ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN

MÔN HỌC:

MẠNG NEURAL VÀ THUẬT GIẢI DI TRUYỀN

Đề tài:

Neural Architecture Search with NAS-Bench-201

Lớp: CS410.M11.KHCL

Giảng viên hướng dẫn: TS. Lương Ngọc Hoàng

Sinh viên thực hiện:

Phan Nguyễn Thành Nhân	19521943
Phạm Minh Long	19521797
Tạ Huỳnh Đức Huy	19521634

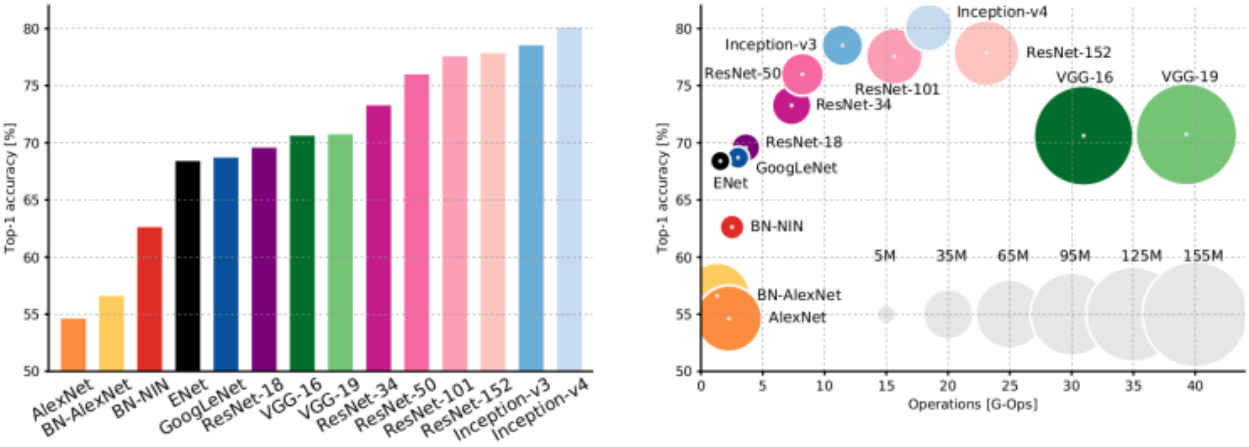
Thành phố Hồ Chí Minh, ngày 14 tháng 12 năm 2021

Mục lục

I. Tổng quan

1. Mô tả bài toán

Trong những năm gần đây, *Mạng Neural học sâu (Deep Learning)* đã tạo ra những bước đột phá, phát triển mạnh mẽ cho lĩnh vực *Máy học (Machine Learning)*. Cùng với sự phát triển, nhiều mạng neural ra đời để giải quyết nhiều tác vụ ở những lĩnh vực khác nhau như *Thị giác máy tính (Computer Vision)*, *Xử lý ngôn ngữ tự nhiên (Natural Language Processing)*. Những kiến trúc mạng phổ biến như ResNet, VGG,... đạt được những kết quả tích cực và tiếp tục phát triển nhiều phiên bản khác nhau. Tuy nhiên việc nghiên cứu và thiết kế một kiến trúc mạng đòi hỏi rất nhiều kinh nghiệm của các chuyên gia trong ngành và hao tốn rất nhiều tài nguyên tính toán cũng như thời gian thử nghiệm.

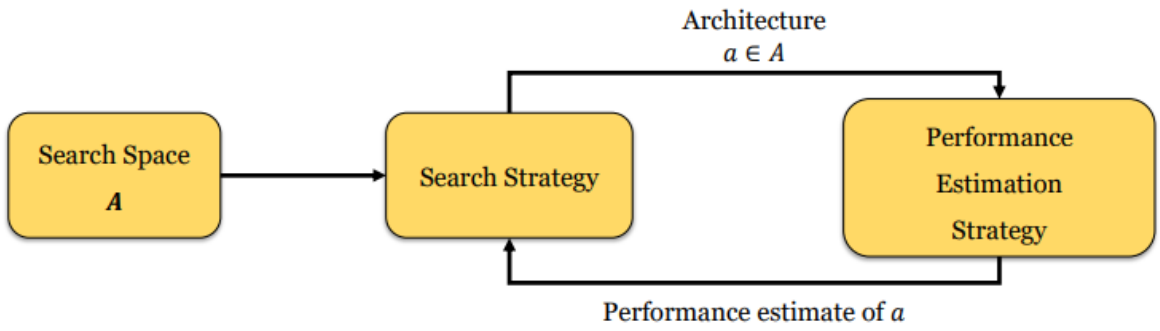


Hình 1: Thống kê các kiến trúc mạng neural và so sánh hiệu suất giữa các kiến trúc

