The 33rd Annual Conference of the Japanese Society for Artificial Intelligence, 2019

貢献度分配を導入した方策勾配による Neural Architecture Search の高速化

Acceleration of Neural Architecture Search by Policy Gradient with Credit Assignment

佐藤 怜 *1　秋本 洋平 *1　佐久間 淳 *1
Rei Sato　Youhei Akimoto　Jun Sakuma

*1筑波大学　*1理化学研究所 革新知能統合研究センター
University of Tsukuba　RIKEN Center for Advanced Intelligence Project

Neural architecture search is gathering attention as an approach to automatically designing the architecture of deep neural networks. Aiming at accelerating neural architecture search without compromising its performance, we propose a novel one-shot architecture search algorithm that optimizes the weights and the architecture of a network simultaneously. Our algorithm is inspired by the notion of credit assignment in policy gradient. An advantage of each architecture component is defined and the gradient of the architecture parameter is computed using the advantage values. In our experiments, we observe that the proposed method achieves, with lower computational time, the final performance comparable to recently proposed gradient-based one-shot architecture search algorithms.

1. はじめに

深層学習と呼ばれる一連の技術が画像認識や音声認識をはじめとする種々のタスクに適用され, 目覚ましい精度向上を成し遂げている. 深層学習ではモデルのアーキテクチャと呼ばれるハイパーパラメータ (e.g. 畳み込み層のフィルタ数, Pooling の種類) が所望のタスクに対する精度に大きく寄与する. しかしアーキテクチャは数十から数百にもなるハイパーパラメータの組み合わせで定義され, これらの組み合わせを改良していく作業には多大な時間と労力を要する. このような背景から近年, 深層学習モデルのアーキテクチャをアルゴリズムによって最適化する研究が盛んになっている.

アーキテクチャの最適化は, 精度 (あるいは精度と計算コストのトレードオフを考慮した指標) を最大化するアーキテクチャを探索する問題として定式化され, 強化学習や遺伝的アルゴリズム等を用いて行われることが多い. しかし異なるアーキテクチャの深層学習モデルは異なるパラメータ (e.g. 重み, バイアス) を持つため, これらを学習させない限りそのアーキテクチャを評価することができない. 従って, 従来のアーキテクチャ探索では候補となるアーキテクチャに対応するモデルを学習し, 評価するというサイクルを繰り返す必要があり, 膨大な学習コストが問題となっていた.

このような課題に対して近年, アーキテクチャとパラメータを同時に最適化することでパラメータの学習コストを削減する手法が複数提案されている. これらの手法は 1 度 (あるいは再学習を含めて 2 度) の学習プロセスで両者を最適化することから one-shot アーキテクチャ探索と呼ばれている.

本研究では one-shot なアプローチのうち, 勾配法に基づく探索法 [Liu 19][Xie 19] に着目する. これらの研究では, 従来カテゴリカル変数として表現されていたアーキテクチャを連続変数を用いて表すことで, 微分可能な目的関数を定義する. 勾配法による最適化は, 遺伝的アルゴリズムを用いた探索手法と異なり, 探索空間に合わせた手法の設計が不要であり, 実装が容易であるというメリットを持つため, 注目を集めている.

本研究では one-shot でアーキテクチャを探索する手法の高速化を目的とし, 強化学習の Actor-Critic 法に着想を得た勾配法による探索手法を提案する. ここでの高速化とは, アーキテクチャ探索にかかる総時間の削減を意味しており, 提案手法は 1 イテレーションの高速化とイテレーション数の削減の両面からアプローチする.

数値実験により, 勾配法に基づく one-shot アーキテクチャ探索を行う先行研究と比較して, 同程度の精度を得られるアーキテクチャをより高速に発見することができることを示す.

連絡先: 佐藤怜, 筑波大学情報科学類, 茨城県つくば市天王台 1-1-1, reisato@bbo.cs.tsukuba.ac.jp

2. Neural Architecture Search

2.1 有向非巡回グラフとしての DNN

本稿では DARTS[Liu 19] の定式化に倣い, 深層学習モデル (以下 DNN) を図 3 のような有向非巡回グラフと見なし, 各ノードを特徴 x, 各エッジをオペレーション $\mathcal{O}$ と捉える.

