Otimizando Saídas Antecipadas em Redes Neurais Profundas: Como Lidar com Buffers?

Roberto G. Pacheco[4], Divya J. Bajpai[3], Mark Shifrin[2], Rodrigo S. Couto[1], Daniel Sadoc Menasché[1], Manjesh K. Hanawal[3] and Miguel Elias M. Campista [1] *

[1] Universidade Federal do Rio de Janeiro, Rio de Janeiro – RJ – Brasil

[2]Ben Gurion University, Negev, Israel

[3]Indian Institue of Technology, Bombay, India

[4]Universidade Federal Fluminense, Rio das Ostras – RJ – Brasil

Abstract. *Redes Neurais Profundas com Saídas Antecipadas (EE-DNNs) inserem ramos laterais que permitem a inferência local quando a confiança ultrapassa um limiar pré-definido, reduzindo a dependência da nuvem. No entanto, um limiar fixo não se adapta às variações contextuais do mundo real. Este trabalho investiga a adaptação dinâmica dos limiares utilizando algoritmos de multi-armed bandits (MABs). Além disso, um buffer de entradas finito é introduzido para equilibrar o compromisso entre acurácia e latência, considerando a confiança e o tamanho da fila. Os resultados experimentais demonstram que os limiares ajustados por MABs convergem rapidamente em diversos contextos, enquanto o buffer garante um equilíbrio eficiente entre acurácia e latência.*

Abstract. *Early-exit Deep Neural Networks (EE-DNNs) insert intermediate branches that enable local inference when confidence exceeds predefined thresholds, reducing reliance on cloud processing. However, fixed thresholds fail to adapt to real-world contextual variations. This work investigates dynamically adaptive threshold using multi-armed bandits (MABs) to address concept drift caused by contextual changes. Additionally, a finite input buffer is introduced to balance the accuracy-latency trade-off based on both confidence levels and queue size. Experimental results demonstrate that MAB-based thresholds converge rapidly, across diverse contexts, while the buffer ensures efficient balance the accuracy-latency trade-off.*

1. Introdução

Redes Neurais Profundas (Deep Neural Networks – DNNs) têm alcançado avanços notáveis em desempenho [Krizhevsky et al. 2012, He et al. 2016], especialmente em tarefas relacionadas à visão computacional. Contudo, o alto requisito de processamento dessas redes dificulta sua aplicação em dispositivos móveis para tarefas de inferência. Uma solução para contornar essas restrições envolve o uso de infraestrutura de computação em nuvem equipada com Unidades de Processamento Gráfico (GPUs) [Satyanarayanan 2017]. Em um modelo exclusivamente baseado na nuvem, os

*O trabalho teve apoio do CNPq (SEI:01300.010794/2023-1, 408255/2023-4 e 315106/2023-9), da CAPES (88881.694462/2022-01) e da FAPERJ (E-26/204.562/2024 e E-26/204.268/2024)

dispositivos móveis capturam dados e os transmitem à nuvem, onde o processamento completo do modelo DNN ocorre. Entretanto, essa abordagem implica em custos adicionais relacionados à comunicação entre os dispositivos móveis e o servidor na nuvem, o que reforça a necessidade de particionar os modelos DNN para uma execução conjunta entre dispositivos móveis e a nuvem [Kang et al. 2017, Hu et al. 2019]. Nesse contexto, surge a abordagem de inferência colaborativa, que se baseia no particionamento do modelo DNN. Esse particionamento é tratado como um problema de otimização, em que uma camada específica é selecionada para dividir o modelo [Pacheco et al. 2021a]. As camadas anteriores à camada de divisão são executadas localmente no dispositivo móvel, enquanto as demais são processadas na nuvem.

Este estudo aborda uma tarefa de classificação em que uma DNN com saídas antecipadas (*early-exit DNN* – EE-DNN) é empregada para identificar objetos em imagens. O trabalho explora um cenário de offloading adaptativo, combinando EE-DNNs e particionamento de modelo dentro de um *framework* de co-inferência entre dispositivos móveis e a nuvem. As DNNs com saídas antecipadas desempenham um papel crucial na implementação da co-inferência [Teerapittayanon et al. 2016,Laskaridis et al. 2020,Wang et al. 2019a], pois são projetadas com múltiplos ramos laterais que permitem a realização de inferências em camadas intermediárias.

Quando o dispositivo móvel captura uma imagem, ela é processada até alcançar um dos ramos laterais da DNN. Caso a confiança da classificação supere um limiar de confiança predefinido, a inferência é concluída localmente. Caso contrário, os dados são enviados à nuvem, que processa as camadas restantes do modelo. Essa abordagem permite que a inferência seja concluída condicionalmente, dependendo da complexidade de classificar a entrada, resultando em economia de recursos como rede, energia e capacidade computacional, além de reduzir o tempo de inferência para entradas menos complexas. Entretanto, optar por classificações antecipadas pode levar a uma redução no desempenho, gerando um trade-off entre latência e acurácia [Pacheco et al. 2021a].

BranchyNet [Teerapittayanon et al. 2016] e SPINN [Laskaridis et al. 2020] empregam um limiar fixo para decidir se a inferência deve ser encerrada. Essas abordagens pressupõem que a distribuição de amostras entre classes é constante, o que pode ser inadequado em cenários reais. Por exemplo, um dispositivo móvel em um porto tende a processar mais imagens de navios, enquanto, em uma rodovia, o mesmo dispositivo precisaria classificar predominantemente carros e caminhões. O ambiente em que as imagens são coletadas, referido como *contexto*, exerce influência direta nos resultados da inferência.

Neste cenário, este trabalho apresenta duas principais contribuições:

Multi-armed bandits para lidar com *concept drift*: Este trabalho demonstra que algoritmos de multi-armed bandits (MABs) são eficazes na adaptação dinâmica dos limiares de confiança para enfrentar o concept drift, ou seja, mudanças nas propriedades estatísticas das imagens ao longo do tempo. Os resultados indicam que os limiares ajustados por MABs convergem após algumas milhares de amostras, maximizando o desempenho em diferentes contextos e cenários de sobrecarga. Os MABs levam em consideração: $i)$ as confianças obtidas nos ramos laterais; $ii)$ os ganhos de confiança estimados ao processar todas as camadas do modelo; e $iii)$ os custos associados à sobrecarga.

Buffer de entradas para gerenciar o trade-off entre precisão e atraso: Um

buffer finito de entradas é introduzido para auxiliar o controlador na tomada de decisões sobre saídas antecipadas, com base tanto na confiança da classificação quanto no tamanho da fila de espera. O *backlog* representa a quantidade de tarefas ou dados acumulados aguardando processamento pelo modelo de EE-DNN. Quando o *backlog* é pequeno, o controlador pode priorizar a acurácia das inferências processando todo o modelo de EE-DNN, garantindo resultados mais precisos. No entanto, se o *backlog* é alto, ou seja, o *buffer* estiver próximo de sua capacidade máxima, o controlador precisa favorecer a velocidade de processamento para evitar a perda de novas entradas, garantindo assim a eficiência do sistema mesmo sob alta demanda.

