Busca por Arquiteturas de Redes Neurais Artificiais Profundas Utilizando Algoritmos Genéticos

Denys Menfredy Ferreira Ribeiro[1], **Otávio Noura Teixeira**[2]

[1]Faculdade de Engenharia de Computação – Universidade Federal do Pará (UFPA)
Campus Tucuruí – Tucuruí – PA – Brazil

denys.ribeiro@tucurui.ufpa.br[1], onoura@ufpa.br[2]

Abstract. *Artificial Neural Networks are the pillars of advances in the last two decades in the field of Artificial Intelligence. However, success in training, aiming to obtain good results, strongly depends on the choice of values of a set of hyperparameters associated with their construction. The choice of these parameters is usually done empirically, wasting time and generating costs. In this work, it is proposed the development of an ENAS (Evolutionary Neural Architecture Search) algorithm using genetic algorithms as a search method to automate the process of designing architectures of convolutional neural network architectures applied to the task of classifying images from the CIFAR-10 dataset. The proposed algorithm was able to find architectures that obtained good accuracy results in the test set.*

Resumo. *As redes neurais artificiais são os pilares dos avanços das duas últimas décadas do campo de Inteligência artificial. Porém, o sucesso no treinamento visando obter bons resultados, depende fortemente da escolha dos valores de um conjunto de hiper-parâmetros associados a construção das mesmas. A escolha desses parâmetros é normalmente feita de forma empírica, desperdiçando tempo e gerando custos. Neste trabalho, é proposto o desenvolvimento de um algoritmo ENAS (Evolutionary Neural Architecture Search) utilizando algoritmos genéticos como mecanismo de busca para automatizar a construção de arquiteturas de redes neurais convolucionais aplicadas a tarefa de classificação de imagens do dataset CIFAR-10. O algoritmo proposto conseguiu encontrar arquiteturas que obtiveram bons resultados de acurácia no conjunto de teste.*

1. Introdução

O Deep Learning é uma das principais ferramentas que impulsionam as últimas inovações na área de Inteligência Artificial, os avanços no desenvolvimento de arquiteturas de redes neurais atrelado ao crescimento expressivo do volume de dados gerados atualmente, e a capacidade de processamento que temos hoje, abrem espaço para a realização de trabalhos de grande notoriedade como, [Silver et al. 2018]. O componente principal que possibilita tais avanços, são as redes neurais. As mesmas são arquiteturas que tentam simular o funcionamento do cérebro humano, sendo primeiramente definidas em 1943 por Warren McCulloch e Walter Pitts, em um artigo que descrevia um possível funcionamento dos neurônios.

Diante disso, ao longo dos anos, diversas arquiteturas de redes neurais foram surgindo, com aplicações nos mais diversos cenários. Um fator que diferencia as redes

neurais dos algoritmos tradicionais de Machine Learning, é a capacidade das mesmas de realizar o treinamento de modelos de forma fim-a-fim, ou seja, a extração dos features do conjunto de dados, é realizada pela própria arquitetura durante o treinamento, diferentemente dos algoritmos de aprendizado de máquina tradicionais que esse processo é feito antes do treinamento e geralmente de forma manual por especialistas. Neste processo de extração dos features, é realizado a redução da dimensionalidade do conjunto de dados em partes (features) que melhor representam o dado, sendo uteis para cumprir o objetivo da tarefa em questão.

As Redes Neurais Convolucionais [Le Cun et al. 1989], são arquiteturas de redes neurais amplamente utilizadas em tarefas de visão computacional. O desempenho das CNN's depende fortemente da sua arquitetura, e cada problema pode necessitar de uma arquitetura diferente. Dessa forma, para se construir um modelo de arquitetura que atinja resultados expressivos, alcançando o estado da arte, como, GoogleNet [Szegedy et al. 2014], ResNet [He et al. 2015], DenseNet [Huang et al. 2016], VGG [Simonyan and Zisserman 2014], EfficientNet [Tan and Le 2019], é necessário um grande conhecimento sobre o tema e também sobre os dados do problema em questão. Outro ponto importante nesse processo de construção, é o desperdício de tempo e recursos computacionais gerado, visto que o mesmo é realizado geralmente quase em totalidade forma empírica, ou seja, é realizado vários testes com diferentes tipos de arquiteturas e valores de parâmetros até se chegar uma arquitetura que os resultados satisfazem a necessidade do problema.

