Modelos de Classificação de Imagens Empregando Projeto Automático de Redes Neurais Convolucionais com Algoritmos Evolucionários

Antony Kevin Ferreira Santos
Universidade Estadual do Ceará (UECE)
Fortaleza-CE, Brasil
antony.kevin@aluno.uece.br

Gustavo Augusto Lima de Campos
Universidade Estadual do Ceará (UECE)
Fortaleza-CE, Brasil
gustavo.campos@uece.br

Thelmo Pontes de Araújo
Universidade Estadual do Ceará (UECE)
Fortaleza-CE, Brasil
thelmo.araujo@uece.br

Ana Luiza Bessa de Paula Barros
Universidade Estadual do Ceará (UECE)
Fortaleza-CE, Brasil
analuiza.barros@uece.br

Thalyson Gomes Nepomuceno da Silva
Universidade Estadual do Ceará (UECE)
Fortaleza-CE, Brasil
thalyson.uece@gmail.com

***Abstract*—This study evaluated two evolutionary algorithms, Population-Based Incremental Learning (PBIL) and Genetic Algorithms (GA), in the automated search for Multi-Layer Perceptron (MLP) and Convolutional Neural Network (CNN) architectures applied to the problem of multiclass image classification. In the experiments, the combination of PBIL with CNN (PBIL-CNN) achieved the highest accuracy, and the GA-CNN combination showed a good balance between performance and execution time. The PBIL-MLP combination was the most effective in searching for MLP networks, being better than the GA-MLP combination. The experiments with PBIL showed more promising results, but the experiments with GA produced good results with less processing time.**

***Keywords*—Evolutionary Algorithms; Neural Networks; Image Classification.**

***Resumo*—Este estudo avaliou dois algoritmos evolucionários, o *Population-Based Incremental Learning* (PBIL) e o *Genetic Algorithms* (GA) na busca automatizada de arquiteturas de redes *Multi-Layer Perceptron* (MLP) e *Convolutional Neural Network* (CNN) para o problema de classificação multiclasse de imagens. Nos experimentos, a combinação de PBIL com CNN (PBIL-CNN) alcançou a maior acurácia, e a combinação GA-CNN apresentou bom equilíbrio entre desempenho e tempo de execução. A combinação PBIL-MLP foi a mais eficaz na busca por redes MLP, sendo melhor que a combinação GA-MLP. Os experimentos com PBIL apresentaram resultados mais promissores, mas os experimentos com GA apresentaram bons resultados com menos tempo de processamento.**

***Palavras-chave*—Algoritmos Evolucionários; Redes Neurais; Classificação de Imagens.**

I. Introdução

Entre as diversas capacidades humanas, destaca-se a habilidade de reconhecer padrões em imagens. No entanto, o uso de máquinas ainda se faz necessário para a execução de determinadas tarefas de reconhecimento, uma vez que esses dispositivos realizam tais atividades com muito mais rapidez, como ocorre na leitura de códigos de barras, na verificação da qualidade de produtos e na identificação por impressões digitais. Assim, mesmo com a superioridade humana no reconhecimento de padrões, as máquinas são consideradas essenciais para atividades cotidianas que exigem agilidade e precisão [1].

As Redes Neurais Convolucionais (*Convolutional Neural Networks* - CNNs) estão entre os melhores modelos no que diz respeito ao reconhecimento de padrões em imagens e vêm alcançando resultados consistentes em competições internacionais de classificação. Grande parte desse desempenho decorre da grande capacidade dessas redes profundas de extrair de forma eficiente características das imagens. Tal habilidade torna as CNNs especialmente relevantes na área da visão computacional, com aplicações diretas em cenários que exigem a conversão de dados visuais em decisões, como, por exemplo, em veículos autônomos e sistemas de robótica [2].

Embora existam princípios que orientam como deve ser a estruturação de arquiteturas de redes neurais ao se resolver certos problemas, essa estruturação ainda está sujeita à subjetividade humana acerca da definição de elementos como o número de camadas, quantidade de neurônios, conexões e hiperparâmetros. Por isso, torna-se relevante investir esforços em métodos que

reduzam essa dependência humana no processo de concepção. Nesse contexto, os algoritmos evolucionários surgem como uma alternativa promissora para a construção automática de redes neurais [3].

Algoritmos Evolucionários (*Evolutionary Algorithms* - EAs) baseiam-se no princípio de que, diante de recursos limitados, uma população submetida a certa pressão evolutiva tende a favorecer tanto a sobrevivência quanto a propagação dos indivíduos mais aptos. Na prática, esses algoritmos almejam simular o processo de seleção natural que se observa na natureza [4].

