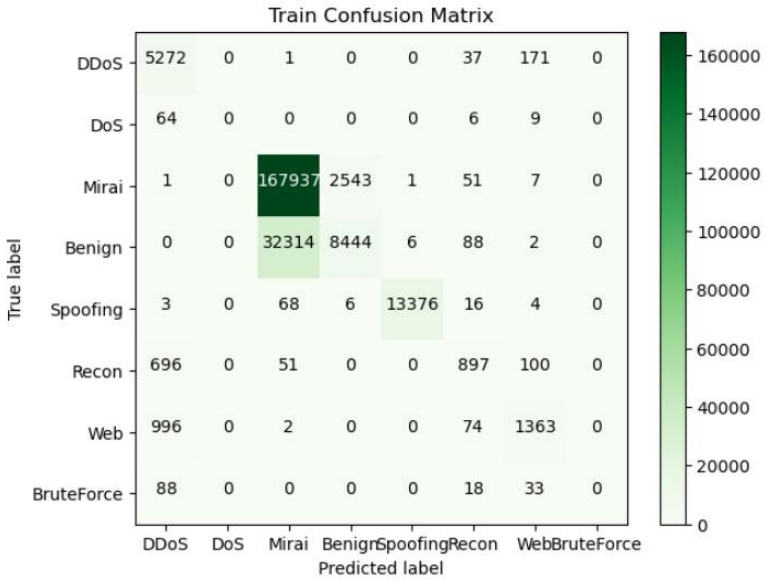

Figura 22: Matriz de confusão.

```
Accuracy: 0.8404396259771241

Micro Precision: 0.8404396259771241

Micro Recall: 0.8404396259771241

Micro F1_score: 0.8404396259771241

Macro Precision: 0.6136058058931655

Macro Recall: 0.5275226362526135

Macro F1_score: 0.5421542157268164

Weighted Precision: 0.831206545710315

Weighted Recall: 0.8404396259771241

Weighted F1_score: 0.802662520543431


Classification Report

              precision    recall  f1-score   support

        DDoS       0.74      0.96      0.84      5481
         DoS       0.00      0.00      0.00        79
       Mirai       0.84      0.98      0.91    170540
      Benign       0.77      0.21      0.33     40854
    Spoofing       1.00      0.99      1.00     13473
       Recon       0.76      0.51      0.61      1744
         Web       0.81      0.56      0.66      2435
  BruteForce       0.00      0.00      0.00       139

    accuracy                           0.84    234745
   macro avg       0.61      0.53      0.54    234745
weighted avg       0.83      0.84      0.80    234745
```

Figura 23: Métricas da rede.

Dados sampling

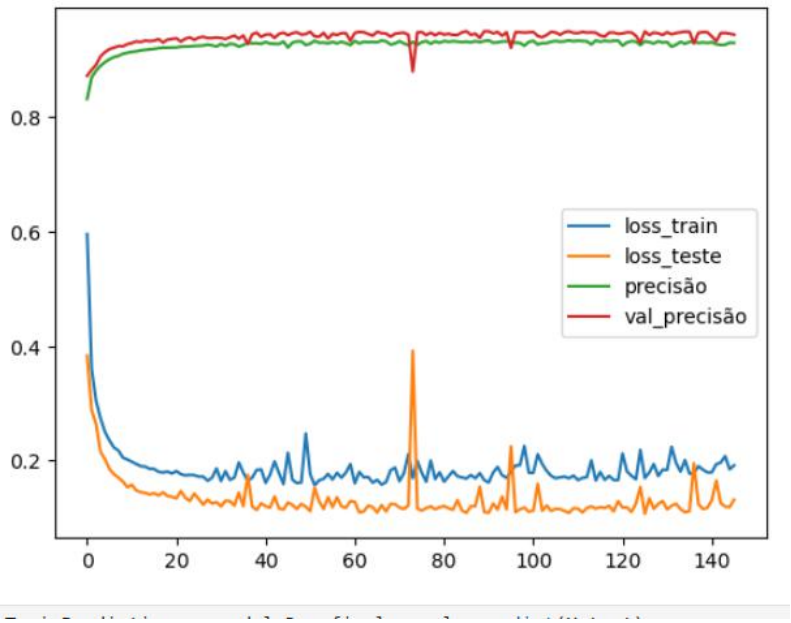

Figura 24: Tabela loss e precisão.

