Resumo de Grandes Volumes de Dados com Filtro de Bloom: Uma Abordagem Eficiente para Aprendizado Profundo com Redes Neurais Convolucionais em Fluxos de Rede

Martin Andreoni Lopez[1], **Diogo M. F. Mattos**[2]

[1] Technology Innovation Institute (TII)
Abu Dhabi – Emirados Árabes Unidos

[2]LabGen/MídiaCom – TET/PPGEET/UFF
Universidade Federal Fluminense (UFF)
Niterói, RJ – Brasil

Abstract. *This paper proposes using Bloom filters to generate summaries of two-dimensional data from flows in a network usage window, forming a bitmap. After generating the summaries, the paper applies deep learning, composed of layers of a convolutional neural network to segment the bitmap. Bitmap segmentation is a computer vision task that convolutional neural networks efficiently provide. The main contributions of the paper are three-fold: (i) the proposal of a two-dimensional data summary technique in a flow window through Bloom filters; (ii) the deployment of deep learning with convolutional neural networks over network flows; and (iii) the optimized execution of our proposal in graphic processing units (GPU). We evaluate the proposal on a real dataset from a broadband access provider, and the results demonstrate the efficiency of the filters and the high precision on the incremental deep learning deployment.*

Resumo. *Este artigo propõe a aplicação de filtros de Bloom para a geração de resumos de dados bidimensionais a partir de fluxos em uma janela de uso da rede formando um mapa de bits. Após a geração dos resumos, o artigo aplica o aprendizado profundo, composto por camadas de rede neural convolucional, para a segmentação do mapa de bits. A segmentação do mapa de bits é uma tarefa da visão computacional que é eficientemente provida por redes neurais convolucionais. As principais contribuições do artigo são (i) a proposta de uma técnica de resumo bidimensional de dados em uma janela de fluxos através de filtros de Bloom; (ii) a aplicação do aprendizado profundo com redes neurais convolucionais em fluxos de redes e (iii) a execução otimizada da proposta em unidades de processamento gráfico (GPU). A proposta é avaliada sobre um conjunto de dados real de um provedor de acesso de banda larga e os resultados demonstram a eficiência dos filtros usados e a precisão superior a 0,90 do aprendizado profundo com treinamento incremental.*

1. Introdução

O aumento do tráfego de Internet em decorrência de ações de afastamento social em resposta à pandemia da Doença Infecciosa de Coronavírus 2019 (COVID-19) desencadeou uma epidemia de ataques virtuais. O relatório recente do *Cyber Peace Institute* aponta que

houve um aumento de 45% dos ataques contra instituições de saúde, enquanto houve aumento de 22% em outras áreas, no período entre novembro de 2020 e janeiro de 2021[1]. Esses ataques são motivados por questões financeiras ou políticas e variam desde sequestro de dados (*ransomware*) até a divulgação de informações falsas [de Oliveira et al., 2021] e ciberespionagem relacionadas à COVID-19. O foco nas organizações de saúde advém de deterem dados valiosos, como registros médicos de pacientes, além do posicionamento estratégico da saúde durante a pandemia de COVID-19. Contudo, o cenário da cibersegurança carece de profissionais habilitados e alinhados às necessidades urgentes dos mais diversos setores da sociedade. A consultoria internacional *Stott and May* indica que mais de 76% dos profissionais acreditam que haja carência de pessoal e de habilidades de cibersegurança em suas companhias[2]. Ademais, os profissionais relatam o uso de ambientes complexos para a proteção do ciberespaço, já que 78% emprega 50 ou mais ferramentas de proteção e 37%, mais de 100 ferramentas. Nesse cenário, aproximadamente metade dos profissionais responsáveis pela segurança dos sistemas computacionais acreditam que ferramentas baseadas em aprendizado de máquina são soluções para atender às necessidades imediatas da segurança da informação.

Aplicações de monitoramento e análise de tráfego de redes baseadas em mecanismos de aprendizado de máquina são desafiadoras, pois exigem o processamento rápido de grandes quantidades de dados heterogêneos [Casas et al., 2017]. Os dados de monitoramento de rede são fluxos de alta velocidade, que demandam processamento e análise rápidos e contínuos. Os fluxos de rede estão sujeitos a mudanças no comportamento estacionário dos dados, provocando mudanças de conceito e implicando a necessidade de adaptação dos modelos de análise [Andreoni Lopez et al., 2018]. Embora técnicas de aprendizado de máquina sejam adequadas para cenários intensivos em dados, essas técnicas incorrem em alta latência devido ao consumo de recursos computacionais no treinamento incremental e na classificação [Andreoni Lopez et al., 2019]. A alta latência de técnicas de aprendizado de máquina implica uma barreira para aplicação de classificação em tempo real. Paralelamente, aplicações de monitoramento e análise de tráfego carecem de avaliação, comparação e implantação precisas devido à escassez de conjuntos de dados adequados [Silva et al., 2020]. A avaliação de propostas sobre conjuntos de dados sintéticos tende a introduzir vieses na avaliação das propostas devido à alta correlação das características extraídas dos dados e a classe-alvo [Silva et al., 2020, Lobato et al., 2018]. Assim, aplicações de análise de tráfego baseadas em aprendizado de máquina devem considerar resumos e agregações de dados para diminuir a latência na classificação.

Este artigo propõe uma abordagem eficiente de Aprendizado Profundo (*Deep Learning*) com Redes Neurais Convolucionais aplicada a fluxos de redes através da geração de resumos de dados com Filtro de Bloom. A abordagem proposta visa executar a classificação de fluxos de redes em unidades de processamento gráfico (*Graphical Processing Unit* – GPU) para garantir a eficiência no uso do hardware. Para tanto, são empregadas redes neurais convolucionais otimizadas para a execução em GPU. As redes neurais convolucionais buscam padrões ocultos em matrizes bidimensionais de fluxos de rede. As matrizes de entrada das redes neurais convolucionais têm dimensões conhecidas *a priori*,

[1]Disponível em `https://cyberpeaceinstitute.org/publications/sar001-healthcare`.

[2]Disponível em `https://resources.stottandmay.com/hubfs/Research/Cyber\%20Security\%20in\%20Focus\%202020_web-2.pdf`.

pois são resumos de fluxos em uma janela saltitante (*hopping window*) gerados através de um filtro de Bloom. A proposta gera resumos com diferentes granularidades e agregações de dados. O treinamento do classificador é incremental, necessário para que a aplicação responda a mudanças de conceito nos dados de entrada.

