Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

# EFC: um estudo de sensibilidade de Hiperparâmetros

Álvaro V. C. Luz

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. João J. C. Gondim

Brasília
2025

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

# EFC: um estudo de sensibilidade de Hiperparâmetros

Álvaro V. C. Luz

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. João J. C. Gondim (Orientador)
ENE/UnB

Prof. Dr. Marcelo Marotta
CIC/UnB

Prof. Dr. Luís Paulo Faina Garcia
CIC/UnB

Prof. Dr. Marcelo G. Mandelli
Coordenador do Bacharelado em Ciência da Computação

Brasília, 16 de julho de 2025

# Dedicatória

Dedico este trabalho à minha família — ao meu pai Glauco, à minha mãe Ana Ester e à minha irmã Luiza — por todo o amor, apoio e incentivo incondicional ao longo da minha trajetória acadêmica.

Agradeço profundamente por estarem sempre ao meu lado, acreditando em mim, especialmente nos momentos mais desafiadores.

À minha namorada Manuela, minha companheira e fonte constante de inspiração, dedico também este trabalho, com imenso carinho e gratidão, por todo o apoio, paciência e encorajamento que tornaram possível a realização deste sonho.

# Agradecimentos

Agradeço, antes de tudo, à minha família e à minha namorada Manuela, pelo apoio incondicional, pelos conselhos e pelo encorajamento constante durante todo o processo de elaboração deste trabalho.

Sou imensamente grato aos meus amigos Ricardo e Átila, que mergulharam comigo nas discussões teóricas e contribuíram diretamente para o aprofundamento do arcabouço conceitual deste estudo.

Estendo meus agradecimentos a João Victor, Maria Eduarda e Paulo, cujas contribuições foram fundamentais para meu progresso na escrita e para a qualidade final deste trabalho.

Agradeço também ao Dr. Murylo e ao Dr. André, profissionais que desempenharam um papel essencial no manejo do meu TDAH, oferecendo suporte, motivação e equilíbrio ao longo da minha trajetória acadêmica.

Por fim, deixo meu sincero agradecimento ao meu orientador, professor João Gondim, por sua atenção constante, pelas orientações precisas e por respeitar meu tempo, sempre me incentivando a seguir em frente com confiança.

# Resumo

O *Energy-Based Flow Classifier* (EFC) é um modelo de classificação de fluxos de rede fundamentado em estatística inversa, inspirado no modelo de Potts, e originalmente proposto para aplicações em sistemas de detecção de intrusão. Apesar de sua aplicabilidade comprovada em diferentes contextos, a literatura carece de estudos que investiguem a sensibilidade do modelo à variação de seus hiperparâmetros. Este trabalho propõe uma análise sistemática do impacto dos principais hiperparâmetros do EFC — o limiar quantílico de classificação, o número de níveis de discretização e os pesos de pseudocontagens — sobre seu desempenho em tarefas de classificação binária e multiclasse. Utilizando os conjuntos de dados CICIDS2017 e CICDDoS2019, foram conduzidos experimentos com múltiplas combinações de valores para esses parâmetros. Os resultados obtidos são discutidos com base nas métricas AUC-ROC e F1-Score, permitindo propor diretrizes para uma calibragem mais eficiente do modelo, otimizando sua precisão e robustez.

**Palavras-chave:** Energy-Based Flow Classifier, hiperparâmetros, detecção de intrusão, estatística inversa, classificação de fluxos.

**Abstract**

The *Energy-Based Flow Classifier* (EFC) is a network flow classification model based on inverse statistical mechanics, inspired by the Potts model, and originally proposed for use in intrusion detection systems. Although its effectiveness has been demonstrated across various domains, few studies have addressed the sensitivity of the model to changes in its hyperparameters. This work presents a systematic analysis of the impact of key EFC hyperparameters — quantilic classification threshold, discretization levels, and pseudo-count weights — on its performance in both binary and multiclass classification tasks. Experiments were conducted using the CICIDS2017 and CICDDoS2019 datasets, testing multiple parameter configurations. The results, evaluated through AUC-ROC and F1-Score metrics, support the proposal of guidelines for a more effective hyperparameter tuning strategy, enhancing both accuracy and robustness of the model.

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Capítulo 1

# Introdução

Neste capítulo são apresentadas a contextualização e a justificativa para o estudo de calibragem de hiperparâmetros do *Energy-based Flow Classifier*, além dos objetivos gerais e das contribuições deste projeto de pesquisa.

## 1.1 Contextualização

As ameaças cibernéticas têm-se tornado cada vez mais sofisticadas e frequentes, acompanhando o crescimento exponencial da conectividade digital. Segundo o Relatório de Ameaças Cibernéticas de 2025 da *Radware*, houve um aumento de 550% em ataques de negação de serviço distribuídos (DDoS, do inglês, *Distributed Denial of Service*) ano após ano quando comparados aos números de 2023 [1], reportando também casos recentes em que houve um pico de 16 milhões de requisições por segundo. Esse cenário é agravado pela transformação digital de dados pessoais e corporativos, aliada à proliferação de dispositivos de Internet das Coisas (do inglês, *Internet of Things*) os quais, devido à sua baixa robustez em termos de segurança, são frequentemente explorados como vetores em ataques cibernéticos [2].

Neste contexto, sistemas de detecção de intrusão em redes (NIDS, Network Intrusion Detection Systems) são ferramentas úteis para lidar com as ameaças cibernéticas. NIDS são *softwares* utilizados em conjunção com *firewalls* e antivírus para proteger dispositivos conectados à rede de diversas ameaças [3]. Suas aplicações podem variar, podendo executar diversas funções em uma rede, tais como detectar ou classificar anomalias [4]. Assim, um sistema deste tipo, capaz de discernir entre diferentes tipos maliciosos de tráfego de rede, pode ser associado a respostas automatizadas de mitigação, garantindo maior resiliência à rede [5].

Uma abordagem comum para a elaboração de ferramentas dedicadas à detecção de intrusão é a análise de fluxos de rede, isto é, de conjuntos de pacotes trafegados pela

rede, que podem ser agrupados levando em consideração as suas características em comum (como endereço *IP* de origem e destino, número de pacotes, número de *bytes* e porta de origem, por exemplo) [6]. Estes pacotes, então, agrupados em fluxos, podem ser analisados como um todo, reduzindo drasticamente o volume de dados a ser analisado, mostrando-se como uma abordagem adequada para a elaboração de NIDS que operem em tempo real [7].

Com isso em mente, foi elaborado em um estudo recente de Pontes *et al.*[8], o *Energy-based Flow Classifier* (do inglês, Classificador de Fluxos baseado em Energia, EFC), um classificador de fluxos de redes direcionado para a aplicação em NIDS. O modelo se baseia na utilização de técnicas de estatística inversa utilizadas na biofísica para a análise direta de acoplamentos de proteínas, adaptando-as para o contexto de classificação de fluxos de pacotes em redes computacionais. O modelo inicialmente foi elaborado para ser um classificador binário, sendo posteriormente adaptado por De Souza *et al.*[4] para ser capaz de tratar de múltiplas classes.

O EFC realiza a inferência de um modelo estatístico para cada classe presente no conjunto de dados de fluxos de rede utilizados em seu treinamento [4]. Os modelos estatísticos inferidos, então, podem ser usados para identificar se novos fluxos pertencem ou não à classe utilizada para a inferência. Este modelo, portanto, se caracteriza como um classificador baseado em anomalias, isto é, um classificador capaz de identificar quando uma nova amostra a ele apresentada não pertence a nenhuma das classes de amostras utilizadas em seu treinamento [8]. Esta característica intrínseca confere ao modelo uma capacidade de melhor adaptação a diferentes domínios de treino e teste, tornando-o uma abordagem simples, que não requer grandes transformações nos dados para operar, o que o torna um algoritmo promissor para classificação baseada em fluxos de rede [8].

## 1.2 Justificativa e objetivos

Entre os trabalhos vistos na literatura, nota-se que o foco central é confirmar o potencial de aplicação do *Energy-based Flow classifier* em diferentes cenários para o monitoramento de tráfego de rede. O modelo já teve sua capacidade de detecção testada em trabalhos relacionados a: detecção de *botnets* [9]; detecção de anomalias em redes móveis [10]; aprendizado federado [11] [12] [13]; entre outros. É válido ressaltar que, nestes trabalhos, o algoritmo em questão teve seu desempenho comparado com modelos tradicionais de aprendizado de máquina (tais como *Random Forest*, *Multilayer Perceptron*, *Support Vector Machine*, etc.), modelos de redes neurais e de aprendizado federado e, em geral, apresentou um desempenho equiparável ou superior a esses modelos, se destacando especialmente em classificações com diferentes domínios de treino e teste. Contudo, apesar dos resultados promissores obtidos, os estudos do modelo de classificação se limitaram principalmente a

novos cenários de aplicação. Portanto, ainda não há trabalhos direcionados a avaliar quesitos mais intrínsecos do comportamento do modelo, tal como a influência da configuração dos hiperparâmetros do EFC no seu desempenho.

Hiperparâmetros, no contexto de *machine learning* (do inglês, aprendizado de máquina, ML) são valores definidos como parâmetros externos ao processo de aprendizagem que, comumente, devem ser cuidadosamente otimizados para que um modelo de classificação consiga alcançar um desempenho otimizado [14]. Nos trabalhos mencionados, comumente, a configuração dos hiperparâmetros do EFC é um tópico não abordado, raramente sendo explicitados quais valores foram atribuídos a cada parâmetro. Em geral, nos casos em que há essa informação, são comumente utilizados os valores padrão, ou seja, os valores genéricos atribuídos por quem implementou o modelo em código. Isso pode ser problemático, dado que nem sempre esta configuração é a mais adequada ao tipo de conjunto de dados que se está utilizando.

O trabalho a seguir busca, portanto, realizar uma análise sobre como a definição de diferentes valores destes parâmetros de configuração para o EFC afeta os resultados da classificação. Para isso, serão treinadas diversas instâncias do modelo com diferentes configurações de hiperparâmetros, as quais serão utilizadas para um conjunto de classificações binárias e multiclasse com os mesmos dados de treino e teste. Posteriormente, serão analisados os resultados obtidos para essas classificações, com o objetivo de traçar uma estratégia para a otimização dos hiperparâmetros do EFC.

## 1.3 Objetivos e estrutura do trabalho

Desta forma, os principais objetivos deste trabalho se resumem a:

- **Avaliação do impacto dos hiperparâmetros no modelo**: Analisar como os hiperparâmetros do modelo (limiar de classificação, níveis de discretização e pesos de pseudocontagens) influenciam o comportamento do EFC, por meio da experimentação com variadas combinações de hiperparâmetros em datasets consolidados (CICIDS2017 e CICDDoS19).
- **Estratégia para otimização de hiperparâmetros**: Proposição de um método para a seleção de valores para os hiperperâmetros baseando-se na análise das configurações que obtiveram os melhores resultados nos experimentos em termos de AUC-ROC e F1-Score.

O restante do trabalho a seguir está organizado da seguinte forma: o Capítulo 2 (O *Energy-based Flow Classifier*) detalha o funcionamento do EFC em todas as etapas da classificação, especificando em que pontos do algoritmo cada hiperparâmetro é utilizado.

No Capítulo 3 (Trabalhos Relacionados), são discutidos os trabalhos já existentes na literatura que utilizaram o EFC para a classificação de fluxos de rede e detecção de anomalias, buscando quais valores de hiperparâmetros já foram utilizados para o modelo, bem como as principais contribuições dos trabalhos. No Capítulo 4 (Metodologia), são descritos os procedimentos experimentais, como a metodologia de ajuste de hiperparâmetros (limiar de classificação, níveis de discretização e pesos de pseudocontagens), as métricas de avaliação (AUC-ROC e F1-Score), os *datasets* CICIDS2017 e CICDDoS2019, técnicas de balanceamento de dados para cenários binários e multiclasse, além dos recursos computacionais utilizados. Já o Capítulo 5 (Resultados e Discussão) apresenta análises comparativas dos experimentos, explorando o impacto dos hiperparâmetros no desempenho do modelo e discutindo suas implicações práticas. Por fim, o Capítulo 6 (Conclusão) sintetiza as contribuições do estudo, suas limitações e perspectivas futuras. Essa estrutura visa guiar o leitor de forma lógica desde os fundamentos teóricos até a validação empírica do modelo proposto.

# Capítulo 2

# O Energy-based Flow Classifier

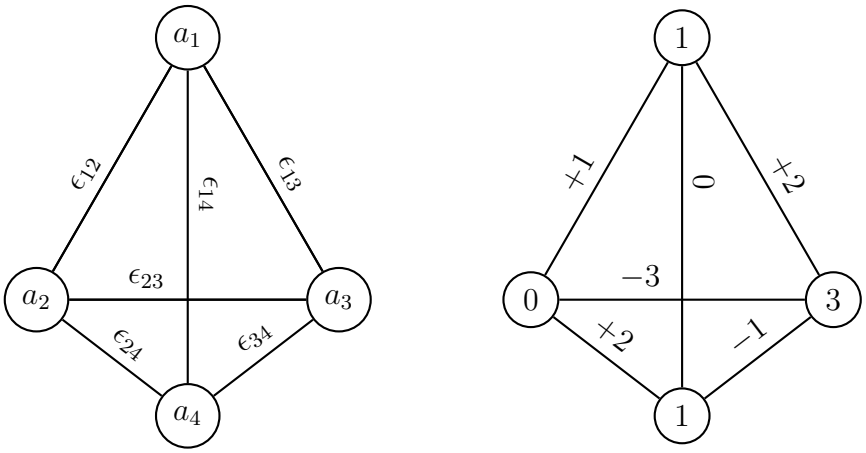
O Classificador de Fluxos baseado em Energia (EFC, do inglês Energy-based Flow Classifier), proposto inicialmente em Pontes *et al.* [8], é um classificador baseado em estatística inversa, cujo objetivo principal é inferir uma distribuição estatística que descreva o comportamento de uma classe específica de dado. Posteriormente, o modelo foi estendido em Souza *et al.* [4] de modo a conferir ao modelo a capacidade de classificação multiclasse.

O restante deste capítulo condensa o arcabouço teórico apresentado nesses estudos, fundamentando-se neles e em suas respectivas referências. O capítulo está estruturado da seguinte forma: a Seção 2.1 apresenta como se baseia a abstração de fluxos de rede para que seja possível inferir seu comportamento por meio da inversa do modelo de Potts; a Seção 2.2 detalha o processo de inferência do modelo estatístico; na Seção 2.3, discute-se como o modelo realiza a classificação de novos fluxos com base nas energias calculadas; e, por fim, a Seção 2.4 formaliza a definição algorítmica do EFC, incluindo sua representação em pseudocódigo e a identificação dos pontos em que os hiperparâmetros atuam no processo de inferência.

## 2.1 Abstração de fluxos de rede para o modelo de Potts

Como mencionado anteriormente, no modelo proposto em Pontes *et al.* [8], a inferência estatística do EFC é fundamentada no modelo de Potts, o qual descreve matematicamente as interações entre *spins* de elétrons em uma rede cristalina. Para tornar esse modelo compatível com a análise de fluxos de rede, é introduzida uma abstração que representa os fluxos de forma adequada ao arcabouço estatístico adotado. Nesta, um fluxo individual $k$ é representado por uma configuração específica de um grafo $G_k(\eta, \epsilon)$. Neste, cada nó do grafo representa um atributo de um fluxo, sendo $\eta$ o conjunto de todos os nós/atributos

Figura 2.1: Imagem ilustrativa do grafo $G_k(\eta, \epsilon)$. A imagem à esquerda exibe um fluxo com 4 atributos completamente conectados. Na imagem à direita são exibidos os valores associados aos atributos e acoplamentos.




do fluxo. Em cada fluxo $k$, cada atributo $i \in \eta$ assume um valor $a_{ki}$, o qual estará contido em $\Omega_i$, um conjunto que contém todos os valores possíveis para aquele atributo específico.

É importante destacar que, para a realização da inferência no modelo, os valores dos atributos $a_{ki}$ devem pertencer a uma grandeza discreta. Assim, é necessário que esses valores sejam mapeados para inteiros pertencentes ao conjunto $\Omega = \{1, 2, \ldots, Q\}$, onde $Q$ representa o número total de categorias discretas. Dessa forma, assume-se que os alfabetos de todos os atributos são iguais, ou seja, $\Omega_i = \Omega$, mesmo nos casos em que um atributo específico possa assumir apenas $M$ valores distintos, com $M < Q$. Nesses casos, os valores $M + 1, \ldots, Q$ simplesmente não ocorrem na prática, tendo probabilidade nula de ocorrência.

Ademais, o grau de granularidade adotado na discretização dos atributos influencia diretamente a capacidade do modelo de identificar padrões nos dados de entrada. Como o modelo não impõe restrições quanto à definição de $Q$, esse valor é tratado como um hiperparâmetro que deve ser estabelecido previamente à etapa de treinamento.

