Introduzindo a qualidade de imagem como uma nova condição de particionamento de DNN na borda

Roberto G. Pacheco, Rodrigo S. Couto *

[1]Universidade Federal do Rio de Janeiro - PEE/COPPE/GTA

{pacheco,rodrigo}@gta.ufrj.br

Abstract. *In Deep Neural Network (DNN) applications with edge computing, it is possible to process the first DNN layers at the edge and the other layers at the cloud. For image classification, DNNs can infer a high-quality image with high confidence in the first DNN layers. Thus, depending on the image quality, we can avoid sending data to the cloud, reducing the inference time and network utilization. This paper evaluates, experimentally, how image quality impacts the inference time in edge computing infrastructures. In the employed scenario, this work shows that considering image quality reduces inference time up to 35.33%.*

Resumo. *Em aplicações de redes neurais profundas (DNNs) com computação na borda, é possível processar as primeiras camadas da DNN na borda e as demais camadas na nuvem. Quando DNNs são utilizadas para classificação de objetos em imagens de alta qualidade, o processo de inferência pode apresentar um nível de confiança satisfatório já nas primeiras camadas da DNN. Assim, dependendo da qualidade da imagem, é possível evitar o uso da nuvem, reduzindo o tempo de inferência e o uso da rede. Este trabalho avalia, de forma experimental, o impacto da qualidade de imagem no tempo de inferência em infraestruturas de computação na borda. No cenário utilizado, mostra-se que considerar a qualidade da imagem reduz o tempo de inferência em até 35,33%.*

1. Introdução

Os avanços recentes na área de Internet das Coisas (IoT) aprimoraram a capacidade em coletar dados utilizando dispositivos móveis, como *smartphones* e sensores espalhados por cidades inteligentes. Em diversas dessas aplicações não basta apenas coletar dados, sendo necessário reconhecer padrões, extrair conhecimento e inferir a partir da informação recebida. Dessa forma, muitas aplicações de IoT empregam técnicas de *Deep Learning*, como as redes neurais profundas (*Deep Neural Networks* - DNNs) [Yao et al., 2018]. As DNNs consistem em camadas de neurônios, nas quais cada neurônio produz uma saída não linear a partir de funções de ativação. A arquitetura de uma DNN é constituída de uma camada de entrada, um conjunto de camadas intermediárias e uma camada de saída. A camada de entrada recebe dados brutos, extrai atributos e os propaga pelas camadas intermediárias em direção à camada de saída. As camadas intermediárias extraem conjuntos de atributos e, por sua vez, a camada de saída produz o resultado da inferência.

*O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. O trabalho também foi financiado pela Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ), auxílios no. E-26/203.211/2017, E-26/201.833/2019 e E-26/010.002174/2019 , pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), auxílio no. 2015/24494-8, e pelo CNPq.

A inferência em uma DNN necessita de alto poder computacional [LeCun et al., 2015], o que impede a sua implementação em dispositivos de IoT de baixa capacidade. Nesse caso, a inferência pode ser realizada em uma infraestrutura de computação em nuvem. Contudo, dada a dependência de condições da Internet, o uso da nuvem pode inviabilizar aplicações que exijam alta responsividade. Além disso, a transferência contínua de dados à nuvem pode gerar sobrecarga na rede. Como uma alternativa à computação em nuvem, é possível realizar a inferência usando computação na borda [Satyanarayanan, 2017]. Nesse tipo de infraestrutura, os recursos computacionais estão na borda da Internet, ou seja, mais próximos dos dispositivos de IoT.

Os dispositivos utilizados na borda geralmente possuem capacidade computacional significativamente inferior à da nuvem. Assim, é necessário adotar estratégias para reduzir o tempo de processamento na borda. No caso específico de DNNs, é possível aplicar uma rede neural do tipo *BranchyNet* [Teerapittayanon et al., 2016]. Essa abordagem é baseada na ideia de que o processo de inferência pode terminar em camadas intermediárias. Quanto mais camadas são percorridas em uma DNN, menor tende a ser o nível de incerteza da classificação. Com base nisso, processos de inferência geralmente processam até a última camada da DNN, de forma a minimizar a incerteza. Já em uma *BranchyNet* há ramos de saída laterais. Em outras palavras, o processo de inferência pode terminar em camadas intermediárias, caso um determinado nível de incerteza já seja alcançado. Assim, um dispositivo de borda pode implementar uma *BranchyNet* para reduzir o número de camadas percorridas em uma inferência.

A saída antecipada na *BranchyNet* pode trazer uma incerteza maior na inferência. Entretanto, percorrer todas as camadas pode ser proibitivo para um dispositivo de borda. Para lidar com esse compromisso, é possível particionar a rede neural, limitando o número de camadas a serem executadas na borda [Ko et al., 2018, Kang et al., 2017]. Caso seja necessário executar mais camadas para diminuir a incerteza, os atributos de camadas intermediárias da DNN são enviados para a nuvem para serem processados pelas demais camadas. O particionamento da rede neural estabelece um compromisso entre o tempo de processamento das camadas no dispositivo de borda e o tempo de comunicação com a nuvem. Por um lado, mais camadas da rede neural executadas no dispositivo em borda aumenta o tempo de processamento, devido ao seu baixo poder computacional. Por outro lado, a execução das camadas na nuvem adiciona o tempo de comunicação no envio de dados à mesma.

A literatura atual propõe técnicas para lidar com o compromisso entre tempo de processamento e de comunicação. A ideia é escolher quais camadas devem ser executadas na borda e quais devem ser executadas na nuvem. Essas técnicas consideram apenas o processamento dos dispositivos na borda e tempo de envio de dados pela rede [Hu et al., 2019, Kang et al., 2017]. Entretanto, outros fatores podem influenciar a escolha. Por exemplo, quando DNNs são utilizadas para identificar objetos em imagens, a qualidade dessas imagens pode influenciar a quantidade mínima de camadas necessárias para a inferência. Uma imagem de alta qualidade pode atingir o nível desejado de incerteza nas primeiras camadas da *BranchyNet*, enquanto imagens de baixa qualidade podem exigir o processamento de todas as camadas. Em uma infraestrutura de computação na borda, isso pode impactar a decisão em enviar para a nuvem ou não. Assim, este trabalho tem o objetivo de mostrar o impacto da qualidade da imagem no tempo de inferência de DNNs

na borda. Por exemplo, mostra-se que imagens sem nenhuma distorção são classificadas em até 468,55 ms a menos do que imagens com alto nível de distorção, o que corresponde a uma redução de 35,33% do tempo de inferência. Além disso, em uma arquitetura *BranchyNet*, este trabalho mostra que quanto mais próximos da camada de entrada os ramos laterais estão posicionados, mais a qualidade da imagem impacta a acurácia da inferência.