As redes neurais convolucionais são normalmente aplicadas na segmentação de imagens [Sultana et al., 2020], enquanto aplicações de classificação de tráfego de rede recaem sobre classificadores lineares como máquina vetor de suporte [Reis et al., 2020] ou árvores de decisão [de Souza et al., 2020]. O principal desafio de aplicar aprendizado profundo em dados de rede é a representação dos fluxos através de matrizes multidimensionais adequadas à operação de convolução [Qin et al., 2019], levando a representações que consideram diretamente o conteúdo de cada pacote [Zhang et al., 2019]. Este artigo propõe a representação dos fluxos em uma janela através de uma forma compacta e bidimensional provida pelo Filtro de Bloom. A avaliação da proposta considera um conjunto de dados real baseado na coleta de dados da rede de acesso de banda larga fixa de uma grande operadora na cidade do Rio de Janeiro [Andreoni Lopez et al., 2019]. A avaliação da proposta demostra a eficiência da abordagem baseada em filtro de Bloom, com baixo tempo de treinamento e aumento na precisão na classificação de ameaças na rede. Ademais, a proposta executa diretamente em GPU, liberando recursos de computação para outras aplicações concorrentes.

O restante do artigo está organizado da seguinte forma. A Seção 2 apresenta o aprendizado profundo com redes neurais convolucionais. Os filtros de Bloom são contextualizados na Seção 3. A Seção 4 propõe a abordagem de aprendizado profundo baseado em filtro de Bloom. A proposta é avaliada e os resultados discutidos na Seção 5. A Seção 6 elenca os trabalhos relacionados. A Seção 7 conclui o artigo.

2. Redes Neurais Convolucionais

Redes neurais convolucionais (*Convolutional Neural Network* - CNN), em aprendizado de máquina, são um tipo de rede neural profunda (*Deep Neural Network* - DNN) amplamente usada para tarefas relacionadas à visão computacional. As redes neurais convolucionais são inspiradas pela organização hierárquica do córtex visual humano. Como nas demais redes neurais artificias, as redes convolucionais são compostas por neurônios artificiais que atuam como unidades básicas para aprender e extrair características da entrada, segmentação da entrada.

As redes convolucionais são compostas prioritariamente por camadas convolucionais, camadas de agrupamento e camadas densa. A **camada convolucional** define filtros que realizam o processo de convolução para transformar as imagens de entrada ou mapas de características da camada anterior nos mapas de características da saída. Os neurônios das camadas convolucionais aplicam filtros de convolução na imagem de entrada ou nas saídas da anterior camada e os resultados são considerados como características aprendidas e armazenadas em um mapa de características na saída da camada. A convolução discreta é definida como um operador linear que recebe duas funções de entrada e retorna uma terceira função formada pela soma do produto das funções de entrada ao longo de uma região subentendida pela superposição das funções de entrada em relação ao deslocamento existente entre elas. As camadas convolucionais aplicam os filtros de convolução em uma pequena região dos dados de entrada para compor uma característica específica

da componente, cor em uma imagem RGB, sendo tratada. Este processo é geralmente seguido por uma função de ativação que adiciona um comportamento não-linear à operação. Funções de ativação comuns são a função linear retificada (ReLu), a tangente hiperbólica, a sigmoide e a SoftMax. Os mapas de características após a função de ativação representam as características extraídas e são entradas para a próxima camada. Em camadas mais profundas, os neurônios tendem a extrair características de mais alto nível que convergem para a classificação final. A **camada de agrupamento ou camada de regularização** é aplicada após uma camada de convolução e executa a operação de redução de amostragem para reduzir a dimensão dos dados dos mapas de característica na entrada da próxima camada. Um exemplo de função de agrupamento é representar os valores de uma determinada janela de agrupamento pelo seu valor máximo (*max_pooling*). Assim, ao se considerar uma janela de agrupamento no formato 2×2, a função de agrupamento retorna o valor máximo entre os quatro valores analisados. A camada de agrupamento reduz a redundância dos dados. A **camada densa** tem cada um dos seus neurônios conectados a todos os neurônios da camada anterior. A camada densa realiza uma avaliação abrangente das características extraídas nas camadas convolucionais e geram um vetor de probabilidades N-dimensional, em que N é o número de características na saída da camada densa. Após a camada densa final, uma função de ativação, como a SoftMax ou a sigmoide, é usada para gerar a probabilidade de classificação final de cada amostra de dados.

O treinamento da rede neural convolucional é baseado na minimização da função de perda definida no modelo. O procedimento de treinamento para redes neurais convolucionais é semelhante ao das redes neurais artificiais convencionais com uso de retropropagação (*backward propagation*). A minimização da função de perda, representando o erro de classificação da rede, é realizada pelo método do gradiente descendente estocástico. Na primeira fase, as informações são propagadas através das camadas na direção de classificação da rede, calculando os valores da saída. Na segunda fase, o erro entre os valores esperados e os reais da saída é calculado. Retropropagando e minimizando esse erro, a matriz de peso é ajustada posteriormente e a rede é então ajustada com precisão [Liu et al., 2017]. A minimização da função de perda pode empregar diferentes algoritmos de resolução do problema do gradiente estocástico [Lobato et al., 2018].

Devido à complexidade das operações de convolução e a intricada interconexão dos neurônios, as redes convolucionais tendem a apresentar alto custo computacional e baixa velocidade de treinamento. A fim de otimizar a operação dessas redes, as operações de convolução são executadas em hardware especializado como em unidades de processamento gráfico (GPU).