Chính vì vậy, để giải quyết vấn đề đó, một kỹ thuật tự động thiết kế kiến trúc mạng neural đã ra đời, mang tên *Tìm kiếm kiến trúc mạng neural (Neural Architecture Search, viết tắt: NAS)* - được chú ý đến qua công trình sử dụng thuật toán *Học tăng cường (Reinforcement Learning)* để tìm ra kiến trúc *Mạng tích chập (Convolutional Neural Network)* và đạt được độ chính xác rất cao trên bộ dữ liệu *CIFAR-10*.

2. Mô hình Neural Architecture Search

Tìm kiếm kiến trúc mạng neural (Neural Architecture Search) hay còn gọi là NAS là một phương pháp tự động hóa việc thiết kế kiến trúc mạng neural để mô hình đạt được hiệu suất cao trên một tác vụ cụ thể như Nhận diện vật thể (Object Detection) hay Xử lý ngôn ngữ tự nhiên (Natural Language Processing). Trong mô hình NAS gồm 3 thành phần chính:



Hình 2: Các thành phần chính của bài toán Neural Architecture Search

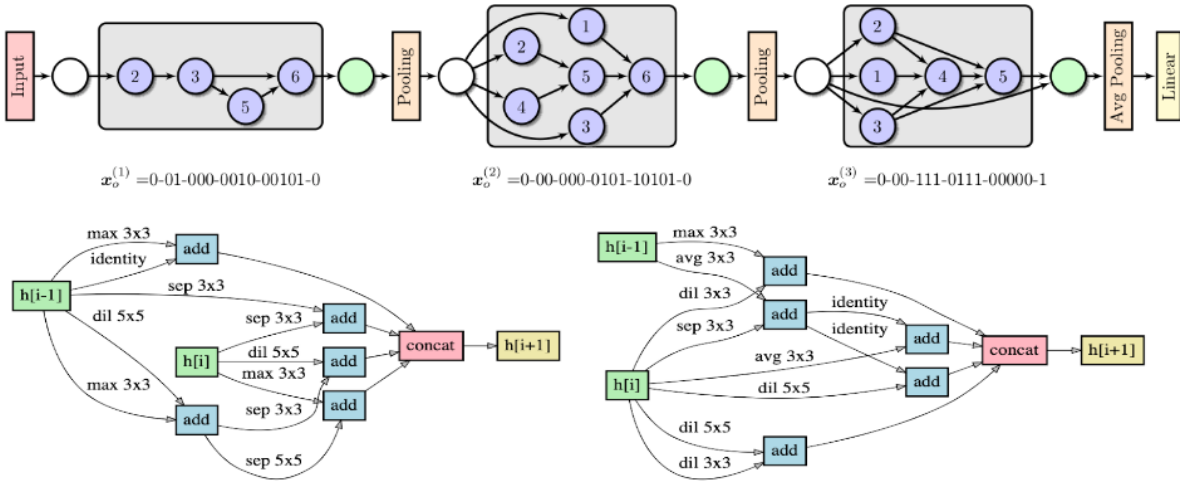
Không gian tìm kiếm (Search Space)

Là không gian chứa toàn bộ các kiến trúc mà chúng ta cần tối ưu. Tùy vào bài toán tối ưu hóa, không gian tìm kiếm được chia làm hai cấp độ phổ biến là *macro* và *micro.*

Về không gian tìm kiếm *macro,* chúng ta sẽ có tập hợp các cells khác nhau, một giải pháp sẽ tương ứng với một trình tự sắp xếp các cells trong kiến trúc sao cho kiến trúc đó đạt được hiệu suất tối ưu nhất.

