Uma nova estratégia para o treinamento distribuído de redes neurais

Lucca Gamballi, Daniel G. Tiglea, Renato Candido e Magno T. M. Silva

***Resumo*— Neste artigo, é proposta uma nova estratégia para combinação dos pesos de redes neurais em um cenário distribuído. A abordagem distribuída ganha espaço em aplicações que levam em conta uma grande quantidade de dados, cuja privacidade deva ser mantida. Contudo, caso um ou mais nós da topologia estejam contaminados por ruído, o modelo global pode ter seu desempenho prejudicado. Isso ocorre, pois o modelo local contaminado propaga o efeito do ruído para o modelo global. A estratégia proposta é avaliada em diferentes topologias, utilizando redes neurais convolucionais. Com o intuito de preservar a privacidade, apenas os pesos dos modelos são compartilhados, de tal forma que uma rede tenha acesso apenas ao conjunto de dados local. Resultados de classificação de imagens mostram que a estratégia proposta é robusta à presença do ruído.**

***Palavras-Chave*— Redes neurais, processamento distribuído, topologia, estratégia de combinação, ruído**

***Abstract*— In this paper, a new strategy for combining weights of neural networks in a distributed scenario is proposed. The distributed approach has been employed in applications that take into account a great amount of data, whose privacy must be maintained. However, if one or more nodes in the topology are contaminated by noise, the performance of the global model may be deteriorated. This occurs since the contaminated local model propagates the noise effect to the global one. The proposed strategy is evaluated with different topologies using convolutional neural networks. In order to preserve privacy, only the models weights are shared, such that a network only has access to the local dataset. Image classification results show that the proposed strategy is robust to the presence of noise.**

***Keywords*— Neural networks, distributed processing, topology, combination strategy, noise.**

I. Introdução

Redes neurais se tornaram o estado da arte na classificação de dados como imagens, vídeos e dados médicos [1]–[3]. O desenvolvimento do modelo de uma rede neural pode ser dividido em duas etapas: a de treinamento e a de validação. No treinamento supervisionado, os parâmetros da rede são ajustados utilizando um conjunto de dados rotulados para minimizar uma função custo. Na validação ou teste, o modelo treinado é testado utilizando um conjunto de dados rotulados, disjunto do conjunto utilizado no treinamento. Em geral, para o sucesso das etapas, é necessária uma grande quantidade de dados para que a rede apresente um bom desempenho [4], [5].

A forma tradicional de treinamento, conhecida como treinamento centralizado, é feita em apenas uma única máquina que contém todo o conjunto de dados. Contudo, com o advento e consolidação da Internet das Coisas, a coleta de dados se tornou cada vez mais veloz e abundante, o que implica uma enorme disponibilidade de dados [6]–[8]. Em certas aplicações, a quantidade de dados pode ser tão grande que acaba sobrecarregando a máquina. Além disso, dada a natureza de algumas aplicações, em especial as da área da saúde, é possível que esses dados contenham informações sensíveis que devem ser mantidas privadas [9], [10]. A fim de respeitar a privacidade, deve-se evitar compartilhar dados desse tipo em uma máquina centralizada [11], [12]. Nessas condições, uma abordagem de aprendizagem descentralizada tem ganhado espaço e várias soluções baseadas em um treinamento distribuído podem ser encontradas na literatura [13]–[16].

No treinamento distribuído de redes neurais, existem diferentes máquinas capazes de trocar informações sobre seus modelos enquanto os dados não são compartilhados [9]–[17]. Nessa abordagem, cada máquina é responsável por treinar uma rede neural de mesma arquitetura, usando apenas o conjunto de dados local e transmitir seu modelo às outras. A comunicação dos modelos entre as máquinas ocorre obedecendo a uma topologia previamente definida. A partir dos modelos recebidos e seu próprio modelo, cada máquina calcula uma combinação local incorporando informações da comunicação. Dessa forma, é possível treinar um modelo que considera todo o conjunto de dados sem que as máquinas sejam sobrecarregadas e preservando a privacidade dos dados [10].

Em geral, a combinação dos modelos é realizada sem levar em conta a existência de ruído nos dados de treinamento de cada máquina. A discrepância entre os níveis de ruído observados em cada conjunto de dados pode ser prejudicial ao desempenho do modelo global. Isso ocorre porque a presença de ruído contamina o modelo local, que por sua vez, é usado durante a combinação. Para contornar essa situação, é proposta neste artigo a estratégia de combinação Aceita/Rejeita (AcRe). Quando adotada, essa estratégia permite que um nó decida aceitar ou rejeitar um modelo na etapa de combinação, amenizando o efeito do ruído.