Os EAs têm como um de seus objetivos encontrar soluções para problemas computacionais que possuam espaços de busca muito amplos ou até infinitos. Esses algoritmos se diferenciam da busca puramente aleatória pelo fato de se basearem na simulação de um processo de seleção natural [5].

Dentre os EAs, os Algoritmos Genéticos (*Genetic Algorithms* - GA) se destacam como uma técnica de busca muito eficiente e com considerável autonomia em relação à intervenção humana, pois conseguem explorar amplamente o espaço de soluções e dessa forma convergem para resultados próximos do ideal para o problema em questão [5].

Além dos GA, o Aprendizado Incremental Baseado em População (*Population-Based Incremental Learning* - PBIL) pode ser exemplificado como uma alternativa viável para a obtenção de soluções em problemas de busca muito amplos, e que especificamente também pode ser usado na busca de topologias de redes neurais profundas.

A abordagem utilizando PBIL se propõe a simplificar o processo evolutivo tornando-o menos complexo do que é visto nos Algoritmos Genéticos tradicionais. Entretanto essa simplificação é feita de forma que não haja comprometimento na eficiência da busca. Além disso, o PBIL demonstrou ser mais efetivo do que o GA na resolução de diversos problemas, apresentando melhores resultados em termos de desempenho e de qualidade das soluções encontradas [6].

Diante da problemática de que a busca não automatizada por arquiteturas de redes neurais impõe um processo repetitivo e baseado na tentativa e erro, torna-se essencial empreender esforços na automatização da busca por arquiteturas de redes neurais, especialmente com algoritmos que proporcionem uma exploração mais eficiente e sistemática como, por exemplo, os algoritmos evolucionários.

Este estudo objetiva realizar uma análise comparativa entre os algoritmos evolucionários PBIL e GA, aplicados à busca de arquiteturas de redes neurais profundas do tipo MLP e CNN aplicadas ao reconhecimento de imagens.

II. Trabalhos Relacionados

[7] utilizou algoritmos genéticos para otimização de quatorze hiperparâmetros de uma rede CNN, especificamente para classificação do uso de máscaras faciais em três condições (sem máscara, máscara correta e máscara incorreta). Nas imagens foram usadas as técnicas de *grayscalling*, *resizing* e *normalization*, além disso os autores utilizaram *data augmentation*. Os resultados obtidos atingiram uma acurácia de 94,82% e foi concluído que usando algoritmos genéticos obteve-se melhores resultados do que usando ajustes manuais.

De forma similar, [8] apresentou em seu trabalho uma abordagem para otimizar hiperparâmetros em Redes Neurais Convolucionais usando Algoritmos Evolucionários. Os autores focaram em otimizar a taxa de aprendizado, o número de filtros, o tamanho do kernel e o número de épocas. A metodologia envolveu a aplicação de Algoritmos Genéticos para ajustar esses parâmetros em uma CNN e a abordagem foi testada usando o conjunto de dados SVHN (*Street View House Numbers*) para classificação de dígitos. A CNN resultante atingiu uma acurácia de validação de 92,31%.

[9] buscou em seu estudo otimizar a arquitetura da rede neural, a função de ativação e o algoritmo de otimização da rede convolucional. O autor utilizou uma base de dados de imagens para diagnosticar doença de Alzheimer. Neste contexto, o modelo obtido pelos autores atingiu uma acurácia de 81,74%.

O trabalho de [10] propõe um algoritmo genético de comprimento variável para a otimização eficiente de hiperparâmetros em Redes Neurais Convolucionais, pois, segundo os autores, cromossomos de tamanho fixo podem não se ajustar adequadamente à otimização de redes neurais profundas. Os hiperparâmetros buscados foram o número de canais de saída, o tamanho dos filtros, funções de ativação, tipo de *pooling*, número de camadas e presença de *skip connections* e *batch normalization*. Alcançou-se uma acurácia de 88,92% com a metodologia proposta.

Nesse contexto, a presente pesquisa se diferencia ao colocar lado a lado dois algoritmos evolucionários distintos, o PBIL e o GA, aplicados tanto a MLP quanto a CNN. Essa escolha possibilita não apenas confirmar tendências observadas na literatura, como o melhor desempenho das CNNs em relação às MLPs, mas também revelar nuances importantes sobre o *trade-off* entre acurácia e custo computacional. A avaliação do PBIL acrescenta uma contribuição ao debate, evidenciando seu potencial para alcançar bons resultados, embora com maior tempo de execução. Assim, o trabalho amplia as discussões existentes e abre espaço para novas investigações sobre a aplicabilidade do PBIL em cenários de busca automática de arquiteturas neurais.