Em decorrência disto, surge-se a necessidade em se automatizar a construção de tais arquiteturas, de modo que seja possível obter-se arquiteturas otimizadas para aquela tarefa, utilizando somente os recursos necessários, além de tornar mais acessível o uso dessas arquiteturas, possibilitando que aqueles usuários que não são especialistas da área, possam também utilizar essas arquiteturas em seus problemas.

Dessa forma, **este trabalho visa desenvolver um algoritmo que otimize esse processo de construção de arquiteturas de Redes Neurais Convolucionais para a classificação de imagens do conjunto de dados CIFAR-10, utilizando Algoritmos Genéticos como mecanismo de busca**. Dentre os objetivos específicos, podemos destacar, o estudo e análise da eficiência dos algoritmos genéticos aplicados a busca por arquiteturas de redes neurais, através dos resultados obtidos responder à pergunta: **é compensatório fazer-se o uso de tal ferramenta para escolha de uma arquitetura para a realização do treinamento de redes neurais?**

Nas seções seguintes, será apresentada os trabalhos correlatos ao tema proposto, em seguida a metologia adotada, detalhando a construção dos componentes que compõem o projeto, seguido pelos resultados obtidos, conclusões e trabalhos futuros, por fim as referências bibliográficas que auxiliaram o desenvolvimento do trabalho.

2. Trabalhos Correlatos

A seguir são relatados alguns trabalhos com propostas similares a que foi definida para elaboração deste projeto.

[Sun et al. 2020] também utiliza algoritmos genéticos como ferramenta de busca para automatizar o processo de construção de arquiteturas de redes neurais convolu-

cionais. Neste trabalho os modelos gerados são treinados e validados utilizando os conjuntos de dados CIFAR-10 e CIFAR-100. As arquiteturas geradas alcançaram resultados expressivos comparadas aos modelos que alcançaram o estado da arte em tarefas de visão computacional.

[Park et al. 2020] aplica o ENAS em uma tarefa linguística, usando um operador de cruzamento especial chamado de cromossomo não–disjuntivo [ref], apesar da complexidade dos dados linguísticos, analisando os resultados, os autores afirmam que a aplicação do NAS nesse tipo tarefa é promissor.

[Liu et al. 2017] utiliza Aprendizagem por reforço [Sutton and Barto 2018] e uma estratégia evolucionária para encontrar arquiteturas de CNN's, utilizando uma estratégia de otimização baseada em modelo sequencial (SMBO), onde é feito a busca por arquiteturas por ordem de complexidade, e, simultaneamente, aprende um modelo substituto para guiar a busca pelo espaço da arquitetura.

[Chrostoforidis et al. 2021] propõe um novo algoritmo evolucionário para busca por arquiteturas de redes neurais de forma hierárquica, aplicável a espaços de pesquisa globais. A representação arquitetônica do algoritmo organiza o modelo em múltiplos módulos hierárquicos, enquanto o processo de construção explora essa representação, a fim de explorar o espaço de busca. Esse algoritmo foi aplicado a dois conjuntos de dados utilizados como benchmarks da área de ENAS e obteve resultados de 93.2% e 94.8% respectivamente.

3. Background

3.1. Algoritmos Genéticos

Os algoritmos genéticos [Holland 1975] são meta-heurísticas bio-inspiradas baseadas na teoria da evolução de Charles Darwin, que acredita no princípio da seleção natural, onde os indivíduos que conseguem sobreviver e passar para as próximas gerações não são necessariamente aqueles mais fortes, e sim aqueles que melhor se adaptam as condições do ambiente, ou seja, os mais aptos. [Goldberg 1989] define uma estrutura básica de um AG, denominado Algoritmo Genético simples, onde o mesmo tem seu fluxo principal definido Algoritmo 1

Algorithm 1 Pseudocódigo Algoritmo Genético Simples

```
populacao ← random_pop()                    ▷ Gera uma população inicial aleatoriamente
avalia_individuos(populacao)                ▷ Avalia os indivíduos da população
melhor_individuo(populacao)                 ▷ Encontra o melhor individuo da população
i ← 0
while i < num_geracoes do                   ▷ Loop até alcançar o número de gerações
    seleciona_individuos(populacao)         ▷ Seleciona indivíduos para reprodução
    cruzamento(populacao)                   ▷ Aplica operador de cruzamento
    mutacao(populacao)                      ▷ Aplica operador de mutação
    avalia_individuos(populacao)            ▷ Avalia os indivíduos da população
    melhor_individuo(populacao)             ▷ Atualiza o melhor individuo
    i ← i + 1
end while
return melhor_individuo                     ▷ Retorna o melhor individuo encontrado
```

Os AG's são amplamente utilizados em problemas de busca e otimização onde os métodos tradicionais sofrem de alguma limitação para encontrar boas soluções, existem diversas aplicações nos mais diferentes horizontes de aplicações possíveis.