3. Filtros de Bloom

O Filtro de Bloom é uma estrutura de dados com espaço eficiente para representar um conjunto $S = \{s_1, s_2, \cdots, s_n\}$ de n elementos [Laufer et al., 2011]. O filtro é constituído por uma matriz de m-bits e por k funções *hash* independentes, $h_1, h_2, \cdots, h_k$, cujas saídas variam uniformemente sobre o espaço discreto $0, 1, \cdots, m-1$. Para a construção do filtro, primeiramente todos os bits na matriz são definidos com valor zero. Em seguida, cada elemento $s_i \in S$ é inserido no filtro, definindo os bits correspondentes às posições $h_1(s_i), h_2(s_i), \cdots, h_k(s_i)$ em 1. No final do processo, as consultas de pertinência são realizadas. Para determinar se um determinado elemento x pertence a S, é preciso verificar se os bits nas posições $h_1(x), h_2(x), \cdots, h_k(x)$ estão todos configurados em 1. Se pelo

menos um bit estiver em 0, então $x \notin S$ com certeza. Caso contrário, se todos os bits forem definidos como 1, é assumido que $x \in S$. No entanto, há uma probabilidade de que um elemento externo $x \notin S$ ser reconhecido como um elemento legítimo de S. Esse evento, é chamado de falso positivo e ocorre quando os bits $h_1(x), h_2(x), \cdots, h_k(x)$ foram todos definidos como 1 por outros elementos anteriormente inseridos. A probabilidade de falso positivo de um elemento $x \notin S$ é dado por $f_p^{BF} = (1-p)^k \approx (1-e^{km/m})^k$ que representa a probabilidade de encontrar um conjunto de bits em 1 para todas as posições k indicadas pelas funções *hash*. O Filtro de Bloom é eficiente no uso de tempo e espaço. O Filtro de Bloom com m bits e k funções *hash* permite a inserção e a pesquisa em $\mathcal{O}(k)$, já que a entrada deve ser executada por todas as funções *hash*. Assim, nessa estrutura de dados a complexidade de tempo não depende do número de elementos nela. Isso ocorre porque os elementos nunca entram na estrutura, apenas seus índices calculados por *hash*. Para ter uma baixa taxa de falsos positivos $m >> 0$. Logo, o espaço da estrutura de dados que contém os dados, é $\mathcal{O}(m)$. Neste artigo, a proposta aplica o conceito de Filtro de Bloom para geração de um vetor de índices que determina a agregação determinística de fluxos em uma janela de uso da rede.

4. Redes Neurais Convolucionais com Filtro de Bloom

O aprendizado profundo com redes neurais convolucionais de duas dimensões assume que os dados de entrada são imagens sobre as quais serão buscados padrões específicos. Os fluxos de rede, no entanto, não são caracterizados naturalmente como dados bidimensionais e os padrões de ameaças nesses fluxos não são triviais. Para tanto, a aplicação de mecanismos baseados em conceitos de visão computacional sobre fluxos de rede exige a representação adequada dos dados, conferindo representatividade e coerência nos padrões formados. Dessa forma, a proposta considera a abordagem de representação de dados pela agregação de fluxos através da adição dos fluxos em um Filtro de Bloom.

A Figura 1 mostra o funcionamento da abordagem. Inicialmente, os pacotes são abstraídos em fluxos. Neste trabalho um fluxo f_i é uma sequência de pacotes caracterizada pela mesma $5-$ tupla, IP de origem, IP de destino, porta de origem, porta de destino e protocolo, durante uma janela de tempo. O tempo é abstraído em janelas saltitantes

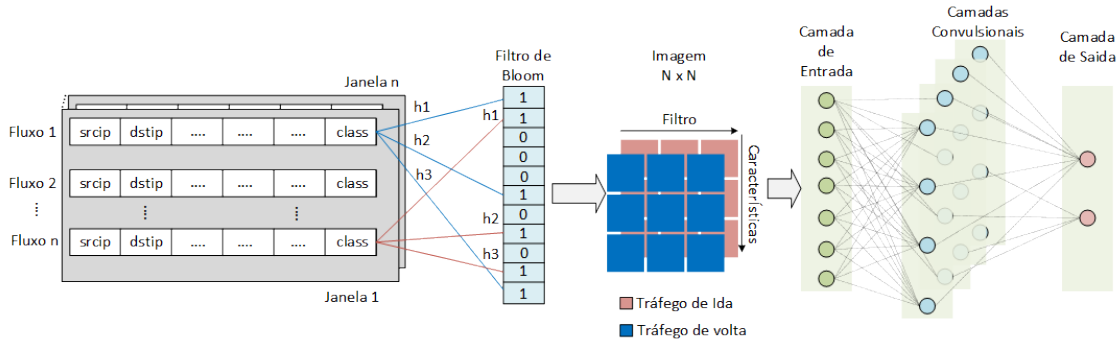


Figura 1. Arquitetura da abordagem proposta para criação de resumos bidimensionais de fluxos de redes. Uma janela de fluxos de tamanho variável é resumida em matrizes bidimensionais com dimensões predefinidas, independentemente da quantidade de fluxos contida na janela. As redes neurais convolucionais demandam que toda amostra de dados apresente mesma forma dimensional.

$W = w_1, w_2, \cdots, w_n$. Cada janela contém um número finito de fluxos f_n durante um período de tempo T_i, $w_i = f_1, f_2, \cdots, f_n$, onde $w_i \in T_i$. Contudo, o número de fluxos varia entre janelas. Dessa forma, considera-se que f_n é um número finito, porém não limitado. Por não ter dimensões limitadas, a janela de fluxos não pode ser diretamente fornecida como entrada às redes neurais convolucionais.

O resumo de fluxos com Filtro de Bloom objetiva representar as janelas de fluxos com dimensões finitas mas não limitadas em um espaço vetorial com dimensionalidade conhecida *a priori*. Após a aplicação do filtro são geradas matrizes em que as linhas representam as posições no filtro e as colunas, as características do conjunto de dados. Para cada janela que é completada em um tempo T_i, os endereços de origem e destino de cada fluxo são adicionados ao Filtro de Bloom, criando dois novos subconjuntos, ida e volta. O subconjunto de ida é gerado a partir do endereço origem com as características do fluxo na direção da origem para o destino. O subconjunto de volta é gerado a partir do fluxo de resposta. Caso não exista reposta dentro da mesma janela, um fluxo zerado é agregado ao subconjunto da volta. Quando o filtro é aplicado em cada um dos endereços, é obtido o resultado das funções *hashes* que é usado para validar se o endereço IP está presente no filtro. Essas funções *hashes* indicam as posições do filtro que correspondem ao armazenamento do fluxo. As posições no Filtro de Bloom indexam as linhas da matriz gerada com o resumo dos fluxos da janela. Se alguma das posições já existe, as características do fluxo são agregadas com os demais fluxos que pertencem àquela posição, reduzindo a quantidade de dados a ser processada. Finalmente, os subconjuntos de ida e volta são agrupados para obter duas matrizes, ida e volta, de $M \times C$, sendo M o tamanho do Filtro de Bloom em bits e C o número de características no conjunto de dados original. A fim de simplificar a representação das matrizes, é definido que o número de bits no filtro é igual ao número de características no conjunto de dados original, levando a matrizes quadradas $M \times M$, já que $M = C$. Essas matrizes são mapas de bits interpretados como imagens resumos, com a ida representando a componente vermelha e a volta a componente azul em uma imagem RGB. A imagem gerada é a entrada para a rede neural convolucional.

