Generative Adversarial Network and ResNet Comparison for Video Super Resolution in Smartphones

André Barbosa da Vitória[1], Francisco de Asis Boldt[2], and Hilario Seibel Junior[3],
[1 2 3] *Instituto Federal de Educação, Ciência e Tecnologia do Espírito Santo* Serra, Brazil
andre_davitoria@hotmail.com[1]

***Abstract*—With the latest evolutions of smartphone hardware regarding neural network processing and advances in Image Super Resolution, one question remains: What is the applicability of SR techniques already consolidated in smartphones? Several works focused on different characteristics of images have been developed and seek to understand how such networks can meet different segments. This work focuses on applying two already consolidated networks in the processing of photo-realistic images, SRResNet and SRGAN, in mobile devices to identify how such networks behave in smartphones of different generations, either in the image quality or in the response time of these networks. After evaluation, the SRResNet network had a better performance both in inferred image quality and lower latency, with a PSNR of 27.7075 versus 21.3843 and 0.19 milliseconds latency, compared to 0.20 milliseconds for SRGAN, thus showing that it is feasible to apply SR techniques already consolidated in smartphones.**

***Index Terms*—super resolution, smartphones, neural network, deep learning**

I. Introdução

As aplicações de super resolução (do inglês *Super Resolution*, ou apenas SR) têm se tornado cada vez mais populares em diversas áreas, como medicina [1], astronomia [2] e serviços de transmissão *streaming* [3]. Segundo [4], o conceito por trás da SR consiste em gerar uma imagem com maior resolução a partir de uma imagem de menor resolução.

Podemos dividir SR em duas categorias: *Single-frame Super Resolution* (comumente conhecida apenas como *Image Super Resolution* ou ISR), que foca na aplicação de técnicas de SR em uma única imagem, e *Multi-frame Super Resolution* (frequentemente conhecida como *Video Super Resolution* ou VSR), que tem como objetivo obter a imagem com super resolução a partir de um conjunto de imagens subsequentes de baixa resolução de uma mesma cena [5].

No trabalho desenvolvido por [6], foi proposta uma abordagem de *ISR* utilizando uma Rede Adversarial Gerativa (*Generative Adversarial Network*, ou apenas GAN), focada na restauração de texturas quando os fatores de aumento de escala das imagens são altos, como em $4x$. A SRGAN, denominada assim pelo autor, foi uma das primeiras técnicas focadas na inferência de imagens foto-realísticas naturais em escalas como $4x$, e utiliza uma rede neural residual (ResNet) como gerador, denominada pelo autor como SRResNet, contra uma CNN (*Convolutional Neural Network*) desenvolvida para ser uma rede descriminadora.

Os autores agradecem ao Ifes, apoio da FAPES e CAPES (processo 2021-2S6CD, nº FAPES 132/2021) por meio do PDPG (Programa de Desenvolvimento da Pós-Graduação, Parcerias Estratégicas nos Estados).

Nos últimos anos houve uma popularização dos serviços de transmissão de vídeo (*streaming*) e, com isso, a necessidade de soluções de VSR focadas em dispositivos móveis, mais especificamente *smartphones*. Enquanto soluções tradicionais de VSR focam em gerar um resultado de altíssima qualidade independente de recursos computacionais, soluções de VSR focadas em *smartphones* precisam lidar com as limitações dos dispositivos, que vão desde a capacidade de processamento limitado, de memória, e até da largura da banda de rede disponível para o dispositivo. Ainda assim, devem gerar uma resposta de alta qualidade e priorizando a redução do tempo de resposta [7].

Diferentes aplicações de SR em dispositivos móveis foram desenvolvidas, como em [8] que aplica ISR em fotografias geradas a partir da câmera de um *smartphone*, permitindo a substituição de *pipelines* ISP (*Image Signal Processor*) por modelos de redes neurais. Em [9], propõe-se a utilização de uma rede neural chamada de RenderSR para realizar *upscaling* das cenas de jogos em *smartphones* em tempo real. Além disso, existe ainda uma competição focada na construção de Redes Neurais para VSR em *smartphones* [7].

