卒 業 研 究 報 告 書

題　　目

TDGA を導入した DARTS による深層学習の構造探索

研 究 グ ル ー プ　　第 1 研究グループ

指 導 教 員　　森直樹 教授

令 和　2　年（　　2020　年 ）度 卒 業

（No.　　1171201092　　）　　杉山 竜弥

大阪府立大学工学域電気電子系学類情報工学課程

TDGA を導入した DARTS による深層学習の構造探索

第 1 グループ　　杉山 竜弥

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大きく精度が向上してきた. 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による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. ベースとなるネットワークを 19 層構造の VGG19 とし, ショートカット位置について DARTS で探索を行う方法と, それに TDGA を導入する方法を提案する.

2. 要素技術

2.1. Differentiable Architecture Search

Differentiable Architecture Search(DARTS)[1] 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 従来手法とは異なり, 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. アーキテクチャ変数 α と, ネットワークの重み w を同時に学習する.

2.2. Thermodynamical Genetic Algorithm

Thermodynamical Genetic Algorithm (TDGA) は熱力学における自由エネルギー最小化をモデルにした, 遺伝的アルゴリズム (Genetic Algorithm: GA) で個体群の多様性を維持する手法である. 選択に温度とエントロピーの概念を導入し, 初期収束問題を防ぐことができる.

3. 提案手法

3.1. 実験 1:DARTS

DARTS を広範囲の構造探索が可能な実装に拡張し, ネットワークを探索する手法について様々なパラメータで実験した.

3.2. 実験 2:DARTS+TDGA

実験 1 では α の学習程度によって重み 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.

そこで実験 1 から得られた最適設定をもとに, 個体表現を α とした GA 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. 単純に個体数を増やす場合, 計算コストが定数倍されるため, 重み w は全体で共有する One-Shot モデルを利用することで高速化した.

以下に TDGA をベースとして, DARTS の学習ステップ (3. と 4.) の追加の検討 と TDGA の多様性項を実数値拡張した提案手法のアルゴリズムを示す.

1. DARTS で事前学習したモデルの重みを引き継いだ初期個体を生成
2. エリート個体選択
3. 重み w を $\nabla_\alpha \mathcal{L}_{\text{train}}(w, \bar{\alpha})$ で更新

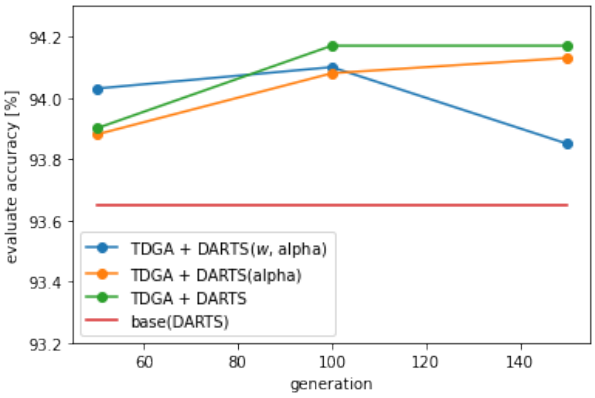


図 1: DARTS+TDGA のアーキテクチャ性能

4. 個体 α_i を $\nabla_\alpha \mathcal{L}_{\text{valid}}(w, \alpha_i)$ で更新
5. 適応度 $\mathcal{L}_{\text{test}}(w, \alpha_i)$ で個体 α_i を評価
6. 交叉で子個体群生成
7. 親個体群と子個体群の突然変異
8. エリート個体と親個体, 子個体に熱力学的選択をして次世代とする
9. 収束するまで 2. に戻る

DARTS の学習ステップが, GA の探索に与える影響を調べるため, 3. 4. のステップを行わない条件も比較する.

4. 数値実験

4.1. 実験 1

DARTS を用いたアーキテクチャ全体の探索ができ, さらにネットワークを構成する方法は, 閾値によるサンプリング手法が有効であるとわかった.

4.2. 実験 2

図 1 では, 50 世代から 150 世代までのアーキテクチャの評価結果を示しており, いずれも DARTS のみの性能を超えるアーキテクチャが探索できることがわかった. 毎世代 DARTS で w, α を更新すると, 適応度の出力が不安定になり効果的に GA が動作しないため, 更新は TDGA のみで行う手法が最も高い性能を達成したと考えられる.

5. まとめと今後の課題

本研究では, まず 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.

また DARTS に TDGA を組み合わせる提案手法によって, DARTS のみの結果を超えることができ提案手法の有効性が確認できた.

今後の課題として, 本研究で用いた VGG 以外のネットワークに対してもこの提案手法を適用することで汎用性を確認し, また他のデータセットや実問題では, 最適なアーキテクチャがどのように変化するか検討することが挙げられる.

参考文献

[1] DARTS: Differentiable Architecture Search, DARTS: Differentiable Architecture Search. DARTS: differentiable architecture search. abs/1806.09055, 2018.

目 次

図 目 次

表 目 次

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大きく精度が向上してきた. 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による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.

またネットワークの構造を組み合わせ最適化と考えることもでき, 導入する手法として今回は遺伝的アルゴリズム (Genetic Algorithm: GA) に着目する. GA は生物の進化の仕組みを模倣した最適化手法であるが, 初期段階で個体群が同じ個体で占められる初期収束問題がある. 一方 Thermodynamical Genetic Algorithm (TDGA) では, GA の選択ルールに多様性を考慮した適応度を導入することでこの初期収束問題を防ぐことができる.

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. ベースとなるネットワークを 19 層構造の VGG19 とし, ショートカット位置について DARTS で探索を行う方法と, それに TDGA を導入する方法を提案する.

以下に本論文の構成を示す．まず，2 章では本研究で用いる要素技術について概説する．3 章で深層学習の構造の設定と探索手法を提案する. そして 4 章において，数値実験により手法の性能を検証し, 本研究で提案する手法の考察をする．5 章で本研究の成果をまとめたうえで，今後の課題について述べる．

2　要素技術

本章では，本研究の提案手法に用いた技術について説明する．

2.1　Neural Architecture Search

本章では本研究で用いた Differentiable Architecture Search (DARTS) をはじめとした Neural Architecture Search (NAS) について説明する．

従来の機械学習では手作業によって設計されたモデルをデータセットで学習し重みを最適化するが, 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．

2.1.1　Neural Architecture Search with Reinforcement Learning

Neural Architecture Search with Reinforcement Learning (NAS with RL)[1] 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)[2] を強化学習 Reinforcement Learning (RL)[3] によって学習する．

RNN はレイヤーごとにフィルタの高さ・幅, ストライドの高さ・幅, フィルタ数を決定し, RNN によって生成された構造は, ニューラルネットワークとしてその重みが学習されテストの正答率によって性能が評価される．その性能から得られた報酬で, 方策勾配法 (Policy gradient method)[4] による RNN の更新を行い, アーキテクチャが最適化される．

NAS with RL は高い性能を達成した一方で, 計算に数千 GPU days が必要となる問題もある．

2.1.2　Differentiable Architecture Search

Differentiable Architecture Search (DARTS)[5] 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 with RL とは異なり, 微分可能な方法で定