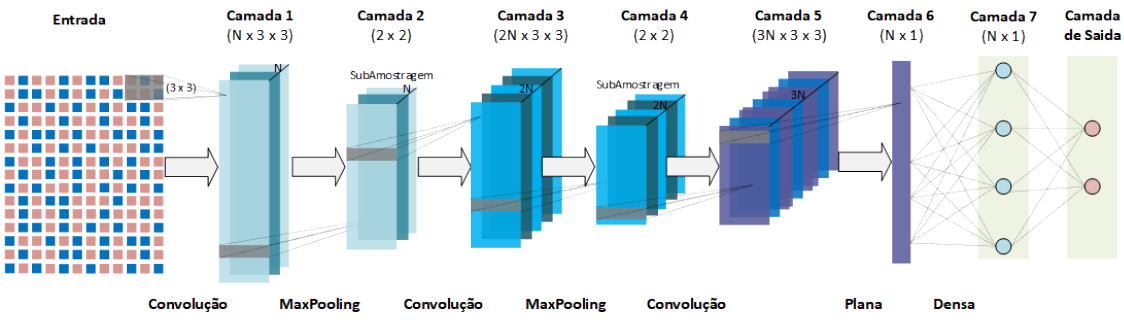


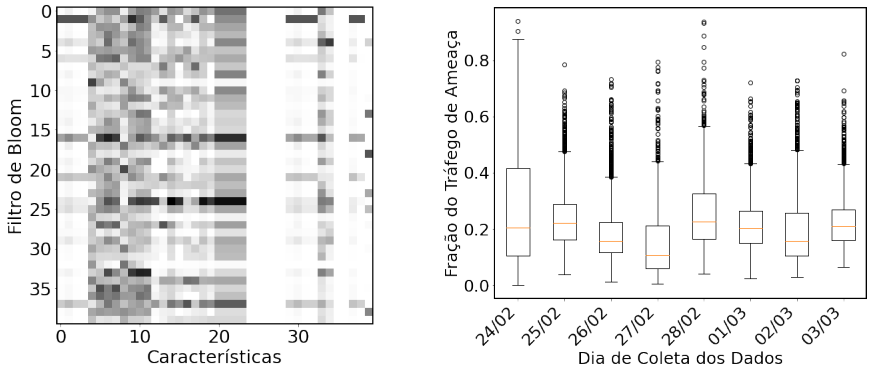
Figura 2. O aprendizado profundo proposto acontece em sete camadas. A Camada 1 recebe a imagem bidimensional e com duas componentes de cores que representa a janela de fluxos e executa a convolução com 64 filtros. As Camadas 2 e 4 executam regularização dos dados. As Camadas 3 e 5 executam a convolução nos dados sub-amostrados. A Camada 6 converte os dados em multidimensionais em um vetor de uma única dimensão. A Camada 7 realiza a classificação baseada nas características extraídas pelas camadas convolucionais. A Camada 8 consolida as classificações.

A Figura 2 exemplifica a rede neural convolucional usada para realizar a classificação das imagens [Wang e Majewicz Fey, 2018]. A rede combina duas camadas convolucionais com camadas de regularização, planificação das saídas, densas e a camada de saída. A rede inicializada com vieses em zero e os pesos são inicializados a partir de uma distribuição gaussiana com média 0 e uma variância de $\sigma_i = 1/N_{i-1}$, em que N_{i-1} é o número de neurônios da camada anterior, $i-1$. A função de perda usada é a entropia cruzada binária (*Log Loss*) entre o rótulo predito e a marcação original do dado. O gradiente descendente estocástico é calculado para lotes pequenos de 5 imagens. A rede se baseia em um solucionador Adam para calcular as taxas de aprendizagem de forma adaptativa para cada parâmetro do neurônio e otimizar a função de perda. As camadas convolucionais aplicam 32, 64 e 64 filtros respectivamente em janelas no formato 3×3 e a função de ativação usada é a linear retificada. A função de ativação linear retificada (ReLU) é uma função linear, contínua por partes, que reproduz a entrada na saída diretamente se for positiva, caso contrário, produz zero, $f(x) = max(0, x)$. A rede também aplica duas camadas de regularização para evitar o sobreajuste. As camadas de regularização (*Max_Pooling*) reduzem a resolução da representação de entrada tomando o valor máximo sobre a janela definida para cada dimensão ao longo do eixo das características. No caso da proposta, são consideradas janelas no formato 2×2. Ao final, as características são processadas por uma camada de planificação, levando dados multidimensionais para o formato vetorizado. O vetor unidimensional é processado por uma camada densa totalmente conectada com 64 nós. A camada de saída é representa por um neurônio ativado com função sigmoide. O neurônio de saída converge as diferentes características para uma classificação binária.

5. Avaliação da Proposta e Resultados Experimentais

A proposta foi avaliada sobre um conjunto de dados real (NetOp), coletado na rede de acesso de uma grande operadora de telecomunicações da cidade do Rio de Janeiro [Andreoni Lopez et al., 2017]. O conjunto de dados analisado foi criado a partir da captura de pacotes brutos contendo informações reais de tráfego IP dos usuários residenciais. O tráfego contém dados reais anonimizado de clientes da operadora de telecomunicações, foi coletado e registrado ininterruptamente por uma semana, entre os dias 24 de fevereiro e 03 de março de 2017. O conjunto de dados é criado capturando 5 TB de dados de acesso de 373 usuários residenciais. O conjunto de dados contém tráfego legítimo, ataques e outras ameaças à segurança. A marcação de tráfego legítimo e ameaça foi realizada através de um sistema de detecção de intrusão (IDS) que inspeciona o tráfego e, em seguida, o resume em conjunto de características de fluxo associado a um alerta IDS ou a uma classe de tráfego legítimo [Andreoni Lopez et al., 2017]. A abstração dos pacotes em fluxo foi realizada através do Flowtbag[3] que abstrai um fluxo em 45 características estatísticas [Andreoni Lopez et al., 2017], (5) informações da tupla de indentificação (IP/portas/protocolo), (4) quantidade de pacotes/bytes na ida/volta, (8) estatísticas de pacotes na ida/volta, (8) tempos entre pacotes na ida/volta, (8) estatísticas de tempo de fluxo, (4) quantidade de pacotes/bytes em subfluxos para ida/volta, (4) número de marcações (*flags*) TCP, (2) quantidade de bytes usados em cabeçalhos, (1) tipo de serviço, (1) qualidade de serviço. A extração de características mais importantes para a classificação é realizada pela composição das camadas de convolução e regularização.

