Avaliação de Desempenho de Modelos Deep Learning para Deteção de Intrusão em Dispositvos IoT

Valdir Carvalho[1], **Ewerton Queiroz**[1], **Júlio Mendonça**[2]
Gustavo Callou[1], **Ermeson Andrade**[1]

[1]Departamento de Computação, Universidade Federal Rural de Pernambuco (UFRPE)
Recife – PE – Brasil

[2]Coordenação de Informática, Instituto Federal de Alagoas (IFAL)
Arapiraca – AL – Brasil

{valdir.carvalhof, ewerton.queiroz, gustavo.callou}@ufrpe.br

ermeson.andrade@ufrpe.br, julio.neto@ifal.edu.br

Abstract. *With the high growth in the number of devices connected to the Internet of Things (IoT), digital security has become one of the main points to be addressed. Many of these devices use outdated technologies, which create vulnerabilities that can be exploited by criminals. New security techniques emerge through the use of Artificial Intelligence models to improve old implementations, such as Deep Learning-based intrusion detection systems to identify cyber attacks. However, the existing works do not evaluate the impacts of these systems on the performance of environments with computational restrictions, such as IoT devices. Thus, this work aims to evaluate the performance of two intrusion detection models based on Deep Learning developed for IoT environments. The results reveal that different models have different impacts on the performance of IoT devices. Additionally, the model that receives a greater volume of attacks, and has greater accuracy, consumes more resources, which can be quite problematic for IoT environments.*

Resumo. *Com o alto crescimento do número de dispositivos conectados à Internet das Coisas (IoT), a segurança digital tornou-se um dos principais pontos a serem tratados. Muitos desses dispositivos utilizam tecnologias defasadas, as quais criam vulnerabilidades que podem ser exploradas por criminosos. Novas técnicas de segurança surgem através da utilização de modelos de Inteligência Artificial para melhorar antigas implementações, por exemplo, os sistemas de detecção de intrusão baseados em Deep Learning para identificar ataques cibernéticos. No entanto, os trabalhos existentes não avaliam os impactos desses sistemas no desempenho de ambientes com restrições computacionais, tais como os dispositivos de IoT. Assim, este trabalho objetiva avaliar o desempenho de dois modelos de detecção de intrusão baseados em Deep Learning desenvolvidos para ambientes de IoT. Os resultados revelam que diferentes modelos têm diferentes impactos no desempenho dos dispositivos de IoT. Adicionalmente, o modelo que recebe um maior volume de ataques, e possui maior precisão, consome mais recursos, o que pode ser bastante problemático para os ambientes de IoT.*

1. Introdução

A Inteligência Artificial (IA) [Pannu 2015], e suas subáreas como *Machine* e *Deep Learning* (ML e DL) [LeCun et al. 2015], têm apresentado novas perspectivas quando integradas a áreas emergentes como a Internet das Coisas (do inglês, *Internet of Things* (IoT)). IoT possui dispositivos eletrônicos conectados à Internet com capacidades e tamanhos variados, e que possuem funções diversas [Miraz et al. 2015]. Tais dispositivos vêm sendo adotados por indivíduos e pela indústria em larga escala para facilitar, monitorar e automatizar ambientes ou atividades como carros, cidades e fazendas. No entanto, apesar dos dispositivos de IoT facilitarem a automação de algumas atividades, com o crescente número de dispositivos conectados à Internet e a alta quantidade de dados trafegando na rede, problemas relacionados à segurança da informação são inevitáveis [Nurse et al. 2017].

Especialmente em ambientes IoT, a IA tem sido aplicada em *Single Boards*, que são dispositivos de baixo poder computacional, para a automação industrial [Gómez et al. 2015], para o reconhecimento facial em tempo real [Dürr et al. 2015] e, mais recentemente, para os sistemas de detecção de intrusão (do inglês, *Intrusion Detection Systems* (IDS)) [Ge et al. 2019]. Os IDS surgem para detectar, por meio de entradas específicas, ataques ou anomalias a sistemas de Tecnologia da Informação e Comunicação (TIC) e, consequentemente, gerar alertas para registrar tal evento. Atualmente, existem diversos sistemas que realizam este tipo de ação, de verificar possíveis anomalias (ou de analisar o estado de conexão) através de uma grande base de dados para comparação [Liao et al. 2013]. No entanto, tal abordagem não é viável para dispositivos IoT que possuem recursos computacionais limitados. Assim, trabalhos recentes têm buscado utilizar ML ou DL para prever ameças rapidamente e garantir a segurança desses dispositivos de forma mais eficiente [Vinayakumar et al. 2019, Ge et al. 2019].