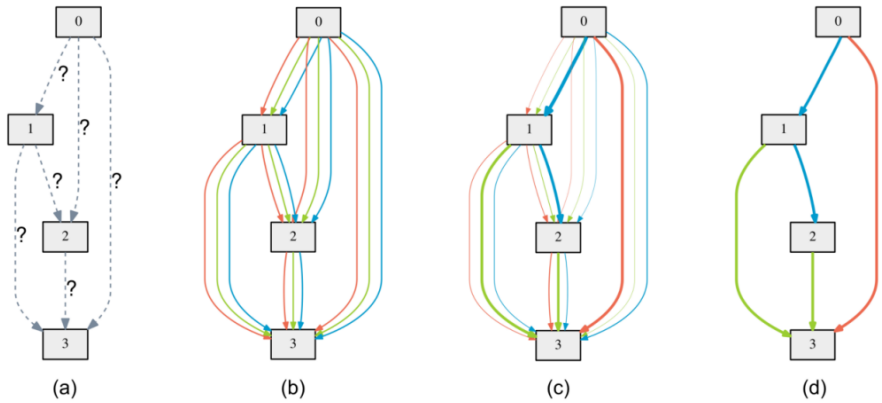


図 2.1: DARTS の概念図
(a) はじめ辺上の演算子は不明 (b) 各辺の演算子候補の混合で置換することで探索空間を連続性緩和 (c) 混合確率とネットワークの重みを最適化 (d) 学習した混合確率から最終的なアーキテクチャを導出 (文献 [5] の 図 1 参照)

式化し [6], 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 (2.1) 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 $\boldsymbol{x}^{(i)}$, エッジを特徴 $\boldsymbol{x}^{(i)}$ が適用される関数 $o(\cdot)$ とすると,

$$\bar{o}^{(i,j)}(\boldsymbol{a}) = \sum_{o \in \mathcal{O}} \frac{\exp(\boldsymbol{\alpha}_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\boldsymbol{\alpha}_{o'}^{(i,j)})} o(\boldsymbol{x}) \tag{2.1}$$

となる. ここで $\mathcal{O}$ は探索する演算子の候補集合, $\boldsymbol{\alpha}^{(i,j)}$ 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 $\boldsymbol{\alpha}$ を学習する.

$\boldsymbol{\alpha}$ とレイヤーの重み $\boldsymbol{w}$ の Bi-Level 最適化問題 [7] を $\boldsymbol{w}$ の近似によって同時に学習し, NAS において 3000 GPU days 必要なタスクに対して DARTS は 3.3 GPU days まで高速化した.

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 $\boldsymbol{\alpha}$ 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固定されたアーキテクチャにしか適用できない点が挙げられる.

2.2 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.

選択は現世代から次世代の個体群を選ぶ操作である.

トーナメント選択は, 個体群から一定数 (トーナメントサイズ) の個体をランダムに選び, 最も適応度の高いものを残す. 適応度の差は反映されないため, 偶然性が高い.

交叉は現世代から子を生成する操作である. 遺伝子型がバイナリ型, 実数型, 順序型かによってそれぞれ交叉方法が存在する. バイナリ型では,

- 1 点交叉 : 染色体中の 1 箇所でランダムに切断し, 親の遺伝子を入れ替える
- 多点交叉 : 染色体中の複数箇所でランダムに切断し, 親の遺伝子を互い違いに入れ替える
- 一様交叉 : 遺伝子座ごとの交叉確率によって, 各遺伝子座でランダムに親の遺伝子を入れ替える

などがある. 実数型ではこれらに加えて,

- BLX-α : 両親の遺伝子が存在する領域に α 倍した範囲から子を生成する手法. 領域の拡大によって多様性を保つ.
- UNDX : 正規分布に従い, 両親の周辺にその中点に対し対称な位置に 2 つの子を生成することで, 多様性を維持する手法.

も選択できる.

突然変異は個体をランダムに変化させる操作である. 解が収束した場合, 交叉にはない局所解からの脱出という効果を持つが, 突然変異率が高すぎるとランダム探索になるため十分に小さな値を用いる. 変異方法には, 各遺伝子座でランダムに対立遺伝子へ置き換える方法, 実数値に対して摂動を与える方法などがある.

また GA には初期収束と呼ばれる, 探索の初期段階で適応度が高い個体だけが選択され続け, 個体群を同じ個体が占められてしまう問題がある. このような多様性が失われた状態になると単純なランダム探索と変わらない効率になるため, パラメータ調整などの方法で回避する必要がある.

交叉・突然変異の手法やパラメータは問題によって異なるため, 適切なものを設定するのは自明ではない.

2.2.1　Thermodynamical Genetic Algorithm

Thermodynamical Genetic Algorithm (TDGA) [8] は熱力学における自由エネルギー最小化をモデルにした, GA で個体群の多様性を維持する手法である. 選択に温度とエントロピーの概念を導入し, 初期収束問題を防ぐことができる.

シミュレーテッドアニーリング法 (Simulated Annealing: SA) は次のエネルギー関数 (2.2) 式を用いて最適化問題を解く一般的な最適化手法である.

$$\min_{\boldsymbol{x}} E(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathcal{F} \tag{2.2}$$

ここで $\mathcal{F}$ は有限集合であることを仮定する. SA では系の状態 $\boldsymbol{x}$ に対して摂動を加え, 新しい状態 $\boldsymbol{x}'$ を得る. そして新しい状態でのエネルギー値 $E(\boldsymbol{x}')$ が旧状態のエネルギー値 $E(\boldsymbol{x})$ より小さければ高い確率で, 大きければ温度パラメータ T に基づいた低い確率で新状態 $E(\boldsymbol{x}')$ への遷移を行う. SA はこのアプローチを使用して最小状態を見つける.

T が定数のとき, SA の典型的な遷移規則であるメトロポリス法の分布はギブス分布となり, そのとき (2.3) 式で定義される自由エネルギー F を最小化することが知られており, これは自由エネルギーの最小化原理と呼ばれている.

$$F = \langle E \rangle - HT \tag{2.3}$$

ここで $\langle E \rangle$ は系の平均エネルギー, H はエントロピーである．

TDGA におけるエントロピーの計算と自由エネルギーの最小化

SA において最小化される自由エネルギー (2.3) 式の右辺第 1 項は, 系がエネルギー最小化という目的を追求する項, 第 2 項は系の状態の多様性を維持する項と解釈でき, これら両者を温度 T をパラメータとして調和させたものと考えられる．そこで TDGA では単純 GA(Simple GA: SGA) で用いられている適応度比例戦略に代えて, 自由エネルギーを最小化するように次世代の個体群を選択することを基本方針とする．

個体群中の個体の種を区別した系の多様性を表すエントロピー H^{ALL} は (2.4) 式で表される．

$$H^{\mathrm{ALL}} = -\sum_{i} p_i \log p_i \tag{2.4}$$

ここで p_i は種 i の存在確率である．GA では可能な状態 $\boldsymbol{x} \in \mathcal{F}$ のうち各個体がとり得る値は個体数 N_{p} 程度であり, $|\mathcal{F}|$ に比べて極めて小さい．よって TDGA は代わりに (2.5) 式を用いて各対立遺伝子座から個体群のエントロピー H^1 を計算する．

$$H^1 = \sum_{k=1}^{M} H_k^1, \quad H_k^1 = -\sum_{j \in \{0,1\}} P_j^k \log P_j^k \tag{2.5}$$

ここで H_k^1 は個体群の遺伝子座 k の遺伝子に関するエントロピーを, P_j^k は遺伝子座 k における対立遺伝子 j の存在確率を表している．TDGA においては H^1 は (2.3) 式の第 2 項の H とみなされる．これにより世代のエントロピーが計算できるが, 自由エネルギーを厳密に最小化する個体群を選ぶことは, それ自体困難な組み合わせ最適化問題であり, 多大な計算量を要する．しかしながら, 自由エネルギーの最小化は, GA における次世代の形成の評価基準にすぎないので, 厳密な最小化は必要ではない．そこで TDGA では近似的な最小化手法として欲張り法を用いる．すなわち, 次世代の個体群を逐次的に形成する

際にその時点で自由エネルギーを最小にする個体を現世代の個体群から選び, 次世代の個体群に付加するという方法を用いる.