Este trabalho está organizado da seguinte forma. Primeiramente, são apresentados trabalhos relacionados na Seção 2 e conceitos básicos de BranchyNet na Seção 3. Já a Seção 4 descreve os diferentes cenários de processamento de DNNs considerados no trabalho e analisa o particionamento de DNN sem distorção na imagem. Em seguida, a Seção 5 avalia o impacto da qualidade da imagem na acurácia e no tempo de inferência. Por fim, a Seção 6 apresenta conclusões e trabalhos futuros.

2. Trabalhos Relacionados

Diversos trabalhos abordam o particionamento de DNNs entre o dispositivo em borda e a nuvem, para acelerar o tempo de inferência. [Kang et al., 2017, Li et al., 2018] propõem medir o tempo de processamento na borda e na nuvem para diversos tipos e configurações de camadas neurais, e assim gerar um modelo preditivo para estimar o tempo de processamento na borda e nuvem. Então, esses modelos são combinados com as condições da rede para selecionar dinamicamente a camada de particionamento. Contudo, esses trabalhos limitam-se a redes com topologia em cadeia, como AlexNet [Krizhevsky et al., 2012] e SqueezeNet [Iandola et al., 2016]. Visando um estudo aplicável a a qualquer rede neural, [Hu et al., 2019] encontra um particionamento ótimo convertendo o problema de particionamento em um problema de corte mínimo. Tais trabalhos objetivam minimizar o tempo de inferência, já [Xu et al., 2019] particiona a DNN para minimizar o consumo de energia no contexto de dispositivos *wearables*.

Os trabalhos citados consideram dois fatores para particionar a DNN: a capacidade de processamento na borda e nuvem, além das condições da rede. Assim, este trabalho contribui para o estado da arte, pois mostra que a qualidade da imagem também deve ser considerada, pois impacta tanto o particionamento da DNN quanto as decisões de saída da BranchyNet, influenciando o tempo de inferência.

3. BranchyNet

Ao contrário de uma DNN padrão, que contém somente um ponto de saída, uma BranchyNet possui ramos de saída laterais, inseridos em diversas camadas intermediárias [Teerapittayanon et al., 2016]. A Figura 1 apresenta uma B-SqueezeNet, que consiste em uma Branchynet composta da arquitetura original de uma SqueezeNet [Iandola et al., 2016], como ramo principal, e dois ramos laterais inseridos em suas camadas intermediárias. No exemplo dessa figura, o processo de inferência consiste em analisar uma imagem e decidir se é uma foto de um gato ou de um cão. Apesar de haver mais camadas, a Figura 1 ilustra, por simplicidade, apenas as camadas convolucionais, as *fully-connected* (fc) e os módulos *fire*. As camadas convolucionais consistem em um conjunto de filtros, cujos componentes são parâmetros a serem aprendidos durante o processo de treinamento. Cada filtro é responsável por gerar um conjunto de atributos de saída a partir de um conjunto de atributos de entrada, utilizando operações de convolução. O conjunto das camadas *fully-connected* de saída recebe os atributos extraídos das camadas

convolucionais anteriores e gera uma probabilidade da amostra pertencer a cada classe. No exemplo, uma imagem pode pertencer à classe cão ou gato. Nota-se que o primeiro ramo lateral é composto por duas camadas convolucionais (*conv*), seguida de uma camada *fully-connected* (fc), enquanto o segundo ramo lateral possui apenas uma camada convolucional e uma *fully-connected*. Por fim, o ramo principal é composto por uma camada convolucional, seguida de oito módulos *fire*, sendo cada módulo composto por três camadas convolucionais [Iandola et al., 2016].

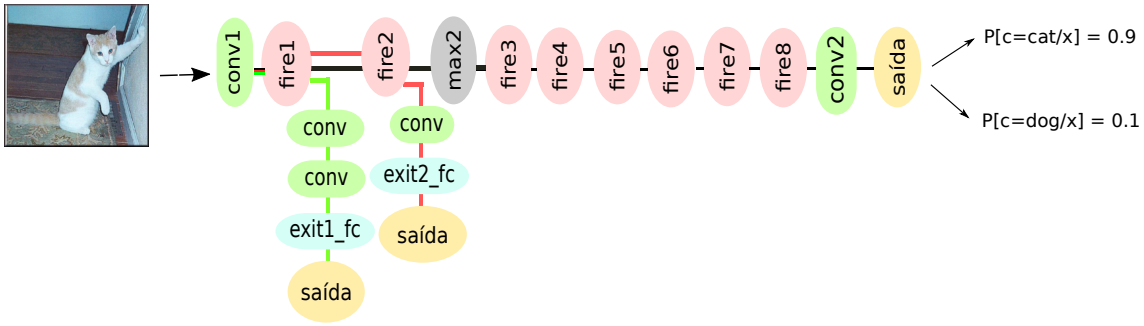


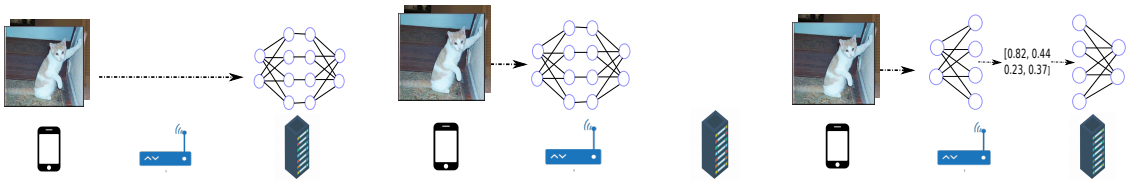
Figura 1. Uma B-SqueezeNet com dois ramos laterais inseridos no ramo principal.