Dispositivos IoT que usam soluções baseadas em ML ou DL, geralmente, desenvolvem modelos para detectar ataques em uma rede ou em uma infraestrutura específica. Em [Ge et al. 2019], os autores propuseram um modelo desenvolvido para detectar intrusão em um sistema computacional utilizando dados capturados da interface de rede de dispositivos IoT, como cabeçalhos HTTP, portas de origem e destino, tamanho do janelamento. [Farahnakian and Heikkonen 2018] apresentaram um modelo que utiliza DL para detecção de intrusão utilizando *Deep Auto-Encoder* como a principal fonte de aprendizado profundo e um *dataset* que possui os dados necessários para o treinamento do modelo. Já em [Aloqaily et al. 2019], os autores propuseram um sistema híbrido de detecção de intrusão para veículos conectados a cidades inteligentes, que é integrado à camada de segurança na monitoração da cidade. No entanto, nenhum dos trabalhos anteriores se preocuparam em como a implementação dos modelos de ML ou DL afetariam o desempenho geral dos dispositivos IoT, considerando a limitação de recursos.

Dessa forma, esse trabalho apresenta uma abordagem baseada em experimentos para avaliar os impactos de IDSs no desempenho dos dispositivos IoT. Mais especificamente, iremos avaliar dois modelos de DL para detecção de intrusão implantados em uma Raspberry Pi, que é um dispositivo IoT de placa única (*single board*). A diferença básica entre os modelos adotados é que o primeiro utiliza a classificação binária para classificar os ataques, enquanto o segundo utiliza a classificação multi-classe e uma função de ativação diferente do primeiro, além de possuir mais uma camada na rede neural e mais entradas de dados. Os resultados obtidos mostram que o ambiente avaliado não suporta

grandes cargas de trabalho e que o segundo modelo, o qual possui maior precisão, consome mais recursos em relação ao primeiro. Ambos, utilizando uma carga de trabalho de 0,3, travam o dispositivo de IoT em alguns minutos. As principais contribuições deste trabalho são listadas abaixo:

- a implantação de modelos de DL em um dispositivo de IoT real para execução de experimentos;
- a avaliação dos impactos da execução de modelos de *Deep Learning* no desempenho de uma *single board* através de uma abordagem experimental;
- a análise dos resultados de experimentos que evidenciam os *trade-offs* entre o desempenho e os modelos adotados.

O restante desse artigo está organizado como segue. A Seção 2 descreve os principais conceitos utilizados nesse trabalho. A Seção 3 descreve os trabalhos relacionados à avaliação de desempenho utilizando DL e IDS. A Seção 4 lista a arquitetura experimental utilizada para avaliação de desempenho de *single board* como dispositivo IoT. A Seção 5 detalha e discute os resultados dos experimentos. Por fim, a Seção 6 apresenta as conclusões e os possíveis trabalhos futuros.

2. Fundamentos

A seguir, são apresentados conceitos fundamentais empregados no trabalho.

2.1. Detecção de Intrusão

Diferentes técnicas para IDS [Lunt 1993] foram descritas na literatura e essas revelaram a importância da identificação do acesso não autorizado a sistemas computacionais e redes de computadores. Utilizando procedimentos que vão de auditorias e avaliações *offline* dos dados até testes em tempo real, os dados são verificados no momento da chegada ao equipamento ou à rede. No entanto, atualmente, os sistemas buscam analisar os dados em tempo real, pois o tempo para sinalização da ameaça necessita ser curto, a fim de adotar medidas efetivas e evitar danos aos sistemas que estão sendo atacados.

Técnicas baseadas em conhecimento, comportamento ou estado de conexão podem ser utilizadas integrando as tecnologias de IDS baseado em *host* (HIDS), IDS baseado em redes (NIDS), IDS baseado em redes sem fio (WIDS), IDS baseado na análise de comportamentos (NBA) e um IDS misto (MIDS) [Liao et al. 2013]. Novos mecanismos utilizando IA, ou mais especificamente DL, surgem diariamente em busca de aperfeiçoar os IDS, a fim de obter uma maior precisão na identificação dos dados, e diminuir ao máximo os falsos positivos que podem ocorrer na execução dos sistemas de segurança [Vinayakumar et al. 2019, Thamilarasu and Chawla 2019]. Ambientes experimentais são desenvolvidos para capturar o tráfego de rede por meio de arquivos PCAP (do inglês, *Packet Capture*) e assim servir de base para análises e experimentos na área de segurança da informação, inclusive no desenvolvimento de novas técnicas de detecção baseadas em IA.

2.2. Internet das Coisas

O conceito da Internet das Coisas surgiu com a intenção de conectar sensores de Identificação por Rádio Frequência (do inglês, *Radio-Frequency IDentification* (RFID)) [Ashton 1999] e realizar a comunicação entre objetos por meio de uma rede sem fio.

Posteriormente, os sensores foram integrados aos sistemas computacionais permitindo a interoperabilidade de diversos sistemas, equipamentos e protocolos. Novas aplicações foram desenvolvidas no decorrer dos anos, como por exemplo, aplicações para áreas da logística e para a área da medicina, além dos ambientes inteligentes (casa, escritório e fábricas) [Atzori et al. 2010]. Na atualidade, objetos, coisas, podem ser conectados às pessoas ao ponto de surgir o conceito de computação ubíqua [de Araujo 2003].