ただし, (2.4) 式は遺伝子がバイナリ型の場合であり, エントロピーは有限集合の確率空間上で定義されるため, 実数型の場合には別の方法が必要となる.

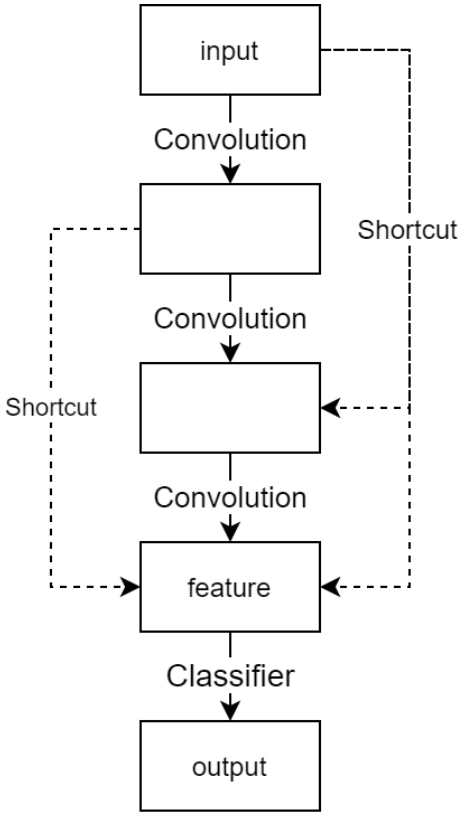


図 3.1: ショートカット概念図

3　ショートカット探索

DARTS で柔軟なアーキテクチャを探索することを実験の目的とし, 構造探索タスクとして深層畳み込みネットワークの VGG19[9] のショートカット接続 [10] を考える. 図 3.1 のように, ショートカット接続は, 畳み込み部において 1 つ以上特徴を飛ばした接続とする. VGG19 は分岐がない単純なネットワーク構造であるため, ベースモデルに適しているとして選択した.

VGG19 は 16 層の畳み込み層と 3 層の線形結合層を持つ. 表 3.1 は, VGG19 の畳み込みニューラルネットワーク (Convolutional Neural Network: CNN) 部分の構造を示している. 構成する関数は, フィルターサイズが 3×3 の畳み込み層 (Conv2d) , Batch Normalization (BN)[11] , 活性化関数 (Rectified Linear Unit: ReLU) , ストライドが 2 の Max Pooling (MaxPool) である. 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.

表 3.1: VGG19 の構造
例として入力する画像を (32, 32, 3) 次元としている.

index	**image size**	**channels**	**applied function**
input	32 × 32	3	-
1	32 × 32	64	3×3_Conv2d, BN, ReLU
2	16 × 16	64	3×3_Conv2d, BN, ReLU, MaxPool
3	16 × 16	128	3×3_Conv2d, BN, ReLU
4	8 × 8	128	3×3_Conv2d, BN, ReLU, MaxPool
5	8 × 8	256	3×3_Conv2d, BN, ReLU
6	8 × 8	256	3×3_Conv2d, BN, ReLU
7	8 × 8	256	3×3_Conv2d, BN, ReLU
8	4 × 4	256	3×3_Conv2d, BN, ReLU, MaxPool
9	4 × 4	512	3×3_Conv2d, BN, ReLU
10	4 × 4	512	3×3_Conv2d, BN, ReLU
11	4 × 4	512	3×3_Conv2d, BN, ReLU
12	2 × 2	512	3×3_Conv2d, BN, ReLU, MaxPool
13	2 × 2	512	3×3_Conv2d, BN, ReLU
14	2 × 2	512	3×3_Conv2d, BN, ReLU
15	2 × 2	512	3×3_Conv2d, BN, ReLU
16	1 × 1	512	3×3_Conv2d, BN, ReLU, MaxPool

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. チャンネル数が違う場合：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

Pointwise Convolution は, 要素ごとの畳み込みでチャンネル数を調整する. Factorized Reduce は, 畳み込み層を偶数列と奇数列の 2 枚 用いることで, 情報の損失を防ぎつつ特徴次元を半分にすることができる.

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 $\boldsymbol{\alpha}$ 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.

3.1　提案手法 : DARTS

DARTS でネットワーク構造を探索するときの, 学習の手順は

1. 探索：アーキテクチャ $\boldsymbol{\alpha}$ の訓練
2. 構成：$\boldsymbol{\alpha}$ 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バックプロパゲーションにより訓練し, テストデータで性能を評価

の 3 段階から成る.

3.1.1　探索

探索段階では, 勾配降下法によって $\boldsymbol{\alpha}$ の更新を行う. このとき探索用のネットワークは, ショートカットの本数も探索するため, $\boldsymbol{\alpha}$ に対する重み補正 $\boldsymbol{\beta}$ を用いて (3.1) 式 で定義する.

$$\boldsymbol{x}_i = f^{\mathrm{c}}_{i-1,i}(\boldsymbol{x}_{i-1}) + \boldsymbol{\beta}_i \sum_{j \in S_i} \boldsymbol{\alpha}_{ij} f^{\mathrm{s}}_{j,i}(\boldsymbol{x}_j) \tag{3.1}$$

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 $\boldsymbol{\beta} = 0$ で勾配の更新ができなくなるので,

$$\hat{\boldsymbol{\beta}} = \begin{cases} \exp(\boldsymbol{\beta} - 1) & (\boldsymbol{\beta} \leq 1) \\ \log(\boldsymbol{\beta}) + 1 & (\text{otherwise}) \end{cases} \tag{3.2}$$

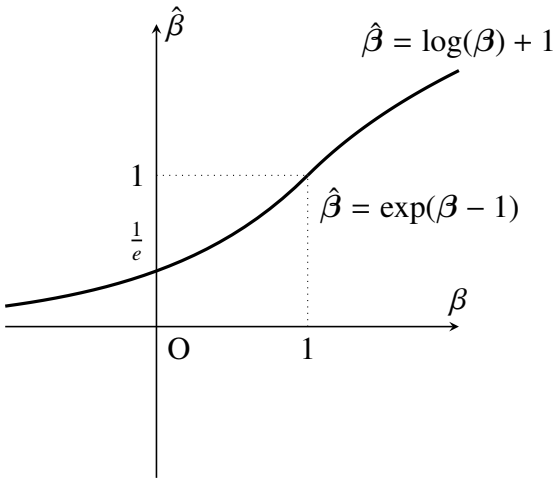


図 3.2: β の補正関数

で 0 とならないように補正した $\hat{\beta}$ を用いた. 図 3.2 には, (3.2) 式 の補正関数を示した.

3.1.2　構成

構成段階では, 探索段階で得られた α から具体的なネットワークをサンプリングする. α の値に対する, ネットワークの構成手法はいくつか考えられるため,

- 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

の 2 通りの手法を設定した.

3.1.3　評価

評価段階では, 構成段階で得られたネットワークを学習し, 最大の正答率をネットワークの性能とする. このときのパラメータは, グリッドサーチによる最適化 API の Optuna [12] で事前学習した結果から設定する.

