DARTS を用いた VGG のショートカット探索と GA による改良

1　はじめに

機械学習の分野では, 深層学習モデルの改良によって高い精度を得てきた. しかしモデルの設計とその性能の関係はブラックボックスであり手作業で行うチューニングには膨大な労力を要する.

ネットワークの探索を自動化する手法として提案された Neural Architecture Search(NAS) はネットワークを機械学習によって探索する. しかし何千もの GPU を必要とするため, NAS に代わり小規模な資源で計算できる Differentiable Architecture Search(DARTS) が大きな注目を集めている. DARTS はネットワークの構造と演算子の候補を探索するが, 一方で DARTS にはネットワーク構造にいくつかの拘束条件がある.

本研究では演算子の種類ではなくネットワークの構造にのみ着目し, DARTS の構造制限をなくしネットワークの柔軟な探索を目的とする. その初期段階として, VGG のショートカット位置について DARTS で探索を行う方法を提案する.

2　要素技術

2.1　Neural Architecture Search

Neural Architecture Search(NAS)[1] は, 機械学習の分野で使用されているニューラルネットワークの設計を自動化する手法である. ニューラルネットワークの設計は直感的でなく, チューニングに人による労力を多く必要とするため, ニューラルネットワークの設計は非常に困難である.

NAS はニューラルネットワークが構造に関する設定の文字列で表現できることを利用して, この文字列を生成する Recurrent Neural Network(RNN) を強化学習 Reinforcement Learning(RL) によって学習する.

2.2　Differentiable Architecture Search

Differentiable Architecture Search(DARTS)[2] は, 離散的なアーキテクチャ探索空間に強化学習を適用した NAS とは異なり, 微分可能な方法で定式化し, 偏微分による勾配降下法を使用してアーキテクチャを効率的に探索する手法である.

探索空間を連続にするため, カテゴリカルな演算子の選択の代わりに, 候補全ての可能性をもつ混合演算子を (1) 式で定義する. アーキテクチャを有向非巡回グラフで表したとき, ノードを潜在的な特徴表現 $x^{(i)}$, エッジを特徴 $x^{(i)}$ が適用される関数 $o(\cdot)$ とすると,

$$\bar{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} o(x) \qquad (1)$$

となる. ここで $\mathcal{O}$ は探索する演算子の候補集合, $\alpha^{(i,j)}$ はエッジ (i,j) の混合演算子の重みベクトルである. DARTS は勾配降下法によって連続変数集合 α を学習する.

α とレイヤーの重み w の Bi-Level 最適化問題を w の近似によって同時に学習し, NAS の 3000 GPU days に対して DARTS は 3.3 GPU days に高速化した.

DARTS では次元を統一するためセルと呼ぶ小さなネットワーク構造を重ねたモデルを利用する. セルを構成するノードは 2 つのノードからの演算子エッジを持ち, どのノードからの演算子を選ぶのかをアーキテクチャを示す重み α によって決定する. DARTS の問題点として位置と演算子の種類は探索できるが, 大局的な構造やノードの持つエッジ数などアーキテクチャが固定されていることが挙げられる.

2.3　Genetic Algorithm

遺伝的アルゴリズム (Genetic Algorithm : GA) は生物の進化の仕組みを模倣した最適化手法である. 問題の解候補を遺伝子の持つ個体として表現し, 適応度によって個体を評価・選択する. 交叉・突然変異などの操作によって解候補の多様性を保ちつつ, 近傍を探索しながら世代を重ねて近似的な最適解を求める.

GA には偶然適応度の高くなった個体だけが選択され続け, 個体群を同じ個体が占める初期収束問題がある. 問題によって適切な交叉・突然変異を行う必要がある.

3　問題

DARTS で柔軟なアーキテクチャを探索するため, 深層畳み込みネットワークの VGG19[3] のショートカット接続を探索する. VGG19 は 16 層の畳み込み層と 3 層の線形結合層を持つ. この VGG19 に対し層を飛ばして接続するショートカットの数と位置を求め, 性能を向上させることを目的とする.

モデル中の潜在的特徴は高さ・幅・チャンネル数を持つデータであるが, 特徴の次元は場所によって異なるため, ショートカットは次元を変換する必要がある. したがってショートカット関数は以下のように設定した.

1. 次元が同じ場合：恒等関数
2. フィルタ数が違う場合：Pointwise Convolution
3. 高さと幅が半分の場合：Factorized Reduce
4. それ以外の場合：ショートカットを定義しない

ショートカットに使用する関数の制限によってショートカット位置の候補は 61 であるため, 探索空間は 2^{61} である. 演算子の種類は固定することで, アーキテクチャ α は畳み込み部に相当するグラフの重みをもつ隣接行列と定義した.

