Análise Comparativa das Arquiteturas de Rajpurkar *et al.* e Ribeiro *et al.* para Classificação de Patologias Cardíacas em Eletrocardiogramas

Sarah Morgana Meurer, Daniel Gomes de Pinho Zanco, Eduardo Vinícius Kuhn e Ranniery Maia

***Resumo*— Este trabalho foca na avaliação de desempenho de duas arquiteturas relevantes introduzidas para detecção e classificação de patologias cardíacas. Tais arquiteturas são implementadas aqui em linguagem Python e treinadas com um conjunto de dados público. O modelo obtido da arquitetura de Rajpurkar *et al.* obteve uma precisão de 0,79, sensibilidade de 0,70 e F1-*score* de 0,72, enquanto o modelo obtido da arquitetura de Ribeiro *et at.* alcançou uma precisão de 0,81, sensibilidade de 0,70 e F1-*score* de 0,74. Os resultados indicam que ambos modelos exibem desempenho similar quando comparados usando um conjunto de dados padronizado.**

***Palavras-Chave*— Aprendizado profundo, classificação multirrótulos, ECG, redes neurais.**

***Abstract*— This research work focuses on assessing the performance of two relevant architectures introduced for detecting and classifying cardiac pathologies. Such architectures are implemented here using Python language and trained with a public dataset. The model obtained from the architecture of Rajpurkar *et al.* reached a precision of 0.79, sensitivity of 0.70, and F1-*score* of 0.72, while the model obtained from the architecture of Ribeiro *et al.* achieved a precision of 0.81, sensitivity of 0.70, and F1-*score* of 0.74. The results indicate that both models exhibit similar performance when compared using a standardized dataset.**

***Keywords*— Deep learning, multi-label classification, ECG, neural networks.**

I. Introdução

No decorrer das últimas décadas, doenças cardiovasculares vêm figurando dentre as principais causas de morte registradas no mundo. Estima-se que, apenas em 2019, 17,9 milhões de pessoas morreram por decorrência de doenças cardiovasculares, representando 32% das mortes registradas ao redor do mundo [1]. Essas doenças, muitas vezes, podem ser detectadas e diagnosticadas por um profissional de saúde (por exemplo, um médico cardiologista) através de um exame cardíaco não invasivo, denominado eletrocardiograma (ECG). O ECG consiste em uma forma de representação da atividade elétrica do coração em um dado intervalo, contendo assim informações essenciais para a análise do ritmo cardíaco bem como da morfologia dos batimentos [2]. Todavia, a interpretação de sinais de ECG não é uma tarefa trivial e depende da experiência do profissional de saúde para identificar (visualmente) alterações de padrão [3–5]. Tal dificuldade na interpretação se deve, sobretudo, à semelhança de morfologias encontradas em diferentes condições cardíacas ou ainda por fadiga humana, o que pode resultar em importantes erros de diagnóstico [6–8]. Vale destacar que até 33% das interpretações de ECG podem conter erros de diagnóstico de grande relevância [6].

Visando auxiliar os profissionais de saúde e melhorar a frequência de acertos dos diagnósticos das patologias cardíacas, ferramentas computacionais vêm sendo continuamente aprimoradas e cada vez mais utilizadas em análises clínicas para superar as dificuldades existentes [9]. Em particular, na interpretação de sinais de ECG, técnicas de aprendizado de máquina (especificamente, aprendizado profundo) vêm sendo amplamente empregadas para construir modelos capazes de detectar automaticamente padrões e identificar alterações morfológicas [9, 10]. Isso se deve, especialmente, aos avanços tecnológicos ocorridos nos últimos anos, possibilitando a concepção e implementação de arquiteturas de redes neurais mais complexas, assim como aos grandes conjuntos de dados disponíveis, obtidos a partir de registros eletrônicos de saúde [9]. Como resultado, um importante número de trabalhos de pesquisa [11–16], utilizando diferentes arquiteturas, pode ser encontrado na literatura versando sobre classificação de patologias cardíacas (veja [8] para detalhes). Contudo, como os conjuntos de dados utilizados em alguns desses importantes trabalhos não são públicos, torna-se difícil validar os resultados obtidos, comparar o desempenho, e/ou propor melhorias.