3.2. Redes Neurais Convolucionais

As Redes Neurais Convolucionais ou CNN's [Le Cun et al. 1989] como são tipicamente chamadas, são um tipo especializado de redes neurais para processamento de dados que tenham uma topologia conhecida, semelhante a uma grade. Exemplos de dados são, series temporais que podem ser vistos como uma grade unidimensional, em que os intervalos de tempo são usados para coletar as amostras de dados, outro exemplo de dado, esse comumente mais utilizado por CNN's são imagens, que podem ser interpretadas como grades bidimensionais para imagens de colorização preto e branco e tridimensionais para imagens coloridas (RGB).

Tais arquiteturas têm demonstrado grande sucesso em diversas aplicações praticas, o termo "convolucional" implica que essa arquitetura aplica uma operação matemática chamada convolução, onde a mesma é um tipo especializado de operação linear. Então, podemos dizer que as redes neurais convolucionais são arquiteturas de redes neurais que aplicam a operação de convolução em pelo menos uma de suas camadas, ao invés de realizarem a tradicional multiplicação de matriz realizada por outras arquiteturas de redes neurais. A Figura 1 mostra a topologia básica de uma rede neural convolucional.

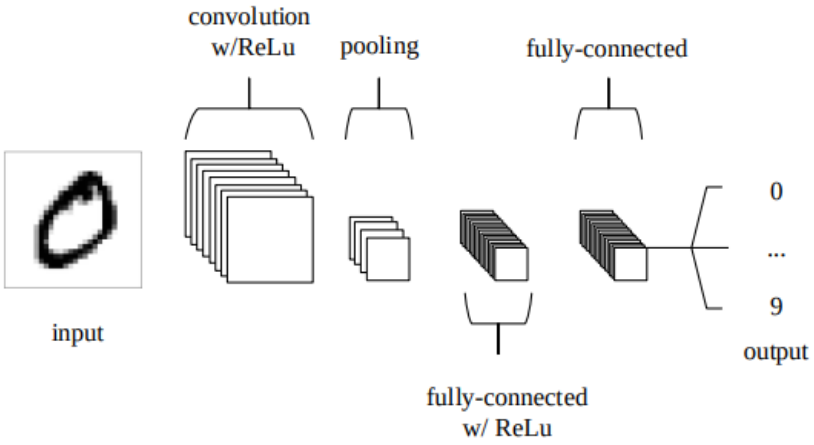

Figure 1. Arquitetura básica de uma CNN.

3.3. Busca por Arquitetura Neural Evolucionária

O processo de busca por arquitetura neural (NAS), possui três estágios: espaço de busca, estratégia de busca e estratégia de estimativa de desempenho [Elsken et al. 2018]. No espaço de busca, temos que definir quais arquiteturas podem ser representadas. Para simplificar o tamanho do espaço de busca, é necessário incorporar conhecimento prévio, este estágio é altamente dependente de especialistas humanos. A estratégia de busca é o processo mais importante no NAS, ela determina como explorar o espaço de busca. A estratégia de estimativa de desempenho avalia o desempenho das redes neurais para atingir o objetivo do NAS, encontrar a arquitetura com maior desempenho. Na Figura 2 temos uma representação de forma abstrata de um algoritmo de NAS.

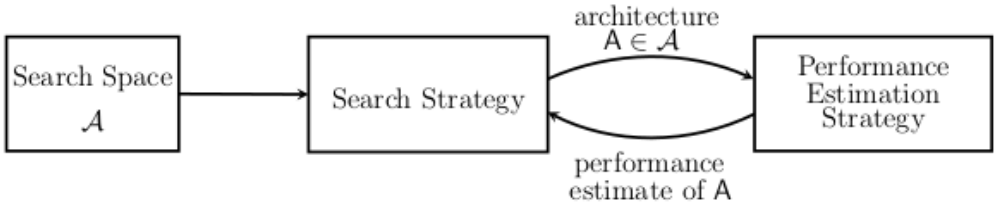

Figure 2. Ilustração abstrata de um algoritmo de NAS. [Elsken, Thomas, et al. 2018]

Diversas estrategias de busca podem ser utilizadas para explorar o espaço de arquiteturas neurais. Utilizando otimização bayesiana, [Bergstra et al. 2012] dá inicio aos primeiros casos de sucesso aplicando NAS, obtendo arquiteturas estado da arte em visão computacional. NAS-RL [Zoph and Le 2016] e MetaQNN [Baker et al. 2016] são considerados os responsáveis por tornar o NAS um dos principais tópicos de pesquisa pela comunidade de machine learning, ambos utilizam aprendizagem por reforço como estratégia de busca, e alcançaram resultado estado da arte em tarefas de classificação de imagens.