Prosseguindo, as arestas do grafo $G_k$ são representadas por $\epsilon = \{(i, j) \mid i, j \in \eta;\ i \neq j\}$, correspondendo a todos os pares possíveis de atributos distintos. Para cada par $(i, j)$, existe um valor de acoplamento associado, determinado pela função $e_{ij}(a_{ki}, a_{kj})$. Além disso, cada atributo $i$ possui um valor associado ao seu campo local, representado por $h_i(a_{ki})$.

Dessa forma, a energia total associada a um fluxo $k$, análoga ao conceito de Hamiltoniano na mecânica estatística, pode ser expressa pelo Hamiltoniano $\mathcal{H}(a_{k1}, \ldots, a_{kN})$, o qual depende de todos os valores dos campos locais e dos acoplamentos definidos sobre os pares de atributos.

## 2.2 Inferência do modelo estatístico

Uma vez definida a abstração para os fluxos de rede, propõe-se a inferência de um modelo estatístico $P(a_1, \ldots, a_N)$ para cada sequência de atributos $(a_1, \ldots, a_N)$ associada a um fluxo $k \in \mathcal{B}$, sendo $\mathcal{B}$ um conjunto arbitrário de fluxos observados. Supõe-se então o conjunto $\mathcal{K}$, que representa o espaço de todas as combinações possíveis de atributos par-a-par, com $\mathcal{K} = \Omega^2$, de forma que $\mathcal{B} \subseteq \mathcal{K}$.

A partir disso, propõe-se o uso de estatística inversa para inferir um modelo que atribua uma probabilidade $P(a_{k1}, \ldots, a_{kN})$ a cada fluxo $k \in \mathcal{K}$, com base nas observações empíricas contidas em $\mathcal{B}$. O objetivo, nesse contexto, é encontrar uma distribuição de Máxima Entropia

$$S(a_{k1}, \ldots, a_{kN}) = -\sum_{k \in \mathcal{K}} P(a_{k1}, \ldots, a_{kN}) log(P(a_{k1}, \ldots, a_{kN})), \tag{2.1}$$

a qual deve ser compatível com as estatísticas observadas nos dados. Para isso, a distribuição está sujeita a restrições impostas pelas frequências empíricas de ocorrência individual de cada valor dos atributos, bem como pelas frequências conjuntas de ocorrência dos pares. Essas restrições são expressas da seguinte forma:

$$\begin{gathered} \boldsymbol{\forall i \in \eta; \forall a_i \in \Omega :} \\ P_i(a_i) = \sum_{k \in \mathcal{K} | a_{ki} = a_i} P(a_{k1}, \ldots, a_{kN}) \equiv f_i(a_i) \end{gathered} \tag{2.2}$$

e

$$\begin{gathered} \boldsymbol{\forall (i,j) \in \eta^2; \forall (a_i, a_j) \in \Omega :} \\ P_{ij}(a_{ij}) = \sum_{k \in \mathcal{K} | a_{ki} = a_i, a_{kj} = a_j} P(a_{k1}, \ldots, a_{kN}) \equiv f_{ij}(a_i, a_j). \end{gathered} \tag{2.3}$$

Em que $f_i(a_i)$ representa a frequência empírica do valor $a_i$ no atributo $i$, e $f_{ij}(a_i, a_j)$ corresponde à frequência empírica do par de valores $(a_i, a_j)$ nos atributos $i$ e $j$, respectivamente.

Ambas as frequências empíricas simples e duplas são obtidas a partir do conjunto $\mathcal{B}$, realizando a contagem das ocorrências de um dado valor de atributo $a_i$ ou de pares de atributos $(a_i, a_j)$, respectivamente, e dividindo pelo número total de fluxos em $\mathcal{B}$. Contudo, dada a diferença de tamanho entre os conjuntos $\mathcal{B}$ e $\mathcal{K}$, inferências baseadas em $\mathcal{B}$ estão sujeitas a efeitos de subamostragem (*undersampling*). Então, para que sejam limitados os efeitos de subamostragem, utiliza-se a inclusão de um fator de pseudocontagem $\alpha$ no cálculo das frequências, resultando nas correções

$$f_i(a_i) \longleftarrow (1 - \alpha) f_i(a_i) + \frac{\alpha}{Q} \tag{2.4}$$

$$f_{ij}(a_i, a_j) \longleftarrow (1-\alpha) f_{ij}(a_i, a_j) + \frac{\alpha}{Q^2}, \tag{2.5}$$

com $(i, j) \in \eta^2$ $(a_i, a_j) \in \Omega^2$ e $0 \leq \alpha \leq 1$. A inclusão das pseudocontagens é equivalente a assumir que o conjunto $\mathcal{B}$ seja estendido, de forma a conter uma fração extra de fluxos com valores de atributos uniformemente distribuídos. Este valor é capaz, portanto, de "diluir"o comportamento dos atributos observados em $\mathcal{B}$ para que seja possível inferir o modelo estatístico. Assim, o valor de $\alpha$ foi implementado como um valor configurável, sendo um dos hiperparâmetros do EFC, uma vez que impacta diretamente na capacidade de inferência do modelo, dado que valores baixos demais podem inviabilizar a inferência do modelo, enquanto valores altos demais podem "diluir"o comportamento dos dados, implicando em inferências mais imprecisas.

Em sequência, uma vez definidas as restrições, busca-se a distribuição que maximiza a entropia sob tais condições. Aplicando-se o Princípio da Máxima Entropia, obtém-se a distribuição que impõe o menor número de suposições adicionais além das informações fornecidas pelos dados. Como resultado, a distribuição $P^*(a_{k1}, \ldots, a_{kN})$ assume a forma de uma distribuição de Boltzmann (ainda que não necessariamente independente e identicamente distribuída)

$$P^*(a_{k1}, \ldots, a_{kN}) = \frac{\exp\{-\beta \mathcal{H}(a_{k1}, \ldots, a_{kN})\}}{Z}. \tag{2.6}$$

com temperatura inversa $\beta = 1$ sem perda de generalidade. Aqui, $Z$ representa a função de partição que normaliza a distribuição, porém, dado que o objetivo aqui não é calcular a probabilidade para fluxos específicos, esta função será desconsiderada. Prosseguindo, o valor de energia da distribuição de um fluxo é determinado pelo Hamiltoniano $\mathcal{H}$ calculado utilizando a forma generalizada do Modelo de Potts

$$\mathcal{H}(a_{k1}, \ldots, a_{kN}) = -\sum_{i,j|i<j} e_{ij}(a_{ki}, a_{kj}) - \sum_i h_i(a_{ki}). \tag{2.7}$$

O Hamiltoniano aqui é completamente determinado pelos multiplicadores de Lagrange $h_i$ e $e_{ij}$, associados às restrições 2.2 e 2.3, respectivamente. No contexto do modelo de Potts, o multiplicador $\{e_{ij}(a_i, a_j)|(a_i, a_j) \in \Omega^2\}$ corresponde ao conjunto de todos os valores possíveis de acoplamentos entre dois atributos $i$ e $j$, enquanto $\{h_i(a_i)|a_i \in \Omega\}$ corresponde ao conjunto de todos os possíveis campos locais associados a um atributo $i$.

Em seguida, os parâmetros do modelo foram ajustados de modo que as restrições 2.2 e 2.3 sejam satisfeitas. Nesse procedimento de ajuste, deve-se considerar que a Eq. 2.6 contém mais parâmetros livres do que há condições independentes nas restrições, o que permite modificar acoplamentos e campos locais conjuntamente sem alterar a soma no expoente. Portanto, múltiplas soluções equivalentes para o ajuste são possíveis. Para

eliminar essa liberdade, consideraram-se todos os acoplamentos e campos locais medidos em relação ao último fluxo, definindo então, sem perda de generalidade que:

$$\begin{gathered}\boldsymbol{\forall (i,j) \in \eta^2; \forall a \in \Omega :} \\ e_{ij}(a,Q) = e_{ij}(Q,a) = h_i(Q) = 0,\end{gathered} \tag{2.8}$$

de forma que não há necessidade de calcular $e_{ij}(a_i, a_j)$ caso $a_i$ ou $a_j$ sejam iguais a $Q$.

A inferência dos acoplamentos par-a-par é realizada por aproximação Gaussiana, a partir da inversão da matriz de correlações dos atributos; assim, os valores resultantes são normalizados. A inferência dos acoplamentos par-a-par é definida da seguinte forma:

$$\begin{gathered}\boldsymbol{\forall (i,j) \in \eta^2, \forall (a_i,a_j) \in \Omega^2, a_i, a_j \neq Q :} \\ e_{ij}(a_i,a_j) = -(C^{-1})_{ij}(a_i,a_j),\end{gathered} \tag{2.9}$$

onde

$$C_{ij}(a_i,a_j) = f_{ij}(a_i,a_j) - f_i(a_i)f_j(a_j) \tag{2.10}$$

é a matriz de correlações obtida a partir das frequências empíricas simples e conjuntas. A inversão da matriz de correlações é realizada como forma de remover os efeitos de correlações indiretas nos dados.

Já a inferência dos campos locais $h_i(a_i)$ é realizada utilizando uma aproximação de campo médio. Neste método, a interação de um atributo com seus vizinhos é substituída pela interação aproximada com a média dos atributos, resultando em um valor aproximado para o campo local associado a ele. Esse cálculo é realizado da seguinte forma:

$$\begin{gathered}\boldsymbol{\forall i \in \eta; a_i \in \Omega; a_i \neq Q :} \\ \frac{f_i(a_i)}{f_i(Q)} = \exp\left(h_i(a_i) + \sum_{j,a_j} e_{ij}(a_i,a_j) f_j(a_j)\right),\end{gathered} \tag{2.11}$$

onde $f_i(Q)$ é a frequência do último elemento $a_i = Q$ para qualquer atributo $i$, utilizada aqui para a normalização. É relevante ressaltar que o elemento $Q$ foi escolhido arbitrariamente, podendo ser substituído por qualquer outro valor $a_i \in \Omega$, contanto que o elemento seja mantido o mesmo para o cálculo de campos locais para todos os atributos $i \in \eta$. Assim, os campos locais podem ser calculados a partir dos valores já conhecidos de frequências simples empíricas $f_i(a_i)$ e de acoplamentos $e_{ij}(a_i, a_j)$ da seguinte forma:

$$h_i(a_i) = \ln\left(\frac{f_i(a_i)}{f_i(Q)}\right) - \sum_{j,a_j} e_{ij}(a_i,a_j) f_j(a_j) \tag{2.12}$$

Assim, com a introdução do arcabouço teórico utilizado para a elaboração do EFC, segue, então, a explicação de como este arcabouço é utilizado para a classificação de fluxos de rede em benignos ou maliciosos.

## 2.3 Classificação baseada em energia

Como dito anteriormente, a energia de um fluxo é calculada conforme a Equação 2.7, utilizando-se dos valores de atributos de um conjunto de fluxos e dos hiperparâmetros definidos na subseção anterior. Dado que a energia de um fluxo é a soma negativa das energias locais e de acoplamentos, tem-se como resultado que fluxos mais semelhantes aos utilizados para a inferência do modelo têm valores de energia mais baixos. Assim, é possível definir um limiar para classificar novas amostras de fluxos como pertencentes ou não à classe utilizada para a inferência do modelo. O EFC opera com base neste princípio, podendo ser empregado em duas modalidades distintas de classificação: binária e multiclasse.

### 2.3.1 Classificação binária (EFC *Single-Class*)

A classificação binária pode ser formalmente descrita da seguinte forma: seja $\mathcal{F}_\ell \subseteq \mathcal{B}$ o subconjunto de fluxos rotulados com a classe $\ell \in L$, tal que $L = \{1, \ldots, n\}$ corresponde ao conjunto de todas as classes de fluxo presentes no conjunto de treinamento $\mathcal{B}$. Para cada fluxo em $F_\ell$, infere-se os parâmetros de acoplamento $e_{ij}^\ell$ e os campos locais $h_i^\ell$. Em seguida, calcula-se o vetor $\{\mathcal{H}_\ell(f)|\forall f \in F_\ell\}$, que corresponde ao Hamiltoniano calculado para as amostras de treino $F_\ell$, obtidas com base nos acoplamentos $e_{ij}^\ell$ e campos locais $h_{\ell_i}$.

A partir dessa distribuição de valores, define-se o limiar de classificação como:

$$\tau_\ell = \mathcal{Q}_p(\{\mathcal{H}_\ell(f)|\forall f \in F_\ell\}), \tag{2.13}$$

onde $\tau_\ell$ é o limiar energético de classificação para a classe $\ell$ e $\mathcal{Q}_p(\{\mathcal{H}_\ell(f)|\forall f \in F_\ell\})$ representa o $p$-ésimo quantil das *energias* calculadas para as amostras em $F_\ell$. Desta forma, o limiar é estabelecido de forma diretamente relacionada à distribuição de energias dos fluxos do treinamento, ajustando-se ao comportamento dos dados. Por influenciar diretamente a sensibilidade e a especificidade do modelo, o limiar quantílico de classificação $p$ é implementado como um dos hiperparâmetros do EFC, dado que assim é possível ajustar o modelo de acordo com o comportamento de classificação que se deseja obter.

A seguir, os valores calculados para $e_{ij}^\ell$, $h_i^\ell$ e $\tau_\ell$ são armazenados em uma estrutura de dados denominada *estimador*, a qual representa o modelo estatístico para a classe $\ell$ e será utilizada no processo de classificação de novos fluxos. Então, para se atribuir uma classe a

um novo fluxo suspeito $s$, utiliza-se a função

$$\mathcal{R}_\ell(s) = \begin{cases} \ell, & \mathcal{H}_\ell(s) \leq \tau_\ell; \\ \Psi, & \mathcal{H}_\ell(s) > \tau_\ell; \end{cases} \tag{2.14}$$

onde a classe é atribuída conforme o valor calculado para o Hamiltoniano associado à classe $\ell$ calculado para $s$. Assim, se $\mathcal{H}_\ell(s) \leq \tau_\ell$, o fluxo é atribuído à classe $\ell$; caso contrário, é atribuído o rótulo arbitrário $\Psi$ , tal que $\Psi \notin L$, que simplesmente representa o fato de que o fluxo $s$ pertence a uma classe desconhecida. Por conseguinte, o EFC originalmente opera como um modelo de classe única (*single-class*), isto é, um classificador que realiza classificações com base apenas no comportamento de uma classe de dados.

### 2.3.2 Classificação multiclasse (EFC *Multi-class*)

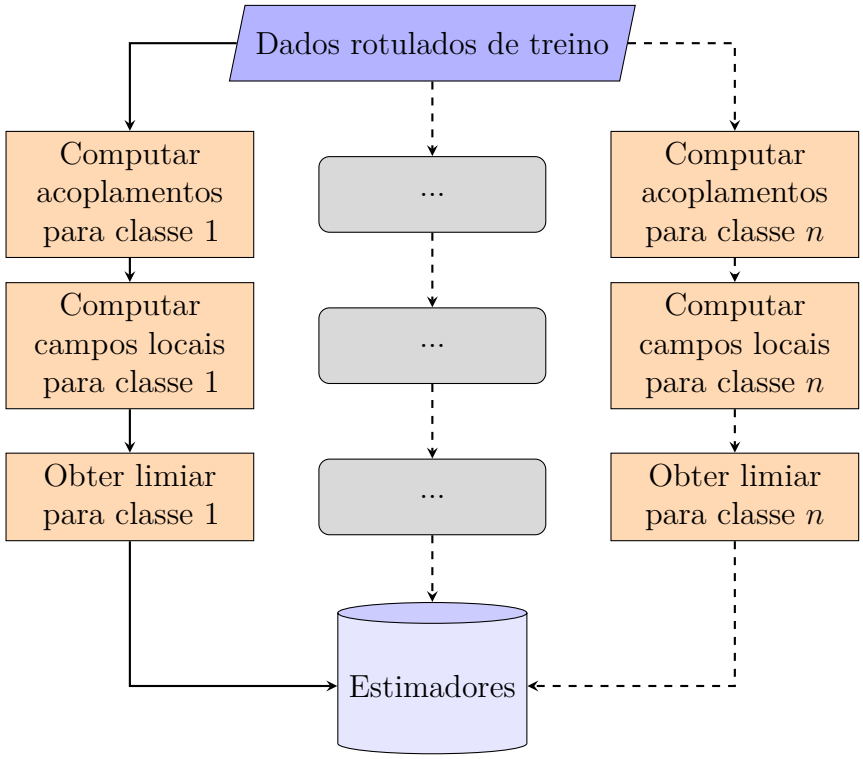
No modelo de classificação multiclasse proposto em Souza *et al.* [4] para o EFC, empregam-se as mesmas técnicas utilizadas no algoritmo de classificação binária. Assim, o parâmetro central para determinar se um fluxo pertence a uma determinada classe continua sendo seu valor de energia. A principal diferença entre as abordagens reside na quantidade de classes consideradas durante a inferência: enquanto a versão binária infere o comportamento de uma única classe, a versão multiclasse realiza a inferência para múltiplas classes de dados, permitindo que sejam comparados os valores de energia calculados para cada classe para a atribuição de um rótulo.