III. Metodologia

A. Base de dados

Para a realização dos experimentos, usou-se a base de dados CIFAR-10 [11] que é composta de um total de 60.000

imagens. Dentre essas, 50.000 foram destinadas ao treinamento e validação dos modelos e as 10.000 restantes reservadas para testes. As imagens são coloridas e apresentam uma resolução de 32×32 píxeis. O conjunto de dados é composto de 10 categorias distintas. Na Figura 1, encontram-se ilustrados exemplos de imagens representativas de cada uma dessas categorias, acompanhadas de seus respectivos rótulos.

Fig. 1. Exemplo de imagens do *dataset* CIFAR-10.

Como etapa inicial, realizou-se o pré-processamento das imagens. Esse procedimento consistiu, primeiramente, na normalização dos valores dos píxeis, convertendo-os de seu intervalo original, que variava de 0 a 255, para um novo intervalo entre 0 e 1. Além disso, os rótulos associados a cada imagem foram transformados por meio da codificação *one-hot*, técnica que representa cada classe como um vetor binário, cuja dimensão corresponde ao número total de categorias existentes no *dataset*.

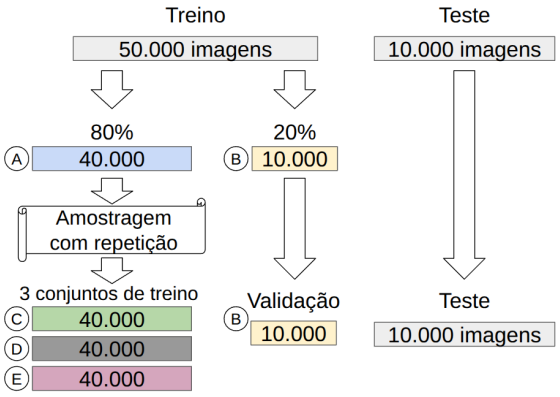

Fig. 2. Divisão da base de dados para os experimentos.

Em seguida, as 50.000 imagens destinadas ao treinamento e validação foram divididas em dois subconjuntos denominados *A* e *B*. O subconjunto *A* correspondeu a 80% do total, reunindo assim 40.000 imagens, enquanto o subconjunto *B* compreendeu os 20% restantes, totalizando 10.000 imagens. Esse processo de divisão foi conduzido de maneira estratificada, ou seja, mantendo a distribuição proporcional de amostras entre as diferentes classes, de modo que cada categoria permanecesse igualmente representada.

Por fim, o subconjunto *A* serviu como base para a criação de três novos conjuntos, nomeados *C*, *D* e *E*. Cada um desses novos conjuntos foi constituído por meio de um processo de amostragem com reposição, até que cada conjunto alcançasse novamente a quantidade de 40.000 imagens. Por outro lado, o subconjunto *B* foi reservado exclusivamente para a validação dos modelos, contendo amostras que não aparecem nos conjuntos *C*, *D* e *E*, o que assegura a independência entre os dados de validação e aqueles usados para o treinamento. A Figura 2 ilustra de maneira mais detalhada como todo o processo de divisão dos dados foi estruturado.

B. Esquema de Especificação do Cromossomo da Rede CNN

Neste trabalho, são propostas duas especificações distintas de cromossomos para a definição da arquitetura de redes neurais. A primeira especificação é direcionada para redes CNN, contemplando diversos aspectos relacionados aos conjuntos de camadas convolucionais e às camadas totalmente conectadas (*fully connected*). A segunda especificação, apresentada posteriormente, é destinada a redes MLP.

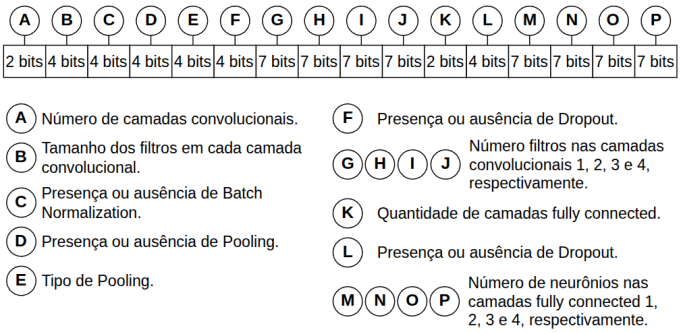

Fig. 3. Especificação do cromossomo para rede CNN.

A Figura 3 ilustra o esquema de codificação do cromossomo para redes CNN. Na parte superior dessa figura, os números dentro de retângulos indicam a quantidade de bits de cada segmento do cromossomo. Por exemplo, em Ⓐ, observa-se que o número de camadas convolucionais a ser utilizado é representado por 2 *bits*, podendo-se assim especificar entre 1 e 4 camadas (4 valores possíveis).

