ANÁLISE EXPLORATÓRIA DE MODELOS DE REDES NEURAIS CONVOLUCIONAIS PARA CLASSIFICAÇÃO DE IMAGENS DE RAIO-X DE PULMÃO

João Felipe Barros Silva[1], Sílvio Roberto Fernandes de Araújo[2]
[1]Graduando – joao.silva27990@alunos.ufersa.edu.br
[2]Orientador – silvio@ufersa.edu.br

Resumo: A triagem eficaz de pacientes infectados pelo covid-19 desempenha um papel crucial no combate a essa doença, sendo uma das principais o exame de radiografia do tórax. No entanto, o diagnóstico pode ser confundido com o de pneumonia. Neste estudo, desenvolvemos quatro modelos utilizando Convolutional Neural Network (CNNs), treinadas com mais de 30.000 imagens. Esses modelos, são capazes de classificar uma imagem de raio-x de pulmão como normal, com pneumonia ou com covid-19, mantendo uma acurácia próxima de 90%. Além disso, realizamos uma análise de custo benefício dos modelos considerando a disponibilidade de recursos computacionais. Por fim, foi criado um aplicativo utilizando um dos modelos desenvolvidos para demonstrar a utilização clínica desta tecnologia.

Palavras-chave: Covid-19, Pneumonia, Redes neurais convolucionais, Radiografia, Acurácia.

1. INTRODUÇÃO

A covid-19 é uma infecção respiratória aguda causada pelo coronavírus SARS-coV-2. Os primeiros casos da doença foram registrados na cidade Wuhan, na província de Hubei, China. O vírus se espalhou rapidamente devido à sua natureza altamente contagiosa e à globalização dos meios de transporte, desencadeando uma crise de saúde global. Em março de 2020 a Organização Mundial da Saúde (OMS) declarou a mesma como uma pandemia global. Desde a descoberta do primeiro caso, na China, ocorreram mais de 700 milhões de infectados com cerca de 6,9 milhões de mortes em todo o mundo (OMS, 2023).

Com o advento das vacinas, houve uma significativa queda nas mortes pela covid-19, contudo, a mesma continua vitimando diariamente. Assim como em outros casos clínicos, o tratamento precoce é essencial para diminuir os índices de mortalidade. Atualmente, a Reação em Cadeia da Polimerase em Tempo Real (RT-PCR) é o método de diagnóstico padrão para esta doença. Entretanto, essa técnica tem apresentado altas taxas de falso negativos, além de seu processo ser invasivo e demorado, levando de 2-3 dias para produzir o resultado.

Pesquisas médicas descobriram que exames de imagem, como Tomografia Computadorizada (TC) e radiografia de tórax, podem ser utilizados para detectar pessoas infectadas com a covid-19. No entanto, as anormalidades causadas pela covid-19 percebidas nas imagens, são similares às encontradas em pessoas com pneumonia, tornando o diagnóstico mais complicado (Chen et al. 2020), (Guan et al. 2020).

Dessa forma, neste trabalho foram desenvolvidos quatro modelos baseados em aprendizado profundo de máquinas, que permitem classificar imagens de raio-x de pulmão em: covid-19, normal e pneumonia. Cada modelo foi desenvolvido com uma arquitetura pré-treinada diferente, utilizando *transfer learning* (transferência de aprendizagem), o que permite aproveitar o conhecimento genérico adquirido pelas arquiteturas treinadas em grandes conjuntos de dados. Essas arquiteturas são VGG16, VGG19, EfficientNetB7 e MobileNet, e o objetivo é adquirir modelos com boa acurácia que possam ser utilizados em sistemas com maior poder computacional do tipo cliente-servidor, quanto em sistemas mais restritos, como sistemas embarcados, utilizando a técnica de TinyML (Aprendizado de Máquina Minúsculo) (TinyML Foundation, 2019).

O artigo está organizado da seguinte maneira: A seção II apresenta a fundamentação teórica, a seção III apresenta um levantamento da literatura de trabalhos relacionados. A seção IV trata do trabalho proposto apresentando as metodologias de desenvolvimento. A seção V aborda os resultados obtidos do trabalho, assim como, uma comparação dos modelos propostos no trabalho. Finalmente, a Seção VI apresenta as conclusões e direções futuras de pesquisa.

2. FUNDAMENTAÇÃO TEÓRICA

Nesta seção será apresentada a fundamentação teórica necessária para o entendimento do trabalho

proposto. As informações são relativas a redes neurais convolucionais, transferência de aprendizado e tinyML.

Redes neurais artificiais são modelos matemáticos inspirados nos modelos biológicos, as quais são formadas por camadas de neurônios artificiais (Aggarwal, 2018). Cada neurônio artificial define uma função de transformação dos dados das suas entradas para gerar uma saída para outros neurônios. A comunicação entre os neurônios de camadas adjacentes também são análogas às sinapses dos modelos biológicos, no entanto são definidos como parâmetros para atribuição de maior ou menor peso (importância) para as informações. Assim, as redes neurais funcionam por meio de duas fases distintas: o treinamento e a inferência. Na primeira fase a rede neural é exposta a diversos exemplos de dados de entradas e os valores de saídas esperados, de modo que a proporção da diferença entre as saídas geradas e as esperadas possam ser usada para ajustes nos pesos de cada neurônio. O objetivo dessa fase é encontrar padrões nas informações do conjunto de dados (dataset). Após várias iterações (épocas) de treinamento, e monitorando métricas de qualidade das respostas da rede, o modelo pode ser usado para fazer inferências. Isto é, realizar classificações ou predições para dados inéditos sem alterar os parâmetros internos da rede.

Ao longo do tempo surgiram diversas ideias de construção de redes neurais, e uma inspirada nos processos de convoluções são as Redes Neurais Convolucionais (Convolutional Neural Network - CNNs), as quais representam uma parte especializada de redes neurais profundas, projetadas para tarefas de reconhecimento de padrões em dados de alta dimensionalidade, como imagens e vídeos. A estrutura de uma CNN é composta por camadas convolucionais e camadas de pooling totalmente conectadas. As camadas convolucionais aplicam filtros sobre os dados para extrair características relevantes, enquanto as camadas de pooling reduzem a dimensionalidade dos dados, garantindo as características mais importantes (LeCun et al. 2019).