順伝播を行う際は, まずこの有向非巡回グラフの入力ノードにデータ (e.g. 画像ベクトル) を代入する. 以降はエッジの接続に基づいてエッジに割り当てられたオペレーション $\mathcal{O}$ で特徴 x を変換して順伝播する.

エッジ $i \in \mathcal{I}$ のオペレーション $\mathcal{O}$ で変換した出力の特徴を y_i と表記する. あるノード ℓ の状態となる特徴 x_ℓ は, ノード ℓ の入力となるエッジの集合を $\mathcal{E}_\ell$ として, $x_\ell = F(\{y_l | l \in \mathcal{E}_\ell\})$ と計算する. 関数 F は複数の特徴を受け取り, 1 つの特徴を返す何らかの関数 (e.g. 和, 結合) である. エッジ i の入力となるノードの添字を $Q(i)$ と表記する. すなわち, エッジ i のオペレーション $\mathcal{O}$ の入力となる特徴は $x_{Q(i)}$ と表記される.

エッジ i でのオペレーション候補の添字の集合を $\mathcal{J}_i$ とする. 本稿では全てのエッジで候補となるオペレーションの数は同じとし, 以降は簡単のため $\mathcal{J} = \mathcal{J}_i$ と表記する. また, エッジ i での $j \in \mathcal{J}$ 番目のオペレーション候補を $\mathcal{O}_j^i$ と表記する.

このような有向非巡回グラフは, エッジに割り当てるオペレーション $\mathcal{O}$ に多様な関数 (e.g. Convolution, Pooling) を選ぶことで, 幅広い DNN アーキテクチャを表現することができる. この有向非巡回グラフの接続は予め決められているが, オペレーション候補に zero-operation(零写像) を含めることで, ノード間の接続の不在も表現できる.

本稿ではアーキテクチャ探索を, DNN 上の $|\mathcal{I}|$ 箇所のエッジにおいて, それぞれ $|\mathcal{J}|$ 個のオペレーション候補から最終的に 1 つを選択する問題として取り扱う. このような各エッジに

ついて 1 つのオペレーションのみを選択し実行するアーキテクチャをバイナリベクトル $a \in \{0,1\}^{|\mathcal{I}||\mathcal{J}|} = \mathcal{A}$ で表現する. $a_i \in \{0,1\}^{|\mathcal{J}|}$,$|a_i| = 1$ は one-hot ベクトルとして各エッジのアーキテクチャを表現しており, a_{ij} が 1 のとき, エッジ i ではオペレーション j が選択されていることを意味する.

従って, エッジの出力 y_i は以下で与えられる.

$$y_i = \sum_{j \in \mathcal{J}} a_{ij} \mathcal{O}_j^i(x_{Q(i)}) \tag{1}$$

2.2 One-shot Architecture Search

関数 $R(a; w_a)$ は, 広義にはアーキテクチャa と a に対応する学習済みパラメータ w_a を受け取り, a の評価を返す関数である. 従来は w_a を得るために a に基づくモデルを学習していたが, 本稿ではこの学習コストの削減を目的として one-shot なアーキテクチャ探索を考える. 本稿で探索時に用いる DNN の各エッジは, そのエッジにおける全てのオペレーション候補が必要とするパラメータを保存する. この冗長な DNN のパラメータは誤差逆伝播を用いてアーキテクチャ変数と同時に更新される. この冗長なモデルのパラメータを $\hat{w}$ と表記すると, あるアーキテクチャa を評価するのに必要な学習済みパラメータ w_a は $\hat{w}$ の一部として得られる. 表記法のわずかな乱用を許せば, $R(a; w_a)$ は $R(a; \hat{w})$ と置き換えられ, a が変化した際に w_a を再学習する必要がなくなるため, 学習コストを大幅に削減できる. 簡単のため, 以降は $R(a; \hat{w})$ を $R(a)$ と表記する.

本稿では画像認識モデルのアーキテクチャ探索を対象とするため, $R(a)$ には a に基づいて適当なミニバッチを伝播した際の負の交差エントロピー誤差関数を採用する.