O artigo está organizado da seguinte forma. Na Seção II, aborda-se o treinamento distribuído de redes neurais. Na Seção III, descreve-se a estratégia proposta. Na Seção IV, detalha-se o banco de dados e são mostrados resultados de classificação de imagens comparando a abordagem distribuída proposta com outras abordagens existentes na literatura. Por fim, a Seção V apresenta as principais conclusões do trabalho.

II. Treinamento distribuído de redes neurais

Em uma abordagem distribuída, existem V máquinas capazes de se comunicar. Cada uma delas contém um conjunto de

Lucca Gamballi, Daniel G. Tiglea, Renato Candido e Magno T. M. Silva, Depto. de Engenharia de Sistemas Eletrônicos, Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brasil, e-mails: lucca.gamballi@usp.br, {dtiglea, renatocan}@lps.usp.br e magno.silva@usp.br. Este trabalho foi financiado pela CAPES (88887.512247/2020-00 e código de financiamento 001), pelo CNPq (303826/2022-3 e 404081/2023-1) e pela FAPESP (2021/02063-6).

treinamento $\mathcal{D}_i$ de cardinalidade N_i, $i = 1,2,\ldots,V$. Convém assumir que tais conjuntos são disjuntos, de tal forma que $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset$, para todo i, $j = 1,2,\ldots,V$ e $i \neq j$. Isso se deve ao fato de que em cenários realistas é esperado que as amostras presentes em cada $\mathcal{D}_i$ tenham sido coletadas de forma independente. As máquinas também têm acesso a um conjunto de teste. Neste artigo, considera-se que esse conjunto, denotado por $\mathcal{D}_T$, seja comum a todas elas, ao contrário dos dados de treinamento. Além disso, por motivos de proteção à privacidade, assume-se que não ocorra transferência de dados entre as V máquinas [11], [12].

A comunicação entre as máquinas ocorre por meio de uma topologia previamente estabelecida. A Fig. 1 representa um exemplo de topologia. Os nós do grafo mostrados nessa figura correspondem a diferentes máquinas, enquanto as arestas indicam a possibilidade de comunicação entre pares de nós. Cada nó i possui sua própria vizinhança $\mathcal{V}_i$, com cardinalidade $|\mathcal{V}_i|$, definida pelo conjunto de nós que se comunicam diretamente com o nó i, incluindo o próprio nó i.

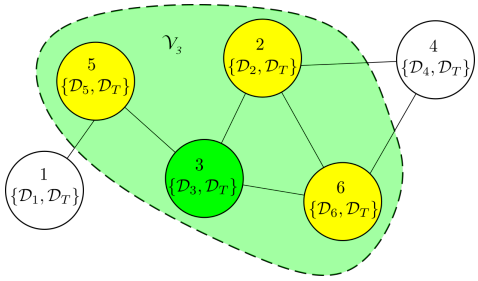

Fig. 1: Exemplo de topologia distribuída, em que cada nó i treina uma rede neural com base no conjunto de treinamento $\mathcal{D}_i$. Neste exemplo, a vizinhança do nó 3 é $\mathcal{V}_3 = \{5,3,2,6\}$.

Assim como na abordagem centralizada, o objetivo de todos os nós é adaptar os pesos e *biases* de suas redes neurais de forma a minimizar uma função custo, comum a todos os nós. É usual considerar em um problema de classificação multiclasse a entropia cruzada categórica como função custo. Com isso, a rede neural do nó i minimiza [18]

$$J_{\mathrm{ECC}_i} = -\sum_{n_i=1}^{N_i}\sum_{\ell=1}^{C} d_{i,\ell}(n_i)\ \ln[y_{i,\ell}(n_i,\mathcal{W}_i)], \qquad (1)$$

em que $y_{i,\ell}(n_i,\mathcal{W}_i)$ é a ℓ-ésima saída da rede para o n_i-ésimo dado de treinamento com rótulo $d_{i,\ell}(n_i)$, $\mathcal{W}_i$ representa o conjunto das matrizes de pesos e *biases* $\mathbf{W}_i^{[k]}$ de todas as camadas, $k = 1,2,\ldots,K$, C representa o número de classes e N_i o número de dados de treinamento da máquina i.

