Combining Deep Learning Model Compression Techniques

J. V. S. Silva, L. N. Matos, F. Santos, H. O. M. Cerqueira, H. T. Macedo,
B. O. P. Prado, G. J. F. Silva and K. A. Bispo

***Abstract*—In this article, we evaluate the performance of combining several model compression techniques. The techniques assessed were dark knowledge distillation, pruning, and quantization. We use the classification of chest x-rays as a scenario of experimentation. From this scenario, we found that the combination of these three techniques yielded a new model capable of aggregating the individual advantages of each one. In the experiments we used a combination of deep models with 95.05% accuracy, a value higher than that reported in some related works but lower than the state of the art, whose accuracy is 96.39%. The accuracy of the compressed model in turn was 90.86%, a small loss compared to the gain obtained from the reduction, in bytes, in relation to the size of the original model. The size has been reduced from $841MB$ to $40KB$, which opens up the possibility for using the compressed model in edge computing applications.**

***Index Terms*—Deep Learning, model compression, dark knowledge distillation, pruning, quantization.**

I. Introdução

A última década representou um grande avanço para o aprendizado profundo, do inglês *deep learning*. A crescente disponibilidade de bases de dados e a evolução dos recursos de hardware são alguns dos motivos que propiciaram este avanço. Estes fatores fizeram com que surgissem máquinas de aprendizado que realizam suas tarefas tão bem quanto os humanos [1]. As máquinas de arquitetura profunda são, por sua natureza, mais complexas que as suas antecessoras – máquinas de aprendizado raso, do inglês *shallow learning*. Uma vez que as redes profundas possuem mais camadas, consequentemente mais neurônios e conexões, elas têm também maior capacidade de aprender padrões complexos. A maioria dos modelos profundos, principalmente aqueles com maior acurácia, necessitam de uma grande quantidade de recursos computacionais, como memória e processador, para poderem operar, isso acaba aumentando os custos com infraestrutura e dificultando a implantação (*deploy*) de aplicações que utilizam esses modelos em dispositivos com baixo poder de processamento, como dispositivos para Internet das Coisas.

A crescente demanda por dispositos inteligentes e sistemas embarcados que operam, na maioria das vezes, em tempo real e cujo poder de processamento é limitado, exercem crescente influência para criação de redes neurais adaptadas a dispositivos com limitação de memória, processador ou bateria. Por conseguinte, a pesquisa por técnicas de compressão de modelos vem crescendo nos últimos anos. De maneira geral, o objetivo dessas técnicas é, a partir de um modelo mais complexo, obter um mais simples que atenda restrições (de memória e latência por exemplo), sem que haja perda significativa da acurácia.

Neste trabalho, fizemos a investigação de três técnicas de compressão de modelos: destilação do conhecimento obscuro [2], poda [3] e quantização [4]. Em nossos experimentos, analisamos cada técnica separadamente e a utilização delas em conjunto. O estudo de caso para investigação das técnicas foi a classificação de imagens de raio X do tórax para detecção de pneumonia. O exame de raio X do tórax é frequentemente a melhor maneira de diagnosticar a pneumonia [5] e desempenha um papel importante no atendimento clínico [6]. Até onde sabemos, somos o primeiro trabalho a utilizar uma abordagem focada no uso de técnicas de compressão para a base de dados utilizada nos experimentos. Os modelos escolhidos para classificação foram as redes neurais convolucionais, do inglês *convolutional neural networks* (CNN's), visto que nos últimos anos este tipo de rede obteve bastante sucesso em problemas de visão computacional.

Os resultados obtidos demonstraram a eficácia da destilação do conhecimento obcuro, bem como a utilização da poda como uma maneira de reduzir o *overfitting*. Além disso, com o uso da quantização pudemos reduzir em mais de três vezes o tamanho do modelo estudante sem perda de acurácia. Por fim, a combinação dessas três técnicas de compressão de modelos resultou em um novo com um desempenho compatível com o estado da arte e com um custo computacional menor que outros modelos da literatura, possibilitando a embarcação do mesmo.

Uma versão preliminar deste trabalho foi publicada no evento ERBASE [7]. A versão preliminar envolveu apenas a investigação da destilação do conhecimento obscuro. O trabalho atual apresenta significativas extensões e melhorias em relação ao anterior, uma vez que diversos novos experimentos envolvendo poda e quantização foram realizados, além de aprimoramentos na redação do texto.

José Vitor Santos Silva, Universidade Federal de Sergipe (UFS), São Cristóvão, Sergipe, Brasil, jose.silva@dcomp.ufs.br.

Leonardo Nogueira Matos, Universidade Federal de Sergipe (UFS), São Cristóvão, Sergipe, Brasil, leonardo@dcomp.ufs.br.

Flávio Santos, Universidade Federal de Pernambuco (UFPE), Recife, Pernambuco, Brasil, faos@cin.ufpe.br.

Hélisson Oliveira Magalhães Cerqueira, Universidade Federal de Sergipe (UFS), São Cristóvão, Sergipe, Brasil, helissonomc@dcomp.ufs.br.