3. 関連研究

本章では, one-shot アーキテクチャ探索に関する既存研究を紹介する. 本稿で扱う探索手法は勾配が確率変数の期待値で定義される場合があるが, 本研究では先行研究 [Pham 18][Cai 19] に倣い期待値をサンプル数 1 のモンテカルロ法で近似する.

3.1 ENAS[Pham 18]

ENAS[Pham 18] は, REINFORCE[Williams 92] を用いたアーキテクチャ探索法である. 関数 $p_\theta(a)$ を θ でパラメトライズされた $\mathcal{A}$ 上の確率分布とする. REINFORCE を用いた探索は, 目的関数 $\mathbb{E}_{a \sim p_\theta(a)}[R(a)]$ を θ について最大化する最適化問題として定式化される. この最適化を行うために, 下式で目的関数の勾配を計算する.

$$\nabla_\theta \mathbb{E}_{a \sim p_\theta(a)}[R(a)] \quad = \quad \mathbb{E}_{a \sim p_\theta(a)}[R(a) \nabla_\theta \log(p_\theta(a))] \tag{2}$$

最終的に, $p_\theta(a)$ が最大となる $a \in \mathcal{A}$ が探索結果のアーキテクチャとなる.

3.2 DARTS[Liu 19]

DARTS[Liu 19] では $\theta \in \mathbb{R}^{|\mathcal{I}||\mathcal{J}|}$ とする. DARTS では $\mathcal{A}$ 上の関数として定義されていた評価関数 R を $[0,1]^{|\mathcal{I}||\mathcal{J}|}$ 上へ拡張する. これは式 (1) の $a \in \mathcal{A}$ を, 以下で定義される operation mixing weights $\mu(\theta)$ で置き換えることで実現される.

$$\mu_{ij}(\theta) = \frac{\exp(\theta_{ij})}{\sum_{k \in \mathcal{J}} \exp(\theta_{ik})} \tag{3}$$

DARTS では目的関数 $R(\mu(\theta))$ の θ についての最大化を行う. この最適化を行うために, 次のように勾配を計算して θ を更新する.

$$\nabla_\theta R(\mu(\theta)) = \frac{\partial R(\mu(\theta))}{\partial \mu(\theta)} \frac{\partial \mu(\theta)}{\partial \theta} \tag{4}$$

最終的に, エッジ i のオペレーションには $\mu_{ij}(\theta)$ が最大となる j が選択され, これが探索結果のアーキテクチャとなる. このアーキテクチャを表現する a は $a_{ij} = \lim_{\lambda \to 0}[\mu_{ij}(\theta/\lambda)]$ で得られる. 本稿ではこの操作を one-hot 化と呼ぶ.

3.3 SNAS[Xie 19]

SNAS[Xie 19] は DARTS と同様に連続値のアーキテクチャ変数を考えるが, その方法が異なる.

SNAS では G_{ij} を標準 Gumbel 乱数, λ_t をステップ t での温度パラメータ ($\lim_{t \to \infty} \lambda_t = 0$), $\theta \in \mathbb{R}^{|\mathcal{I}||\mathcal{J}|}$ として, 次のように確率的なアーキテクチャ変数を定義する.

$$m_{ij}(\theta, G) = \frac{\exp((-\log(\mu_{ij}(\theta)) + G_{ij})/\lambda_t)}{\sum_{k \in \mathcal{J}} \exp((-\log(\mu_{ik}(\theta)) + G_{ik})/\lambda_t)} \tag{5}$$

SNAS では $\mathbb{E}_G[R(m(\mu(\theta), G))]$ の θ に関する最大化を考える. 各エッジでの順伝播は, 式 (1) の a_{ij} の代わりに式 (5) で計算した $m_{ij}(\theta)$ を用いて行う. 勾配は下式で計算される.

$$\begin{aligned} &\nabla_\theta \mathbb{E}_G[R(m(\mu(\theta), G))] \\ &\quad = \mathbb{E}_G \left[\frac{\partial R(m(\mu(\theta), G))}{\partial m(\mu(\theta), G)} \frac{\partial m(\mu(\theta), G)}{\partial \mu(\theta)} \frac{\partial \mu(\theta)}{\partial \theta} \right] \end{aligned} \tag{6}$$

探索結果のモデルは DARTS と同様にして得られる.