Dessa forma, o processo de treinamento do EFC ocorre conforme ilustrado na Figura 2.2, onde para cada $\ell \in L$ é inferido um *estimador* com base nos subconjuntos $F_\ell \subset \mathcal{B}$. Cada *estimador* armazena os parâmetros do modelo estatístico para sua respectiva classe $\ell$, ou seja, os valores de acoplamento $e_{ij}^\ell$, os campos locais $h_i^\ell$ e o limiar de classificação $\tau_\ell$. Na fase de classificação, ilustrada na Figura 2.3, calcula-se para um novo fluxo $s$ o vetor de energias $\{\mathcal{H}_\ell(s) \mid \forall \ell \in L\}$, em que cada $\mathcal{H}_\ell(s)$ é o Hamiltoniano obtido para o fluxo $s$ com base nos parâmetros do *estimador* correspondente à classe $\ell$. Com isso, é possível determinar a função

$$\mathcal{R}_L(s) = \begin{cases} r, & \mathcal{H}_r(s) \leq \tau_r \mid r = \arg\min_{\ell \in L} \mathcal{H}_\ell(s); \\ \Psi, & \mathcal{H}_r(s) > \tau_r \mid r = \arg\min_{\ell \in L} \mathcal{H}_\ell(s); \end{cases} \tag{2.15}$$

para atribuir um rótulo ao fluxo $s$. Nesta, a classe à qual pertence $s$ é determinada com base em $r$, que corresponde ao rótulo da classe que calculou o menor valor de energia para o fluxo $s$. Assim, de forma semelhante ao EFC *single-class*, avalia-se se o $s$ é anômalo com base no valor de energia calculado com base na classe $r$. Portanto, se $\mathcal{H}_r(s) \leq \tau_r$, atribui-se

Figura 2.2: Processo de treinamento multiclasse para o EFC.




ao fluxo a classe $r$; caso contrário, o fluxo é rotulado como $\Psi$, ou seja, pertencente a uma classe desconhecida.

Na seção a seguir, este processo pode ser melhor compreendido pela versão em pseudocódigo do modelo, onde são explicitados os hiperparâmetros vistos nas últimas duas seções e é ordenado em forma algorítmica o procedimento matemático descrito na Seção 2.2.

## 2.4 Definição algorítmica do EFC

Como modo de simplificar a compreensão do modelo, é elaborada a seguir uma implementação escrita em pseudocódigo de forma simplificada. A princípio, estabelecem-se os hiperparâmetros para o modelo:

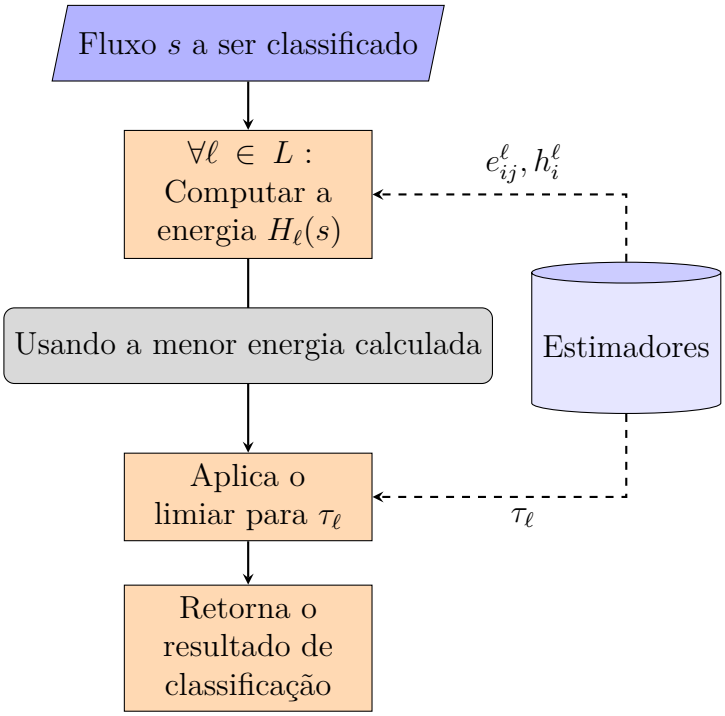
---

1: **Hiperparâmetros para o EFC:**
2: $\boldsymbol{\alpha}$ $|0 \leq \boldsymbol{\alpha} \leq 1$ : peso das pseudocontagens para o cálculo das frequências empíricas;
3: $\mathbf{p}$ $|0 \leq \mathbf{p} \leq 1$: quantil em percentual para o limiar de energia utilizado para a classificação;
4: $\mathbf{Q}$ $|\mathbf{Q} \in \mathbb{Z}$ e $\mathbf{Q} > 0$: número de níveis utilizado para discretização de atributos;

---

Em seguida, pode-se ver o Algoritmo 1, que contém a função `InferirModelo` utilizada para computar a inferência de um *estimador*. A função apresentada recebe $F_\ell$, uma

Figura 2.3: Classificação multiclasse com o EFC.




matriz de tamanho $(K \times N)$, composta por $K$ fluxos associados à mesma classe $\ell$ e que possuem o número $N$ de atributos. Ademais, o procedimento retorna um *estimador*, que aqui representa meramente uma estrutura de dados que contém os valores de $e_{ij}^\ell$,$h_i^\ell$ e $\tau_\ell$ calculados para as amostras de treino. A partir desse *estimador*, pode-se, posteriormente, acessar os valores de acoplamentos e campos locais para calcular a energia de um fluxo. Para computar esses valores, são usadas as funções `SiteFreq`, `PairFreq`, `Couplings`, `LocalFields`, `ComputeEnergies` e `CutoffQuantile`. Essas funções não serão explicitamente escritas, mas adaptam o processo descrito na Seção 2.2, ficando explícito nos comentários do pseudocódigo qual cálculo está sendo realizado.

**Algorithm 1** Inferência do modelo estatístico para o EFC

importar todas as funções de inferência do modelo
**function** INFERIRMODELO($F_\ell$)
    $f_i \leftarrow$ `SiteFreq`$(F_\ell, Q, \alpha)$; ▷ calcula as frequências individuais dos atributos
    $f_{ij} \leftarrow$ `PairFreq`$(F_\ell, f_i, Q, \alpha)$; ▷ calcula as frequências par-a-par dos atributos
    $e_{ij}^\ell \leftarrow$ `Couplings`$(f_i, f_{ij}, Q, )$; ▷ calcula os acoplamentos conforme as eqs.2.9 e 2.10
    $h_i^\ell \leftarrow$ `LocalFields`$(e_{ij}^\ell, f_i, Q)$; ▷ calcula os campos locais conforme a eq.2.12
    `energias` $\leftarrow$ `ComputeEnergies`$(F_\ell, e_{ij}^\ell, h_i^\ell)$; ▷ computa o vetor $\{\mathcal{H}_\ell(f) \mid f \in F_\ell\}$
    $\tau_\ell \leftarrow$ `CutoffQuantile(energias,`$p$`)`; ▷ computa o $p$-ésimo quantil para `energias` conforme a eq. 2.13
    **return** *estimador*$(e_{ij}^\ell, h_i^\ell, \tau_\ell)$;
**end function**

Adiante, para representar o processo de treinamento do classificador, tem-se o Algoritmo 2, que contém a função `Train`. Nesta, há os parâmetros de entrada da função $\mathcal{B}$ e $\mathcal{L}$,

respectivamente correspondendo ao conjunto de fluxos de treino na forma de uma matriz de forma $K \times N$, e um vetor de rótulos de tamanho $N$ associados aos fluxos de $\mathcal{B}$. Já `base_class` é um parâmetro opcional apenas usado para identificar qual classe de fluxo será utilizada para treinar em cenários de classificação binária. Aqui é válido citar que cada *estimador* gerado pela saída da função `InferirModelo` é armazenado de forma global em *estimadores*, para que possa(m) ser acessado(s) posteriormente na função de classificação. É válido destacar a presença da função `Tamanho`, aqui usada para determinar o tamanho de um vetor.

**Algorithm 2** Procedimento de treino para o Energy-Based Flow Classifier

```
1: function TRAIN(B,L,base_class)
2:     B ← valores de atributos em B discretizados quantilicamente em Q níveis;
3:     L ← rótulos distintos presentes em L
4:     if Tamanho(L) = 2 then
5:         if base_class não especificada then
6:             base_class ← primeiro rótulo presente em L;
7:         end if
8:         F_base_class ← todas as amostras de B rotuladas como base_class;
9:         estimadores[base_class]← InferirModelo(F_base_class);
10:    else
11:        for all ℓ in L do
12:            F_ℓ ← todas as amostras de B rotuladas como ℓ;
13:            estimadores[ℓ] ← InferirModelo(F_ℓ);
14:        end for
15:    end if
16: end function
```

Por fim, o Algoritmo 3, que inclui a função `Classify`, que representa o processo de classificação utilizando os *estimadores* treinados. Esta função, de forma similar às demais, recebe como parâmetro uma matriz de forma $K \times N$ com fluxos que serão testados, representada pelo parâmetro $\mathcal{S}$. Pode ser observado também que o pseudocódigo em questão retorna um vetor de rótulos, que é representado pela variável *predictions* (do inglês, predições). Note que no algoritmo, o rótulo arbitrário escolhido como equivalente a $\Psi$, visto nas eqs. 2.14 e 2.15, foi "*unknown*", do inglês, desconhecido. Ademais, os métodos `InsereNoFim` utilizados no pseudocódigo meramente representam a inserção de um elemento no fim de um vetor, sendo "[]"a representação de um vetor vazio.

É válido notar que nenhuma característica intrínseca do *Energy-based Flow Classifier* limita-o a ser exclusivamente um classificador de fluxos de rede, sendo plausível sua aplicação a outras áreas. Contudo, aplicações do modelo em outras áreas do conhecimento ainda não foram vistas na literatura.

Neste capítulo, foi detalhado o arcabouço teórico do EFC, abordando a abstração dos fluxos de rede em grafos, o método de inferência do modelo estatístico por meio de estatística inversa e o cálculo da Hamiltoniana utilizado para estimar a energia de cada fluxo, a qual fundamenta o processo de classificação. Também foram explicitados os pontos do modelo em que cada hiperparâmetro influencia diretamente seu comportamento, além da formalização do método de classificação baseado em energias. Por fim, foi apresentado o

**Algorithm 3** Classificação de fluxos com o Energy-Based Flow Classifier

**function** CLASSIFY($\mathcal{S}$)
  $\mathcal{S} \leftarrow \mathcal{S}$ discretizado quantilicamente em $Q$ níveis;
  *predictions* $\leftarrow$ [];
  *energias* $\leftarrow$ []
  **for all** $s$ in $\mathcal{S}$ **do**
    *hamiltonianas*$\leftarrow$[]
    **for all** estimador in *estimadores* **do**
      hamiltonianas.InsereNoFim($\mathcal{H}_{\text{estimador}}(s)$)
    **end for**
    *energias*.`InsereNoFim`(hamiltonianas)
  **end for**
  **for all** hamiltonianas in `energias` **do**
    **if** `Tamanho`(hamiltonianas)= 1 **then** ▷ classificação binária, equivalente à eq. 2.14
      $\mathcal{H}_{\texttt{base\_class}} \leftarrow$ valor de energia calculado para o único *estimador*;
      **if** $\mathcal{H}_{\texttt{base\_class}} \leq \tau_{\texttt{base\_class}}$ **then**
        *predictions*.`InsereNoFim`(rótulo da classe `base_class`);
      **else**
        *predictions*.`InsereNoFim`("*unknown*"); ▷ $\Psi$="*unknown*"
      **end if**
    **else** ▷ classificação multiclasse, equivalente à eq. 2.15
      Encontrar o $estimador_{min}$ que calculou o menor valor $\mathcal{H}_{min}$ em hamiltonianas;
      **if** $\mathcal{H}_{min} > \tau_{min}$ **then**
        *predictions*.`InsereNoFim`("*unknown*"); ▷ $\Psi$="*unknown*"
      **else**
        *predictions*.`InsereNoFim`(rótulo da classe associada ao $estimador_{min}$);
      **end if**
    **end if**
  **end for**
  **return** *predictions*;
**end function**

pseudocódigo que estrutura, de forma algorítmica, as etapas de treinamento e classificação, destacando onde cada hiperparâmetro atua no processo.

Assim, com a fundamentação teórica sobre o funcionamento do modelo estabelecida, o próximo capítulo abordará a metodologia adotada para a realização dos testes com os hiperparâmetros do modelo, detalhando os experimentos planejados, os conjuntos de dados selecionados e as especificações do sistema utilizado para a execução dos testes.

# Capítulo 3

# Trabalhos Relacionados

Nos últimos anos, diversos trabalhos foram produzidos com o intuito de explorar cenários de aplicação para o *Energy-based Flow Classifier*(EFC). Esta seção abordará alguns desses estudos relevantes para a presente pesquisa. Primeiramente, será apresentado o trabalho original que propôs o EFC, seguido pela extensão que introduziu sua versão multiclasse. Em seguida, serão discutidos estudos que analisam o desempenho do modelo em diferentes cenários de aplicação, bem como pesquisas que o integraram a sistemas de aprendizado federado. Na sequência, abordam-se investigações que utilizaram o EFC como base de referência para avaliar modelos propostos. Por fim, será discutida a aparente lacuna na literatura quanto à calibragem de seus hiperparâmetros.

## 3.1 Proposição original do modelo de classificação e sua versão multiclasse

Como mencionado anteriormente, o *Energy-based Flow Classifier* (EFC) foi inicialmente proposto em [8] como uma abordagem de detecção de intrusão baseada em fluxos de rede que dispensa amostras maliciosas para seu treinamento. Inspirado no modelo de Potts da física quântica, o método infere um modelo estatístico exclusivamente a partir de fluxos benignos, calculando uma "energia" para cada fluxo com base em acoplamentos entre características (portas, protocolos, duração etc.) e campos locais. Fluxos cujo valor energético ultrapassa um limiar (percentil 95 da distribuição de energia de treinamento) são sinalizados como anômalos. O EFC foi validado empiricamente em três conjuntos de dados (CIDDS-001, CICIDS17 e CICDDoS19), demonstrando desempenho adequado (F1-score em torno de 0.97 e AUC próximo de 0.99 em testes intrasset) e destacando-se em cenários com treino e teste realizados em conjuntos diferentes de dados, onde supera classificadores tradicionais como SVM e MLP em até 52% de ganho de F1. Além disso,

o modelo apresenta interpretabilidade, uma vez que permite decompor a energia em contribuições de pares de características, facilitando a análise de decisões e a identificação de padrões específicos de ataque.

De Souza *et al.* [4] estendem o EFC original para um cenário multiclasse e *open-set*, isto é, de modo a tratar não só de tráfego benigno e diferentes classes de ataques (*DDoS*, *PortScan*, *Bot* etc.), mas também identificar ataques desconhecidos. Mantendo a mesma formulação energética, o EFC multiclasse infere um modelo estatístico para cada classe — benigna e cada tipo de ataque conhecido — e aplica limiares energéticos individuais (percentil 95) para atribuir rótulos. Validado no CICIDS2017, o classificador multiclasse alcança macro-F1 de 0.752 quando não se avalia a detecção de uma classe desconhecida, superando o segundo maior pontuador, *Decision Tree* (0.731), em 8 de 13 classes, e demonstra capacidade de detecção para classes de ataque desconhecido, detectando mais de 80% dos ataques não vistos, com AUPRC média de 0,993, superior a redes neurais avançadas como OCN [15]. O estudo também ressalta a eficiência computacional em termos de tempo de treinamento do modelo, o que viabiliza a implantação em sistemas de monitoramento de alto desempenho em tempo real.

Esses artigos apresentam a fundamentação teórica do algoritmo EFC, incluindo a formulação matemática do processo de inferência do modelo estatístico. O conteúdo foi sintetizado e será exposto no Capítulo 2 deste trabalho, acompanhado de uma representação em pseudocódigo do algoritmo e pela indicação precisa dos pontos em que os hiperparâmetros intervêm na inferência do modelo.

## 3.2 Trabalhos que avaliam o desempenho do modelo em diferentes aplicações

Uma vez proposto, o *Energy-Based Flow Classifier* tem sido explorado em distintas aplicações com foco na detecção de anomalias de maneira supervisionada. No contexto de segurança de rede, Lopes *et al.* [9] aplicou o EFC para reconhecer comportamento de *botnets* em fluxos de rede, comparando-o sistematicamente com classificadores binários (*KNN*, *SVM*, *MLP*, *Random Forest* etc.) e unários (*OCSVM*, *Isolation Forest*, *LOF* e *Elliptic Envelope*). Os experimentos em cenários intra-domínio (CTU-13 e ISOT HTTP) revelaram F1-score acima de 0,98 e AUC acima de 0,99 em ISOT HTTP, e 0,87/0,96 em CTU-13, sendo superado apenas pelo algoritmo *Local Outlier Factor* (LOF). Enquanto nos testes inter-domínio o EFC manteve desempenho superior aos métodos comparados em termos de F1-score em um dos experimentos, com pontuação de 0.66 e em outro, obteve a maior pontuação em termos de AUC 0.73, evidenciando robustez a variações de distribuição dos fluxos.