Hendrik Teixeira Macedo, Universidade Federal de Sergipe (UFS), São Cristóvão, Sergipe, Brasil, hendrik@academico.ufs.br.

Bruno Otávio Piedade Prado, Universidade Federal de Sergipe (UFS), São Cristóvão, Sergipe, Brasil, bruno@dcomp.ufs.br.

Gilton José Ferreira da Silva, Universidade Federal de Sergipe (UFS), São Cristóvão, Sergipe, Brasil, gilton@dcomp.ufs.br.

Kalil Araujo Bispo, Universidade Federal de Sergipe, São Cristóvão (UFS), Sergipe, Brasil, kalil@dcomp.ufs.br.

II. Revisão da Literatura

A. Destilação do Conhecimento Obscuro

A destilação do conhecimento obscuro [2] é uma técnica utilizada para melhorar a capacidade de generalização de uma rede neural. O processo visa transferir o conhecimento de um modelo para outro, sendo feito geralmente de um modelo profundo (modelo professor) para um de arquitetura mais simples (modelo estudante). Usaremos o termo estudante como referência ao modelo com arquitetura mais simples e, para diferenciar a forma como ele é treinado, acrescentaremos quando necessário a designação "treinamento convencional" e "destilação de conhecimento obscuro"ou "destilado".

Durante o treinamento do modelo estudante, ao invés de receber apenas os rótulos, em inglês *labels*, ele também recebe um conjunto de probabilidades geradas pelo modelo professor. Para tal, aplica-se nas *logits* (saídas das camadas lineares) do modelo professor a função *softmax*, modificada pela introdução de um coeficiente de suavização. O coeficiente T incorporado à função *softmax* foi chamado por Hinton de temperatura, Eq. (1).

$$Softmax(x_i) = \frac{e^{\frac{x_i}{T}}}{\sum_j \frac{x_j}{T}} \tag{1}$$

A função *softmax* em sua forma padrão (T=1) gera probabilidades referentes às classes do problema, de modo que uma das classes terá um valor muito próximo de 1 e as restantes valores muito próximos de 0. Isso não é muito diferente das chamadas *hard targets*, comumente usadas no treinamento de redes neurais. Com o aumento do coeficiente T, a distribuição da probabilidade entre as classes fica mais suave. Essa alteração possibilita que o modelo estudante utilize uma distribuição probabilística que conserve não apenas informações de uma das classes, mas também das outras. Nos métodos convencionais de treinamento, essa informação, isto é, os escores das classes perdedoras, não é conhecida, por isso a técnica se chama destilação do conhecimento obscuro. Além de prover mais informação ao modelo estudante, o uso das *soft targets* torna o treinamento mais rápido, uma vez que resulta numa menor variação do gradiente entre as amostras de treinamento, possibilitando que o modelo estudante possa ser treinado com menos dados e frequentemente com uma taxa de aprendizado maior [2].

Durante o treinamento do modelo estudante, a função de perda é calculada usando a saída do modelo estudante, as *soft targets* geradas pelo modelo professor e as *hard targets* (em algumas variações o treinamento é feito usando apenas a saída do modelo estudante e as *soft targets*). Com essas informações a perda é calculada, sendo a função de perda uma soma ponderada composta por duas componentes. A primeira, vide equação (2), consiste na entropia cruzada das probabilidades do modelo professor e das probabilidades do modelo estudante, aplicando-se às *logits* de ambos os modelos a função softmax, com o mesmo valor de T. A segunda componente, vide equação (3), consiste na entropia cruzada das *hard targets* e das probabilidades do modelo estudante (aplicando-se a softmax com T=1). A função de perda resultante pode ser vista na equação (4):

$$L_1 = H(\sigma(z_t; T=\tau), \sigma(z_s; T=\tau)) \tag{2}$$

$$L_2 = H(y, \sigma(z_s; T=1) \tag{3}$$

$$L = \alpha * L_1 + \beta * L_2 \tag{4}$$

Onde z_s e z_t são as *logits* do modelo professor e do modelo estudante, respectivamente. H é a função entropia cruzada. σ é a função softmax parametrizada pela temperatura T. α e β são os pesos. Nota-se, portanto, que essa técnica de treinamento introduz três novos hiperparâmetros: T, α e β. Hinton usou nos experimentos valores de T entre 1 e 20 e obteve resultados melhores com β consideravelmente menor que α.

B. Poda

A poda se baseia na ideia de remover conexões redundantes de uma rede neural. Ela envolve o descarte dos pesos menos relevantes, aqueles que impactam menos na saída da rede (geralmente pesos com valor absoluto menor), tal descarte consiste em zerar esses pesos. Essa técnica vem sendo usada desde o fim da década 80 [8], porém, nos últimos anos, devido à popularização das redes neurais profundas e a crescente demanda por sistemas otimizados para dispositivos para Internet das Coisas, o interesse por ela aumentou.