Este estudo amplia os resultados apresentados em [Pacheco et al. 2024], introduzindo duas contribuições principais: 1) a implementação de um buffer de entrada para aprimorar as decisões relacionadas às saídas antecipadas e 2) a avaliação do *concept drift*, em que mudanças no contexto refletem alterações nas categorias de imagens. Diferentemente dos trabalhos anteriores [Pacheco et al. 2024], que focaram a influência de diferentes níveis de desfoque (*blur*) nas imagens, este trabalho considera explicitamente variações contextuais associadas às classes de imagens existentes no conjunto de dados.

A organização do artigo é a seguinte: a Seção 2 apresenta uma revisão dos trabalhos relacionados. Na Seção 3, são explorados os conceitos de DNNs com saídas antecipadas e a aplicação de multi-armed bandits. A Seção 4 aborda a formulação do problema considerando a introdução de um buffer de entradas. A Seção 5 discute o impacto do contexto sobre as decisões de saída antecipada, demonstra a convergência dos MABs para a escolha de limiares adequados e analisa o impacto do *buffer*. Por fim, a Seção 6 apresenta as conclusões e sugere direções para pesquisas futuras.

2. Trabalhos Relacionados

BranchyNet [Teerapittayanon et al. 2016] é uma DNN com saídas antecipadas que decide se uma amostra pode ser classificada antecipadamente, com base na entropia da classificação. A decisão é tomada quando o valor da entropia é inferior a um limiar fixo predefinido, permitindo que a amostra seja classificada em um ramo lateral. Arquiteturas similares, como SPINN [Laskaridis et al. 2020] e SEE [Wang et al. 2019b], utilizam a confiança da classificação, representada pela probabilidade da classe mais provável, para decidir sobre saídas antecipadas. Este trabalho também considera uma arquitetura em que a saída antecipada ocorre quando a confiança ultrapassa um limiar específico.

Além da BranchyNet e da SPINN, outras abordagens exploram DNNs com saídas antecipadas para reduzir o tempo de inferência. FlexDNN [Fang et al. 2020] e Edgent [Li et al. 2019] utilizam essas DNNs para selecionar a profundidade ideal do modelo, enquanto Dynexit [Wang et al. 2019a] implementa DNNs com saídas antecipadas em hardware FPGA, visando otimizar a latência e o consumo energético. Paul et al. [Kim and Park 2020] também destacam as vantagens de implementar DNNs com saídas antecipadas em placas FPGA, alcançando ganhos significativos em tempo de inferência e eficiência energética.

Casale e Roveri [Casale and Roveri 2023] introduzem o conceito de buffer de entradas no contexto de DNNs com saídas antecipadas (EE-DNNs), no qual o *backlog* do buffer influencia diretamente as decisões de saída antecipada. No entanto, este trabalho se diferencia ao abordar o impacto das mudanças no contexto, o que está fora do escopo

de [Casale and Roveri 2023].

Os trabalhos existentes geralmente assumem limiares fixos [Fang et al. 2020, Li et al. 2019, Kim and Park 2020, Pacheco et al. 2021a, Pacheco et al. 2021b] ou pré-definidos durante a inferência. Em contraste, este trabalho propõe uma abordagem adaptativa que ajusta os limiares dinamicamente por meio de aprendizado por reforço, respondendo às mudanças no contexto.

A maioria das soluções atuais define saídas antecipadas comparando valores de confiança ou entropia a limiares fixos. Exceções são LEE [Ju et al. 2021b] e DEE [Ju et al. 2021a], que utilizam MABs para aprender políticas de saída. Apesar de motivado por objetivos semelhantes, este trabalho se distingue em aspectos fundamentais, como o uso de MABs para ajustar os limiares que determinam saídas ótimas, enquanto trabalhos como LEE e DEE definem explicitamente a saída ótima como variável de controle.

3. DNN com Saídas Antecipadas Através de Multi-Armed Bandits

Este trabalho utiliza a DNN MobileNetV2 [Krizhevsky et al. 2012] com um ramo lateral como modelo para DNNs com saídas antecipadas, considerando uma inferência colaborativa entre borda e nuvem, como ilustrado na Figura 1(a). Após receber uma entrada $\boldsymbol{x}$, o modelo é processado no dispositivo de borda, camada por camada, até atingir o ramo lateral, no qual um vetor de saída intermediário $\boldsymbol{z}_I$ é calculado. Em seguida, o modelo aplica a função *softmax* para gerar o vetor de probabilidade $\boldsymbol{p}_I \triangleq \text{softmax}(\boldsymbol{z}_I)$, em que $\text{softmax}(\boldsymbol{z}_I) \propto \exp(\boldsymbol{z}_I)$. Cada componente de $\boldsymbol{p}_I$ representa a probabilidade de $\boldsymbol{x}$ pertencer a uma classe específica. A partir de $\boldsymbol{p}_I$, calcula-se a confiança intermediária de classificação da entrada $\boldsymbol{x}$ como $C_I = \max \boldsymbol{p}_I$. Se $C_I \geq \alpha$, onde α é o limiar de confiança, o ramo lateral classifica $\boldsymbol{x}$ como $\hat{y} = \arg\max(\boldsymbol{p}_I)$, encerrando o processo de inferência. Caso contrário, se $C_I < \alpha$, a borda transfere dados à nuvem que processa as camadas subsequentes até a camada final, gerando a confiança C_L. Este processamento adicional resulta em uma sobrecarga o, que pode ser significativa dependendo do cenário. Por exemplo, em aplicações de offloading adaptativo com partição de DNN, a sobrecarga inclui o tempo de comunicação para transferir os dados do dispositivo em borda à nuvem e o tempo de processamento das camadas restantes na nuvem. Em aplicações puramente móveis, onde todo o modelo é processado no dispositivo, a sobrecarga está associada ao tempo de execução das camadas subsequentes no dispositivo e, possivelmente, ao consumo de energia. Embora EE-DNNs possam ter múltiplos ramos laterais, este trabalho considera apenas um ramo lateral, simplificando a análise.

Tradicionalmente, as EE-DNNs empregam um limiar de confiança fixo α. Contudo, aplicações reais podem requerer o emprego de limiar adaptativo a depender do contexto, como explicado anteriormente. Este estudo aborda a adaptação de EE-DNNs a diferentes contextos utilizando algoritmos baseados em *multi-armed bandits* (MABs). A principal contribuição deste trabalho é a definição de um limiar condicional de confiança, que determina se a saída antecipada será utilizada. Este limiar é adaptativo, ajustando-se de acordo com a acurácia observada no ramo lateral e as condições do contexto. A Seção 4 detalha o algoritmo proposto para a escolha adaptativa do limiar, utilizando aprendizado por reforço via MABs.