Dessa forma, a proposta deste trabalho é avaliar e comparar de maneira quantitativa as técnicas SRGAN e SRResNet, apresentadas no trabalho desenvolvido por [6], seguindo a metodologia apresentada por [7] e [10], utilizando o *REDS Dataset* [11] como base de treinamento, e como medida de avaliação de desempenho a combinação do PSNR (*Peak Signal-to-Noise Ratio*), do SSIM (*Structural Similarity Index Measure*) e do tempo de execução. Por fim, como ferramenta de avaliação dos modelos em *smartphone*, será utilizado o *AI Benchmark*, ferramenta disponibilizada por [7]. O sistema operacional utilizado nas comparação será apenas o *Android* e não será avaliada a largura de banda de rede. Ao fim, será identificado qual modelo possui o melhor desempenho segundo os resultados obtidos.

II. Materiais e Métodos

A. Conjunto de dados

O *REDS Dataset* (*REalistic and Dynamic Scenes dataset*) é um conjunto de dados criado especificamente para a *NTIRE Challenge*, uma competição focada em *video deblurring* (VD) e SR. Sua primeira menção data de 2019, e é composto por 30000 imagens de 1280×720 pixels (HD, do inglês *High Definition*), a serem utilizadas na comparação das técnicas de SR e VD da competição [11]–[13]. Delas, 24000 são para treinamento, 3000 para teste e 3000 para validação.

A Fig. 1 apresenta um exemplo da diferença entre as imagens contidas no *dataset*, comparando o zoom em um seguimento da imagem entre a imagem em qualidade original e sua contraparte reduzida à um quarto da qualidade ($4x$).

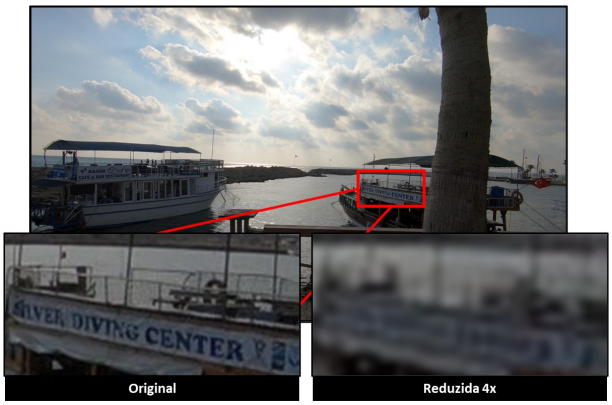


Fig. 1. Amostra de uma imagem do conjuntos de validação do *REDS Dataset*, comparando a diferença de qualidade entre a imagem original e sua redução à um quarto da resolução original.

B. Medidas de Avaliação

A avaliação dos modelos será baseada na metodologia proposta em [14] para a *Mobile AI & AIM 2022 Challenge*, a competição focada em avaliar modelos de redes neurais profundas em *smartphones*. Os principais critérios são fidelidade da imagem resultante, percepção e tempo de execução, e tais critérios são avaliados através da ferramenta *AI Benchmark* [1], um aplicativo móvel para *smartphones* que utilizam o sistema operacional Android.

Segundo [7], "esta ferramenta contém as versões mais recentes do Android NNAPI, TFLite GPU, Hexagon NN, Samsung Eden e MediaTek Neuron delegates, portanto, suportando todas as plataformas móveis atuais e fornecendo aos usuários a capacidade de executar redes neurais em NPUs, APUs, DSPs, GPUs e CPUs de *smartphones*".

A Fig. 2 apresenta as principais telas do aplicativo. A primeira é a tela de incio, que apresenta a pontuação do *smartphone* dentro do aplicativo. A segunda tela mostra o conjunto de opções disponíveis para o carregamento e a parametrização do modelo apresentado [7]. Por fim, a terceira tela apresenta o resultado da avaliação feita pelo aplicativo do modelo fornecido.