Existem várias maneiras de realizar a poda em uma rede neural, isto é, diferentes critérios usados para remover os pesos. Alguns métodos podam os parâmetros individualmente (*unstructured pruning*), essa abordagem cria uma rede neural esparsa, o que possibilita o uso de recursos de hardware e software otimizados para operações com matrizes desse tipo. Outros métodos consideram parâmetros em grupos (*structured pruning*), removendo neurônios inteiros, filtros ou canais, essa abordagem equivale a remover blocos inteiros das matrizes de pesos [9]. A maioria das estratégias de poda deriva do algoritmo proposto por [3]. Nesse algoritmo, é feito um treinamento inicial do modelo. Em seguida, cada parâmetro da rede recebe um score, e a rede é podada de acordo com ele. Depois, um novo treinamento é realizado para ajustar o modelo (devido à perda de acurácia que pode ocorrer com o uso da poda), o processo de poda e reajuste pode ser repetido várias vezes.

C. Quantização

De maneira geral, a quantização é o processo de mapear valores de um conjunto maior para um menor, idealmente sem perder muita informação no processo. No contexto das redes neurais, uma aplicação comum é mapear os pesos, normalmente codificados como ponto flutuante de 32 bits, para representações com precisão reduzida como 16 bits [10], inteiros de 8 bits [4] e até mesmo redes binárias (1 bit) [11]. Além de reduzir a quantidade de memória necessária para armazenar e carregar o modelo, o processo de quantização reduz a latência, como demonstrado por [4] que propôs um esquema de quantização baseado no uso de aritmética de inteiros e instruções específicas pra esse tipo de operação. [4] foi capaz acelerar a inferência de diferentes modelos (MobileNets) testados em CPU's ARM.

III. Materiais e Métodos

A metodologia empregada contempla três fases: a primeira envolve a elaboração e treinamento de um modelo alvo por meio do método de destilação do conhecimento obscuro. A segunda envolve a aplicação da poda e quantização nesse modelo alvo. Por fim, a terceira fase envolve a análise da eficácia das técnicas utilizadas, bem como a comparação dos resultados obtidos com outros trabalhos da literatura.

A. Base de Dados

Os testes iniciais dos modelos propostos foram feitos com a base de dados *Chest X-Ray Images* (Pneumonia) [12]. Esta base de dados possui três classes: imagens de raio X do tórax de pessoas saudáveis, de pessoas com pneumonia bacteriana e de pessoas com pneumonia viral, a Fig. 1 apresenta algumas amostras da base de dados. Em nossos experimentos agrupamos as duas classes de pneumonia (bacteriana e viral) em uma só, de modo que o problema passou a ser uma classificação binária. O *dataset* possui ao todo 5840 imagens, sendo 1575 de pessoas saudáveis e 4265 de pessoas com pneumonia viral ou bacteriana. As partições do*dataset* usadas nos experimentos para treinamento e teste dos modelos e a escolha por unir as classes de pneumonia em uma só, basearam-se no que os outros trabalhos da literatura usaram, permitindo que a comparação dos resultados fosse justa.

Fig. 1. Exemplos de amostras do *dataset*.

Durante o treinamento dos modelos propostos, foram usadas técnicas para mitigar os efeitos do sobreajuste, ou, em inglês, *overfitting*. Este é um problema comum nas redes neurais e caracteriza-se por um modelo operar bem nos dados de treinamento, mas possuir um desempenho ruim em dados nunca antes vistos [13]. Nos treinamentos, usamos técnicas de *data augmentation* e regularização L2 ($\lambda = 0,000125$) para mitigar os efeitos do sobreajuste. As transformações de *data augmentation* usadas foram *flip* horizontal aleatório (para a rede aprender a lidar com sintomas da pneumonia em ambos os pulmões) e redimensionamento com corte aleatório (para a rede aprender a associar uma gama mais ampla de ativações espaciais a uma determinada classe) [14]. Durante o treinamento do modelo professor, adicionamos ruído gaussiano às amostras de treino.

B. Destilação do Conhecimento Obscuro

Essa etapa envolve a definição, treinamento e análise do desempenho dos modelos professor e estudante, bem como a análise da eficácia da técnica de destilação do conhecimento obscuro [7]. Os modelos propostos foram implementados em linguagem Python, utilizando o framework de aprendizado de máquina PyTorch.

A arquitetura escolhida para o modelo professor foi um agrupamento de redes, similar ao que foi feito por Chouhan et al. (2020). Foram usadas cinco arquiteturas de modelos previamente treinados na base de dados ImageNet. Os modelos escolhidos para o agrupamento foram: AlexNet [15], DenseNet121 [16], ResNet18 [17], VGG19 [18], e GoogLeNet [19]. Para o treinamento dos modelos do agrupamento (modelo professor), utilizamos a técnica *transfer learning*, esta técnica consiste em aproveitar os pesos de uma rede neural e ajustá-la para um domínio diferente do qual ela foi originalmente treinada. Para tal, a última camada linear de todos os modelos foi modificada e os pesos das camadas iniciais 'congelados', isto é, não foram atualizados durante o treinamento [7].

Para o modelo estudante foi escolhida uma arquitetura simples, vide Fig. 2, com apenas quatro camadas convolucionais, ele também tinha camadas de *maxpooling* e *batch normalization*. Em todos os modelos o treinamento foi feito com o otimizador Adam [20], com tamanho do *batch* 32, com uma taxa de aprendizagem de 0,001, que a cada três épocas era dividida por dois [7].