Uma estratégia de busca bastante utilizada são as estratégias evolucionárias, o processo de busca utilizando esses métodos é denominado Evolutionary Neura Architecture Search (ENAS). A Figura 3 mostra uma ilustração do fluxo de um algoritmo de ENAS. Primeiro é criado um espaço de busca inicial inicializando a população de forma aleatória, cada indivíduo da população representa uma possível solução do problema, ou seja, uma arquitetura de rede neural. Em seguida, esses indivíduos são avaliados e atribuídos um valor de fitness, a seguir inicia-se o processo evolucionário até o critério de parada ser atingido, após o termino desse processo, é retornado o individuo que obteve o melhor desempenho baseado na função de avaliação definida.

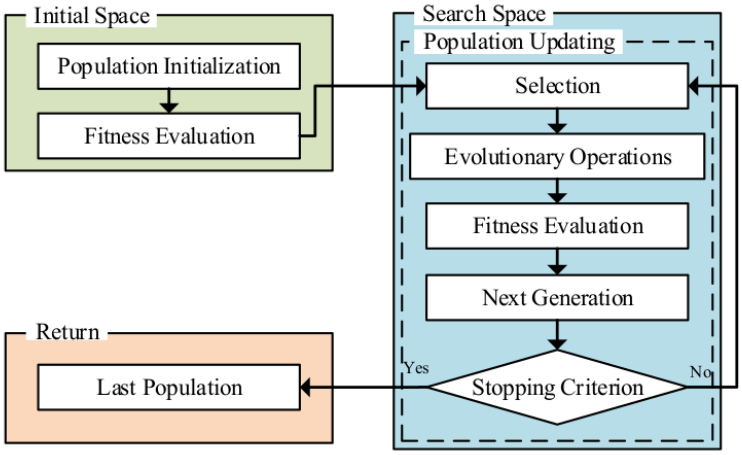

Figure 3. Fluxo básico de um algoritmo de ENAS. [Liu et al. 2017]

4. Metodologia

Nesta seção são apresentados os procedimentos metodológicos utilizados para desenvolvimento do trabalho. Estes foram divididos em 4 etapas, as mesmas serão detalhadas

a seguir: linguagem de programação e bibliotecas; armazenamento dos dados; algoritmo implementado; e treinamento dos modelos.

4.1. Linguagem de programação e bibliotecas

O software implementado, desde a construção do algoritmo genético até o desenvolvimento dos pipelines de treinamento das redes neurais, foram todos desenvolvidos utilizando a linguagem de programação Python (https://www.python.org).

O desenvolvimento do algoritmo genético utilizou, além da linguagem Python e suas bibliotecas que fazem parte do seu pacote padrão, a biblioteca Numpy (https://numpy.org) para realização de manipulação de dados de arrays, e a biblioteca Matplotlib (https://matplotlib.org) para plotagem dos gráficos.

Os pipelines de treinamento das redes neurais foram construídos utilizando o framework Tensorflow (http://tensorflow.org) desenvolvido pela Google, com a biblioteca Keras (https://keras.io) como interface para construção dos modelos.

4.2. Conjunto de dados

O CIFAR-10 é um conjunto de dados rotulados de 60.000 imagens minúsculas. Essas imagens foram coletadas por Alex Krizhevsky, Vinod Nair e Geoffrey Hinton. O mesmo consiste em 60.000 imagens coloridas 32x32 em 10 classes, com 6.000 imagens por classe, existem 50.000 imagens de treinamento e 10.000 imagens de teste.

O conjunto de dados é dividido em cinco lotes de treinamento e um lote de teste, cada um com 10.000 imagens. O lote de teste contém exatamente 1.000 imagens selecionadas aleatoriamente de cada classe. Os lotes de treinamento contêm exatamente 5.000 imagens de cada classe. A Figura 4 mostra as classes presentes no dataset, juntamento com exemplos de imagens que compõem os dados.