A DNN já treinada recebe uma imagem, que é processada pelas camadas até um dos pontos de saída da Figura 1. Em um dado ramo lateral, a camada *fc* gera a probabilidade da amostra pertencer a cada das classes. A partir dessas probabilidades, calcula-se o nível de confiança da classificação da amostra baseando-se em uma medida de incerteza da classificação, como a entropia. Por exemplo, uma imagem com baixa qualidade, como a da Figura 5(d), tende a apresentar entropia superior a uma de alta qualidade, como a da Figura 5(a). Se a entropia for inferior a um limiar, então a inferência termina e a amostra é rotulada conforme a classe que possuir a maior probabilidade. Dessa forma, essa amostra não é mais processada por nenhuma camada posterior, reduzindo o número de camadas processadas, o que diminui o tempo de processamento. Caso contrário, a amostra segue sendo processada pelas próximas camadas até alcançar o próximo ramo lateral e repetir o procedimento.

4. Análise do Tempo de Inferência de Imagens sem Distorção

Todos os experimentos deste trabalho são baseados em DNNs treinadas para inferir se um animal em uma imagem é um cão ou um gato. Apesar de essa não ser uma tarefa com importância prática, sua simplicidade permite o controle dos cenários para realizar as análises desejadas. Para o treinamento, utiliza-se um *dataset* composto de imagens de cães e gatos em diferentes ambientes sem nenhum tipo de distorção da qualidade das imagens [Parkhi et al., 2012]. Para treinar as redes neurais, divide-se o *dataset* em um conjunto de treinamento com 20.000 amostras, além de um conjunto de validação e um outro de testes com 2.500 amostras cada. É importante ressaltar que o *dataset* é balanceado, ou seja, o número de imagens rotuladas como gatos é aproximadamente igual às de cães. Os experimentos deste trabalho são avaliados em três diferentes cenários ilustrados na Figura 2: processamento exclusivamente na nuvem, exclusivamente na borda e baseado em particionamento de camadas entre a borda e a nuvem. Nos cenários de processamento apenas na borda e baseado em particionamento, implementam-se redes

neurais B-SqueezeNet. Quando o processamento ocorre somente na nuvem, utiliza-se a DNN AlexNet [Krizhevsky et al., 2012], que consiste em uma DNN sem ramos laterais que atinge uma acurácia maior do que a B-SqueezeNet, mas também com maior demanda computacional. A acurácia da DNN consiste na porcentagem de inferências realizadas corretamente dado um conjunto de amostras. Por exemplo, dadas diversas imagens de cães e gatos, a acurácia da DNN corresponde à porcentagem de imagens nas quais a inferência indicou corretamente qual é o animal da foto.



(a) Somente na nuvem. (b) Somente na borda. (c) Com particionamento.

Figura 2. Cenários de processamento da DNN.

Como dispositivo de borda este trabalho utiliza o Raspberry Pi 3 Model B+, pois trata-se de uma plataforma comumente utilizada para esse fim na literatura [Hu et al., 2019, Zhang et al., 2018]. O Raspberry Pi 3 Model B+ é equipado com um processador quad-core ARMv8 1.2 GHz com 1 GB de RAM. Em relação à nuvem, utiliza-se o Google Colaboratory [1]. O Google Colaboratory trata-se de um serviço de computação em nuvem para aprendizado de máquina. Nessa plataforma, utiliza-se uma máquina virtual equipada com processador Intel 2-core Xeon(R)@ 2.20GHz e 12 GB VRAM e uma GPU Tesla K80. O dispositivo de borda e a nuvem são utilizados para medir o tempo de processamento da DNN e realizar a inferência. O tempo de comunicação entre a borda e a nuvem é simulado a partir de experimentos com o iPerf e informações da literatura, como visto mais adiante. É válido ressaltar que todas as redes neurais utilizadas nos experimentos são previamente treinadas na nuvem. Em relação às ferramentas de *software*, os experimentos deste trabalho são escritos em Python 3, utilizando a biblioteca Pytorch [2], que é uma solução de código aberto desenvolvida pelo Facebook para executar técnicas de *deep learning*.

Os diferentes cenários utilizados neste trabalho são detalhados a seguir, juntamente com análises preliminares. Em todo o trabalho, os resultados apresentados correspondem a médias de diversas amostras com o intervalo de confiança de 95%.

4.1. Processamento exclusivamente na nuvem ou na borda

No cenário de processamento exclusivamente na nuvem, todas as camadas da DNN são executadas em um servidor em nuvem, como ilustrado na Figura 2(a). Para esse cenário, o tempo de inferência é a soma do tempo de comunicação com o tempo de processamento. O tempo de comunicação é o tempo de envio da imagem da borda para a nuvem. O tempo de processamento, por sua vez, é o tempo decorrido entre a recepção da imagem e seu processamento por todas as camadas da DNN na nuvem. O tempo de processamento depende da capacidade computacional do servidor da nuvem, enquanto

[1] `https://colab.research.google.com`
[2] `https://pytorch.org/`

o tempo de comunicação é dependente das condições da rede e das tecnologias de rede de acesso utilizadas. Dessa forma, com objetivo de avaliar o tempo de inferência nesse cenário, adotam-se taxas de envio correspondentes a 1,1 Mbps e 18,8 Mbps para 3G e Wi-Fi, respectivamente. As taxas de envio utilizadas neste trabalho são baseadas nos valores apresentados em [Hu et al., 2019].

Para medir o tempo de comunicação entre a borda e a nuvem, seleciona-se uma imagem pertencente ao *dataset*, cujo tamanho é de 748,5 KB sem qualquer compressão. Em seguida, envia-se essa quantidade de dados de um PC no Rio de Janeiro a diversas localidades ao redor do mundo para simular o cenário de nuvens distribuídas geograficamente. Para tal, utiliza-se a ferramenta iPerf3[3], a qual oferece servidores iPerf gratuitos e geograficamente distribuídos. Os valores das taxas do 3G e Wi-Fi são utilizados para limitar a taxa enviada pelo cliente iPerf (isto é, parâmetro "-b" da ferramenta). O tamanho da imagem é utilizado para limitar o tamanho do arquivo enviado (isto é, parâmetro "-n" da ferramenta). A Tabela 1 mostra os resultados de tempo de comunicação para cinco servidores iPerf espalhados pelo mundo. A partir desses resultados, os demais experimentos consideram os casos extremos: EUA e Rússia. É importante frisar que, neste trabalho, a taxa escolhida para o 3G e Wi-Fi é inferior à banda disponível entre o PC do Rio de Janeiro e os servidores iPerf. Ou seja, o gargalo é a rede de acesso da borda.