Ngược lại với cấp độ *macro,* trong không gian tìm kiếm *micro*, một kiến trúc sẽ bao gồm các cells như nhau, và nhiệm vụ của chúng ta là sắp xếp các operations trong cell sao cho kiến trúc được cấu tạo từ N cells đó đạt được hiệu suất tối ưu nhất.

Một điều lưu ý ở hai cấp độ *macro* và *micro* là khi xây dựng các không gian tìm kiếm phải cần xác định số lượng cells tối đa bên trong một kiến trúc đối với cấp độ *macro* cũng như số lượng operations tối đa bên trong một cell đối với cấp *micro*, đồng thời phải xác định được các loại cells hay các loại operations có thể chọn được trong không gian tìm kiếm.



Hình 3: Hình ảnh mô phỏng các không gian tìm kiếm có trong bài toán NAS. Hình ở trên là không gian tìm kiếm Macro, hình ở dưới là không tìm kiếm Micro

Chiến lược tìm kiếm (Search Strategy)

Là chiến lược khám phá, tìm kiếm các kiến trúc trong không gian tìm kiếm để tối ưu. Trong bài toán NAS, các chiến lược tìm kiếm sẽ tương ứng với các thuật toán, giải pháp phổ biến như *Thuật toán tiến hóa (Evolutionary Algorithm)*, *Học tăng cường (Reinforcement Learning)* hay *Tối ưu hóa Bayes (Bayesian Optimization)*. Ngoài ra thay vì chúng ta sử dụng đơn lẻ từng thuật toán, chúng ta cũng có thể kết hợp các thuật toán lại thành một chiến lược tìm kiếm.

Chiến lược ước lượng hiệu suất (Performance Estimation Strategy)

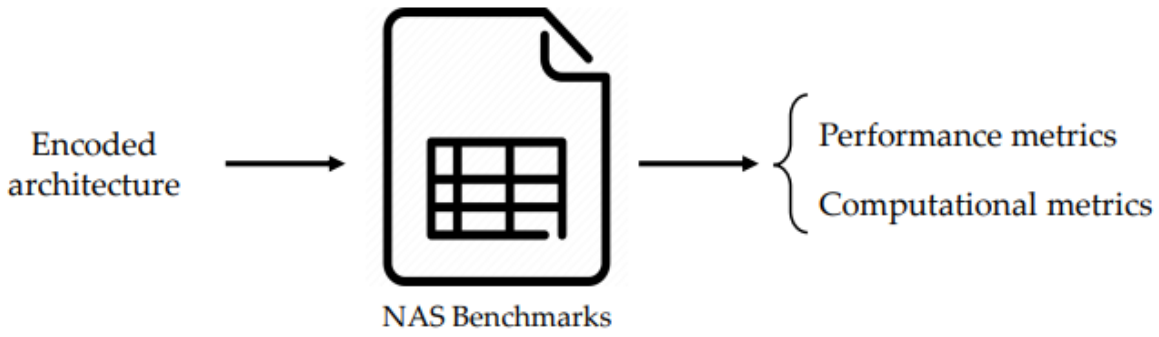
Mục tiêu của bài toán NAS là tìm kiếm một hoặc nhiều các kiến trúc tối ưu cho một hoặc nhiều các giá trị mục tiêu. Thông thường để đánh giá hiệu suất của kiến trúc, chúng ta cần huấn luyện các kiến trúc đó qua vài trăm các epochs sau đó

đánh giá dựa trên tập validation hoặc tập test. Tùy vào bài toán chúng ta đang giải quyết là đơn mục tiêu hay đa mục tiêu, mà chúng ta sẽ có các chiến lược đánh giá hiệu suất cho phù hợp, nếu chúng ta đang làm bài toán tối ưu hóa đơn mục tiêu thì thông thường các nhà nghiên cứu họ sẽ chọn *độ chính xác* làm giá trị mục tiêu, còn đối với bài toán đa mục tiêu thì có thể chọn các mục tiêu như *độ chính xác, số lượng tham số, số lượng phép tính,...*

Quy trình căn bản của một bài toán NAS là chúng ta sẽ dựa vào *Chiến lược tìm kiếm* để chọn các kiến trúc trong *Không gian tìm kiếm* để tối ưu, sau đó chúng ta sẽ dùng *Chiến lược ước lượng hiệu suất* để đánh giá hiệu suất của kiến trúc. Tùy thuộc mục tiêu bài toán mà ta tiến hành nhiều lần thử nghiệm trên các kiến trúc trong *Không gian tìm kiếm* cho đến khi kiến trúc đạt được hiệu suất tối ưu mà chúng ta kỳ vọng.

