Tìm kiếm kiến trúc mạng bằng tiến hoá tăng cường

Ứng dụng thuật toán tiến hoá tăng cường trong tìm kiếm kiến trúc mạng

Nhóm 8 - Neural Architecture Search

Ngày 17 tháng 11 năm 2023

Mục lục

Tại sao nên kết hợp EA và RL?

1. Như mọi người đã biết thì cả EA và RL đều là những thuật toán rất tốt trong việc tìm kiếm kiến trúc mạng tối ưu. Tuy vậy, không phải lúc nào, EA hay RL cũng tỏ ra hiệu quả và mỗi thuật toán sẽ có những nhược điểm riêng:

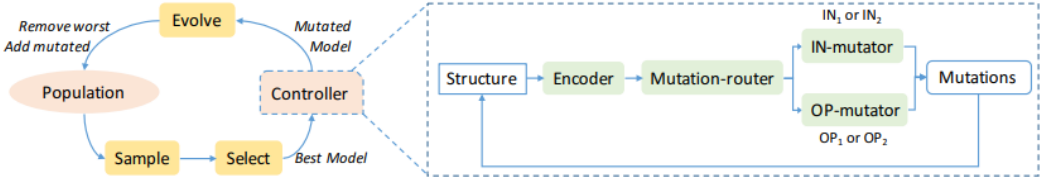
- Khi chỉ sử dụng EA cho NAS, nó có xu hướng là sẽ tiến hoá các kiến trúc mạng sao cho vẫn bảo đảm sự đa dạng hoá quần thể. Tuy nhiên, nó lại bị phụ thuộc quá nhiều vào sự đột biến ngẫu nhiên nên một số trường hợp, độ hiệu quả của EA không được đảm bảo.

- Khi chỉ sử dụng RL cho NAS, nó sẽ phụ thuộc nhiều vào các siêu tham số để đảm bảo tính ổn định. Nhưng khi quyết định kiến trúc mạng cho từng layer, bộ điều khiển RL cần phải thử hàng chục hành động để có thể có được "phần thưởng". Chính điều này có thể làm quá trình train thiếu hiệu quả.

→ Sự kết hợp giữa EA và RL cho NAS được ra đời.

Tích hợp RL vào mô hình thuật toán tiến hoá

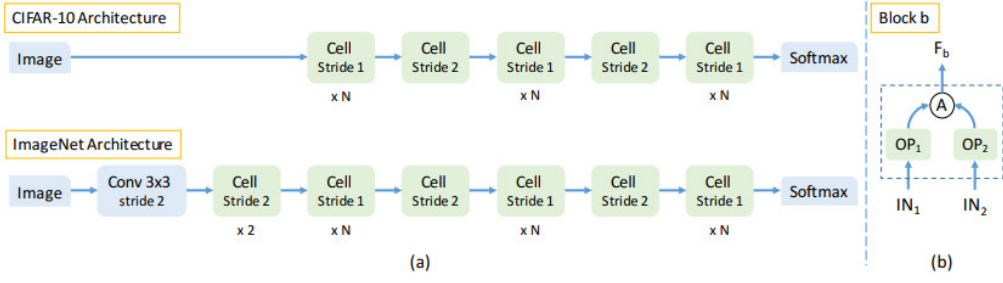
Phương pháp được sử dụng ở đây chính là thiết kế một bộ điều khiển đột biến tăng cường nhằm học tạo ra con đường phù hợp cho việc tiến hoá quần thể, từ đó quần thể có thể tiến hoá tới một phiên bản tốt trong ít vòng lặp hơn.



Hình 1: Mô hình thuật toán tiến hoá cho NAS và cấu trúc bộ điều khiển đột biến tăng cường

Nguyên tắc khởi tạo

Thay vì thiết kế toàn bộ mạng cùng một lúc, chúng ta sẽ học từng kiến trúc cho từng cell. Trong phần này, ta sẽ chia không gian tìm kiếm thành các cells và các blocks.



Hình 2: Mô hình minh hoạ cho CIFAR-10 và ImageNet

Chi tiết mạng - Block

1. Mỗi **block** sẽ bao gồm 2 inputs và 1 output, với input $\{i_1, i_2\}$ là các feature maps, áp dụng 2 phép toán tương ứng với 2 feature maps $\{o_1, o_2\}$ và kết hợp tạo thành ouput O qua element-wise addition A.

2. Mỗi **block** sẽ được chỉ định là một string có độ dài 4, $\{i_1, i_2, o_1, o_2\}$, trong đó:

• $\{i_1, i_2\}$ chọn từ $\left\{O_1^c, O_2^c, O_3^c, ..., O_{b-1}^c, O_B^{c-1}, O_B^{c-2}\right\}$, với $\left\{O_1^c, O_2^c, O_3^c, ..., O_{b-1}^c\right\}$ là ouputs của các blocks trước trong cell hiện tại và $\left\{O_B^{c-1}, O_B^{c-2}\right\}$ là outputs của 2 cells ngay trước cell hiện tại.