3.4 ProxylessNAS[Cai 19]

$\theta \in \mathbb{R}^{|\mathcal{I}||\mathcal{J}|}$ とし, $z_i(\mu(\theta), N)$ を $\mu(\theta)$ と乱数 N を受け取り two-hot ベクトル $z_i(\mu(\theta), N) \in \{0,1\}^{|\mathcal{J}|}, |z_i(\mu(\theta), N)| = 2$ を出力する関数として定義する. この関数値 $z_{ij}(\mu(\theta), N)$ は, $\mu_{ij}(\theta)$ を確率パラメータとするサンプル数 2 の多項分布の実現値だが, 必ず $z_i(\mu(\theta), N)$ が two-hot ベクトルになるよう修正を行う.

ProxylessNAS[Cai 19] では $\mathbb{E}_N[R(z(\mu(\theta), N))]$ の θ についての最大化を考える. 各エッジでの順伝播は式 (1) の a_{ij} の代わりに $z_{ij}(\mu(\theta), N)$ を用いて行う. 勾配は次のように計算する.

$$\begin{aligned} &\nabla_\theta \mathbb{E}_N[R(z(\mu(\theta), N))] \\ &\quad = \mathbb{E}_N \left[\frac{\partial R(z(\mu(\theta), N))}{\partial z(\mu(\theta), N)} \frac{\partial z(\mu(\theta), N)}{\partial \mu(\theta)} \frac{\partial \mu(\theta)}{\partial \theta} \right] \\ &\quad \approx \mathbb{E}_N \left[\frac{\partial R(z(\mu(\theta), N))}{\partial z(\mu(\theta), N)} \frac{\partial \mu(\theta)}{\partial \theta} \right] \end{aligned} \tag{7}$$

各イテレーションで θ の更新を行った後, $z_{ij}(\mu(\theta), N) = 0$ となる ij について $\mu_{ij}(\theta)$ が更新されないよう後処理を行う. 探索結果のモデルは DARTS, SNAS と同様にして得られる.

3.5 計算コストの比較

本研究ではアーキテクチャ探索の総探索時間の削減に着目しており, このうち 1 イテレーションの計算時間の観点から以上の先行研究を比較する. DARTS, SNAS のような連続値のアーキテクチャ変数を扱う手法は, 式 (1) の順伝播において各エッジで $|\mathcal{J}|$ 個のオペレーション候補の出力を計算する必要があるため, 1 イテレーションの計算コストが $|\mathcal{J}|$ に依存するという欠点を持つ. 一方, REINFORCE や ProxylessNAS のような離散値のアーキテクチャをサンプリングする手法の場合, 式 (1) では常に 1 つまたは 2 つのオペレーション候補の出力を計算するため, 1 イテレーションの計算コストが $|\mathcal{J}|$ に依存しないという利点がある.

4. 提案手法

本研究では REINFORCE に貢献度分配の考え方を導入することで効率的にアーキテクチャの確率分布を更新する手法を提案する．提案手法はサンプリングベースの手法である REINFORCE を改良したものであり，DARTS や SNAS とは異なり 1 イテレーションの計算コストが $|\mathcal{J}|$ に依存しない．

4.1 貢献度

関数 $p_{\theta_i}(a_i)$ を $\theta_i \in \mathbb{R}^{|\mathcal{J}|}$ でパラメトライズされたカテゴリ分布とする．つまり，オペレーション候補が各エッジについて独立にサンプリングされるようなアーキテクチャの分布を考える．ここで，$a_{ij} = 1$ となる確率は $\mu_{ij}(\theta)$ で与えられる．最適化のプロセスでは，まず全てのエッジ $i \in \mathcal{I}$ について $p_{\theta_i}(a_i)$ から one-hot な列ベクトル a_i を生成し，エッジ i での順伝播を式 (1) で計算する．