[1]https://ai-benchmark.com/download

Fig. 2. Carregamento e execução de modelo customizado utilizando TensorFlow Lite dentro do aplicativo *AI Benchmark* apresentado por [7]

A partir dos resultados obtidos pelo *AI Benchmark* serão avaliados os critérios de fidelidade através da medida PSNR, representada pelas Equação 3, e de percepção, calculada utilizando a medida SSIM, Equação 7. O terceiro critério é o tempo de execução, medido em milissegundos, usando tanto a CPU quanto a GPU do *smartphone*. Com isso o resultado final é obtido pela Equação 1, baseada na equação desenvolvida por [14]. O consumo de energia, que não foi possível de ser medido, foi substituído pelo tempo de execução TE dividido por δ (o tempo dos quadros em uma taxa de quadros de 30 FPS, ou seja $33,3ms$). O principal objetivo é obter uma solução eficiente, balanceando as medidas e o tempo de resposta.

O $PSNR$, representado pela Equação 3, é definido através do valor máximo de pixels L e do erro quadrático médio MSE entre as imagens. Já o $SSIM$, representado pela Equação 7, mede a semelhança estrutural entre imagens, comparando independentemente luminância (Equação 4), contraste (Equação 5) e estruturas (Equação 6) da imagem. Nesta medida, x é o valor dos pixels predito, y é o valor dos pixels reais, μ_x e μ_y são as média de x e y, σ_x e σ_y são as variâncias de x e y, σ_{xy} é a covariância de x e y, $c_1 = (k_1 \cdot L)^2$ e $c_2 = (k_2 \cdot L)^2$, sendo c_1 e , $c_3 = c_2/2$, e os pesos α, β, γ igual a 1 [4], [7], [11].

$$Score = \alpha * PSNR + \beta * (1 - (\frac{TE}{\delta})) \tag{1}$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 \tag{2}$$

$$PSNR = 10 \cdot \log_{10}\left(\frac{L^2}{MSE}\right) \tag{3}$$

$$l(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \tag{4}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \tag{5}$$

$$s(x,y) = \frac{\sigma_{xy} + c3}{\sigma_x \sigma_y + c3} \tag{6}$$

$$SSIM(x,y) = \left[l(x,y)^{\alpha} \cdot c(x,y)^{\beta} \cdot s(x,y)^{\gamma} \right] \tag{7}$$

III. Treinamento dos Modelos

A. Implementação

Para utilização do *AI Benchmark* para avaliação das redes treinadas, dois pré-requisitos precisam ser atendidos durante o desenvolvimento das redes: Os modelos devem ser desenvolvidos utilizando a API do Tensorflow 2, e os componentes utilizados pela implementação dos modelos devem ser compatíveis com Tensorflow Lite. Para isso, foi utilizada uma implementação baseada no *dataset* Div2K [2], adaptada para utilizar o *REDS Dataset*.

A seguir, as seções III-A1 e III-A2 explicam, respectivamente, os detalhes de implementação das redes SRResNet e SRGAN.

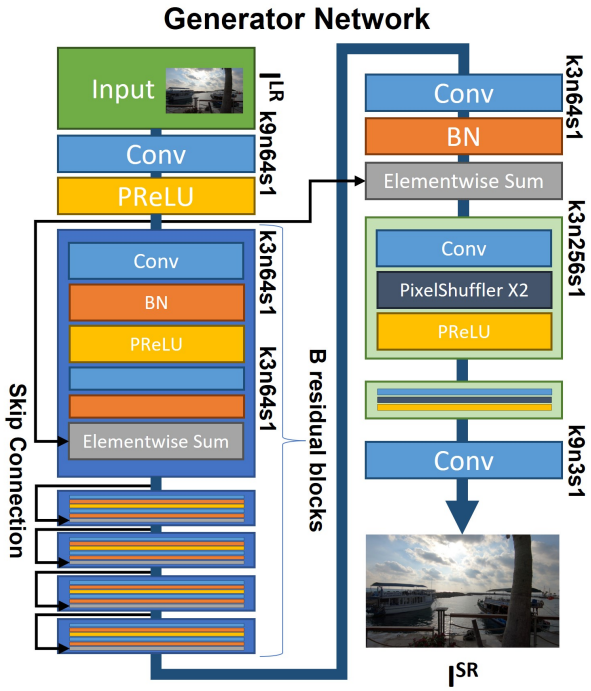
1) SRResNet: Para a construção e treinamento da ResNet proposta por [6] é necessário a normalização dos valores dos píxel das imagens de entrada. Dessa forna, foi adicionada uma camada de normalização antes da primeira convolução e, em seguida, adicionada a primeira camada de convolução combinada com uma função de ativação PReLU. Na sequência, foram adicionados os blocos residuais (16 ao todo), compostos por uma convolução seguida de uma normalização em lotes, uma função de ativação PReLU, outra convolução e normalização em lotes, e uma soma de elemento a elemento com o resultado do bloco anterior. Após os blocos residuais, uma convolução é aplicada, seguida da normalização em lotes e uma soma elemento a elemento com a saída da primeira ativação. Finalmente, é iniciado o processo de aumento de escala, realizado por 2 blocos compostos de uma convolução, seguida de uma camada de *PixelShuffle* (que altera o resultado da convolução de um domínio unidimensional para um domínio bidimensional, contendo altura e largura), ativando na sequência. Por fim, uma ultima convolução é realizada ajustando a saída da rede para o formato final da imagem. Todo esse processo é representado pela Fig. 3 construída com base na arquitetura proposta por [6].