Modelo. Para cada entrada (imagem), as confianças de classificação nas camadas intermediária e final, representadas por C_I e C_L, respectivamente, permanecem desconhe-

Algorithm 1: Algoritmo de seleção de limiar de confiança (MAB)

Entrada: $\tilde{c} > 0$, o (sobrecarga), K (número de limiares)
Inicialização Tente cada limiar; atribua $Q_1(\alpha_1) = r_1(\alpha_1), \ldots, Q_K(\alpha_K) = r_K(\alpha_K)$
for $t = K+1, K+2, \ldots,$ **do**
    Receba uma amostra $\alpha_t \leftarrow \text{argmax}_{\alpha \in \mathcal{A}} \left(Q_{t-1}(\alpha) + \tilde{c}\sqrt{\frac{\ln(t)}{N_{t-1}(\alpha)}} \right)$;
    Obtenha a confiança C_I na camada intermediária
    if $C_I \geq \alpha_t$ **then**
        Realiza classificação antecipada e define $r_t(\alpha_t) \leftarrow 0$
    else
        Use a última camada e observe C_I and C_L
        $r_t(\alpha_t) \leftarrow \max(C_L - C_I, 0) - o$
    end
    $N_t(\alpha_t) = N_{t-1}(\alpha_t) + 1$;
    $Q_t(\alpha_t) = \sum_{u=1}^{t} r_u(\alpha_u) \mathbb{1}\{\alpha_u = \alpha_t\} / N_t(\alpha_t)$;
end

cidas até que a entrada seja processada por essas camadas da DNN. Dado cada exemplo, o algoritmo decide uma ação $\alpha \in \mathcal{A}$, correspondente a um limiar de confiança. Posteriormente, a confiança intermediária C_I é comparada com o limiar α. A combinação da ação α e do valor de C_I determinará se a inferência será concluída antecipadamente na camada intermediária ou se o processamento seguirá para as camadas subsequentes.

Considere ΔC como o incremento de confiança obtido ao avançar da camada intermediária para a camada final, definido por $\Delta C = \max\{C_L - C_I, 0\}$. Denotando por C a confiança de classificação associada a uma observação escolhida aleatoriamente, a confiança média esperada sobre todas as observações é expressa como: $\mathbb{E}[C] = \mathbb{E}[\Delta C + C_I \mid C_I < \alpha] \cdot P[C_I < \alpha] + \mathbb{E}[C_I \mid C_I \geq \alpha] \cdot P[C_I \geq \alpha]$. Neste caso, o primeiro termo considera as observações que continuam até a camada final ($C_I < \alpha$), enquanto o segundo abrange as que são classificadas antecipadamente ($C_I \geq \alpha$). Para qualquer limiar $\alpha \in \mathcal{A}$, seja $r(\alpha)$ a recompensa instantânea associada a esse limiar. A recompensa instantânea é definida como: $r(\alpha) = 0$ se $C_I \geq \alpha$ (classificação antecipada), e $r(\alpha) = \Delta C - o$, caso contrário. A sobrecarga o está associada ao processamento da amostra até a camada final. Neste trabalho, a sobrecarga o é normalizada, ou seja, é escalada por um fator que torna as unidades de energia ou tempo comparáveis com ΔC.

Então, a recompensa média por escolher o limiar de confiança α é dada por: $\mathbb{E}[r(\alpha)] = \mathbb{E}[\Delta C - o | C_I < \alpha] \cdot P[C_I < \alpha]$. O objetivo é encontrar um limiar de confiança que atinja a recompensa ótima dada por $\alpha^* = \arg\max_{\alpha \in \mathcal{A}} \mathbb{E}[r(\alpha)]$. Este trabalho define o desempenho em escolher uma política Π_{MAB} ao longo de $n \in \mathbb{N}$ rodadas em termos de arrependimento esperado (*expected regret*) como: $R(\Pi_{\text{MAB}}, n) = n \cdot \mathbb{E}[r(\alpha^*)] - \sum_{t=1}^{n} \mathbb{E}[r(\alpha_t) | \Pi_{\text{MAB}}]$, onde α_t denota o limiar selecionado pelo aprendiz na rodada t, com base nas entradas anteriores. O arrependimento de Π_{MAB} é dado pela diferença entre a recompensa coletada ao usar um limiar ótimo e a recompensa coletada por Π_{MAB}.

Algoritmo. O pseudo-código do algoritmo é apresentado no Algoritmo 1, baseando-se no Upper Confidence Bound (UCB) [Auer et al. 2002]. Os parâmetros de entrada do algoritmo são o fator de compromisso entre exploração e exploração, $\tilde{c}$, e

o número de limiares, K. Inicialmente, cada limiar é aplicado uma vez para os primeiros K inputs, gerando uma amostra para cada braço. Nas rodadas subsequentes, o limiar com o índice UCB mais alto é selecionado. Se a confiança da classificação na camada intermediária exceder o limiar, uma saída antecipada ocorre (linhas 6-7). Caso contrário, o algoritmo processa todas as camadas subsequentes. Neste caso, a classificação da camada com a maior confiança é considerada como a saída final (linhas 8-10). Em seguida, as estatísticas $N_t(\alpha_t)$ e $Q_t(\alpha_t)$ são atualizadas (linhas 13-14). Aqui, $N_t(\alpha_t)$ representa o número de vezes que a ação α_t foi escolhida na t-ésima rodada, e $Q_t(\alpha_t)$ é a recompensa média associada à ação α_t na mesma rodada.

4. Buffer de Entrada

A seguir, considera-se um buffer finito de observações com capacidade B. As entradas chegam ao buffer para serem classificadas por uma EE-DNN, e o controlador pode determinar, com base no *backlog* do buffer e na confiança intermediária, se deve realizar a saída antecipada. Vale lembrar que o *backlog* do buffer corresponde à quantidade de entradas acumuladas no *buffer* aguardando processamento, o qual é denotado por n. Enquanto na seção anterior as decisões em relação à classificação antecipada eram feitas exclusivamente com base em C_I, agora essas decisões podem depender de n e C_I.