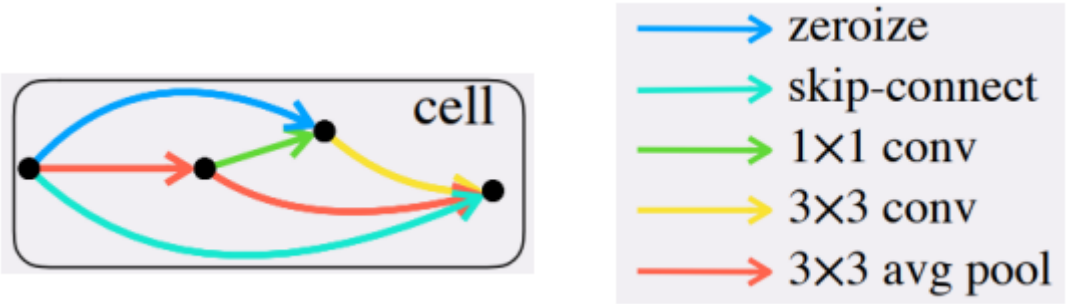
3. Hướng tiếp cận

Việc đánh giá hiệu suất của một kiến trúc mới một cách trực quan với những phương pháp đã được xây dựng trước đó là rất khó khăn do khác nhau về các thông số, và đòi hỏi rất nhiều tài nguyên, thời gian. Chính vì vậy, nhóm sử dụng NAS Benchmark - một phương thức đánh giá hiệu suất kiến trúc, tương ứng với thông số hiệu suất được ghi lại của các kiến trúc trong không gian tìm kiếm.



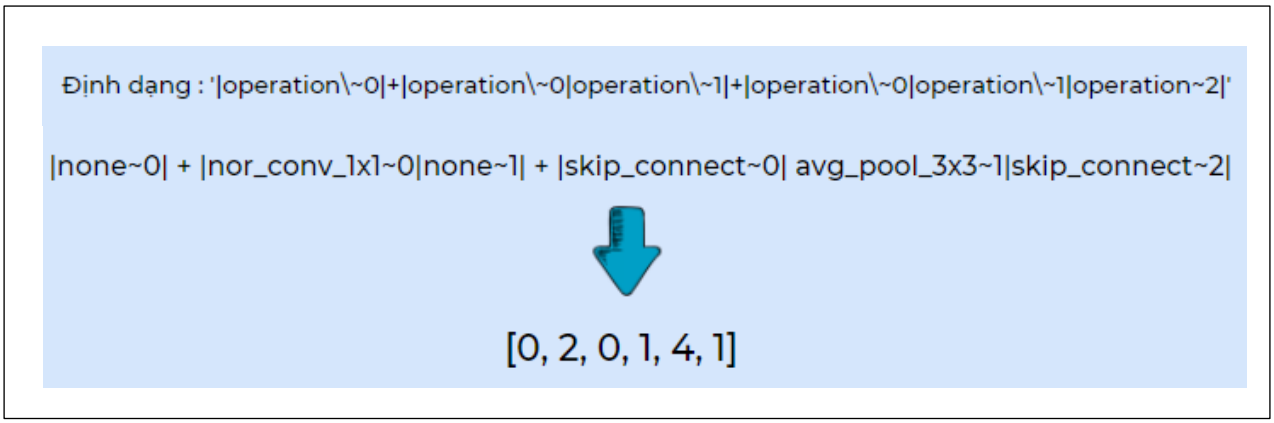
Hình 4: Hình ảnh minh họa các hoạt động của các hàm NAS Benchmarks

NAS-Bench-201 là bộ dữ liệu benchmark được xây dựng cho bài toán NAS với không gian tìm kiếm ở cấp độ micro. Bao gồm bộ dữ liệu của 15,625 kiến trúc với mỗi kiến trúc bao gồm hai thông số chính là *Chỉ số hiệu suất (Performance metrics)* bao gồm *Độ chính xác huấn luyện (training accuracy)*; *Độ chính xác kiểm định (validation accuracy)*; Độ chính xác thử nghiệm *(Testing accuracy)* và Chỉ số hao phí (Computational metrics) bao gồm *FLOPs, Số lượng tham số (params), latency*. Có 3 bộ dataset là CIFAR-10; CIFAR-100; ImageNet16-120. Với mỗi kiến trúc được biểu diễn dưới dạng *đồ thị không chu trình có hướng* (Directed Acyclic Graph, viết tắt là DAG).