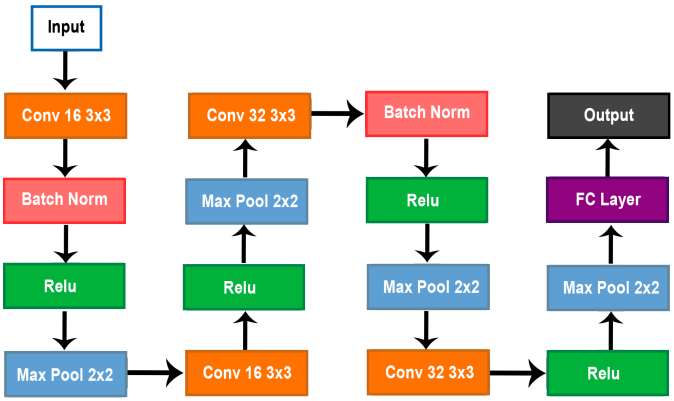


Fig. 2. Arquitetura do modelo estudante.

C. Poda e Quantização

A estratégia de poda utilizada foi do tipo *unstructered pruning*, ou poda não estruturada [9]. Nossa estratégia consistiu na remoção dos pesos com menor valor absoluto, sendo os pesos menores que um limiar definido zerados. O limiar usado para podar os modelos se baseou no desvio padrão dos pesos da rede, nós dividimos o desvio padrão de cada modelo por fatores (chamaremos esses fatores de τ) partindo de τ=0,5 até τ=15. Para cada valor de τ, calculamos a acurácia dos modelos e selecionamos o que manteve a maior acurácia e ao mesmo tempo podou a maior quantidade de parâmetros. Utilizamos essa estratégia para podar o modelo estudante e também os modelos do agrupamento de redes (modelo professor). Para quantizar o modelo estudante utilizamos três diferentes estratégias de quantização pós treinamento disponíveis no *framework* TensorFlow Lite (convertemos o modelo do PyTorch para o TensorFlow Lite), são elas:

- *Dynamic range quantization*: Essa estratégia mapeia os pesos para inteiros de 8 bits e armazena as ativações da rede como ponto flutuante. Durante a inferência, os

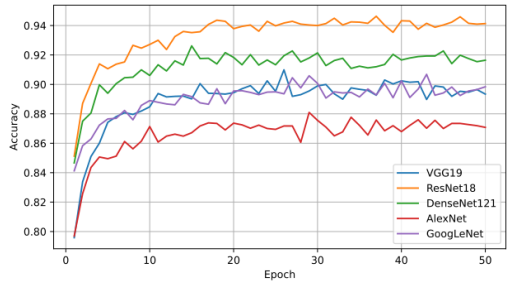
parâmetros das ativações são convertidos dinamicamente para inteiros de 8 bits e em seguida novamente para ponto flutuante. Essa estratégia reduz o tamanho do modelo em até quatro vezes.

- Quantização para ponto flutuante de 16 bits: Essa abordagem converte todos os parâmetros do modelo para ponto flutuante de 16 bits e reduz o tamanho do modelo em até duas vezes.
- Quantização para inteiros de 8 bits: Converte todos os parâmetros do modelo para inteiros de 8 bits. Essa estratégia reduz o tamanho do modelo em até quatro vezes e tende a ter uma latência menor que a *dynamic range quantization*, pois não precisa converter os parâmetros das ativações durante a inferência.

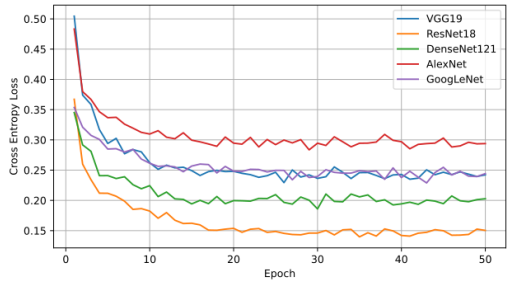
IV. Resultados e Discussões

A. Aplicação das Técnicas de Compressão

O treinamento de cada um dos modelos do agrupamento (modelo professor) foi feito por 50 épocas. A Fig. 3 apresenta a evolução da acurácia e *loss* durante o treinamento de cada um dos modelos. O modelo professor usa a combinação das saídas dos modelos treinados para fazer as predições. As saídas desses modelos foram combinadas calculando a média das *logits*, isto é, das ativações da camada imediatamente anterior à camada *softmax*. Desta forma, o agrupamento formado por várias redes profundas passou a se comportar como uma única rede cujas *logits* foram empregadas no treinamento do modelo estudante. Este, por sua vez, treinado com esta abordagem obteve acurácia de $94,71\%$. Também foi experimentado fazer a predição por votação, utilizando a predição de cada modelo separadamente e escolhendo a classe com mais votos, contudo, essa abordagem obteve uma acurácia de 94,55%.



(a) Acurácia.



(b) *Loss*.