Além disso, novos conceito foram criados a partir da IoT, a exemplo de cidades inteligentes (do inglês, Smart City) [Eremia et al. 2017], o estudo para tratar e trabalhar com grandes volumes de dados (do inglês, Big Data) [Chen et al. 2014] e o estudo de uma rede de veículos autônomos (do inglês, Vehicular Ad hoc Networks) [Martinez et al. 2011]. No entanto, criar novos padrões para essas tecnologias se tornou um grande desafio, considerando o rápido desenvolvimento dos novos serviços, a comunicação entre dispositivos independentes e a segurança da informação. Alinhando a isso, os dispositivos de IoT possuem recursos limitados, a exemplo da Raspberry Pi. As novas tecnologias de ML e DL são desenvolvidas diariamente para o cenário de IoT [Din et al. 2019], integrando inclusive a área de segurança da informação [Ge et al. 2019].

3. Trabalhos Relacionados

Esta seção apresenta trabalhos relacionados a avaliação de desempenho de modelos de DL em dispositivos IoT. [Morabito 2017] avaliou o desempenho da utilização de instâncias em equipamentos com poucos recursos (ex.: *single boards* Raspberry Pi e Odroid), os quais são normalmente usados no ambiente IoT. O autor focou na avaliação de consumo de energia em cenários que requisitavam um alto processamento. Além disso, ele também realizou um comparativo entre o desempenho e o consumo de energia dos equipamentos para servir de base para futuras implementações. Em [Xu et al. 2017], foi feito a avaliação do desempenho de contêineres Docker executando modelos de DL em dois equipamentos através de *benchmarks*. Os testes avaliaram o uso de CPU, GPU e E/S do disco. Os resultados mostraram que a utilização de ferramentas de DL em contêineres Docker é uma solução viável, da qual pode beneficiar os sistemas com mais flexibilidade e com isolamento de recursos.

Em [Tama and Rhee 2017], os autores realizaram um estudo comparando o desempenho de cinco técnicas renomadas de IDS baseados em três classificadores, os quais visavam melhorar a capacidade de detecção de intrusão em sistemas computacionais. As técnicas utilizadas foram combinadas para maximizar a precisão na detecção de intrusão. Ao final, os resultados mostraram que as técnicas de *boosting* e *stacking* superaram em desempenho em relação aos demais, considerando os indicadores utilizados no experimento. Já em [Quincozes et al. 2018] os autores avaliaram um conjunto de atributos para a detecção de ataques de personificação em um ambiente de IoT, de modo a obter um maior sucesso na detecção de identidades falsas na rede. No entanto, os experimentos não foram realizados em ambiente IoT real, e sim em um desktop.

Considerando os trabalhos apresentados, é notável a falta de pesquisas que buscaram avaliar o desempenho dos dispositivos IoT que executam algoritmos ou modelos de DL no contexto de IDS. Diferentemente, este trabalho visa avaliar o desempenho de dois modelos baseados em DL, desenvolvidos para detecção de intrusão em ambientes IoT e implantados em uma Raspberry Pi objetivando analisar os *trade-offs* entre o desempenho

e os modelos adotados.

4. Arquitetura Experimental e Método de Avaliação dos Resultados

Esta seção apresenta a arquitetura experimental usada para avaliar o desempenho dos modelos de DL para detecção de intrusão em uma Raspberry Pi 4 Model B. O objetivo principal da arquitetura experimental é analisar o desempenho e verificar o impacto da execução de modelos de DL na *single board*, considerando que algoritmos de DL tendem a consumir bastante recursos. Foram considerados dois modelos DL definidos por [Ge et al. 2019] a fim de detectarem as intrusões executadas na *single board*. Neste trabalho iremos chamá-los de Modelo 1 e Modelo 2. O primeiro possui uma rede neural com duas camadas ocultas com a última camada utilizando a função de ativação *sigmoid* realizando a classificação binária. Já o segundo possui uma camada oculta a mais, utiliza a função de ativação *softmax* e considera entradas do cabeçalho HTTP, utilizando assim mais entradas na rede neural.

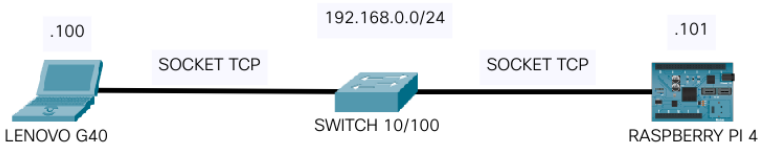

Figura 1. Infraestrutura experimental utilizada para realização dos experimentos.

A Figura 1 apresenta a infraestrutura experimental utilizada para realização dos experimentos. O ambiente foi configurado com um desktop Lenovo G40, um switch (10/100Mbps) e uma *single board* (Raspberry Pi 4 Model B). No desktop, foi utilizado o sistema operacional GNU/Linux distribuição Debian Stretch. Já na Raspberry Pi, foi utilizado o sistema operacional padrão, Raspberry Pi OS Buster, baseado na distribuição Debian Buster, executando, inclusive, o ambiente gráfico. A Tabela 1 apresenta a configuração do desktop e da Raspberry Pi utilizado nos experimentos.