Diferentemente do caso centralizado, o treinamento distribuído pode ser dividido em etapas de adaptação e combinação. Durante a adaptação, ocorre a atualização dos parâmetros locais das redes neurais. Nessa etapa, as estimativas locais $\mathbf{\Psi}_i^{[k]}$, de mesma dimensão das matrizes de pesos $\mathbf{W}_i^{[k]}$, são atualizadas com o algoritmo *backpropagation* [4], [5]. Dessa maneira, a atualização de $\mathbf{\Psi}_i^{[k]}$ é dada por

$$\mathbf{\Psi}_i^{[k]}(m+1) = \mathbf{W}_i^{[k]}(m) - \eta\frac{\partial J_{\mathrm{ECC}_i}}{\partial \mathbf{W}_i^{[k]}(m)}, \qquad (2)$$

em que η é o passo de adaptação, $m = 1,2,\ldots M$ é a iteração do algoritmo. Considera-se o modo de treinamento *mini-batch* em que M iterações do *backpropagation* são realizadas dentro de uma época [5].

Em seguida, ao final das M atualizações realizadas em uma época, ocorre a transmissão de modelos locais para todos os nós da vizinhança, o que possibilita a combinação dos modelos. Na combinação, cada nó agrega os modelos recebidos de seus vizinhos em um único modelo. Dessa forma, o ajuste de $\mathbf{W}_i^{[k]}$ pode ser descrito por [10]

$$\mathbf{W}_i^{[k]}(p) = \textstyle\sum_{j\in\mathcal{V}_i} a_{ij}\mathbf{\Psi}_j^{[k]}(pM), \qquad (3)$$

em que p é a época do algoritmo e $a_{ij} \geq 0$ são pesos de combinação, tal que $a_{ij} = 0$ se as máquinas i e j não estão diretamente conectadas e $\sum_{j=1}^{V} a_{ij} = 1$ para todo i, j. Existem diversas regras possíveis para a escolha desses pesos. Neste trabalho, adota-se a regra Metropolis dada por [9], [19]

$$a_{ij} = \begin{cases} \dfrac{1}{\max\{|\mathcal{V}_i|,|\mathcal{V}_j|\}}, & \text{se } j \in \mathcal{V}_i \text{ e } j \neq i \\ 1 - \sum_{q\in\mathcal{V}_i} a_{iq}, & \text{se } j = i \\ 0, & \text{caso contrário} \end{cases}. \qquad (4)$$

A situação em que os pesos de combinação valem $a_{ij} = 1/V$, para todo i, j, corresponde a um cenário especial de topologia completamente conectada. Nesse caso, a Equação (3) é executada V vezes (uma em cada nó) sem que os parâmetros mudem, dado que o nó i transmite seu modelo a todos os outros nós e também recebe modelos de todos eles. Sendo assim, para economizar custo computacional, é possível reorganizar a topologia removendo as conexões entre os nós e adicionando o nó $V+1$ que é unicamente responsável por receber os modelos dos V nós previamente existentes, executar uma única vez a Equação (3) e retransmitir o resultado para cada nó. Dessa forma, a Equação (3) deixa de ser executada V vezes e passa a ser computada uma única vez em uma estratégia centralizada comumente chamada de aprendizado federado.

III. Uma nova estratégia de combinação

A presença de nós com dados ruidosos pode afetar o modelo global devido à inclusão de seus modelos na etapa de combinação [20]. Para evitar a deterioração do modelo global, é proposta a seguir uma estratégia para a realização da etapa de combinação da Equação (3), denominada de Aceita/Rejeita (AcRe). A ideia é permitir aos nós selecionar quais modelos serão considerados nessa etapa com base no desempenho de cada um deles, amenizando o impacto do ruído.

Considerando uma topologia distribuída, em cada época as máquinas devem adaptar as estimativas locais, transmitir tais estimativas e então combiná-las. Após ter recebido $|\mathcal{V}_i| - 1$ modelos, para decidir quais deles serão efetivamente usados, cada nó compara o valor da função custo dos modelos recebidos com a de seu modelo. Dessa forma, o nó i, compara o valor de J_{ECC_i} obtido a partir do seu modelo com o valor de J_{ECC_j} obtido para cada nó j na vizinhança de i. Vale ressaltar que para a comparação ser justa, os valores de J_{ECC} devem ser obtidos a partir do mesmo conjunto de dados de validação $\mathcal{D}_T$. Caso $J_{\mathrm{ECC}_j} \leq J_{\mathrm{ECC}_i}$ o nó i decide aceitar o modelo do nó j na etapa de combinação. Caso contrário, o modelo do nó j é rejeitado. Como essa decisão altera a vizinhança de cada nó, torna-se necessário definir para cada