Fig. 3. Acurácia e *loss* de cada um dos modelos do agrupamento ao longo do treinamento. Os dados são do conjunto de treinamento.

Após o treinamento dos modelos do agrupamento eles foram podados. Com o uso da poda a acurácia dos modelos aumentou, com exceção da VGG19, que manteve a mesma acurácia. Esse aumento da acurácia indica que a estratégia de poda utilizada reduziu o *overfitting* dos modelos, demonstrando que a estratégia usada pode servir como uma técnica de regularização. Para realização da poda, vários limiares (várias frações do desvio padrão dos pesos da rede) foram testados (os pesos de valor absoluto inferior ao limiar eram zerados), sendo salvos os modelos que conservavam a maior acurácia. A acurácia de cada um deles no conjunto de teste, antes e depois da poda, e a quantidade de parâmetros podada em relação ao total de parâmetros do modelo são apresentadas na Tabela I.

TABELA I
ACURÁCIA DOS MODELOS DO AGRUPAMENTO, ANTES E DEPOIS DA PODA, E PORCENTAGEM DE PARÂMETROS ZERADOS. OS DADOS SÃO DO CONJUNTO DE TESTE.

Modelo	Acurácia (%)	Acurácia após poda (%)	Percentual podado (%)
VGG19	91,35	91,35	87,35 ($\tau = 1$)
AlexNet	92,47	92,95	91,15 ($\tau = 1$)
GoogLeNet	92,63	92,79	24,75 (τ =6)
Densenet121	93,11	93,27	23,63 ($\tau = 6$)
ResNet18	94,39	94,87	39,43 ($\tau = 4$)
Agrupamento de redes (predição por votação)	94,55	**95,03**	—
Agrupamento de redes (predição por média)	94,71	**95,03**	—

Em seguida, foi feito o treinamento do modelo estudante. Inicialmente, o modelo estudante foi treinado de maneira convencional, isto é, com os dados originais do conjunto de treino. Isto permitiu comparar o desempenho do modelo estudante treinado com as duas diferentes abordagens: com e sem destilação de conhecimento obscuro. Durante os experimentos, percebemos que 25 épocas eram suficientes para o modelo estudante convergir, além disso, durante o treinamento utilizando a destilação do conhecimento obscuro também utilizamos 25 épocas. Na etapa usando a destilação, fizemos vários treinamentos para encontrar os melhores valores para T, α e β e percebemos que 25 épocas eram suficientes para o modelo convergir. Portanto, decidimos usar 25 épocas para os experimentos serem mais rápidos. O modelo estudante treinado convencionalmente obteve acurácia de $86,22\%$ nos dados do conjunto de teste.

Como discutido anteriormente, na seção II-A, o uso da técnica de destilação de conhecimento obscuro introduz três novos hiperparâmetros: T, α e β. Nos experimentos desse trabalho utilizamos valores de T entre 2 e 5, sendo que os melhores resultados foram obtidos com $T = 2$. Já para α e β, definiu-se $\beta = 1 - \alpha$ e foram experimentados valores de $\alpha = 0,9$, $\alpha = 0,95$ e $\alpha = 0,99$. Os melhores resultados foram obtidos com α=0,9. Ao treinar o modelo estudante com a técnica de destilação do conhecimento obscuro (T=2, α=0,9 e β=0,1), atingiu-se uma acurácia de 89,90% no conjunto de teste. Os resultados da destilação obtidos foram aproveitados dos experimentos da versão preliminar deste trabalho [7].

Em seguida, ao aplicarmos a poda, de maneira semelhante aos modelos do agrupamento, a acurácia também aumentou, mais uma vez indicando que a estratégia de poda utilizada serviu como uma técnica de regularização e reduziu o *overfitting*. A acurácia resultante do modelo estudante podado foi de 90,38%, com 4,01% dos parâmetros eliminados. Como o modelo estudante já é um modelo pequeno, com cerca de 30 mil parâmetros, cerca de 360 vezes menor que uma rede profunda como a Resnet18 que possui 11 milhões de parâmetros, era esperado que menos parâmetros fossem podados, uma vez que um modelo pequeno tem menos redundância. A acurácia, precisão, cobertura e F1-Score dos modelos professor e estudante (treinado convencionalmente, com a destilação do conhecimento obscuro e combinando a destilação e a poda) são apresentadas na Tabela II.

TABELA II
RESUMO DA ACURÁCIA, PRECISÃO, COBERTURA E F1-*score* PARA OS MODELOS ESTUDANTE E PROFESSOR. OS DADOS SÃO DO CONJUNTO DE TESTE.