O *transfer learning* é uma técnica que utiliza o conhecimento adquirido em uma tarefa específica, para melhorar o desempenho em outras tarefas relacionadas. Dessa forma, o principal objetivo dessa técnica é mitigar a escassez de dados e acelerar o processo de treinamento em novos domínios, aproveitando conhecimento prévio (TensorFlow, 2024c). No contexto de CNNs, o *transfer learning* é comumente aplicado ao reutilizar os pesos de redes neurais treinadas em grande conjunto de dados como, o ImageNet (Deng et al., 2009), e adaptá-los para um domínio específico com conjunto de dados menores. Dessa forma, uma arquitetura de rede neural treinada para um grande dataset como o ImageNet, tem suas primeiras camadas especializadas em encontrar padrões genéricos enquanto as últimas, utilizando os padrões identificados pelas camadas anteriores, encontram padrões específicos de cada classe. Logo, a técnica de transfer learning consiste em “congelar” as primeiras camadas e substituir as últimas camadas por outras relativas a outras classes. Isso permite reduzir o tempo de treinamento e aumentar a acurácia dos novos modelos.

É notório como as técnicas de aprendizado de máquina estão presentes em praticamente qualquer domínio, e essa evolução se deu principalmente pelo desenvolvimento de hardwares com grande poder computacional como as GPUs (Graphics Processing Unit) e processadores com vários núcleos e grande quantidade de memória. Para o treinamento dos modelos ainda é exigido tamanho poder computacional. Contudo, a partir de 2012 surgiu a ideia de realizar inferência em dispositivos com muitas restrições computacionais, sobretudo energia, conhecidos como TinyML (Warden; Situnayake, 2019). TinyML aborda a implementação de modelos de Machine Learning (Aprendizado de Máquina) em dispositivos com recursos computacionais limitados, como microcontroladores e sistemas embarcados de baixa potência. Devido às restrições de memória, processamento e energia dos dispositivos, o TinyML geralmente é otimizado para operar com eficiência em dispositivos de baixa potência. Isso é alcançado utilizando técnicas como quantização e *pruning* de redes neurais (TinyML Foundation, 2019). As técnicas de otimização mencionadas vão ser abordadas na seção 4.

3. TRABALHOS RELACIONADOS

Ao longo da pandemia, várias abordagens utilizando aprendizado profundo de máquina foram desenvolvidas para predição de covid-19. Em (Mohammadi et al., 2020), foram propostos quatro modelos utilizando Convolutional Neural Network (CNN) pré-treinadas, VGG16, VGG19, MobileNet e InceptionResNetV2, com apenas 545 imagens de raio-x para treino teste e validação, obtendo assim, os seguintes resultados de acurácia, 93.6%, 90.8%, 99.1%, 96.8%, respectivamente. Para classificação entre covid-19 ou Pneumonia.

O trabalho de Yang et al. (2021) também utilizou quatro modelos de CNNs pré-treinadas, VGG16, DenseNet121, ResNet50 e ResNet152. Neste caso, foi utilizado um conjunto de dados próprio, obtido da junção de outros dois conjuntos de dados públicos, alcançando assim 8.461 imagens, separadas em três classes: covid-19, normal e pneumonia. No mesmo trabalho, foram feitas três abordagens de classificação diferentes. Primeiramente, foi utilizada uma predição binária, ou seja, a CNN classifica em duas classes, covid-19 ou Pneumonia. Na segunda abordagem, também binária, em covid-19 ou Normal, e, por último, a predição em múltiplas classes: covid-19, normal ou pneumonia. Em todas as abordagens a precisão passou dos 90%.

Hernandez et al. (2020) desenvolveram quatro modelos, um deles uma arquitetura customizada e os outros utilizando as redes pré-treinadas ResNet50, VGG16 e DenseNet121. Todos os modelos utilizaram o mesmo conjunto de dados, contendo mais de 30.000 imagens, para treino, validação e teste, e obtiveram as

seguintes acurácias: 68%, 90%, 82% e 83%, respectivamente.

Outros trabalhos focaram especificamente em construir modelos próprios para a classificação das imagens de tórax. Em (Ozturk et al. 2020), foi proposto um modelo de aprendizado profundo chamado DarkCovidNet, inspirado na arquitetura DarkNet-19. O modelo proposto no trabalho possui 17 camadas convolucionais, assim como, várias filtragens em cada camada, sendo capaz de executar tarefas binárias e de múltiplas classes com acurácia de 98.8% e 87.02% respectivamente.

Wang et al. (2020) apresentaram o Covid-Net, um modelo de CNN para a detecção de covid-19, de código aberto, composto por 13.975 imagens, que obteve 92.2% de acurácia na classificação das classes normal, pneumonia e covid-19.

Ao analisar os trabalhos relacionados, foi identificado que a maioria deles apresenta apenas a métrica de acurácia para avaliação da qualidade dos seus modelos. Aqueles que apresentam maiores taxas de acurácia utilizam conjunto de dados relativamente pequenos ou para classificação binárias (COVID/Normal ou Pneumonia/Normal). Além disso, tais trabalhos não apresentam quaisquer restrições de inferência ao ambiente computacional onde tais soluções poderiam ser empregadas. Portanto, a contribuição do trabalho proposto diz respeito ao preenchimento das lacunas identificadas nos trabalhos relacionados.