[3]flowtbag: https://github.com/DanielArndt/flowtbag, Acessado em Março 2021.



(a) Representação da janela com a agregação por Filtro de Bloom.

(b) Distribuição da fração de fluxos de ameaça em relação ao total por janela.

Figura 3. Os fluxos de rede são agrupados em janelas de 30 s. A classificação da janela é função do percentual de fluxos de ameaça contido na janela. a) Imagem normalizada em escala de cinza. O eixo vertical representa a agregação de fluxos por posição no filtro, enquanto o eixo horizontal representa as características que compõem os fluxos. Fluxo de ida é representado pela componente de cor vermelha e o fluxo de volta, pela componente de cor azul. b) A fração de fluxos de ameaça nas janelas ao longo dos dias varia, mas a mediana é próxima a 0,20.

O aprendizado profundo foi implementado baseado na biblioteca Keras [4] através das interfaces providas pelo arcabouço TensorFlow [5]. O TensorFlow aplica a extensão para execução em unidades de processamento gráfico através das bibliotecas de tempo de execução Nvidia CUDA [6]. O ambiente de execução dos testes é composto por quatro máquinas equipadas com processador Intel Core i7 9700 a 3,00 GHz, com 16 GB de RAM DDR4 e placa de vídeo NVIDIA GTX 1660 SUPER com 6GB de RAM GDDR5. O ambiente executa o sistema operacional Ubuntu 18.04 com Python 3.6, TensorFlow 2.4.1 e pacote de bibliotecas CUDA 11.2 e cuDNN 8.1.1.33.

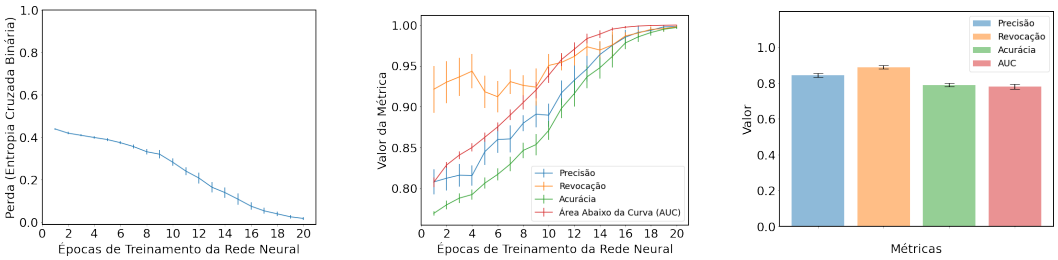
A avaliação da proposta é dividida em quatro etapas. Na primeira etapa é verificada a proporção de fluxos de ameaça e fluxos de tráfego normal contidos nas janelas. Na segunda etapa é avaliado o comportamento evolutivo do treinamento do modelo em relação ao número de épocas consideradas na rede neural. Na terceira etapa é realizado o treinamento de um modelo por dia do conjunto de dados, com o objetivo de verificar a convergência das métricas de avaliação. Por fim, na quarta etapa, é realizada a avaliação de um modelo incremental em que os dados de treinamento são os dados dos dias anteriores e os dados de teste, do dia avaliado. Os resultados são as médias de rodadas do experimento em um intervalo de confiança de 95%.

A proposta se baseia na geração de imagens de resumo representativas das janelas de fluxos de rede através da agregação com Filtro de Bloom, a exemplo da mostrada na Figura 3(a) em escala de cinza. O protótipo desenvolvido para a avaliação da proposta considera um Filtro de Bloom simplificado com apenas uma função *hash* e, portanto, agindo como uma Tabela *Hash*. O conjunto de dados original conta com 45 características, sendo cinco categóricas de identificação do fluxo (portas de origem e destino, endereços IP de

[4]Disponível em https://keras.io/.

[5]Disponível em https://www.tensorflow.org/.

[6]Disponível em https://docs.nvidia.com/cuda/cuda-runtime-api/index.html.



(a) Decaimento da função de perda em relação às épocas.

(b) Evolução das métricas em função das épocas.

(c) Métricas consolidadas para todo o conjunto de dados.

Figura 4. Análise do conjunto de dados completo para identificação de janelas com no mínimo 20% de tráfego suspeito. Treinamento e validação com todos os dias, particionamento dos dados em 80% para treinamento e 20% para teste. (a) Decaimento da função de perda é acentuado até 15 épocas de treinamento da rede neural. (b) Convergência das métricas de classificação com 20 épocas de treinamento. (c) Aplicação do modelo final treinado sobre os dados de teste. O modelo atinge acurácia superior a 0,80 e precisão aproximada de 0,85.

origem e destino e tipo de protocolo) que foram excluídas na agregação com Filtro de Bloom. Foram consideradas 40 características numéricas para a geração da imagem de resumo. Por simplicidade, o Filtro de Bloom adota o número de bits no filtro igual ao número de características, levando a imagens quadradas, 40×40 pixels. O protótipo desenvolvido emprega janelas saltitantes de 30 s para a geração dos resumos.

A primeira etapa da avaliação da proposta analisa as imagens de resumo geradas a partir das janelas de fluxos. Verifica-se que a taxa de fluxos suspeitos de conter ameaças em relação ao total varia entre 0 e 0,60 para mais de 75% das janelas analisadas entre os dias 25 de fevereiro e 3 de março de 2017, Figura 3(b). Contudo, ressalta-se que para todos os dias, a mediana da taxa de fluxos de ameaça em relação ao total é próxima a 0,20, indicando que aproximadamente metade das janelas de dados apresentam até 20% dos seus fluxos identificados como ameaças, considerando o conjunto de dados NetOp [Andreoni Lopez et al., 2017]. Neste trabalho, adota-se o limiar de 0,20 para a classificação de janelas que contenham fluxos suspeitos. Dessa forma, caso o número de fluxos suspeitos contidos em uma imagem resumo seja superior a 20% do total de fluxos, a imagem de resumo é marcada como ameaça.