Cada uma dessas camadas convolucionais poderá ser complementada ou não, por camadas adicionais de *Batch Normalization*, *Pooling* e *Dropout*, cuja presença ou ausência é determinada pelos valores presentes em outros segmentos do cromossomo, mais precisamente em Ⓒ, Ⓓ e Ⓕ, respectivamente. O cromossomo completo possui um total de 84 bits.

Outros segmentos importantes do cromossomo são aqueles representados por Ⓖ, Ⓗ, Ⓘ e Ⓙ, os quais determinam, respectivamente, a quantidade de filtros a ser utilizada em cada uma das possíveis camadas convolucionais. Além disso, o segmento indicado por *K* especifica quantas camadas *fully connected* a rede terá. Para cada uma dessas camadas completamente conectadas, o segmento correspondente a Ⓛ define, de forma individual, se haverá ou não a aplicação de *Dropout* imediatamente depois da camada. Por fim, os segmentos Ⓜ, Ⓝ, Ⓞ e Ⓟ indicam a quantidade de neurônios a serem utilizados em cada uma das possíveis camadas *fully connected*.

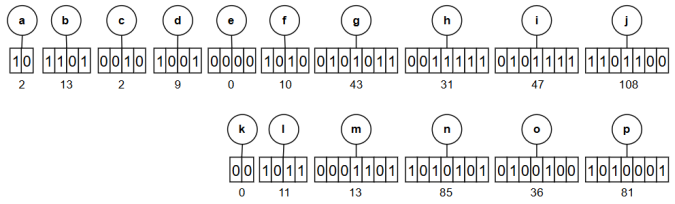

Fig. 4. Exemplo de cromossomo para rede CNN.

Na Figura 4, é apresentado um exemplo de cromossomo da especificação vista na Figura 3. Os dois primeiros *bits* (“10”), destacados em ⓐ, indicam que a rede resultante terá 3 conjuntos de camadas. Os *bits* indicados por ⓑ, correspondem aos tamanhos dos filtros utilizados em cada um dos possíveis conjuntos de camadas convolucionais. No exemplo, o valor “1” na primeira e na segunda posições do segmento ⓑ define que os dois primeiros conjuntos de camadas utilizarão filtros com dimensões de 3x3. Por outro lado, o valor “0” na terceira posição indica que o terceiro conjunto de camadas terá um filtro de 5x5. Contudo, como o exemplo possui apenas 3 camadas convolucionais, o *bit* da quarta posição torna-se irrelevante e, portanto, é desconsiderado.

No segmento representado por ⓒ, que contém a sequência de bits “0010”, é informado que existirá *Batch Normalization* apenas após a terceira camada convolucional. O valor “0” presente na quarta posição do segmento é ignorado pois teremos apenas 3 camadas. Entretanto, caso fosse considerado, indicaria ausência de *Batch Normalization* por possuir o valor zero.

Já ⓓ, que contém a sequência "1001", indica a presença da camada de *Pooling* após a primeira e a quarta camadas convolucionais, enquanto a segunda e a terceira não possuirão. Novamente, como no exemplo não possuirá quarto conjunto de camada, o último valor é ignorado. O tipo de *Pooling* é especificado pelos *bits* do segmento ⓔ representado por "0000". Quando o bit for “1”, isso indica o uso de *Average Pooling*, caso seja “0”, o *Pooling* adotado é do tipo *Max Pooling*.

Quanto ao segmento ⓕ, é codificada a presença ou ausência da camada de *Dropout*. A sequência “1010” indica que teremos *Dropout* na primeira e na terceira camadas. Os segmentos ⓖ, ⓗ, ⓘ, ⓙ, quando convertidos para valores decimais e acrescidos de 1, informam a quantidade de filtros a serem utilizados nos conjuntos de camadas convolucionais. No exemplo, os valores obtidos serão 44, 32, 48 e 109 filtros, respectivamente. Entretanto, como a rede possui apenas 3 conjuntos, o valor referente ao quarto conjunto é ignorado.

Em relação à parte *fully connected* da rede, observa-se que, conforme indicado pelo segmento ⓚ, será utilizada apenas uma camada. O segmento Ⓘ define a presença ou ausência da camada de *Dropout* nessa região. Portanto, nota-se que a primeira camada será composta de *Dropout*. Por fim, os segmentos ⓜ, ⓝ, ⓞ e ⓟ especificam a quantidade de neurônios constituintes de cada camada *fully connected*. Para o exemplo analisado, os valores correspondem a 14, 86, 37 e 81 neurônios, respectivamente. Contudo, como a configuração determina apenas uma camada, apenas o valor de 14 neurônios será considerado, enquanto os demais são descartados.