4. TRABALHO PROPOSTO

Neste trabalho, foram desenvolvidos quatro modelos para a classificação de imagens de radiografia do tórax. Para isso, utilizamos o *framework* TensorFlow (TensorFlow, 2024d) e quatro arquiteturas de aprendizado profundo disponíveis na API Keras (Oneiros, 2015). O TensorFlow tem a vantagem de também disponibilizar as bibliotecas TFLite e TFLite Micro, as quais foram especialmente projetadas para atender a dispositivos móveis e embarcados, como smartphones, tablets, dispositivos IoT (Internet das Coisas) e microcontroladores. Esses modelos implementam técnicas como a quantização que permite representar os pesos do modelo com uma menor quantidade de bits, o que resulta em uma significativa redução no tamanho de memória requerida para armazenar o modelo, conforme a documentação oficial TensorFlow (2024a). Além disso, outra técnica utilizada é o *pruning*, que envolve a poda de conexões redundantes e menos importantes na rede neural, contribui para uma redução adicional no tamanho do modelo, sem comprometer significativamente a acurácia, segundo a documentação oficial TensorFlow (2024b). A utilização conjunta dessas técnicas contribuem para que esses modelos sejam ideais para dispositivos com baixo poder computacional, mantendo o equilíbrio entre tamanho e desempenho.

No entanto, para criação e treino dos modelos, é necessário usar o TensorFlow padrão. Assim, os quatro modelos desenvolvidos neste trabalho fizeram uso dessa biblioteca, os quais também utilizaram os mesmos parâmetros para treino. Primeiramente, as camadas das arquiteturas escolhidas foram congeladas evitando a alteração dos seus pesos e garantindo a transferência de aprendizado. Em seguida, adicionamos três camadas treináveis para compor a rede, uma camada "Flatten", responsável por converter o tensor em um vetor unidimensional, em seguida, uma camada densa com 64 neurônios e a função de ativação "Relu", logo após, incluímos o Dropout com taxa de 0,2 para evitar overfitting e, por último, adicionamos uma camada densa para classificação com três neurônios, correspondente às três classes.

Para as configurações específicas de treinamento dos modelos, utilizamos o algoritmo Adam para ajustar os pesos, "SparseCategoricalCrossentropy" para calcular função de perda e a métrica "Accuracy" para avaliar o desempenho do modelo e treinamos por até 30 épocas. Além disso, também utilizamos a técnica de "Earlystopping" com paciência de dez, monitorando o valor de perda. Ou seja, cada modelo pode ser treinado até 30 épocas, mas se, por 10 épocas o valor da perda não diminuir, o treinamento é encerrado. Isso evita o *overfitting* e interrompe os treinamentos quando a rede não obtém melhorias. A máquina utilizada para todos os testes possui um processador Rayzen 7 de 3.7 GHz de 8 núcleos e 16 GB de memória RAM. Utilizamos ambiente online Edge Impulse (Impulse, 2024) para modelagem e implantação de TinyML. A seguir são apresentados detalhes de cada uma das arquiteturas utilizadas.

4.1. Arquitetura VGG16

A arquitetura VGG16 emprega 16 camadas de convolução 3x3 e pooling 2x2, resultando em modelo que ocupa 528 megabytes, de acordo com Oneiros (2015). Utilizando essa arquitetura, obtivemos 16.320.891 milhões de parâmetros, dos quais 1.605.891 são treináveis. Para treinar esse modelo foram necessárias 15 épocas, com cada época durando em média 3.410 segundos, o que acarretou em cerca de 14 horas.

4.2. Arquitetura VGG19

O VGG19 é uma extensão do VGG16, com mais camadas de convolução mantendo a mesma estrutura básica. Ele emprega 19 camadas, possuindo um tamanho de 549 megabytes em disco, segundo Oneiros (2015). O treinamento com essa arquitetura exigiu 7 épocas para ser concluído. Cada época teve uma duração média de 4.387 segundos totalizando aproximadamente 8 horas para o treinamento completo. Durante esse processo,

foram utilizados 21.630.275 milhões de parâmetros, sendo que 1.605.891 eram treináveis.

4.3. Arquitetura MobilNet

O MobileNet, uma arquitetura indicada para aplicações em dispositivos móveis, consiste em 55 camadas e possui um tamanho de aproximadamente 16 megabytes (Oneiros, 2015). Por ser um arquitetura mais compacta, adequada para uso em dispositivos embarcados, o tempo de treinamento foi de aproximadamente duas horas, com um total de 14 épocas. Cada época teve, em média, uma duração de 602 segundos. Possuindo 6.272.323 milhões de parâmetros com 4.014.339 treináveis

4.4 Arquitetura EfficientNetB7

Por último, o EfficientNetB7, que utiliza 438 camadas, possuindo um tamanho de 256 megabytes (Oneiros, 2015). Esse modelo possui 72.126.106 milhões de parâmetros, sendo que 8.028.419 são treináveis, o que acarretou no maior tempo de treino dentre os modelos propostos com cerca de 28 horas em 16 épocas, ou seja, 6.208 segundos por época.

4.5 Comparativo dos modelos desenvolvidos

Na Tabela 1 é apresentado um comparativo das características das dimensões dos quatro modelos desenvolvidos, a partir do treinamento deles. A coluna "Tamanho" indica a quantidade de memória (em MBbytes) necessário para executar cada arquitetura treinado, segundo a documentação oficial Keras Documentation (2023), e está diretamente relacionado com o número de parâmetros e o número de camadas (segunda e terceira colunas). Na quarta coluna são apresentadas a quantidade de épocas para treinar cada modelo, uma vez que o máximo seria 30 épocas utilizando a *earlystopping* com paciência de 10 épocas. A última coluna apresenta o tempo, em horas, para o treinamento dos modelos pré-treinados (usando *transfer learning* com base em dados do ImageNet (Deng et al., 2009)), que não tem uma relação de dependência exclusiva da quantidade de épocas, mas também da quantidade de parâmetros e de camadas.

Tabela 1. Comparativo entre os modelos propostos

Arquitetura	**Tamanho (MB)**	**Nº de Parâmetros**	**N° de camadas**	**N° de Épocas (*early stopping*)**	**Tempo para treino (h)**
VGG16	528	16.320.821	20	15	14
VGG19	549	21.630.275	23	7	8
MobileNet	16	6.272.323	59	14	2
EfficientNetB7	256	72.126.106	442	16	28

5. RESULTADOS

Nesta seção, apresentaremos os resultados obtidos nesta pesquisa, que vão desde o desenvolvimento de um conjunto de dados a partir de outros, até os resultados de qualidade dos modelos e otimização para sistemas embarcados. Os resultados apresentados na seção 5.3 foram obtidos através do TensorFlow completo, ou seja, sem a utilização de otimizações específicas. Com a conversão dos modelos padrões para TFlite, foram utilizadas técnicas de otimização para redução do tamanho dos modelos e latência durante a inferência, cujos resultados são apresentados na seção 5.4. Ao final desta seção, apresentamos um aplicativo desenvolvido a partir de um dos modelos.