Servidor	País	Tempo de Envio (s)
iperf.he.net	EUA	1,029
iperf.worldstream.nl	Nova Zelândia	1,024
speedtest.wtnet.de	Alemanha	1,259
bouygues.iperf.fr	França	1,263
speedtest.hostkey.ru	Rússia	1,435

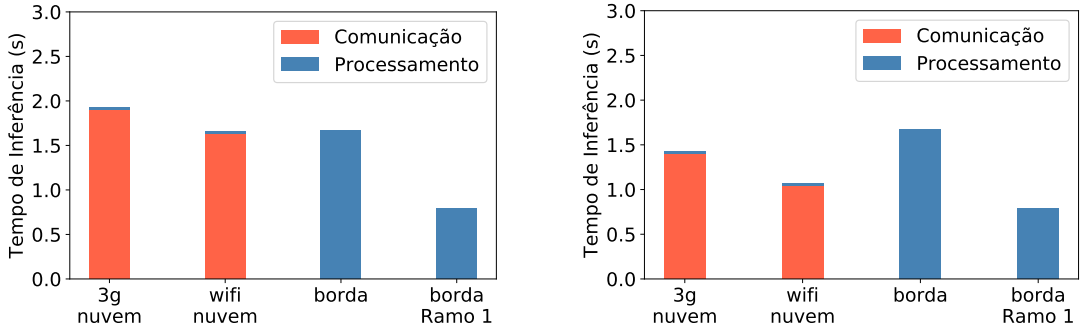
Tabela 1. Tempo de envio de 748.5KB para os servidores iPerf.

Em relação ao cenário de processamento exclusivamente na borda da Figura 2(b), o tempo de inferência corresponde apenas ao tempo de processamento das camadas na borda, sem necessidade de envio para a nuvem. Esse tempo é então influenciado apenas pela capacidade computacional do Raspberry Pi 3. Para comparar o tempo total de inferência nos cenários de processamento somente na nuvem ou na borda, as Figuras 3(a) e 3(b) apresentam os tempos de comunicação e processamento e, consequentemente, o tempo de inferência com diferentes taxas de envio, considerando nuvens localizadas na Rússia e EUA, respectivamente. No caso da borda, apresentam-se resultados nos quais todas as camadas são percorridas e o caso no qual a inferência termina no primeiro ramo.

De forma geral, a Figura 3 mostra que o tipo de rede e sua respectiva taxa de envio representam um fator decisivo para determinar a estratégia de processamento adotada [Hu et al., 2019]. Em relação ao tempo total de inferência, é possível notar na Figura 3(a) que implementar apenas um dispositivo em borda não soluciona o problema de reduzir o tempo de inferência, pois o tempo de processamento na borda praticamente equivale ao tempo de envio de uma imagem bruta à nuvem, além de utilizar uma DNN com menor acurácia. A solução de inserir um ramo lateral consegue reduzir significativamente o tempo de inferência, contudo, reduz também a acurácia do modelo, como mostrado mais

[3] `https://iperf.cc/`

adiante na Figura 6. Assim, é necessário adotar o particionamento de DNNs para lidar com esse compromisso.

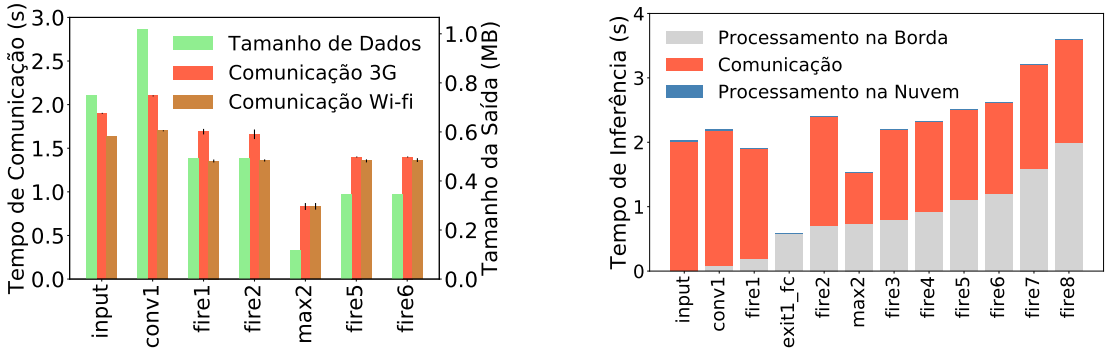


(a) Nuvem na Rússia. (b) Nuvem nos EUA.

Figura 3. Tempos de inferência na nuvem e na borda.

4.2. Processamento Baseado em Particionamento de DNN

No cenário de particionamento, a quantidade de dados enviada à nuvem corresponde aos dados de saída da última camada processada na borda, denominada de camada de particionamento. A Figura 4(a) mostra o tempo de comunicação e tamanho dos dados enviados para alguns exemplos de particionamento. Na Figura 4(a), cada barra verde (isto é, a mais à esquerda) representa o tamanho dos dados enviados à nuvem caso uma dada camada seja escolhida para particionamento. Além das barras verdes, a figura também mostra barras relacionadas ao tempo de comunicação no envio de dados em cada camada. Os experimentos consideram a nuvem na Rússia para enfatizar o pior caso de tempo de comunicação. O primeiro conjunto de barras, denominado *input*, corresponde a enviar a imagem bruta de 748,5 KB. Nota-se que a primeira camada convolucional (*conv1*) aumenta consideravelmente o tamanho de dados a serem enviados à nuvem [Kang et al., 2017]. Portanto, não é desejável particionar a B-SqueezeNet logo na primeira camada convolucional. Contudo, supondo um particionamento logo após a camada *max2*, os resultados mostram uma redução significativa na quantidade de dados enviados.



(a) Tempo de comunicação e tamanho da saída. (b) Tempo de inferência para rede 3G.

Figura 4. Análise da nuvem da Rússia para diferentes camadas de particionamento.