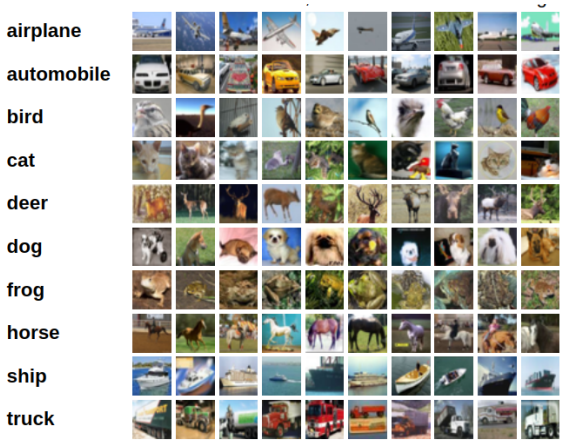

Figure 4. Exemplos de imagens do dataset.

4.3. Treinamento das arquiteturas

A avaliação dos indivíduos é realizada através do treinamento de suas arquiteturas neurais representadas pelo cromossomo dos indivíduos. Dessa forma, foi utilizado uma estratégia de treinamento utilizando a nuvem através da ferramenta Colaboratory da Google, que disponibiliza ambientes com acesso a unidades de processamento gráficos (GPUs).

O processo de treinamento do algoritmo requer tempo, e necessita de uma boa GPU para ser realizado, dessa forma, foi necessário implementar uma forma de se recuperar o estado de uma execução quando a mesma não conseguir ser finalizada durante o tempo de execução que o Colab oferece. O tempo de execução necessário para avaliar todos os indivíduos da população foi de 4 dias, utilizando uma GPU Tesla T4 fornecida pelo plano PRO do Colaboratory.

4.4. Armazenamento dos dados

O armazenamento de dados é um ponto crucial para a execução do algoritmo, uma vez que os dados gerados além de servirem para a realização de análises posteriores, são úteis na continuação de uma execução, onde o algoritmo busca as informações da execução armazenadas na base dados. A estratégia de armazenamento é composta por um banco de dados relacional utilizando o PostgreSQL, um banco de dados não relacional sendo representado pelo MongoDB, e um bucket S3 na Amazon Web Services (AWS).

O PostgreSQL é um banco de dados relacional avançado de código aberto de classe empresarial que oferece suporte a consultas SQL (relacionais) e JSON (não relacionais). É um sistema de gerenciamento de banco de dados altamente estável, apoiado por mais de 20 anos de desenvolvimento comunitário que contribuiu para seus altos níveis de resiliência, integridade e correção. A Figura 5 mostra o diagrama relacional das tabelas implementadas para armazenar os dados relacionais do projeto.

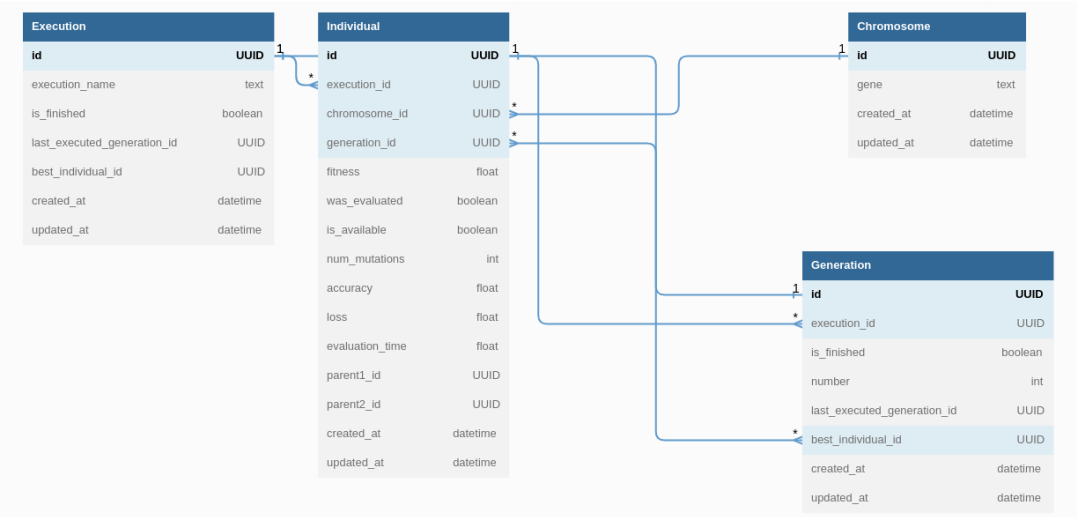

Figure 5. Diagrama Entidade Relacionamento.