Neste contexto, considerando a relevância das arquiteturas apresentadas em [11] e [16], o presente trabalho de pesquisa tem por objetivo:

i) implementar a arquitetura de [11] adaptada para o caso multirrótulo assim como aquela descrita em [16], utilizando a linguagem Python [17], juntamente com as bibliotecas Tensorflow [18] e Keras [19];
ii) realizar o treinamento das arquiteturas implementadas, considerando (exclusivamente) o conjunto de dados, público e já rotulado por cardiologistas, disponível em [20]; e
iii) avaliar o desempenho dos modelos obtidos por meio de métricas de avaliação adequadas para o caso de classificação multirrótulo, conforme metodologia descrita recentemente em [21], frente a um conjunto de dados padronizado.

Sarah Morgana Meurer e Eduardo Vinícius Kuhn estão vinculados ao LAPSE–Laboratório de Processamento de Sinais e Eletrônica do Departamento de Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná (UTFPR), Toledo, PR, Brasil (e-mails: sarah_morgana18@hotmail.com e kuhn@utfpr.edu.br).

Daniel Gomes de Pinho Zanco está vinculado ao LINSE–Laboratório de Circuitos e Processamento de Sinais do Departamento de Engenharia Elétrica e Eletrônica da Universidade Federal de Santa Catarina, Florianópolis, SC, Brasil (e-mail: dangpzanco@gmail.com).

Ranniery Maia está vinculado ao Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte (UFRN), Natal, RN, Brasil (e-mail: rmaia@dimap.ufrn.br).

II. Formulação do Problema

Nesta seção, uma descrição do conjunto de dados utilizado é apresentada; em seguida, as arquiteturas consideradas aqui para a detecção e classificação de patologias cardíacas são brevemente revisitadas.

A. Conjunto de dados considerado

O conjunto de dados considerado, disponível em [20], consiste de um grande número de registros de sinais de ECG. Esse conjunto público (denominado PTB-XL) é composto por 21.837 registros de sinais de ECG, de 12 derivações, com 10 segundos de duração, gravados com 18.885 pacientes (52% homens e 48% mulheres), bem como frequência de amostragem de 100 Hz (usada aqui) e 500 Hz. Os dados são compostos por sinais de ECG rotulados por cardiologistas e por metadados contendo declarações de diagnóstico de acordo com o padrão descrito na Norma ISO 11073-91064:2009 [22], podendo as 44 declarações de diagnósticos possíveis serem agrupadas (conforme indicado em [20, Table 5]) em 5 superclasses distintas, a saber: normal (NORM), alteração do segmentos ST e onda T (STTC), distúrbios de condução (CD), infarto do miocárdio (MI) e hipertrofia (HYP). Tais registros, denominados exemplos, são divididos em três conjuntos, a saber: treinamento, validação e teste, seguindo a recomendação dada em [20]; especificamente, os *folds* de 1 a 8 são utilizados para o treinamento, o *fold* 9 é utilizado para a validação e o *fold* 10 é utilizado para o teste. Exemplos que não possuem declarações de diagnóstico são removidos do conjunto de dados. A partir dos exemplos de sinais de ECG com diagnóstico, a rotulagem considerada para o treinamento é realizada de acordo com a probabilidade de diagnóstico constante nos metadados, isto é, declarações de diagnóstico com probabilidade maior que 50% estabelecem a superclasse correspondente como um possível rótulo do exemplo. Cada exemplo pode conter mais de uma declaração de diagnóstico com probabilidade maior que 50%, dado que diferentes patologias podem ocorrer de forma simultânea; por isso, os rótulos de cada exemplo são unidos de forma a produzir um conjunto de dados multirrótulos. Dessa forma, o conjunto de treinamento conta com 20.631 rótulos, o conjunto de validação conta com 2.575 rótulos e o conjunto de teste conta com 2.593 rótulos.