A segunda etapa da avaliação foca no treinamento da rede neural convolucional. Esta etapa avalia a evolução das métricas de classificação ao longo de 20 épocas de treinamento da rede, mostrada na Figura 4. Para tanto, o treinamento e o teste da rede neural foram realizados utilizando o conjunto de imagens de resumos na proporção de 80% das imagens para treinamento e 20% das imagens para teste.

A Figura 4(a) explicita o comportamento da função de perda usada como erro a ser minimizado pela rede neural convolucional durante o seu treinamento. O protótipo adota a função de Entropia Cruzada Binária (*Log Loss*) como a função de perda. A entropia cruzada binária pondera o custo de uma classificação errada proporcional ao $log(p_i)$, em que p_i é a probabilidade de a amostra pertencer à classe alvo. Assim, o treinamento da rede visa minimizar a entropia cruzada entre as classes. Tal função é usada devido ao seu desempenho em problemas de classificação binária com probabilidades próximas de

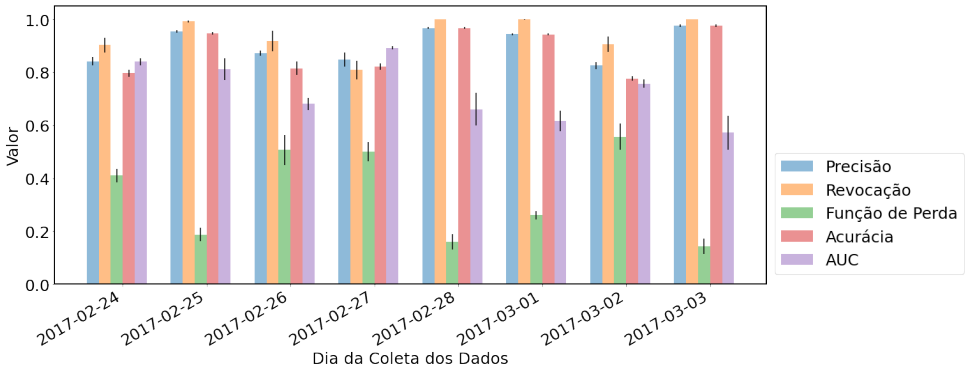


Figura 5. Evolução das métricas de classificação ao longo dos dias da semana. Os dias que apresentam a função de perda com os maiores valores e menor acurácia na classificação correspondem aos dias com janelas com menores proporções de fluxos de ameaça em relação ao tráfego total.

uma amostra pertencer à classe alvo ou não. A Figura 4(a) mostra que a queda na função de perda é mais acentuada após a décima época, mostrando um ponto de inflexão e com queda menos acentuada próximo à décima quinta época de treinamento. A Figura 4(b) analisa as métricas de acurácia, precisão, revocação e área abaixo da curva Característica de Operação do Receptor (*Receiver Operating Characteristic* - ROC). Ressalta-se que mesmo no início do treinamento, a rede neural apresenta um bom desempenho, com precisão acima de 0,85 e revocação acima de 0,90. As métricas tendem a convergir para o valor ótimo ao longo do treinamento, mostrando um ponto de inflexão próximo à décima quinta época, em consonância com o resultado apresentado pela função de perda, indicando uma situação de compromisso entre generalização e especialização da rede. Ao final da segunda etapa de avaliação, o modelo treinado foi testado sobre a partição dos dados dedicada ao teste, resultando em acurácia próxima a 0,80, revocação, a 0,85 e área abaixo da curva ROC de aproximadamente 0,80, Figura 4(c). Destaca-se que para o treinamento e teste do modelo foram usados os dados de todos os dias, que apresentam características estatísticas distintas a cada dia, inclusive com variações no número de ameaças por dia [Andreoni Lopez et al., 2017]. Assim, a Figura 4(c) destaca o compromisso da rede proposta entre especialização e generalização.

A terceira etapa avalia as métricas de acurácia, precisão, revocação, área abaixo da curva ROC e a função de perda para modelos treinados em lote a cada dia na proporção de 80% de dados para treinamento e 20% para teste, conforme mostra a Figura 5. O modelo de aprendizado de um dia é descartado no dia seguinte e, assim, mudanças no comportamento estacionário entre diferentes dias da semana não afetam a classificação. A Figura 5 mostra que as métricas variam entre os dias de coleta dos dados, com resultados consistentes de alta precisão e alta revocação. Destacam-se os dias 27 de fevereiro e 02 de março de 2017, em que houve a classificação com menor desempenho entre os dias considerados. Ao comparar o resultado da Figura 5 e o da Figura 3(b), verifica-se o fato de que ambos os dias apresentam queda na taxa de fluxos de ameaça nas janelas, sendo que o dia 27 de fevereiro é o dia do conjunto de dados com o maior número de janelas com baixa taxa de fluxos de ameaça. Esse resultado ratifica o fato de que a rede proposta tem melhor desempenho ao reconhecer janelas com maior número de ameaças. Vale notar que a queda de desempenho na classificação é evidente pelo maior valor atingido pela função de perda ao final do treinamento.

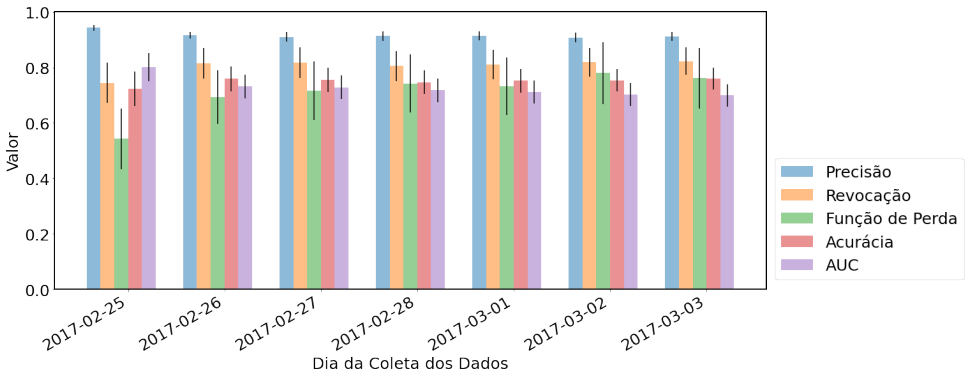


Figura 6. Evolução das métricas de classificação ao longo dos dias da semana ao usar classificador incremental. O treinamento é realizado com o conjunto de dados consolidado de todos os dias anteriores. Cada novo dia é treinado de forma incremental no modelo já treinado do dia anterior. Os dados de teste são as entradas do dia avaliado. O primeiro dia do conjunto de dados não é apresentado, pois não há dados para teste, já que os dados desse dia são usados para treinar o dia segundo dia.