Para o treinamento foi utilizado como função de otimização (*optimizer*) a *Adam* com uma taxa de aprendizado (*learning rate*) de $1e-4$, e MSE (*Mean Squared Error*) como função de perda (*loss*), ao longo de 1000 épocas com 100 passos por época, As imagens foram recortadas em blocos de 96 pixels de largura e altura rotacionados, invertidos e com ruídos adicionados às imagens em baixa resolução.

2) SRGAN: Para construção da GAN, foi utilizado um modelo da SRResNet treinado por 1000 épocas, apos isso foi utilizado este modelo treinado como Rede Generativa e treinada contra a rede descriminadora representada pela CNN na Fig. 4. Sendo esta uma rede focada em classificação, composta de uma camada de convolução combinada com uma função de ativação Leaky ReLU, seguida de 7 blocos de descriminação, compostos de uma convolução, uma normalização em lotes e uma função de ativação Leaky ReLU, variando o valor dos filtros (*filters* ou *output channels*) e os passos (*strides*). E por fim, duas camadas densa, a primeira com a mesma função de ativação das anteriores e a segunda com uma função de ativação *sigmoid* para realizar a classificação em imagem em alta resolução (verdadeira) ou super resolução (inferida).

[2]https://www.github.com/jlaihong/image-super-resolution

Fig. 3. Arquitetura da SRResNet construída por [6], sendo *kernel* k, o canal de saída (*filters* ou (*output channels*) n, e os passos (*stride*) s.



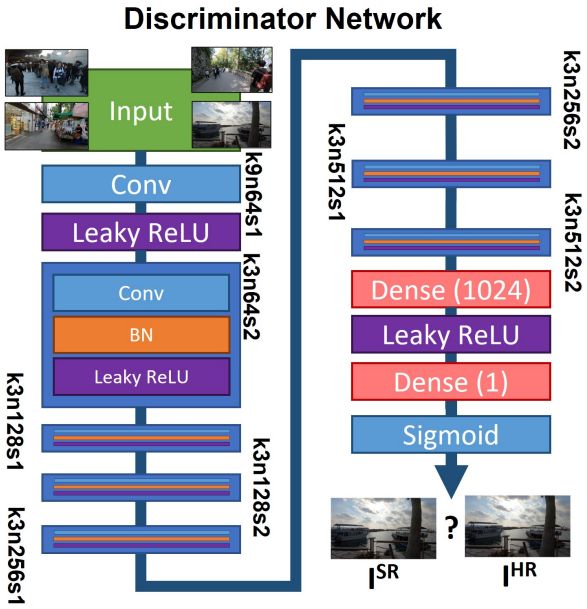
Para o treinamento da Rede Generativa foi alterado a taxa de aprendizado da função de otimização de 10^{-4} para um cronograma de decaimento constante por partes (*Piecewise Constant Decay*) com limites de 100000 e valores de 10^{-4} à 10^{-5}, a mesma foi aplicada para a Rede Discriminadora. Quanto a função de perda utilizada pela Rede Discriminadora foi a entropia cruzada binária (*Binary Cross-entropy*).