Já no contexto de detecção de intrusão em redes móveis, de Almeida *et al.* [10] empregou o EFC integrado a técnicas de agrupamento (*clustering*) para segmentar regiões urbanas, suburbanas e rurais a partir de registros de detalhe de chamada (CDR). Validado em 62 dias de dados de Milão, utilizando registros de referência de eventos em estádios, o método alcançou F1-score de 0.96, pontuação essa mais de 35% acima do melhor detector comparado (método de agrupamento baseado em K-médias). Esses trabalhos reafirmam a aplicabilidade do EFC em cenários realistas e diversos, desde redes de computadores com tráfego potencialmente malicioso até redes móveis com padrões de uso regionais distintos.

## 3.3 Utilização do EFC como componente em aprendizado federado

Ademais, o EFC tem sido empregado como componente auxiliar em diversos sistemas recentes de detecção de intrusão, especialmente no contexto de aprendizado federado (*Federated Learning*, FL) e redes heterogêneas. Seu uso mostra-se promissor para aprimorar a capacidade de generalização e reduzir a necessidade de supervisão em cenários onde os dados não são independentes nem identicamente distribuídos (*non-IID*).

O trabalho de Bertoli *et al.* [11] propõe uma arquitetura de detecção de intrusão baseada em aprendizado federado não supervisionado com uma abordagem empilhada, combinando um *autoencoder* profundo (*Deep Autoencoder*, DAE) com o EFC como fonte de novas *features*. A proposta é avaliada em quatro bases de dados (UNSW-NB15, CSE-CIC-IDS-2018, Bot-IoT e ToN-IoT), simulando um ambiente federado do tipo interdomínio (*cross-silo*). Os autores mostram que sua abordagem supera métodos tradicionais de aprendizado local e bases de referência como *Isolation Forest* e LOF, obtendo F1-score médio de 0.84 no 10º round de FL. Ademais, o trabalho demonstra como o uso do EFC tem impacto significativo no desempenho do modelo, evidenciado pelo F1-score de 0.47 obtido pelo *Deep Autoencoder* no mesmo estágio de treinamento quando não associado ao EFC.

Em outro trabalho relevante, Zhu *et al.* [12], o EFC é utilizado em combinação com um modelo de *Gaussian Mixture Model* (GMM) no contexto de aprendizado federado em redes heterogêneas. Neste esquema, cada cliente local obtém novas *features* por meio do uso do EFC e utiliza-as somadas ao conjunto de dados para o treinamento do GMM, utilizando a estratégia *FedAdagrad*. A proposta é avaliada com os mesmos conjuntos de dados e metodologia propostos em Bertoli *et al.* [11], dessa vez obtendo F1-score de 84.94% no 10º round de FL, superando significativamente o GMM isolado (52.47%) e algoritmos de detecção clássicos. Os resultados em diferentes datasets confirmam a eficácia

da abordagem mesmo sob forte desbalanceamento de classes e variação de distribuição entre as organizações.

Já Wan *et al.* [13], apresentam o STIN-IDS, um sistema voltado para redes integradas satélite-terrestres, no qual satélites LEO realizam a coleta e pré-processamento dos dados, enquanto satélites GEO participam de um esquema de FL não supervisionado. Neste sistema, o EFC é utilizado para extrair a energia de cada fluxo de rede e adicioná-la ao conjunto de características processadas por um *autoencoder*. Os testes mostram desempenho consistente em diferentes bases de dados, com F1-score superior a 0.91 e acurácia de até 0.97, demonstrando que a arquitetura é capaz de lidar com variações regionais e mobilidade na rede. O modelo também se mostra resiliente a mudanças abruptas na distribuição dos dados ao longo do tempo.

Esses trabalhos demonstram que o EFC contribui significativamente para a generalização e adaptação a diferentes domínios em modelos de aprendizado federado, consolidando-se como um componente essencial para o funcionamento dessas propostas.

## 3.4 O EFC usado como modelod de referência para avaliação de outros modelos

Ademais, o *Energy-based Flow Classifier* tem sido utilizado como modelo de referência em estudos recentes voltados para a detecção de intrusões em redes, especialmente pela sua capacidade de adaptação a novos domínios sem a necessidade de um processo complexo de treinamento como o de redes neurais. Diversos trabalhos se apoiam no EFC para avaliar a eficácia de modelos mais complexos, com foco em robustez e capacidade de generalização. A seguir, são apresentados três estudos que utilizam os resultados obtidos pelo EFC em seus respectivos trabalhos como referência para validar suas propostas.

Nguyen *et al.* [16] propõe um sistema de detecção de intrusão baseado em uma abordagem sequencial utilizando o modelo BERT, originalmente aplicado em Processamento de Linguagem Natural. O sistema modela a sequência temporal de fluxos de rede como sentenças, permitindo que padrões de comportamento contextualizados sejam capturados. A arquitetura do sistema utiliza o BERT para extrair vetores de características de sequências de fluxos e um classificador MLP para a tomada de decisão. Em testes realizados com os conjuntos CIDDS-001 e CIDDS-002, o modelo proposto apresentou desempenho superior ao EFC, especialmente em ambientes de domínio distinto, evidenciado por métricas como F1-score e acurácia. Ainda assim, o EFC se destacou como o segundo melhor modelo em diversos cenários, mostrando sua eficácia relativa mesmo diante de arquiteturas mais complexas.

Outro trabalho relevante é o de de Melo *et al.* [17], que propõe o *framework Anomaly-Flow*, voltado à detecção de ataques de negação de serviço distribuídos (*Distributed Denial of Service*, DDoS) em ambientes multi-domínio (*multi-silo*), utilizando uma combinação de Aprendizado Federado e Redes Generativas Adversariais (GANs). A proposta baseia-se em treinar localmente modelos GANomaly adaptados a dados tabulares de fluxo de rede, compartilhando apenas parâmetros agregados entre os domínios (*silos*), o que preserva a privacidade dos dados. Após o treinamento, os modelos são utilizados para gerar fluxos sintéticos, que alimentam classificadores heterogêneos em ambientes externos. O EFC foi empregado como modelo de comparação nos experimentos de detecção inter-domínio com conjuntos CICIDS2018, Bot-IoT e TON-IoT. O EFC demonstra resultados competitivos, sendo o terceiro melhor classificador em geral com F1-score de 0.648; contudo, o *Anomaly-Flow* demonstrou maior capacidade de generalização ao integrar dados distribuídos, alcançando o maior F1-score relatado (0.747).

Em outro trabalho, Melo *et al.* [18] explora também a generalização de classificadores de ataques DDoS por meio de Aprendizado Federado, aplicando-o a múltiplos domínios de dados (*silos*) extraídos de redes distintas (TON-IoT, CICIDS2018 e Bot-IoT). A proposta consiste no treinamento distribuído de modelos de regressão logística com técnicas de balanceamento (subamostragem e SMOTE) e seleção de atributos, sem compartilhamento direto de dados entre os domínios. O EFC foi utilizado como referência de comparação em todos os experimentos. Os resultados mostram que a combinação de FL com subamostragem e seleção de atributos produziu F1-score médio comparável ao do EFC, obtendo a pontuação de 0.50 de F1-score, enquanto o EFC apresentou os maiores resultados do trabalho pontuando 0.53 na mesma métrica.

Esses estudos evidenciam a importância do EFC como base comparativa sólida na literatura de detecção de intrusão. Embora não seja sempre o modelo com melhor desempenho absoluto, seu equilíbrio entre simplicidade e robustez o torna um candidato adequado para testes comparativos.

## 3.5 A lacuna de pesquisa identificada

Apesar dos trabalhos apresentados evidenciarem a versatilidade do EFC em cenários heterogêneos e variados, quase nenhum destes aborda a configuração de hiperparâmetros utilizada para a realização de seus experimentos. De forma resumida, o EFC possui três hiperparâmetros: o limiar de classificação, o peso de pseudocontagens e o número de níveis de discretização para os dados (mais informações sobre estes parâmetros são apresentadas no próximo capítulo). Dentre os estudos trabalhados neste capítulo, apenas o estudo de Souza *et al.* [4] menciona quais foram os valores utilizados para todos estes

parâmetros, especificando os seguintes valores: 95º percentil das energias calculadas como limiar de classificação, peso de 0.5 para as pseudocontagens de frequências e 30 níveis de discretização para os dados. Não por acaso, esses são os valores padrão atribuídos aos hiperparâmetros do classificador, conforme verificado na implementação disponível na plataforma GitHub [19]. Diante disso, é razoável supor que os trabalhos que adotaram o EFC como base tenham utilizado essa configuração ou variações muito próximas, o que indica a possibilidade de existirem oportunidades de otimização de desempenho ainda não investigadas. Para uma investigação desta hipótese, torna-se necessário aprofundar a fundamentação teórica do modelo, a fim de compreender de forma mais precisa o papel e o impacto de seus hiperparâmetros no processo de inferência, tema este que será abordado no próximo capítulo.

# Capítulo 4

# Metodologia

Como relatado no capítulo anterior, o *Energy-based Flow classifier* possui três hiperparâmetros: $\alpha$, $Q$ e $p$; sendo necessária a atribuição de valores a estes antes do treinamento do modelo. Neste capítulo, então, será descrito o método utilizado para a avaliação do desempenho de classificação sob diferentes configurações para estes parâmetros globais. Para isto, primeiramente serão descritos os experimentos planejados. Em seguida, serão apresentados os conjuntos de dados que serão utilizados nos experimentos e as técnicas de balanceamento aplicadas a estes. Por fim, será descrito o ambiente de execução utilizado para a realização dos experimentos.

## 4.1 Metodologia de teste para o impacto dos hiperparâmetros nos resultados de classificação

Dado que o modelo é capaz de realizar tanto classificações binárias quanto multiclasse, conforme destacado na Seção 2.3, ambas as modalidades serão contempladas nos experimentos. Para cada uma delas, serão conduzidas múltiplas iterações de classificação com diferentes configurações dos hiperparâmetros, com o objetivo de avaliar as variações no desempenho do modelo em função das combinações de valores dentro de uma faixa previamente definida. Ainda, em cada iteração de classificação, serão usados 80% dos dados dos *datasets* para treino e 20% para teste.

A primeira etapa planejada para os experimentos aborda dois dos hiperparâmetros do modelo, que serão avaliados em conjunto: o $p$, responsável por determinar o quantil das energias de treino utilizado como limiar de classificação, e o número de níveis utilizados na discretização dos atributos dos dados de treino ($Q$). A segunda etapa, por sua vez, busca avaliar o impacto de diferentes pesos para as pseudocontagens ($\alpha$), que também serão analisados junto ao parâmetro $Q$, por razões que serão justificadas na subseção

dedicada a este experimento. A seguir, são descritas as faixas de valores analisadas para cada parâmetro, além das especificidades de cada experimento realizado.

### 4.1.1 Experimentos com o limiar de classificação e com o número de níveis de discretização dos dados

A escolha das faixas de valores para os hiperparâmetros do modelo foi baseada em torno dos valores selecionados como padrão para o modelo visto no GitHub [19] e em Souza *et al.*[4]. Estes valores são, especificamente: 0.95 para o limiar de classificação ($p$) e 30 para o número de níveis de discretização ($Q$).

A faixa de valores escolhida para a experimentação com o limiar de classificação foi definida entre 0.90 e 0.99, com incremento uniforme de 0.01 a cada iteração. De forma similar, a faixa de valores inteiros de 10 a 100, com incremento de 10 em cada iteração, será utilizada para a avaliação de diferentes níveis de discretização para os dados. Todos os valores escolhidos para cada parâmetro são apresentados por extenso na Tabela 4.1, como forma de facilitar sua visualização.

Tabela 4.1: Faixa de valores testados para os hiperparâmetros do modelo EFC.

| **Hiperparâmetro** | **Valores Testados** |
|---|---|
| $p$ | {0.90, 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99} |
| $Q$ | {10, 20, 30, 40, 50, 60, 70, 80, 90, 100} |

A cada iteração de teste, dois valores dentro das faixas estabelecidas (um para cada hiperparâmetro) serão selecionados e empregados no treinamento do modelo. O desempenho de cada modelo treinado será avaliado com base nos dados reservados para teste. Esse processo será repetido para todos os pares de valores possíveis dentro das faixas estabelecidas para os hiperparâmetros, garantindo que os mesmos dados serão usados em todas as iterações de treino e teste.

### 4.1.2 Experimentos com os pesos de pseudocontagens

Para avaliar como os pesos de pseudocontagens de frequência $\alpha$ afetam os resultados de classificação, foram elaborados dois experimentos:

#### Experimentos com os pesos de pseudocontagens e com o número de níveis de discretização dos dados

Primeiramente, serão realizados testes de classificação sobre o hiperparâmetro $\alpha$, que define o peso atribuído às pseudocontagens das frequências durante a etapa de inferência do modelo

estatístico. Como descrito na seção 2.2, a introdução destes pesos é realizada para evitar problemas de subamostragem (*undersampling*) nos dados de treino, equivalendo à adição de uma fração extra de fluxos com valores distribuídos uniformemente. Para a avaliação do impacto deste parâmetro na classificação, foram realizadas múltiplas classificações utilizando diferentes valores para $\alpha$ e para níveis de discretização dos atributos.

Assim como nos experimentos sobre $Q$ e $p$, é estabelecida uma faixa de valores a serem testados para $\alpha$ e $Q$. A faixa testada para *alpha* contempla valores de 0.1 a 0.9, incrementados em 0.1 a cada iteração. Os níveis de discretização foram avaliados na faixa de 10 a 100, incrementados em 10 a cada iteração, assim como no experimento anterior. A Tabela 4.2 apresenta as faixas estabelecidas para o experimento por extenso.

Tabela 4.2: Faixa de valores testados para os hiperparâmetros do modelo EFC.

| **Hiperparâmetro** | **Valores Testados** |
|---|---|
| $\alpha$ | {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9} |
| $Q$ | {10, 20, 30, 40, 50, 60, 70, 80, 90, 100} |

A combinação de diferentes valores de $\alpha$ com os níveis de discretização $Q$ foi realizada para investigar possíveis interações entre esses parâmetros. A escolha de valores baixos para $\alpha$ resulta em um modelo mais precisamente adaptado aos dados de treino; contudo, supõe-se o risco de que esta escolha venha a inviabilizar o treinamento do modelo em cenários de alta discretização dos dados. Isso supostamente ocorreria em casos em que a baixa frequência de determinados valores de atributos possa resultar no determinante da matriz de correlações ser nulo. O que, por sua vez, ocasiona a impossibilidade de realizar a inversão de matriz presente na eq. 2.9, então tornando impossível o processo de treinamento do modelo. Ademais, supõe-se a possibilidade de que esse ajuste aos dados prejudique a capacidade de adaptação a diferentes domínios de dados, o que será investigado no próximo experimento.

**Avaliação com o objetivo de verificar se valores menores de $\alpha$ tendem a provocar perda de adaptabilidade no modelo.**

Para confirmar se há problemas relacionados à perda de adaptabilidade, foi realizado um teste cruzado utilizando dois conjuntos de dados distintos. Em cada experimento, um dos conjuntos foi usado integralmente para treinamento e o outro integralmente para teste, invertendo-se os papéis dos conjuntos em uma segunda iteração. Quanto à configuração de hiperparâmetros definida para estes experimentos, buscou-se avaliar diferentes valores apenas para os pesos das pseudocontagens de frequências, sendo testados três valores: 0.1,

0.5 e 0.9. Os níveis de discretização e o limiar de classificação permaneceram fixos nos valores padrão (30 e 0.95, respectivamente).

Esta perda de adaptabilidade pode ser melhor visualizada ao montar um histograma das energias calculadas na eq. 2.7 para cada um dos fluxos classificados. Neste histograma, se não houver uma distinção entre os níveis de energia da classe utilizada no treinamento do modelo e os níveis das demais classes, pode-se dizer que o modelo não foi capaz de inferir o comportamento da classe utilizada para treino.

Este segundo experimento foi realizado apenas na modalidade de classificação binária, dado que os tipos de classes de ataque dos dois conjuntos de dados escolhidos não são compatíveis. Ademais, a visualização da separação entre os níveis de energia das classes no histograma é mais simples na classificação binária, dado que só é necessário observar as energias calculadas por um *estimador*.

### 4.1.3 Métricas de Avaliação de Desempenho: AUC-ROC e F1-score

Para a avaliação de desempenho do modelo de classificação, foram escolhidas duas métricas comumente utilizadas: AUC-ROC e F1-Score. Ambas são calculadas utilizando as predições realizadas pelo classificador ao atribuir uma classe aos fluxos de teste. As métricas escolhidas fornecem informações sobre a capacidade do modelo de discriminação entre as classes e o equilíbrio entre precisão e revocação.