B. Revisitando a arquitetura de Rajpurkar et al. [11]

A arquitetura de Rajpurkar *et al.* (ilustrada em [11, Fig. 2]) consiste de 34 camadas (33 camadas convolucionais e 1 camada totalmente conectada), organizadas em 1 bloco de entrada, 16 blocos residuais e 1 bloco de saída. A composição dos blocos é dada como segue. O bloco de entrada é constituído por uma camada convolucional, seguida por *batch normalization* e função de ativação ReLU. Os blocos residuais contêm uma sequência de camadas convolucional, *batch normalization*, ReLU e *dropout*, além de conexões de atalho formadas por uma camada *max pooling*, utilizada para adequar a dimensão da entrada de cada bloco. Modificações ocorrem em 15 desses blocos residuais, adicionando camadas de *batch normalization*, ReLU e *dropout* ao bloco residual inicial. Por fim, o bloco de saída conta com camadas de *batch normalization*, ReLU, *dense* e função de ativação *sigmoid* (usada para acomodar o caso de classificação multirrótulos [23] considerado aqui). Na composição das camadas convolucionais, o número de filtros aumenta a cada 4 blocos residuais, alternando entre os valores 64, 128, 192 e 256. Esses filtros possuem, para todas as camadas convolucionais, tamanho igual a 16. Ainda, os valores dos passos (isto é, *stride*) são determinados de forma a subamostrar a entrada em um fator de 2 a cada 2 blocos residuais. Desse modo, a entrada original é subamostrada por um fator de 2^8 ao final da arquitetura. Os coeficientes das diferentes camadas são inicializados conforme [24]. Vale destacar que, segundo os autores, o modelo obtém precisão, sensibilidade e F1-*score* de 80,9%, 82,7% e 80,9%, respectivamente.

C. Revisitando a arquitetura de Ribeiro et al. [16]

A arquitetura de Ribeiro *et al.* (ilustrada em [16, Fig. 3]) contém 10 camadas (9 camadas convolucionais e 1 camada totalmente conectada), organizadas em 1 bloco de entrada, 4 blocos residuais e 1 bloco de saída. Tais blocos podem ser descritos como segue. O bloco de entrada consiste de 1 camada convolucional, seguida de *batch normalization* e ReLU. Cada bloco residual é composto por camadas convolucional, *batch normalization*, ReLU e *dropout*. Uma conexão de atalho (denominada conexão A) contendo camadas *max pooling* e convolucional é adicionada à segunda camada convolucional e outra conexão de atalho (denominada conexão B), com uma função identidade, parte dessa adição. No primeiro bloco residual, a conexão A tem como entrada a saída do bloco de entrada da arquitetura, enquanto que nos outros blocos residuais, a entrada da conexão A consiste na saída da conexão B do bloco residual anterior. Por consequência, em cada bloco residual, a saída da conexão B equivale à entrada da conexão A do próximo bloco residual, com exceção do quarto bloco em que a conexão de atalho B não se faz presente. Por fim, o bloco de saída é composto por uma camada totalmente conectada, seguida da função de ativação *sigmoid*. O número de filtros é alterado no decorrer das camadas convolucionais, de modo que na primeira camada convolucional o número de filtros equivale a 64 e nas camadas posteriores, o número de filtros é acrescido de 64 a cada 2 blocos residuais. Esses filtros, para todas as camadas convolucionais, possuem um tamanho igual a 16. Por outro lado, as camadas *max pooling* e convolucional que compõem a primeira conexão de atalho dos blocos residuais possuem filtros com tamanho igual a 1, visando adequar a dimensão dos dados. Ainda, o passo (*stride*) é determinado de forma a subamostrar a entrada em um fator de 4 a cada bloco residual. Os coeficientes das diferentes camadas da rede são inicializados conforme [24]. O desempenho alcançado, segundo os autores, está acima de 80% para o F1-*score* e acima de 99% para a especificidade.