O MongoDB é um banco de dados de código-fonte aberto, multiplataforma e baseado em documentos distribuídos, projetado para facilitar o desenvolvimento e o dimensionamento de aplicativos. Neste projeto eleé utilizado para armazenar uma estrutura de coleção que representa um objeto contendo os parâmetros de uma execução do algoritmo, através desse documento armazenado é possível realizar a continuação de uma execução necessitando apenas da mesma. Na Tabela 1 é mostrada quais informações são armazenadas utilizando o MongoDB.

Parâmetro	Tipo	Descrição
execution_id	UUID	Id da execução salva no banco relacional
crossover_rate	Float	Taxa de cruzamento usada pelo AG
num_generations	Int	Número de gerações do AG
pop_size	Int	Tamanho da população de individuos
mutation_rate	Float	Taxa de mutação utilizado pelo AG
crossover_method	Função	Método de cruzamento utilizado pelo AG
mutation_method	Função	Método de mutação utilizado pelo AG.
selection_method	Função	Método de seleção utilizado pelo AG.
batch_size	Int	Tamanho do lote de dados utilizado cada iteração do treinamento das arquiteturas.
num_epochs	Int	Número de epócas utiliazado para realizar o treinamento das arquiteturas.

Table 1. Descrição do objeto armazenado no MongoDB com informações de uma execução do algoritmo.

4.5. Algoritmo implementado

O ponto de partida do processo de desenvolvimento do trabalho foi a realização do levantamento de requisitos, visando definir os componentes que seriam implementados. Diante disso, foi-se feito um estudo sobre trabalhos anteriores com o propósito de encontrar os componentes principais que constituem um algoritmo de ENAS.

A estratégia de busca escolhida é um algoritmo genético simples [Goldberg 1989]. Para execução do algoritmo, assim como a maioria dos algoritmos evolucionários, é preciso definir alguns parâmetros cruciais no processo aprendizagem do AG. Dentre eles, podemos destacar: número de gerações, tamanho da população, método de seleção dos indivíduos, métodos dos operadores genéticos de cruzamento e mutação, com suas respectivas taxas. Na Tabela 2 é mostrado os valores utilizados destes parâmetros.

Parâmetro	Tipo	Valor
Tamanho população	Inteiro	30
Nº. gerações	Inteiro	20
Metódo de seleção	Função	Torneio
Metódo de cruzamento	Função	Uniforme
Metódo de mutação	Função	Bit Swap
Taxa de cruzamento (%)	Inteiro	85%
Taxa de mutação (%)	Inteiro	20%

Table 2. Valores parâmetros AG.

A forma de avaliação dos indivíduos será através da função de fitness que irá atribuir um valor com base no desempenho obtido pelo indivíduo. A definição desta função é um dos maiores desafios ao se utilizar algoritmos genéticos, visto que a mesma precisa ser definida a ponto que consiga avaliar os indivíduos adequadamente, obtendo as melhores soluções para o problema. Neste trabalho, esta definição foi realizada de forma empírica, através de testes com valores e operações matemáticas, com o objetivo de obter

arquiteturas com bons valores de acurácia e loss, tendo o menor número de pesos possível para realizar a tarefa. A função de aptidão definida é mostrada na equação abaixo.

$$função\_fitness = (100 * ((20 * acuracia) + 30 * (\frac{1}{loss}))) - (10 * ((1.5 * \text{num\_camadas}) + (2 * \log(num\_parametros)))(1)$$

O espaço de busca ou espaço de codificação, é constituído pelo espaço de busca inicial, e o espaço de busca geral. O espaço de busca inicial é um subconjunto do espaço de busca geral, como mostra a Figura 6. Os parâmetros que definem os limites do espaço de codificação, são o número máximo de camadas da arquitetura, onde o mesmo é definido um valor de 10 blocos de camadas por arquitetura.

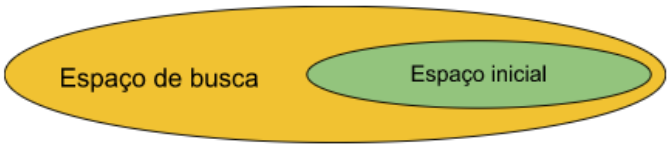

Figure 6. Representação do espaço de busca.

A codificação do cromossomo do indivíduo, que representa a arquitetura de uma rede neural, é realizada utilizando uma lista de listas, onde cada lista interna presenta um bloco de camadas, e cada camada é representada por uma string que define o tipo da camada e os parâmetros da mesma. As camadas possíveis são, convolucionais (Conv2d), pooling (MaxPooling e AveragePooling), dropout, e camadas densas (Dense).