#### AUC-ROC

A métrica AUC-ROC (Área sob a Curva Característica de Operação do Receptor) quantifica a habilidade do modelo em classificar corretamente as instâncias entre as classes. A curva ROC é construída a partir da Taxa de Verdadeiros Positivos (TPR) em relação à Taxa de Falsos Positivos (FPR) para diferentes limiares de classificação. A AUC é a área sob essa curva e varia entre 0 e 1. A fórmula para calcular a AUC é:

$$AUC = \int_0^1 \mathrm{TPR}(FPR)\, dFPR$$

#### F1-score

A métrica F1-score, por sua vez, é definida como a média harmônica entre a precisão e a revocação, considerando o desempenho do modelo em relação a falsos positivos e falsos negativos. A fórmula do F1-score é:

$$F1 = 2 \times \frac{\text{Precisão} \times \text{Revocação}}{\text{Precisão} + \text{Revocação}}$$

Onde a precisão (Precisão) é calculada como:

$$\text{Precisão} = \frac{TP}{TP + FP}$$

E a revocação (Revocação) é calculada como:

$$\text{Revocação} = \frac{TP}{TP + FN}$$

Onde $TP$ representa os Verdadeiros Positivos, $FP$ os Falsos Positivos e $FN$ os Falsos Negativos. O valor do F1-score varia de 0 a 1, com valores mais próximos de 1 indicando um melhor equilíbrio entre precisão e revocação.

## 4.2 Os *datasets*

Primeiramente, para a escolha dos conjuntos de dados que serão utilizados, foi dada preferência a *datasets* amplamente utilizados na literatura que já houvessem sido utilizados em experimentos *inter-dataset* utilizando o EFC. Portanto, foram escolhidos dois dos conjuntos de dados utilizados no estudo de Pontes *et al.*[8], o CICIDS17 [20] e o CICDDoS19 [21]; ambos publicados pelo Canadian Institute of Cybersecurity.

Ambos os *datasets* foram gerados a partir de dados de tráfego simulado de rede armazenados em arquivos de captura (PCAP). Estes, então, tiveram as informações dos fluxos extraídas pela ferramenta *CICFlowMeter*, que define e rotula os fluxos de rede baseando-se nas informações de *Timestamp*, *IPs* de origem e destino, portas, protocolos e ataques. Essas informações são armazenadas em um arquivo do formato *CSV*, onde cada linha corresponde a um dos fluxos de rede identificados, rotulado apropriadamente conforme a classe de tráfego que ele representa. Os arquivos de dados utilizados neste trabalho foram os CSV, por estarem em um formato já compatível para o uso do EFC.

Ambos os conjuntos de dados compartilham o método de geração de tráfego benigno, no qual é simulado o comportamento abstrato de 25 usuários para atuar como tráfego de fundo, cobrindo protocolos como *HTTP*, *HTTPS*, *FTP*, *SSH* e *e-mail*; garantindo que há um padrão de comportamento benigno que possa ser aprendido pelo EFC. Contudo, apesar de haver semelhanças entre os dados, os conjuntos trabalhados possuem suas particularidades, fazendo com que seja necessário que se discorra brevemente sobre suas características e as abordagens de tratamento de dados aplicadas a cada um deles.

### 4.2.1 CICIDS17

O *dataset* CICIDS2017 contém dados benignos e de ataques cibernéticos comuns e atualizados. Neste estão contidos dados de tráfego de ataques cibernéticos simulados, sendo eles: *Brute Force* ( baseado em *FTP* e baseado em *SSH*), *DoS*, *Heartbleed*, *Web Attack*, *Infiltration*, *Botnet* e *DDoS*. O arquivo CSV deste conjunto de dados é composto por 78 colunas correspondentes às características dos fluxos extraídos, além de uma coluna dedicada a rotular a categoria à qual cada fluxo pertence. Para este trabalho, todas as colunas de características deste foram utilizadas.

Para este conjunto de dados, foi realizada uma re-rotulagem dos fluxos inicialmente rotulados para a categoria "*Web Attack*", nominalmente: "*Web Attack - Brute Force*", "*Web Attack - XSS*"e "*Web Attack - SQL Injection*". Estes fluxos foram rotulados para apenas "*Web Attack*", para que as amostras desta categoria não fossem descartadas por falta de amostras suficientes para a classificação. Desta forma, tem-se que o número de amostras de cada categoria de fluxo deste *dataset* pode ser visto na Tabela 4.3.

Tabela 4.3: Distribuição original das amostras de fluxo do *dataset* CICIDS2017

| CICIDS2017 | |
|---|---|
| Classe | número de amostras |
| BENIGN | 2271320 |
| DoS Hulk | 230124 |
| PortScan | 158804 |
| DDoS | 128025 |
| DoS GoldenEye | 10293 |
| FTP-Patator | 7935 |
| SSH-Patator | 5897 |
| DoS slowloris | 5796 |
| DoS Slowhttptest | 5499 |
| Web Attack | 2180 |
| Bot | 1956 |
| Infiltration | 36 |
| Heartbleed | 11 |

### 4.2.2 CICDDoS19

Já quanto ao conjunto de dados CICDDoS19, este possui dados benignos e de ataques DDoS. Este contém os dados de diversos ataques DDoS baseados em protocolos da camada de aplicação (por exemplo: *NetBIOS*, *LDAP*, *MSSQL*, *DNS*, *SNMP*, *NTP*, *TFTP* e

*WebDDoS*), em vulnerabilidades de protocolos da camada de transporte (*UDP Lag*, *UDP Flood* e *SYN Flood*), assim como ataques de enumeração de porta (*PortMap*).

O arquivo CSV deste *dataset*, diferentemente do CICIDS17, é composto por 82 colunas correspondentes às características dos fluxos extraídos, além de uma coluna dedicada a rotular a categoria à qual cada fluxo pertence. Uma vez que este conjunto possui colunas que não estão presentes no CICIDS17, algumas colunas foram descartadas para que fosse possível realizar testes cruzados entre os dois. Especificamente, foram removidas as colunas "*Source Port*", "*SimillarHTTP*", "*Inbound*"e "*Protocol*".

Ademais, foi necessário que houvesse uma re-rotulação dos dados deste conjunto, uma vez que a rotulação original possuía inconsistências na seleção de nomes para os rótulos, como pode ser notado na Tabela 4.4. Para tal re-rotulação dos fluxos, foi consultado o trabalho de Sharafaldin *et al.*[21] no qual o conjunto de dados foi proposto. Com base nele, foram feitos os seguintes ajustes:

- primeiramente, os fluxos rotulados como "*UDP-lag*"e "*UDPLag*"foram interpretados como representando a mesma classe de fluxos, portanto tiveram seu rótulo ajustado para uniformizar a grafia como "*UDPLag*";
- ademais, não foi identificada nenhuma razão para que houvesse uma distinção entre os fluxos rotulados com o prefixo "*DrDoS_*"e sua contraparte rotulada sem tal prefixo (por exemplo: "*DrDoS_MSSQL*"e "*MSSQL*"), uma vez que tal diferença não é exposta em nenhum momento na publicação associada ao *dataset*. Portanto os fluxos assim rotulados tiveram esse prefixo removido na re-rotulação.

Enfim, é válido mencionar que a categoria de ataque "UDP"é ambígua e não é propriamente descrita no trabalho de Sharafaldin *et al.*[21], então não é possível confirmar que tipo de ataque ela representa. Porém, considerando o que é apresentado no trabalho, supõe-se que esta classe corresponda ao ataque *UDP-flood*, portanto a classe não será removida.

### 4.2.3 Balanceamento dos conjuntos de dados para a classificação binária

Para os experimentos de classificação binária com o EFC, os *datasets* foram balanceados de forma a conter 10,000 amostras de fluxos benignos e uma soma de 10,000 amostras de fluxos maliciosos, buscando manter uma proporção de representatividade similar entre todas as classes, embora algumas destas possuam menos amostras de fluxos que o necessário para que se mantenha o mesmo número das demais. Estas classes que possuem um número menor que o mínimo necessário para se igualar com a proporção das outras classes foram descartadas da classificação.

Tabela 4.4: Distribuição original das amostras de fluxo do *dataset* CICDDoS19

| CICDDoS2019 | |
|---|---|
| Classe | número de amostras |
| BENIGN | 113828 |
| DrDoS_DNS | 5071011 |
| DrDoS_LDAP | 2179930 |
| DrDoS_MSSQL | 4522492 |
| DrDoS_NTP | 1202642 |
| DrDoS_NetBIOS | 4093279 |
| DrDoS_SNMP | 5159870 |
| DrDoS_SSDP | 2610611 |
| DrDoS_UDP | 3134645 |
| LDAP | 1915122 |
| MSSQL | 5787453 |
| NetBIOS | 3657497 |
| Portmap | 186960 |
| Syn | 6473789 |
| TFTP | 20082580 |
| UDP | 3867155 |
| UDP-lag | 366461 |
| UDPLag | 1873 |
| WebDDoS | 439 |

Isto é, para o dataset CICIDS17, as amostras das classes "Infiltration"e "Heartbleed"foram descartadas. Enquanto que para o CICDDoS19, somente as amostras da categoria "WebDDoS"foram descartadas. O balanceamento final para ambos os *datasets* para os experimentos de classificação binária pode ser visto na Tabela 4.5.

### 4.2.4 Balanceamento dos conjuntos de dados para a classificação multiclasse

Para o balanceamento dos dados para classificação multiclasse, foi estabelecido o limite máximo de 5,000 amostras para cada categoria de fluxo. Para as classes com número de amostras inferior a 5,000, foram consideradas apenas aquelas que possuíam mais de 1,000 amostras. As classes que possuem menos de mil amostras foram descartadas, uma vez que não seria possível realizar um treinamento de detecção adequado para estas. Desta forma, as amostras classificadas como "*Infiltration*", "*Heartbleed*"e "*WebDDoS*"foram descartadas

Tabela 4.5: Número de amostras por classe nos *datasets* CICDS2017 e CICDDoS2019 após o balanceamento para experimentos de classificação binária

| CICIDS2017 | |
|---|---|
| Classe | número de amostras |
| BENIGN | 10000 |
| DoS Hulk | 1000 |
| PortScan | 1000 |
| DDoS | 1000 |
| DoS GoldenEye | 1000 |
| FTP-Patator | 1000 |
| SSH-Patator | 1000 |
| DoS slowloris | 1000 |
| DoS Slowhttptest | 1000 |
| Web Attack | 1000 |
| Bot | 1000 |

| CICDDoS2019 | |
|---|---|
| Classe | número de amostras |
| BENIGN | 10000 |
| DNS | 834 |
| LDAP | 834 |
| MSSQL | 834 |
| NetBIOS | 834 |
| NTP | 833 |
| SNMP | 833 |
| SSDP | 833 |
| UDP | 833 |
| Portmap | 833 |
| Syn | 833 |
| TFTP | 833 |
| UDPLag | 833 |

destes experimentos também. O balanceamento final para ambos os *datasets* para os experimentos de classificação multiclasse pode ser visto na Tabela 4.6.

Tabela 4.6: Número de amostras por classe nos *datasets* CICDS2017 e CICDDoS2019 após o balanceamento para os experimentos de classificação multiclasse

| CICIDS2017 | |
|---|---|
| Classe | número de amostras |
| BENIGN | 5000 |
| DoS Hulk | 5000 |
| PortScan | 5000 |
| DDoS | 5000 |
| DoS GoldenEye | 5000 |
| FTP-Patator | 5000 |
| SSH-Patator | 5000 |
| DoS slowloris | 5000 |
| DoS Slowhttptest | 5000 |
| Web Attack | 2180 |
| Bot | 1956 |

| CICDDoS2019 | |
|---|---|
| Classe | número de amostras |
| BENIGN | 5000 |
| DNS | 5000 |
| LDAP | 5000 |
| MSSQL | 5000 |
| NetBIOS | 5000 |
| NTP | 5000 |
| SNMP | 5000 |
| SSDP | 5000 |
| UDP | 5000 |
| Portmap | 5000 |
| Syn | 5000 |
| TFTP | 5000 |
| UDPLag | 5000 |

## 4.3 Equipamentos Utilizados

Os experimentos foram realizados remotamente utilizando a plataforma *Google Colab*, um ambiente de *notebook Python* baseado em nuvem. A plataforma disponibiliza uma unidade virtualizada com duas *CPU Intel(R) Xeon(R) CPU* 2.20GHz alocadas pela plataforma e 13 GB de memória RAM. A plataforma também disponibiliza GPUs de forma gratuita, mas para os experimentos realizados não foi necessário o uso desse equipamento. O ambiente utiliza *Python* versão 3.10, sendo feito também o uso de múltiplas bibliotecas complementares para: o manuseio dos conjuntos de dados (*NumPy* e *Pandas*), a elaboração de imagens para a visualização dos resultados dos experimentos (*Matplotlib* e *Seaborn*), o manuseio de arquivos dentro do ambiente virtual (*Tkinter* e *Glob*), o uso de ferramentas gerais de aprendizado de máquina (*SciKitLearn*).

Neste capítulo, foram descritas as estratégias experimentais adotadas para avaliar sistematicamente os três principais hiperparâmetros do EFC em tarefas de classificação binária e multiclasse. Detalharam-se as faixas de valores testadas para o limiar quantílico de classificação ($p$), os níveis de discretização ($Q$) e o peso das pseudocontagens ($\alpha$), bem como o particionamento dos dados em 80% para treino e 20% para teste, os procedimentos de balanceamento de classes e as métricas de desempenho utilizadas (AUC-ROC e F1-Score). Além disso, foi descrito o ambiente computacional utilizado para a execução dos experimentos. No capítulo seguinte, são apresentados e discutidos os resultados obtidos a partir das experimentações aqui delineadas.

# Capítulo 5

# Resultados e Discussão

Neste capítulo, são apresentados os resultados dos experimentos descritos no Capítulo 4, seguidos de uma discussão sobre os achados obtidos. Com base nesses resultados, é sugerida uma abordagem para a calibragem dos hiperparâmetros do *Energy-Based Flow Classifier*. Inicialmente, são analisados os resultados referentes aos experimentos com o limiar de classificação e o número de níveis de discretização. Em seguida, são discutidos os experimentos relacionados aos pesos das pseudocontagens.

## 5.1 Experimentos com níveis de discretização e limiar de classificação

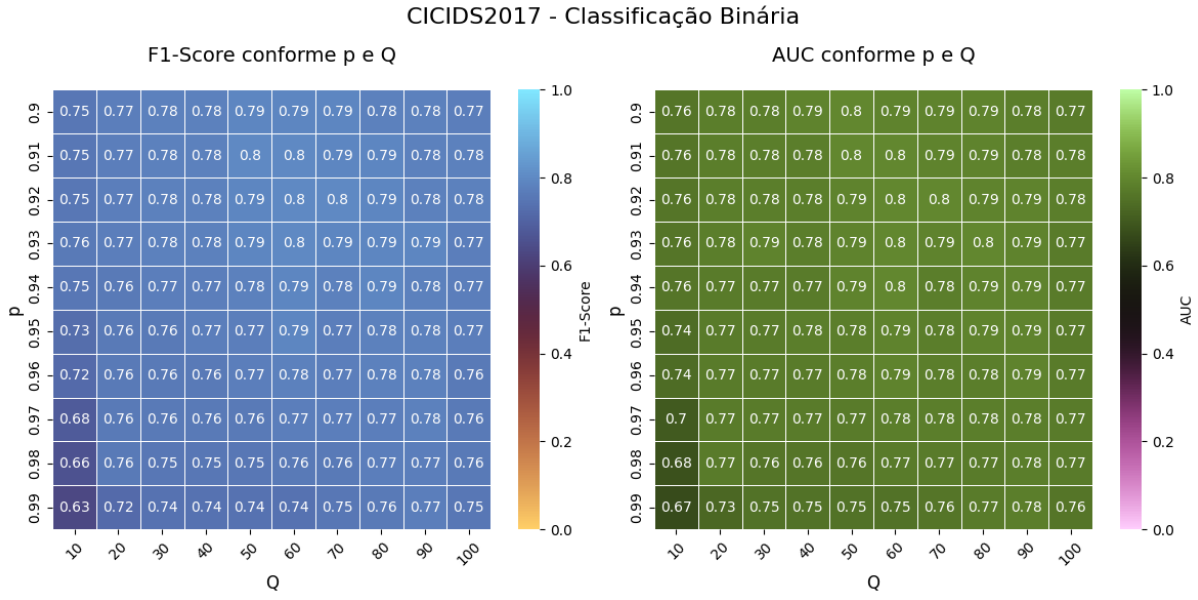
Nessa seção, estão dispostas as tabelas com as pontuações de F1-Score e AUC obtidas pelo EFC nos experimentos correspondentes à seção 4.1.1. Serão discutidos primeiramente os resultados obtidos para o conjunto de dados CICIDS2017, então seguidos pelos resultados para o *dataset* CICDDoS2019. Por fim, serão abordados os resultados dos testes sobre os pesos de pseudocontagens, primeiro sendo abordado o experimento em conjunto com os níveis de discretização, que é então seguido pelas classificações *inter-dataset*.