III. Implementação e Avaliação

Nesta seção, alguns detalhes sobre a implementação e treinamento dos modelos, e ajuste de hiperparâmetros são primeiramente apresentados; em seguida, a metodologia de

avaliação é revisitada visando especialmente o caso de classificação multirrótulos.

A. Implementação, treinamento e ajuste de hiperparâmetros

Quanto aos recursos de *software* necessários na implementação das arquiteturas de [11] e [16], considera-se aqui o uso da linguagem Python (versão 3.9.15) juntamente das bibliotecas Keras / TensorFlow (versão 2.10.0) e Scikit-learn (versão 1.1.3), dentre outras. A IDE Visual Studio Code (versão 1.73.1) foi utilizada para o desenvolvimento e teste do código[1]. Quanto aos recursos computacionais, o treinamento dos modelos se deu em um computador com processador AMD Ryzen 5800x, 32 GB de RAM e 1 GPU NVIDIA RTX 3080, no sistema operacional Ubuntu 22.04. Ainda, com respeito ao processo de treinamento, a otimização da função custo (entropia cruzada binária) é realizada através do otimizador Adam [25] com uma taxa de aprendizagem de 0,1. Essa taxa de aprendizagem é reduzida por um fator de 2 se o custo de validação não apresentar melhora nas últimas 10 épocas. Os modelos são treinados por no máximo 100 épocas com um tamanho de lote (*batch size*) de 256 exemplos, sendo o treinamento interrompido caso o custo no conjunto de validação não apresente melhora nas últimas 15 épocas. Uma taxa de *dropout* de 0,8 é utilizada aqui. Vale salientar que a adequação da taxa de aprendizagem e a interrupção do treinamento são realizadas, respectivamente, pelos *callbacks* `ReduceLROnPlateau` e `EarlyStopping` disponíveis na biblioteca Keras [19].

B. Metodologia de avaliação

Visando avaliar o desempenho das arquiteturas implementadas, a metodologia introduzida recentemente em [21] para o caso de classificação multirrótulo é considerada aqui, diferindo assim da abordagem convencional utilizada até então na literatura. A partir dessa metodologia, é construída uma matriz de confusão expandida $\mathbf{M}$ (veja a Fig. 1 adiante), de dimensão $(C+1)\times(C+1)$ em que C representa o número de classes, a qual é denominada matriz de confusão multirrótulos (MLCM). Nessa matriz $\mathbf{M}$, a linha $C+1$ corresponde à inexistência de rótulo tanto verdadeiro quanto predito (*no true label*, NTL) para um dado exemplo, enquanto a coluna $C+1$ corresponde à situação em que um ou mais rótulos verdadeiros do exemplo não são preditos (*no predicted label*, NPL). A partir de tal matriz, torna-se possível determinar o número de ocorrências verdadeiro positivo (TP), verdadeiro negativo (TN), falso positivo (FP) e falso negativo (FN) para o caso multirrótulo [21, Eqs. (15)-(18)]; consequentemente, as métricas precisão, sensibilidade e F1-*score*, que relacionam o número de ocorrências TP, TN, FP e FN para uma dada classe, assim como seus valores médios *micro*, *macro* e *weighted* podem ser computados de forma precisa [21, Eqs. (2)-(7)]. Dessa forma, definida a metodologia de avaliação, pode-se proceder para a análise e discussão dos resultados.

[1]O código fonte desenvolvido se encontra disponível em https://github.com/SarahMeurer/ECG-classification.