Modelo	Acurácia (%)	Precisão	Cobertura	F1-*Score*
Modelo professor	94,71	0,954	0,934	0,942
Modelo estudante treinamento convencional	86,22	0,893	0,822	0,841
Modelo estudante destilado	89,90	0,913	0,874	0,881
Modelo estudante destilado e podado	90,38	0,915	0,881	0,894

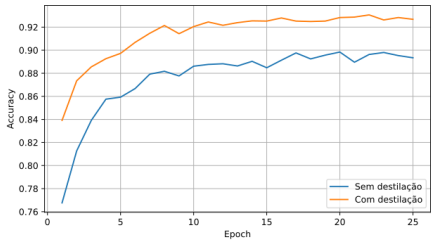
Após a poda do modelo destilado, foi feita a conversão para o TensorFlow Lite e as três estratégias de quantização apresentadas na metodologia Seção III foram aplicadas. A Tabela III apresenta o resumo da acurácia, F1-score, tamanho em KB e taxa de compressão após a utilização de cada tipo de quantização no modelo destilado e podado, modelo este que tinha, antes da aplicação das técnicas de quantização, 127,41 KB de tamanho e 90,38% de acurácia. Com a aplicação das técnicas de quantização, pudemos reduzir em mais de três vezes o tamanho do modelo estudante, ao custo de uma pequena perda na acurácia (*dynamic range quantization*). Realizando a quantização para *float* de 16 bits, reduzimos o modelo em quase duas vezes, sem mudança na acurácia. Com a utilização da quantização para inteiros de 8 bits, a acurácia do modelo aumentou (90,86%), sendo este o nosso melhor resultado.

B. Análise dos Resultados

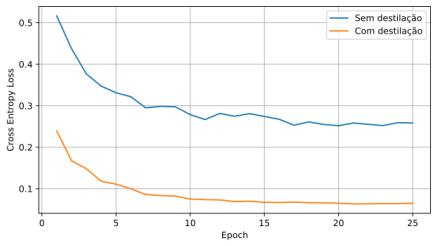
Baseado nesses resultados, percebe-se que é possível melhorar o desempenho de uma rede neural com a utilização da técnica de destilação do conhecimento obscuro. O modelo estudante original, treinado com o modo convencional, obteve uma acurácia 3,68% abaixo do modelo treinado com a destilação do conhecimento obscuro, confirmando ser possível transferir conhecimento de uma rede complexa para um modelo mais simples. Além disso, foi possível tornar o treinamento mais rápido, uma vez que com o uso da técnica de destilação do conhecimento obscuro foi possível atingir um desempenho melhor com um número menor de épocas. Isso pode ser melhor visualizado na Fig. 4, que apresenta os gráficos de acurácia e *loss* ao longo do treinamento do modelo estudante com e sem o uso da destilação do conhecimento obscuro.

TABELA III
ACURÁCIA, F1-*score*, TAMANHO EM KB E TAXA DE COMPRESSÃO APÓS QUANTIZAÇÃO DO MODELO ESTUDANTE DESTILADO E PODADO. OS DADOS SÃO DO CONJUNTO DE TESTE.

Modelo	Acurácia (%)	F1-*Score*	Tamanho (KB)	Taxa de compressão
Modelo estudante destilado e podado sem quantizar	90,38	0,894	127,41	1,0
Dynamic range quantization	90,22	0,891	39,23	3,24
Quantização para *float* de 16 bits	90,38	0,893	67,82	1,87
Quantização para inteiros de 8 bits	90,86	0,899	40,70	3,13



(a) Acurácia.



(b) *Loss*.

Fig. 4. Acurácia e *loss* ao longo das épocas com e sem destilação do conhecimento. Os dados são do conjunto de treinamento.

A estratégia de poda utilizada se mostrou eficaz, principalmente nos modelos do agrupamento de redes (modelo professor). Com a poda pudemos descartar mais de 90% dos pesos da AlexNet e mais de 87% da VGG19. Além disso, a poda foi capaz de reduzir o *overfitting* dos modelos, indicando que a estratégia utilizada pode servir como uma técnica de regularização. Com a aplicação das técnicas de quantização, foi possível reduzir em mais de 3 vezes o tamanho do modelo estudante. Inesperadamente, a quantização para inteiros de 8 bits melhorou a acurácia do modelo estudante. Por fim, o modelo resultante da combinação das técnicas de compressão (modelo estudante destilado, podado e quantizado) obteve uma acurácia $4,64\%$ acima do modelo estudante treinado de modo convencional ($90,86\%$ contra $86,22\%$), sendo ainda mais de 3 vezes menor.

C. Comparação com Outros Trabalhos

Outros trabalhos da literatura já utilizaram modelos de aprendizado profundo para o diagnóstico de pneumonia e outras doenças com base em imagens de raio X do tórax. Os trabalhos de Kermany et al. (2018) [12] e de Chouhan et al. (2020) [14] atingiram bons resultados, vide Tabela IV, usando a técnica de *transfer learning* para aproveitar os pesos de diferentes arquiteturas e ajustá-las para a mesma base de dados que foi usada neste trabalho. Em ambos os trabalhos citados, os autores optaram por agrupar as classes de pneumonia viral e bacteriana em uma só e, assim como foi realizado nos nossos experimentos, realizar classificação binária.