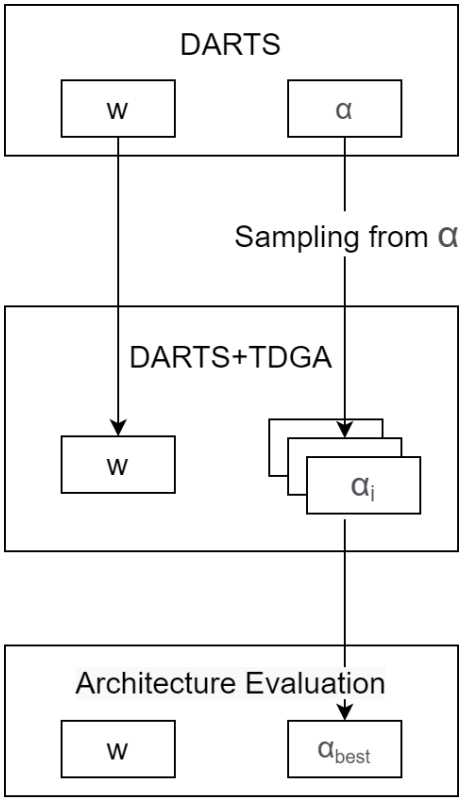


図 3.3: 提案手法 : DARTS + TDGA 概念図

3.2　提案手法 : DARTS + TDGA

実験 1 では $\boldsymbol{\alpha}$ の学習程度によって重み $\boldsymbol{w}$ 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.

そこで実験 1 から得られた構造探索のための実験設定をもとに, 個体表現を $\boldsymbol{\alpha}$ とした遺伝的アルゴリズムによって, アーキテクチャを複数同時に管理, 最適化し, 個体群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. 単純に個体数を増やすと計算コストが定数倍されるため, 重み $\boldsymbol{w}$ は全体で共有する One-Shot モデル [13] を利用することで高速化した.

3.2.1　多様性項の拡張

(2.4) 式では, 多様性の計算にエントロピーを利用していたが, 個体 $\boldsymbol{\alpha}$ は実数値の行列であるためエントロピーは計算できない.

$$H = \sum_{k \in \mathcal{P}} \sqrt{\mathrm{MSE}(\boldsymbol{\alpha}_k, \bar{\boldsymbol{\alpha}})} \tag{3.3}$$

そこで行列の各要素の標準偏差の平均として, H の実数値拡張をした.

3.2.2　実験手順

実験は 3 段階からなり, 事前学習した DARTS から, $\boldsymbol{w}$, $\boldsymbol{\alpha}$ を引き継ぎ, TDGA で再探索し, 得られた最良個体のアーキテクチャを学習して性能を評価する. 図 3.3 に $\boldsymbol{w}$, $\boldsymbol{\alpha}$ の関係を示す.

1. 事前学習
2. アーキテクチャ再探索
3. 性能評価

3.2.3　事前学習

事前学習では, TDGA に引き継ぐための初期値を DARTS で学習する. 実験 1 の結果から辺ごとに計算する手法が有効であるとわかったため, 正規化として先行研究の DARTS でも利用されていた, Softmax ではなく Hard Sigmoid に変更する. 図 3.4 には Hard Sigmoid と Sigmoid 示す. この変更によって, α の正規化された値と生の値の相互変換ができるようになり, 実験間のデータ受け渡しが容易となった.

3.2.4　アーキテクチャ再探索

以下に TDGA をベースとして, DARTS の学習ステップの追加と多様性項の実数値拡張をした提案手法のアルゴリズム

- 提案手法 1. DARTS + TDGA ($\boldsymbol{w}$, $\boldsymbol{\alpha}$)
- 提案手法 2. DARTS + TDGA ($\boldsymbol{\alpha}$)

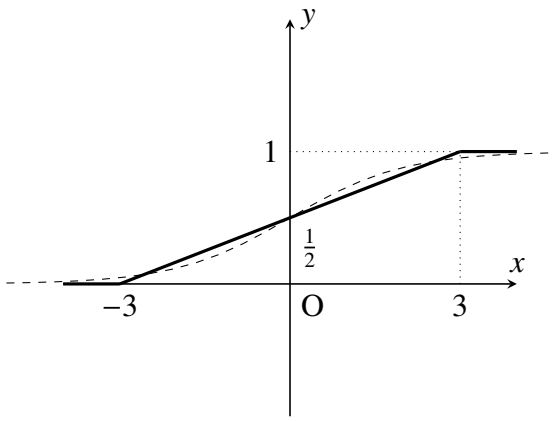


図 3.4: Hard Sigmoid 関数
Sigmoid 関数は破線, Hard Sigmoid 関数は実線.

- 提案手法 3. DARTS + TDGA

を アルゴリズム 1, 2, 3 に示す. アルゴリズム 1 をベースとして, DARTS による勾配更新のステップを無効にすることで, DARTS が TDGA の学習に与える影響をみる.

3.2.5　性能評価

性能評価の段階では, 提案手法による探索結果をもとに, 学習後最終世代の個体の性能を実験 1 と同じ条件で学習し評価した.

Algorithm 1 提案手法 1. DARTS + TDGA ($\boldsymbol{w}$, $\boldsymbol{\alpha}$)

1. DARTS で事前学習したモデルの重みを引き継いだ初期個体を生成
2. 重み $\boldsymbol{w}$ を $\nabla_{\boldsymbol{\alpha}}\mathcal{L}_{\text{train}}(\boldsymbol{w}, \bar{\boldsymbol{\alpha}})$ で更新
3. 個体 $\boldsymbol{\alpha}_i$ を $\nabla_{\boldsymbol{\alpha}}\mathcal{L}_{\text{valid}}(\boldsymbol{w}, \boldsymbol{\alpha}_i)$ で更新
4. 適応度 $\mathcal{L}_{\text{test}}(\boldsymbol{w}, \boldsymbol{\alpha}_i)$ で個体 $\boldsymbol{\alpha}_i$ を評価
5. エリート個体選択
6. 交叉で子個体群生成
7. 親個体群と子個体群の突然変異
8. エリート個体と親個体群, 子個体群に熱力学的選択によって次世代とする
9. 収束するまで 2. に戻る

Algorithm 2 提案手法 2. DARTS + TDGA ($\boldsymbol{\alpha}$)

1. DARTS で事前学習したモデルの重みを引き継いだ初期個体を生成
2. 個体 $\boldsymbol{\alpha}_i$ を $\nabla_{\boldsymbol{\alpha}}\mathcal{L}_{\text{valid}}(\boldsymbol{w}, \boldsymbol{\alpha}_i)$ で更新
3. 適応度 $\mathcal{L}_{\text{test}}(\boldsymbol{w}, \boldsymbol{\alpha}_i)$ で個体 $\boldsymbol{\alpha}_i$ を評価
4. エリート個体選択
5. 交叉で子個体群生成
6. 親個体群と子個体群の突然変異
7. エリート個体と親個体群, 子個体群に熱力学的選択によって次世代とする
8. 収束するまで 2. に戻る

Algorithm 3 提案手法 3. DARTS + TDGA

1. DARTS で事前学習したモデルの重みを引き継いだ初期個体を生成
2. 適応度 $\mathcal{L}_{\text{test}}(\boldsymbol{w}, \boldsymbol{\alpha}_i)$ で個体 $\boldsymbol{\alpha}_i$ を評価
3. エリート個体選択
4. 交叉で子個体群生成
5. 親個体群と子個体群の突然変異
6. エリート個体と親個体群, 子個体群に熱力学的選択によって次世代とする
7. 収束するまで 2. に戻る

4　数値実験

4.1　実験 1

4.1.1　実験概要

提案手法 : DARTS の実験を 実験 1 とする. 表 4.1, 4.2 に実験 1 の探索段階と評価段階の実験設定を示す. 探索段階は先行研究の DARTS の設定を参考に, 評価段階は Optuna で最適化した値を使用した. α の Optimizer は Adam[14] で, 評価段階では, 訓練中に学習率を変化させる Scheduler[15] を用いた.

データセットは, 32 pixel × 32 pixel × 3 channel の訓練画像を 50000 枚 持つ CIFAR-10[16] を利用して, 10 クラス分類問題を解いた.

