Reaproveitamento de TV Boxes para Aplicação de Contagem de Pessoas na Borda em Cidades Inteligentes

Gabriel Massuyoshi Sato[1], Gustavo P. C. P da Luz[1], Luis Fernando Gomez Gonzalez[1], Juliana Freitag Borin[1]

[1]Instituto de Computação (UNICAMP), Campinas – São Paulo – Brasil

{g172278, g271582}@dac.unicamp.br, {gonzalez, jufborin}@unicamp.br

Abstract. *In recent years, large quantities of illegal TV Box equipment have been confiscated in Brazil. According to news released in March of this year, it is estimated that there are around 2.5 million TV Boxes in the warehouses of the Federal Revenue Service. On the other hand, the advancement of smart cities applications based on Internet of Things (IoT) and machine learning has driven research in edge computing using hardware-constrained devices. This article presents a study on the feasibility of repurposing TV Boxes for edge computing in an application of people counting from images collected by cameras. A comparison between the performance of 2 models of TV Boxes and widely used hardware in IoT solutions during the execution of the YOLOv8 and EfficientDet deep learning models demonstrates this feasibility.*

Resumo. *Nos últimos anos, grandes quantidades de equipamentos de TV Box ilegais tem sido apreendidos no Brasil. Segundo notícia divulgada em março deste ano, estima-se que haja em torno de 2,5 milhões de TV Boxes nos depósitos da Receita Federal. Por outro lado, o avanço das aplicações baseadas em Internet das Coisas (IoT) e aprendizado de máquina em cidades inteligentes tem impulsionado pesquisas em computação na borda usando dispositivos com limitação de hardware. Este artigo apresenta um estudo sobre a viabilidade de se reaproveitar TV Boxes para computação na borda em uma aplicação de contagem de pessoas a partir de imagens coletadas por câmeras. Uma comparação entre o desempenho de 2 modelos de TV Boxes e hardwares amplamente utilizados em soluções de IoT durante a execução dos modelos de aprendizado profundo YOLOv8 e EfficientDet evidenciam esta viabilidade.*

1. Introdução

Em 2019 foram descartados 53,6 milhões de toneladas de resíduos eletrônicos (*e-waste*) e estima-se que esta quantidade aumente em aproximadamente 40% até 2030 [Eisenstein 2022]. Tal estatística vai de encontro aos Objetivos de Desenvolvimento Sustentável (ODSs), particularmente, à meta cinco do ODS 12 que consiste em, "*até 2030, reduzir substancialmente a geração de resíduos por meio da prevenção, redução, reciclagem e reuso*"[Nations 2015].

No Brasil, a Receita Federal têm apreendido inúmeras TV Boxes (estima-se que 2,5 milhões destes equipamentos estejam armazenados em depósitos)[1], pequenos aparelhos eletrônicos importados de forma irregular para realizar a pirataria de canais pagos,

[1]Cigarro, bebida e aparelho de 'gatonet': como universidades usam apreensões da Receita na pesquisa

filmes e outros conteúdos restritos. Este tipo de material, quando apreendido, costuma ser enviado para destruição, o que, além de ser um processo custoso para a Receita Federal e, em consequência, para os cofres públicos, gera uma quantidade enorme de resíduos eletrônicos que precisam ser descartados. A fim de evitar a produção de *e-waste* e dar um destino mais apropriado para os equipamentos apreendidos, a Receita Federal estabeleceu parcerias com instituições de ensino superior para que pudessem explorar as oportunidades de uso dos equipamentos, especialmente, para fins de educação e inclusão digital [Ribeiro et al. 2023]. A reutilização das TV boxes se encaixa no conceito de reaproveitamento da economia circular, realizando uma recontextualização da função do produto para evitar o seu descarte, assim como aponta [Morseletto 2020], promovendo um *loop* aberto e prolongando o tempo de utilização do dispositivo [Willskytt et al. 2016].

A Unicamp é uma das instituições que estabeleceu parceria com a Receita Federal tendo recebido quase 1.000 TV boxes[2][3]. O reaproveitamento desses equipamentos tem sido explorado por pesquisadores da universidade em diversas frentes, tais como computadores de baixo custo para acesso a máquinas na nuvem com finalidade educacional [Ribeiro et al. 2023] e plataforma para coleta de dados de sensores [de Carvalho Borges and Borin 2023], também pela prefeitura do campus na visualização de dados[4] em projetos de Internet das Coisas (IoT).

Neste trabalho, exploramos o reaproveitamento das TV Boxes como dispositivos de borda para aplicações de cidades inteligentes que envolvem aprendizado de máquina. O aprendizado de máquina tem sido utilizado nas mais variadas aplicações dentro do contexto de cidades inteligentes, tais como mobilidade, segurança, energia (*smart grids*) e saúde [Band et al. 2022]. Essas aplicações, no entanto, envolvem o processamento de grandes quantidades de dados e, muitas delas, requerem baixa latência e privacidade no tratamento dos dados sendo, portanto, a computação na borda considerada uma tecnologia habilitadora [Khan et al. 2020]. Como estudo de caso para este trabalho escolhemos uma aplicação que envolve a contagem de pessoas com base em imagens capturadas por câmeras. As imagens são capturadas em frente aos restaurantes universitários da Unicamp com o objetivo de oferecer informações para os usuários sobre a situação das filas de acesso. Em um contexto mais abrangente de cidades inteligentes, a estimativa do número de pessoas presentes em um local público pode ser utilizada para embasar tomadas de decisões em outras aplicações como aquelas mencionadas anteriormente.