Tabela 1. Configuração do desktop e da raspberry Pi utilizada nos experimentos.

Equipamento	**Processador**	**Qtd. núcleos**	**RAM**	**Arquitetura**
Raspberry Pi4 Model B	BCM2711 - 1.5 GHz	4	4 GB	64 bits
Lenovo G40-80	I5-5200U - 2.2 Ghz	4	4 GB	64 bits

Para a execução do experimento, foi usado o *dataset* proposto por [Koroniotis et al. 2019] para treinar e testar os modelos de DL. O *dataset* foi modificado e tratado conforme definido em [Ge et al. 2019], para que os modelos pudessem realizar as classificações. Os modelos de DL utilizados buscaram classificar se o dado de entrata era um ataque de Negação de Serviço (do inglês, *Denial of Service* (DoS)), de Negação de Serviço Distribuído (do inglês, *Distributed Denial of Service* (DDoS)), de reconhecimento ou de roubo de informação ou se era simplesmente tráfego normal de rede. Além disso, todos os ataques possuíam subcategorias (ex.: categoria DDoS, subcategoria TCP, UDP ou HTTP).

Assim, o Modelo 1 realizava uma classificação binária, verificando se era um tráfego malicioso ou um tráfego normal. Já o Modelo 2 buscava classificar a classe do ataque (ex.: DoS, DDoS, reconhecimento ou roubo de informação) com a subclasse associada (ex.: DDoS TCP ou roubo de informação por *Keylogging*). Vale ser ressaltado que o Modelo 2 recebeu mais dados de entrada para realizar a classificação, em relação ao Modelo 1. Adicionalmente, não houve o carregamento de pesos, *bias* ou de um modelo salvo previamente treinado para não alterar os dados coletados, pois, tais ações resultariam em diferentes consumos de memória, baseado no arquivo que seria carregado.

O experimento foi executado como descrito a seguir. O *dataset* de teste foi carregado na máquina desktop. Após isso, utilizando um algoritmo desenvolvido em Python, os dados do *dataset* foram lidos e enviados (amostra por amostra) através de um socket TCP para a Raspberry Pi. Quando a amostra enviada chega à Raspberry Pi, o modelo de DL realiza a classificação de tal dado. Além dos modelos de DL para detecção de intrusão, foi executada na Raspberry um *script* para coletar os dados de utilização do processador, disco, memória RAM e swap. Também foram coletados os tempos inicial e final de cada classificação das amostras, a fim de comparar o tempo de classificação de cada modelo e em cada carga de trabalho.

Diferentes cargas de trabalho foram configuradas para enviar as amostras do desktop à Raspberry. Foram enviadas, em média, uma linha do *dataset* a cada 1,0, 0,5, 0,4 e 0,3 segundos, respectivamente. Em outras palavras, quanto menor a carga de trabalho, maior é o estresse no ambiente. Vale ressaltar ainda que o ambiente de experimentos foi isolado, para evitar interferências externas, como uma atualização de *software* não programada, por exemplo. Foram considerados 8 cenários, sendo cada um deles formados pela combinação entre modelo a ser executado (Modelo 1 e 2) e a carga de trabalho a ser utilizada (1,0, 0,5, 0,4 e 0,3). O tempo total de execução de cada cenário foi de 40 minutos, sendo que os cinco minutos iniciais e finais foram para medir o sistema ocioso, de modo a capturar o comportamento do sistema ocioso e em execução. Durante os demais 30 minutos, foi realizado o envio das amostras seguindo a carga de trabalho estabelecida.

5. Resultados e Discussões

Esta seção apresenta e discute os resultados obtidos através dos experimentos executados utilizando a arquitetura experimental descrita anteriormente. Primeiro, são apresentados e discutidos os resultados obtidos durante a execução do Modelo 1 e, em seguida, os resultados do Modelo 2. Por fim, um comparativo entre os modelos é apresentado.

5.1. Modelo 1

A Figura 2 apresenta o consumo médio dos núcleos do processador da *single board*. Através do gráfico exibido nesta figura, é possível observar que utilizando a carga de trabalho de 1,0 (enviando uma linha do *dataset* por segundo), o equipamento obtém bom desempenho, consumindo menos de 25% do processador, em média, considerando o momento em que a carga de trabalho é executada. No entanto, utilizando a carga de 0,4 o equipamento consome praticamente 100% do processamento da *single board*. Já com uma carga de trabalho de 0,3, o modelo trava o equipamento e o sistema operacional encerra os processos que executavam o modelo de DL para classificação dos dados de entrada.