nó uma vizinhança temporária que contém apenas os nós que tiveram seus modelos aceitos e que muda a cada época. A vizinhança temporária do nó i ao final das M iterações da p-ésima época é denotada por $\mathcal{T}_i(p)$ e contém apenas os nós cujo desempenho é superior ao do nó i e o próprio nó i, tendo cardinalidade $|\mathcal{T}_i(p)|$. A atualização de $\mathcal{T}_i(p)$ consiste em excluir temporariamente os nós relacionados aos modelos rejeitados da vizinhança. Isso permite ajustar os pesos de combinação $a_{ij}(p)$, recalculando-os de acordo com a regra adotada, como a Metropolis por exemplo. Cabe observar que esse ajuste deve ser feito ao final de cada época e por isso, esses pesos passam a ser denotados em função de p. Assim, a nova combinação passa a ser descrita por

$$\mathbf{W}_i^{[k]}(p) = \sum_{j\in\mathcal{T}_i(p)} a_{ij}(p)\boldsymbol{\Psi}_j^{[k]}(pM). \tag{5}$$

O funcionamento da estratégia de combinação AcRe é descrito pelo pseudocódigo apresentado no Algoritmo 1. Por simplicidade, é mostrado apenas o cálculo regressivo do *backpropagation* modificado. No entanto, o cálculo progressivo é necessário para a execução do algoritmo.

Algoritmo 1: Combinação AcRe

$\mathbf{W}(0) \leftarrow incializa$
para $p \leftarrow 1; p < P; p \leftarrow p+1$ **faça**
  para $i \leftarrow 1; i < V; i \leftarrow i+1$ **faça**
    para $m \leftarrow 1; m < M; m \leftarrow m+1$ **faça**
      para $k \leftarrow K; k > 0; k \leftarrow k-1$ **faça**
        $\boldsymbol{\Psi}_i^{[k]} \leftarrow \mathbf{W}_i^{[k]} - \eta \frac{\partial J_{\mathrm{ECC}_i}}{\partial \mathbf{W}_i^{[k]}}$
      fim
    fim
    $\boldsymbol{\Psi}_i^{[k]}$ é transmitida a cada nó do conjunto $\mathcal{V}_i$
  fim
  para $i \leftarrow 1; i < V; i \leftarrow i+1$ **faça**
    $\mathcal{T}_i(p) \leftarrow \{\}$
    para $j \in \mathcal{V}_i$ **faça**
      se $J_{\mathrm{ECC}_j} \leq J_{\mathrm{ECC}_i}$ **então**
        Inserir elemento j em $\mathcal{T}_i(p)$
      fim
      Atualizar peso de combinação $a_{ij}(p)$
    fim
    para $k \leftarrow 1; k < K; k \leftarrow k+1$ **faça**
      $\mathbf{W}_i^{[k]}(p) \leftarrow \sum_{j\in\mathcal{T}_i(p)} a_{ij}(p)\boldsymbol{\Psi}_j^{[k]}(pM)$
    fim
  fim
fim

Uma forma simplificada de entender os efeitos dessa nova estratégia de combinação está ilustrada na Fig. 2. Nela, é representada a mesma topologia da Fig. 1, porém durante uma das épocas do treinamento quando foi adotada a estratégia AcRe. Caso o nó i transmita seu modelo ao nó j, isso significa que $J_{\mathrm{ECC}_i} \leq J_{\mathrm{ECC}_j}$. Considerando que $J_{\mathrm{ECC}_i} < J_{\mathrm{ECC}_j}$, o nó j não transmite seu modelo ao nó i. Dessa forma o grafo que representa a topologia deixa de ser bidirecional e passa a ser interpretado como um grafo direcionado, como ilustra a Fig. 2. Na rara ocasião em que $J_{\mathrm{ECC}_i} = J_{\mathrm{ECC}_j}$, a conexão entre i e j continua sendo bidirecional. Cabe observar que no exemplo da Fig. 2, apenas $J_{\mathrm{ECC}_6} > J_{\mathrm{ECC}_3}$ e por isso, o nó 6 deixa de pertencer temporariamente à vizinhança do nó 3.

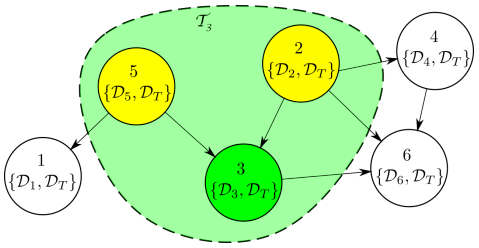

Fig. 2: Exemplo de topologia distribuída, em que cada nó i treina uma rede neural com base no conjunto de treinamento $\mathcal{D}_i$ adotando a estratégia AcRe. Neste exemplo, temos $J_{\mathrm{ECC}_5} < J_{\mathrm{ECC}_3}$ e $J_{\mathrm{ECC}_2} < J_{\mathrm{ECC}_3}$. Com isso, a vizinhança temporária do nó 3 é $\mathcal{T}_3 = \{5, 3, 2\}$.