B. Implantação

Uma vez disponíveis os arquivos das redes neurais em TensorFlow Lite, foi iniciado o processo de testes de latência. Para carregar e executar os modelos, o *AI Benchmark* disponibiliza uma coleção de aceleradores para o processamento das redes, sendo eles:

- **CPU**: Utiliza a própria CPU do dispositivo para processar as instruções do modelo e realizar a inferência.
- **TFLite GPU Delegate**: Permite o uso de GPUs e outros processadores especializados por meio de *drivers* de hardware disponibilizados através das APIs do TensorFlow

Fig. 4. Arquitetura da rede descriminadora da SRGAN construída por [6], sendo kernel k, o canal de saída (*filters* ou *output channels*) n, e os passos (*stride*) s.



Lite[3].

- **Android NNAPI**: É uma API do Android C desenvolvida para executar operações com uso intenso de recursos computacionais para aprendizado de máquina em dispositivos Android[4].
- **Qualcomm Hexagon**: Permite o acesso a recursos de computação embarcados no Hexagon DSP para permitir aceleração de áudio, imagem, visão incorporada e computação heterogênea no DSP incorporado em processadores Snapdragon[5].
- **Qualcomm QNN DSP**: Permite a economia de tempo e esforço na otimização do desempenho de redes neurais em dispositivos com produtos Qualcomm AI utilizando a DSP para tal atividade[6].
- **Qualcomm QNN HTP**: Reduz o tempo e esforço na otimização do desempenho de redes neurais em dispositivos com produtos Qualcomm AI utilizando a módulos específicos da DSP em gerações a partir da versão 888 dos processadores Snapdragon, como o HVX (Hexagon Vector Extensions) e o HMX (Hexagon Matrix Extensions) para tal atividade[7][8].
- **Qualcomm QNN GPU**: Permite a economia de tempo e esforço na otimização do desempenho de redes neurais em dispositivos com produtos Qualcomm AI utilizando a GPU para tal atividade.[9]
- **MediaTek Neuron**: Permite execução mais eficiente de redes neurais em dispositivos equipados com processadores MediaTek[10]
- **Samsung ENN**: Permite execução eficiente de redes neurais pré-treinadas em dispositivos Samsung[11].

Dentre os aceleradores disponíveis tanto para o Redmi Note 8 quanto o Samsung S22 estavam, CPU, TFLite GPU Delegate, Android NNAPI, Qualcomm QNN GPU. Além destes o Samsung Galaxy S22 possuía também o Qualcomm QNN HTP. Todos os testes foram realizados com 900 quadros, simulando um vídeo de 30 segundos a 30 quadros por segundo (FPS), além disso foi configurado um total de 4 *threads* de CPU, tempo de espera entre inferências de $0ms$ e foram realizados testes para todas os opções de modos de inferência (*Inference Mode*) disponíveis, sendo elas: INT8, FP16 e FP32. Porém, todas as execuções usando INT8 tiveram falhas em suas execuções e seus resultados foram descartados.

IV. Resultados

Para avaliação das redes treinadas foram utilizados dois *smartphones*: um *Redmi Note 8* e um *Samsung Galaxy S22*, ambos com suas descrições detalhadas na tabela I, que apresenta o modelo dos dispositivos, *chipset* equipado, modelo da GPU disponível, quantidade de memória RAM e o *score* do dispositivo no aplicativo *AI Benchmark*.

TABLE I
Tabela de especificação técnica dos *Smartphones* utilizados na avaliação das Redes Neurais

Modelo	Chipset	GPU	RAM	APP Score
Redmi Note 8	Qualcomm SDM665 Snapdragon 665 (11 nm)	Adreno 610	4GB	32.4
Samsung Galaxy S22	Qualcomm SM8450 Snapdragon 8 Gen 1 (4 nm)	Adreno 730	8GB	1239