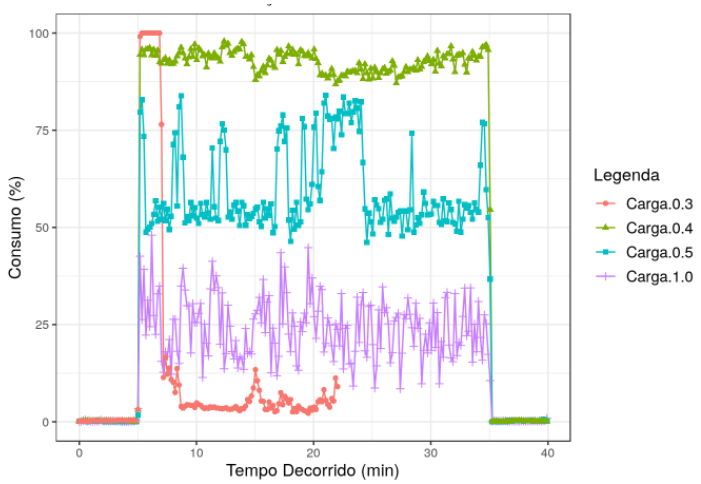

Figura 2. Média de utilização dos núcleos da CPU executando o Modelo 1.

A Figura 3 demonstra o alto consumo da memória RAM logo após o início do experimento, com carga de trabalho 0,3. Como pode ser observado, no minuto 6, o sistema da *single board* trava, levando a sua indisponibilidade. Vale ser ressaltado que o consumo de memória swap se comportou de maneira semelhante ao da memória RAM. Uma vez que a memória principal foi exaurida, rapidamente a secundária tentou realizar o armazenamento dos dados, porém também se esgotou. Diferentemente, nos experimentos com baixa carga de trabalho, como 1,0 e 0,5, não houve um consumo de memória excessivo. Apesar dos recursos de processamento terem sido bastante usados. Dessa forma, é possível notar que o Modelo 1 consome pouca memória principal, se utilizado em ambientes que possuem uma carga de trabalho não elevada.

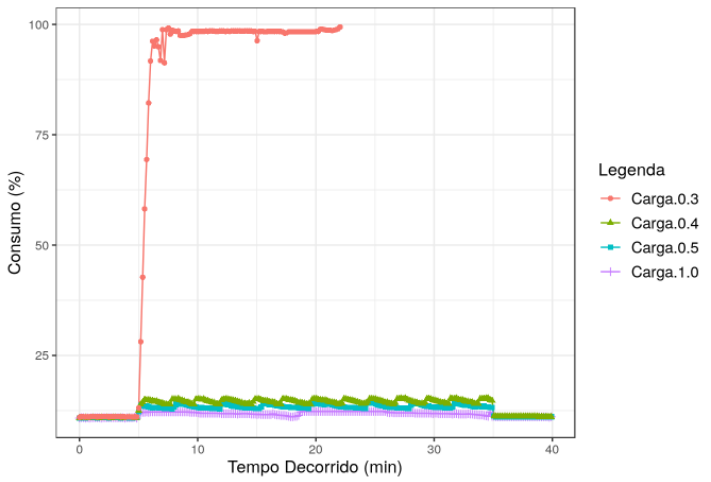

Figura 3. Modelo 1 - Consumo de memória RAM.

5.2. Modelo 2

O modelo 2 apresentou uma utilização de processamento mais intensa, principalmente considerando cargas de trabalho mais baixa. A Figura 4 apresenta a utilização da média do processador considerando as quatro diferentes cargas de trabalho adotadas. É possível

notar que utilizando uma carga de trabalho de 0,5, o equipamento teve uma utilização média de processamento aproximadamente de 75%. Quando os experimentos foram executados com as cargas de trabalho 0,3 e 0,4, o sistema parava de executar entre o minuto 6 e o minuto 8 do experimento, respectivamente. Dessa forma, quando a carga de trabalho é um pouco mais alta (menor que 0,5), os recursos da raspberry pi são consumidos rapidamente, levando a indisponibilidade do sistema.

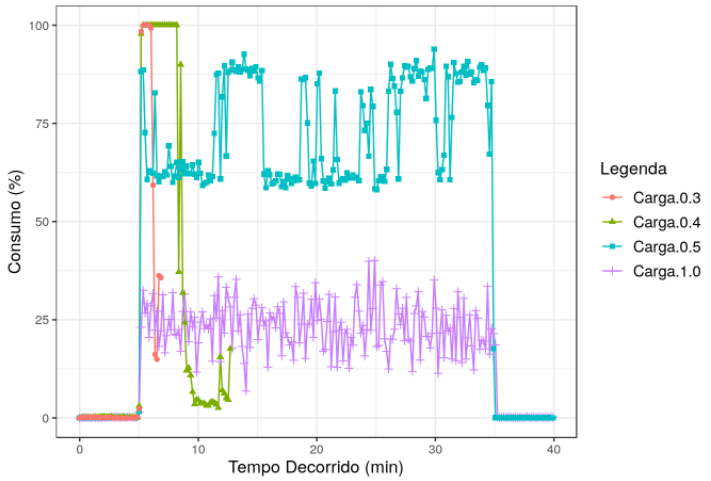

Figura 4. Modelo 2 - Média de utilização dos núcleos da CPU.