IV. Banco de dados e resultados de simulação

Nesta seção, são apresentados resultados de classificação de imagens. O banco de dados considerado é o CIFAR-10[1] [21], composto por 60000 imagens coloridas com dimensões 32×32. Cada imagem representa uma de dez possíveis classes, definidas como: aviões, automóveis, pássaros, gatos, veados, cães, sapos, cavalos, navios e caminhões. O conjunto de dados contém 5000 imagens de cada classe para o treino, assim como 1000 imagens de cada classe para o teste. Na Fig. 3 são mostrados exemplos das classes automóvel e cavalo.

Fig. 3: Exemplos de imagens do conjunto CIFAR-10.

Para avaliar o efeito de se adotar a estratégia AcRe, foram consideradas três topologias distribuídas. Cada uma possui seis nós e diferem entre si apenas no número de conexões. Essas topologias estão mostradas nas Figs. 4(a), 4(b) e 4(c). A topologia da Fig. 4(a) contém 5 conexões e nós com no máximo 2 conexões. Por sua vez, as topologias das Figs. 4(b) e 4(c) contêm, respectivamente, 10 e 15 conexões, e nós com no máximo 4 e 5 conexões. Nas três topologias, o desempenho obtido pela estratégia AcRe é comparado com os desempenhos de outras duas estratégias de combinação: (i) o treinamento distribuído *Convencional* e (ii) e uma estratégia de combinação *Aleatória*. No caso *Convencional*, o treinamento ocorre como descrito pelas Equações (2) e (3). Em contrapartida, a estratégia *Aleatória* consiste em amostrar aleatoriamente metade dos nós durante a etapa de combinação, e incluir apenas os modelos dos nós selecionados nessa etapa. Além disso, considera-se também o desempenho obtido pelo treinamento com a estratégia *Centralizada* como referência.

Nas três topologias representadas na Fig. 4, cada nó representa uma máquina que possui um subconjunto $\{\mathcal{D}_i, \mathcal{D}_T\}$

[1] https://www.cs.toronto.edu/~kriz/cifar.html

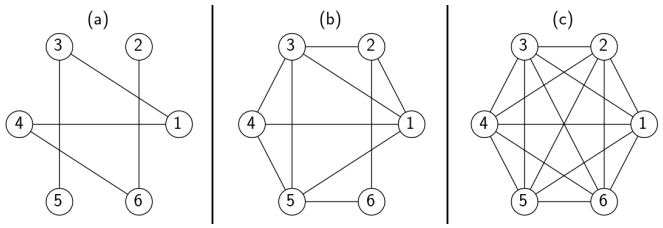

Fig. 4: Topologias consideradas

de [21] e treina uma rede neural convolucional (*Convolutional Neural Network – CNN*) [5]. Cada $\mathcal{D}_i$ é construído de tal forma que contenha $\lfloor 5000/6 \rfloor$ imagens de cada classe, mantendo os conjuntos $\mathcal{D}_i$ disjuntos. Por sua vez, $\mathcal{D}_T$ considera 1000 amostras de cada classe e é comum a todas as máquinas. A CNN considerada é composta por três camadas convolucionais com 32, 64, e 64 filtros, todos de dimensões (3×3). Entre camadas convolucionais é considerado o procedimento de *max-pooling* [5]. A saída da última camada convolucional é vetorizada e alimenta uma rede perceptron multicamada (*Multilayer Perceptron – MLP*) [4] com três camadas ocultas contendo 256, 128 e 64 neurônios respectivamente, e uma camada de saída contendo 10 neurônios. A camada de saída conta com a função de ativação *Softmax*, ao passo que as camadas ocultas e convolucionais valem-se da função de ativação ReLU. Em cada camada, os pesos são inicializados de acordo com a regra Glorot [22]. No treinamento, foi considerado o algoritmo de otimização Adam [23] com parâmetros $\beta_1 = 0{,}9$, $\beta_2 = 0{,}99$ e $\epsilon = 10^{-7}$. Por fim, utilizaram-se passo de adaptação $\eta = 0{,}001$, tamanho de *mini-batch* $b = 32$ e 15 épocas de treinamento.