IV. Resultados e Discussão

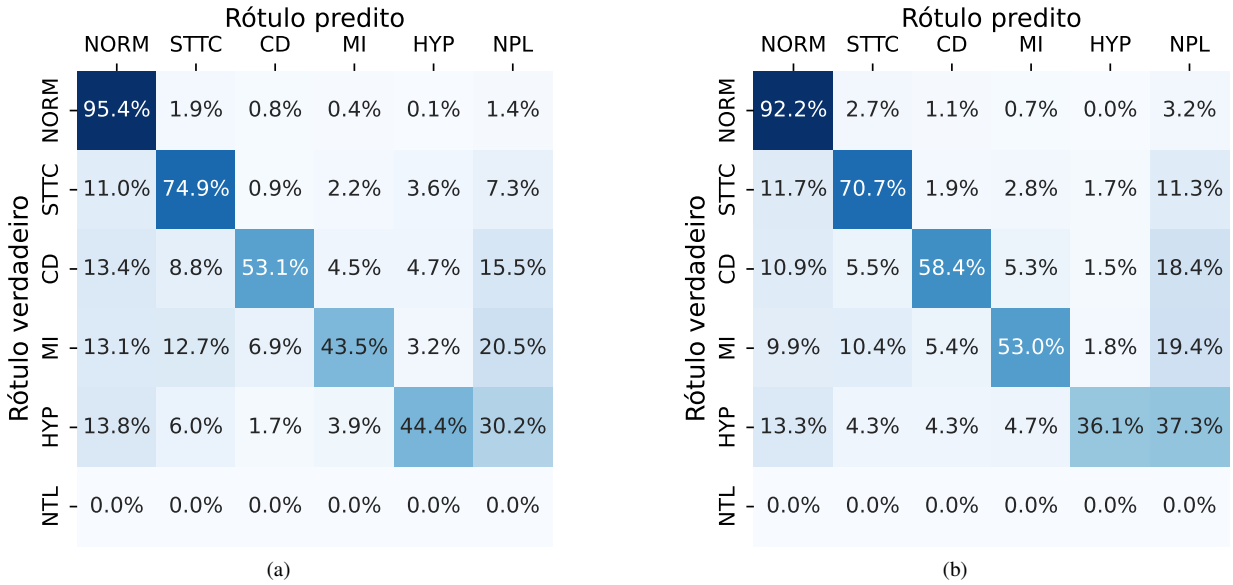
Nesta seção, algumas características observadas a partir da análise da MLCM são descritas. Em seguida, os resultados verificados através das métricas de desempenho são discutidos. Por fim, alguns aspectos afeitos ao custo de treinamento são apresentados.

A. Análise a partir da matriz de confusão multirrótulo

A Fig. 1 apresenta as matrizes de confusão multirrótulos obtidas a partir das predições do conjunto de teste com os modelos resultantes. Tais matrizes são normalizadas com base no número total de ocorrências TP e FN correspondente a cada classe; especificamente, a Fig. 1(a) ilustra a MLCM obtida para o modelo decorrente da arquitetura de [11] e a Fig. 1(b) para o modelo de [16]. Observa-se a partir dos valores dispostos sobre a diagonal principal das matrizes que as classes com mais rótulos estabelecidos (isto é, maior número de exemplos) exibem um desempenho superior na predição das diferentes condições cardíacas, conforme constatado, por exemplo, com a classe NORM em que mais de 90% dos rótulos são preditos corretamente. Por outro lado, classes que possuem menor número de rótulos estabelecidos apresentam (proporcionalmente) menos predições corretas (independente do modelo utilizado), como é o caso da classe HYP que teve apenas 44,4% das predições corretas usando o modelo de [11] e 36,1% através do modelo de [16]. Isso ratifica a necessidade de se investigar metodologias para o tratamento do desbalanceamento entre as classes. Também, verifica-se que mesmo usando a arquitetura de [11], que possui um número significativamente maior de camadas, a capacidade do modelo de predizer corretamente rótulos pertencentes às classes CD e MI é inferior à capacidade alcançada pelo modelo de [16]. Ainda, é possível constatar que, quando ocorre uma predição errônea de um dado exemplo, ambos os modelos tendem a atribuir o rótulo NORM para aquele dado exemplo (como se constata ao longo das linhas da primeira coluna) ou a não realizar predição alguma daquele rótulo (como indicado nas linhas da última coluna intitulada NPL); na verdade, grande parte dos rótulos preditos incorretamente são decorrentes da situação em que aquele dado rótulo não é sequer predito, o que se torna mais evidente quando se tem um menor número de exemplos. Vale mencionar que a última linha (intitulada NTL) se encontra zerada dado que, no tratamento do conjunto de dados (descrito na Seção II-A), os exemplos não rotulados foram removidos. Portanto, salvo pequenas diferenças, os modelos gerados pelas arquiteturas de [11] e de [16] apresentam desempenho similar na tarefa considerada.