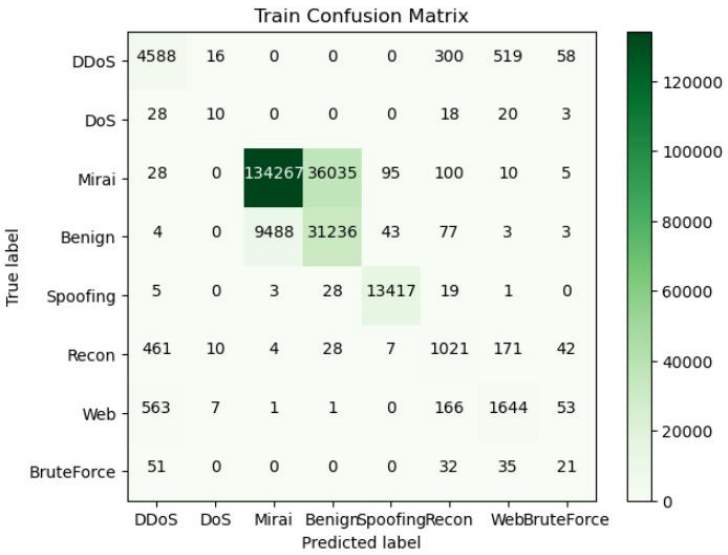

Figura 25: Matriz de confusão.

```
Accuracy: 0.7932181729110311

Micro Precision: 0.7932181729110311

Micro Recall: 0.7932181729110311

Micro F1_score: 0.7932181729110311

Macro Precision: 0.6009423534898843

Macro Recall: 0.6153811989887291

Macro F1_score: 0.6004394222478875

Weighted Precision: 0.8463466348626126

Weighted Recall: 0.7932181729110311

Weighted F1_score: 0.8088252115997443


Classification Report

              precision    recall  f1-score   support

       DDoS       0.80      0.84      0.82      5481
        DoS       0.23      0.13      0.16        79
      Mirai       0.93      0.79      0.85    170540
     Benign       0.46      0.76      0.58     40854
   Spoofing       0.99      1.00      0.99     13473
      Recon       0.59      0.59      0.59      1744
        Web       0.68      0.68      0.68      2435
 BruteForce       0.11      0.15      0.13       139

   accuracy                           0.79    234745
  macro avg       0.60      0.62      0.60    234745
weighted avg       0.85      0.79      0.81    234745
```

Figura 26: Métricas de rede.

Dados smote

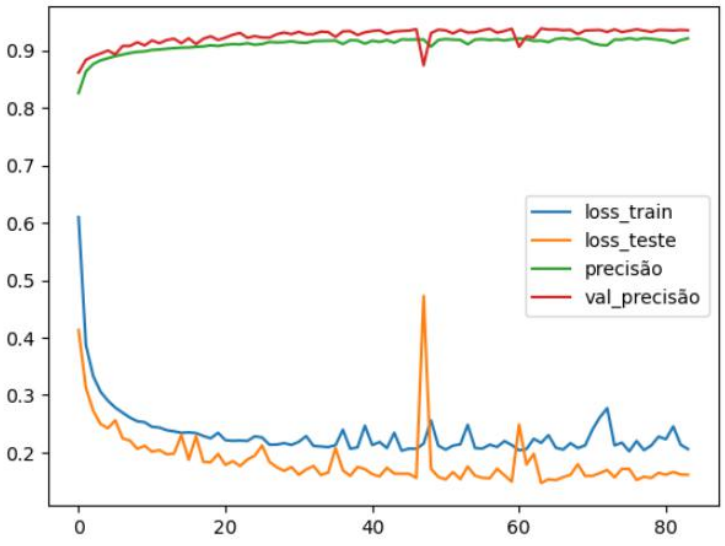

Figura 27: Tabela loss e precisão.