4　手法と実験 1

ショートカットの本数も探索するため, α に対する重み補正 β を (2) 式 で定義する.

$$x_i = f^{\mathrm{c}}_{i-1,i}(x_{i-1}) + \beta_i * \sum_{j \in S_i} \alpha_{ij} * f^{\mathrm{s}}_{j,i}(x_j) \quad (2)$$

ここで $f^{\mathrm{c}}(\cdot)$, $f^{\mathrm{s}}(\cdot)$ は, VGG の畳み込み関数とショートカット関数, S_i はノード i とショートカットで接続する先行 (predecessor) ノードのインデックス集合である.

ただし $\beta = 0$ で勾配の更新ができなくなるので,

$$\hat{\beta} = \begin{cases} \exp(\beta - 1) & (\beta \leq 1) \\ \log(\beta) + 1 & (\text{otherwise}) \end{cases} \quad (3)$$

で 0 とならないように補正した $\hat{\beta}$ を用いた.

学習の手順は以下.

1. 探索：アーキテクチャ α の訓練
2. 構成：α からネットワークを構成
3. 評価：得られたネットワークを訓練し, テストデータで性能を評価

表 1: 実験 1 の設定

Loss	Cross Entropy Loss
batch size	64
Step	Architecture Search
Optim(w)	SGD(lr=0.001, momentum=0.9)
Optim(α)	Adam(lr=0.003, β=(0.5, 0.999))
data size	train : valid : test = 25000 : 25000 : 10000
Step	Evaluation
Optim(w)	SGD(lr=0.0090131, momentum=0.9)
Scheduler(w)	Step(γ=0.23440, stepsize=100)
data size	train : valid : test = 50000 : 0 : 10000

構成手法は複数考えられるため,

- 構成手法 A:predecessors の中で大きい順に採択
- 構成手法 B:閾値以上のエッジを採択

で実験した.

4.1　実験設定

表 1 に探索段階と評価段階の実験設定を示す. 探索段階は DARTS を参考に, 評価段階は optuna で最適化した値を使用した.

データセットは, 訓練画像が 32 pixel 四方で訓練データを 50000 枚持つ CIFAR-10 を利用して, 10 クラス分類問題を解いた.

探索時間は, 150 epoch とし, 50 epoch ごとにその時点の α の性能を評価した.

構成段階では手法 A, B に加えて比較のため, ショートカット数が同じとなる条件でランダムに選択する手法でも実験する. 各手法において 10 回試行して統計的な性能を比較した.

4.2　結果

表 2 に各構成手法におけるテストデータの精度を示す. 図 1, 2 には表 2 の精度に対するショートカット数とパラメータ数の関係を図示する. 最も性能が高かったのは 100 epoch 時点の手法 A で 94.02 %(baseline+0.99%) となり, 100 epoch 時点の手法 B は 93.93 %(baseline+0.90%) となった. しかしランダム手法と比較すると, 手法 A は+0.35%, 手法 B は+0.46%とな

表 2: 各アーキテクチャの精度

architecture		test accuracy (%)	param (M)	number of shortcuts	random architect accuracy (%)
method A	50 epoch	93.70 ± 0.22	21.06 ± 0.07	12.7 ± 1.4	93.60 ± 0.15
	100 epoch	94.02 ± 0.12	21.50 ± 0.11	18.2 ± 0.9	93.67 ± 0.14
	150 epoch	93.90 ± 0.17	21.57 ± 0.25	18.9 ± 0.6	93.64 ± 0.09
method B	50 epoch	93.57 ± 0.19	20.45 ± 0.09	5.8 ± 1.2	93.36 ± 0.19
	100 epoch	93.93 ± 0.08	20.73 ± 0.10	9.8 ± 1.0	93.47 ± 0.17
	150 epoch	93.92 ± 0.12	20.76 ± 0.15	10.6 ± 1.0	93.48 ± 0.15
baseline (VGG19)		93.03 ± 0.10	20.04	0	-