Hình 5: Hình ảnh minh họa cấu trúc của một kiến trúc trong không gian tìm kiếm NAS-Bench-201

Mỗi kiến trúc trong bộ dữ liệu NAS-Bench-201, mỗi DAG được biểu diễn gồm 4 nodes và 6 cạnh. Kiến trúc trong không gian tìm kiếm được mã hóa bằng một vector 1 chiều có 6 phần tử tương ứng số operations tối đa của một kiến trúc, mỗi phần tử là một số nguyên ngẫu nhiên có giá trị từ 0 đến 4. Hiểu đơn giản, NAS Benchmarks như một dictionary, với mỗi kiến trúc có một "Key" được mã hóa theo định dạng các operation là một trong năm loại: 'none', 'skip_connect' , 'nor_conv_1x1' , 'nor_conv_3x3' , 'avg_pool_3x3'



Hình 6: Hình ảnh minh họa ví dụ mã hóa một kiến trúc trong NAS-Bench-201

Trong khuôn khổ đồ án, nhóm sẽ tiến hành thực nghiệm áp dụng NAS-Bench-201 để giải quyết 2 bài toán tối ưu đơn mục tiêu và tối ưu hóa đa mục tiêu. Quá trình thực nghiệm sẽ được thực hiện bên dưới.

II. Thực nghiệm trên bộ dữ liệu

1. Đơn mục tiêu

Trong bài toán về tối ưu hóa đơn mục tiêu, nhóm đã sử dụng thuật toán *Thuật giải di truyền (Genetic Algorithm)* để tối ưu hóa một mục tiêu duy nhất là *Độ chính xác thử nghiệm*. Mỗi cá thể là một kiến trúc trong *Không gian tìm kiếm* của hàm NAS-Bench-201, sau đó tiến hành cài đặt thuật toán *Thuật giải di truyền* với bản cài đặt *POPOP* để tiến hành lai ghép và tìm kiến trúc có *Độ chính xác thử nghiệm* là cao nhất.

Quá trình thực hiện chi tiết được xây dựng qua các bước như sau:

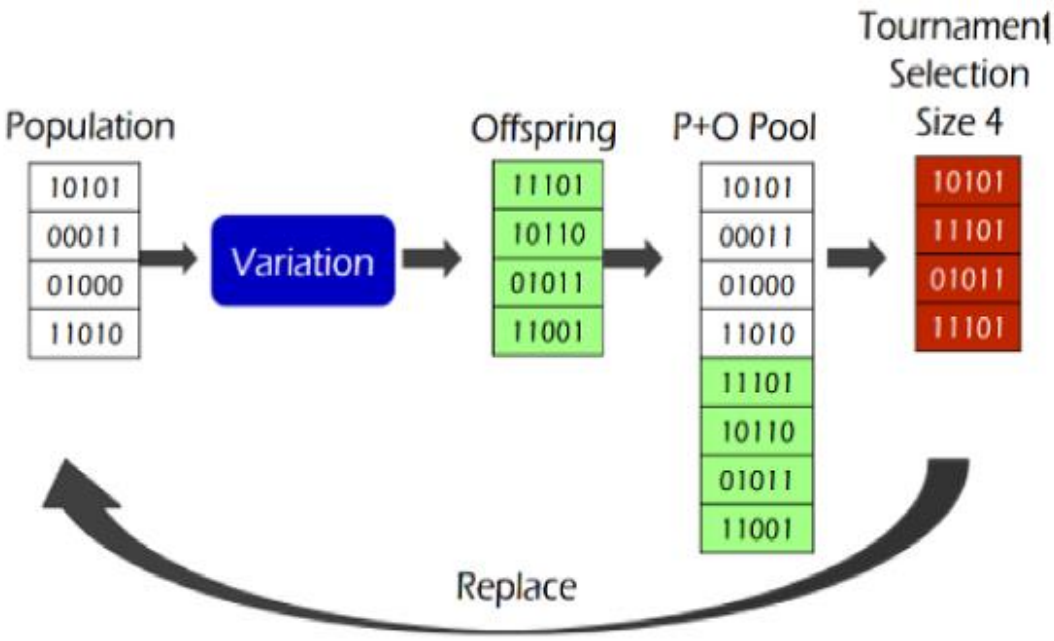
Bước 1: Khởi tạo một quần thể ngẫu nhiên chứa các kiến trúc cần tối ưu hóa.