Kermany et al. (2018) conseguiu 92,8% de acurácia aproveitando os pesos de uma rede Inception v3 [21]. Chouhan et al. (2020) fez experimentos com cinco diferentes modelos previamente treinados: AlexNet [15], DenseNet [16], ResNet18 [17], Inception V3 [21] e GoogleNet [19], além de fazer experimentos combinando as saídas dos cinco modelos anteriores (agrupamento de redes neurais), nos quais obteve o melhor desempenho (96,39% de acurácia). Em ambos os trabalhos citados, os resultados foram superiores aos obtidos pelo modelo estudante, ao custo de maior esforço computacional. Nos modelos Resnet18, AlexNet e Densenet121, vide Tabela I, nossos resultados foram superiores aos de Chouhan et al. (2020). A Tabela IV apresenta a comparação dos resultados com os outros trabalhos da literatura (acurácia e tamanho). Ao analisarmos o tamanho em bytes dos modelos, fica clara a vantagem de se utilizar o modelo estudante em relação aos demais, sendo ele o único viável para utilização em arquiteturas não convencionais, como o MCU ESP32, que conta com 520KB de RAM e 4MB de memória FLASH, mais que o suficiente para o modelo estudante.

TABELA IV
QUADRO COMPARATIVO GERAL.

Modelo	Acurácia (%)	Tamanho (bytes)
Estudante (treinamento convencional)	**86,22**	**127 KB**
Estudante (destilado, podado e quantizado)	**90,86**	**40 KB**
DenseNet121 [14]	92,62	27 MB
Inception V3 [12]	92,80	93 MB
AlexNet [14]	92,86	217 MB
GoogLeNet [14]	93,12	21 MB
ResNet18 [14]	94,23	43 MB
Modelo professor agrupamento de redes	**95,03**	**841 MB**
Agrupamento de redes [14]	96,39	401 MB

V. CONCLUSÃO

Os resultados obtidos mostraram a eficácia das técnicas de compressão no cenário dos experimentos. Ao analisar empiricamente apenas a destilação do conhecimento obscuro observamos o emprego de uma técnica que permitia obter, a partir de um modelo já treinado, um novo com uma nova arquitetura, mais simples que a original, que apresentava acurácia pouco inferior. No que concerne a utilização da poda verificamos que permitia, além de reduzir a quantidade de parâmetros, reduzir também o sobreajuste. Por fim, nos experimentos desenvolvidos notamos que o uso da quantização permitiu reduzir em mais de três vezes o tamanho dos modelos testados sem perda de acurácia. Observamos, portanto, as vantagens próprias que cada uma destas técnicas isoladamente possui e verificamos também que seu uso combinado produz uma vantagem cumulativa, abrindo possibilidade para o emprego do modelo comprimido em computação na borda. Por fim, embora os nossos experimentos tenham utilizado somente a base de dados Chest x-ray images (Pneumonia), a abordagem empregada é genérica o suficiente e também pode ser aplicada à outros problemas. O código fonte usado nos experimentos pode ser acessado no seguinte repositório: JvitorS23/combining-model-compression-techniques.

AGRADECIMENTOS

Os autores agradecem a CAPES e FAPITEC-SE pelo suporte financeiro [Edital CAPES/FAPITEC/SE No 11/2016 - PROEF, Processo 88887.160994/2017-00] e [Edital CAPES/FAPITEC/SE No 10/2016 - PROMOB, 88887.157914/2017-00]. Os autores agradecem ainda ao CNPq pela bolsa de produtividade de Hendrik Macedo [DT-II, Processo 304738/2020-4].

REFERÊNCIAS

[1] P. Rajpurkar, J. Irvin, R. L. Ball, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Ding, A. Bagul, C. P. Langlotz, *et al.*, "Deep learning for chest radiograph diagnosis: A retrospective comparison of the chexnext algorithm to practicing radiologists," *PLoS medicine*, vol. 15, no. 11, p. e1002686, 2018.

[2] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[3] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[4] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2704–2713, 2018.

[5] W. H. O. WHO, "Standardization of interpretation of chest radiographs for the diagnosis of pneumonia in children," tech. rep., World Health Organization, 2001.

[6] T. Franquet, "Imaging of pneumonia: trends and algorithms," *European Respiratory Journal*, vol. 18, no. 1, pp. 196–208, 2001.

[7] J. V. Silva and L. Matos, "Detecção de pneumonia usando redes neurais convolucionais treinadas com destilação do conhecimento obscuro," in *Anais da XX Escola Regional de Computação Bahia, Alagoas e Sergipe*, pp. 51–60, SBC, 2020.

[8] S. A. Janowsky, "Pruning versus clipping in neural networks," *Physical Review A*, vol. 39, no. 12, p. 6600, 1989.

[9] D. Blalock, J. J. G. Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?," *arXiv preprint arXiv:2003.03033*, 2020.

[10] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *International conference on machine learning*, pp. 1737–1746, PMLR, 2015.

[11] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 4114–4122, 2016.

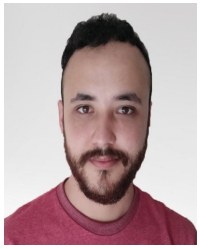
[12] D. S. Kermany, M. Goldbaum, W. Cai, C. C. Valentim, H. Liang, S. L. Baxter, A. McKeown, G. Yang, X. Wu, F. Yan, *et al.*, "Identifying medical diagnoses and treatable diseases by image-based deep learning," *Cell*, vol. 172, no. 5, pp. 1122–1131, 2018.