A Figura 5 mostra a consumo da memória RAM para o Modelo 2. Se considerarmos a carga de trabalho 0,3, a memória da *single board* é consumida aproximadamente no minuto 6. Porém, apesar da carga de trabalho de 0,4 não exaurir os recursos tão rapidamente quanto a anterior, a mesma chega a consumir todos os recursos disponíveis no minuto 8. A memória SWAP obteve um crescimento similar ao da memória RAM. Já para as carga de trabalho 1,0, e 0,5 não houve consumo exagerado do sistema, sendo possível assim realizar as classificações utilizando apenas recursos do processador.

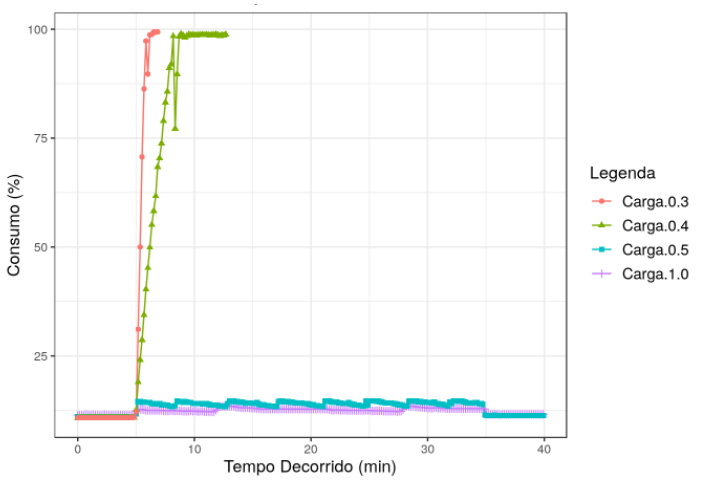

Figura 5. Modelo 2 - Consumo de memória RAM.

5.3. Comparativo entre os modelos adotados

Quando compara-se os resultados dos modelos, podemos observar pontualmente o quanto de recurso cada um consome no equipamento avaliado. Utilizando a carga de trabalho de 0,5, observou-se que o Modelo 2 consome mais recursos em comparação com o Modelo 1, conforme mostrado na Figura 6 para o processador e na Figura 7 para a memória RAM. A memória utilizada para a referida carga de trabalho durante a execução dos modelos atingiu uma média de 13,4% e 14,7% do consumo total do equipamento, para o Modelo 1 e 2, respectivamente. Alcançando assim uma diferença média menor do que 1% para o consumo da memória RAM. Já para o processador, obteve-se uma média de 59,1% de utilização para o Modelo 1 e 72,2% para o Modelo 2, evidenciando a diferença média de pouco mais de 13% no consumo.

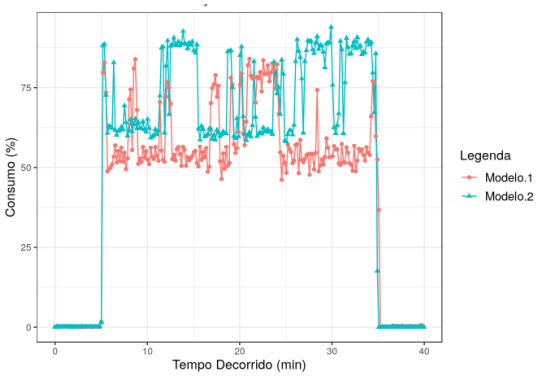

Figura 6. Comparativo do processador - carga 0.5

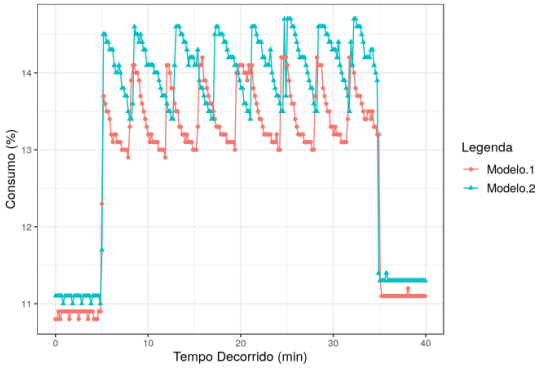

Figura 7. Comparativo da memória RAM - carga 0.5

Comparando os resultados utilizando a carga de trabalho 0,4, observou-se que o segundo modelo consome mais recurso em comparação com o primeiro, assim como na carga de trabalho de 0,5. Vale ser ressaltado que o segundo modelo recebe mais parâmetros de entrada e obtém maior precisão na classificação dos dados. Especificamente nesse cenário, não foi possível finalizar o experimento com o segundo modelo, pois os recursos da *single board* foram consumidos no minuto 8, em média, (ver Figuras 8 e 9). No entanto, considerando o tempo de execução dos experimentos até o travamento do equipamento, obteve-se em relação a memória RAM uma média de consumo de 14,5% e 64% em relação ao Modelo 1 e Modelo 2, respectivamente. Já em relação ao processador, o Modelo 1 atingiu 93,9% de utilização e Modelo 2 100%. Considerando a carga de trabalho de 0,3, pode-se observar que até o travamento do equipamento, o consumo de memória RAM foi de 68,8% para o Modelo 1 e 78,8% para o Modelo 2, no entanto, o consumo de processamento foi de 100% em ambos os modelos.