Bước 2: Sau đó đánh giá độ thích nghi của các các thể trong quần thể vừa mới tạo, nếu các cá thể chưa hội tụ thì tiến hành lai ghép.

Bước 3: Lai ghép các cá thể trong quần thể bằng phép lai đồng nhất. Chúng ta sẽ lấy ngẫu nhiên 2 cá thể trong quần thể, sau đó ứng với từng vị trí của các mã gen, nhóm sẽ phát sinh một con số ngẫu nhiên trong khoảng từ 0 đến 1, nếu số đó nhỏ hơn 0.5 thì nhóm sẽ hoán đổi vị trí mã gen tương ứng của 2 cá thể.

Bước 4: Sau khi lai ghép tạo ra các cá thể con, chúng ta sẽ gộp các cá thể cha và các cá thể con, sau đó tiến hành phương pháp chọn lọc *Tournament Selection* với *tournament size* là 4. Chúng ta sẽ so sánh bằng cách chọn 1 trong 4 cá thể có độ thích nghi cao nhất để làm cá thể cho thế hệ sau.

Bước 5: Sau khi đã có thể một quần thể mới, chúng ta sẽ đánh giá độ thích nghi của quần thể đã hội tụ chưa, nếu hội tụ chúng ta sẽ dừng thuật toán, nếu chưa chúng ta tiếp tục cho lai ghép rồi chọn lọc cho đến khi quần thể hội tụ.



Hình 7: Mô hình hoạt động của thuật toán Genetic Algorithm với bản cài đặt POPOP

Trong quá trình thực nghiệm, nhóm nhận thấy để quần thể có thể hội tụ về kiến trúc có *Độ chính xác thử nghiệm* cao nhất thì sẽ thấy phụ thuộc rất nhiều vào

kích thước quần thể, đồng thời do phát sinh quần thể ngẫu nhiên nên mỗi lần thực nghiệm lại kết quả khác nhau. Do đó nhóm đã tiến hành chạy thí nghiệm với 31 lần khác nhau để tìm *kích thước quần thể nhỏ nhất cần thiết (minimally required population size, viết tắt là MRPS)* để quần thể có thể hội tụ về kiến trúc tối ưu nhất.

Bước đầu nhóm sẽ cho kích thước quần thể bằng 4, sau đó sẽ chạy thí nghiệm trên 31 lần khác nhau để đánh giá với kích thước quần thể đó thì quần thể có hội tụ được trong 31 lần chạy khác nhau hay không. Nếu quần thể chưa hội tụ được thì nhóm sẽ tăng kích thước quần thể lên gấp 2 lần sau đó tiếp tục cho thí nghiệm 31 lần khác nhau. Đến khi quần thể hội tụ thì nhóm sẽ quy định số quần thể hiện tại là số quần thể tối đa mà quần thể chắc chắn sẽ hội tụ, sau đó sẽ giảm kích thước quần thể từ từ để tìm *kích thước quần thể nhỏ nhất cần thiết*.

Nhóm đã chạy thí nghiệm trên cả bộ dữ liệu mà hàm NAS-Bench-201 đã cung cấp bao gồm: cifar100, cifar10-valid và ImageNet16-200. Kết quả thực nghiệm như sau:

Dataset	MRPS
CIFAR10-VALID	320
CIFAR100	80
ImageNet16-120	256

Hình 8: Kết quả thực nghiệm tìm kiếm MRPS của bài toán tối ưu hóa đơn mục tiêu

Có thể nhận thấy rằng mỗi bộ dữ liệu lại cần các *kích thước quần thể nhỏ nhất cần thiết* khác nhau để hội tụ. Với bộ *cifar10-valid* thì cần 320 số lượng cá thể để quần thể hội tụ, còn đối với bộ *cifar100* thì lại ít hơn với chỉ 80 số lượng cá thể, còn bộ *ImageNet16-120* thì cần tới 256 số lượng cá thể.