### 5.1.1 CICIDS2017

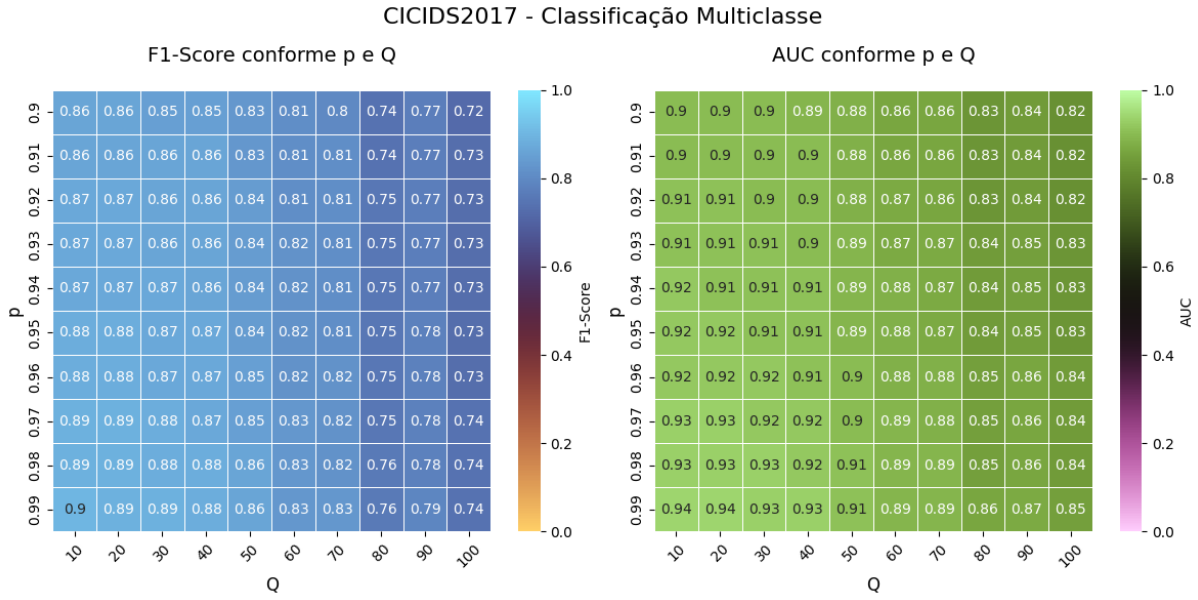
As Figuras 5.1a e 5.1b apresentam, respectivamente, os resultados dos experimentos de classificação binária e multiclasse com o conjunto de dados CICIDS2017. Os valores das métricas AUC e F1-Score são exibidos em forma de mapas de calor (*heatmaps*), construídos a partir de diferentes combinações entre os níveis de discretização ($Q$) e os limiares de classificação ($p$). Em cada imagem, o lado esquerdo mostra os resultados de F1-Score, nas cores azul e laranja, enquanto o lado direito exibe os resultados de AUC, representados em verde e rosa.

Figura 5.1: Resultados das classificações binária e multiclasse para o *dataset* CICIDS2017, com diferentes valores para os níveis de discretização $Q$ e diferentes limiares de classificação ($p$), medidos em AUC e F1-Score

(a) Resultados para a classificação binária




(b) Resultados para a classificação multiclasse



Nestes resultados, é possível notar que em cada tipo de classificação o EFC exibiu comportamentos distintos durante a variação dos valores para os hiperparâmetros. Na classificação binária, o modelo foi capaz de obter um melhor desempenho para ambas as métricas quando configurado com o número de níveis de discretização na faixa de 50 a 70 e pareado com o limiar de classificação definido entre 0.91 e 0.93, atingindo uma pontuação máxima de 0.80 em ambas as métricas. Ademais, foi notado que o uso de maiores quantidades de níveis de discretização não necessariamente resulta em uma melhora no desempenho da classificação, enquanto aumenta muito o custo computacional em termos de uso de memória.

Já na classificação multiclasse, observa-se que o modelo alcançou sua melhor pontuação quando configurado com 10 níveis de discretização e com o limiar de discretização definido em 0,99, obtendo 0,90 para F1-Score e 0,94 para AUC. Ainda, nota-se que o modelo teve uma piora considerável nos resultados conforme o aumento de níveis de discretização, assim como nota-se que os resultados melhoram conforme o valor atribuído ao limiar de classificação se aproxima de 1.

### 5.1.2 CICDDoS2019

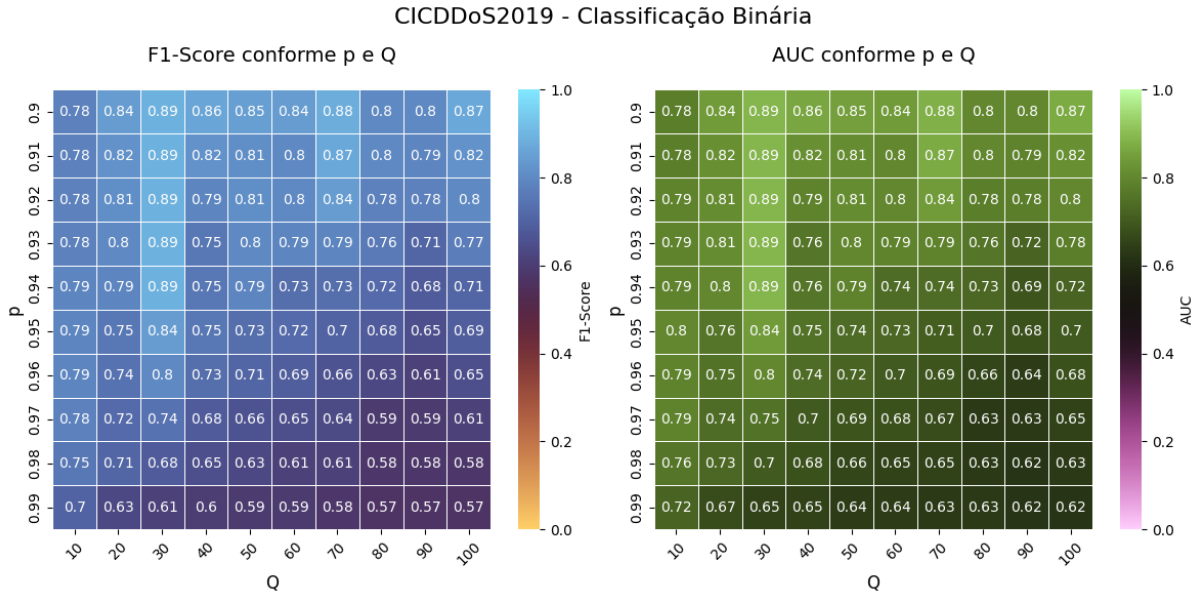
De forma similar ao experimento anterior, as Figuras 5.2a e 5.2b apresentam, respectivamente, os resultados dos experimentos de classificação binária e multiclasse realizados com o conjunto de dados CICDDoS2019 para treino e teste do modelo. Nelas, os resultados também são exibidos em mapas de calor (*heatmaps*), construídos a partir de diferentes combinações entre os níveis de discretização ($Q$) e os limiares de classificação ($p$). O lado esquerdo de cada figura mostra os valores de F1-Score, nas cores azul e laranja, enquanto o lado direito apresenta os resultados de AUC, em verde.

Nos resultados para este *dataset* observa-se um padrão semelhante ao anterior. Para a classificação binária, o melhor desempenho foi obtido na configuração com 30 níveis de discretização para os dados e com o limiar de discretização na faixa de 0.90 a 0.94, que resultou em uma pontuação de 0.89 em ambos F1-Score e AUC. Novamente, níveis mais altos de discretização dos dados não necessariamente melhoraram o desempenho do classificador.

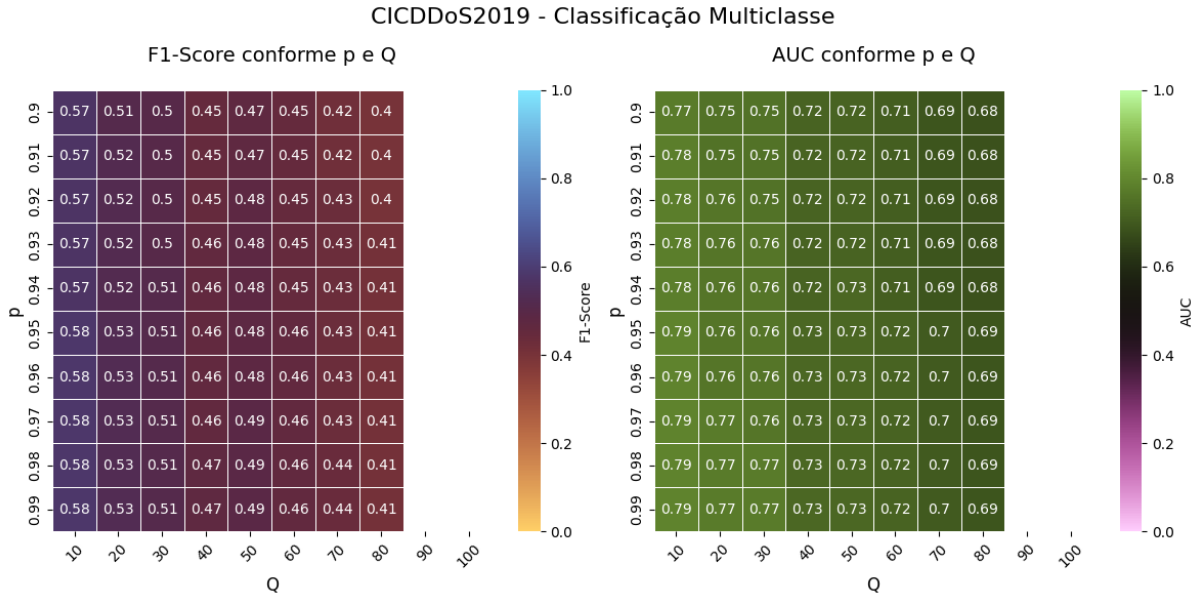
Já para a classificação multiclasse, o modelo teve um desempenho consideravelmente inferior ao que foi visto no *dataset* anterior. Isso pode ser causado pelo fator de o conjunto de dados possuir principalmente fluxos de ataques de negação de serviço, que comumente possuem atributos similares entre si, do ponto de vista da rede, como grande volume de pacotes de mesmo tamanho vindos de apenas um dos lados do fluxo. Ainda assim, o melhor desempenho do modelo foi obtido sob as mesmas configurações vistas no experimento anterior. Quando configurado com apenas 10 níveis de discretização para os dados e com

Figura 5.2: Resultados das classificações binária e multiclasse para o *dataset* CICDDoS2019, com diferentes valores para os níveis de discretização $Q$ e diferentes limiares de classificação ($p$), medidos em AUC e F1-Score

(a) Resultados para a classificação binária




(b) Resultados para a classificação multiclasse



o limiar de classificação definido para 0.99 das energias dos fluxos de treino, configuração essa na qual o modelo pontuou 0.58 de F1-score e 0.76 de AUC.

É importante explicar que a ausência de resultados nas configurações $Q = 90$ e $Q = 100$ na Figura 5.2b se deve ao fato de que não foi possível realizar a classificação multiclasse com 90 e 100 níveis de discretização. Isso ocorreu devido ao alto custo computacional de memória do treinamento do modelo com essa configuração, o que ocasionou a interrupção do programa por parte do sistema operacional em todas as tentativas de realizar a classificação.

## 5.2 Experimentos com pesos de pseudocontagens

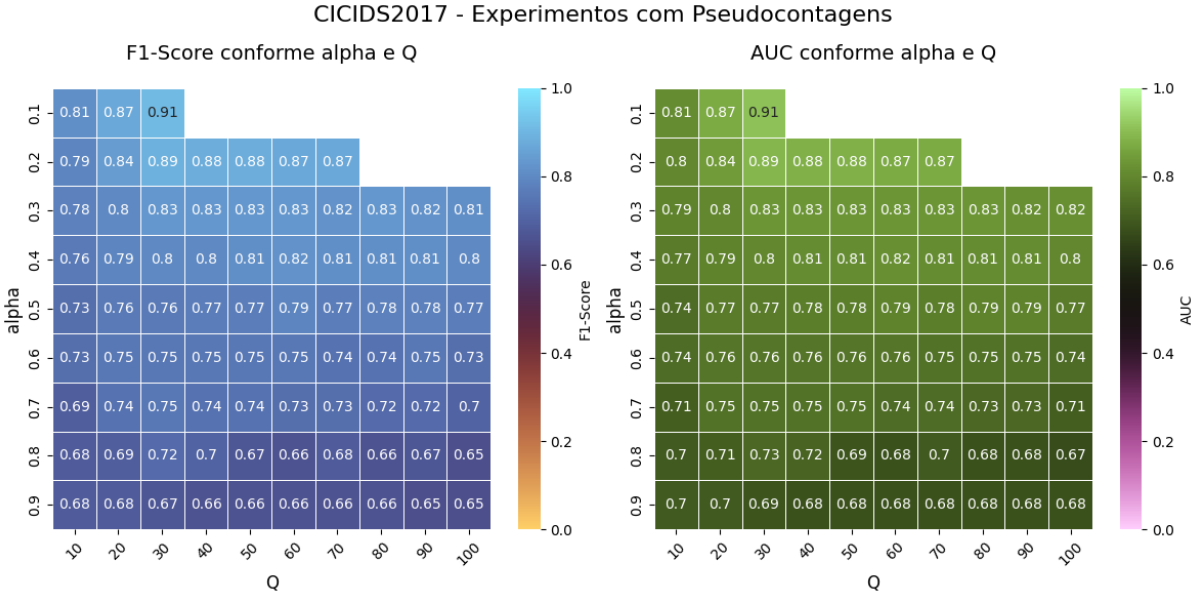
Para os experimentos relativos aos pesos de pseudocontagens ($\alpha$), foram obtidos os dados presentes nas Figuras 5.3a e 5.3b e na Tabela 5.1. Nas Figuras 5.3a e 5.3b, são apresentados os resultados dos experimentos que avaliam os pesos de pseudocontagens juntamente a diferentes níveis de discretização dos dados. Tal como nos experimentos anteriores, os resultados são exibidos em forma de mapas de calor, desta vez em função de $\alpha$ e $Q$, mostrando a variação do desempenho do classificador em termos das métricas AUC (à direita, em verde e rosa) e F1-Score (à esquerda, em azul e laranja). Já a Tabela 5.1 sintetiza os resultados dos experimentos interdataset, permitindo observar o impacto da calibragem de $\alpha$ na capacidade de adaptabilidade do modelo entre diferentes conjuntos de dados.

Analisando os resultados obtidos nas Figuras 5.3a e 5.3b, nota-se que para ambos os conjuntos de dados, o melhor desempenho foi obtido quando o modelo foi configurado com 30 níveis de discretização para os dados e com o peso de 0.10 para $\alpha$, obtendo a pontuação 0.91 para ambos F1-Score e AUC no *dataset* CICIDS2017 e 0.95 para as mesmas métricas no *dataset* CICDDoS2019. Este segundo conjunto, contudo, também foi obtida a mesma pontuação quando calibrado com os níveis de discretização definidos em 40 e 50, mas ainda com 0.10 nos pesos para pseudocontagens. Observa-se, também, que consistentemente, menores pesos de pseudocontagens ocasionaram melhores resultados na classificação. Contudo, também nota-se que os menores pesos ocasionaram a impossibilidade de treinamento do modelo para níveis de discretização maiores que 30 no conjunto CICIDS2017 e maiores que 50 no CICDDoS2019, tal como foi suposto na subseção 4.1.2.