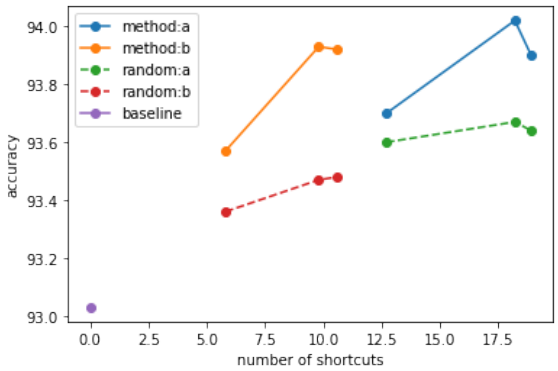

図 1: ショートカット数に対する精度

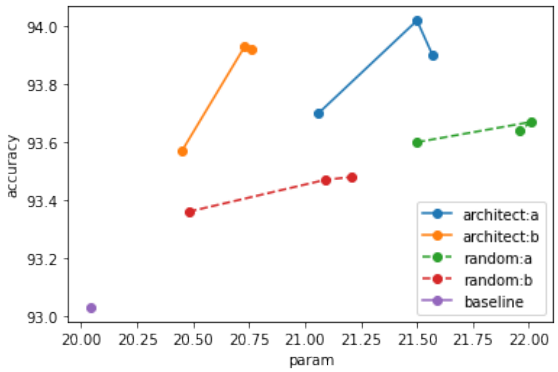

図 2: パラメータ数に対する精度

り, 図 2 を参照しても少ないパラメータ数でより有効に探索できているのは手法 B と言える.

また 100 epoch 時点と 150 epoch 時点を比較すると, 学習によって性能が悪化している. 問題に対して過度に適合していることが原因であると考えられる.

探索時間は 150 epoch でおよそ 5 gpu hours を要したが, DARTS と比べ演算子の探索を行っていないことや. 最適な重み w^* の近似を 1 次下げていることで高速になったと思われる.

5 手法と実験 2(GA)

実験 1 では α の学習度によって重み w の学習しやすさに偏りがあったため, 収束するグラフ構造にばらつきが見られた.

そこで個体表現を α とした遺伝的アルゴリズムによって, アーキテクチャの多様性を維持しつつ, 安定的なネットワーク構造の学習を図った.

各パラメータ集合の学習ステップを分離し個体間で不平等がないように設計した.

1. 一様乱数で初期個体生成
2. 重み w を $\nabla_w \mathcal{L}_{\text{train}}(w^*, \bar{\alpha})$ で更新
3. 個体 α_i を $\nabla_\alpha \mathcal{L}_{\text{valid}}(w^*, \alpha_i)$ で更新
4. 適応度 $\mathcal{L}_{\text{test}}(w, \alpha^{\text{sampled}})$ で個体 α を評価・選択
5. 交叉・突然変異
6. 収束するまで 2. に戻る

ただし $\bar{\alpha}$ は各個体の平均, α^{sampled} は構成手法 B で隣接行列にサンプリングした α .

学習後最終世代の個体の性能を実験 1 と同じ条件で評価した.

5.1 実験設定

表 3, 4 にモデルと GA の実験設定を示した. モデルの重み w は Image Net で訓練された事前学習の重みを畳み込み層の部分に適用した. 初期収束を防ぐため, 交叉にはエッジに相当する遺伝子座ごとに 0.5 の確率で操作する一様交叉を使用した. 突然変異には遺伝子

表 3: 実験 2 の設定

Optim(w)	SGD(lr=0.001, momentum=0.9)
Optim(α)	Adam(lr=0.003, β=(0.5, 0.999))
Loss	Cross Entropy Loss
pretrain	true
batch size	64
train size	25000
valid size	10000

表 4: GA の設定

個体数	15
世代数	20
選択	トーナメント
サイズ	2
交叉	一様交叉
交叉率	0.8
変異	ガウス分布
変異率	0.2

座ごとに 0.1 の確率で $\mu = 0, \gamma = 0.2$ となるガウス分布からの摂動を与えた.

5.2 結果

図 3

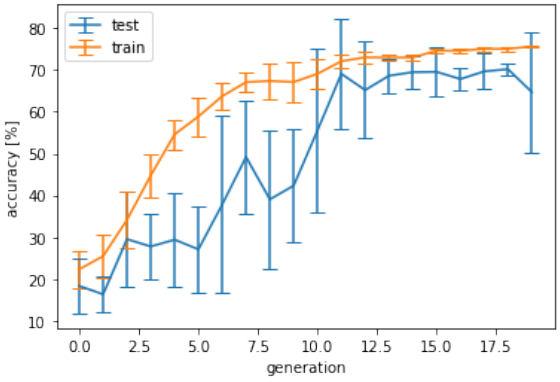

図 3: 世代ごとの精度

6 まとめと今後の課題

DARTS の欠点であるアーキテクチャ構造の制限を緩和するようなネットワーク探索ができた.

ネットワークの構成手法は改善の余地がある. 選択しないという候補を導入して, 他のショートカットと妥当な比較ができると考えられる.

GA を導入することでショートカットの本数の分析もできた.

他のデータセットや実問題に対して提案手法によるアーキテクチャの汎用性を確認したい.

参考文献

[1] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. abs/1611.01578, 2016.

[2] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. abs/1806.09055, 2018.

[3] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. 2015.