Como nas seções anteriores, o Tomador de Decisão (TD) deve ponderar duas opções: aproveitar a saída antecipada, que oferece menor acurácia, mas um atraso mínimo, ou processar todo modelo, que oferece maior acurácia, mas resulta em um maior atraso. O objetivo principal é maximizar a eficiência do sistema, medida pela vazão (*throughput*), que corresponde à taxa de observações classificadas corretamente por unidade de tempo, equilibrando esses trade-offs. Por exemplo, considere um cenário em que o TD deve decidir entre classificar uma entrada rapidamente para obter um processamento mais rápido ou continuar pelas camadas mais profundas para melhorar a precisão da classificação, às custas de um maior atraso. Essa decisão depende de vários fatores, como a carga atual do sistema, a confiança na classificação e o *backlog* de entradas aguardando processamento.

Compromisso (*tradeoff*) fundamental. Quando o *buffer* está cheio, atrasar a classificação pode resultar em perdas devido ao descarte de novas entradas. Tais perdas causam uma diminuição na vazão. Por outro lado, quando o *buffer* está subutilizado, depender excessivamente das saídas antecipadas pode causar perda desnecessária de acurácia, afetando mais uma vez a vazão, medida em observações corretamente classificadas por unidade de tempo. Assim, para maximizar a vazão é necessário achar um balanço ótimo no uso de saídas antecipadas, conforme detalhado nas seções a seguir.

Modelo: CTMDP para DNNs com Saídas Antecipadas. Este artigo utiliza um Processo de Decisão de Markov em Tempo Contínuo (*Continuous-Time Markov Decision Process* – CTMDP) para determinar a política ótima que equilibra tempo de inferência e acurácia. O modelo é inspirado em [Shifrin et al. 2020] e no exemplo de controle de admissão no Capítulo 6 de [Puterman 2014].

Este trabalho assume que as entradas chegam de acordo com um processo de Poisson com taxa λ, e que o tempo para processar cada entrada é distribuído exponencialmente com média $1/\mu_u$, onde u é a ação escolhida: $u = I$ caso seja realizada uma saída antecipada, e $u = L$, caso contrário. Seja θ a sobrecarga média, em unidades de tempo, para

se transmitir cada entrada à nuvem e processar as camadas finais do modelo de EE-DNN. Então, $1/\mu_L = 1/\mu_I + \theta$, onde $1/\mu_I$ é o tempo médio para processar uma observação na borda, usando a saída antecipada.

Os principais componentes do CTMDP são: 1) **espaço de estados:** cada estado representa o *backlog* do sistema e, possivelmente, a decisão de classificação atual para a entrada na cabeça da fila (*Head of Line* - HOL), ou seja, se será realizada uma saída antecipada ou não; 2) **ações:** o TD escolhe, em última instância, se deve realizar uma saída antecipada ou continuar o processamento por camadas mais profundas, dependendo do tamanho do *backlog*; 3) **objetivo:** maximizar a *vazão*, equilibrando o compromisso entre melhoria na acurácia e o tempo de inferência.

Nos cenários em que se considera um *buffer*, assume-se que o TD determina diretamente se realizará a saída antecipada ou não, em função do *backlog*. Para estender os resultados ao caso em que o TD escolhe o limiar α, que impacta indiretamente a eventual saída antecipada, é preciso adicionar uma função adicional que mapeie α em uma probabilidade de saída antecipada, levando em conta a distribuição de C_I. Essa extensão, unificando os modelos MAB e CTMDP, será tratada em trabalhos futuros.

Equação de Bellman. Em um processo de decisão Markoviano, a função de valor é definida para cada estado e é computada iterativamente utilizando as equações de Bellman [Puterman 2014]. A política ótima é definida a partir da função de valor. Devido a restrições de espaço, esta versão do trabalho não apresenta as equações de Bellman e o algoritmo de iteração de valor (*value iteration*) em sua generalidade. Ao invés disso, é aproveitado o resultado derivado em [Shifrin et al. 2020] para tratar de uma classe especial de políticas, chamadas de políticas baseadas em limiar (*threshold policies*). Em [Shifrin et al. 2020] os autores provam que a política ótima para tomada de decisão no tipo de problema considerado neste artigo é do tipo *limiar de backlog* (*backlog threshold*). Segundo tal política, se o *backlog* contiver mais que um determinado número de requisições pendentes, deve-se adotar a ação mais rápida (neste caso, *saída antecipada*). Caso contrário, pode-se adotar a ação mais demorada (neste caso, *processar até a última camada*). Os leitores interessados nas equações de Bellman e no algoritmo de iteração por valor podem consultá-los no relatório técnico.

Seja τ o limiar de *backlog*. Se, após um serviço, o *backlog* n for tal que $n \geq \tau$, adota-se a saída antecipada para o próximo serviço. Caso contrário, se $n < \tau$, o próximo serviço ocorre sem saída antecipada. A seguir, descreve-se brevemente a cadeia de Markov que caracteriza o sistema para um determinado τ, e nos estudos numéricos varia-se o limiar τ, entre 0 e B, buscando-se pelo ótimo que maximiza a vazão.

Fluxos envolvidos no modelo Markoviano. A Figura 1(a) ilustra o modelo Markoviano considerado. Um fluxo de observações com taxa λ alimenta o buffer local, que possui capacidade B. As entradas que excedem a capacidade do buffer são descartadas, gerando um fluxo de descarte com taxa δ. As observações não descartadas são processadas por um dispositivo de borda, que decide localmente, com base na ocupação da fila, se as classificações devem ser realizadas em camadas intermediárias ou enviadas à nuvem para processamento adicional.

Cadeia de Markov. A cadeia de Markov modela o comportamento de um sistema com saídas antecipadas e um buffer finito, para um dado limiar τ fixo. As transições

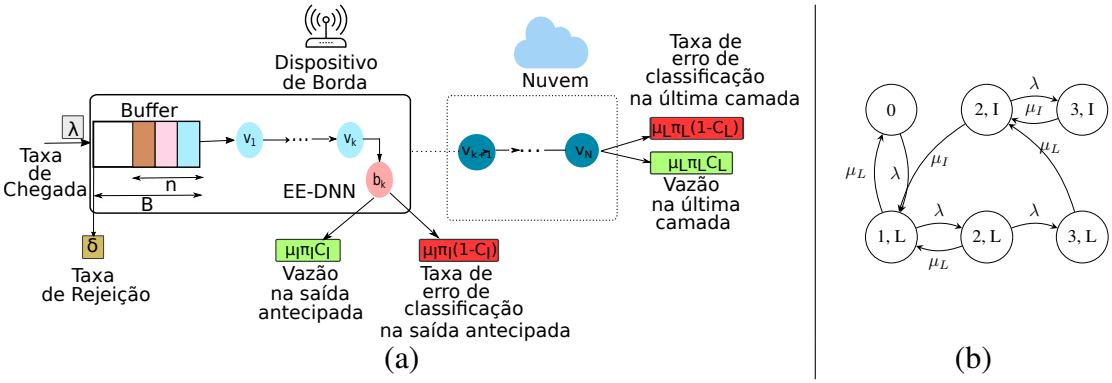


Figura 1. (a) Modelo do sistema; (b) Cadeia de Markov ilustrativa: $\tau = 2, B = 3$**.**

de estado são determinadas pelas taxas de chegada (λ) e pelas taxas de atendimento (μ_I e μ_L), que correspondem ao processamento pela saída antecipada e pela rede completa, respectivamente. Cada estado da cadeia é representado por um par (n, u), onde n indica o número de itens no buffer e u especifica a classe de processamento, que pode ser saída antecipada ($u = I$) ou modelo de EE-DNN completo ($u = L$). O estado (0) é usado para indicar que o buffer está vazio. O objetivo da modelagem é capturar o impacto de diferentes políticas e configurações de buffer sobre o desempenho do sistema. A Figura 1(b) ilustra a cadeia de Markov considerada, para $\tau = 2$ e $B = 3$.