Por fim, a quarta etapa da avaliação da proposta visa o teste do modelo incremental, em que o treinamento do modelo é realizado com os dados dos dias anteriores e o teste é realizado com o dia presente. Como os dados do dia anterior são usados para o treinamento, o primeiro resultado apresentado na Figura 6 é para o dia 25 de fevereiro, já que os dados do primeiro dia do conjunto de dados, 24 de fevereiro, foram usados para o treinamento do classificador do dia seguinte. Ao comparar os resultados das Figuras 5 e 6, verifica-se que o modelo incremental tende a ter desempenho mais previsível, já que o modelo é treinado com maior quantidade de dados. A função de perda apresenta maiores valores no treinamento incremental. O aumento na função de perda no modelo incremental levanta a hipótese de que haja mudança de conceito [Viegas et al., 2019] entre os dias considerados. Contudo, a hipótese não foi testada, já que não há queda significativa na acurácia, na precisão ou na revocação do modelo incremental.

6. Trabalhos Relacionados

Filtros de Bloom são amplamente utilizados em diversas aplicações em redes de computadores [Laufer et al., 2011, de Oliveira et al., 2020]. Laufer *et al.* analisam uma desvantagem dos Filtros de Bloom, a taxa de falsos positivos, propondo um Filtro Bloom Generalizado. O Filtros de Bloom generalizado é adequado para aplicações distribuídas, como rastreamento de IP, cache da *web* e redes ponto a ponto. Embora a proposta diminua a taxa de falsos positivos, introduz falsos negativos nas consultas aos filtros [Laufer et al., 2011]. A seleção de juízes para o monitoramento de nós mineradores em cadeias de blocos também é proposta através do uso de Filtro de Bloom [de Oliveira et al., 2020]. Xiao *et al.* propõem um sistema de detecção projetado para Rede Definida por Software (SDN) [Xiao et al., 2016] em que, na coleta de dados, o sistema verifica a tabela de fluxo para a coleta. O fluxo de tráfego é coletado usando uma inspeção de cabeçalho IP. Depois de receber os pacotes, as características de IP são extraídas. As características extraídas são verificadas em um Filtro Bloom para detectar os fluxos anormais. O Filtro de Bloom é útil no armazenamento das informações do hospedeiro, no processamento de grande volume de tráfego em alta velocidade e no armazenamento das informações de ataque anormais.

Kato *et al.* propõem o uso de aprendizado profundo para extrair características do tráfego da rede. Os autores afirmam que o aprendizado profundo é capaz de extrair padrões mais complexos do que outras técnicas [Kato et al., 2017]. De forma semelhante, Hang *et al.* apresentam um mecanismo de detecção de tráfego de anomalia, o D-PACK, que consiste em uma Rede Neural Convolucional (CNN) e um modelo de aprendizado profundo não supervisionado para detectar padrões de tráfego e filtrar o tráfego anormal [Hwang et al., 2020]. Kwon *et al.* desenvolvem três arquiteturas diferentes de Rede Neural Convolucional com base na escalabilidade estrutural. As três arquiteturas são CNN rasa, CNN moderada e CNN profunda, dependendo da quantidade de camadas utilizadas na rede neural. As redes são avaliadas em três conjuntos de dados de tráfego público e o modelo raso da CNN mostra maior precisão de detecção que os demais [Kwon et al., 2018]. A LUCID é uma ferramenta leve baseada em CNN para a detecção de ataques de negação de serviço distribuídos (*Distributed Denial of Service* - DDoS) [Doriguzzi-Corin et al., 2020]. Os autores comparam o desempenho da proposta na execução de CPU e GPU mostrando uma clara melhora na execução em GPU. Além disso, é verificado o desempenho da proposta em três conjunto de dados diferentes. Jo *et al.* propõem o uso de Rede Neural Convolucional para o pré-processamento de pacotes [Jo et al., 2020]. No trabalho, é proposto o método de pré-processamento para IDS de rede que pode usar as características do *kernel*, usando as informações mínimas de protocolo, tamanho do campo e deslocamento.

Diferentes das propostas anteriores, neste artigo, é utilizada a combinação do Filtros de Bloom com as redes neurais convolucionais. Os filtros de bloom são utilizados por sua baixa complexidade e simplicidade para gerar resumos de fluxos de rede em matrizes bidimensionais gerando mapas de bits. Os mapas são inseridos em redes neurais de convolucional para a caracterização e classificação de fluxos de rede.

7. Conclusão

A análise dos dados trafegados em redes de alta velocidade é dificultado pelo volume, velocidade, variabilidade e volatilidade dos dados. Viabilizar o tratamento dos dados perante restrições de tempo e de recursos de hardware exige a aplicação de técnicas de resumo de dados. Paralelamente, o uso de técnicas de aprendizado profundo com redes neurais convolucionais é uma tendência atual e com alta precisão para a extração de características e a identificação de padrões ocultos em dados representados como imagens multidimensionais. Esse artigo apresentou a proposta de utilização de Filtros de Bloom para gerar resumos de fluxos de rede em matrizes bidimensionais. As principais vantagens do uso da técnica de resumo com Filtro de Bloom são a geração de mapas de bits com formato conhecido *a priori*, independentemente do número de fluxos em uma janela saltitante de observação da rede, e a simplicidade de implementação da técnica. A proposta foi avaliada em um conjunto de dados real, coletado entre os dias 24 de fevereiro e 03 de março de 2017 na rede de acesso de uma grande operadora de telecomunicações da cidade do Rio de Janeiro. O conjunto de dados contém fluxos de acesso à Internet de 373 usuários domésticos conectados à rede banda larga fixa. Um protótipo da proposta foi desenvolvido e executado em máquinas com suporte ao processamento na unidade gráfica (GPU), resultando em modelo eficiente e possível de ser executado diretamente na GPU, liberando recursos para outras tarefas intensivas em computação. O modelo treinado apresentou altas precisão e revocação tanto no treinamento em lote quanto no treinamento incremental.