• $\{o_1, o_2\}$ chọn từ set 6 function sau: 3×3 depth-wise separable convolution, 5×5 depthwise-separable convolution, 7×7 depth-wise separable convolution, 3×3 avg pooling, 3×3 max pooling, identity.

Chi tiết mạng - Cell, Network

1. Mỗi **cell** là đồ thị có hướng không chu trình chứa $\#B$ block. Giả sử input là feature map $h \times w \times f$, với h, w - độ cao, độ rộng của nó, f là số channels thì cell với stride $= 2$ có output của nó có shape $\frac{h}{2} \times \frac{w}{2} \times 2f$, còn cell với stride $= 1$ thì output của nó sẽ có shape không đổi.

2. Mỗi **mạng** xác định bởi ba yếu tố: cấu trúc cell, số lượng cell($\#N$), và số bộ lọc trong lớp đầu tiên ($\#F$). Khi tìm kiếm, ta cố định $\#N$ và $\#F$, giới hạn không gian tìm kiếm của ta cho tất cả các cấu trúc cell có thể.

3. **Mạng** được xác định bằng $5\#B$ token, với $4\#B$ là số biến trong quá trình tìm kiếm.

Tóm tắt thuật toán

Thuật toán 1: Mô hình của RENAS

Input: số blocks cho mỗi ô $\#B$, số epochs tối đa $\#E$, số filters trong lớp đầu tiên $\#F$, số cells trong mô hình $\#N$, kích thước quần thể $\#P$, kích thước mẫu $\#S$, tập huấn luyện D_{train}, tập xác thực D_{val}.
Output: Quần thể các mô hình P.

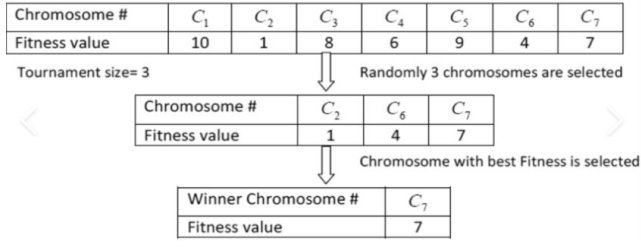
1. $P^{(0)} \leftarrow$ *initialize*($\#F$, $\#N$, $\#P$).
2. for $i = 1 :$ **#E do:**
- $S^{(i)} \leftarrow$ *sample*($P^{(i-1)}$, $\#S$).
- $B, W \leftarrow$ *select*($S^{(i)}$).
- $C, w^B \leftarrow$ *reinforced-mutate*(B).
- $w^C \leftarrow$ *finetune*(C, w^B, D_{train}).
- $f_C \leftarrow$ *evaluate*(C, w^C, D_{val}).
- $P^{(i)} \leftarrow$ *push-pop*($P^{(i-1)}, C, f_C, W$).

3. end

Mô hình

• Để tìm kiếm các kiến trúc mô hình có hiệu suất cao một cách tự động, một quần thể các mô hình P được khởi tạo ngẫu nhiên. Mỗi cá thể trong P được huấn luyện trên bộ dữ liệu huấn luyện D_{train} và được đánh giá trên bộ dữ liệu validation D_{val}.

• Fitness f của mỗi cá thể định nghĩa dựa trên độ chính xác trên tập validation. Ta sẽ chọn cá thể theo phương pháp Tournament Selection.

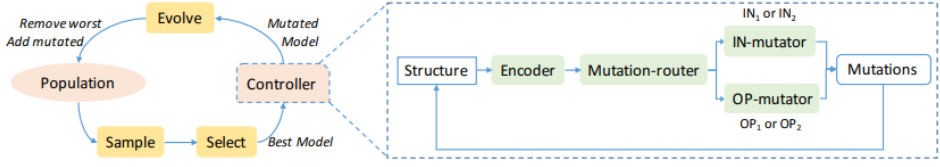


Hình 3: Cách hoạt động của Tournament Selection

Đột biến tăng cường

Reinforced mutation được thực hiện với mutation controller để học sự ảnh hưởng bởi các biến đổi nhỏ và thực hiện đột biến. Controller nhận vào một chuỗi có độ dài $5\#\mathbf{B}$ đại diện cho kiến trúc cell được cung cấp, cụ thể nó bao gồm 4 phần:

1. 1 Encoder (*Enc*) theo sau một lớp nhúng để học ảnh hưởng của mỗi phần của cell.
2. 1 Mutation-router (*Mut-rt*) để chọn một từ $\{i_1, i_2, o_1, o_2\}$ của block.
3. 1 Input-mutator (*IN-mut*) để thay đổi đầu vào của nút bằng một đầu vào mới i_{new}.
4. 1 OP-mutator (*OP-mut*) để thay đổi toán tử của nút bằng một toán tử mới o_{new}.