ここで，各エッジのアーキテクチャの分布 $p_{\theta_i}(a_i)$ が独立であることに注意すると，REINFORCE で計算される θ への勾配は下式である．

$$\mathbb{E}_{a\sim p_\theta(a)}[R(a)\nabla_\theta \log(p_\theta(a))] \tag{8}$$

$$= \mathbb{E}_{a\sim p_\theta(a)}\left[R(a)\nabla_\theta \log(\prod_{i\in\mathcal{I}} p_{\theta_i}(a_i))\right] \tag{9}$$

$$= \mathbb{E}_{a\sim p_\theta(a)}\left[\sum_{i\in\mathcal{I}} R(a)\nabla_{\theta_i} \log(p_{\theta_i}(a_i))\right] \tag{10}$$

上式より，REINFORCE では全てのエッジの確率分布 $p_{\theta_i}(a_i)$ を更新する勾配を，アーキテクチャ全体の評価 $R(a)$ でスケールしていることがわかる．

ここで，REINFORCE の探索の効率化を目的として，貢献度分配の考え方を導入する．アーキテクチャは複数のエッジの集合だから，実行結果に基づくアーキテクチャ全体の評価 $R(a)$ が良くても (悪くても)，各エッジではところどころ悪い (良い) オペレーション候補が実行されているはずだと考えられる．そこで，アーキテクチャ全体の評価 $R(a)$ ではなく，特定のエッジの評価だけを計算し，これを各エッジの評価として方策勾配を計算することを考える．各エッジの評価で方策勾配を計算することで $p_{\theta_i}(a_i)$ が的確に更新され，探索を高速化 (すなわち同じイテレーション数で探索を行ったときにより良いアーキテクチャを発見) できると考える．

$S_i \in \{0,1\}^{|\mathcal{I}-1||\mathcal{J}|}$ を，S_i と a_i を連結することで a が復元されるような a の要素の集合として，$S_i = \{a_g | g \in \mathcal{I}, g \neq i\}$ と定義する．あるエッジで選択されたアーキテクチャを表す one-hot ベクトル a_i の評価を返す関数を貢献度と呼ぶことにし，これを次のように定義する．

$$A(a_i, S_i) = R(a_i, S_i) - R(\vec{0}, S_i) \tag{11}$$

式 (11) は，a 全体の評価から S_i の評価を引いたものは，あるエッジ a_i だけの評価に相当するという解釈に基づく．

4.2 効率的な貢献度の計算

定義 (11) に基づいて貢献度を計算すると R を複数回評価する必要があり計算コストが課題になる．そこで次のように $R(a)$ の a_i 方向への方向微分を考え貢献度を近似する．

$$A(a_i, S_i) = R(a_i, S_i) - R(\vec{0}, S_i)$$

$$= \frac{R(a_i, S_i) - R(a_i - \Delta a_i, S_i)}{\Delta}|_{\Delta=1} \approx \frac{\partial R(a)}{\partial a_i^t} a_i \tag{12}$$

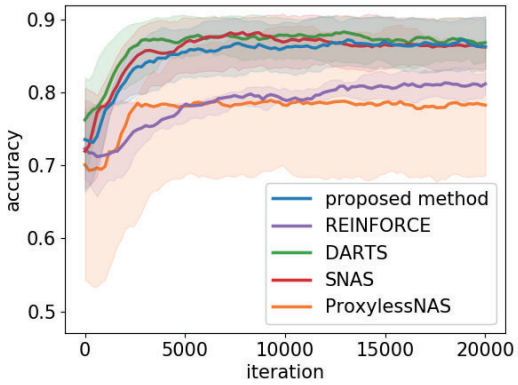

図 1: 探索過程で one-hot 化したモデルの accuracy

a は $\{0,1\}^{|\mathcal{I}||\mathcal{J}|}$ を取るが，微分を計算するために擬似的に連続空間での点として取り扱う．近似した式 (12) は 1 度の順逆伝播で計算できるため，式 (11) と比較して計算効率が向上している．

以上の議論から，θ_i を更新する勾配を式 (14) で計算する．

$$\mathbb{E}_{a\sim p_\theta(a)}\left[A(a_i, S_i)\nabla_{\theta_i} \log(p_{\theta_i}(a_i))\right] \tag{13}$$

$$\approx \mathbb{E}_{a\sim p_\theta(a)}\left[\frac{\partial R(a)}{\partial a_i^t} a_i \nabla_{\theta_i} \log(p_{\theta_i}(a_i))\right] \tag{14}$$