C. Esquema de Especificação do Cromossomo da Rede MLP

De maneira semelhante à especificação do cromossomo utilizada para redes CNN, porém com uma estrutura mais simples, apresenta-se na Figura 5 o esquema de codificação do cromossomo destinado às redes MLP. Na Figura 6, encontra-se um exemplo de indivíduo gerado a partir desse esquema de codificação. Neste exemplo pode-se observar que a rede MLP resultante terá 7 camadas (segmento ⓐ), *Dropout* nas camadas 1, 2, 4 e 7 (segmento ⓑ) e, respectivamente, 44, 32, 48, 109, 14, 86 e 37 neurônios nas camadas. É importante notar que a informação referente à oitava camada é desconsiderada, visto que não será utilizada na arquitetura dessa rede específica.

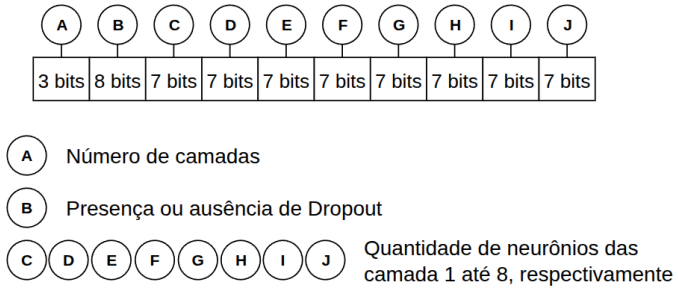

Fig. 5. Especificação do cromossomo para rede MLP.

IV. Experimentos

Os experimentos realizados neste estudo consistiram na aplicação de dois algoritmos evolucionários, PBIL e GA,

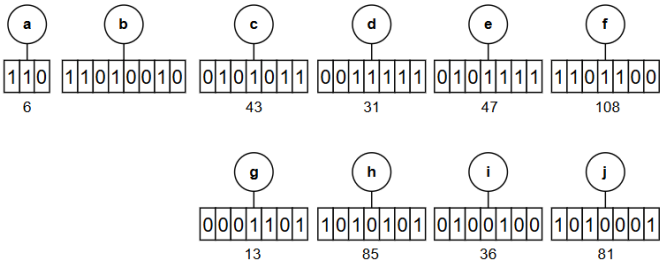

Fig. 6. Exemplo de cromossomo para MLP.

para identificar automaticamente a melhor arquitetura de redes neurais MLP e CNN na tarefa de classificação de imagens.

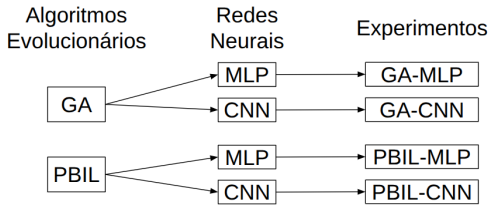

Fig. 7. Visão geral dos experimentos.

A Figura 7 apresenta uma visão geral dos quatro experimentos, cada um explorando diferentes combinações de algoritmos. A Tabela I apresenta os parâmetros, fixos e variáveis, utilizados nas redes CNN, enquanto a Tabela II exibe os parâmetros empregados nas redes MLP. Já as Tabelas III e IV detalham os parâmetros utilizados nos algoritmos GA e PBIL, respectivamente. Quaisquer outros parâmetros que por ventura não estejam explicitados foram deixados com os valores padrão dos respectivos *frameworks* utilizados nas implementações dos experimentos.

No caso específico do PBIL, segundo [12], recomenda-se adotar uma taxa de aprendizagem entre 0,05 e 0,2, uma taxa de mutação entre 0,001 e 0,02 e um valor de *mutation shift* de 0,05. Com base nessas recomendações, optou-se por utilizar uma taxa de aprendizagem de 0,2, visando acelerar a convergência do vetor de probabilidades, e uma probabilidade de mutação de 0,02, a fim de manter variabilidade suficiente nos indivíduos gerados.

Já em relação ao GA, optou-se por acompanhar a probabilidade de mutação de 0,2, o tamanho 10 da população de indivíduos e a quantidade de 15 gerações, vistos no PBIL, para uma melhor comparabilidade dos resultados. A taxa de mutação por gene (0,05) foi definida a priori como parâmetro inicial com o objetivo de avaliar o comportamento geral do algoritmo sem efetuar qualquer ajuste paramétrico específico.