A. Avaliação da Inferência

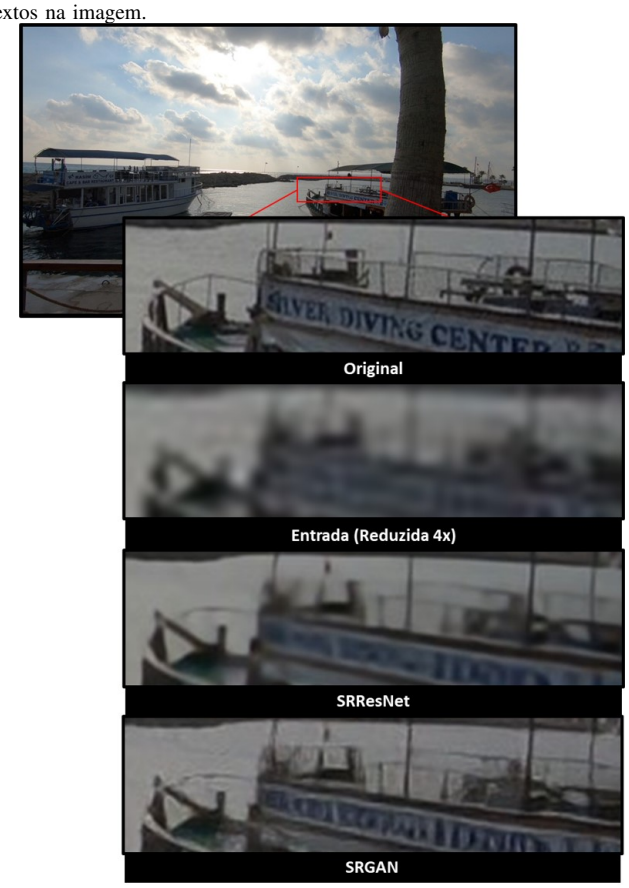
Após o treinamento, foi realizada a avaliação dos modelos utilizando as métricas descritas anteriormente. A rede SRResNet obteve um PSNR de ***27.7075*** e um SSIM de ***0.7881***, enquanto a rede SRGAN atingiu resultados inferiores nas duas medidas, com ***21.3843*** de PSNR e ***0.6636*** de SSIM.

Foi observado nos resultados que a qualidade de reconstrução de ambas as redes estão muito próximas quanto à percepção humana, embora detalhes e bordas estejam mais realçadas nas inferências da SRGAN quando comparada com a SRResNet, como observado na Fig. 5. Também foi observado que ambas as redes possuem dificuldades na reconstrução

[3]https://www.tensorflow.org/lite/performance/gpu
[4]https://developer.android.com/ndk/guides/neuralnetworks
[5]https://developer.qualcomm.com/software/hexagon-dsp-sdk
[6]https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk
[7]https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk
[8]https://ai-benchmark.net/index.php?threads/difference-between-the-htp-and-dsp-delegate.140/
[9]https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk
[10]https://neuropilot.mediatek.com/resources/public/latest/en/docs/readme
[11]https://developer.samsung.com/neural/overview.html

de faces em diversos quadros ao longo do *dataset*, como apresentado pela Fig. 6.

Fig. 5. Zoom do resultado das redes em uma amostra do *dataset* comparados com a imagem original e a imagem com escala reduzida em 4x, focando em textos na imagem.



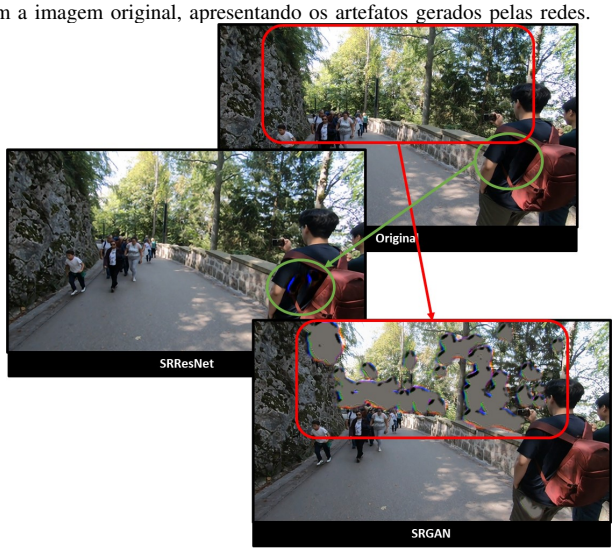
Também foram observados alguns artefatos nos resultados das imagens, em um número de ocorrências reduzida na SRResNet quando comparada com a SRGAN. Uma das cenas em que havia maior expectativa de bons resultados, por conta das texturas, era a filmagem 27 da amostra de validação (Fig. 7), devido o número de árvores e texturas de folhas da imagem. Porém, esta foi a que teve uma maior ocorrência de artefatos ao longo dos quadros, como pode ser visto na Fig. 7.