A principal contribuição deste artigo está no estudo da viabilidade de reaproveitamento de TV Boxes para aplicações em cidades inteligentes que envolvem aprendizado de máquina na borda. O estudo se dá por meio de uma avaliação comparativa entre dois modelos de TV Boxes e dois modelos de *hardware* de propósito geral amplamente utilizados em aplicações de IoT considerando o desempenho ao executar modelos de aprendizado profundo para contagem de pessoas em imagens. Adicionalmente, este trabalho mostra que a conversão de YOLOv8 para TensorFlow Lite (TFLite) [TensorFlow Lite] ocupa menos memória do que a versão original, sendo uma alternativa viável para todos os dispositivos de borda testados respeitando latência e uso de processamento, além de manter um alto desempenho de modelos.

[2]Receita Federal destina 200 TV Boxes para a Unicamp.
[3]Receita Federal destina 700 TV Boxes para a Unicamp.
[4]Smart Campus – TV Box nos restaurantes universitários.

O restante deste artigo está organizado da seguinte forma: a Seção 2 apresenta trabalhos relacionados, a Seção 3 descreve os materiais e métodos utilizados no estudo proposto, a Seção 4 discute os resultados obtidos e, finalmente, a Seção 5 traz as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

[Moreira et al. 2022] propõem a plataforma AgroLens, que envolve a coleta de dados de sensores de uma fazenda e a detecção de doenças em plantas por meio de análise de imagens. Na solução proposta, os autores avaliam a viabilidade de se usar um aparelho de TV Box como dispositivo de borda, tanto para o recebimento dos dados de sensores, por meio de um *broker* MQTT, quanto para a análise de imagens coletadas por um smartphone. Para a análise das imagens, os autores consideraram três modelos de redes neurais convolucionais (CNN): AlexNet [Krizhevsky et al. 2017], DenseNet [Huang et al. 2017] e SqueezeNet [Iandola et al. 2016]. Os três modelos foram avaliados com relação à acurácia, ao tempo de predição e ao consumo de CPU. A TV Box utilizada nos testes tinha um chipset Cortex-A53, 16GB de armazenamento e 2GB de RAM. A rede SqueezeNet apresentou o melhor resultado de custo-benefício dado que retornou uma acurácia (97,94%) próxima a obtida com as outras duas redes, porém com menor consumo de CPU e menor tempo de predição.

[Monti et al. 2022] propõem uma solução para contagem de pessoas em uma sala de aula. A solução é baseada em imagens de câmeras Intel RealSense (esta câmera consiste em um projetor de infravermelho, um par de sensores de profundidade e um sensor RGB) e computação na borda viabilizada por um dispositivo Raspberry Pi 4 Modelo B. A contagem de pessoas é realizada por um modelo customizado da YOLOv3 [Redmon and Farhadi 2018], cujos pesos foram otimizados para lidar com salas de aula pequenas e grandes. Ao analisar os erros produzidos pelo sistema, com foco nos resultados de falso positivo, os autores observaram que a maioria dos erros ocorreu devido à contagem excessiva quando mais de 30 alunos estavam presentes em uma sala de aula. Ao observar as imagens, os autores notaram que a área crítica correspondia às fileiras de trás da sala de aula, onde as *bounding boxes* eram muito pequenas e as pessoas apareciam mais próximas umas das outras.

[Mohd Razif et al. 2024] apresentam um sistema de computação na borda para contagem de pessoas durante a entrada ou saída de uma loja de varejo. Os autores comparam o desempenho dos modelos YOLOv5 e Mobilenet-SSD com relação a acurácia e tempo de processamento em quadros por segundo (FPS) em um dispositivo NVIDIA Jetson Nano (CPU Quad-core ARM Cortex-A57 MPCore, RAM 4 GB 64-bit LPDDR4, GPU NVIDIA Maxwell com 128 núcleos NVIDIA CUDA e 0.5 TFLOPs- FP16). Para o modelo YOLOv5 os autores reportaram uma precisão média (*average precision* - AP) de 44,6% com um limiar de Interseção sobre União (Intersection over Union - IoU) de 50%, enquanto para o modelo Mobilenet-SSD o AP foi de 21,3%. Por outro lado, a Mobilenet-SSD foi capaz de processar as imagens à 30 FPS, enquanto a YOLOv5 alcançou 6 FPS. Este resultado evidencia a importância de se avaliar o custo-benefício entre acurácia e tempo de processamento entre diferentes modelos que rodam na borda para aplicações que precisam de respostas em tempo real.

Nosso trabalho se destaca pela utilização de diferentes *hardwares* disponíveis no

mercado, sendo dois deles Raspberry Pi's e dois TV Boxes, rodando arquiteturas de redes neurais convolucionais conhecidas na literatura: YOLOv8 e EfficientDet, com o objetivo de contar pessoas em imagens. Também foi testado o impacto da utilização do Tensorflow Lite Runtime no processo de inferência, com medidas dos recursos de CPU e memória e o tempo de inferência. Dessa forma, neste trabalho evidenciamos que dispositivos com hardware limitado são viáveis para utilização na borda de uma cidade inteligente para contagem de pessoas em aplicações como a disponibilizada pelo projeto Smart Campus, na Unicamp. Para essa finalidade, os TV Boxes são de particular interesse por serem reaproveitados, como discutido na próxima seção.