A probabilidade em estado estacionário do estado (n, u) é dada por $\pi_{n,u}$, para $1 \leq n \leq B$, e do estado 0 é dada por π_Z. Seja π_I a fração de tempo em que o sistema está processando uma entrada usando a saída antecipada, enquanto π_L corresponde a fração de tempo para processar até a última camada. Note que $\pi_I = \sum_{n=1}^{B} \pi_{n,I}$ e $\pi_L = \sum_{n=1}^{B} \pi_{n,L}$, de modo que $\pi_I + \pi_L + \pi_Z = 1$. A matriz de transição da cadeia de Markov contínua é construída considerando três tipos de eventos principais. O primeiro é a chegada de itens ao buffer, que ocorre com taxa λ. Esse evento resulta na transição para estados com maior ocupação do buffer, até o limite máximo B ser atingido. O segundo é o processamento pela saída antecipada, com taxa μ_I, no qual itens deixam o buffer e podem ser classificados corretamente ou incorretamente. O terceiro evento é o processamento pelo modelo completo, com taxa μ_L, cujo comportamento é análogo ao da saída antecipada.

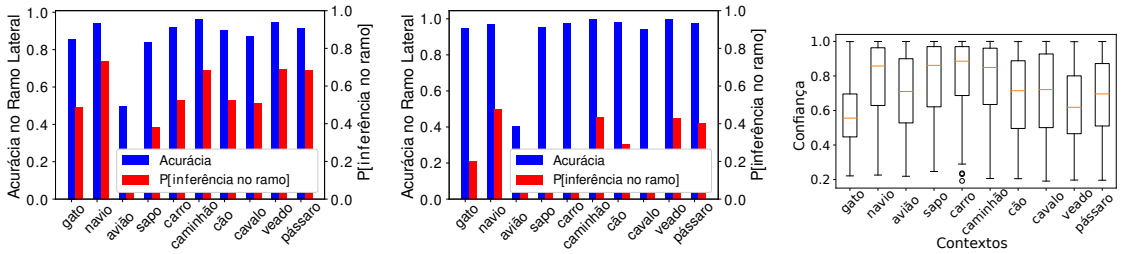
A partir do equilíbrio estacionário da cadeia de Markov, podem ser derivadas três métricas principais. A vazão, ou *throughput* (T), representa a taxa de itens processados corretamente, considerando tanto a saída antecipada quanto o modelo completo. A taxa de classificações incorretas (M) quantifica a taxa de itens processados com erro. A taxa de descarte (δ) mede a taxa de itens descartados devido ao congestionamento do buffer. Essas métricas são inter-relacionadas pela conservação de fluxo no sistema, de forma que a soma das três é igual à taxa de chegada: $T + M + \delta = \lambda$, conforme Figura 1(a), onde

$$T = \mu_I \pi_I C_I + \mu_L \pi_L C_L, \quad M = \mu_I \pi_I (1 - C_I) + \mu_L \pi_L (1 - C_L), \quad \delta = \lambda(\pi_{B,L} + \pi_{B,I}). \tag{1}$$

A seções a seguir avaliam as métricas acima experimentalmente ao considerar, conjuntamente, o impacto do contexto e do buffer.

5. Avaliação

As avaliações empregam o modelo MobileNetV2 com um ramo lateral inserido conforme a metodologia descrita em [Laskaridis et al. 2020]. O modelo é treinado se-



(a) Desempenho com limiar α=0.7 (b) Desempenho com limiar α=0.9 (c) Confiança intermediária C_I

Figura 2. Acurácia em ramo lateral e a probabilidade de classificação antecipada: (a) $\alpha = 0.7$ e (b) $\alpha = 0.9$; (c) confiança na camada intermediária, C_I.

guindo o procedimento apresentado em [Teerapittayanon et al. 2016] usando o conjunto de dados CIFAR-10 [Krizhevsky et al. 2009].

O CIFAR-10 contém imagens distribuídas em dez classes. O modelo EE-DNN pré-treinado tem como objetivo classificar cada amostra em sua respectiva classe. Conforme descrito na Seção 5.1, em cada experimento, o modelo é alimentado com amostras pertencentes a uma única classe, denominada aqui como contexto. Neste caso em particular, temos falsos negativos quando a rede neural infere que a classe é diferente daquela correspondente ao contexto, e verdadeiros positivos caso contrário. Na Seção 5.2, o conceito de contexto é relaxado. Todo código desenvolvido está disponível em um repositório aberto[1], incluindo a descrição detalhada da cadeia de Markov e o código para reproduzir os resultados deste artigo.

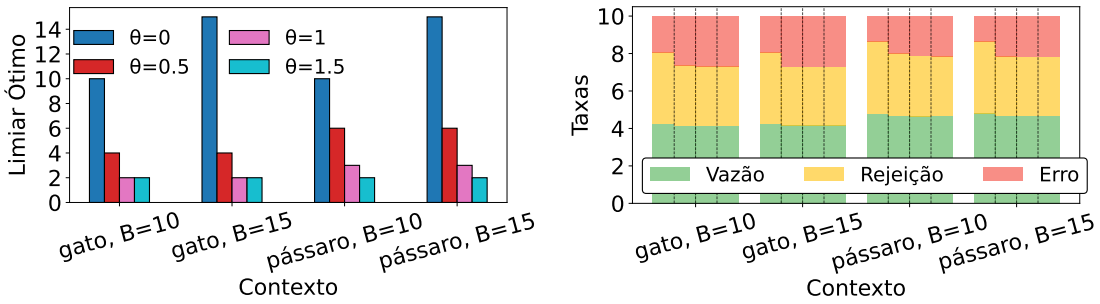
5.1. Como Lidar com o Contexto e o Buffer?