5.1. Conjunto de Dados

Para a realização deste trabalho, foi desenvolvido um conjunto de dados a partir de três conjuntos de dados públicos para treinar, validar e testar os modelos propostos. O primeiro conjunto de dados utilizado foi RSNA Pneumonia Detection Challenge (Radiological Society of North America, 2020), disponibilizado pela plataforma de competição Kaggle. Ele consiste em 29.687 imagens de radiografias torácicas, no formato DICOM, divididas em duas classes, normal e pneumonia. O segundo conjunto de dados foi o COVIDx

CXR-3-classification (Hossein et al., 2022), também adquirido na plataforma Kaggle, tendo 29.986 imagens no formato PNG, classificadas em duas classes: positivo e negativo para a covid-19. O último conjunto de dados utilizado foi o Chest X-Ray Images (Kermany et al., 2018), contendo 5.857 imagens no formato JPEG, separadas em três classes: normal e pneumonia.

O conjunto de dados desenvolvido consiste em 30.600 imagens divididas em três partes: treino (80%), teste (10%) e validação (10%). É importante destacar que todas as classes foram balanceadas para que possuíssem a mesma quantidade de imagens em cada uma das partes, a fim de evitar viés ou desequilíbrio, conforme mostra a Tabela 2.

Tabela 2. Distribuição do conjunto de dados obtido

Categoria	Dados de Treinamento	Dados de Validação	Dados de Teste
covid-19	8160	1020	1020
Normal	8160	1020	1020
Pneumonia	8160	1020	1020

5.2. Pré-processamento de dados

Para lidar com a falta de uniformidade dos dados, redimensionamos todas as imagens para a resolução 224 x 224 e o formato PNG com 3 canais de cores RGB. Essa abordagem é comumente utilizada para padronizar as dimensões e formato das imagens, tornando-as adequadas para a alimentação em modelos de aprendizado profundo.

5.3. Análise da Qualidade dos modelos

Os resultados obtidos nesta pesquisa demonstram o desempenho dos quatro modelos de classificação de imagens de radiografia torácica desenvolvidos. Esses modelos foram treinados utilizando o conjunto de dados descrito na Seção 5.1 e os parâmetros abordados na Seção 4. A avaliação do desempenho dos modelos foi realizada a partir da acurácia, precisão, recall e f1-score para as três classes: covid-19, normal e pneumonia. Além disso, a matriz de confusão foi utilizada para fornecer mais detalhamento do desempenho dos modelos em relação às classes previstas e reais.

As Figuras 1, 2 e 3 mostram os gráficos das métricas de desempenho, precisão, recall e f1-score, das classes: covid-19, normal e pneumonia, para cada um dos modelos propostos. Observa-se que em todos os modelos, a classe covid-19 obtém resultados acima dos 90% e com baixa variação em todas as métricas. Isso indica um desempenho consistente e preciso na detecção de casos dessa doença, minimizando os erros de predição para essa classe específica. No entanto, para as outras duas classes normal e pneumonia, constata-se a ocorrência de mais oscilações nas métricas.

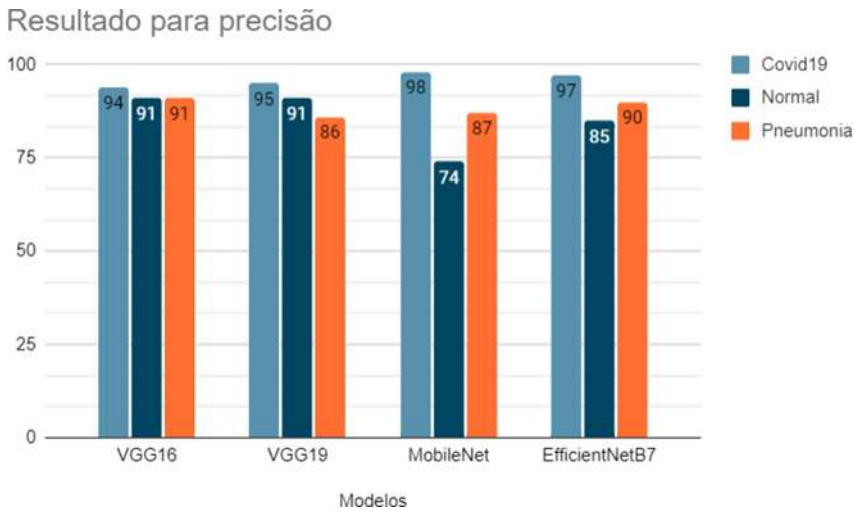


Figura 1. Precisão dos 4 modelos para cada classe. (Autoria própria)

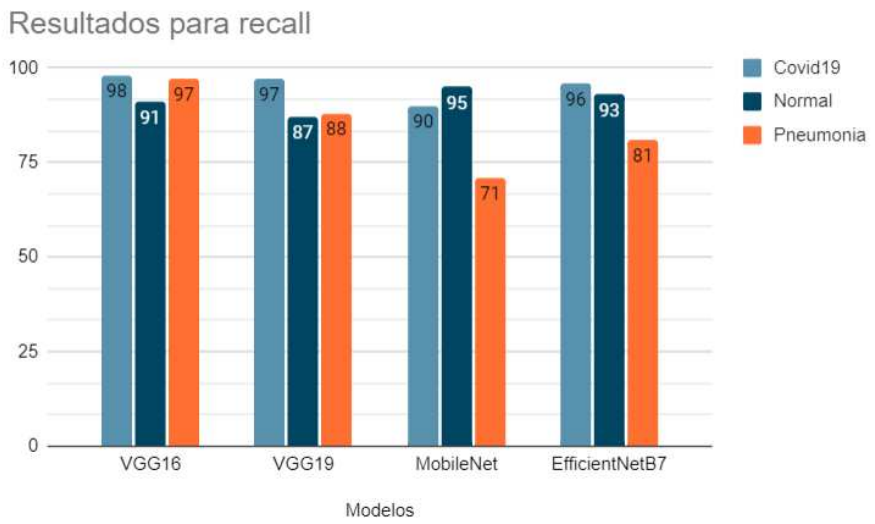


Figura 2. Recall dos 4 modelos para cada classe. (Autoria própria)