A Tabela 2 exibe o comparativo entre os tempos de classificação médio de cada modelo, considerando as diferentes cargas de trabalho. O Modelo 2 leva aproximadamente 30% mais de tempo se considerarmos a carga de trabalho de 1,0 e 14% para a carga de trabalho de 0,5. Ou seja, consome mais tempo de execução considerando as duas cargas de trabalho. No entanto, a partir da carga 0,4 o grande aumento na diferença está relacionado a exaustão dos recursos disponíveis no equipamento, como memória principal (ver Figura 9). Dessa forma, caso o usuário do dispositivo de IoT queira ter uma alta precisão da detecção de ataques e que a carga de trabalho seja moderada (maior que 0,4),

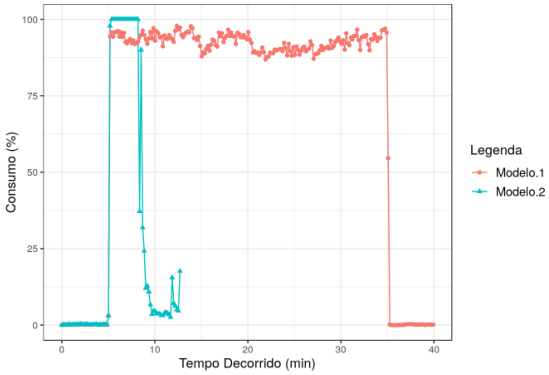

Figura 8. Comparativo do processador - carga 0,4

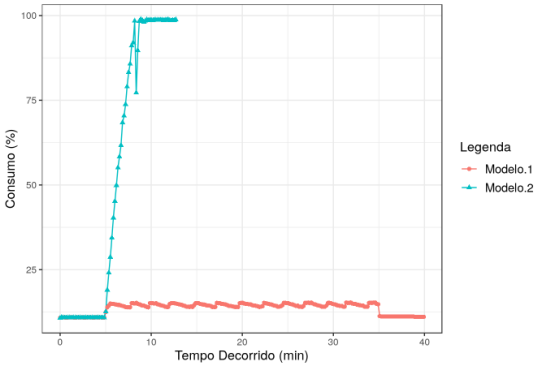

Figura 9. Comparativo da memória RAM - carga 0,4

o mais indicado é o Modelo 2. Por outro lado, caso o usuário necessite ter uma precisão moderada e uma alta carga de trabalho (menor que 0,4), o mais indicado é o Modelo 1.

Tabela 2. Comparativo entre a média do tempo de classificação.

Modelo	**Tempo médio de classificação (ms)**			
	Carga 1,0	**Carga 0,5**	**Carga 0,4**	**Carga 0,3**
Modelo 1	844,77	1226,73	1443,02	64427,44
Modelo 2	1155,51	1416,49	22526,71	21458,65

5.4. Limitações

Os modelos foram desenvolvidos utilizando algumas interfaces de programação como o Tensorflow 2.0, que necessita de um processador com arquitetura 64 bits. Dessa forma, foi necessário utilizar a Raspberry Pi 4 Model B para execução dos experimentos. Além disso, utilizamos o recurso de multiprocesso para executar as tarefas com o máximo de recursos possíveis. Isso foi necessário, pois verificamos que o *Global Interpreter Lock*[1] do Python não estava permitindo executar os modelos com *multithreads* e, consequentemente, utilizar do máximo os recursos disponíveis na raspberry. Adicionalmente, em nosso estudo, a carga de trabalho que escolhemos considera quatro níveis de intensidade, de forma que não pretendemos representar ataques cibernéticos, uma vez que o objetivo deste trabalho é analisar o desempenho do dispositivo IoT, e para esse objetivo, é preferível uma carga de trabalho fixa.

6. Conclusões e Trabalhos Futuros

Este trabalho apresentou uma avaliação de desempenho de dois modelos de *Deep Learning* para detecção de intrusão em um dispositivo IoT cuja função principal era classificar os dados de entrada como tráfego normal ou tráfego malicioso, inclusive identificando a subclasse caso o tráfego fosse malicioso. Os resultados obtidos através dos experimentos constataram que o segundo modelo consumiu mais recursos em relação ao primeiro,

[1]https://wiki.python.org/moin/GlobalInterpreterLock.

principalmente, quando cargas de trabalho mais intensas foram utilizadas. Considerando a carga de trabalho de 1,0, ambos os modelos obtiveram resultados satisfatórios, apesar do Modelo 2 requerer mais tempo e recursos de processador. Por outro lado, se considerarmos as cargas de trabalho de 0,5 e 0,4, obtivemos um consumo médio de 13,4% e 14,5% da memória RAM, assim como 59,1% e 93,9% do processador, respectivamente, para o Modelo 1. Já para o Modelo 2, obtivemos 14,7% e 64% de consumo da memória RAM e 72,2% e 100% de processador. Vale ser ressaltado que nessa comparação apenas o Modelo 2 com a carga de trabalho de 0,4 travou o dispositivo em 8 minutos após o início do experimento. Por fim, quando a carga de trabalho de 0,3 foi utilizada, em ambos os modelos, o ambiente travou em 8 e 6 minutos, respectivamente, para os Modelos 1 e 2. Esses resultados podem ajudar desenvolvedores de sistemas para IoT, assim como pesquisadores, a escolherem as melhores técnicas e configurações a fim de otimizar e garantir o melhor desempenho de tais dispositivos em ambientes de produção. Como trabalhos futuros, pretendemos avaliar os modelos de *Deep Learning* considerando ataques reais, coletar o consumo dos processos executados em cada equipamento e analisar diferentes *acurácias* para os modelos.