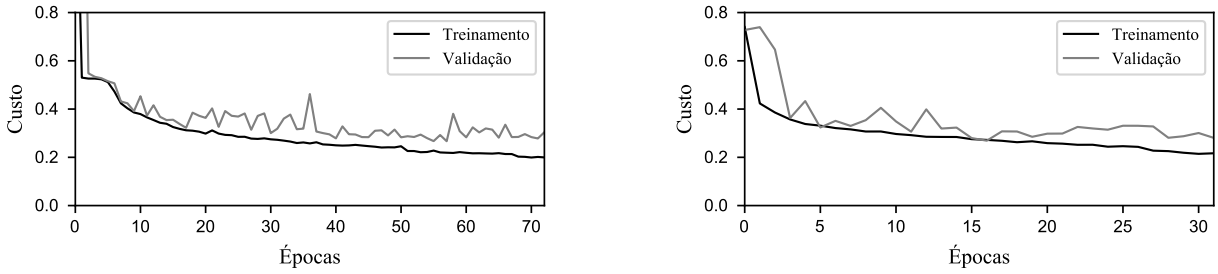
B. Avaliação de desempenho frente às métricas consideradas

A Tabela I apresenta uma análise comparativa do desempenho dos modelos, para as diferentes classes, a partir das métricas precisão, sensibilidade e F1-*score* bem como dos valores médios *micro*, *macro* e *weighted* (ponderada pela soma das ocorrências TP, FN, NTL e NPL da classe). Dessa tabela, verifica-se que, com exceção de algumas classes específicas (devidamente destacadas para facilitar a interpretação), o modelo de [16] exibe um desempenho levemente superior ao



(a) (b)

Fig. 1. Matriz de confusão multirrótulos. (a) Arquitetura de Rajpurkar *et al.* [11]. (b) Arquitetura de Ribeiro *et al.* [16].



(a) (b)

Fig. 2. Evolução do custo sobre os conjuntos de treinamento e de validação. (a) Arquitetura de Rajpurkar *et al.* [11]. (b) Arquitetura de Ribeiro *et al.* [16].

alcançado pelo modelo de [11]; sobretudo, frente às métricas médias *micro*, *macro* e *weighted*. Vale destacar que os resultados obtidos não devem ser diretamente comparados com aqueles descritos em [11] e [16], devido, por exemplo, i) a diferença na tarefa de classificação considerada; ii) a metodologia utilizada para computar as métricas; e/ou iii) ao conjunto de dados usado. Todavia, ainda assim, os resultados atingidos pelos modelos obtidos aqui (a partir do conjunto de dados público) são próximos dos fornecidos por [11] e [16]; em particular, [11] indica ter atingido uma precisão de 0,81 (contra 0,79), uma sensibilidade de 0,82 (contra 0,70) e um F1-*score* de 0,81 (contra 0,72), enquanto [16] indica ter atingido um F1-*score* maior do que 0,80 (contra 0,74). Diante disso, infere-se que ambas as arquiteturas foram implementadas e treinadas com sucesso, produzindo assim modelos com resultados compatíveis aos da literatura.