TABELA I
HIPERPARÂMETROS UTILIZADOS NAS REDES NEURAIS CNN

Parâmetros das redes CNN					
Parâmetros fixos					
Convolução		Pooling		Fully-connected	
Parâmetro	Valor	Parâmetro	Valor	Parâmetro	Valor
stride	(1, 1)	pool size	(2, 2)	activation (camadas intermediárias)	relu
padding	same	padding	valid	activation (camada de saída)	softmax
activation	relu			dropout rate	0.5
dropout rate	0.5				
Parâmetros variáveis					
Convolução		Pooling		Fully-connected	
Parâmetro	Valor	Parâmetro	Valor	Parâmetro	Valor
Quantidade de filtros	1 a 128	Tipo de pooling	avg ou max	Número de neurônios	1 a 128
Tamanho do filtro	(3x3) ou (5x5)			número de camadas	1 a 8

TABELA II
HIPERPARÂMETROS UTILIZADOS NAS REDES NEURAIS MLP

Parâmetros das redes MLP			
Parâmetros fixos		**Parâmetros variáveis**	
Parâmetro	Valor	Parâmetro	Valor
activation	relu	Quantidade de Neurônios	1 a 128
dropout rate	0.5	Quantidade de camadas	1 a 8

Cada indivíduo gerado pelos algoritmos evolucionários corresponde a uma arquitetura de rede neural que é avaliada sob a forma de um *ensemble*, composto por três redes neurais com a mesma arquitetura. Cada uma dessas redes é treinada utilizando um dos três conjuntos de dados (C, D e E), conforme apresentado na Figura 2. As previsões do *ensemble*

TABELA III
PARÂMETROS DO GA

Parâmetro	**Valor**
Probabilidade de cruzamento	0.6
Probabilidade de mutação	0.02
Tamanho da população	10
Número de gerações	15
Tamanho da seleção por torneio	3
Probabilidade de mutação do gene	0.05

TABELA IV
PARÂMETROS DO PBIL

Parâmetro	Valor
Taxa de aprendizagem	0.2
Probabilidade de mutação	0.02
Tamanho da população	10
Número de gerações	15
Mutation Shift	0.05

ocorreram por meio da técnica de *soft-voting*, na qual as saídas probabilísticas das redes são agregadas para determinar a predição final. É importante destacar que, em ambos os algoritmos evolucionários, a função de avaliação, ou *fitness*, adotada nos processos evolutivos corresponde à acurácia obtida pelo *ensemble* de redes neurais.

O processo de treinamento de cada *ensemble* foi realizado ao longo de um máximo de 50 épocas, com a utilização da técnica de *Early Stopping*. Esse mecanismo interrompia o treinamento antecipadamente sempre que não fosse observada uma melhora significativa na acurácia por no mínimo 15 épocas consecutivas, evitando, assim, o *overfitting* e economizando recursos computacionais. Além disso, para o treinamento de todas as redes foi utilizado o otimizador Adam com uma taxa de aprendizagem de 0,001.

Nos experimentos, utilizou-se um computador com processador AMD Ryzen 7 7700, 32GB de memória ram DDR5 e sistema operacional Linux *Mint* 22. O trabalho foi desenvolvido na linguagem *Python* em conjunto com as bibliotecas *Tensorflow* [13] e *Keras* [14] para as redes neurais e *DEAP* [15] para os algoritmos evolucionários.

V. RESULTADOS

A Figura 8 apresenta a evolução da acurácia média das redes neurais ao longo das 15 gerações para todos os experimentos. Observa-se que, conforme as gerações avançam, a acurácia média melhora progressivamente até estabilizar, indicando que os algoritmos convergiram para soluções mais consistentes.

Ao analisar a Figura 9, que mostra a evolução da *Fitness* máxima, percebe-se que o ganho na acurácia máxima não foi tão expressivo. Houve um pequeno incremento nas gerações iniciais, seguido de uma fase de estabilidade, marcada apenas por variações discretas nos valores alcançados.

As Figuras 10 e 11 exibem as melhores arquiteturas convolucionais selecionadas pelos algoritmos evolutivos GA e PBIL, respectivamente. Por simplificação optou-se por omitir as camadas de entrada, saída e *flatten* antes da camada *fully connected*. Nota-se que a rede escolhida pelo PBIL possui

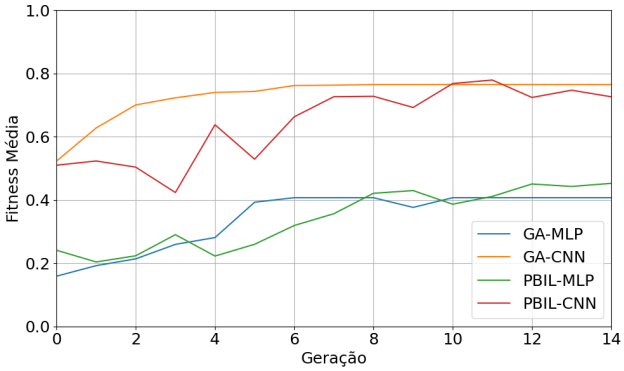

Fig. 8. *Fitness* média por geração para todos os experimentos.