Cada topologia foi simulada com diferentes números de nós ruidosos, variando de 0 (sem nó ruidoso) a 6 (todos os nós ruidosos). No nó ruidoso, as imagens do conjunto de treinamento foram corrompidas com ruído branco gaussiano de modo a ter uma relação sinal ruído (*signal-to-noise ratio - SNR*) de -20 dB. A Fig. 5 mostra o efeito da adição do ruído nas mesmas imagens da Fig. 3.

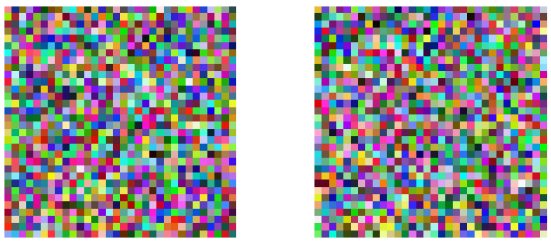

Fig. 5: Exemplos de imagens após adição de ruído.

Na Fig. 6, são mostradas a acurácia média obtida com os dados de teste em função do número de nós ruidosos para as topologias da Fig. 5. Foram feitas 100 realizações de cada experimento e considerado um intervalo de confiança de 95%. Nota-se que para as três topologias, quanto maior o número de nós ruidosos, menor a acurácia média. Dentre as estratégias consideradas, o desempenho da estratégia AcRe é o que mais se aproxima do desempenho da estratégia *Centralizada*. Esse resultado é menos expressivo no caso da topologia da Fig. 4(a), como mostrado na Fig. 6(a). Para esse caso, mesmo quando não há nenhum nó ruidoso, a menor diferença de acurácia entre o caso centralizado e distribuído ocorre para a estratégia *Convencional* e vale aproximadamente $0{,}1$. Analisando os resultados para essa topologia, observa-se que quando há nós ruidosos, a estratégia AcRe supera levemente as demais estratégias distribuídas. Na ausência de nó ruidoso, a estratégia AcRe acaba rejeitando modelos que não estão contaminados por ruído, o que reduz ainda mais a já baixa conectividade e descarta informação útil sobre o problema. Isso faz com que a estratégia *Convencional*, na qual não há rejeição, aproveite melhor as poucas conexões entre as máquinas e supere a AcRe neste caso. Contudo, nota-se que quando o ruído é introduzido nos dados, torna-se mais vantajoso rejeitar o modelo por causa do ruído envolvido do que manter a conexão.

Analisando as Figs. 6(b) e 6(c), percebe-se que os efeitos positivos da estratégia AcRe se tornam mais pronunciados. Nesses casos, tanto a estratégia *Convencional* como a *Aleatória* apresentam desempenho inferior ao da AcRe para todos os números de nós ruidosos, inclusive para o caso em que não há nenhum nó ruidoso. Isso ocorre pois, por mais que alguns modelos sejam rejeitados, a maior quantidade de conexões entre as máquinas permite que a troca de informação seja suficiente para superar a estratégia *Convencional*, ao contrário do que é visto para a topologia da Fig. 4(a). Além disso, nota-se também que para as topologias das Figs. 4(b) e 4(c) há um ganho de desempenho maior em relação ao que é observado na topologia da Fig. 4(a). Considerando o caso de 4 nós ruidosos, por exemplo, observa-se que a estratégia AcRe supera as demais em aproximadamente $0{,}38$ de acurácia nas topologias com maior número de conexões, enquanto que na topologia com menos conexões o ganho está em torno de $0{,}08$.

O mesmo experimento foi repetido para diferentes valores de SNR, considerando a topologia da Fig. 1(b) e 10 realizações. A Tabela I expõe os resultados de acurácia média obtidos para as três estratégias distribuídas em cada valor de SNR considerado. As estratégias com os maiores valores de acurácia para cada par de SNR e número de nós ruidosos estão destacados em negrito. Nota-se que as estratégias AcRe e *Convencional* possuem desempenho semelhante, em torno $0{,}63$, para as relações sinal ruído de 20 e 10 dB, independentemente do número de nós ruidosos. Ao passo que a *Aleatória* é ligeiramente inferior, próxima de $0{,}61$. Esse comportamento é esperado para os casos em que a SNR é alta, uma vez que o ruído não é capaz de alterar as características dos dados. Entretanto, o ganho da estratégia AcRe torna-se evidente para os casos em que a SNR é negativa. Considerando a SNR de -10 dB, a AcRe supera as demais estratégias distribuídas para todos os números de nós ruidosos. No caso em que a SNR vale -20 dB esse efeito é ainda mais pronunciado. Em especial, observa-se o maior ganho de desempenho da AcRe quando há 4 nós ruidosos.