5. 実験

貢献度分配の効果の確認や類似手法との比較を行うため，2 種類の実験を行う．まず，各手法の最適化性能を比較するため，ニューラルネットの重みを固定し，アーキテクチャの探索のみ行う．次に CIFAR-10 データセットを用いて，one-shot アーキテクチャ探索としての各手法の比較を行う．

5.1 Toy Problem

まず Toy Problem での実験を行う． 比較対象は REINFORCE, DARTS, SNAS, ProxylessNAS, 提案手法の 5 つである．教師モデルを 2 層 (入力ノード数 500, 中間ノード数 500 出力ノード数 2) の全結合ニューラルネットとし，中間層には活性化関数を適用する．活性化関数には x,x^2,x^3,ReLU(x),sigmoid(x) の 5 つの候補を用意する．教師モデルの各中間ノードの活性化関数は予めランダムに選択する．

入力として，ミニバッチサイズ 100 の一様乱数を毎イテレーション生成する．生徒モデルは教師モデルと同じ構造の全結合ニューラルネットとし，教師モデルと等しい重みを与える．重みは学習せず，生徒モデルの中間層の活性化関数を探索することで教師モデルとのソフトマックス後の出力の交差エントロピー誤差を最小化する．手法毎の勾配ノルムの違いを吸収するために，アーキテクチャ変数の更新には Adam を用いる．

5.1.1 結果

Toy Problem の実験を行った際の探索中の accuracy を図 1 に示す．線は 10 試行の中央値で，領域は四分位範囲である．ここで，DARTS, SNAS, ProxylessNAS は複数の候補の重み付き和を計算する手法だが，全ての手法で one-hot 化を行って accuracy を測っている．提案手法は DARTS, SNAS に迫る効率的な探索を行えていることがわかる．ここではイテレーション数で比較をしているが，実際のアーキテクチャ探索では 1 イテレーションにかかる時間が DARTS, SNAS に比べ $1/|\mathcal{J}|$ 程度に減少するため，総探索時間の短縮が期待される．

表 1: 探索によって得られたモデルの比較. accuracy は再学習の結果. 提案手法と REINFORCE は 3 試行の結果

method	#param(K)	accuracy	time(hour)
提案手法	509 ± 3	92.34 ± 0.21	25.83 ± 0.84
REINFORCE	486 ± 10	91.85 ± 0.16	22.85 ± 0.81
DARTS	508	92.16	79.77
SNAS	512	92.11	75.63
ProxylessNAS	473	91.12	37.44
ランダム (5 回)	473 ± 5	91.37 ± 0.54	0

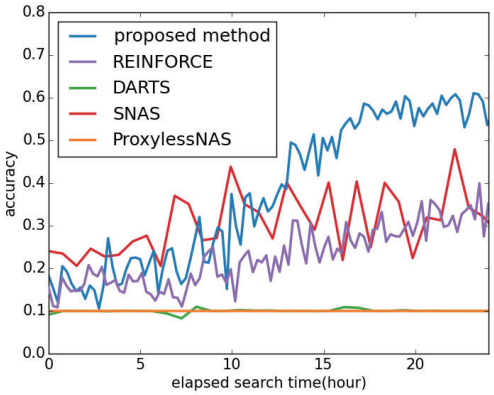

図 2: 探索過程で one-hot 化したモデルの test accuracy

5.2 アーキテクチャ探索

次に, 実際のアーキテクチャ探索で各探索手法の性能を比較する. 探索空間やハイパーパラメータについては全ての手法で [Liu 19] に準ずる. 詳細を以下に示す.

DNN はセルと呼ばれるマイクロアーキテクチャの 8 つの積み重ねで構成する. 各セルは 14 箇所のエッジと 7 箇所のノードを持つ. c 番目のセルには $c-1$, $c-2$ 番目のセルの出力が入力され, これがセルの 1, 2 番目のノードの特徴 x となる. セルの 7 番目のノードの特徴は当該セルの出力として以降のセルの入力となる. 実験では normal cell と reduction cell(セルの 1, 2 番目のノードを入力とするオペレーションのストライドが 2) の 2 種類の探索を行う. 2 種類のセルの計 28 箇所のエッジにおいて, 3*3/5*5 separable convolution, 3*3/5*5 dilated separable convolution, 3*3 max pooling, 3*3 average pooling, identity mapping(恒等写像), zero-operation(零写像) の計 8 つの候補から 1 つを選択する. 各 Convolution の入出力チャネル数は 16 とする. データセットは CIFAR-10 を用いる.