No cenário de particionamento, o tempo de inferência corresponde à soma dos seguintes fatores: tempo de processamento na borda das camadas da DNN até a de parti-

cionamento, tempo de comunicação e tempo de processamento das camadas restantes na nuvem. Para avaliar o tempo de inferência, implementa-se a rede neural B-SqueezeNet no Raspberry Pi. Já na nuvem implementa-se a SqueezeNet, ou seja, apenas o ramo principal da rede neural. Considera-se o uso de rede 3G e nuvem na Rússia, para uma análise de pior caso. Os resultados são mostrados na Figura 4(b), na qual cada barra corresponde a uma decisão de particionamento diferente. Por exemplo, a barra correspondente à camada *fire1* refere-se ao processamento dessa camada e das anteriores no dispositivo em borda e as demais na nuvem. Consequentemente, o tempo de comunicação nesse caso corresponde ao tempo de comunicação da rede 3G da camada *fire1* na Figura 4(a). Já a barra da camada *input* corresponde à decisão de particionamento em enviar a imagem bruta à nuvem, portanto, trata-se do cenário de processamento exclusivamente na nuvem. Ressalta-se que a Figura 4(b) mostra todas as possíveis decisões de particionamento sem ramos laterais, que apresentam uma acurácia de 92,2%. A camada *exit1_fc*, por sua vez, corresponde ao cenário com um ramo lateral, que resulta em acurácia de 83,33%. Nesse caso, configura-se o limiar de entropia como 1, ou seja, as amostras sempre são classificadas pelo ramo lateral. Os resultados da Figura 4(b) mostram que a escolha do particionamento pode afetar significativamente o tempo de inferência. Nota-se que o tempo de processamento na nuvem é desprezível em todos os casos. De todas as camadas, é possível observar que o menor tempo de inferência ocorre na camada *exit1_fc*. Entretanto, esse caso corresponde a uma acurácia inferior às demais, dada a saída obrigatória no ramo lateral. Caso fosse necessária uma maior acurácia, deveriam ser adotadas estratégias de particionamento. Para o cenário estudado, a melhor estratégia é particionar em *max2*. Ou seja, torna-se mais interessante aguardar o processamento na borda nas camadas iniciais até alcançar a *max2* para, então, enviar à nuvem.

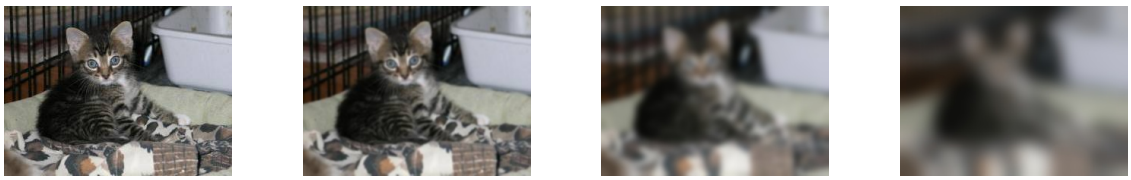
5. Análise do Impacto da Qualidade da Imagem

Esta seção avalia, primeiramente, o efeito da qualidade da imagem na acurácia em uma rede neural B-SqueezeNet, implementada na borda, e uma AlexNet, implementada na nuvem. Em seguida, os experimentos mostram o impacto da qualidade da imagem no tempo de inferência. Para tanto, esses experimentos são implementados nos três cenários apresentados na Seção 4, utilizando o mesmo conjunto de imagens.

5.1. Impacto da Distorção na Acurácia

Em aplicações IoT com câmeras, os dispositivos, por serem normalmente de baixo custo, eventualmente capturam imagens com diversos tipos de distorção, como contraste, ruído, e *blur*. Essas distorções impactam a qualidade da imagem e, consequentemente, o desempenho das DNNs [Dodge e Karam, 2016]. Este trabalho avalia os efeitos do *blur* em DNNs, o qual atua como um filtro passa-baixa na imagem, atenuando detalhes em alta frequência. O *blur* pode emular distorções ocorridas devido à distância focal do objeto de interesse, ou ao movimento desse objeto em relação à câmera ou vice-versa. Esse tipo de distorção pode ocorrer mesmo em câmeras de alta resolução. Por exemplo, em aplicações de identificação de placas de trânsito em veículos inteligentes, o movimento da câmera veicular pode causar distorção similar ao *blur*. A inserção de *blur* em uma imagem é implementada pela aplicação da operação de convolução entre a imagem original e um filtro. Este trabalho utiliza um filtro gaussiano, variando as suas dimensões, enquanto mantém-se a variância da imagem original. Em imagens digitais, o valor de um *pixel* tende

a estar correlacionado a seus vizinhos. Assim, aplicando filtros com dimensões maiores, altera-se valores de *pixels* cada vez mais distantes e, por sua vez, menos correlacionados. Dessa forma, filtros com dimensões maiores resultam em imagens com maior nível de *blur*. Utilizando o conjunto de imagens da Seção 4, aplicam-se diferentes níveis de *blur* (isto é, dimensões do filtro) nas imagens do conjunto testes, resultando em diversos *datasets* com imagens distorcidas. Um exemplo de diferentes níveis aplicados é apresentado na Figura 5.



(a) Nível = 0. (b) Nível = 5. (c) Nível = 15. (d) Nível = 65.

Figura 5. Exemplos de imagem com diferentes níveis de *blur*.

Para analisar o impacto do *blur* na acurácia, implementa-se uma B-SqueezeNet no Raspberry Pi da borda e uma AlexNet na nuvem. Como na Seção 4, ambas as redes neurais são treinadas com um conjunto de imagens sem distorção. Na inferência, aplica-se o conjunto de imagens distorcidas para avaliar a acurácia nas duas redes neurais. Em relação à DNN implementada na borda, avalia-se a acurácia dos ramos laterais e do ramo principal. Para avaliar o primeiro ramo lateral, define-se o valor do limiar de entropia como 1, para que todas as amostras sejam classificadas nesse ramo. Já para o segundo ramo, o limiar de entropia do primeiro ramo é configurado como 0 e o segundo limiar como 1, assim todas as amostras do conjunto de dados são classificadas pelo segundo ramo lateral. Por fim, para o ramo principal, os limiares de entropia dos dois ramos laterais presentes são definidos como 0 para que todas as amostras sejam processadas por todas as camadas da rede neural. É válido frisar que a acurácia independe de condições da rede, então essa análise não considera diferentes tecnologias de acesso ou localidade da nuvem.