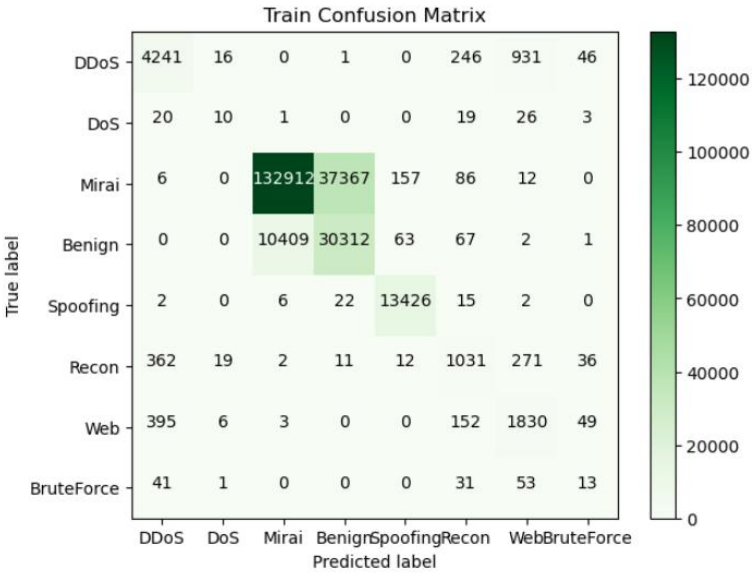

Figura 28: Matriz de confusão.

```
Accuracy: 0.782870774670387

Micro Precision: 0.782870774670387

Micro Recall: 0.782870774670387

Micro F1_score: 0.782870774670387

Macro Precision: 0.5857881130854843

Macro Recall: 0.6068014412712484

Macro F1_score: 0.5885685675595468

Weighted Precision: 0.8383785801108642

Weighted Recall: 0.782870774670387

Weighted F1_score: 0.7994859718261382

Classification Report

              precision    recall  f1-score   support

       DDoS       0.84      0.77      0.80      5481
        DoS       0.19      0.13      0.15        79
      Mirai       0.93      0.78      0.85    170540
     Benign       0.45      0.74      0.56     40854
   Spoofing       0.98      1.00      0.99     13473
      Recon       0.63      0.59      0.61      1744
        Web       0.59      0.75      0.66      2435
  BruteForce       0.09      0.09      0.09       139

   accuracy                           0.78    234745
  macro avg       0.59      0.61      0.59    234745
weighted avg       0.84      0.78      0.80    234745
```

Figura 29: Métricas de rede.

Após vários testes em três redes diferentes, verificamos que o desempenho das redes com dados normais foi superior ao desempenho com dados gerados por SMOTE e Samply. Portanto, optamos por utilizar a Rede 1 para realizar mais experimentos com o objetivo de melhorar seu desempenho nos dados de teste Figura 1.

Devido ao baixo poder de processamento de nossa máquina, foi mais viável escolher uma rede menor, permitindo testar diversos parâmetros. Após muitos testes, encontramos os melhores parâmetros ajustando a taxa de aprendizado (lr) para 0.001, o tamanho do lote (batch_size) para 200 e o número de épocas (epochs) para 200 Figuras 30 e 31. Com esses ajustes, alcançamos uma acurácia de 87% nos dados de teste, superando os resultados anteriores. As Figuras 32, 33 e 34 abaixo ilustram esses resultados:

```python
def modeloDense(X_train, metricas):
    model = tf.keras.models.Sequential([
        tf.keras.layers.Dense(500, activation="relu", input_shape=(X_train.shape[-1],)),
        tf.keras.layers.Dense(units=500, activation='relu', kernel_initializer="random_uniform", bias_initializer="random_uniform"),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(units=500, activation='relu', kernel_initializer="random_uniform", bias_initializer="random_uniform"),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(8,activation='softmax')

    ])
    model.compile(loss='categorical_crossentropy',optimizer=Adam(0.001), metrics=metricas)
    return model
```

Figura 30: Modificação da rede 1.

```python
model_nor = modeloDense(X_train,metricas)
model_normal = model_nor.fit(X_train, y_train, validation_data=(X_validation, y_validation),
                  batch_size=200, epochs=200, verbose=1, callbacks=[Mycallback(), checkpoint_callback_normal_melhoria_final,
                                                                    callback_stop_patience])
```

Figura 31: Modificação dos parametros de treinamento.