2. Đa mục tiêu

Đối với bài toán tối ưu hóa đa mục tiêu, nhóm đã sử dụng thuật toán *Nondominated Sorting Genetic Algorithm II (NSGA-II)* để tối thiểu hóa hai mục tiêu. Hai mục tiêu mà nhóm đã chọn để tối thiểu hóa là *Độ lỗi thử nghiệm* và *Số lượng parameters.* Trong đó, *Độ lỗi thử nghiệm* được xác định theo công thức:

$$\textit{Độ lỗi thử nghiệm} = 100 - \textit{Độ chính xác thử nghiệm}$$

Hình 9: Công thức tính toán Độ lỗi thử nghiệm

Cũng tương tự như bài toán tối ưu hóa đơn mục tiêu, nhóm sẽ quy định mỗi kiến trúc trong NAS-Bench-201 là một cá thể trong quần thể. Mục tiêu của bài toán là sử dụng thuật toán *NSGA-II* để tìm ra các kiến trúc tối ưu - các kiến trúc nằm trên *biên không bị thống trị (nondominated front)*. Về quá trình thực hiện chi tiết sẽ có đôi chút khác biệt so với bài toán đơn mục tiêu.

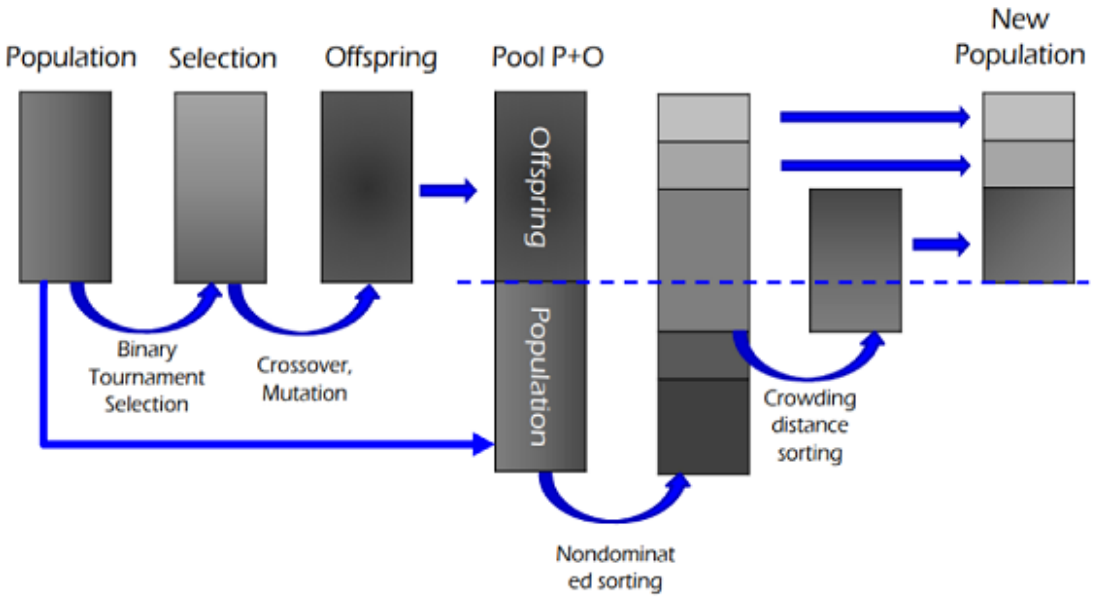
Bước 1: Chúng ta cũng sẽ khởi tạo một quần thể ngẫu nhiên với kích thước quần thể là 40, sau đó chúng ta sẽ chọn các cá thể có độ thích nghi tốt nhất với phép chọn lọc *Binary Tournament Selection.* Ở đây thay vì chúng ta so sánh bằng độ thích nghi, thì chúng ta sẽ so sánh giữa 2 cá thể, cá thể nào có *thứ hạng (rank)* thấp hơn thì cá thể đó tốt hơn, trong trường hợp cả 2 cá thể cùng *rank,* thì chúng ta sẽ chọn cá thể có *mật độ thưa thớt (crowding distance)* cao hơn hoặc chọn random nếu cả 2 cá thể có cùng *mật độ thưa thớt.*

Bước 2: Sau đó, chúng ta sẽ cho các cá thể đã được chọn lai ghép với nhau với phép lai đồng nhất xác suất là 0,9.