Algorithm 2 Codificação do cromossomo

```
cromossomo ← []
num_camadas ← randint(2, MAX_DEPTH)
pooling ← choice(MAX_POOL, AVG_POOL)
i ← 0
while i < num_camadas do
    num_filtros ← choice(conv_filters)
    conv_layer ← "CONV_num_filtros"
    cromossomo.append(conv_layer)
    pool_layer ← pooling
    cromossomo.append(pool_layer)
    if uniform(0, 1) > 0.5 then
        rate ← uniform(0.1, 0.5)
        drop_layer ← "DROP_rate"
        cromossomo.append(drop_layer)
    end if
    i ← i + 1
end while
return cromossomo
```

Existem dois cenários possíveis ao se utilizar o algoritmo, aquele que se inicia

uma execução nova, e outro que se continua o processo a partir de uma execução já existente na base dados que não foi finalizada. No primeiro cenário, é necessário passar os parâmetros requeridos para se iniciar uma nova execução, são eles os parâmetros do algoritmo genético definidos na Tabela 2, e os parâmetros específicos para o treinamento das arquiteturas de redes neurais. O segundo cenário é bem simples, ao executar o algoritmo, caso haja execuções não finalizadas no banco de dados, o algoritmo mostra os ids das mesmas e pergunta se o usuário deseja continuar alguma dessas execuções ou iniciar uma nova. Diante disso, caso seja escolhido continuar uma execução, o algoritmo irá buscar na base de dados as informações daquela execução, e continuar o processo de onde parou. O fluxo de execução completo do algoritmo pode ser observado na Figura 7.

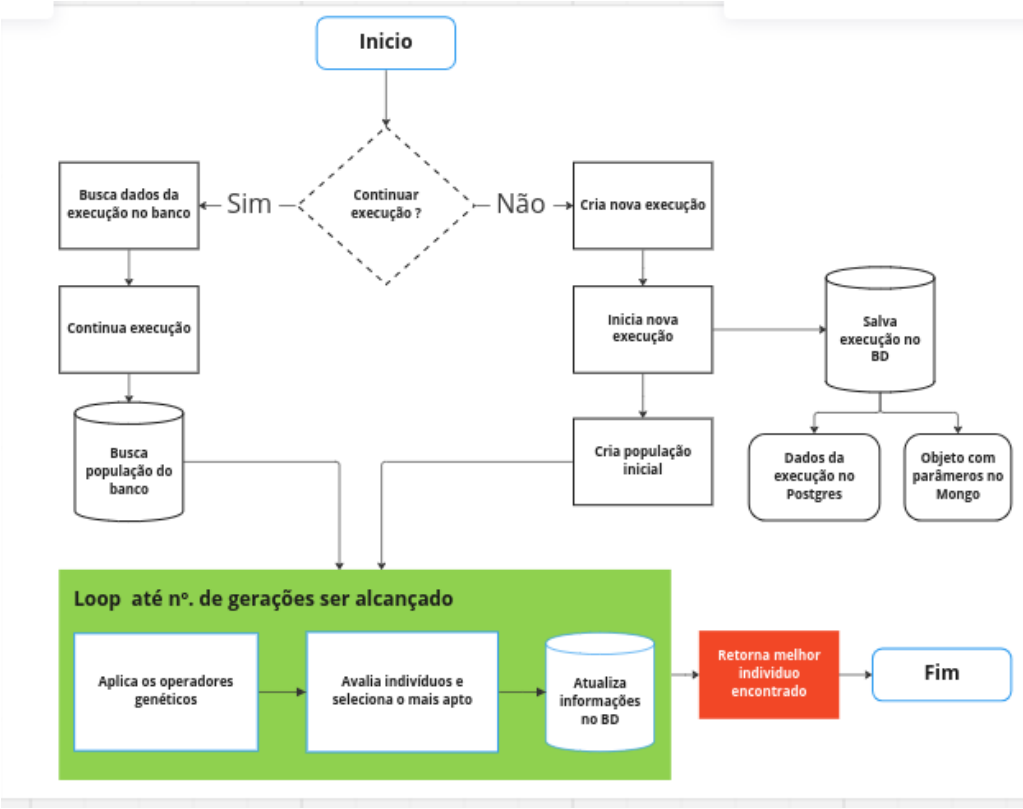

Figure 7. Fluxo de execução do Algoritmo. Fonte: Autor

5. Resultados

Nesta seção, é mostrado uma comparação entre o resultado obtido e alguns trabalhos relacionados, além dos resultados de algumas arquiteturas que atingiram o estado da arte e foram manualmente projetadas. Além do resultado de acurácia usado para comparar os desempenhos dos modelos, o número de parâmetros do modelo, e número de dias de GPU usados para treinamento das arquiteturas também é utilizado. Na Tabela 3 é mostrado esses valores.