探索 epoch 数は, 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した. 各手法において 10 回試行して統計的な性能を比較した.

表 4.1: 実験 1 : 探索段階の設定

Step	Architecture Search
Loss	Cross Entropy Loss
batch size	64
Optimizer ($\boldsymbol{w}$)	SGD (lr = 0.001, momentum = 0.9)
Optimizer ($\boldsymbol{\alpha}$)	Adam (lr = 0.003, $\boldsymbol{\beta}$ = (0.5, 0.999))
epoch	150
train data	25000
valid data	25000
test data	10000

表 4.2: 実験 1 : 評価段階の設定

Step	Evaluation
Loss	Cross Entropy Loss
batch size	64
Optimizer ($\boldsymbol{w}$)	SGD (lr = 0.0090131, momentum = 0.9)
Scheduler ($\boldsymbol{w}$)	Step (γ = 0.23440, stepsize = 100)
epoch	150
train data	50000
test data	10000

表 4.3: 実験 1 : 各アーキテクチャの精度

architecture		test accuracy (%)	param (M)	shortcut #	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	-

4.1.2　結果

表 4.3 に各構成手法におけるテストデータの精度を示す. 図 4.1, 4.2 には表 4.3 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 % となり, 図 4.2 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, 150 epoch の方が性能が悪化していることが分かる. 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, 先行研究の DARTS では 3 GPU days であった. 先行研究と比べ演算子を探索していないことや. 最適な重み $\boldsymbol{w}$ 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.

図 4.3, 4.4 には, 手法 A, B のアーキテクチャの 50 epoch ごとの結果として, 10 回試行のうちの 1 つを示した. ランダム探索では見られなかったような, 比較的離れた位置からの接続が多いという特徴があった. さらに, 学習を経るごとに離れていくため, 接続の距離がネットワークの性能に関係していると考え

られる. これは離れた位置からの接続が, あまり変換されていない特徴の情報を引き継げることによって性能が向上したと解釈できる.

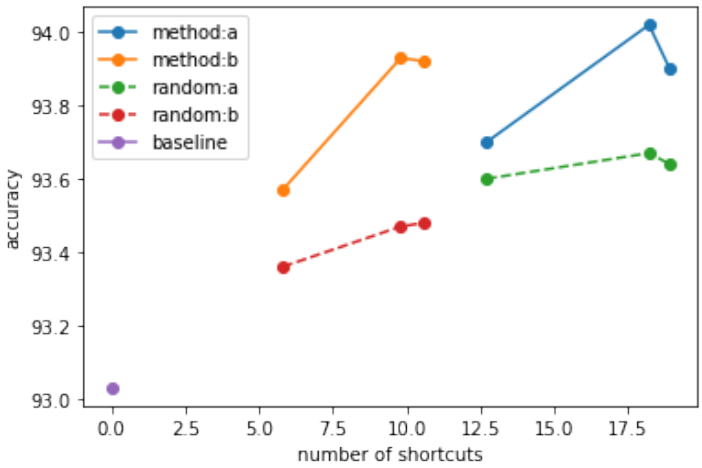


図 4.1: 実験 1 : ショートカット数に対する精度

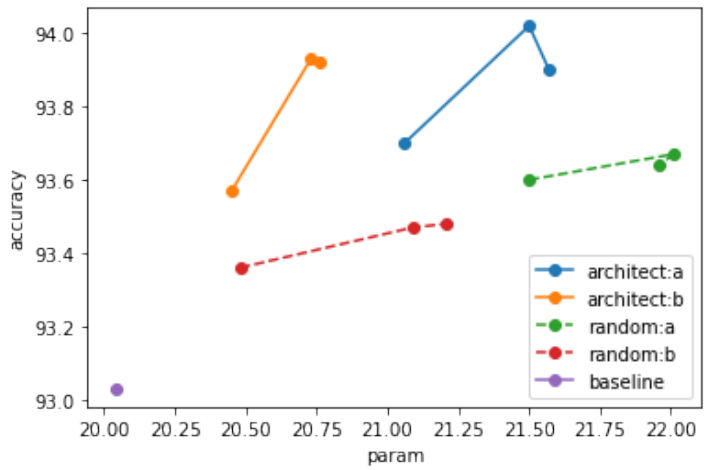
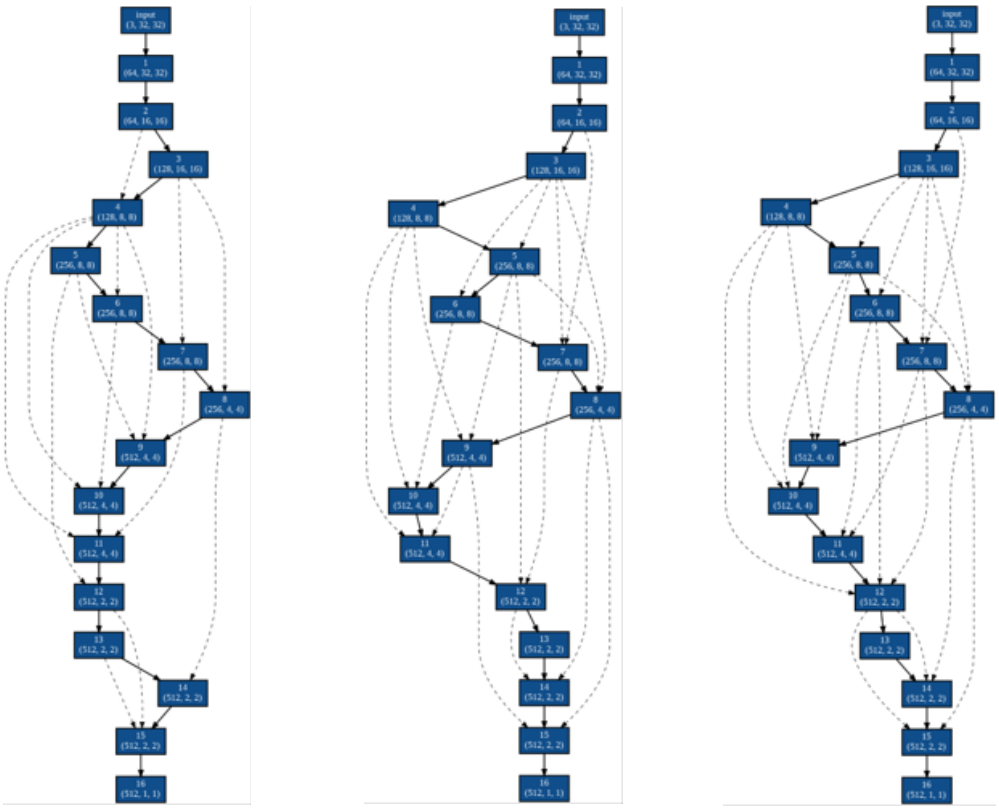
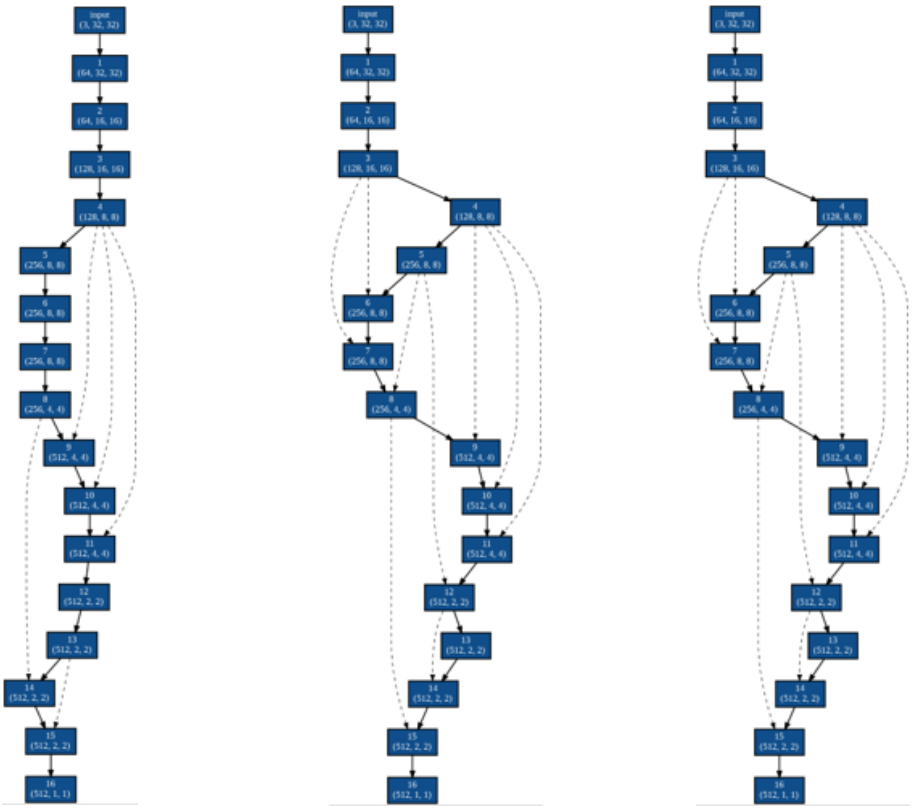