C. Sobre o custo de treinamento e de validação

A Fig. 2 apresenta a evolução do custo sobre os conjuntos de treinamento e de validação para as arquiteturas de [11] e de [16]. A partir de tal figura, verifica-se custo elevado no conjunto de treinamento e validação no início do processo de treinamento, o qual decresce gradualmente até atingir a condição de parada (isto é, quando não se observa melhoria no custo sobre o conjunto de validação após 15 épocas). Essa condição de interrupção do treinamento ocorre em 73 épocas na arquitetura de [11] contra apenas 32 épocas na arquitetura de [16], o que se deve especialmente à diferença (de quase 3 vezes) na quantidade de camadas (ou número de parâmetros) treináveis. Ainda, pode-se inferir a partir das curvas apresentadas a ocorrência de um *overfitting* pouco significativo, devido à interrupção precoce do treinamento de ambas as arquiteturas. Consequentemente, conclui-se que os modelos gerados a partir das arquiteturas de [11] e de [16] apresentam uma boa capacidade de generalização na classificação de patologias cardíacas em exemplos não vistos até então (novos).

V. Considerações Finais

Neste trabalho, a implementação das arquiteturas de Rajpurkar *et al.* [11] e Ribeiro *et al.* [16] foi realizada visando a detecção e classificação multirrótulo de patologias cardíacas a partir de sinais de ECG. Tais arquiteturas foram treinadas usando o conjunto de dados público, disponibilizado por [20]. O desempenho dos modelos foi avaliado por meio de uma metodologia padronizada, introduzida recentemente em [21], para o caso de classificação multirrótulos. A partir disso, observou-se que ambas arquiteturas apresentaram um bom desempenho na tarefa, atingindo uma precisão média de 80%, sensibilidade média de 70% e F1-*score* médio acima de 70%. Vale destacar

TABELA I
DESEMPENHO DAS ARQUITETURAS DE RAJPURKAR *et al.* [11] E DE RIBEIRO *et al.* [16] FRENTE ÀS MÉTRICAS CONSIDERADAS.

	Precisão		**Sensibilidade**		**F1-*score***		***Weight***	
	Rajpurkar	Ribeiro	Rajpurkar	Ribeiro	Rajpurkar	Ribeiro	Rajpurkar	Ribeiro
NORM	0,80	0,82	0,95	0,92	0,87	0,87	961	964
STTC	0,74	0,77	0,75	0,71	0,74	0,74	534	540
CD	0,86	0,85	0,53	0,58	0,66	0,69	580	543
MI	0,80	0,79	0,44	0,53	0,56	0,64	464	443
HYP	0,62	0,77	0,44	0,36	0,52	0,49	232	233
Micro avg	0,70	0,70	0,70	0,70	0,70	0,70	2771	2723
Macro avg	0,76	0,80	0,62	0,62	0,67	0,69	2771	2723
Weighted avg	0,79	0,81	0,70	0,70	0,72	0,74	2771	2723

que o modelo de [16] exibiu um desempenho similar, ou ainda ligeiramente melhor, para as diferentes métricas consideradas; logo, o significativo número de parâmetros da arquitetura de [11] não se traduziu em acréscimo expressivo de desempenho na tarefa considerada. Diante disso, concluiu-se que a arquitetura de [16] deve ser preterida; sobretudo, em dispositivos com baixa capacidade de processamento. Sugestões de trabalhos futuros incluem investigar metodologias para tratamento do desbalanceamento dentre as classes, estender as comparações para outras arquiteturas, assim como avaliar o desempenho na classificação das 24 subclasses de patologias presentes no conjunto de dados.

AGRADECIMENTOS

Os autores agradecem aos revisores pelas sugestões valiosas e comentários construtivos que visaram o aprimoramento deste trabalho de pesquisa.

REFERÊNCIAS

[1] World Health Organization (WHO). (2021) Cardiovascular diseases (CVDs). [Online]. Available: https://www.who.int/en/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)

[2] G. Sannino and G. D. Pietro, "A deep learning approach for ECG-based heartbeat classification for arrhythmia detection," *Future Generation Comput. Syst.*, vol. 86, pp. 446–455, 2018.