Esta seção visa demonstrar o impacto do contexto no desempenho de DNNs com saídas antecipadas e buffers. Para avaliar o desempenho, esta seção considera duas métricas: probabilidade de inferência no ramo lateral e acurácia no ramo lateral. A probabilidade de inferência no ramo lateral é definida como a proporção de entradas classificadas neste ramo em relação ao número total de entradas, enquanto a acurácia no ramo lateral é calculada como a razão entre as classificações corretas realizadas nesse ramo e o número total de entradas classificadas nele.

As Figuras 2(a) e 2(b) mostram, respectivamente, o desempenho do modelo considerando dois limiares de confiança diferentes $\alpha = 0.7$ e $\alpha = 0.9$. Especificamente, as barras azuis mostram a acurácia da saída antecipada, e as barras vermelhas mostram a probabilidade de classificação no ramo lateral. Nota-se que as diferentes classes apresentam variações significativas na acurácia da saída antecipada e na probabilidade de classificação no ramo lateral. Por exemplo, para um limiar $\alpha = 0.7$, a classe de pássaros corresponde a uma probabilidade de saída antecipada de aproximadamente 0.7, com a acurácia correspondente de 0.9. Por outro lado, a probabilidade e a acurácia da saída antecipada para a classe de aviões são 0.5 e 0.2, respectivamente. De forma clara, a distribuição das observações entre as classes impacta o desempenho da saída antecipada.

Desafio: alta variância intra-classe. A Figura 2(c) apresenta um *boxplot* com as confianças intermediárias C_I para cada classe do CIFAR-10. Esta figura mostra a alta

[1] `https://github.com/pachecobeto95/QAdaEE`.



(a) Limiar de *backlog* ótimo.

(b) Decomposição da taxa de chegada.

Figura 3. Impacto do contexto no limiar de *backlog* ótimo e demais métricas.

variância nas confianças das entradas, mesmo para amostras da mesma classe. Essa alta variância intra-classe indica a necessidade de limiares adaptativos. Embora os MABs sejam uma solução natural para implementar limiares adaptativos, a alta variância intra-classe das confianças desafia a convergência de algoritmos clássicos de MAB, como o algoritmo UCB [Auer et al. 2002], o que requer uma divisão adicional das entradas com base nas suas respectivas confianças de classificação, conforme detalhado a seguir.

Impacto do contexto no buffer. A Figura 3(a) apresenta o limiar de *backlog* ótimo, considerando diferentes tamanhos de *buffer*, níveis de sobrecarga e contextos, cada contexto representando uma classe do conjunto CIFAR-10. Os resultados evidenciam que o limiar de *backlog* ótimo varia conforme o contexto, reforçando a importância de limiares de *backlog* adaptativos em modelos EE-DNNs. Além disso, observa-se uma relação inversa entre o nível de sobrecarga e o limiar de *backlog*: à medida que a sobrecarga associada à transmissão e processamento na nuvem aumenta, os limiares ótimos tendem a diminuir. Esse comportamento sugere que, em cenários com maior sobrecarga, é mais vantajoso concluir a inferência nos ramos laterais, adotando saídas antecipadas, e que o valor ótimo depende do contexto. A Figura 3(b), obtida usando os limiares de *backlog* ótimos apresentados na Figura 3(a), indica como que a decomposição apresentada na Eq. (1) se traduz na prática. Em particular, vemos que a vazão aumenta sutilmente quando passamos da classe ‘gato’ para a classe ‘pássaro’. Tal aumento da vazão (verde) se deve primordialmente a uma diminuição na taxa de erro de classificação (vermelho), notando-se que taxa de rejeição de observações se mantém praticamente estável ao longo das configurações consideradas (amarelo). Tal estabilidade se deve, em parte, ao ajuste no limiar, que nesta seção é ilustrado sempre em seu valor ótimo. Na Seção 5.3, analisa-se a classe ‘gato’, ilustrando como os componentes da Eq. (1) variam em cenários subótimos.

5.2. Avaliação do Algoritmo de Aprendizado

Classes e contextos. A seguir, o conceito de *contexto* é refinado. Para isso, as entradas de interesse são divididas em bins. Assim, o contexto se refere à distribuição das entradas entre os bins. Cada bin pode corresponder a uma classe alvo (por exemplo, avião, navio ou cachorro, como na Figura 2), ou a um intervalo de confiança (por exemplo, $C_I \in [0, 0.1]$, como na Figura 4). Neste último caso, no qual o contexto corresponde à distribuição das entradas conforme a confiança intermediária C_I, agrupam-se as entradas em m bins de intervalo de tamanho igual $B_1, \ldots, B_m$ com base na confiança C_I para capturar a complexidade da entrada. Assim, cada bin B_i contém imagens cuja confiança

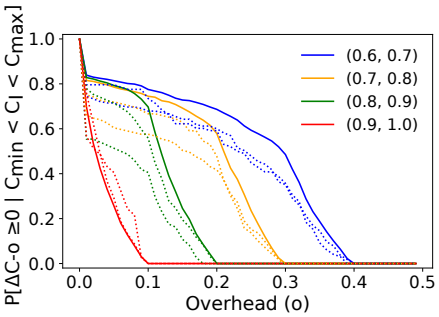


Figura 4. Probabilidade de valer a pena processar até a última camada.

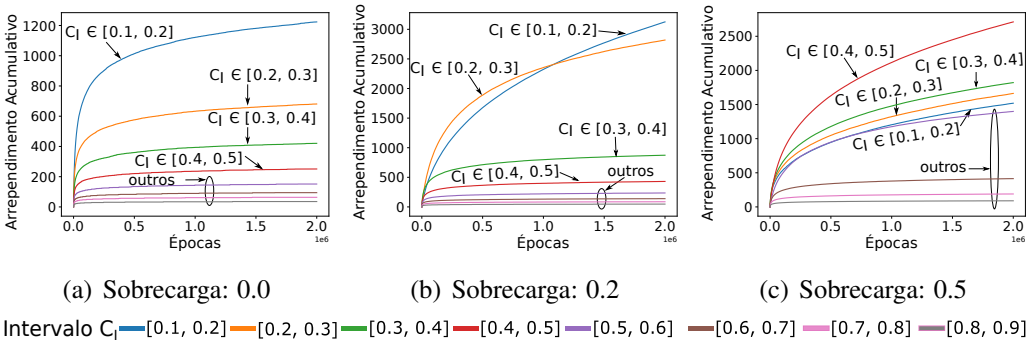
C_I satisfaz $C_I \in \left(\frac{i-1}{m}, \frac{i}{m}\right]$, por exemplo, se $m = 10$, o bin B_1 contém imagens com $C_I \in (0, 0.1]$. Como antes, o contexto muda quando a distribuição das observações entre os bins é modificada, observando que o contexto impacta a avaliação do algoritmo, mas não afeta explicitamente sua operação.