Por fim, independentemente da estratégia adotada, as três topologias têm complexidade semelhante, levando aproximadamente 634 segundos[2] para completar o treinamento para qualquer das estratégias distribuídas. Isso é explicado pelo

[2]Experimentos foram executados em uma máquina com processador Intel Xeon E5-2620, de frequência 2,4 GHz e 32 GB disponíveis de RAM.

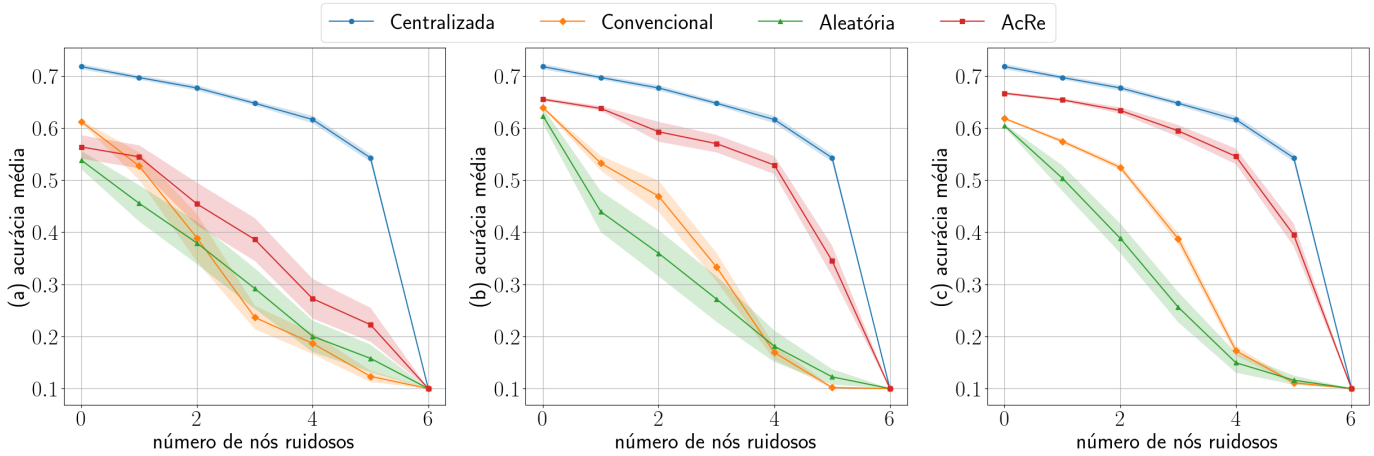

Fig. 6: Acurácia média para: (a) Topologia da Fig. 6(a), (b) Topologia da Fig. 6(b) e (c) Topologia da Fig. 6(c)

TABELA I: Acurácia média para a topologia da Fig. 6(b) para os diferentes valores de SNR.

		Nós ruidosos						
SNR	Estratégia	0	1	2	3	4	5	6
20 dB	AcRe	**0,642**	**0,643**	**0,648**	**0,657**	**0,652**	**0,649**	**0,651**
	Tradicional	0,631	0,634	0,640	0,640	0,633	0,634	0,638
	Aleatória	0,605	0,612	0,607	0,617	0,609	0,615	0,624
10 dB	AcRe	**0,646**	**0,640**	**0,648**	**0,649**	**0,649**	**0,647**	**0,647**
	Tradicional	0,632	0,635	0,639	0,638	0,632	0,633	0,635
	Aleatória	0,611	0,603	0,604	0,614	0,618	0,615	0,615
-10 dB	AcRe	**0,648**	**0,636**	**0,629**	**0,594**	**0,573**	**0,519**	0,476
	Tradicional	0,634	0,590	0,556	0,529	0,512	0,493	**0,479**
	Aleatória	0,612	0,523	0,496	0,470	0,444	0,423	0,420
-20 dB	AcRe	**0,655**	**0,638**	**0,593**	**0,570**	**0,529**	**0,345**	**0,100**
	Tradicional	0,639	0,532	0,469	0,333	0,169	0,102	**0,100**
	Aleatória	0,623	0,439	0,359	0,271	0,181	0,122	**0,100**

fato de que o fator determinante no tempo de treinamento é o algoritmo *backpropagation*. Esse algoritmo é executado em todas as máquinas e envolve o mesmo número de operações independentemente da estratégia e da topologia. Dessa forma, vale notar que a estratégia AcRe é capaz de prover um melhor desempenho sem que a complexidade aumente.