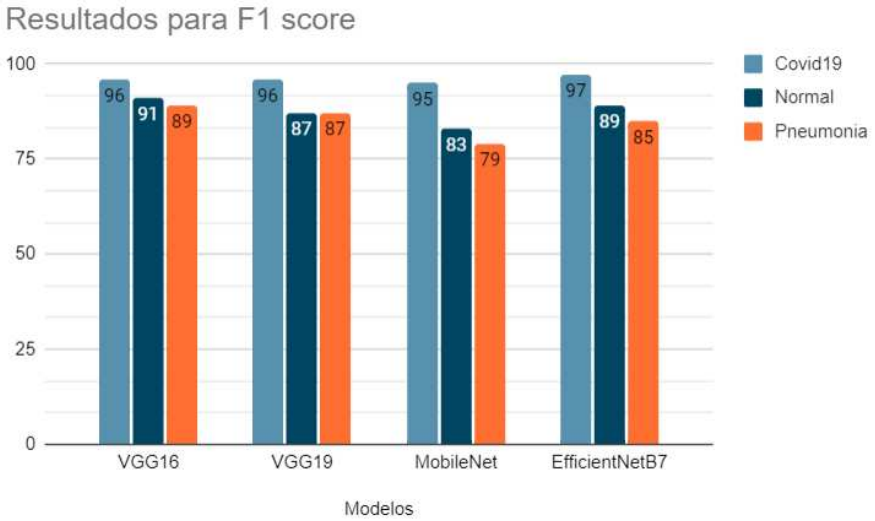


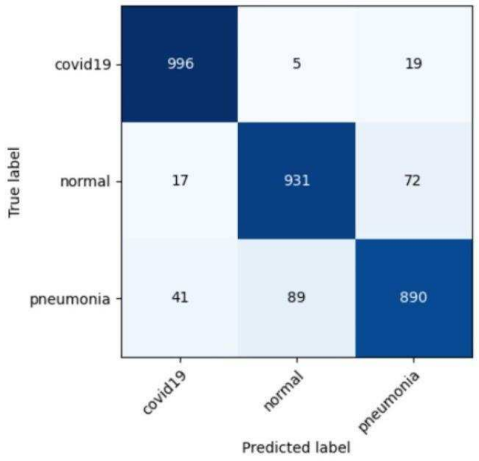
Figura 3. F1 score dos 4 modelos para cada classe. (Autoria própria)

A Figura 4 apresenta as matrizes de confusão, que permitem visualizar de forma mais detalhada os acertos e erros de classificação para cada classe nos modelos propostos. Os resultados demonstrados nas matrizes corroboram com as métricas anteriormente mencionadas. Observa-se que em todos os modelos, a classe covid-19 apresenta poucas confusões, o que indica um alto nível de acerto nas previsões desses casos, em acordo com os resultados das métricas de desempenho acima.

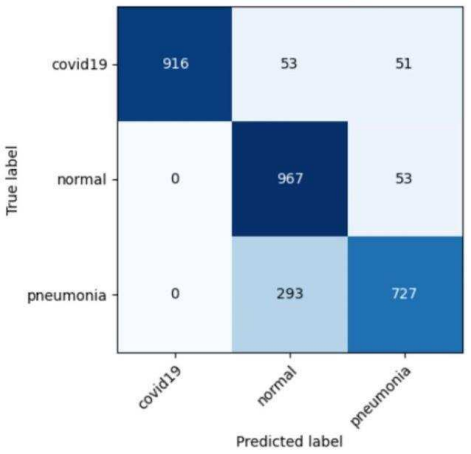
Por outro lado, é notável que a classe pneumonia apresenta bastante confusão com a classe normal em todos os modelos, especialmente no modelo que utiliza MobileNet. Indicando que os modelos apresentam mais dificuldades em discriminar pneumonia de casos normais.

A classe normal mantém uma situação intermediária, entre a classe covid-19 e a classe pneumonia em termos de erros de classificação, exceto no modelo MobileNet onde a mesma apresenta uma taxa de acertos maior que a classe covid-19.

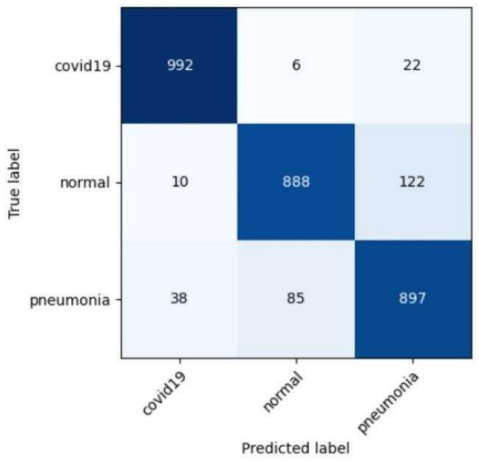
a) VGG16



c) MobileNet



b) VGG19



d) EfficientNetB7

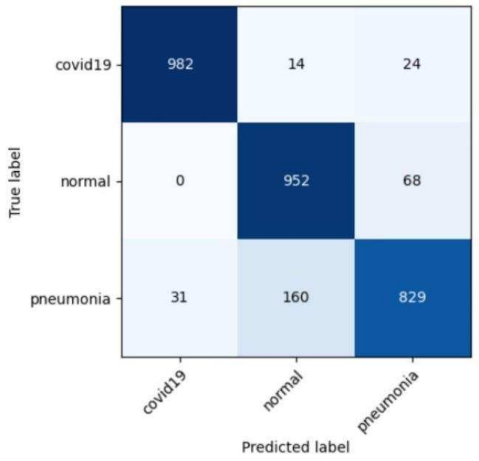


Figura 4. Matriz de confusão para os dados de teste, a) Modelo com VGG16; b) Modelo com VGG19; c) Modelo com MobileNet; d) Modelo com EfficientNetB7. (Autoria própria)

Além das métricas já destacadas é importante considerar outra medida de desempenho, a acurácia, que fornece uma média geral de acertos do modelo, sendo relevante para avaliar a eficiência do mesmo. A Figura 5, exibe os resultados de acurácia de cada modelo testado. Onde é possível observar que o modelo que utiliza o VGG16 alcançou melhor acurácia em comparação com os outros modelos e MobileNet a menor acurácia. Contudo, nenhum modelo teve acurácia abaixo de 85%, indicando uma validação da metodologia de treinamento/teste e qualidade da solução independente do modelo.