Bước 3: Khi đã các cá thể con, chúng ta sẽ gộp các cá thể cha và các cá thể con rồi tiến hành áp dụng thuật toán *Nondominated Sorting* để chia *rank* cho các cá thể.

Bước 4: Sau khi đã chia rank cho các cá thể, chúng ta sẽ chọn 40 cá thể tốt nhất để làm quần thể mới. Để chọn được các thể tốt, chúng ta sẽ ưu tiên chọn các cá thể có *rank* thấp nhất, trong trường hợp các cá thể cần chọn còn lại không đủ cho một rank thì chúng ta sẽ ưu tiên chọn các cá thể có *crowding distance* cao nhất tương tự ở *bước 2.*

Bước 5: Khi đã có quần thể mới, chúng ta sẽ tiếp tục thực hiện lại từ đầu bước 1 cho đến khi hết *số thể hệ (generation)* mà chúng ta quy định. Đối với thực nghiệm này nhóm đã chọn *số thể hệ* là 100.



Hình 10: Mô hình hoạt động của thuật toán Nondominated Sorting Genetic Algorithm II (NSGA-II)

Nhóm đã chạy thực nghiệm thuật toán NSGA-II với 3 bộ dữ liệu được cung cấp sẵn bởi hàm NAS-Bench-201 tương tự như bài toán đơn mục tiêu. Ứng với mỗi bộ dữ liệu, kết quả thực nghiệm trả về là một *biên không bị thống trị* chứa các kiến trúc tối ưu nhất mà thuật toán đã tìm ra được. Đồng thời khi đã có được các *biên không bị thống trị*, nhóm đã tiến hành đánh giá bằng chỉ số *Inverted Generation Distance (IGD)* so với *biên không bị thống trị* của hàm NAS-Bench-201. Chỉ số *IGD* được tính theo công thức như sau:

$$\mathrm{IGD}(A) = \frac{1}{|Z|}\left(\sum_{i=1}^{|Z|} \hat{d}_i^{\,p}\right)^{1/p}$$

Hình 11: Công thức tính chỉ số IGD

Đối với chỉ số *IGD, biên không bị thống trị* nào chỉ số *IGD* nhỏ hơn thì *biên* đó có kết quả tốt hơn. *Kết* quả thực nghiệm sau khi đánh giá:

Dataset	IGD
CIFAR10-VALID	0.034
CIFAR100	0.046
ImageNet16-120	0.117

Hình 12: Kết quả thực nghiệm đánh số chỉ số IGD của bài toán tối ưu hóa đa mục tiêu

Dựa vào bảng kết quả ở trên, chúng ta có thể thấy được đối với bộ dữ *cifar10 và cifar100* thì thuật toán NSGA-II tối ưu các kiến trúc tốt hơn so với bộ dữ liệu *ImageNet16-210.*

III. Đánh giá chung

Với việc áp dụng NAS-Bench-201 vào bài toán *NAS* giúp cho quá trình đánh giá hiệu suất của các kiến trúc trở nên trực quan hơn và tiết kiệm rất nhiều về mặt chi phí cũng như thời gian thực nghiệm. Điều này giúp ích cho các nhà nghiên cứu sẽ tiếp cận với bài toán *Tìm kiếm kiến trúc mạng neural* cũng như tạo ra các kiến trúc mạng tối ưu được dễ dàng hơn.

IV. Tài liệu tham khảo

[1] L. Weng, "Neural Architecture Search," 6 August 2020. [Online]. Available: https://lilianweng.github.io/lil-log/2020/08/06/neural-architecture-search.html#hierarchical-structure. [Accessed 2 December 2021].

[2] "Neural Architecture Search là gì," 15 April 2020. [Online]. Available: https://ngbao161199.github.io/research/2020/04/15/Neural-Architecture-Search-l%C3%A0-g%C3%AC.html. [Accessed 2 December 2021].