図 4.2: 実験 1 : パラメータ数に対する精度



50 epoch　100 epoch　150 epoch

図 4.3: 実験 1 : 手法 A のアーキテクチャ 例



50 epoch　　100 epoch　　150 epoch

図 4.4: 実験 1 : 手法 B のアーキテクチャ 例

表 4.4: 実験 2 : 事前学習の設定

Optimizer ($\boldsymbol{w}$)	SGD (lr = 0.001, momentum = 0.9)
Optimizer ($\boldsymbol{\alpha}$)	Adam (lr = 0.0005, $\boldsymbol{\beta}$ = (0.5, 0.999))
Loss	Cross Entropy Loss
batch size	64
train data	25000
valid data	25000
test data	10000
epoch	150

4.2　実験 2

4.2.1　実験概要

提案手法 : DARTS + TDGA の実験を 実験 2 とする.

表 4.4 には事前学習の設定を示した. 実験 1 の設定を継承しつつ, モデルの重み $\boldsymbol{w}$ は Image Net[17] 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.

表 4.5, 4.6 にモデルと GA の実験設定を示した. 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 0.8 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. 交叉と突然変異の操作確率は, 個体ごとの確率で, それぞれは独立であり, 交叉の後に突然変異をした.

表 4.5: 実験 2 : DARTS + TDGA の設定 (DARTS)

Optimizer ($\boldsymbol{w}$)	SGD (lr = 0.001, momentum = 0.9)
Optimizer ($\boldsymbol{\alpha}$)	Adam (lr = 0.001, $\boldsymbol{\beta}$ = (0.5, 0.999))
Loss	Cross Entropy Loss
batch size	64
train data	25000
valid data	25000
test data	2000

表 4.6: 実験 2 : DARTS + TDGA の設定 (TDGA)

Population	10
Generation	150
Selection	Elite
Elite #	1
Selection	TD Select
Temperature	$1 \rightarrow 0.001$
Crossover	Uniform Crossover (0.5 / locus)
Crossover Rate	0.8
Mutation	Gaussian Mutation ($\mu = 0$, $\sigma = 0.2, 0.2$ / locus)
Mutation Rate	0.2

表 4.7: 実験 2 : ネットワーク評価の設定

Optimizer ($\boldsymbol{w}$)	SGD (lr = 0.009, momentum = 0.9)
Scheduler ($\boldsymbol{w}$)	Step (γ = 0.2344, step epoch = 100)
Loss	Cross Entropy Loss
batch size	64
train data	50000
test data	10000
epoch	150

表 4.8: 実験 2 : 各提案手法のアーキテクチャ性能
ベースラインは事前学習時のネットワーク.

	50 世代	**100** 世代	**150** 世代
DARTS + TDGA ($\boldsymbol{w}$, $\boldsymbol{\alpha}$)	94.03 %	94.10 %	93.85 %
DARTS + TDGA ($\boldsymbol{\alpha}$)	93.88 %	94.08 %	94.13 %
DARTS + TDGA	93.90 %	94.17 %	94.17 %
Baseline (DARTS)	93.65 %		

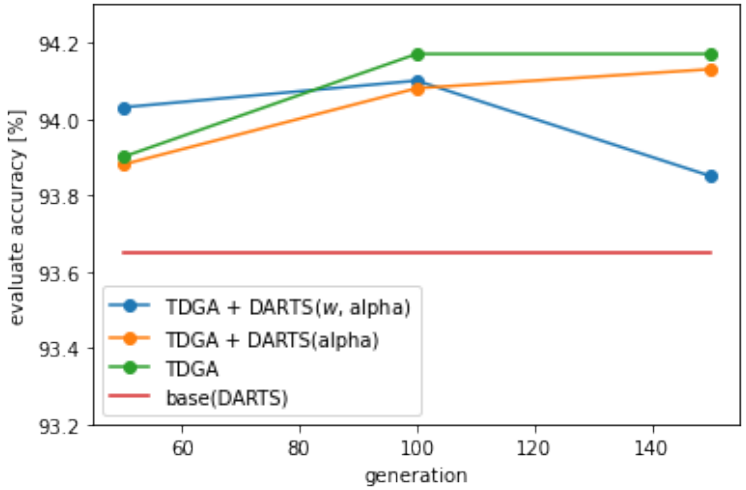


図 4.5: 実験 2 : 各提案手法のアーキテクチャ性能

4.2.2　結果

表 4.8 に, 3 つの提案手法の最終的なアーキテクチャ性能を示す. 50 世代ごとの結果全てに対して, DARTS のみとなる事前学習のアーキテクチャ 93.65 % をいずれも超えている. 図 4.5 は 横軸に世代数, 縦軸に accuracy をとった, 表 4.8 のグラフを示す. $\boldsymbol{w}$ を更新する提案手法 1 が 150 世代目で性能が下がっている. また DARTS の更新をしない提案手法 3 が最もよい結果となった.

次に 図 4.6 ～ 4.8 で探索段階のデータを考察する.

図 4.6, 図 4.7 に提案手法の適応度計算時のテストデータの正解率と損失を示す. ただし個体群の中で最良の適応度を得た個体をその世代の代表としている. 提案手法 1 は重み $\boldsymbol{w}$ も更新しているため, 最も訓練時の正解率が高く,