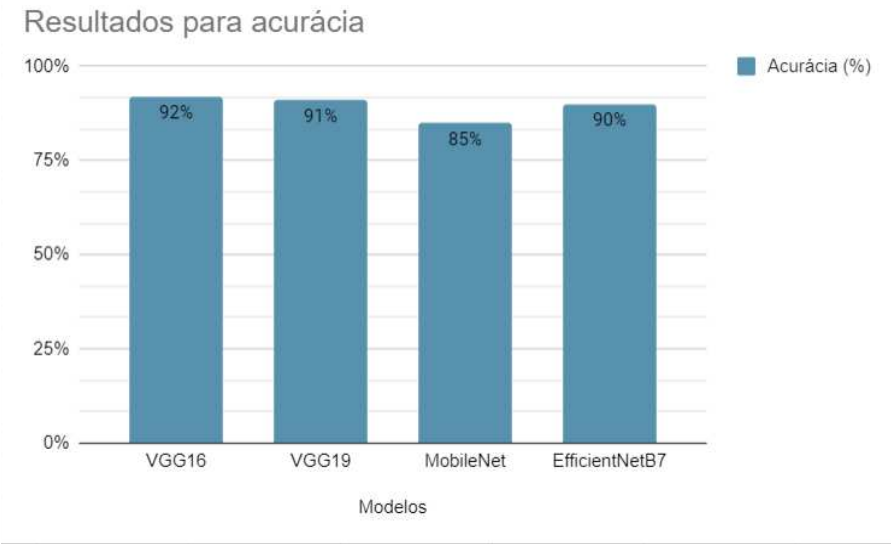


Figura 5. Acurácia média dos quatro modelos. (Autoria própria)

5.4. Recursos computacionais necessários para cada modelo

Considerando outros aspectos é possível estabelecer uma relação de custo-benefício para determinar a eficiência e viabilidade dos modelos em cenários práticos, uma vez que, o cenário de aplicação tem necessidades específicas considerando os recursos computacionais disponíveis e as restrições de tempo. Para tanto, listamos na Tabela 3 informações acerca do tamanho em megabytes do modelo que nós treinamos usando TensorFlow padrão, o tamanho dos modelos quantizados (Tensor Flow Lite), acurácia e tempo médio de predição/inferência para cada modelo, com desvio padrão 0,01902 para o VGG16, 0.01877 para o VGG19, 0.01871 para o MobileNet e 0.01775 para o EfficientNetB7.

Tabela 3. Custo-benefício para os modelos

Arquitetura	**Tamanho do modelo padrão (MB)**	**Tamanho do modelo quantizado (MB)**	**Acurácia modelo padrão (%)**	**Acurácia modelo quantizado (%)**	**Latência para predição no PC (ms)**	**Latência para predição no Arduino (ms)**
VGG16	76,3	63,7	92	90	123	1297
VGG19	97,1	80,5	91	88	155	1511
MobileNet	56,3	23,8	85	88	20	199
EfficientNet B7	68	52,9	90	91	223	2259

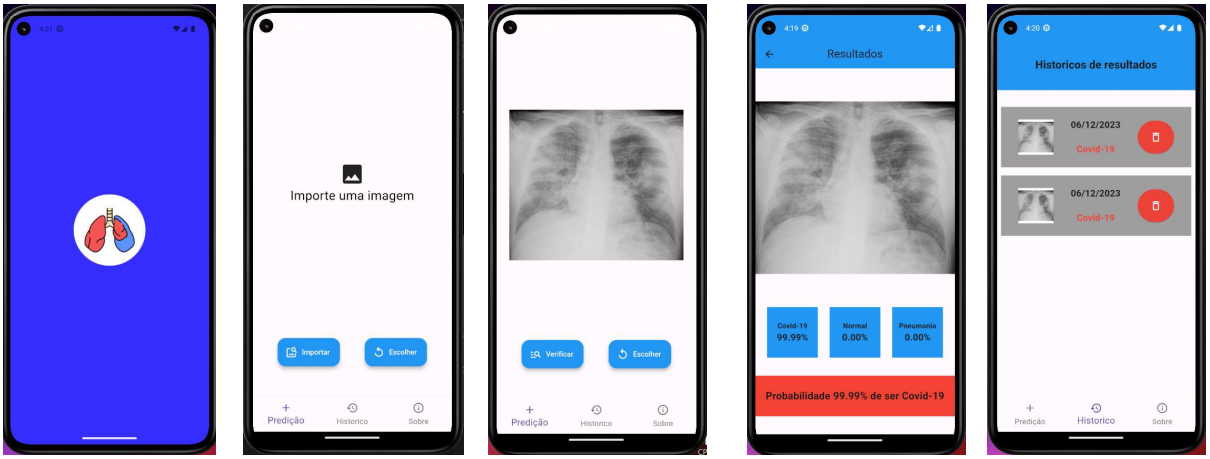
Com base na análise da tabela percebe-se que o MobileNet se destaca pelo tamanho compacto e baixíssima latência. Além disso, este modelo apresenta uma acurácia razoável e tempo de latência/predição significativamente mais rápido do que o modelo com melhor acurácia (VGG16). Essa combinação torna o MobileNet mais adequado para sistemas que possuem recursos computacionais limitados, como sistemas embarcados. A importância de ter um tempo de inferência mais baixo possível pode ser determinante para o desenvolvimento de um sistema embarcado. Contudo, é importante destacar que, mesmo possuindo o menor tamanho, tanto para o TensorFlow completo quanto para o TensorFlow lite, ainda não é possível utilizar esse modelo no Arduino nano 33 BLE, uma vez que excedem a sua capacidade máxima de armazenamento. O tempo de predição exibido na última coluna da tabela é uma estimativa fornecida pela plataforma Edge Impulse.