O modelo incremental apresentou precisão superior a 0,90 e revocação próxima a 0,80 para a detecção de janelas com taxa de ameaças superior a 20% do tráfego total, para um intervalo de confiança de 95%. Como trabalhos futuros pretende-se ampliar a avaliação da proposta para outros conjuntos de dados e realizar novas avaliações considerando redes neurais de memória longa de curto prazo (Long Short-Term Memory - LSTM) e a regressão da taxa de ameaças por janela.

Agradecimentos

Este trabalho foi realizado com recursos do CNPq, FAPERJ, RNP, CAPES, CGI/FAPESP (2018/23062-5) e Prefeitura de Niterói/FEC/UFF (Edital PDPA 2020).

Referências

Andreoni Lopez, M., Mattos, D. M. F., Duarte, O. C. M. B. e Pujolle, G. (2019). Toward a monitoring and threat detection system based on stream processing as a virtual network function for big data. *Concurrency and Computation: Practice and Experience*, 31(20):1–17.

Andreoni Lopez, M., Sanz, I. J., Lobato, A. G. P., Mattos, D. M. F. e Duarte, O. C. M. B. (2018). Aprendizado de máquina em plataformas de processamento distribuído de fluxo: Análise e detecção de ameaças em tempo real. *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC*, 2018:150–206.

Andreoni Lopez, M., Silva, S. R., Alvarenga, I. D., Rebello, G. A. F., Sanz, J. I., Lobato, A. G. P., Mattos, D. M. F., Duarte, O. C. M. B. e Pujolle, G. (2017). Collecting and Characterizing a Real Broadband Access Network Traffic Dataset. Em *IEEE/IFIP 1st Cyber Security in Networking Conference (CSNet'17)*, p. 1–8, Brazil.

Casas, P., Soro, F., Vanerio, J., Settanni, G. e D'Alconzo, A. (2017). Network security and anomaly detection with big-dama, a big data analytics framework. Em *2017 IEEE 6th International Conference on Cloud Networking (CloudNet)*, p. 1–7.

de Oliveira, M. T., Reis, L. H. A., Medeiros, D. S. V., Carrano, R. C., Olabarriaga, S. D. e Mattos, D. M. F. (2020). Blockchain reputation-based consensus: A scalable and resilient mechanism for distributed mistrusting applications. *Comput. Networks*, 179:107367.

de Oliveira, N. R., Pisa, P. S., Lopez, M. A., de Medeiros, D. S. V. e Mattos, D. M. F. (2021). Identifying fake news on social networks based on natural language processing: Trends and challenges. *Information*, 12(1).

de Souza, L. A. C., Antonio F. Rebello, G., Camilo, G. F., Guimarães, L. C. B. e Duarte, O. C. M. B. (2020). Dfedforest: Decentralized federated forest. Em *2020 IEEE International Conference on Blockchain (Blockchain)*, p. 90–97.

Doriguzzi-Corin, R., Millar, S., Scott-Hayward, S., Martínez-del-Rincón, J. e Siracusa, D. (2020). Lucid: A practical, lightweight deep learning solution for ddos attack detection. *IEEE Transactions on Network and Service Management*, 17(2):876–889.

Hwang, R.-H., Peng, M.-C., Huang, C.-W., Lin, P.-C. e Nguyen, V.-L. (2020). An unsupervised deep learning model for early network traffic anomaly detection. *IEEE Access*, 8:30387–30399.

Jo, W., Kim, S., Lee, C. e Shon, T. (2020). Packet preprocessing in cnn-based network intrusion detection system. *Electronics*, 9(7):1151.

Kato, N., Fadlullah, Z. M., Mao, B., Tang, F., Akashi, O., Inoue, T. e Mizutani, K. (2017). The deep learning vision for heterogeneous network traffic control: Proposal, challenges, and future perspective. *IEEE Wireless Communications*, PP(99):2–9.

Kwon, D., Natarajan, K., Suh, S. C., Kim, H. e Kim, J. (2018). An empirical study on network anomaly detection using convolutional neural networks. Em *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, p. 1595–1598. IEEE.

Laufer, R. P., Velloso, P. B. e Duarte, O. C. M. B. (2011). A generalized bloom filter to secure distributed network applications. *Computer Networks*, 55(8):1804–1819.

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y. e Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26.

Lobato, A. G. P., Andreoni Lopez, M., Sanz, I. J., Cardenas, A. A., Duarte, O. C. M. B. e Pujolle, G. (2018). An adaptive real-time architecture for zero-day threat detection. Em *2018 IEEE International Conference on Communications (ICC)*, p. 1–6.

Qin, Y., Shen, G.-w., Zhao, W.-b., Chen, Y.-p., Yu, M. e Jin, X. (2019). A network security entity recognition method based on feature template and cnn-bilstm-crf. *Frontiers of Information Technology & Electronic Engineering*, 20(6):872–884.

Reis, L. H. A., Murillo Piedrahita, A., Rueda, S., Fernandes, N. C., Medeiros, D. S. V., de Amorim, M. D. e Mattos, D. M. F. (2020). Unsupervised and incremental learning orchestration for cyber-physical security. *Transactions on Emerging Telecommunications Technologies*, 31(7):e4011.

Silva, J. V. V., Andreoni Lopez, M. e Mattos, D. M. F. (2020). Attackers are not stealthy: Statistical analysis of the well-known and infamous kdd network security dataset. Em *2020 4th Conference on Cloud and Internet of Things (CIoT)*, p. 1–8.

Sultana, F., Sufian, A. e Dutta, P. (2020). Evolution of image segmentation using deep convolutional neural network: A survey. *Knowledge-Based Systems*, 201-202:106062.

Viegas, E., Santin, A., Bessani, A. e Neves, N. (2019). Bigflow: Real-time and reliable anomaly-based intrusion detection for high-speed networks. *Future Generation Computer Systems*, 93:473 – 485.

Wang, Z. e Majewicz Fey, A. (2018). Deep learning with convolutional neural network for objective skill evaluation in robot-assisted surgery. *International Journal of Computer Assisted Radiology and Surgery*, 13(12):1959–1970.

Xiao, P., Li, Z., Qi, H., Qu, W. e Yu, H. (2016). An efficient ddos detection with bloom filter in sdn. Em *2016 IEEE Trustcom/BigDataSE/ISPA*, p. 1–6. IEEE.

Zhang, Y., Chen, X., Guo, D., Song, M., Teng, Y. e Wang, X. (2019). Pccn: Parallel cross convolutional neural network for abnormal network traffic flows detection in multi-class imbalanced network traffic flows. *IEEE Access*, 7:119904–119916.