3. Materiais e Métodos

Nossa proposta foca no reaproveitamento das TV Boxes que foram concedidas no contexto da parceria entre Unicamp e Receita Federal. Escolhemos explorar dois dos modelos recebidos, o Youit Tx2 e o BTV E10 devido às suas limitações e capacidades diferentes de processamento e arquitetura. Utilizamos como comparativo as plataformas de *hardware* Raspberry Pi 3 Modelo B+ e Raspberry Pi 4 Modelo B, que são dispositivos embarcados amplamente utilizados em aplicações de IoT. A Figura 3 mostra uma imagem de cada um dos dispositivos considerados, enquanto a Tabela 1 apresenta as especificações técnicas.

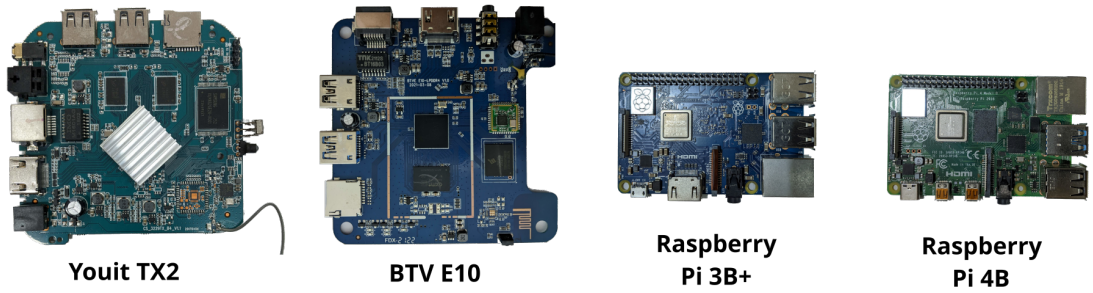

Figura 1. Fotografia dos quatro dispositivos utilizados nos testes

Tabela 1. Comparação das especificações dos dispositivos utilizados

Dispositivo	CPU			Memória RAM	
	Arquitetura	Nº de núcleos	Frequência	Tecnologia	Capacidade
Raspberry Pi 3B+	ARM Cortex-A53 (64 bits)	4	1,4 GHz	LPDDR2	1 GB
Raspberry Pi 4B	ARM Cortex-A72 (64 bits)	4	1,5 GHz [5]	LPDDR4	2 GB [5]
TV Box Gen 1 (Tx2)	ARM Cortex-A7MP (32 bits)	4	1,2 GHz	LPDDR3	2 GB
TV Box Gen 2 (E10)	ARM Cortex-A53 (64 bits)	4	1,9 GHz	LPDDR4	2 GB

[5]A Raspberry Pi 4B possui diversas opções com memória entre 1 GB e 8 GB. Além disso, dependendo da versão, ela pode ser encontrada com CPU de frequência de 1,5 GHz ou 1,8 GHz. Na Tabela 1, descrevemos o modelo utilizado para os testes.

Para realizarmos os testes nas TV Boxes, utilizamos os tutoriais [6] [7] disponibilizados pela equipe do projeto Smart Campus Unicamp para instalação de distribuições Linux nos equipamentos. Na TV Box Gen 1 foi instalado o Armbian 24.2.5 Bookworm para RK322X e na TV Box Gen 2, o Armbian 24.5.0 Bookworm para Aml-s9xx. Dessa forma, conseguimos tornar os dispositivos em computadores com hardware restrito que tem um sistema operacional Linux de propósito geral.

Vamos comparar o desempenho das duas TV Boxes em um estudo de caso utilizando dois modelos de aprendizado de máquina pré-treinados que realizam análise em imagens para contagem de pessoas. São eles:

1. YOLOv8 [Jocher et al. 2023]: Vindo da família de modelos YOLO, que significa *You Only Look Once*, já que é um modelo que detecta objetos em uma imagem com somente uma passada na rede, diferentemente de detectores que propõem regiões e depois processam os seus candidatos. Na versão 8, utiliza uma versão modificada do *backbone CSPDarknet53*, com uma camada *Spatial Pyramid Pooling Fast* (SPPF) para detectar várias escalas de objetos em uma imagem. Neste trabalho, foram testadas as versões "n"e "x", correspondendo à versão mais leve e mais pesada. Para os dispositivos de 32 bits, foi necessário converter os modelos para TFLite, uma alternativa para aprendizado de máquina na borda, reduzindo o seu tamanho e acelerando o tempo de inferência.
2. EfficientDet [Tan et al. 2020]: Rede que utiliza a *EfficientNet* como *backbone* combinado com uma *Weighted Bi-directional Feature Pyramid Network* (BiFPN), que detecta objetos e faz uma fusão de características (*features*) em várias escalas. Também é uma rede de passada única, o que motivou a escolha para este trabalho, além do uso popular em dispositivos de borda. Foram testadas as versões "D0"e "D4", correspondendo à versão mais leve e a mais pesada possível para os dispositivos testados, de forma a ter um tempo comparável com a YOLOv8. Foram utilizados também os modelos na versão lite, otimizados para dispositivos na borda.