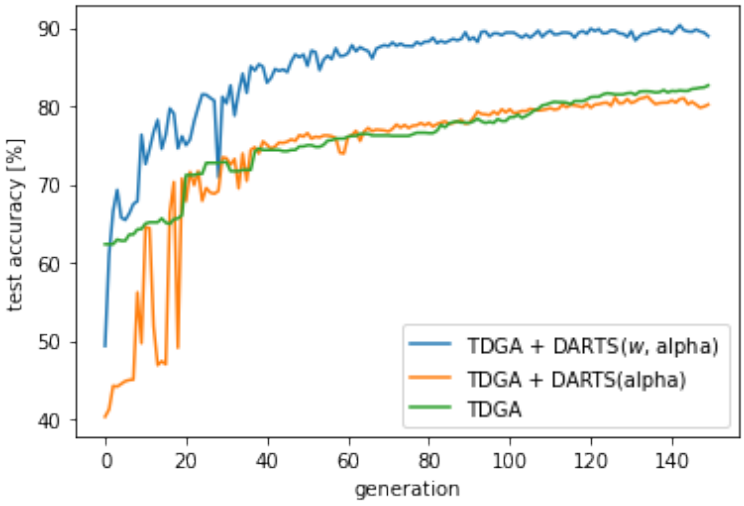


図 4.6: 実験 2 : 各提案手法の探索時の正解率

90 % 付近まで学習している. それと比べて, 提案手法 2, 3 は 10 % 程度低くなっているが, 提案手法 3 は 提案手法 2 と比較して, 150 世代でも正答率の増加傾向が見られる. 図 4.5 の最終的な評価から考えて, 学習時の正答率の高さがネットワークの性能と直結しているわけではないことが分かった. また 提案手法 2, 3 は $\boldsymbol{w}$ を固定しており, 同じ個体に対しては同じ適応度が計算されるため, 純粋にアーキテクチャが学習される過程を示している. この同じ適応度が計算されるという信頼性が, 提案手法 2, 3 の結果が 提案手法 1 よりも優れていた原因であると考えられる. したがって適応度計算には, 学習過程の正答率よりも同じ結果となることがより重要であると言える.

図 4.8 には, 各探索段階での採用されるショートカット数を示した. ただしショートカット数は最良個体のものとした. 提案手法 2, 3 は 世代数が 50 から 100 のときに主に改善されているが, 図 4.8 を参照するとショートカット数は減少している最中である. ネットワークの計算の邪魔になる, 不必要な辺を削除できたことで性能が改善されたことが分かった.

図 4.9, 4.10 には, 事前学習と各提案手法の最終世代時の最良個体のアーキテクチャを示した. 実験 1 でもみられたような離れた位置からのショートカットが多いという特徴が, 提案手法 2, 3 でもみられ, 逆に提案手法 1 では無秩序を感じさせるような様々な位置で接続する構造となった.

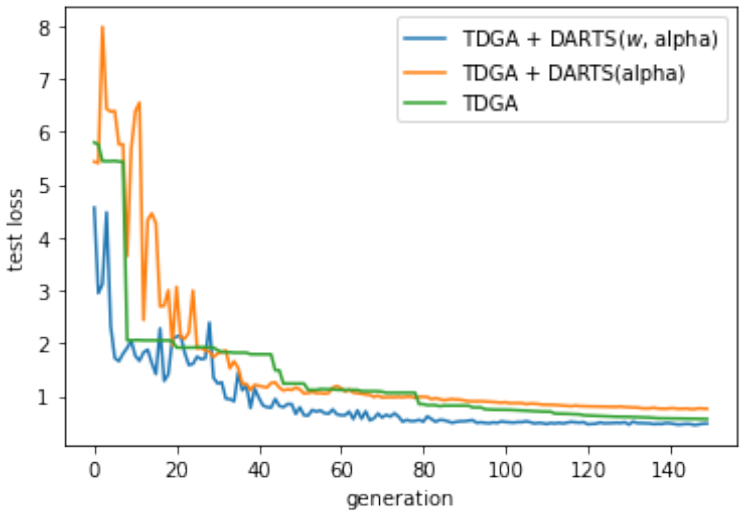


図 4.7: 実験 2 : 各提案手法の探索時の損失

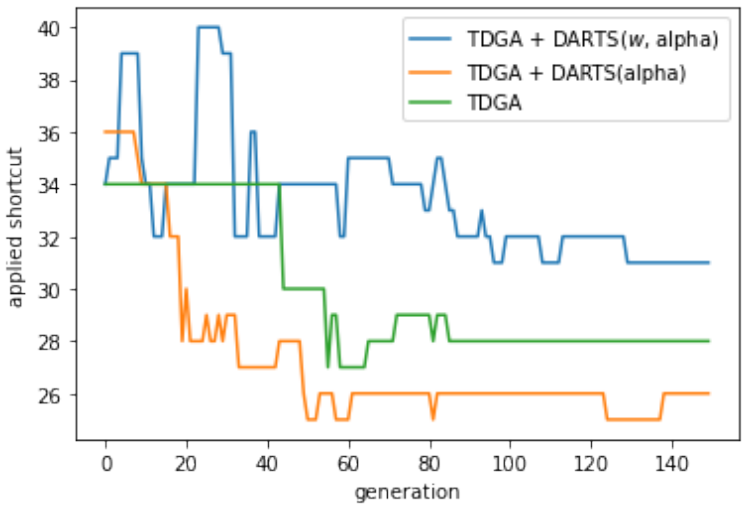
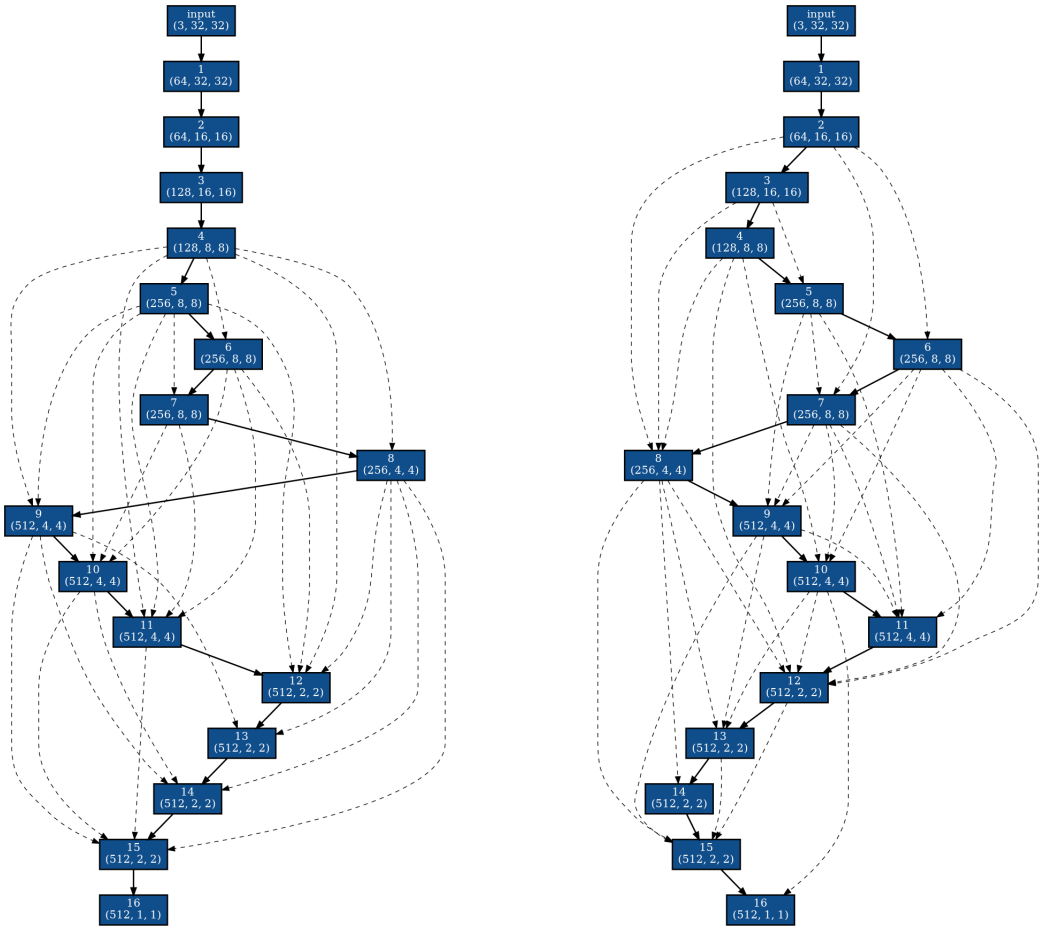
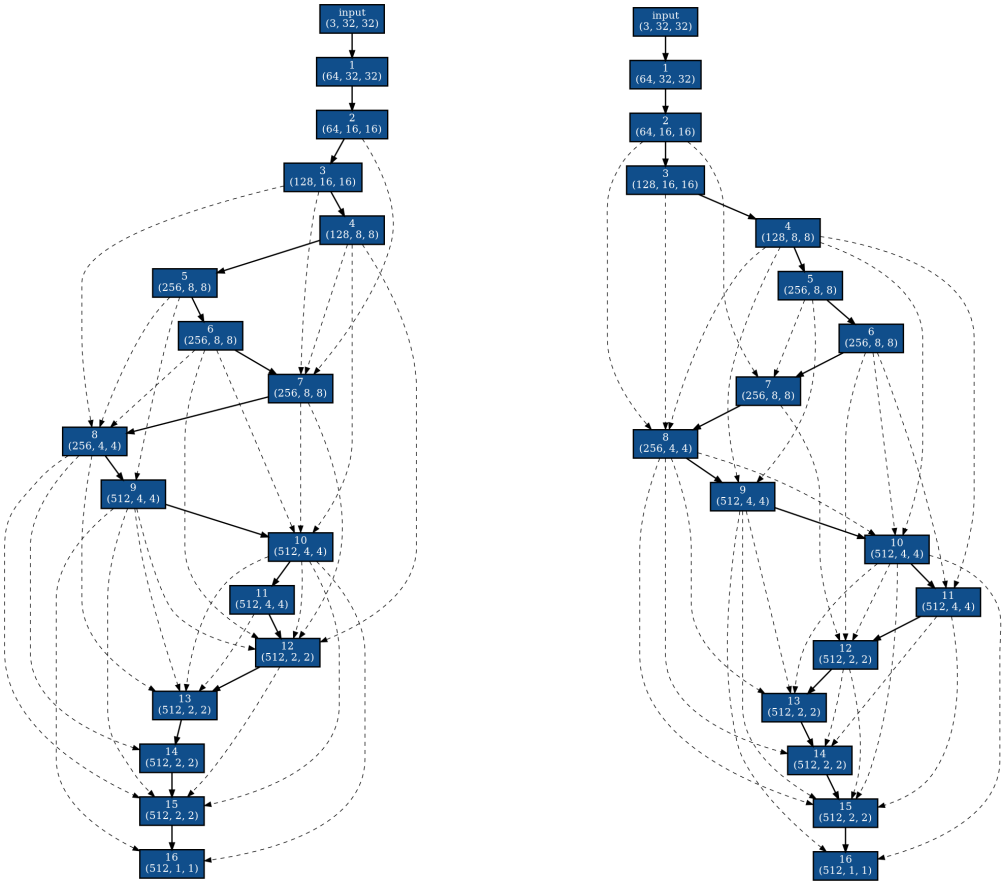