[3] S. M. Salerno, P. C. Alguire, H. S. Waxman, and A. C. of Physicians, "Training and competency evaluation for interpretation of 12-lead electrocardiograms: recommendations from the american college of physicians," *Ann. Int. Med.*, vol. 138, pp. 747–750, 2003.

[4] P. Mele, "Improving electrocardiogram interpretation in the clinical setting," *J. Electrocardiology*, vol. 41, pp. 438–439, 2008.

[5] A. W. Cairns *et al.*, "A computer-human interaction model to improve the diagnostic accuracy and clinical decision-making during 12-lead electrocardiogram interpretation," *J. Biomed. Inform.*, vol. 64, pp. 93–107, 2016.

[6] S. M. Salerno, P. C. Alguire, and H. S. Waxman, "Competency in interpretation of 12-lead electrocardiograms: A summary and appraisal of published evidence," *Ann. Int. Med.*, vol. 138, pp. 751–760, 2003.

[7] U. R. Acharya *et al.*, "A deep convolutional neural network model to classify heartbeats," *Comput. Biol. Med.*, vol. 89, pp. 389–396, 2017.

[8] S. Somani *et al.*, "Deep learning and the electrocardiogram: A review of the current state-of-the-art," *Eur. Soc. Cardiology*, vol. 23, pp. 1179–1191, 2021.

[9] A. Mincholé, J. Camps, A. Lyon, and B. Rodríguez, "Machine learning in the electrocardiogram," *J. Electrocardiology*, vol. 57, pp. S61–S64, 2019.

[10] K. Shameer *et al.*, "Machine learning in cardiovascular medicine: Are we there yet?" *Heart*, pp. 1–9, 2018.

[11] P. Rajpurkar *et al.*, "Cardiologist-level arrhythmia detection with convolutional neural networks," *ArXiv*, vol. abs/1707.01836, 2017.

[12] Z. Xiong, M. K. Stiles, and J. Zhao, "Robust ECG signal classification for detection of atrial fibrillation using a novel neural network," *Compu. Cardiology*, pp. 1–4, 2017.

[13] C. D. Galloway *et al.*, "Development and validation of a deep-learning model to screen for hyperkalemia from the eletrocardiogram," *JAMA Cardiology*, vol. 5, pp. 428–436, 2019.

[14] Z. I. Attia *et al.*, "An artificial intelligence-enabled ECG algorithm for the identification of patients with atrial fibrillation during sinus rhythm: A retrospective analysis of outcome prediction," *Lancet*, vol. 394, pp. 861–867, 2019.

[15] R. R. van de Leur *et al.*, "Automatic triage of 12-lead ECGs using deep convolutional neural networks," *J. Amer. Heart Assoc.*, 2020.

[16] A. H. Ribeiro *et al.*, "Automatic diagnosis of the 12-lead ECG using a deep neural network," *Nature Commun.*, vol. 11, 2020.

[17] G. Van Rossum and F. L. Drake, *Python 3 reference manual*. Scotts Valley, CA: CreateSpace, 2009.

[18] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: http://tensorflow.org/

[19] F. Chollet *et al.* (2015) Keras. [Online]. Available: https://github.com/fchollet/keras

[20] P. Wagner *et al.*, "PTB-XL, a large publicly available electrocardiography dataset," *Sci Data*, vol. 7, 2020.

[21] M. Heydarian, T. E. Doyle, and R. Samavi, "MLCM: Multi-label confusion matrix," *IEEE Access*, vol. 10, pp. 19 083–19 095, 2022.

[22] ISO Central Secretary, "Health informatics – Standard communication protocol – Part 91064: Computer-assisted electrocardiography," International Organization for Standardization, Geneva, CH, International Standard ISO 11073-91064:2009, 2009.

[23] A. Maxwell *et al.*, "Deep learning architectures for multi-label classification of intelligent health risk prediction," *Bioinformatics*, vol. 18, 2017.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," *arXiv*, 2015.

[25] D. P. Kingma and J. L. Ba, "Adam: a method for stochastic optimization," *arXiv*, 2017.