Agradecimentos

Esta pesquisa foi parcialmente financiada pelo CNPq - Brasil, processo 406263/2018-3.

Referências

Aloqaily, M., Otoum, S., Al Ridhawi, I., and Jararweh, Y. (2019). An intrusion detection system for connected vehicles in smart cities. *Ad Hoc Networks*, 90:101842.

Ashton, K. (1999). An introduction to the internet of things (iot). *RFID Journal*.

Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer networks*, 54(15):2787–2805.

Chen, M., Mao, S., and Liu, Y. (2014). Big data: A survey. *Mobile networks and applications*, 19(2):171–209.

de Araujo, R. B. (2003). Computação ubíqua: Princípios, tecnologias e desafios. In *XXI Simpósio Brasileiro de Redes de Computadores*, volume 8, pages 11–13.

Din, I. U., Guizani, M., Rodrigues, J. J., Hassan, S., and Korotaev, V. V. (2019). Machine learning in the internet of things: Designed techniques for smart cities. *Future Generation Computer Systems*, 100:826–843.

Dürr, O., Pauchard, Y., Browarnik, D., Axthelm, R., and Loeser, M. (2015). Deep learning on a raspberry pi for real time face recognition. In *Eurographics (Posters)*, pages 11–12.

Eremia, M., Toma, L., and Sanduleac, M. (2017). The smart city concept in the 21st century. *Procedia Engineering*, 181:12–19.

Farahnakian, F. and Heikkonen, J. (2018). A deep auto-encoder based approach for intrusion detection system. In *2018 20th International Conference on Advanced Communication Technology (ICACT)*, pages 178–183. IEEE.

Ge, M., Fu, X., Syed, N., Baig, Z., Teo, G., and Robles-Kelly, A. (2019). Deep learning-based intrusion detection for iot networks. In *2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 256–25609. IEEE.

Gómez, A., Cuiñas, D., Catalá, P., Xin, L., Li, W., Conway, S., and Lack, D. (2015). Use of single board computers as smart sensors in the manufacturing industry. *Procedia engineering*, 132:153–159.

Koroniotis, N., Moustafa, N., Sitnikova, E., and Turnbull, B. (2019). Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Generation Computer Systems*, 100:779–796.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Liao, H.-J., Lin, C.-H. R., Lin, Y.-C., and Tung, K.-Y. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24.

Lunt, T. F. (1993). A survey of intrusion detection techniques. *Computers & Security*, 12(4):405–418.

Martinez, F. J., Toh, C. K., Cano, J.-C., Calafate, C. T., and Manzoni, P. (2011). A survey and comparative study of simulators for vehicular ad hoc networks (vanets). *Wireless Communications and Mobile Computing*, 11(7):813–828.

Miraz, M. H., Ali, M., Excell, P. S., and Picking, R. (2015). A review on internet of things (iot), internet of everything (ioe) and internet of nano things (iont). In *2015 Internet Technologies and Applications (ITA)*, pages 219–224. IEEE.

Morabito, R. (2017). Virtualization on internet of things edge devices with container technologies: a performance evaluation. *IEEE Access*, 5:8835–8850.

Nurse, J. R., Creese, S., and De Roure, D. (2017). Security risk assessment in internet of things systems. *IT professional*, 19(5):20–26.

Pannu, A. (2015). Artificial intelligence and its application in different areas. *Artificial Intelligence*, 4(10):79–84.

Quincozes, S. E., Kazienko, J. F., and Copetti, A. (2018). Avaliação de conjuntos de atributos para a detecção de ataques de personificação na internet das coisas. In *Anais Estendidos do VIII Simpósio Brasileiro de Engenharia de Sistemas Computacionais*. SBC.

Tama, B. A. and Rhee, K.-H. (2017). Performance evaluation of intrusion detection system using classifier ensembles. *International Journal of Internet Protocol Technology*, 10(1):22–29.

Thamilarasu, G. and Chawla, S. (2019). Towards deep-learning-driven intrusion detection for the internet of things. *Sensors*, 19(9):1977.

Vinayakumar, R., Alazab, M., Soman, K., Poornachandran, P., Al-Nemrat, A., and Venkatraman, S. (2019). Deep learning approach for intelligent intrusion detection system. *IEEE Access*, 7:41525–41550.

Xu, P., Shi, S., and Chu, X. (2017). Performance evaluation of deep learning tools in docker containers. In *2017 3rd International Conference on Big Data Computing and Communications (BIGCOM)*, pages 395–403. IEEE.