A Figura 6 apresenta a acurácia para diferentes níveis de *blur* para as B-SqueezeNet da borda e a AlexNet da nuvem. A Figura 6 mostra que a acurácia das redes neurais é sensível ao aumento do nível de blur [Dodge e Karam, 2016]. Além disso, a figura mostra que o primeiro ramo é ainda mais sensível do que o segundo, que, por sua vez, é mais sensível ao *blur* que o ramo principal. Esse comportamento pode ser explicado por duas razões. A primeira deve-se ao *blur* remover texturas da imagem, as quais são as características utilizadas para classificar a imagem nas primeiras camadas [Dodge e Karam, 2016]. A segunda razão consiste no fato de que quanto menos camadas são percorridas em uma DNN, maior tende a ser o nível de incerteza da classificação e, consequentemente, há uma maior sensibilidade ao *blur*. Dessa forma, é possível estabelecer uma hierarquia de sensibilidade, na qual a acurácia fornecida pelos ramos posicionados mais próximos da camada de entrada é mais sensível do que a de ramos mais próximos da camada de saída do ramo principal. Os resultados apresentados a seguir mostram como é possível reduzir o tempo de inferência se a aplicação aceita uma redução de acurácia.

5.2. Impacto da Distorção no Tempo de Inferência

O objetivo deste experimento é demonstrar que considerar o nível de *blur* em uma imagem permite reduzir o tempo de inferência para a classificação da mesma. Para esse

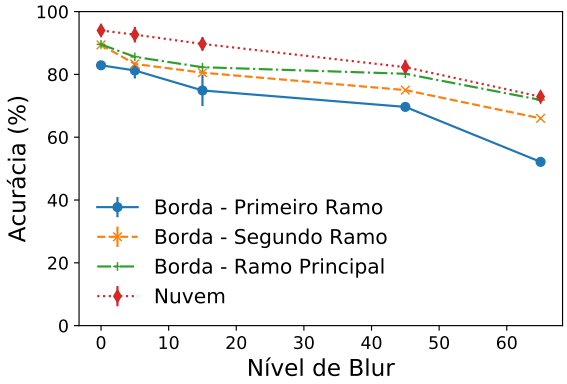


Figura 6. Acurácia na classificação das redes neurais B-SqueezeNet na borda e AlexNet na nuvem, utilizando diferentes níveis de *blur*.

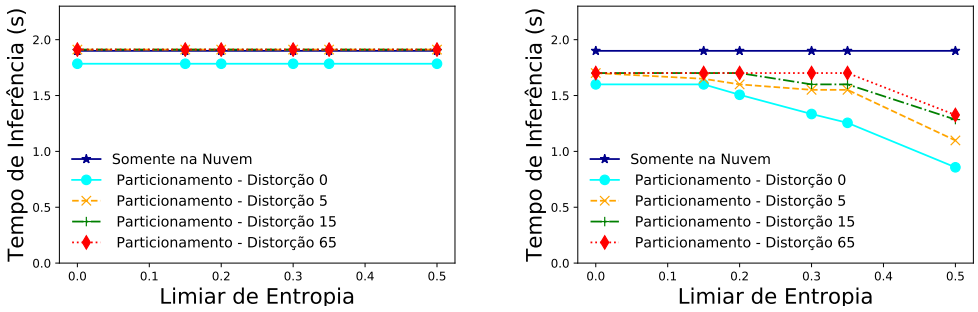
experimento, implementa-se uma rede neural B-SqueezeNet com somente um ramo lateral na borda e seu respectivo ramo principal na nuvem. A escolha de apenas um ramo lateral tem como objetivo facilitar a análise. Já em relação à posição do ramo lateral, o experimento utilizou o primeiro ramo posicionado após a camada *fire1*, como apresentado na Figura 1, para evitar processamento desnecessário na borda. Diferentemente da Seção 5.1, o limiar de entropia do ramo é ajustado em valores menores que 1 e, assim, a amostra pode não ser classificada no ramo lateral. Caso isso ocorra, as amostras são processados por camadas posteriores e seguindo o particionamento de DNNs descrito na Seção 4, no qual parte da rede neural é executada na borda e a outra na nuvem. Os resultados também apresentam o cenário de processamento somente na nuvem (isto é, com a AlexNet) para fins comparativos. Nesse caso, não há distorção nas imagens mas os resultados se aplicam a imagens com qualquer nível de blur, já que não há ramos laterais. A partir do cenário de particionamento, o experimento avalia o tempo de inferência em cada conjunto de imagens com três níveis de *blur*: 5, 15 e 65. Esses níveis são escolhidos para verificar o comportamento para imagens com distorção baixa, intermediária e alta, respectivamente. Além disso, os resultados apresentados a seguir também consideram o particionamento de DNN em imagens sem nenhum *blur* (isto é, nível 0) para fins comparativos.

A análise realizada consiste em dividir cada conjunto de imagens distorcidas em lotes compostos por 48 imagens. Então, insere-se tais lotes na rede neural e mede-se o tempo de inferência médio de cada lote. Como visto na Seção 3, para que uma dada imagem seja classificada no ramo lateral, essa deve atender a um critério de confiança. Caso atenda, o tempo de inferência corresponde ao tempo de processamento na borda das camadas anteriores ao ramo lateral e ao das camadas pertencentes ao ramo lateral. Caso não atenda, a imagem segue sendo processada pelas próximas camadas até a camada de particionamento, a partir da qual os dados de saída dessa camada são enviados à nuvem. Nesse caso, o tempo de inferência é a soma do tempo de processamento na borda até a camada de particionamento, do tempo de comunicação no envio dos dados de saída dessa camada para a nuvem e do tempo de processamento das camadas restantes na nuvem.

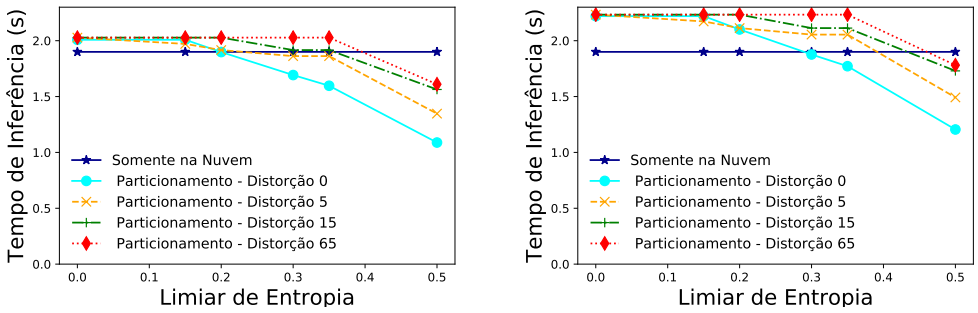
Dado o exposto, o experimento avalia o tempo de inferência para o particionamento após cada camada da rede neural, variando os valores de limiar de entropia entre 0 e 0,5. Este experimento avalia dois casos extremos de condições da nuvem, considerando as