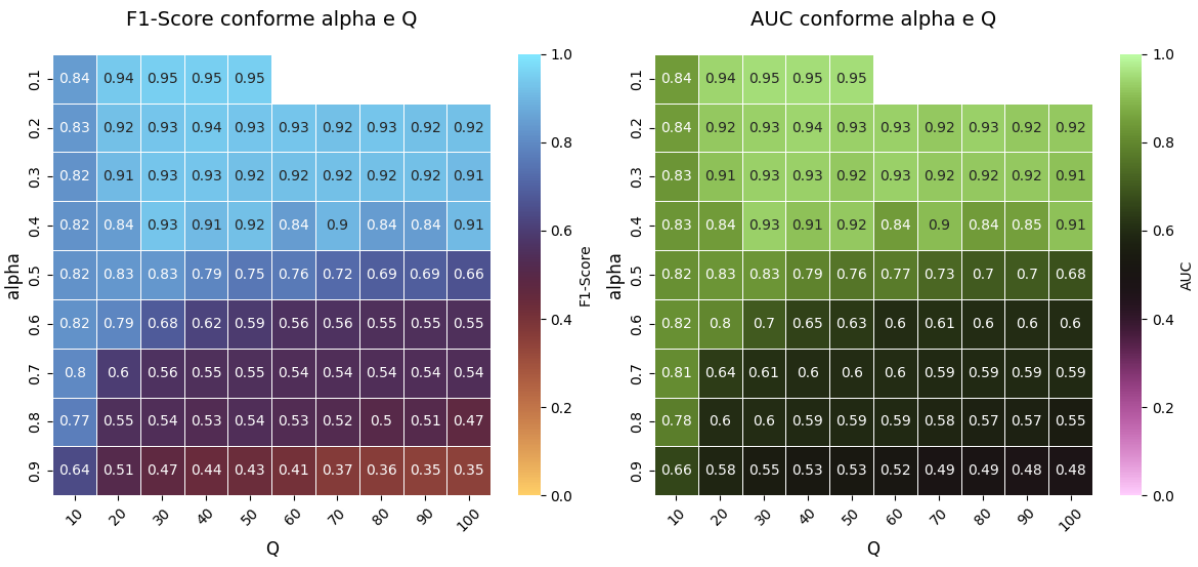
Já na Tabela 5.1, são apresentados os resultados dos experimentos *inter-dataset*, avaliados pelas métricas F1-Score e AUC. Complementando os dados da tabela, as Figuras 5.4a, 5.4b e 5.4c ilustram a curva ROC, a matriz de confusão e o histograma das energias dos fluxos classificados para cada configuração empregada no teste cruzado entre datasets, no qual o modelo foi treinado com o conjunto de dados CICIDS2017 e avaliado

Figura 5.3: Resultados da classificação binária para os *datasets* CICIDS2017 e CICDDoS2019, para diferentes valores para os níveis de discretização $Q$ e diferentes pesos para as pseudocontagens de frequências $\alpha$ (alpha), medidos em AUC e F1-Score

(a) Resultados para o *dataset* CICIDS2017




(b) Resultados para o *dataset* CICDDoS2019



no conjunto CICDDoS2019. De forma análoga, as Figuras 5.5a, 5.5b e 5.5c apresentam essas mesmas informações para o caso em que o modelo foi treinado com o conjunto de dados CICDDoS2019 e testado com o conjunto CICIDS2017.

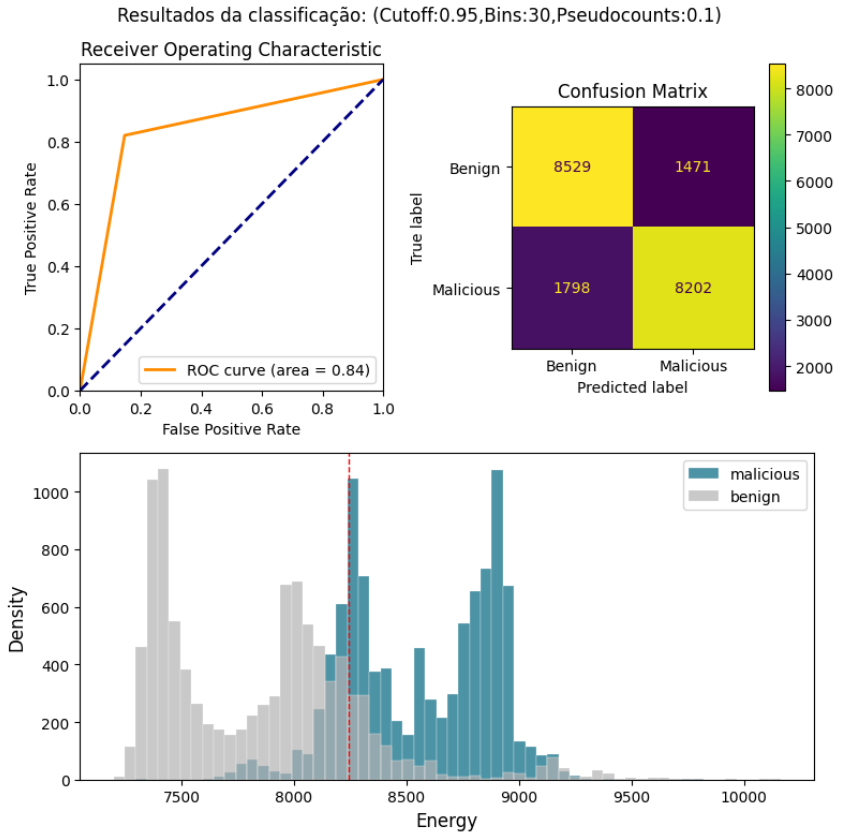
Nas imagens, é possível observar os histogramas montados para as diferentes configurações testadas, onde também nota-se o limiar de classificação(linha vermelha no histograma) separando os níveis de energia classificados como benignos à esquerda e como maliciosos à direita. Como é possível observar ao analisar em conjunto a Tabela 5.1 e as Figuras 5.4a, 5.4b e 5.4c, o modelo alcançou um desempenho de 0,84 em ambas as métricas, F1-Score e AUC, quando configurado com o valor de 0,1 para os pesos de pseudocontagens. Além disso, é possível verificar visualmente que o limiar de classificação encontra-se no vale entre as classes *benign* (do inglês, benigno) e *malicious* (do inglês, malicioso), o que indica que o modelo conseguiu inferir com precisão o comportamento da classe *benign*. À medida que o valor dos pesos de pseudocontagens é aumentado, contudo, o modelo perde a capacidade de diferenciação entre as duas classes e o histograma das duas classes fica mais sobreposto, como resultado. Assim, o pior desempenho foi observado ao ter o $\alpha$ definido em 0.9, onde o modelo obteve 0.33 de F1-Score e 0.48 de AUC, aparentando confirmar a suposição feita de que valores mais altos para este parâmetro prejudicam o desempenho do modelo.

Ademais, ao analisar as Figuras 5.5a, 5.5b e 5.5c em conjunção com a Tabela 5.1, observa-se que o comportamento do modelo foi semelhante ao observado anteriormente, porém com maior dificuldade em separar as classes *benign* e *malicious*. Essa dificuldade pode ser observada ao comparar o histograma correspondente a $\alpha = 0.1$ das duas , onde as distribuições das duas classes apresentam maior sobreposição. Apesar disso, o modelo ainda demonstra uma capacidade razoável de distinção entre as classes para $\alpha = 0.1$, com F1-Score de 0.69 e AUC de 0.71. No entanto, à medida que o valor de $\alpha$ aumenta, culminando em 0.9, o desempenho do modelo sofre uma deterioração significativa. Nesse cenário, a pontuação de F1-Score cai para 0.58, enquanto o valor de AUC reduz-se para 0.60, indicando uma redução expressiva na capacidade do modelo de separar corretamente as classes, que pode ser confirmada visualmente pelos histogramas.

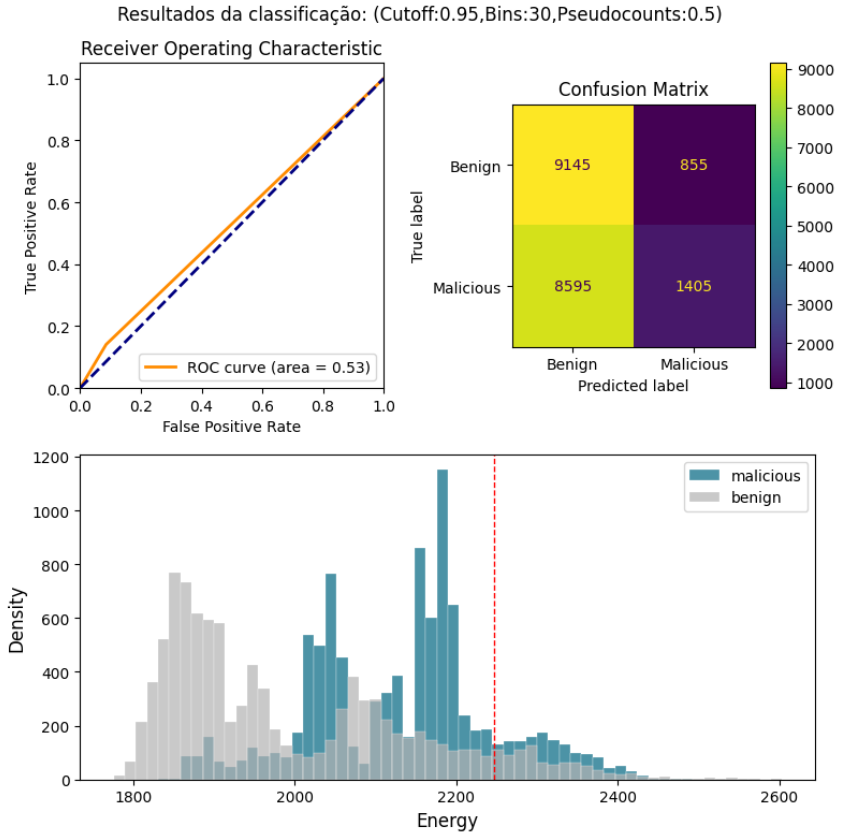
## 5.3 Discussão dos resultados e proposição de abordagem para calibragem dos hiperparâmetros

A partir dos resultados obtidos, é possível traçar alguns efeitos resultantes das variações nos hiperparâmetros do modelo. Conforme os dados sugerem, verifica-se que o uso de baixos valores para os pesos de pseudocontagens é recomendado, sendo preferível adotar os menores valores possíveis que ainda mantenham o modelo funcional. É visto nos

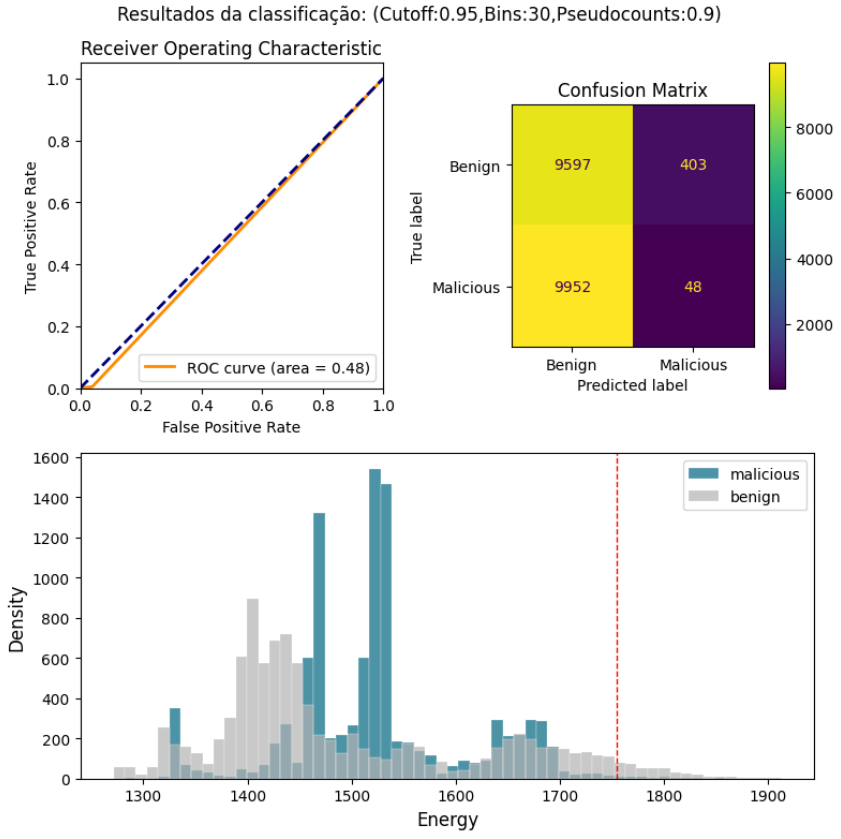
Figura 5.4: Curva ROC, matriz de confusão e histograma das energias dos fluxos classificados para os classificadores treinados com o conjunto de dados CICIDS2017 e testados com o conjunto CICDDoS2019 com diferentes valores de $\alpha$.




(a) $\alpha = 0.1$

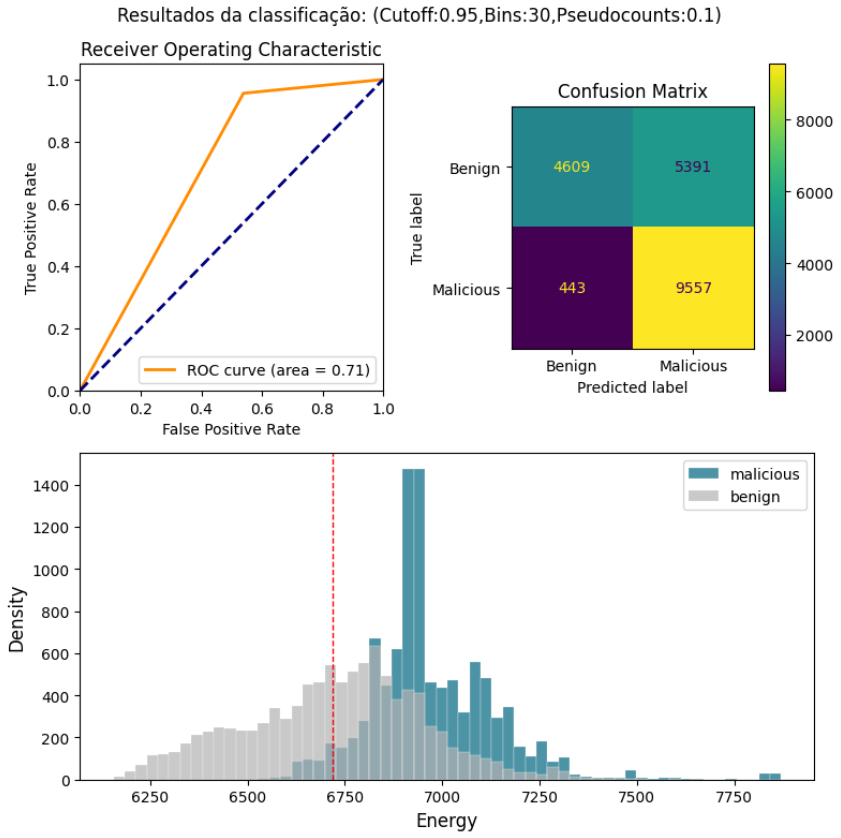



(b) $\alpha = 0.5$

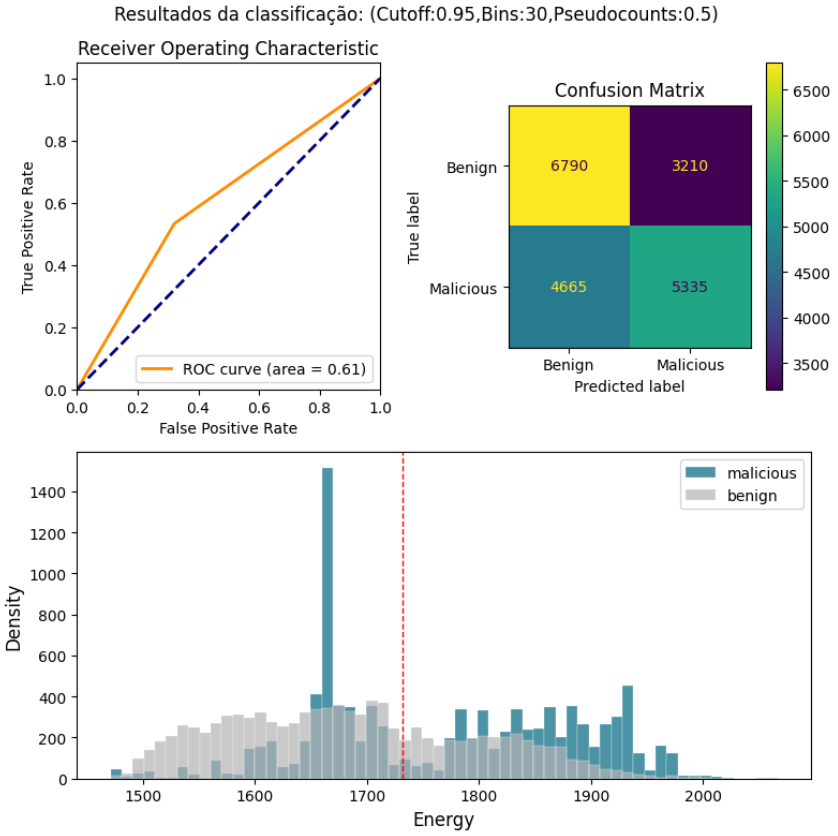



(c) $\alpha = 0.9$

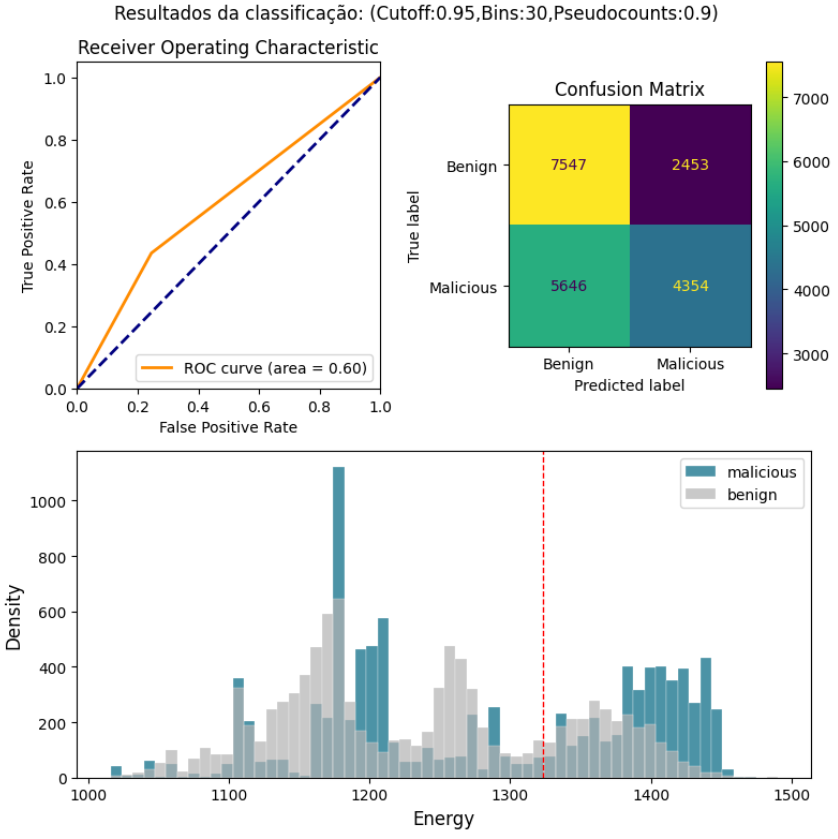
Figura 5.5: Curva ROC, matriz de confusão e histograma das energias dos fluxos classificados para os classificadores treinados com o conjunto de dados CICDDoS2019 e testados com o conjunto CICIDS2017 com diferentes valores de $\alpha$.