Hình 4: Mô hình của controller.

Encoder

• Đối với block b trong Enc, các trạng thái ẩn của nó là $\left\{H^b_{i_1}, H^b_{i_2}, H^b_{o_1}, H^b_{o_2}\right\}$, với $H^b_{o_1}$ biểu thị ảnh hưởng của o_1 của b lên toàn bộ mạng. Mỗi bước, Enc tạo ra $5\#B$ trạng thái ẩn. Khởi tạo hai trạng thái ban đầu, H^{c-1}, H^{c-2} - thông tin của 2 cells trước đó.

• Đối với khối b, bộ điều khiển thực hiện 2 quyết định theo trình tự. Ban đầu, dựa vào $\left\{H^b_{i_1}, H^b_{i_2}, H^b_{o_1}, H^b_{o_2}\right\}$, *Mut-rt* quyết định sửa đổi 1 trong i_1, i_2, o_1, o_2 ở block b thông qua *softmax*. Nếu i_1 hoặc i_2 được chọn, thì *IN-mut* sẽ được kích hoạt để chọn một trong $\left\{O^c_1, ..., O^c_{b-1}, O^{c-1}_B, O^{c-2}_B\right\}$. Ngược lại, *OP-mut* sẽ chọn một toán tử mới.

→ Quá trình này được lặp lại B lần để sửa đổi một kiến trúc đã cho.

Mutation-router

• Phần *Mut-rt* được thiết kế để tìm ra thành phần nào trong mỗi block cần được sửa đổi. Đối với mỗi khối, đầu vào của *Mut-rt* là một tập con của đầu ra của Enc là $\left\{H^b_{i_1}, H^b_{i_2}, H^b_{o_1}, H^b_{o_2}\right\}$ và đầu ra của nó là một trong các ID để đột biến: i_1, i_2, o_1, o_2.

• Áp dụng một lớp kết nối đầy đủ cho mỗi trạng thái ẩn và sử dụng softmax để tính toán xác suất sửa đổi cho mỗi thành phần $\left\{P^b_{i_1}, P^b_{i_2}, P^b_{o_1}, P^b_{o_2}\right\}$ và lấy mẫu một trong i_1, i_2, o_1, o_2 với những xác suất này.

IN-mutator và OP-mutator

1. IN-mutator

• *IN-mut* chọn đầu vào mới cho nút, nếu ID $\in (i_1, i_2)$. Đầu vào gồm trạng thái ẩn H^b_{ID} của ID, trạng thái ẩn của tất cả đầu ra của các khối trước $[H^1_A, ..., H^{b-1}_A]$, và trạng thái ẩn của 2 cell trước H^{c-1}, H^{c-2}.

• Nối $[H^1_A, ..., H^{b-1}_A, H^{c-1}, H^{c-2}]$ với H^b_{ID} và áp dụng một lớp kết nối đầy đủ cho chúng. Sử dụng softmax để tính toán xác suất thay thế đầu vào gốc bằng mỗi đầu vào thay thế và sau đó xác định i_{new} bằng cách chọn từ $1, ..., b-1, c-1, c-2$ với những xác suất này.

2. OP-mutator

• *P-mut* xuất ra một toán tử mới o_{new} phụ thuộc vào đầu vào H^b_{ID}. Quá trình tương tự như *Mut-rt*.

Đột biến tạo bởi Controller

Algorithm 2: Mutation generated by Controller

input : num blocks each cell `#B`, a sequence a of `4#B` number specifying a cell

output: a sequence of mutation actions m

$H^{c-1}, H^{c-2} \leftarrow$ `Enc.begin()`

$H^1, ..., H^B \leftarrow$ `Enc`(H^{c-1}, a)

for b=*1:#B* **do**

  $H^b_{i_1}, H^b_{i_2}, H^b_{o_1}, H^b_{o_2}, H^b_A \leftarrow H^b$

  $ID \leftarrow$ `Mut-rt`$([H^b_{i_1}, H^b_{i_2}, H^b_{o_1}, H^b_{o_2}])$

  if $ID \in (i_1, i_2)$ **then**

    $i_{new} \leftarrow$ `IN-mut` $(H^b_{ID}, [H^1_A, ..., H^{b-1}_A, H^{c-1}, H^{c-2}])$

    $m^{(b)} \leftarrow (ID, i_{new})$

  else

    $o_{new} \leftarrow$ `OP-mut` (H^b_{ID})

    $m^{(b)} \leftarrow (ID, o_{new})$

  end

end