Hierarchical Representations for Efficient Architecture Search

Lucas Becirspahic, Stieban Fernandez

Sorbonne Université Sciences, Paris VI

Points clés de l'article

- Définition d'une structure hiérarchique pour décrire l'architecture d'un réseau de neurones.
- Une architecture compétitive pour la classification d'images peut être obtenue même avec des méthodes simples telles que le Random Search.
- Mise en place d'un algorithme évolutionniste distribué pour trouver la meilleure architecture.

Introduction

Le domaine du traitement de l'image a vu l'apparition d'architectures de réseau de neurones profond à base de motifs, donnant des résultats très impressionnants.

(a) ResNet (b) GoogleNet

Figure – Architectures de réseau de neurones réalisé par des humains

Néanmoins, déterminer une architecture de réseau de neurones performante est une tâche difficile qui nécessite beaucoup d'expertises humaines. Plusieurs algorithmes de recherche d'architecture ont vu le jour utilisant par exemple une approche Monte Carlo ou encore le Reinforcement Learning.

L'objectif de ce papier est de montrer qu'un algorithme évolutionniste simple peut permettre de trouver une architecture performante et en moins de temps.

Structure hiérarchique

Une architecture est représentée sous la forme d'un graphe dirigé acyclique (DAG) où un noeud correspond à un features map et un arc à une opération. On représente l'architecture par (G, o) avec G la matrice d'adjacence où $G_{ij} = k$ signifie que l'on effectue la k ième opération entre les noeuds i et j, et o l'ensemble des opérations.
La structure hiérarchique consiste en la création de motifs à un niveau l en assemblant des motifs du niveau l-1. L'assemblage est effectué de sorte à avoir un calcul séquentiel en ordonnant les noeuds. Chaque motif d'un niveau correspond à une opération au niveau supérieur. Les motifs du niveau 1 correspondent alors à des opérations primitives. En pratique, ces dernières sont :

- 1×1 convolution de C filtres
- 3×3 depthwise convolution
- 3×3 separable convolution de C filtres
- 3×3 max pooling
- 3×3 average pooling
- identité (retourne l'entrée)

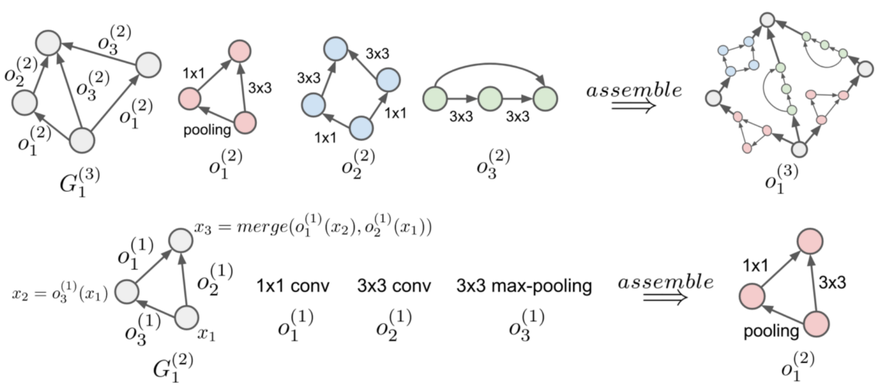


Figure – Exemple de représentation d'une architecture à trois niveaux

Recherche évolutionniste

Une architecture correspond à un *genotype*. Effectuer une *mutation* sur un génotype consiste alors à tirer aléatoirement une opération entre deux noeuds i et j à un niveau l et de la remplacer par une autre opération au niveau l-1. Le processus est suffisamment puissant pour simuler l'ajout, la modification et la suppression d'un arc.

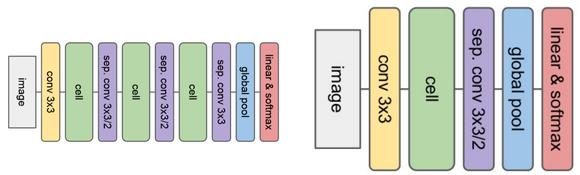
Initialisation

1. Créer une population de genotypes où toutes les opérations correspondent à l'identité.
2. Effectuer un grand nombre de mutations pour chacun des génotypes (~1000)

Algorithme de recherche

Nous avons maintenant une population de génotypes aléatoires.

1. Population de génotypes aléatoires
2. Prendre les plus prometteurs et placer leurs descendants dans la population
3. Apprentissage et évaluation des génotypes
4. Le génotype avec le meilleur *fitness* est sélectionné pour la sortie final après un certain temps.



(a) small CIFAR-10 (article) (b) Notre modèle

Figure – Architectures de réseau de neurones réalisé par des humains

Le modèle proposé par les auteurs nécessite une immense puissance de calcul (200 GPU workers et 1.5 jours de calcul). Nous proposons donc un modèle simplifié moins coûteux. D'après les auteurs, le Random Search obtient seulement 0.3 % d'accuracy de moins que l'algorithme évolutif et se finit en une heure. C'est pourquoi, nous avons implémenté le Random Search qui est plus économe en calcul.

Resultats

Nous avons réalisé nos expériences sur le dataset CIFAR-10.

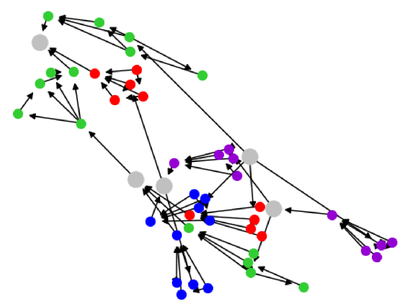
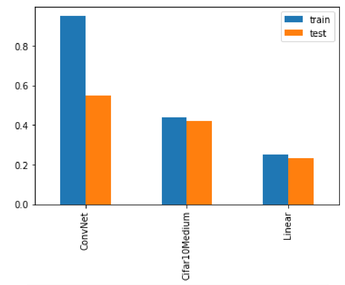


Figure – Architecture générée par l'algorithme



	train	test
ConvNet	0.95	0.55
Cifar10 medium	0.44	0.42
Linear	0.25	0.23

Figure – Résultats des expériences

Model	**Error (%)**
ResNet-1001 + pre-activation (He et al., 2016b)	4.62
Wide ResNet-40-10 + dropout (Zagoruyko & Komodakis, 2016)	3.8
DenseNet (k=24) (Huang et al., 2016)	3.74
DenseNet-BC (k=40) (Huang et al., 2016)	3.46
MetaQNN (Baker et al., 2016)	6.92
NAS v3 (Zoph & Le, 2016)	3.65
Block-QNN-A (Zhong et al., 2017)	3.60
NASNet-A (Zoph et al., 2017)	3.41
Evolving DNN (Miikkulainen et al., 2017)	7.3
Genetic CNN (Xie & Yuille, 2017)	7.10
Large-scale Evolution (Real et al., 2017)	5.4
SMASH (Brock et al., 2017)	4.03
Evolutionary search, hier. repr., $c_0 = 64$	3.75 ± 0.12
Evolutionary search, hier. repr., $c_0 = 128$	3.63 ± 0.10

Figure – Résultats des auteurs du papier