Considerando a presença de recursos robustos com alto poder de processamento, é possível priorizar o desempenho máximo utilizando o modelo com VGG16 que oferece a melhor acurácia entre os modelos testados, porém, é importante considerar que ele possui o segundo maior tamanho em MB entre todos os modelos. Também é importante destacar que as medidas de tempo apresentadas aqui, foram úteis para efeito de comparação entre os modelos, de modo que em um ambiente mais restrito, como sistemas embarcados, o tempo de inferência deve ser maior e em um ambiente computacional mais robusto poderia reduzir todos esses tempos como é mostrado na tabela 3, onde o tempo de latência/inferência, quinta coluna, em um computador pessoal (PC) é cerca de dez vezes menor que no Arduino 33 BLE Sense (sexta coluna).

Ainda nessa perspectiva é possível notar que o modelo que faz uso do EfficientNetB7 possui acurácia muito próxima do VGG16, ao mesmo tempo em que apresenta um tempo de latência bem maior. Por último temos o VGG19 que possui uma acurácia de 91%, com o maior tamanho e segundo maior tempo de predição em comparação com os quatros modelos, nesse sentido comparado esse modelo aos outros ele seria inviável para uma arquitetura com baixo poder computacional e também não oferece um bom custo benefício para uma arquitetura mais robustas, uma vez que, o VGG16 obteve melhores resultados utilizando menos recursos.

5.5. Software desenvolvido

Após a análise dos resultados dos modelos, foi desenvolvido um aplicativo móvel utilizando o modelo VGG16, que apresentou o melhor resultado médio em qualidade para classificação de imagens de raio-x de pulmão para as três categorias já mencionadas: normal, com pneumonia ou com COVID-19. O mesmo foi devidamente registrado no Instituto Nacional da Propriedade Industrial (INPI) sob o número BR512023003717-6 (Silva; Oliveira; Araújo, 2023). Este aplicativo foi desenvolvido como um sistema cliente-servidor, em que o servidor, desenvolvido em Flask/Python, é responsável por todo o processamento das imagens. Os resultados são então enviados para o frontend, feito em Flutter/Dart, como ilustrado na Figura 6 a seguir, que apresenta a interface do aplicativo e como ele exibe os resultados de classificação. Na Figura 6.a, observa-se a tela de entrada do aplicativo, exibida brevemente durante o carregamento inicial. Posteriormente, na Figura 6.b, os usuários são direcionados para a tela inicial, onde podem iniciar o processo de classificação das imagens, selecionando uma imagem de raio-x. Em seguida, na Figura 6.c, os usuários têm a opção de escolher uma nova imagem de raio-x ou verificar a imagem previamente selecionada. Logo após, na Figura 6.d, os resultados da classificação são apresentados. Nesta tela, os profissionais de saúde podem consultar a classificação da imagem como normal, com pneumonia ou com COVID-19. Por fim, a Figura 6.e exibe a tela de histórico, mostrando um registro das imagens previamente classificadas. Isso permite aos usuários revisar os resultados anteriores e acompanhar o histórico de classificações realizadas. Este aplicativo representa uma ferramenta útil para auxiliar profissionais de saúde na interpretação rápida e precisa de imagens de raio-x de pulmão, contribuindo para um diagnóstico mais eficiente e precoce das condições respiratórias em questão.



a) Splash screen b) Tela Inicial c) Imagem escolhida d) Resultados e) Histórico

Figura 6. Telas do aplicativo desenvolvido utilizando o modelo com VGG 16. (Autoria própria)

6. CONCLUSÕES

Nessa pesquisa, foram apresentados quatro modelos CNN pré-treinados, VGG16, VGG19, MobileNet, EfficientNetB7, para classificação de imagens de raio-x de pulmão em três categorias: normal, com pneumonia ou com covid-19, utilizando um conjunto de dados de 30.600 imagens obtidos da junção de três conjunto de dados públicos. Além disso, foi desenvolvido um aplicativo móvel para demonstrar a aplicabilidade prática desses modelos. O aplicativo utiliza o modelo VGG16 para classificar imagens de raio-x de pulmão em tempo real.

Os resultados mostraram que os modelos propostos foram capazes de realizar a classificação com acurácia próxima dos 90%. Entre os quatro modelos, o VGG16 obteve o melhor desempenho, alcançando uma taxa de acurácia de 92%. No entanto, esse modelo também apresentou o segundo maior tamanho em MB além de possuir o segundo menor tempo de inferência em comparação aos outros modelos testados. Para utilização em uma arquitetura cliente-servidor, onde não há restrição severa de recursos, o modelo VGG16 seria o mais indicado devido ao seu melhor desempenho, de acurácia.

No entanto, para sistemas com recursos limitados (sistemas embarcados), o modelo MobileNet, embora tenha a menor taxa de acurácia entre os modelos propostos com 85%, pode ser uma opção mais adequada, devido à sua menor quantidade de parâmetros e consequentemente menor quantidade de memória requerida, além do menor tempo de inferência.

Para direções futuras, a detecção de covid-19 em conjunto com pneumonia surge como uma direção promissora, uma vez que, a pneumonia, uma doença oportunista, frequentemente é observada em pacientes com baixa imunidade resultante da covid-19.

7. REFERÊNCIAS BIBLIOGRÁFICAS

AGGARWAL, Charu C. Neural Networks and Deep Learning: A Textbook. 1. ed. Springer, 2018.

ARDUINO. Especificação do produto nRF52840 v.1.1. Junho de 2023. 1 documento online. Disponível em: https://content.arduino.cc/assets/Nano_BLE_MCU-nRF52840_PS_v1.1.pdf?_gl=116psgzk_gaMTMzOTczNzM5Ny4xNjkyMDQ3NTU0_ga_NEXN8H46L5*MTY5MjA0NzU1My4xLjEuMTY5MjA0ODAwMC4wLjAuMA. Acesso em: 06 de ago. 2023.