Os modelos escolhidos possibilitam um comparativo entre a arquitetura mais antiga (2020) e consolidada para dispositivos restritos da EfficientDet com a rede moderna YOLOv8 (2023). Foram testadas tanto as redes pré-treinadas disponíveis pelos pacotes Ultralytics [Jocher et al. 2023], no caso da YOLO, e Kaggle Hub [Kaggle Hub Models], no caso da EfficientDet, bem como as mesmas arquiteturas convertidas para o Tensorflow Lite. A exceção foi a TV Box Tx2, na qual não foi possível rodar a versão completa do Tensorflow [TensorFlow] e Ultralytics, devido a arquitetura de 32 bits do processador.

Ao testar essas variações das arquiteturas, pudemos investigar o impacto dessa conversão em termos de desempenho da inferência e, consequentemente, da capacidade de uso das TV Boxes para aplicações de análise de imagens. Para isso, usaremos as métricas de tempo de inferência para cada imagem, assim como o uso de memória RAM e a carga de trabalho no processador durante o processo de análise. É importante observar que o uso do processador não é uma métrica diretamente interpretável: nos cenários analisados ele parece ter uma correlação com outros gargalos do *hardware*, ou seja, um uso de processador baixo demonstra que a aplicação encontra outros pontos com gargalo,

[6] Tutorial de instalação de Linux para TV Box - modelo Tx2

[7] Tutorial de instalação de Linux para TV Box - modelo BTVE10

como acesso à memória e ao armazenamento. Isso fica evidente no Raspberry Pi 3B+ que necessita de memória SWAP para rodar o modelo EfficientDet D4 e o modelo YOLOv8x. Diferentemente dos modelos convertidos, os modelos EfficientDet lite0 e lite4 foram baixados diretamente no formato TFLite e todos os modelos desse formato foram usados com o Tensorflow Lite Runtime, independente do *hardware* testado. Essa escolha foi feita para permitir usar uma única ferramenta de execução da inferência tanto em sistemas ARM 32 bits quanto em ARM 64 bits.

3.1. Estudo de caso - Contagem de Pessoas

Contagem de pessoas em um ambiente é uma operação computacional muito comum utilizada em diversas aplicações [Gao et al. 2024]. Em cidades inteligentes, por exemplo, a detecção de pessoas possibilita o monitoramento para fins de segurança pública e turismo. Além disso, há a possibilidade de uma tomada de decisão prévia para situações críticas que podem acontecer, detectadas pela quantidade de pessoas em um determinado local [Collini et al. 2024].

Nosso estudo de caso tem como objetivo realizar a contagem de pessoas na fila de um dos restaurantes universitários da Unicamp. Atualmente, o sistema em funcionamento realiza a inferência para a detecção de pessoas na nuvem para disponibilização da informação aos usuários finais no aplicativo da universidade [Prefeitura Universitária Unicamp]. Dessa forma, em nossos testes utilizaremos as imagens já coletadas durante dois dias.

Realizando o processo de contagem de pessoas localmente, além do foco no reaproveitamento dos dispositivos apreendidos, podemos desfrutar das vantagens da computação na borda. Com o atual sistema, onde todas as fotos são armazenadas na nuvem para processamento, deve-se criar camadas a mais de segurança na comunicação entre as câmeras e o servidor. Com a análise na borda, retiramos a etapa de carregamento das fotos na rede, evitando ataques durante a transmissão dos dados, além de diminuir drasticamente o consumo de largura de banda.

Para o estudo de caso, foram utilizadas 346 fotos tiradas em horários distintos durante dois dias, 01/04/2024 e 02/04/2024. Foram escolhidas somente imagens com no mínimo uma pessoa, selecionando as imagens de interesse já que o projeto se trata de contagem de pessoas. A anotação foi realizada por três anotadores distintos a fim de evitar vieses na hora de avaliar o desempenho dos modelos.

Figura 2. Exemplo da imagem de imagem da fila após detecção - [Prefeitura Universitária Unicamp]

4. Avaliação

Para a avaliação de desempenho foram utilizadas as métricas:

1. Erro Médio Absoluto, ou *Mean Absolute Error* (MAE), sendo quanto menor melhor.
2. Raiz Quadrada da Média dos Quadrados dos Erros, ou *Root Mean Square Error* (RMSE), sendo quanto menor melhor.
3. Média do tempo de inferência de cada imagem de acordo com o dispositivo utilizado.
4. Uso de memória RAM.
5. Carga de trabalho no processador.

O MAE mede a média da diferença absoluta entre os valores reais e preditos e o RMSE penaliza erros maiores, sendo mais sensível a *outliers*, porém complementando a análise de desempenho. Outras métricas de avaliação de problemas de regressão como o Coeficiente de Determinação e Média dos Quadrados dos Erros foram desconsideradas para este trabalho. Além das métricas dos modelos de regressão, consideramos o tempo médio de inferência, uso de memória e carga de trabalho para medir o desempenho do sistema como um todo e avaliar quais combinações de modelos e dispositivos são viáveis.