[13] R. Caruana, S. Lawrence, and L. Giles, "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping," *Advances in neural information processing systems*, pp. 402–408, 2001.
[14] V. Chouhan, S. K. Singh, A. Khamparia, D. Gupta, P. Tiwari, C. Moreira, R. Damaševičius, and V. H. C. De Albuquerque, "A novel transfer learning based approach for pneumonia detection in chest x-ray images," *Applied Sciences*, vol. 10, no. 2, p. 559, 2020.
[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.
[16] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.
[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
[18] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[19] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.
[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[21] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

José Vitor Santos Silva é aluno do curso de graduação em engenharia da computação na Universidade Federal de Sergipe, onde também atua como pesquisador na área de Aprendizado de Máquina (AM). Seus interesses envolvem aprendizado profundo, visão computacional e *edge computing*. Alguns de seus trabalhos científicos foram premiados em eventos no Brasil. Atualmente trabalha como desenvolvedor de software.

Leonardo Matos é professor associado do Departamento de Computação da Universidade Federal de Sergipe, Brasil. Sua principal área de interesse é *Machine Learning*, particularmente, *Explainable Artificial Intelligence*, compressão de modelos e aplicações, incluíndo reconhecimento de fala e visão computacional. Possui diversos artigos publicados em conferências e periódicos científicos. Alguns de seus trabalhos recentes com alunos de graduação foram premiados em eventos no Brasil. Ele tem lecionado *Machine Learning* no Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe, orientado alunos e participado como membro de juri de defesas de dissertação de mestrado. É membro da Sociedade Brasileira de Computação, colaborador do grupo ISLab na Universidade do Minho em Portugal e revisor de artigos em conferências e periódicos científicos na área de Computação.

Flávio Santos é um estudante de doutorado em ciência da computação no Centro de Informática (CIn) da Universidade Federal de Pernambuco (UFPE). Além disso, é também um pesquisador no Centro Algoritmi da escola de engenharia da Universidade do Minho, Braga, Portugal. Ele tem desenvolvido trabalhos científicos em Deep Learning, com aplicações em processamento de linguagem natural e visão computacional. Atualmente, seu interesse de pesquisa principal é interpretação e robustez quanto aos ataques adversários dos modelos de aprendizado profundo.



Hélisson Oliveira Magalhães Cerqueira é um estudante de ciência da computação na Universidade Federal de Sergipe e atua profissionalmente na área de ciência de dados da Secretaria da Fazenda de Sergipe. Seus interesses envolvem aprendizado de máquina, big data, desenvolvimento de software.

Hendrik Teixeira Macedo obteve o diploma de bacharel em Ciência da Computação pela Universidade Federal de Sergipe (UFS) em 1998 e concluiu o doutorado em Ciência da Computação pela Universidade Federal de Pernambuco (UFPE), com estágio "Sandwich" na Universidade de Paris VI, em 2002. Desde 2006, Hendrik ocupa o cargo de docente efetivo do DCOMP/UFS e, desde 2010, faz parte do quadro de docentes permanentes do Núcleo de Pós-Graduação em Ciência da Computação (PROCC/UFS). Desde 2015, Hendrik é bolsista de Produtividade em Desenvolvimento Tecnológico e Extensão Inovadora do CNPq. Áreas de pesquisa de maior interesse são Aprendizado de Máquina, Processamento de Linguagem Natural e Sistemas Multiagentes.

Bruno Otávio Piedade Prado é Professor Associado do Departamento de Computação (DCOMP) da Universidade Federal de Sergipe (UFS) e atua nesta instituição deste 2012. O foco principal de suas pesquisas tem sido a utilização de Aprendizado de Máquina (AM) para resolução de problemas complexos de classificação em ambiente restrito de sistemas embarcados. Pelo seu baixo custo unitário e escalabilidade, os esforços de pesquisa estão concentrados em reduzir a complexidade computacional, sem comprometer a utilidade das classificações obtidas. Para tanto, vem sendo exploradas estratégias de compressão de redes, utilização de instruções vetoriais (SIMD) ou aceleração por hardware reprogramável dedicado com FPGA.

Gilton José Ferreira da Silva é Professor (DCOMP/UFS), Orientador de Pós-graduação (PROCC/UFS), Mentor e Influenciador Digital (@giltonmal). Doutor em Ciência da Propriedade Intelectual. Trabalha com as linhas de pesquisas de Sistemas de Informação (SI), Cidades Inteligentes, Design Thinking, UX, Criatividade, Inovação, Tecnologias Educacionais, Propriedade Intelectual e Marketing Digital.

Kalil Araujo Bispo possui graduação em Ciência da Computação pela Universidade Federal de Sergipe (2006), mestrado em Ciências da Computação pela Universidade Federal de Pernambuco (2009) e doutorado em Ciências da Computação pela Universidade Federal de Pernambuco (2015). Atualmente é professor adjunto da Universidade Federal de Sergipe. Tem experiência na área de Ciência da Computação, com ênfase em Sistemas Distribuídos.