(a) $\alpha = 0.1$




(b) $\alpha = 0.5$




(c) $\alpha = 0.9$

Tabela 5.1: Resultados da classificação binária para os testes cruzados entre os datasets CICIDS2017 e CICDDoS2019, com diferentes pesos para a as pseudocontagens de frequências. Desempenho medido em AUC e F1-Score.

| | | Train CICIDS2017 / Test CICDDoS2019 | | Train CICDDoS2019 / Test CICIDS2017 | |
|---|---|---|---|---|---|
| Q | $\alpha$ | F1 Score | AUC | F1 Score | AUC |
| 30 | 0.1 | **0.84** | **0.84** | **0.69** | **0.71** |
| | 0.5 | 0.44 | 0.53 | 0.60 | 0.61 |
| | 0.9 | 0.33 | 0.48 | 0.58 | 0.60 |

resultados experimentais que valores reduzidos de pseudocontagens aparentam resultar em um controle mais refinado da discriminação entre as classes, resultando em melhor desempenho nas métricas F1-Score e AUC.

Uma vez que o valor de pseudocontagens seja suficientemente baixo para permitir a discretização dos dados em até 30 níveis, recomenda-se discretizar os dados em 20 ou 30 níveis, conforme o cenário. Essa abordagem equilibra uma granularidade apropriada e o custo computacional do modelo, ao mesmo tempo que possibilita ajustes do limiar de classificação de acordo com o tipo de tarefa. Para classificações binárias, valores de pseudocontagens entre 0.90 e 0.95 se mostraram mais adequados, enquanto para classificações multiclasse, é preferível utilizar valores próximos de 0.99, uma vez que essa configuração reduz a ocorrência da classe anômala.

Por fim, recomenda-se evitar níveis altos de discretização dos dados, já que essa abordagem aumenta significativamente o custo computacional, podendo até causar falhas no modelo devido a limitações de memória. Além disso, altos níveis de discretização não demonstraram melhorias substanciais no desempenho, reforçando a importância de um equilíbrio entre granularidade e eficiência computacional na escolha dos parâmetros. Adicionalmente, foi identificada uma piora no desempenho do modelo para o *dataset* CICDDoS2019 em comparação ao CICIDS2017 em classificações multiclasse. Esse fenômeno pode ser atribuído à similaridade entre os ataques do tipo DDoS presentes no CICDDoS2019, que frequentemente exibem comportamentos semelhantes quando analisados sob a perspectiva de fluxos de rede. Essa similaridade provavelmente dificulta a distinção das classes pelo modelo, resultando na redução de desempenho.

Neste capítulo, apresentaram-se de forma comparativa os resultados obtidos nos experimentos de variação conjunta dos hiperparâmetros $Q$ e $p$, bem como de $\alpha$ e $Q$, aplicados a ambos os *datasets*. Analisou-se o efeito de cada hiperparâmetro sobre as métricas AUC-ROC e F1-Score, além de investigar se configurações com valores reduzidos para os pesos de pseudocontagens ocasionavam a perda da capacidade do modelo de adaptação a

diferentes domínios de dados. Com base nessas evidências, propôs-se uma estratégia prática de calibragem que busca equilibrar desempenho e custo computacional. As Tabelas 5.3, 5.2 e 5.4 apresentam as melhores configurações avaliadas em termos de F1-Score e AUC. No capítulo seguinte, será apresentada a conclusão deste estudo, com a síntese dos principais resultados e sugestões para trabalhos futuros.

Tabela 5.2: Tabela com as melhores configurações encontradas nos experimentos com $Q$ e $p$ no *dataset* CICIDS2017 e sua pontuação em F1-Score e AUC

| | CICIDS2017 | | | |
|---|---|---|---|---|
| | $Q$ | $p$ | F1-score | AUC |
| Classificação Binária | 50 | 0.91 | 0.80 | 0.80 |
| | 60 | 0.91-0.93 | 0.80 | 0.80 |
| | 70 | 0.91 | 0.80 | 0.80 |
| Classificação Multiclasse | 10 | 0.99 | 0.90 | 0.94 |

Tabela 5.3: Tabela com as melhores configurações encontradas nos experimentos com $Q$ e $p$ no *dataset* CICDDoS2019 e sua pontuação em F1-Score e AUC

| | CICDDoS2019 | | | |
|---|---|---|---|---|
| | $Q$ | $p$ | F1-score | AUC |
| Classificação Binária | 30 | 0.90-0.94 | 0.89 | 0.89 |
| Classificação Multiclasse | 10 | 0.95-0.99 | 0.58 | 0.79 |

Tabela 5.4: Tabela com as melhores configurações encontradas nos experimentos com pseudocontagens e sua pontuação em F1-Score e AUC

| | Pseudocontagens | | | |
|---|---|---|---|---|
| | $Q$ | $\alpha$ | F1-score | AUC |
| CICIDS2017 | 30 | 0.1 | 0.91 | 0.91 |
| CICDDoS2019 | 30-50 | 0.1 | 0.95 | 0.95 |

# Capítulo 6

# Conclusão

Neste trabalho, foi realizado um estudo de abordagens para a calibração dos hiperparâmetros do Energy-based Flow Classifier (EFC), um modelo de classificação baseado em estatística inversa elaborado para detecção de anomalias em fluxos de redes. Para determinar uma abordagem para a calibragem dos hiperparâmetros, foram realizados experimentos de classificação binária e multiclasse com diferentes configurações, a fim de determinar uma possível abordagem de configuração para o modelo de classificação de forma a obter melhores resultados em termos de F1-Score e AUC.

Os resultados obtidos sugerem que o valor estabelecido para $\alpha$, referente ao hiperparâmetro que estabelece o peso das operações de pseudocontagens de frequências, impacta diretamente na capacidade de inferência do modelo. Foi observado que quanto menores os valores para $\alpha$, especialmente para ($\alpha = 0.1$), a capacidade de inferência do comportamento das classes foi aguçada, aparentemente sem causar perda da capacidade de adaptação a diferentes domínios de dados. Contudo, descobriu-se que usar valores baixos ($\alpha < 0.3$) impede que o modelo seja treinado com níveis mais altos de discretização dos dados. Ademais, notou-se que a escolha de níveis mais altos para a discretização dos dados ($Q > 50$) aparenta não resultar necessariamente em melhores resultados, além de impedir o modelo de ser executado em alguns sistemas, devido a um maior uso de memória e processamento, como foi o caso observado neste estudo. Ainda, notou-se que a configuração otimizada para o limiar de classificação depende do contexto da classificação. Os resultados da seção anterior sugerem que, para classificações binárias, o limiar estabelecido na faixa $0.90 \leq p \leq 0.95$ pode gerar melhores resultados, enquanto para classificações multiclasse, o valor de limiar estabelecido na faixa $0.95 \leq p \leq 0.99$ pode gerar um melhor desempenho.

Cabe destacar que, como a execução dos experimentos deste trabalho foi realizada em ambiente remoto, não foi possível aplicar validação cruzada. Isso porque o elevado tempo de processamento necessário para tal procedimento resultaria em desconexões por ociosidade do servidor. Ademais, a limitação dos recursos de *hardware* do ambiente remoto

impossibilitou a obtenção de resultados para níveis mais elevados de discretização, uma vez que tais experimentos demandavam quantidade de memória superior à disponibilizada pelo Google Colab.Com isso em mente, sugere-se, como direcionamento para trabalhos futuros envolvendo o EFC, a realização dos experimentos utilizando validação cruzada.

Além disso, a fim de evitar possíveis vieses decorrentes da inclusão dos dados de teste no processo de balanceamento, recomenda-se que, em estudos posteriores, os dados empregados no balanceamento sejam utilizados apenas para a etapa de validação do modelo, reservando-se para o teste uma porção distinta dos conjuntos de dados. Além disso, sugere-se a possibilidade de testar diferentes métodos de discretização, uma vez que este passo é necessário para a execução do modelo e pode afetar a representatividade dos dados. De forma similar, pode ser abordado em algum trabalho futuro a elaboração de métodos para estimar um nível ótimo de discretização para os dados. Outro possível aprimoramento do modelo, voltado à sua principal aplicação — a detecção de fluxos maliciosos de rede — consiste na implementação da classificação multirrótulo (*multilabel*). Tal extensão permitiria ao EFC, quando treinado para identificar múltiplas classes, isolar e analisar fluxos suspeitos de forma mais detalhada.

# Referências


[1] *2025 cyber threat report: The escalating risks every CISO must prepare for*. https://www.radware.com/blog/threat-intelligence/2025-cyber-threat-report/, acesso em 2025-05-14. 1

[2] Mustapha, Hanan e Ahmed M Alghamdi: *DDoS attacks on the internet of things and their prevention methods*. Em *Proceedings of the 2nd International Conference on Future Networks and Distributed Systems*, páginas 1–5. ACM, ISBN 9781450364287. https://dl.acm.org/doi/10.1145/3231053.3231057, acesso em 2025-05-14. 1

[3] Tidjon, Lionel N., Marc Frappier e Amel Mammar: *Intrusion detection systems: A cross-domain overview*. 21(4):3639–3681, ISSN 1553-877X, 2373-745X. https://ieeexplore.ieee.org/document/8735821/, acesso em 2025-05-16. 1

[4] Souza, Manuela M.C., Camila T. Pontes, João J.C. Gondim, Luís P.F. Garcia, Luiz DaSilva, Eduardo F.M. Cavalcante e Marcelo A. Marotta: *A novel open set energy-based flow classifier for network intrusion detection*. 157:104569, ISSN 01674048. https://linkinghub.elsevier.com/retrieve/pii/S0167404825002585, acesso em 2025-07-01. 1, 2, 5, 11, 17, 20, 23

[5] Ahmad, Atif, Sean B. Maynard, Kevin C. Desouza, James Kotsias, Monica T. Whitty e Richard L. Baskerville: *How can organizations develop situation awareness for incident response: A case study of management practice*. 101:102122, ISSN 0167-4048. https://www.sciencedirect.com/science/article/pii/S0167404820303953, acesso em 2025-05-16. 1

[6] Ring, Markus, Sarah Wunderlich, Deniz Scheuring, Dieter Landes e Andreas Hotho: *A survey of network-based intrusion detection data sets*. 86:147–167, ISSN 01674048. https://linkinghub.elsevier.com/retrieve/pii/S016740481930118X, acesso em 2025-05-16. 2

[7] Sperotto, Anna, Gregor Schaffrath, Ramin Sadre, Cristian Morariu, Aiko Pras e Burkhard Stiller: *An overview of IP flow-based intrusion detection*. 12(3):343–356, ISSN 1553-877X, 2373-745X. https://ieeexplore.ieee.org/document/5455789/, acesso em 2025-05-16. 2

[8] Pontes, Camila F. T., Manuela M. C. De Souza, Joao J. C. Gondim, Matt Bishop e Marcelo Antonio Marotta: *A new method for flow-based network intrusion detection using the inverse potts model*. 18(2):1125–1136, ISSN 1932-4537, 2373-7379. https://ieeexplore.ieee.org/document/9415676/, acesso em 2025-05-14. 2, 5, 16, 26

[9] Lopes, Daniele A. G., Marcelo A. Marotta, Marcelo Ladeira e Joao J. C. Gondim: *Botnet detection based on network flow analysis using inverse statistics*. Em *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*, páginas 1–6. IEEE, ISBN 9789893334362. https://ieeexplore.ieee.org/document/9820318/, acesso em 2025-05-14. 2, 17

[10] DeAlmeida, Jonathan M., Camila F. T. Pontes, Luiz A. DaSilva, Cristiano B. Both, Joao J. C. Gondim, Celia G. Ralha e Marcelo A. Marotta: *Abnormal behavior detection based on traffic pattern categorization in mobile networks*. 18(4):4213–4224, ISSN 1932-4537, 2373-7379. https://ieeexplore.ieee.org/document/9600445/, acesso em 2025-05-14. 2, 18

[11] Carvalho Bertoli, Gustavo de, Lourenço Alves Pereira Junior, Osamu Saotome e Aldri Luiz dos Santos: *Generalizing intrusion detection for heterogeneous networks: A stacked-unsupervised federated learning approach*. 127:103106, ISSN 0167-4048. https://www.sciencedirect.com/science/article/pii/S0167404823000160, acesso em 2025-06-06. 2, 18

[12] Zhu, Yuedi, Chao Li e Yong Wang: *Network intrusion detection scheme based on federated learning in heterogeneous network environments*. Em *2024 13th International Conference on Communications, Circuits and Systems (ICCCAS)*, páginas 491–496. IEEE, ISBN 9798350386271. https://ieeexplore.ieee.org/document/10652649/, acesso em 2025-06-08. 2, 18

[13] Wan, Mengke, Jiang Fang, Chen Guo, Liru Geng, Yinlong Liu, Wei Ma, Chao Xu e Mohan Su: *A federated learning-based intrusion detection system for satellite-terrestrial integrated networks*. Em *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, páginas 1–5. IEEE, ISBN 9798350368741. https://ieeexplore.ieee.org/document/10890330/, acesso em 2025-06-04. 2, 19

[14] Probst, Philipp, Anne Laure Boulesteix e Bernd Bischl: *Tunability: Importance of hyperparameters of machine learning algorithms*. 20(53):1–32, ISSN 1533-7928. http://jmlr.org/papers/v20/18-444.html, acesso em 2025-05-19. 3

[15] Zhang, Zhao, Yong Zhang, Da Guo e Mei Song: *A scalable network intrusion detection system towards detecting, discovering, and learning unknown attacks*. 12(6):1649–1665, ISSN 1868-808X. https://doi.org/10.1007/s13042-020-01264-7, acesso em 2025-06-07. 17

[16] Nguyen, Loc Gia e Kohei Watabe: *A method for network intrusion detection using flow sequence and BERT framework*. Em *ICC 2023 - IEEE International Conference on Communications*, páginas 3006–3011. IEEE, ISBN 9781538674628. https://ieeexplore.ieee.org/document/10279335/, acesso em 2025-06-10. 19

[17] De Melo, Leonardo Henrique, Gustavo De Carvalho Bertoli, Michele Nogueira, Aldri Luiz Dos Santos e Lourenço Alves Pereira: *Anomaly-flow: A multi-domain federated generative adversarial network for distributed denial-of-service detection*. páginas 1–1, ISSN 0890-8044, 1558-156X. https://ieeexplore.ieee.org/document/10988602/, acesso em 2025-06-10. 20

[18] De Melo, Leonardo H, Gustavo De C Bertoli, Lourenco A Pereira, Osamu Saotome, Marcelo F Domingues e Aldri Luiz Dos Santos: *Generalizing flow classification for distributed denial-of-service over different networks.* Em *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, páginas 879–884. IEEE, ISBN 9781665435406. https://ieeexplore.ieee.org/document/10001530/, acesso em 2025-06-10. 20

[19] *GitHub - EnergyBasedFlowClassifier/EFC-package: For the full documentation, see https://efc-package.readthedocs.io/en/latest/.* https://github.com/EnergyBasedFlowClassifier/EFC-package, acesso em 2025-06-11. 21, 23

[20] Sharafaldin, Iman, Arash Habibi Lashkari, Ali A Ghorbani *et al.*: *Toward generating a new intrusion detection dataset and intrusion traffic characterization.* ICISSp, 1(2018):108–116, 2018. 26

[21] Sharafaldin, Iman, Arash Habibi Lashkari, Saqib Hakak e Ali A. Ghorbani: *Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy.* Em *2019 International Carnahan Conference on Security Technology (ICCST)*, páginas 1–8. https://ieeexplore.ieee.org/abstract/document/8888419, acesso em 2025-06-13, ISSN: 2153-0742. 26, 28