A Tabela 2 indica os resultados das métricas de aprendizado de máquina para quatro modelos diferentes de aprendizado profundo, na versão original e otimizada para dispositivos restritos para a contagem de pessoas no restaurante universitário utilizando 346 imagens.

Tabela 2. Comparação das Métricas de Avaliação dos Modelos em Problema de Regressão

Modelo	MAE	RMSE
YOLOv8n	8.78	14.30
YOLOv8n (TFLite)	8.77	14.18
YOLOv8x	7.63	13.38
YOLOv8x (TFLite)	7.75	13.56
EfficientDet D0	9.57	15.50
EfficientDet lite0	14.44	21.89
EfficientDet D4	8.44	14.47
EfficientDet lite4	13.02	20.05

Os resultados de desempenho do modelo variaram de acordo com o anotador e com o horário do dia. Existe uma dificuldade de encontrar uma concordância entre anotadores [Artstein 2017] de qual é o rótulo de imagens, principalmente as mais cheias com obstrução de pessoas. A diferença entre o número de pessoas predito e o real por foto de acordo com o horário está exibida na Figura 3. Observa-se que independente do modelo executado, o erro se mostrou maior para os horários em que a fila estava cheia, o que pode indicar uma necessidade de tratar melhor esses casos, se houver necessidade de boa exatidão no número absoluto de pessoas. A curva do melhor modelo em termos de métricas de aprendizado de máquina mostra que existe um limite de pessoas que ele acerta, por

volta de 30 pessoas. Fica evidente a diferença entre o melhor e o pior modelo, que é ainda mais aparente em horários de pico.

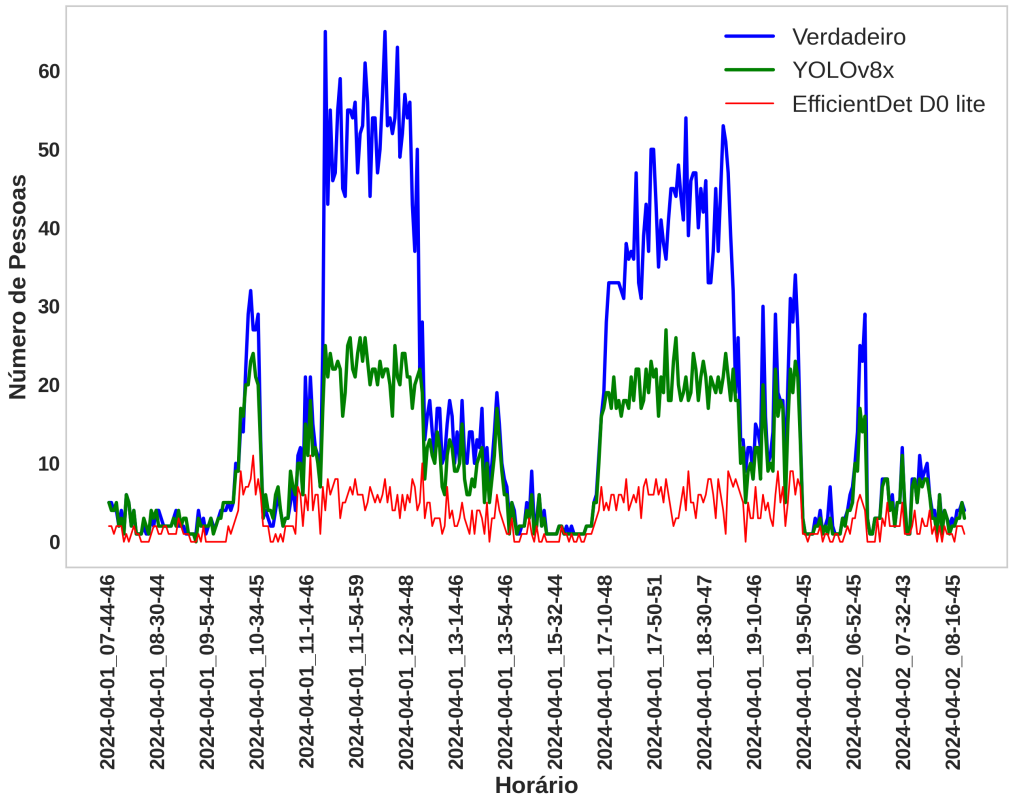

Figura 3. Valor Verdadeiro e Predito pelo melhor e pior Modelo pelos Horários.

A Tabela 3 indica os resultados do tempo de inferência para os diferentes dispositivos. A Figura 4 indica como os tempos se compararam, sendo possível analisar o impacto do aumento de parâmetros no tempo de inferência, com uma grande diferença para os modelos mais complexos de cada família de modelos, como esperado.