V. Conclusões

O treinamento distribuído de redes neurais se apresenta como uma abordagem de vasto potencial, principalmente para as aplicações que devem respeitar a privacidade ou ainda que podem sobrecarregar a máquina. Neste trabalho, foi proposta uma nova estratégia de combinação dos pesos de redes neurais distribuídas que é robusta à presença de ruído. Após comparar a nova estratégia com as outras alternativas de treinamento distribuído, verificou-se um ganho no desempenho considerando diferentes proporções de nós ruidosos na rede, conectividade da topologia e níveis de ruído. Foi observado que o custo computacional introduzido com as operações da nova estratégia se torna desprezível perto da complexidade do algoritmo *backpropagation*. Nesse contexto, a adoção da nova estratégia se torna vantajosa, em especial, quando parte dos nós têm dados contaminados por ruído.

Referências

[1] G. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks: The state of the art," *Int. J. Forecast.*, vol. 14, pp. 35–62, 1998.

[2] B. Macukow, "Neural networks – state of art, brief history, basic models and architecture," in *Comput. Inf. Syst. and Ind. Manag.*, Springer, 2016, pp. 3–14.

[3] X. Liu *et al.*, "Learning in high-dimensional multimedia data: the state of the art," *Multimedia Syst.*, vol. 23, pp. 303–313, 2017.

[4] S. Haykin, *Neural networks and learning machines*, Pearson Education, Upper Saddle River, 3 edition, 2009.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[6] R. Taylor, D. Baron, and D. Schmidt, "The world in 2025 - predictions for the next ten years," in *Int. Microsyst., Packag., Assembly and Circuits Technol. Conf.*, 2015, pp. 192–195.

[7] M. M. Najafabadi *et al.*, "Deep learning applications and challenges in big data analytics," *J. Big Data*, vol. 2, pp. 1–21, 2015.

[8] S. Wiedemann, K. R. Müller, and W. Samek, "Compact and computationally efficient representation of deep neural networks," *IEEE Trans. Neural Netw. and Learn. Syst.*, vol. 31, pp. 772–785, 2019.

[9] B. Liu and Z. Ding, "Distributed heuristic adaptive neural networks with variance reduction in switching graphs," *IEEE Trans. Cybern.*, vol. 51, pp. 3836–3844, 2019.

[10] B. Liu, Z. Ding, and C. Lv, "Distributed training for multi-layer neural networks by consensus," *IEEE Trans. Neural Netw. and Learn. Syst.*, vol. 31, pp. 1771–1778, 2020.

[11] F. Sattler *et al.*, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Trans. Neural Netw. and Learn. Syst.*, vol. 31, pp. 3400–3413, 2020.

[12] Z. K. Taha *et al.*, "A survey of federated learning from data perspective in the healthcare domain: Challenges, methods, and future directions," *IEEE Access*, vol. 11, pp. 45711–45735, 2023.

[13] T. Zhang *et al.*, "Federated learning for the internet of things: Applications, challenges, and opportunities," *IEEE IoT Mag.*, vol. 5, pp. 24–29, 2022.

[14] S. Banabilah *et al.*, "Federated learning review: Fundamentals, enabling technologies, and future applications," *Inf. Process. & Manag.*, vol. 59, pp. 103061, 2022.

[15] F. Zerka *et al.*, "Systematic review of privacy-preserving distributed machine learning from federated databases in health care," *JCO Clinical Cancer Informatics*, vol. 4, pp. 184–200, 2020.

[16] L. Li *et al.*, "A review of applications in federated learning," *Comput. & Ind. Eng.*, vol. 149, pp. 106854, 2020.

[17] L. Gamballi, D. G. Tiglea, R. Candido, and M. T. M. Silva, "Redes mlp distribuídas para classificação de arritmias cardíacas," in *Anais do SBrT*, Santa Rita do Sapucaí, MG, 2022.

[18] Y. Ho and S. Wookey, "The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling," *IEEE Access*, vol. 8, pp. 4806–4813, 2020.

[19] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Syst. & Control Lett.*, vol. 53, pp. 65–78, 2004.

[20] K. Tam *et al.* "Federated noisy client learning," *IEEE Trans. Neural Netw. and Learn. Syst.*, 2023.

[21] A. Krizhevsky, "Learning multiple layers of features from tiny images," MSc. Thesis, Univ. Toronto, 2009.

[22] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th AISTATS*, Y. W. Teh and M. Titterington, Eds., 2010, vol. 9 of *Proc. Machine Learning Research*, pp. 249–256.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.