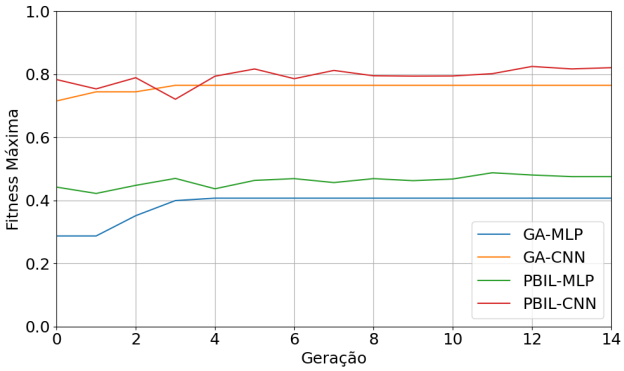

Fig. 9. *Fitness* máxima por geração para todos os experimentos.

o número máximo de camadas convolucionais permitido pelo cromossomo e também o maior número de camadas *fully connected*. Por sua vez, a rede selecionada pelo GA apresenta uma arquitetura levemente menos complexa, possuindo três camadas convolucionais e duas camadas *fully connected*. Contudo, essa menor complexidade não se mostrou vantajosa, pois houve uma queda perceptível na acurácia.

Considerando agora as redes MLP, representadas nas Tabelas V e VI observa-se que, no caso do GA, algumas camadas de *Dropout* foram selecionadas, enquanto que para o PBIL nenhuma camada de *Dropout* foi incluída. Apesar disso, o PBIL conseguiu obter um desempenho superior em termos de acurácia.

Na Tabela VII, são apresentados todos os resultados de desempenho de todos os experimentos. É possível destacar que, independentemente do tipo de rede utilizada, o PBIL superou o GA em termos de acurácia. A combinação PBIL-CNN obteve a melhor performance entre todos os experimentos, alcançando uma acurácia de 82,52%. Contudo, esse resultado demandou um tempo de execução consideravelmente longo, totalizando

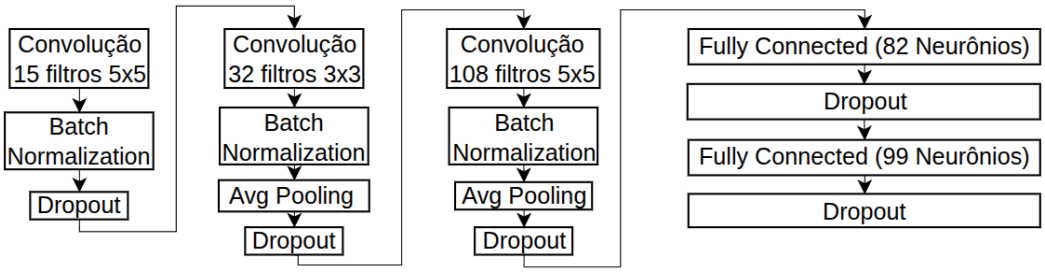

Fig. 10. Melhor rede do experimento GA-CNN.

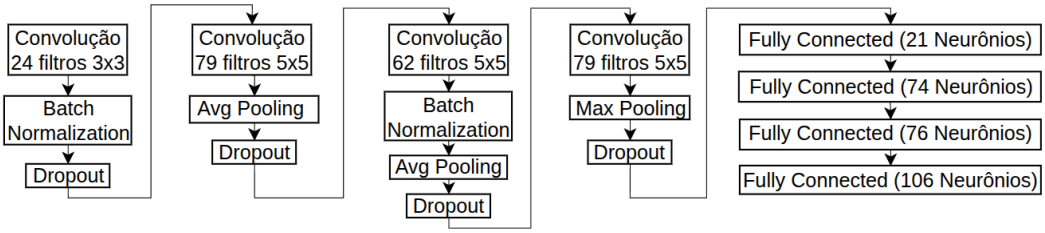

Fig. 11. Melhor rede do experimento PBIL-CNN.

TABELA V
MELHOR REDE DO EXPERIMENTO GA-MLP

Camada	Tipo de camada	Nº de neurônios
1	**Fully Connected**	114
2	**Fully Connected**	54
3	Dropout	-
4	**Fully Connected**	74
5	Dropout	-
Acurácia: *40,73%*		

TABELA VI
MELHOR REDE DO EXPERIMENTO PBIL-MLP

Camada	Tipo de camada	Nº de neurônios
1	**Fully Connected**	93
2	**Fully Connected**	57
Acurácia: *48,79%*		

TABELA VII
ACURÁCIAS PARA TODOS OS EXPERIMENTOS

Experimento	Melhor Acurácia	Duração
PBIL-CNN	**82,52%**	6d 9h 46min
GA-CNN	76,53%	14h 51min
PBIL-MLP	48,79%	1h 47min
GA-MLP	40,73%	21min

6 dias, 9 horas e 46 minutos. Em contrapartida, o experimento GA-CNN foi executado em um tempo significativamente menor, apenas 14 horas e 51 minutos, mas com uma acurácia reduzida para 76,53%.