最適化は 150 エポックで行うが, 候補選択における重みの初期値への依存を減らすために, いずれの手法も最初の 50 エポックでは θ の更新を行わない. また, 重みの学習には安定性のために GradientClipping(1.0) を適用した.

提案手法を用いたアーキテクチャ探索では, 計算コストの削減を目的としていない場合でもスパースなモデルが得られる傾向にある. これは, zero-operation の貢献度は恒等的に 0 であるが, 他のオペレーションは実験的に負の貢献度を得る場合が多いため, zero-operation が選ばれやすいことに起因する. ここで直感的には, zero-operation は計算コスト 0 で識別には貢献しない候補だから, 最も低い貢献度を得ている他の候補に近い貢献度を与える. これを $\min_j \ \mathbb{E}_{a\sim p_\theta(a)}[\frac{\partial R(a)}{\partial a_{ij}} a_{ij}]$ のように定義し, エッジ i の zero-operation の貢献度として与えた. 期待値は指数移動平均で近似した.

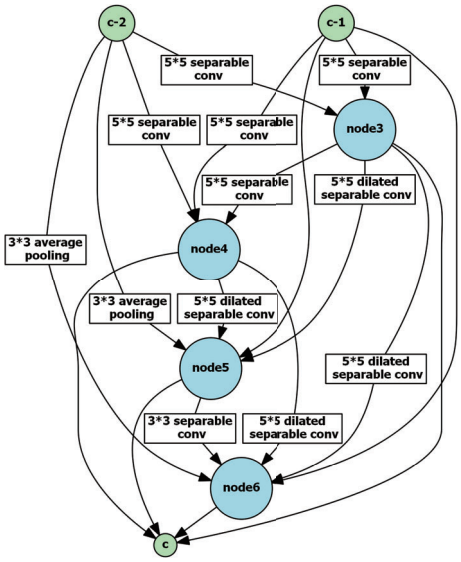

図 3: 提案手法を用いて得られた reduction cell

5.2.1 結果

探索結果で得られたモデルを one-hot 化し, 300 エポックで重みを再学習する. 重みの再学習は従来研究においても実施されている. この結果を表 1 に示す. 提案手法による探索では, DARTS, SNAS より短い探索時間でより高い精度のモデルを発見することができた. また, 探索中にアーキテクチャを one-hot 化した際の accuracy を図 2 に示す. 提案手法は先行研究と比較して同程度の探索時間でより accuracy の高い a とパラメータ w_a の組を得ていることがわかる. しかしいずれの手法においてもより高い accuracy を得るためには再学習が必要であることが示唆されている. 図 3 には, 提案手法を用いて探索を行った結果の reduction cell を図示する.

6. おわりに

本研究では, アーキテクチャ探索の高速化を目的として方策勾配に貢献度分配を導入した手法を提案した. 提案手法は DARTS, SNAS と比較して 1 イテレーションの計算コストが候補数 $|\mathcal{J}|$ に対して定数になるというメリットがあり, イテレーション回数で比較した際もこれらに迫る性能を示した.

参考文献

[Cai 19] Cai, H., Zhu, L., and Han, S.: ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware, in *ICLR* (2019)

[Liu 19] Liu, H., Simonyan, K., and Yang, Y.: DARTS: Differentiable Architecture Search, in *ICLR* (2019)

[Pham 18] Pham, H., Guan, M., Zoph, B., Le, Q., and Dean, J.: Efficient Neural Architecture Search via Parameters Sharing, in *ICML*, pp. 4095–4104 (2018)

[Williams 92] Williams, R. J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning, *Machine Learning*, Vol. 8, No. 3, pp. 229–256 (1992)

[Xie 19] Xie, S., Zheng, H., Liu, C., and Lin, L.: SNAS: stochastic neural architecture search, in *ICLR* (2019)