Tabela 3. Comparação dos Tempos de Inferência em segundos por Dispositivo

Modelo	TV Box Gen 1	TV Box Gen 2	Raspberry Pi 3B+	Raspberry Pi 4B
YoloV8n	-	1.11 ± 0.36	3.99 ± 1.31	0.86 ± 0.06
YoloV8n (TFLite)	5.18 ± 0.04	0.38 ± 0.0	1.19 ± 0.03	0.53 ± 0.0
YoloV8x	-	19.03 ± 0.8	97.87 ± 21.06	12.63 ± 1.30
YoloV8x (TFLite)	92.17 ± 0.89	9.93 ± 0.14	34.24 ± 0.84	18.08 ± 0.14
EfficientDet D0	7.14 ± 0.04	0.68 ± 0.0	1.91 ± 0.07	0.71 ± 0.0
EfficientDet lite0	0.66 ± 0.01	0.16 ± 0.0	0.54 ± 0.05	0.12 +- 0.0
EfficientDet D4	97.24 ± 0.49	10.23 ± 0.01	151.61 ± 7.18	7.71 ± 0.08
EfficientDet lite4	7.71 ± 0.08	1.98 ± 0.02	6.47 ± 0.09	1.37 ± 0.02

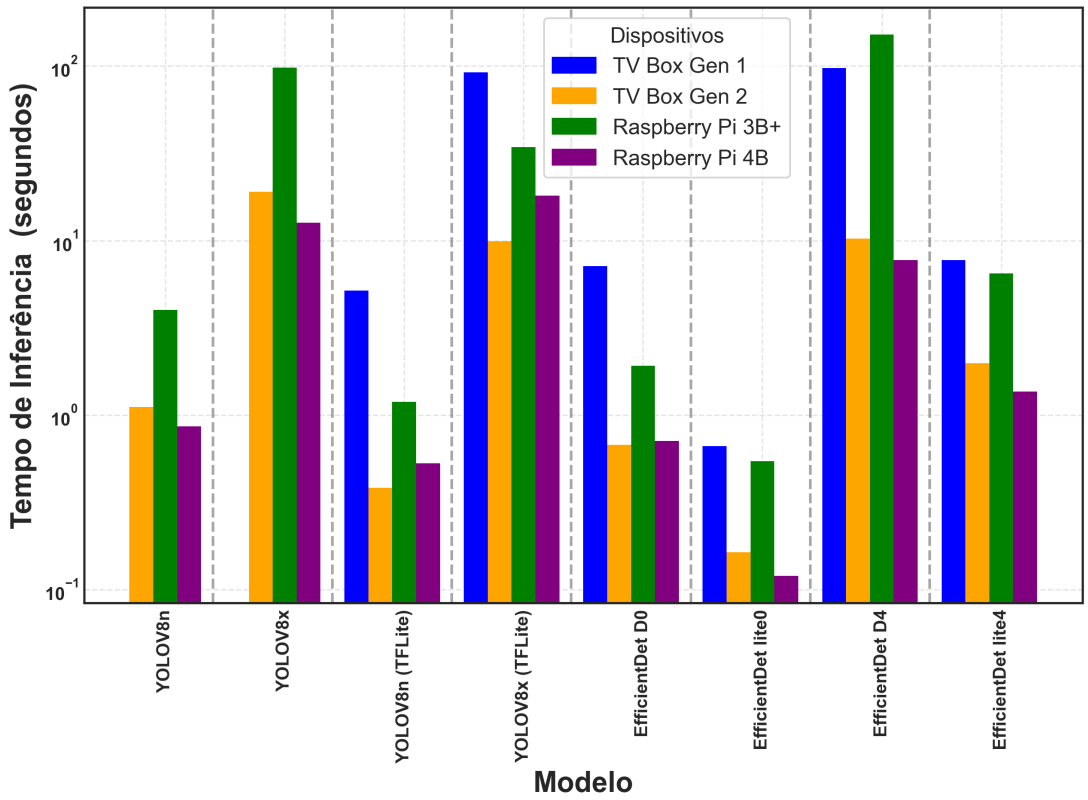

Figura 4. Tempo de Inferência para diferentes Modelos e Dispositivos.

O tempo do Raspberry Pi 3B+ é comparável com a primeira geração de TV Box, tendo desempenho superior para modelos simples e inferior para modelos que necessitem de mais de 1 GB de memória RAM. Já o tempo de inferência do Raspberry Pi 4B é comparável com a segunda geração de TV Box. Além disso temos que, do ponto de vista de tempo de inferência, a YOLOv8x é comparável com a EfficientDet D4. O impacto de usar a EfficientDet na versão lite é observado nas duas versões, o que mostra os ganhos de reduzir o tamanho do modelo em memória RAM para dispositivos na borda [TensorFlow Lite Runtime]. Para uma aplicação prática de contagem de pessoas em uma solução de cidade inteligente, o tempo de inferência pode ser uma das variáveis mais críticas, por isso realizar otimizações nos modelos traz ganhos práticos para esse contexto com restrições.

Tabela 4. Uso de Processamento e Memória

	YOLOv8x			YOLOV8x (TFLite)	
Dispositivo	**CPU (%)**	**Mem (MB)**	**SWAP (MB)**	**CPU (%)**	**Mem (MB)**
TV Box Gen 1	-	-	-	74.53	365.29
Raspberry Pi 3B+	66.96	529.11	709.32	76.65	573.51
TV Box Gen 2	89.08	797.84	-	92.83	548.11
Raspberry Pi 4B	85.19	859.71	-	92.43	583.53

A Tabela 4 mostra a utilização de recursos nos dispositivos com o modelo com

menor MAE na versão tradicional e TFLite. Podemos notar que a segunda geração da TV Box consegue realizar a tarefa de contagem de pessoas com certa facilidade, visto que a média do uso de memória na realização das inferências é de aproximadamente 0.78 GB utilizando o modelo tradicional. Assim era esperado, já que a Raspberry Pi 4B, dispositivo disponível no mercado que utilizamos para comparação, obteve medidas parecidas do uso de memória e processamento.