medidas de rede da Seção 4.1. No pior caso, apresentado na Figura 7, utiliza-se uma rede de acesso 3G e uma nuvem na Rússia. No melhor caso, apresentado na Figura 8, utiliza-se uma rede de acesso Wi-Fi e uma nuvem nos EUA. Os gráficos da Figura 7 apresentam os resultados considerando o particionamento após as camadas *conv1*, *fire2*, *fire5* e *fire6*. Essas camadas foram escolhidas para analisar o comportamento do tempo de inferência em camadas anteriores aos ramos (*conv1*), camada posterior mais próxima (*fire2*) e camadas posteriores mais distantes do ramo lateral (*fire5* e *fire6*). Já a Figura 8, por questões de concisão, apresenta apenas os resultados de particionamento após as camadas *fire2* e *fire5*.

Considerando o cenário de rede 3G e o servidor localizado na Rússia, a Figura 7(a) mostra o tempo de inferência de imagens com diferentes níveis de *blur* para o caso de particionamento após a camada *conv1*. Nesse caso, os tempos de inferência de imagens com diferentes níveis de *blur* são iguais e não dependem do limiar de entropia do ramo lateral. Isso ocorre, pois, como o particionamento é realizado antes do ramo lateral, então nenhuma amostra é processada e classificada nele e a inferência sempre termina na nuvem. Como a DNN da nuvem não possui ramos laterais, o tempo de inferência não é impactado. Nota-se que o cenário sem particionamento (isto é, o "Somente nuvem") possui um tempo de inferência menor, mesmo executando a AlexNet, que é uma rede mais complexa do que a B-SqueezeNet. Isso ocorre, pois, como mostra a Figura 4(a), enviar os dados brutos é menos custoso do que enviar dados após a camada *conv1*.



(a) Particionamento na camada *conv1*. (b) Particionamento na camada *fire2*.



(c) Particionamento na camada *fire5*. (d) Particionamento na camada *fire6*.

Figura 7. Tempo de inferência em função da distorção, para diferentes camadas de particionamento, utilizando uma nuvem na Rússia e rede 3G.

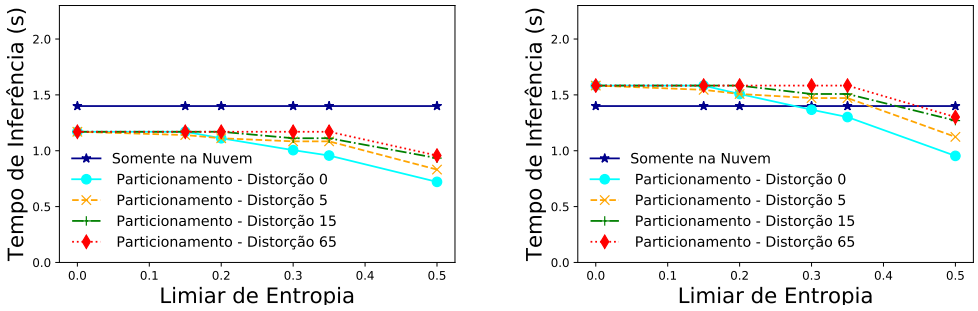
De forma geral, a Figura 7 mostra que limiares de entropia cada vez maiores reduzem o tempo de inferência para todos os níveis de distorção da imagem. Esse comporta-

mento ocorre porque quanto maior o limiar de entropia, mais imagens atendem ao critério de confiança e assim são classificadas no primeiro ramo lateral, portanto não são enviadas à nuvem. Isso está de acordo com os resultados da Figura 4, que mostra que o tempo de inferência para classificar uma imagem no primeiro ramo lateral (isto é, o *exit1_fc*) é inferior em comparação ao tempo de inferência com particionamento após qualquer camada e nas duas redes de acesso avaliadas. Para a um lote de imagens, a classificação antecipada no ramo lateral de mais amostras resulta na redução do tempo de inferência médio do lote.

Em relação ao particionamento na camada *fire2*, os resultados da Figura 7(b) mostram que o tempo de inferência de imagens com nível de *blur* intermediário (isto é, 15) e alto (isto é, 65) coincidem até um determinado limiar de entropia, porque nenhuma das imagens do conjunto de dados foi suficientemente confiável para ser classificada no primeiro ramo lateral. Contudo, a partir de um dado limiar, as imagens com alto nível de *blur* seguem sem nenhuma amostra sendo classificada no ramo lateral. Já na curva de nível intermediário, há amostras sendo classificadas no primeiro ramo, o que resulta em redução do tempo de inferência. É importante notar que a classificação antecipada reduz a acurácia, como mostrado na Figura 6. Ou seja, há um compromisso entre redução de tempo de inferência e perda de acurácia. A Figura 7(b) mostra que, em redes 3G, o tempo de inferência de imagens com baixo nível de *blur* é até 17,2% inferior em comparação a imagens com alto nível de *blur*, o que corresponde a 228,09 ms. Já em imagens sem nenhum *blur* (isto é, 0), o tempo de inferência é até 21,9% e 35,33% inferior em comparação com as imagens de baixo e alto nível de *blur*, respectivamente. Em relação ao cenário de processamento somente na nuvem, as imagens de baixo nível de *blur* são classificadas 34,25% mais rápidas. Como imagens com níveis de *blur* diferentes possuem tempo de inferência diferentes, é possível realizar uma estratégia de particionamento ciente da qualidade. Ou seja, caso a qualidade a imagem seja conhecida e o nível mínimo de acurácia definido, ajusta-se o limiar de entropia para obter um tempo desejado de inferência.