Visando testar a melhor arquitetura encontrada, após a realização de uma etapa de treinamento completa algoritmo, do foi realizado o treinamento da melhor arquitetura obtida por cinco vezes sbre o conjunto de dados, obtendo o maior valor de acurácia alcançado no conjunto de teste. O melhor individuo encontrado obteve uma acurácia de

70%, e uma loss de validação de **0.94**, o valor de fitness atribuído após a validação de seu cromossomo foi de **3663.68**.

	Arquitetura	Acurácia	Nº. Parametros	Dias de GPU
Manualmente	ResNet	93.57%	1.7M	-
	DenseNet-BC	95.49%	1.7M	-
	VGGNet	93.44%	20.04M	-
Automático	NAS	93.99%	2.5M	23
	CNN-GA	95.22%	2.9M	35
	CNN-GA + cutout	96.78%	2.9M	35
	Autor	70%	24.1M	**4**

Table 3. Comparação de resultados.

6. Conclusões e trabalhos futuros

Este trabalho propôs implementar uma solução de busca por arquiteturas neurais evolucionária utilizando algoritmos genéticos. Através dos experimentos realizados, podemos constatar que mesmo com um algoritmo genético simples sem nenhuma modificação específica para o problema, podemos obter resultados que podem ser satisfatórios. Quando comparamos, o poder de processamento utilizado e o tempo de treinamento do trabalho com outros trabalhos relacionados, vemos que nosso algoritmo mesmo sendo treinado em somente uma GPU, por um curto período, alcançou bons resultados.

Como trabalhos futuros, podemos destacar um estudo mais aprofundado sobre os componentes do AG aplicados a construção de topologias de redes neurais, como, por exemplo, o desenvolvimento de operadores genéticos de cruzamento e mutação específicos para o cromossomo que representa uma arquitetura neural, e a implantação de uma função de aptidão que consiga melhor avaliar os indivíduos em prol do objetivo.

7. Agradecimentos

A realização deste trabalho só se tornou possível através das oportunidades recebidas do meu orientador e amigo Otávio Noura Teixeira, bem como o laboratório MEC^{2}A e seus integrantes, que tiveram uma papel primordial no meu desenvolvimento acadêmico, fomentando meu interesse pela área de IA e computação em geral.

Esta pesquisa foi parcialmente mantida pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPQ, através do Programa Institucional de Bolsas de Iniciação Científica - PIBIC, por meio da Universidade Federal do Para - UFPA, campus Tucuruí. Eterna gratidão pelo apoio e incentivo.

References

Baker, B., Gupta, O., Naik, N., and Raskar, R. (2016). Designing neural network architectures using reinforcement learning.

Bergstra, J., Yamins, D., and Cox, D. D. (2012). Making a science of model search.

Chrostoforidis, A., Kyriakides, G., and Margaritis, K. (2021). A novel evolutionary algorithm for hierarchical neural architecture search.

Elsken, T., Metzen, J. H., and Hutter, F. (2018). Neural architecture search: A survey.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., USA, 1st edition.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI. second edition, 1992.

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2016). Densely connected convolutional networks.

Le Cun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Handwritten digit recognition with a back-propagation network. In *Proceedings of the 2nd International Conference on Neural Information Processing Systems*, NIPS'89, page 396–404, Cambridge, MA, USA. MIT Press.

Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.-J., Fei-Fei, L., Yuille, A., Huang, J., and Murphy, K. (2017). Progressive neural architecture search.

Park, K.-m., Shin, D., and Yoo, Y. (2020). Evolutionary neural architecture search (nas) using chromosome non-disjunction for korean grammaticality tasks. *Applied Sciences*, 10(10).

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition.

Sun, Y., Xue, B., Zhang, M., Yen, G. G., and Lv, J. (2020). Automatically designing cnn architectures using the genetic algorithm for image classification. *IEEE Transactions on Cybernetics*, 50(9):3840–3854.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. The MIT Press, second edition.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014). Going deeper with convolutions.

Tan, M. and Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks.

Zoph, B. and Le, Q. V. (2016). Neural architecture search with reinforcement learning.