Já no caso da TV Box Gen 1, devido à restrição da arquitetura do processador de 32 bits, foi necessário a conversão do modelo tradicional para versão TFLite. Apesar disso, ela obteve um resultado muito bom na utilização de processamento e memória sem uma perda significativa nas métricas de avaliação MAE e RMSE. Podemos reafirmar isso, olhando para as medidas que a Raspberry Pi 3B+ ao rodar o modelo TFLite.

A Raspberry Pi 3B+ com o modelo tradicional da YOLOv8x obteve um uso de CPU consideravelmente mais baixo causado pelo tempo de espera no processador para acesso à memória SWAP. Dado isso, a conversão do modelo YOLOv8x para a versão TFLite se mostrou eficiente, possibilitando a utilização de um dispositivo com 1GB de memória RAM sem memória SWAP. Essa diminuição do uso de memória se dá por meio da reutilização dos tensores intermediários feita pelo TFLite com o intuito de minimizar a alocação total dos objetos de buffer de memória [TensorFlow Lite Runtime].

De todos os modelos testados, o melhor resultado de MAE e RMSE foi obtido pelo YOLOv8x, seguido de perto pelo mesmo modelo convertido para TFLite. Considerando aplicações que necessitem de resultados em menos de 100 segundos, esse é o melhor modelo para todos os dispositivos testados. Para aplicações nas quais um tempo de inferência menor é necessário, destaca-se o desempenho obtido pelo modelo YOLOv8n, com um MAE aproximadamente 4% inferior ao modelo EfficientDet D4, mas com o tempo de inferência ao menos uma ordem de magnitude menor, variando de acordo com o dispositivo usado. Dessa forma, verificamos que os modelos YOLOv8 na versão TFLite possuem um desempenho por tempo de inferência superior aos modelos EfficientDet, também na versão TFLite, em todos os dispositivos testados.

5. Conclusão

Este artigo apresentou um estudo sobre a viabilidade do reaproveitamento de TV Boxes para computação na borda em uma aplicação de contagem de pessoas em imagens utilizando aprendizado profundo. Diferentes modelos das famílias YOLOv8 e EfficientDet foram testadas em duas configurações de TV Boxes e dois mini-computadores amplamente utilizados em projetos de IoT. Os resultados alcançados apontam que as TV Boxes consideradas são capazes de executar um dos modelos de detecção de objetos mais recentes do estado da arte, a YOLOv8, retornando bom desempenho do modelo sem exaurir os recursos computacionais do dispositivo e com tempo de inferência dentro do aceitável para a aplicação considerada.

Os modelos foram testados usando o framework Tensorflow, bem como o Tensorflow Lite Runtime para realização de inferência, resultando em uma grande vantagem no tempo de processamento para o último. O desempenho de detecção de pessoas também foi testado em ambos os *frameworks*, demonstrando uma penalidade muito pequena ao utilizar o TFLite. Dentre as duas famílias de modelos testadas, obtivemos um melhor desempenho por tempo de inferência nos modelos YOLOv8, sendo nossa escolha para

utilização nos dispositivos restritos dentro da aplicação de contagem de pessoas na borda.

Todos os modelos testados demonstraram um erro de contagem absoluto maior para imagens com mais pessoas, demonstrando um aparente limite superior nas capacidades de detectar pessoas. Esse pode ser um problema importante a ser abordado em aplicações onde a exatidão da contagem absoluta é crítica.

Por último, também é importante observar que mesmo dispositivos com processadores mais antigos, como a primeira geração de TV Box testada, podem ser utilizados como dispositivos de borda, se consideradas as limitações impostas, principalmente por processadores de 32 bits. Nesse cenário, a utilização do TFLite Runtime se mostrou mandatória, sem opção da utilização da versão completa do Tensorflow. Isso seria um problema caso estivéssemos realizando treinamento do modelo no dispositivo, mas pode ser contornado para uso apenas para inferência, com a pequena vantagem de menor uso de memória RAM por conta do sistema operacional 32 bits.

Como trabalhos futuros, espera-se avaliar a resiliência e robustez dos equipamentos em ambientes externos reais, pois, especialmente no caso das TV Boxes, esta é uma avaliação que não se encontra na literatura. A aplicação de técnicas de processamento de imagens como a divisão da imagem em blocos ou a junção de mais câmeras com um processamento conjunto pode melhorar o desempenho do modelo de contagem de pessoas, principalmente em horários de pico. Além disso, os dispositivos podem ser testados para outras aplicações de cidades inteligentes avaliando o seu desempenho em outras tarefas, como a de classificação.

Agradecimentos

Este projeto foi apoiado pelo CNPq (processo 131653/2023-7) e programa PPI Softex, Acordo de Parceria nº 126/2022, financiado pelo Ministério da Ciência, Tecnologia e Inovações com recursos da Lei nº 8.248, de 23 de outubro de 1991 [01245.013778/2020-21]. Os autores agradecem à equipe do Projeto Smart Campus Unicamp pelo fornecimento do conjunto de dados.