No caso das arquiteturas MLP, tanto a acurácia quanto o tempo de execução foram inferiores em comparação às CNNs. O experimento PBIL-MLP atingiu uma acurácia de 48,79% com um tempo de execução de 1 hora e 47 minutos. Já o GA-MLP foi o mais rápido de todos, concluído em apenas 21 minutos, mas apresentou o pior desempenho, com uma acurácia de apenas 40,73%.

Esses resultados demonstram claramente a superioridade das redes convolucionais em tarefas que envolvem imagens. Além disso, reforçam a existência de um *trade-off* entre a qualidade dos resultados obtidos e o tempo necessário para executar os algoritmos de busca utilizados.

VI. CONCLUSÃO

Os resultados indicaram que para ambas as redes neurais MLP e CNN o algoritmo PBIL se sobressaiu diante do GA no que diz respeito à acurácia, entretanto, ao se observar os tempos de execução, fica evidente que houve um custo computacional mais elevado para se atingir esse melhor resultado. Comparando-se as redes neurais foi possível concluir que as redes CNN foram superiores às redes MLP para o *dataset* escolhido. A combinação entre PBIL e CNN mostra-se adequada

para cenários que exigem alta acurácia, mesmo com tempo de execução elevado. Em contrapartida, a combinação GA-CNN oferece bom equilíbrio entre desempenho e eficiência, sendo mais viável em contextos com restrições de tempo ou recursos computacionais.

Para trabalhos futuros, almeja-se a realização de experimentos adicionando outros algoritmos de busca para investigar melhoras nas redes encontradas. Além disso, também pretende-se utilizar placas de vídeo dedicadas com o intuito de oferecer melhora no tempo de processamento dos experimentos de forma que se possa explorar mais soluções. Sugere-se também a inclusão de novos hiperparâmetros das redes neurais no cromossomo para que participem da busca automatizada.

Referências

[1] R. C. Gonzalez and R. E. Woods, *Digital Image Processing, Global Edition*. Pearson Education, 2018.
[2] J. Patterson and A. Gibson, *Deep learning: A practitioner's approach*. O'Reilly Media, 2017.
[3] G. F. Miller, P. M. Todd, and S. U. Hegde, "Designing neural networks using genetic algorithms." in *ICGA*, vol. 89, 1989, pp. 379–384.
[4] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, 2nd ed., ser. Natural computing series. Springer, Jul. 2015.
[5] L. Ricardo, *Algoritmos genéticos: uma importante ferramenta da inteligência computacional*. Brasport, 2008.
[6] S. Baluja and R. Caruana, "Removing the genetics from the standard genetic algorithm," in *Machine Learning Proceedings 1995*. Elsevier, 1995, pp. 38–46.
[7] A. H. Pratomo, N. H. Cahyana, and S. N. Indrawati, "Optimizing cnn hyperparameters with genetic algorithms for face mask usage classification," *Science in Information Technology Letters*, vol. 4, no. 1, pp. 54–64, 2023.
[8] A. Al-Hyari and M. Abu-Faraj, "Hyperparameters optimization of convolutional neural networks using evolutionary algorithms," in *2022 International Conference on Emerging Trends in Computing and Engineering Applications (ETCEA)*. IEEE, 2022, pp. 1–6.
[9] S. Lee, J. Kim, H. Kang, D.-Y. Kang, and J. Park, "Genetic algorithm based deep learning neural network structure and hyperparameter optimization," *Applied Sciences*, vol. 11, no. 2, p. 744, 2021.
[10] X. Xiao, M. Yan, S. Basodi, C. Ji, and Y. Pan, "Efficient hyperparameter optimization in deep learning using a variable length genetic algorithm," *arXiv preprint arXiv:2006.12703*, 2020.
[11] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.
[12] R. Kruse, C. Borgelt, F. Klawonn, C. Moewes, M. Steinbrecher, and P. Held, *Computational Intelligence*, 2013rd ed., ser. Texts in computer science. London, England: Springer, Mar. 2013.
[13] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/
[14] F. Chollet, "Keras," https://github.com/fchollet/keras, 2015.
[15] . F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. G. ", "DEAP: Evolutionary algorithms made easy," *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, jul 2012.