図 4.8: 実験 2 : 各提案手法の探索過程のショートカット数



Baseline (DARTS)　　DARTS + TDGA ($\boldsymbol{w}$, $\boldsymbol{\alpha}$)

図 4.9: 実験 2 : 手法のアーキテクチャ 1 / 2



DARTS + TDGA (α)　　DARTS + TDGA

図 4.10: 実験 2 : 手法のアーキテクチャ 2 / 2

5　まとめと今後の課題

本研究では，ネットワーク探索の自動化手法 DARTS について, その改良を検討した.

実験 1 では, VGG19 に対して DARTS を適用し, ネットワーク構造全体の接続を考え, DARTS によって VGG19 の性能を改善することを目的とした.

結果としてベースラインやランダムに接続した場合よりも, DARTS の探索したネットワーク構造が高い性能となり, また DARTS の出力結果からネットワークを構築する手法についても実験し, $\boldsymbol{\alpha}$ の順序のみではなく, 辺ごとに $\boldsymbol{\alpha}$ を閾値で採用する辺を構築する手法が, より少ない辺でネットワークの性能を高めている事が分かった.

DARTS のために設計されたネットワーク以外に対してもネットワーク構造が探索でき, DARTS の欠点といえるアーキテクチャ構造の制限を緩和できることが示された.

また実験 2 では, DARTS に TDGA を組み合わせる手法を実験した. 実験 1 では, 複数回試行した実験で部分的な共通点は見られたものの, 全体としてはほとんど異なる構造となっていた. このような初期値依存性による学習結果の偏りを解消するため, GA による多点探索を採用した. GA で複数個体を同時に学習することで $\boldsymbol{\alpha}$ や $\boldsymbol{w}$ の学習しやすさを平均化することを考えた. DARTS における $\boldsymbol{\alpha}$ を個体とし, 多様性を維持するため TDGA を導入した. また計算時間の問題も発生するため, 個体群全体で $\boldsymbol{w}$ を共有する One-Shot モデルとし, $\boldsymbol{w}$ の分の計算時間を削減した. さらに 2 つの手法を組み合わせる際に, DARTS は実数値だが, TDGA は整数値しか扱えないという問題もあったため, $\boldsymbol{\alpha}$ の標準偏差でエントロピーの代わりとして, 多様性の維持をした.

実験 2 の結果として, DARTS のみの性能を提案手法が上回ることができ, 提案手法の有効性が確認できた. 3 つの提案手法の中では, $\boldsymbol{w}$ を固定した提案手法 2, 3 がうまく学習でき, TDGA の適応度計算に $\boldsymbol{w}$ を更新する DARTS が与える悪影響が存在することが分かった.

また TDGA は DARTS に比べ計算時間が大幅に短いため, TDGA で広範囲を探索した後, DARTS によって正確に探索するなどの今回の提案手法の改良も考えられた.

今後の課題として, 本研究で用いた VGG 以外のネットワークに対してもこの提案手法を適用することで汎用性を確認し, また他のデータセットや実問題では, 最適なアーキテクチャがどのように変化するか検討することが挙げられる.

謝辞

本研究を進めるにあたり御指導，御鞭撻を賜りました森直樹教授, 岡田真助教に深く感謝申し上げます. また研究における技術面，精神面をサポートしてくださった研究室の先輩方，同期たちにも重ねて御礼申し上げます．なお，本研究は一部，日本学術振興会科学研究補助金基盤研究 (B) (課題番号 19H04184) の補助を得て行われたものである．

2021 年 2 月 26 日

参考文献

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

[2] Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

[3] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[4] Richard S Sutton, David A McAllester, Satinder P Singh, Yishay Mansour, et al. Policy gradient methods for reinforcement learning with function approximation. In *NIPs*, volume 99, pages 1057–1063. Citeseer, 1999.

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

[6] Adam Paszke, S. Gross, Soumith Chintala, G. Chanan, E. Yang, Zachary Devito, Zeming Lin, Alban Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[7] B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. *Annals of Operations Research*, 153:235–256, 2007.

[8] Hajime Kita, Yasuyuki Yabumoto, Naoki Mori, and Yoshikazu Nishikawa. Multi-objective optimization by means of the thermodynamical genetic algorithm. In *International Conference on Parallel Problem Solving from Nature*, pages 504–512. Springer, 1996.

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

[10] Xiao-Jiao Mao, Chunhua Shen, and Yu-Bin Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. *arXiv preprint arXiv:1603.09056*, 2016.

[11] S. Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ArXiv*, abs/1502.03167, 2015.

[12] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

[13] Nilotpal Sinha and Kuan-Wen Chen. Evolving neural architecture using one shot model, 2020.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[15] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv: Learning*, 2017.

[16] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[17] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.