Em relação ao particionamento na camada *fire5*, a Figura 7(c) mostra que é mais vantajoso enviar uma imagem à nuvem para ser classificada até o limiar de entropia de 0,15, independentemente do nível de *blur* da imagem. A partir desse limiar, é mais vantajoso adotar a estratégia de particionamento para imagens de alta qualidade, ou seja, sem *blur*. Entretanto, em relação a imagens com baixo e intermediário nível de *blur* ainda é mais vantajoso processar a DNN apenas na nuvem até o limiar de entropia de 0,3, a partir do qual torna-se mais vantajoso o particionamento. Assim, esse resultado reafirma que a qualidade da imagem é importante no particionamento. Por exemplo, se for detectado que uma imagem tem baixa qualidade, em alguns casos é desejável enviá-la diretamente para a nuvem ao invés de utilizar o particionamento. Finalmente, considerando o particionamento na camada *fire6*, a Figura 7(d) mostra que para limiares de entropia inferiores a 0,35 a estratégia de processamento com o menor tempo de inferência é o processamento somente na nuvem. Nessa camada, em relação a imagens com alto nível de *blur*, a estratégia de particionamento só é vantajosa para limiar de entropia de 0,5, o que pode causar uma redução de acurácia significativa, inviabilizando algumas aplicações.

Considerando uma rede Wi-Fi e o servidor localizado nos EUA, a Figura 8(a) mostra que no particionamento após a camada *fire2* é mais vantajosa a abordagem de particionamento em comparação a processar a imagem apenas na nuvem, independente do nível de *blur* existente na imagem. Considerando particionamento, essa figura mostra que o



(a) Particionamento na camada *fire2*. (b) Particionamento na camada *fire5*.

Figura 8. Tempo de inferência em função da distorção, para diferentes camadas de particionamento, utilizando uma nuvem nos EUA e rede Wi-Fi.

tempo de inferência de imagens sem *blur* é até 48,47% inferior em comparação ao processamento apenas na nuvem e até 31,64% para imagens com maior nível de distorção. Em relação ao particionamento após a camada *fire5*, a Figura 8(b) mostra que, diferentemente da Figura 7(c), apenas a partir do limiar de entropia 0,3 é mais vantajoso utilizar o particionamento na borda para imagens sem distorção. Na nuvem da Rússia, para a camada *fire5*, isso ocorre a partir do limiar 0,2. Ou seja, para o cenário de rede Wi-Fi e servidor nos EUA com limiar de entropia de 0,2, a melhor opção é processar na nuvem, enquanto na Rússia com rede 3G é mais vantajoso o processamento baseado em particionamento. Assim, é possível notar que as condições da rede também devem ser consideradas para ajustar o limiar de entropia. O emprego de maior poder computacional na borda fará seu tempo de processamento se aproximar ao da nuvem. Assim, mais camadas serão processadas na borda e, no melhor cenário, o processamento pode ser realizado completamente nesse local. Neste trabalho, considerou-se baixo poder computacional na borda, de forma a analisar o pior caso, no qual o particionamento é essencial.

6. Conclusões e trabalhos futuros

Este trabalho avaliou o impacto da qualidade da imagem no tempo de inferência para classificar imagens. Para isso, foram implementadas redes neurais na borda e na nuvem e comparou-se o tempo de inferência em imagens com diferentes níveis de *blur* nos cenários de processamento somente na borda ou somente na nuvem, como também em um cenário de particionamento no qual a borda e a nuvem são usadas em conjunto. A partir dos resultados obtidos, foi possível concluir que imagens com baixo nível de *blur* tendem a resultar em uma incerteza de classificação inferior em comparação a imagens com maior nível de *blur*. Dessa forma, mais imagens com baixo nível de *blur* podem ser classificadas no ramo lateral e, consequentemente, na borda da rede. Isso resulta em uma redução do tempo de inferência. Portanto, no cenário avaliado, um método de particionamento que ajuste o limiar de entropia de acordo com a qualidade da imagem, permite reduzir o tempo de inferência em até 35,33%. Assim, este trabalho introduz a qualidade da imagem como um novo fator a ser considerado em propostas de particionamento de DNNs, além da abordagem comum da literatura de utilizar o tempo de processamento na borda e as condições da rede. Em trabalhos futuros, pretende-se implementar distorções provocadas por fenômenos reais, como efeitos da chuva e névoa. Além disso, pretende-se

desenvolver um método de otimização para determinar o limiar de entropia que melhor atenda ao compromisso entre acurácia e tempo de inferência.

Referências

Dodge, S. e Karam, L. (2016). Understanding how image quality affects deep neural networks. Em *IEEE International Conference on Quality of Multimedia Experience (QoMEX)*, p. 1–6.

Hu, C., Bao, W., Wang, D. e Liu, F. (2019). Dynamic adaptive DNN surgery for inference acceleration on the edge. Em *IEEE Conference on Computer Communications (INFOCOM)*, p. 1423–1431.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J. e Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and $<$ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*.

Kang, Y., Hauswald, J., Gao, C., Rovinski, A., Mudge, T., Mars, J. e Tang, L. (2017). Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. Em *ACM Computer Architecture News (SIGARCH)*, volume 45, p. 615–629.

Ko, J. H., Na, T., Amir, M. F. e Mukhopadhyay, S. (2018). Edge-host partitioning of deep neural networks with feature space encoding for resource-constrained internet-of-things platforms. Em *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, p. 1–6.

Krizhevsky, A., Sutskever, I. e Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Em *Advances in Neural Information Processing Systems (NIPS)*, p. 1097–1105.

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

Li, E., Zhou, Z. e Chen, X. (2018). Edge intelligence: On-demand deep learning model co-inference with device-edge synergy. Em *Proceedings of the 2018 Workshop on Mobile Edge Communications*, p. 31–36. ACM.

Parkhi, O. M., Vedaldi, A., Zisserman, A. e Jawahar, C. (2012). Cats and dogs. Em *IEEE Conference on Computer Vision and Pattern Recognition*, p. 3498–3505.

Satyanarayanan, M. (2017). The emergence of edge computing. *Computer*, 50(1):30–39.

Teerapittayanon, S., McDanel, B. e Kung, H.-T. (2016). Branchynet: Fast inference via early exiting from deep neural networks. Em *IEEE International Conference on Pattern Recognition (ICPR)*, p. 2464–2469.

Xu, M., Qian, F., Zhu, M., Huang, F., Pushp, S. e Liu, X. (2019). Deepwear: Adaptive local offloading for on-wearable deep learning. *IEEE Transactions on Mobile Computing*, 19(2):314–330.

Yao, S., Zhao, Y., Zhang, A., Hu, S., Shao, H., Zhang, C., Su, L. e Abdelzaher, T. (2018). Deep learning for the internet of things. *Computer*, 51(5):32–41.

Zhang, X., Wang, Y. e Shi, W. (2018). pcamp: Performance comparison of machine learning packages on the edges. Em *USENIX Workshop on Hot Topics on Edge Computing (HotEdge)*.