CHEN, N., ZHOU, M., DONG, X., et al. Epidemiological and clinical characteristics of 99 cases of 2019 novel coronavirus pneumonia in Wuhan, China: a descriptive study. Lancet. 2020; 395(10223):507-13. Disponível em: https://doi.org/10.1016/S0140-6736(20)30211-7

DENG, Jia; DONG, Wei; SOCHER, Richard et al. ImageNet: A large-scale hierarchical image database. In: ieee conference on computer vision and pattern recognition, 2009. Proceedings... Piscataway, NJ: IEEE, 2009. p. 248-255.

GUAN, W. J., NI, Z. Y., LIANG, W. H., et al. Clinical Characteristics of Coronavirus Disease 2019 in China. N Engl J Med. 2020; 382(18):1708-20. Disponivel em: https://doi.org/10.1016/j.jemermed.2020.04.004

IMPULSE, E. Disponível em: https://edgeimpulse.com/. Acesso em: 10 abr. 2024.

HERNANDEZ, D.; PEREIRA, R.; GEORGEVIA, P. COVID-19 detection through X-Ray chest images. In: International Conference Automatics and Informatics (ICAI). 2020.

.

HOSSEIN, A.; WONG, A.; GUNRAJ, H.; et al. COVIDx CXR-3. Kaggle. Fevereiro de 2022. Disponível em: https://www.kaggle.com/datasets/andyczhao/covidx-cxr2. Acesso em: 20 de abr. 2023.

Keras Documentation. Keras API Applications. 2023. Disponível em: https://keras.io/api/applications/. Acesso em: 09 abr. 2024.

KERMANY, Daniel; ZHANG, Kang; GOLDBAUM, Michael. Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification. Mendeley Data. 2018. v. 2. doi: 10.17632/rscbjbr9sj.2. Disponível em: https://data.mendeley.com/datasets/rscbjbr9sj/2. Acesso em: 20 de abr. 2023.

LECUN, Yann; BOTTOU, Léon; BENGIO, Yoshua; HAFFNER, Patrick. Aprendizado por gradiente aplicado ao reconhecimento de documentos. In: Proceedings of the IEEE. v. 86, n. 11, p. 2278-2324, 1998.

MOHAMMADI, R.; SALEHI, M.; GHAFFARI, H.; et al. Transfer Learning-Based Automatic Detection of Coronavirus Disease 2019 (COVID-19) from Chest X-ray Images. Journal of Biomedical Physics and Engineering. 2020, vol. 10, no. 5, p. 559-568.

ONEIROS. Keras API Reference – Applications. 2015. Disponível em: https://keras.io/api/. Acesso em: 20 de jun. 2023.

ORGANIZAÇÃO MUNDIAL DA SAÚDE. COVID-19 Weekly Epidemiological Update. Junho de 2023. Disponível em: https://www.who.int/publications/m/item/weekly-epidemiological-update-on-covid-19---8-june-2023. Acesso em: 22 de jun. 2023.

ÖZTÜRK, T.; TALO, M.; YILDIRIM, E.; BALOĞLU, U.; YILDIRIM, O. et al. Automated detection of COVID-19 cases using deep neural networks with X-ray images. Computers in Biology and Medicine. 2020, vol. 121, p. 103792. Disponível em: https://doi.org/10.1016/j.compbiomed.2020.103792.

RADIOLOGICAL SOCIETY OF NORTH AMERICA (RSNA). RSNA Pneumonia Detection Challenge. 2018. Disponível em: https://www.kaggle.com/competitions/rsna-pneumonia-detection-challenge/data. Acesso em: 20 de abr. 2023.

SILVA, J. F. B; OLIVEIRA, L. C.; ARAÚJO, S. R. F.. Detector de COVID-19 em Raio-X de Pulmão. Versão 1.0. Mossoró: Universidade Federal Rural do Semi-Árido, 2023. Software registrado no Instituto Nacional da Propriedade Industrial (INPI) sob o número BR512023003717-6, de 12 de dezembro de 2023.

TANGUDU, V. S.; KAKARLA, J.; VENKATESWARLU, I. B. COVID-19 detection from chest x-ray using MobileNet and residual separable convolution block. Soft Computing. 2022, vol. 26, no. 5, p. 2197-2208. DOI 10.1007/s00500-021-06579-3.

TensorFlow. Quantização pós-treinamento. Disponível em: https://www.tensorflow.org/lite/performance/post_training_quantization?hl=pt-br. Acesso em: 09 de abr. 2024a

TensorFlow. Remover pesos insignificantes. Disponível em: https://www.tensorflow.org/model_optimization/guide/pruning?hl=pt-br. Acesso em: 09 de abr. 2024b

TensorFlow. Transfer learning and fine-tuning, 2024. Disponível em: https://www.tensorflow.org/tutorials/images/transfer_learning?hl=pt-br\. Acesso em: 22 abr. 2024c.

TensorFlow. Uma plataforma completa de machine learning. Disponível em: https://www.tensorflow.org/?hl=pt-br. Acesso em: 10 abr. 2024d.

TinyML Foundation. *TinyML*, 2019. Disponível em: https://www.tinyml.org/. Acesso em: 09 de abr. 2024.

WANG, L.; LIN, Z. Q.; WONG, A. COVID-Net: a tailored deep convolutional neural network design for detection of COVID-19 cases from chest X-ray images. Scientific Reports. 2020, vol. 10, no. 1, p. 19531. DOI 10.1038/s41598-020-76550-z.

WARDEN, P; SITUNAYAKE, D. TinyML: machine learning with TensorFlow Lite on Arduino and ultra-low-power microcontrollers. 1. ed. O'Reilly Media, 2019.

YANG, D.; MARTINEZ, C.; VISUÑA, L. et al. Detection and analysis of COVID-19 in medical images using deep learning techniques. Scientific Reports. 2021, vol. 11, no. 1, p. 19638. DOI 10.1038/s41598-021-99015-3.