Qual é o impacto da sobrecarga? A Figura 4 mostra a probabilidade de valer a pena processar até a última camada, em função da sobrecarga, sob diferentes intervalos de C_I. Cada linha sólida representa um contexto particular composto por entradas cujos C_I estão em um bin de confiança específico, usando as observações de todas as classes. Primeiramente, essa figura quantifica que uma sobrecarga menor aumenta a probabilidade de processamento até a última camada. De fato, observe que essas curvas são funções de distribuição acumulada complementar (CCDFs). Assim, dado o intervalo de C_I, a área sob essas curvas corresponde ao ganho médio condicional de confiança. A área sob as curvas vermelha, verde, laranja e azul sólida é 0.07, 0.22, 0.35 e 0.48, respectivamente, correspondendo a valores decrescentes de C_I.

A saída antecipada é ótima sob um critério de ganho médio quando a área é menor que a sobrecarga. Quanto maior o valor de C_I, mais provável é que a saída antecipada seja ótima. A Figura 4 também nos permite avaliar o impacto das decisões além das médias, levando em conta a distribuição dos ganhos. Por exemplo, quando a sobrecarga é 0.15, a saída antecipada é ótima para 50% e 80% das entradas se $C_I \in (0.8, 0.9)$ e $C_I \in (0.6, 0.7)$, respectivamente. As linhas pontilhadas filtram ainda mais as entradas com base em suas classes-alvo, ou seja, representando uma mudança de contexto baseada em classes. Para cada um dos quatro intervalos de C_I considerados, adicionamos duas linhas pontilhadas que correspondem a amostras das classes-alvo de cães e gatos. A figura mostra que as áreas sob as curvas, e portanto os limiares ótimos, mudam de acordo com o contexto. De fato, esse comportamento exige uma avaliação adaptativa de limiares, na qual os benefícios da saída antecipada são continuamente avaliados em função das estatísticas coletadas de novas entradas.

Arrependimento logarítmico é viável? Os resultados demonstram experimentalmente a convergência para selecionar o limiar ótimo α^* que se adapta ao contexto. Para isso, executa-se o algoritmo considerado utilizando imagens de teste do conjunto de dados CIFAR-10. As imagens são agrupadas em bins de confiança, conforme descrito na Seção 5.1. A cada época, uma imagem do bin é sorteada com reposição. Em seguida, realiza-se uma inferência adaptativa de saída antecipada, escolhendo o limiar α utilizando o algoritmo baseado em MAB considerado.

A Figura 5 mostra o arrependimento acumulado em função das épocas para um



(a) Sobrecarga: 0.0 (b) Sobrecarga: 0.2 (c) Sobrecarga: 0.5

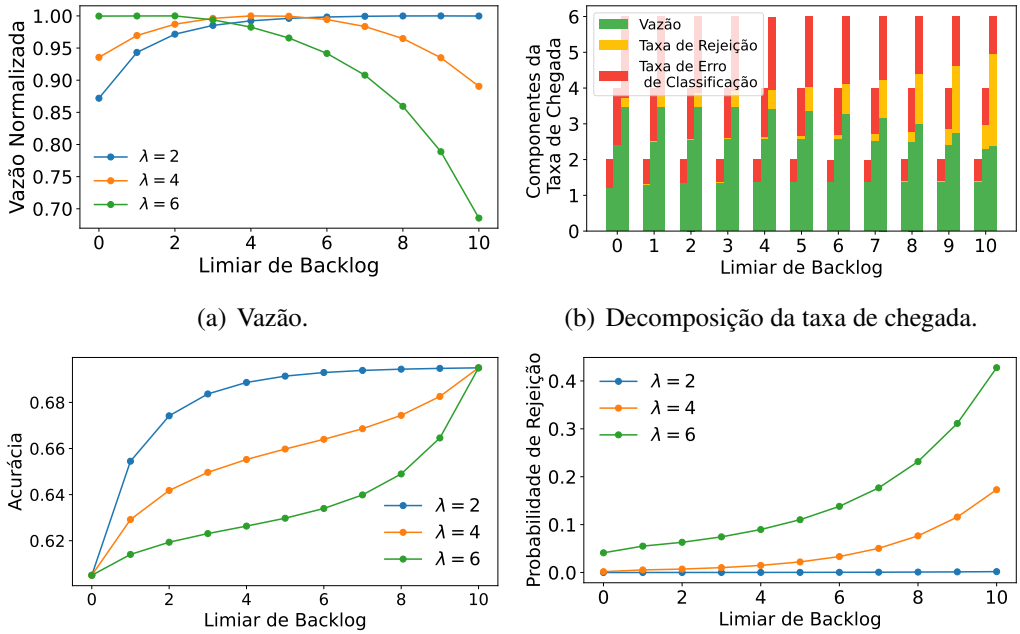
Figura 5. Arrependimento acumulado para diferentes valores de sobrecargas.

valor específico de sobrecarga e vários contextos, representados pelos bins. Cada linha corresponde a um contexto específico, enquanto cada gráfico mostra os resultados para um valor de sobrecarga específico. O arrependimento acumulado R_t é calculado como a soma do arrependimento instantâneo $r(\alpha)$ obtido até a t-ésima época. A Figura 5 confirma experimentalmente que o arrependimento logarítmico é alcançável, ou seja, a convergência normalmente ocorre após a coleta de algumas milhares de observações para diferentes contextos e valores de sobrecarga. Na Figura 5(a), quando a sobrecarga é zero, o limiar ótimo é o máximo permitido, e, portanto, nunca é realizada uma saída antecipada. Em contraste, a Figura 5(c) mostra que o limiar ótimo é o mínimo permitido quando a sobrecarga é 0,5, realizando sempre uma saída antecipada. Para outros valores de sobrecarga, o limiar ótimo está entre os dois extremos. A Figura 5 mostra que, à medida que a sobrecarga aumenta de 0 para 0,5, a dificuldade de encontrar o limiar ótimo primeiro aumenta e depois diminui. De fato, para todos os contextos considerados, o arrependimento acumulado cresce logaritmicamente após 100.000 observações coletadas quando a sobrecarga é zero, permanecendo abaixo de 1.200 na época 2×10^6. Em contraste, quando a sobrecarga é igual a 0,2, encontramos que em quatro dos oito contextos considerados, o arrependimento acumulado atinge 3.000 sob o mesmo horizonte. Em seguida, quando a sobrecarga é igual a 0,5, o arrependimento acumulado não ultrapassa 1.600 sob o horizonte considerado, exceto para um dos contextos considerados.