B. Avaliação da Latência

Uma vez disponibilizadas as redes no formato do *TensorFlow Lite* para os dispositivos, foi realizada a coleta dos resultados. Como mencionado anteriormente, todas as execuções com o *Inference Mode* em INT8 resultaram em erro, e não foram observadas diferenças significativas entre os modos FP16 e FP32. A maior diferença foi de fato entre os aceleradores, como apresentado na Fig. 8. Para o dispositivo Redmi Note 8, a melhor opção foi o uso da CPU diretamente, sem utilizar outros acelerados. Os piores desempenhos foram com o uso do acelerador Android NNAPI (que em ambas as redes levou em média de 12 a 13 milissegundos para realizar a inferência) e o Qualcomm QNN GPU (que, além de levar de 8 a 10 milissegundos, teve ainda o maior tempo de inicialização, chegando a aproximadamente 3 segundos e meio).

Fig. 6. Zoom do resultado das redes em uma amostra do *dataset* comparados com a imagem original e a imagem com escala reduzida em 4x, focando em faces.

Fig. 7. Zoom do resultado das redes em uma amostra do *dataset* comparados com a imagem original, apresentando os artefatos gerados pelas redes.



Quanto ao Samsung Galaxy S22, este obteve os menores tempos, porém os resultados observados no Redmi Note 8 se repetem com este dispositivo, com exceção do Acelerador extra, disponível apenas para este modelo, o Qualcomm QNN HTP, que teve o melhor desempenho de inferência, com uma redução de mais de 50% no tempo de inferência.

C. Medidas de Avaliação

Uma vez calculadas as métricas de PSNR e SSIM das redes, foi realizado o cálculo de pontuação utilizando o melhor tempo de cada rede, ambas utilizando os valores obtidos pelo

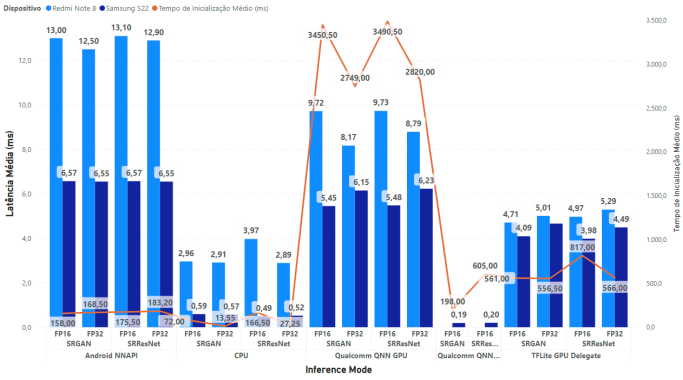


Fig. 8. Resultados obtidos das redes SRResNet e SRGAN a partir do *AI Benchmark* coletados e compilados.

Samsung Galaxy S22 combinado com o acelerador Qualcomm QNN HTP. A Tabela II apresenta as pontuações, e em uma comparação direta entre as duas redes, a rede SRResNet seria melhor classificada.

TABLE II
TABELA DE RESULTADOS DA AVALIAÇÃO DAS REDES SRRESNET E SRGAN

Rede	SSIM	PSNR	Latência	Pontuação
SRREsNet	0,7881	27,7075	0,19	95,71
SRGAN	0,6636	21,3843	0,2	85,20

V. CONCLUSÃO

O trabalho apresentado mostra que a utilização de técnicas de *Deep Learning* em *smartphones* para realização de aumento de escala dos vídeos é algo aplicável, sendo favorecido quando implementados em dispositivos de maior capacidade de processamento. Além disso, também é factível em dispositivos de menor capacidade, dado os tempos de latência apresentados. Algo não esperado foi na utilização de APIs de aceleração, que acabaram por aumentar a latência das respostas das redes, exceto pela Qualcomm QNN HTP, que apresentou uma redução nas latências. Também foi observado que a SRGAN apresentou melhores acabamentos em texturas se comparada com a SRResNet, mas a sua instabilidade e geração de artefatos impactava negativamente no resultado final da rede, colocando-a abaixo da SRResNet, tanto em tempo de resposta quando nas métricas.