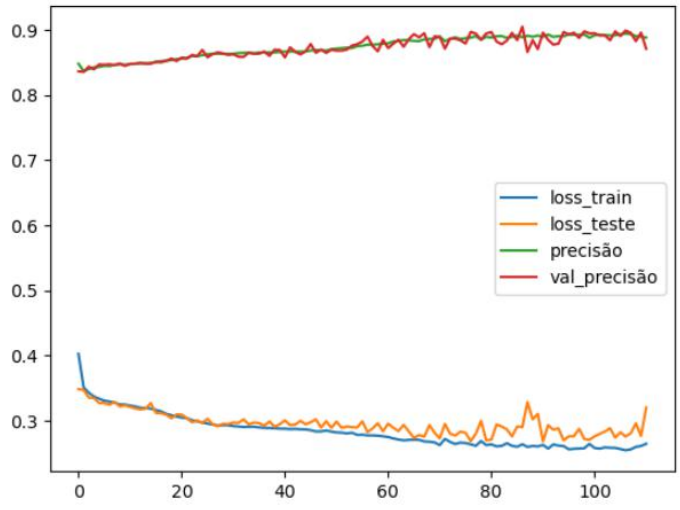

Figura 32: Tabela loss e precisão.

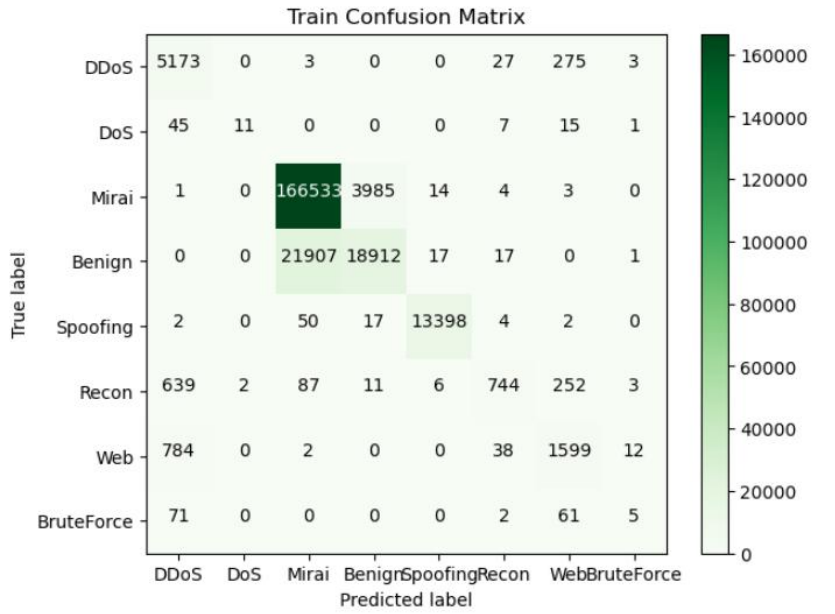

Figura 33: Matriz de confusão.

```
Accuracy: 0.8791454557072568

Micro Precision: 0.8791454557072568

Micro Recall: 0.8791454557072568

Micro F1_score: 0.8791454557072568

Macro Precision: 0.766108857204681

Macro Recall: 0.5795188381212784

Macro F1_score: 0.6161073212350453

Weighted Precision: 0.8748186497032512

Weighted Recall: 0.8791454557072568

Weighted F1_score: 0.8654899220390474

Classification Report

              precision    recall  f1-score   support

       DDoS       0.77      0.94      0.85      5481
        DoS       0.85      0.14      0.24        79
      Mirai       0.88      0.98      0.93    170540
     Benign       0.82      0.46      0.59     40854
   Spoofing       1.00      0.99      1.00     13473
      Recon       0.88      0.43      0.58      1744
        Web       0.72      0.66      0.69      2435
 BruteForce       0.20      0.04      0.06       139

   accuracy                           0.88    234745
  macro avg       0.77      0.58      0.62    234745
weighted avg       0.87      0.88      0.87    234745
```

Figura 34: Métricas da rede.

Conclusões

Com base nos resultados obtidos, podemos concluir que os modelos treinados com a base de dados normais, sem a aplicação de técnicas de sampling ou SMOTE, são capazes de atingir uma acurácia de 84% nos dados de teste. Após realizar melhorias na Rede 1, conseguimos aumentar essa acurácia para 87%.

O modelo treinado pode fazer classificações com boa precisão. No entanto, ainda há espaço para melhorias, especialmente em relação a certas classes que foram mais difíceis de prever, conforme mostrado nas figuras. Passos futuros podem incluir a experimentação com diferentes arquiteturas de rede, o ajuste de hiperparâmetros e o aumento da quantidade de dados para aprimorar ainda mais o desempenho do modelo.