5.3. Qual o Impacto do Limiar do *Backlog*?

A seguir, considera-se um buffer de tamanho 10, ou seja, capaz de armazenar até 10 entradas. O modelo de buffer proposto é parametrizado usando dados da classe 'gato' da seguinte forma: $\mu_I = 6.9$, $C_I = 0.605$, $C_L = 0.695$ e $\mu_L = \mu_I/2$, para capturar efeitos do atraso em rede. A Figura 6 ilustra os resultados obtidos, variando λ no conjunto $\{2, 4, 6\}$. Seja a vazão normalizada igual a vazão obtida para um determinado limiar de *backlog* dividida pela vazão máxima alcançada no cenário em questão. A Figura 6(a) ilustra o fato de que para taxas de chegada muito pequenas ou muito grandes, vale a pena sempre usar a rede completa ou sempre fazer a saída antecipada, respectivamente. Por outro lado, para uma taxa de chegada intermediária, existe um valor ótimo para o limiar, nesse caso igual a 4, tal que a vazão é maximizada. A Figura 6(b) mostra métricas adicionais, como a taxa de descarte e a taxa de classificação incorreta, em função do limiar de *backlog* (vide Eq. (1)). Na medida em que a taxa de chegada aumenta, a taxa de descarte também aumenta, principalmente quando o limiar de *backlog* é alto.

O compromisso entre a taxa de descarte e a taxa de classificação incorreta é ilus-



(a) Vazão. (b) Decomposição da taxa de chegada.

(c) Acurácia. (d) Probabilidade de Perda.

Figura 6. Efeito do limiar de *backlog* nas métricas de interesse.

trado nas Figuras 6(c) e 6(d). Na medida em que o limiar de *backlog* aumenta, a acurácia aumenta, mas a probabilidade de perda também aumenta. O que determina o limiar ótimo é a sensibilidade de cada uma dessas métricas com relação a variações no limiar de *backlog*. Conforme ilustrado nas figuras, a probabilidade de perda se torna mais sensível à medida em que a taxa de chegada aumenta, favorecendo um aumento correspondente do limiar de *backlog*.

6. Conclusões e Trabalhos Futuros

Este trabalho mostrou que o contexto tem um impacto significativo no desempenho das DNNs de saída antecipada, o que exige uma abordagem adaptativa para a parametrização da saída antecipada bem como para lidar com *buffers*. Para lidar com mudanças de contexto, apresentou-se um algoritmo online que escolhe eficientemente o limiar ótimo de confiança para diferentes contextos. Para lidar com *buffers*, apresentou-se uma abordagem baseada em CTMDP. O algoritmo online é baseado em *multi-armed bandits* para equilibrar o compromisso entre exploração e exploração, escolhendo efetivamente o melhor limiar de confiança, enquanto que o algoritmo offline para lidar com *buffers* visa encontrar o limiar de *backlog* ótimo. Os experimentos demonstram a convergência de ambos os algoritmos para todos os contextos e vários valores de sobrecarga.

Este trabalho abre inúmeros caminhos para investigações futuras, incluindo 1) unificação dos algoritmos online (MAB) e offline (CTMDP), 2) consideração da possibilidade de mais de uma saída antecipada, 3) análise do impacto do limiar de confiança α sobre a confiança de classificação na camada final, C_L, ou seja, da interdependência entre as ações e os parâmetros do problema, p.ex., usando algoritmos online para aprender C_L.

Referências

Auer, P. et al. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235.

Casale, G. and Roveri, M. (2023). Scheduling inputs in early exit neural networks. *IEEE Transactions on Computers*.

Fang, B., Zeng, X., Zhang, F., Xu, H., and Zhang, M. (2020). Flexdnn: Input-adaptive on-device deep learning for efficient mobile vision. In *IEEE/ACM Symposium on Edge Computing (SEC)*, pages 84–95.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 770–778.

Hu, C., Bao, W., Wang, D., and Liu, F. (2019). Dynamic adaptive DNN surgery for inference acceleration on the edge. In *INFOCOM*, pages 1423–1431.

Ju, W., Bao, W., Ge, L., and Yuan, D. (2021a). Dynamic early exit scheduling for deep neural network inference through contextual bandits. In *ACM Conf. Info*, pages 823–832.

Ju, W., Bao, W., Yuan, D., Ge, L., and Zhou, B. B. (2021b). Learning early exit for deep neural network inference on mobile devices through multi-armed bandits. In *IEEE/ACM CCGrid*, pages 11–20.

Kang, Y., Hauswald, J., Gao, C., Rovinski, A., et al. (2017). Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. In *ACM Computer Architecture News*, volume 45, pages 615–629.

Kim, G. and Park, J. (2020). Low cost early exit decision unit design for cnn accelerator. In *IEEE Int. SoC Design Conf.*, pages 127–128.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images. *Masters Thesis: University of Toronto*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

Laskaridis, S., Venieris, S. I., Almeida, M., Leontiadis, I., and Lane, N. D. (2020). SPINN: synergistic progressive inference of neural networks over device and cloud. In *ACM MobiCom*, pages 1–15.

Li, E., Zeng, L., Zhou, Z., and Chen, X. (2019). Edge ai: On-demand accelerating deep neural network inference via edge computing. *IEEE Transactions on Wireless Communications*, 19(1):447–457.

Pacheco, R., Couto, R., and Simeone, O. (2021a). Calibration-aided edge inference offloading via adaptive model partitioning of deep neural networks. In *IEEE Int. Conf. Communications (ICC)*, pages 1–6.

Pacheco, R., Oliveira, F. R., and Couto, R. (2021b). Early-exit deep neural networks for distorted images: providing an efficient edge offloading. In *IEEE GLOBECOM*, pages 1–6.

Pacheco, R. G., Bajpai, D. J., Shifrin, M., Couto, R. S., Menasché, D. S., Hanawal, M. K., and Campista, M. E. M. (2024). UCBEE: A Multi Armed Bandit Approach for Early-Exit in Neural Networks. *IEEE Transactions on Network and Service Management*.

Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

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

Shifrin, M., Menasché, D. S., Cohen, A., Goeckel, D., and Gurewitz, O. (2020). Optimal phy configuration in wireless networks. *IEEE/ACM Transactions on Networking*, 28(6):2601–2614.

Teerapittayanon, S., McDanel, B., and Kung, H.-T. (2016). Branchynet: Fast inference via early exiting from deep neural networks. In *IEEE Int. Conf. on Pattern Recognition (ICPR)*, pages 2464–2469.

Wang, M., Mo, J., Lin, J., Wang, Z., and Du, L. (2019a). Dynexit: A dynamic early-exit strategy for deep residual networks. In *IEEE Int. Workshop on Signal Processing Systems (SiPS)*, pages 178–183.

Wang, Z., Bao, W., et al. (2019b). SEE: Scheduling early exit for mobile dnn inference during service outage. In *ACM Conf. Modeling, Analysis and Simulation of Wireless and Mobile Sys.*, pages 279–288.