Para trabalhos futuros, serão adicionados aspectos que não foram abordados neste trabalho, como, avaliação de outras arquiteturas (como a ELSR [10], rede campeã da edição 2022 da *Mobile AI & AIM Challenge*, assim como, outras arquiteturas como sugeridas por [4]). Além disso este trabalho não levou em consideração o consumo de energia das redes, outro fator que pode influenciar significativamente no embarcamento de redes neurais em dispositivos móveis.

Podemos ainda observar que ambos os dispositivos utilizados neste trabalho equipam processadores desenvolvidos pelo mesmo fabricante de processadores. Dessa forma, outra possibilidade futura de pesquisa é uma comparação entre o comportamento, otimização e tempos de execução de redes neurais em diferentes processadores de diferentes fabricantes, tais como, MediaTek, Samsung, Apple, Huawei e etc.

Dessa forma, podemos concluir que apesar de nova, a área de pesquisa de aplicação de SR em dispositivos móveis, a gama de possibilidades de novas linhas de pesquisa é grande. Além disso, diferentes arquiteturas de redes neurais podem ser aplicadas nos mais diferentes segmentos, tais como, *streaming*, jogos, entretenimento, fotográfico e etc.

REFERENCES

[1] H. Greenspan, "Super-resolution in medical imaging," *The computer journal*, vol. 52, no. 1, pp. 43–63, 2009.

[2] A. C. Bovik, *Handbook of image and video processing*. Academic press, 2010.

[3] Y. Zhang, Y. Zhang, Y. Wu, Y. Tao, K. Bian, P. Zhou, L. Song, and H. Tuo, "Improving quality of experience by adaptive video streaming with super-resolution," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pp. 1957–1966, IEEE, 2020.

[4] Z. Wang, J. Chen, and S. C. Hoi, "Deep learning for image super-resolution: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 10, pp. 3365–3387, 2020.

[5] L. Yue, H. Shen, J. Li, Q. Yuan, H. Zhang, and L. Zhang, "Image super-resolution: The techniques, applications, and future," *Signal processing*, vol. 128, pp. 389–408, 2016.

[6] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4681–4690, 2017.

[7] A. Ignatov, A. Romero, H. Kim, and R. Timofte, "Real-time video super-resolution on smartphones with deep learning, mobile ai 2021 challenge: Report," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2535–2544, 2021.

[8] A. Ignatov, L. Van Gool, and R. Timofte, "Replacing mobile camera isp with a single deep learning model," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 536–537, 2020.

[9] T. T. Dong, H. Yan, M. Parasar, and R. Krisch, "Rendersr: A lightweight super-resolution model for mobile gaming upscaling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3087–3095, 2022.

[10] T. Xu, Z. Jia, Y. Zhang, L. Bao, and H. Sun, "Elsr: Extreme low-power super resolution network for mobile devices," *arXiv preprint arXiv:2208.14600*, 2022.

[11] S. Nah, S. Baik, S. Hong, G. Moon, S. Son, R. Timofte, and K. Mu Lee, "Ntire 2019 challenge on video deblurring and super-resolution: Dataset and study," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019.

[12] S. Son, S. Lee, S. Nah, R. Timofte, and K. M. Lee, "Ntire 2021 challenge on video super-resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 166–181, 2021.

[13] S. Nah, R. Timofte, S. Gu, S. Baik, S. Hong, G. Moon, S. Son, and K. Mu Lee, "Ntire 2019 challenge on video super-resolution: Methods and results," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019.

[14] A. Ignatov, R. Timofte, C.-M. Chiang, H.-K. Kuo, Y.-S. Xu, M.-Y. Lee, A. Lu, C.-M. Cheng, C.-C. Chen, J.-Y. Yong, *et al.*, "Power efficient video super-resolution on mobile npus with deep learning, mobile ai & aim 2022 challenge: report," in *European Conference on Computer Vision*, pp. 130–152, Springer, 2022.