Referências

[Artstein 2017] Artstein, R. (2017). Inter-annotator agreement. *Handbook of linguistic annotation*, pages 297–313.

[Band et al. 2022] Band, S. S., Ardabili, S., Sookhak, M., Chronopoulos, A. T., Elnaffar, S., Moslehpour, M., Csaba, M., Torok, B., Pai, H.-T., and Mosavi, A. (2022). When smart cities get smarter via machine learning: An in-depth literature review. *IEEE Access*, 10:60985–61015.

[Collini et al. 2024] Collini, E., Palesi, L. A. I., Nesi, P., Pantaleo, G., and Zhao, W. (2024). Flexible thermal camera solution for smart city people detection and counting. *Multimedia Tools and Applications*, 83(7):20457–20485.

[de Carvalho Borges and Borin 2023] de Carvalho Borges, I. G. C. and Borin, J. F. (2023). Abordagens para Superar Limitações de Hardware em TV Box RK3229 para uso em IoT. Technical Report IC-PFG-23-48, Instituto de Computação, Universidade Estadual de Campinas.

[Eisenstein 2022] Eisenstein, M. (2022). Short-circuiting the electronic-waste crisis. *Nature*, 611:8–10.

[Gao et al. 2024] Gao, M., Souri, A., Zaker, M., Zhai, W., Guo, X., and Li, Q. (2024). A comprehensive analysis for crowd counting methodologies and algorithms in internet of things. *Cluster Computing*, (27):859–873.

[Huang et al. 2017] Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Iandola et al. 2016] Iandola, F. N., Moskewicz, M. W., Ashraf, K., Han, S., Dally, W. J., and Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*, abs/1602.07360.

[Jocher et al. 2023] Jocher, G., Chaurasia, A., and Qiu, J. (2023). Ultralytics YOLO.

[Kaggle Hub Models] Kaggle Hub Models. `https://www.kaggle.com/models`. Acessado em 03/04/2024.

[Khan et al. 2020] Khan, L. U., Yaqoob, I., Tran, N. H., Kazmi, S. M. A., Dang, T. N., and Hong, C. S. (2020). Edge-computing-enabled smart cities: A comprehensive survey. *IEEE Internet of Things Journal*, 7(10):10200–10232.

[Krizhevsky et al. 2017] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90.

[Mohd Razif et al. 2024] Mohd Razif, M. H., Ismail, A. P., Che Abdullah, S. A., Shafie, M. A., Isa, I. S., Sulaiman, S. N., and Che Soh, Z. H. (2024). On edge crowd traffic counting system using deep learning on jetson nano for smart retail environment. *Journal of Advanced Research in Applied Sciences and Engineering Technology*, 42(1):1–13.

[Monti et al. 2022] Monti, L., Tse, R., Tang, S.-K., Mirri, S., Delnevo, G., Maniezzo, V., and Salomoni, P. (2022). Edge-based transfer learning for classroom occupancy detection in a smart campus context. *Sensors*, 22(10).

[Moreira et al. 2022] Moreira, R., Rodrigues Moreira, L. F., Munhoz, P. L. A., Lopes, E. A., and Ruas, R. A. A. (2022). Agrolens: A low-cost and green-friendly smart farm architecture to support real-time leaf disease diagnostics. *Internet of Things*, 19:100570.

[Morseletto 2020] Morseletto, P. (2020). Targets for a circular economy. *Resources, conservation and recycling*, 153:104553.

[Nations 2015] Nations, U. (2015). Transforming our world: The 2030 agenda for sustainable development. *New York: United Nations, Department of Economic and Social Affairs*, 1:41.

[Prefeitura Universitária Unicamp] Prefeitura Universitária Unicamp. `https://www.prefeitura.unicamp.br/cardapio/`. Acessado em 03/04/2024.

[Redmon and Farhadi 2018] Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *CoRR*, abs/1804.02767.

[Ribeiro et al. 2023] Ribeiro, A. F., Yoshioka, D. D. K., Ramirez, G., Silva, J. B., dos Santos, M., Souza, M. A., Cândido, M. A. S., Rodrigues, M. O., Pietrobom, V. R., and

Borin, J. F. (2023). Transformação de TV Boxes piratas em computadores de baixo custo suportados por nuvem computacional. Technical Report IC-PFG-23-49, Instituto de Computação, Universidade Estadual de Campinas.

[Tan et al. 2020] Tan, M., Pang, R., and Le, Q. V. (2020). Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790.

[TensorFlow] TensorFlow. `https://www.tensorflow.org/`. Acessado em 05/04/2024.

[TensorFlow Lite] TensorFlow Lite. `https://www.tensorflow.org/lite/guide`. Acessado em 05/04/2024.

[TensorFlow Lite Runtime] TensorFlow Lite Runtime. `https://blog.tensorflow.org/2020/10/optimizing-tensorflow-lite-runtime.html`. Acessado em 05/04/2024.

[Willskytt et al. 2016] Willskytt, S., Böckin, D., André, H., Ljunggren Söderman, M., and Tillman, A.-M. (2016). Framework for analysing resource-efficient solutions. In *Eco-Balance Conference 2016*.