情報理論を用いた深層学習におけるネットワーク構造の自動最適化手法の提案

滑川 静海† 手塚 太郎‡

†筑波大学情報学群 〒305-8550 茨城県つくば市春日 1-2
‡筑波大学図書館情報メディア系 〒305-0821 茨城県つくば市春日 1-2
E-mail: †s1511536@u.tsukuba.ac.jp, ‡tezuka@slis.tsukuba.ac.jp

あらまし 近年、深層学習が様々な分野に用いられている。しかしこれらのモデルは専門家によって長い時間をかけて設計され作られている。そのような背景があり、深層学習モデルを自動的に生成する Neural Architecture Search（以降 NAS）という研究がなされている。本研究は NAS 分野の一つの手法である、Neural Architecture Search with hill climbing（以降 NASH）の改良をするものである。また、本研究の目的は深層学習に関して知識や経験が薄い人でも深層学習モデルを生成し、深層学習を行うことができるようにすることである。研究方法については、NASH の手法をベースに情報理論を取り入れ、より良いモデルを探索するものである。この手法を用いた結果 NASH の性能を上回ることはなかったが、より無駄のないネットワークを作成することに成功した。この結果をもとに、さらに NAS の発展をしていきたいと考えている。

キーワード 深層学習, 情報理論, Neural Architecture Search, CNN

1. はじめに

1.1 研究背景

近年、機械学習の一種である深層学習が様々な分野に用いられている。深層学習とは図 1 のようなニューラルネットワークを多層化し、入力層、中間層、出力層の 3 種類の層で構成されるモデルを指し、入力データに含まれる潜在的な特徴を捉え、より正確で効率的な判断を実現する技術である。

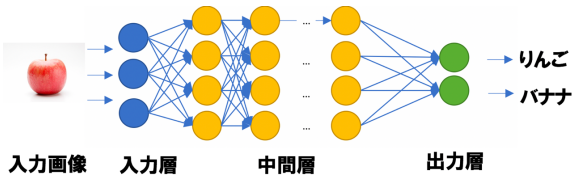

図 1 ニューラルネットワークのイメージ図

これは正解ラベルのついたデータを複数用意してそれを入力とし、予測の出力が正解ラベルと一致するようにノード間の結合の強さを自動調整（学習）していくものである。この深層学習モデルの例を挙げると、VGG や GoogleNet、AlexNet など様々なものが考案されている。これらの深層学習モデルは専門家の手によって長い時間をかけて設計され作られていて、どれも高いパフォーマンスを示す。しかしこれを深層学習初学者のような専門知識のない人が設計するのは困難である。このように、深層学習を扱うには専門的な知識や経験が多く必要とされている。ネットワーク構造が少し異なるだけでもモデルの性能が大きく変わってくるため、データの特性に合わせたネットワーク構造の調整に知識や経験が要求される。このような背景があり、深層学習について知識や経験が浅い技術者でも簡単に深層学習が利用できるようにするため、ネットワーク構造自動探索の研究が行われている。

1.2 研究目的と研究意義

この研究の目的は先にも述べたように、深層学習初学者やツールとして深層学習を使いたいだけといった人たちが簡単に深層学習を行うことができるようにするための研究である。また、この研究が達成されることによって深層学習を誰でも簡単に扱うことができるようになり、深層学習分野だけでなく深層学習を用いる全ての分野に対して大きな貢献が得られると考えている。

1.3 本論文の構成

本論文では、研究概要、関連研究、提案手法、結果・考察の順に本研究の目的の実現のための過程を説明する。研究概要では、本研究で扱う分野である Neural Architecture Search という分野や手法の概略を説明している。また、関連研究では本研究と関連する研究や、本研究のベースとなる先行研究の説明をしている。提案手法では本研究のベースとなる手法と比較して新たに改良を加えた点の説明をしている。結果・考察では本研究で行なった実験の結果を示し、それの改善案を含む様々な考察を行なっている。

2. 研究概要

私が行う研究は AutoML と呼ばれる理論の中の Neural Architecture Search（以下 NAS）と呼ばれる分野である。通常ニューラルネットワークは人間がネットワーク構造を設計し、そこに使われるパラメータの最適化を行なった上で初めて学習をすることができるようになる。しかし NAS はこのネットワーク構造の設計からパラメータの最適化、学習までの全ての範囲を自動で行うものを指す。本研究では NAS の中でもネットワーク構造の最適化部分に焦点を当てたものである。本研究では進化的アルゴリズムを採用しているため、パラメータの最適化においては構造を設計する段階でランダムにパラメータが設定され、最適でないパラメータを持つ層は進化的アルゴリズムの性質によって淘汰されていくものである。

NAS におけるネットワーク構造探索手法はいくつかあるが、強化学習を用いてネットワーク構造の探索を行う手法やネットワーク構造を微分可能な形で表現し、効率的に探索を行う手法、進化的アルゴリズムを取り入れて探索を行う手法など様々な手法が考案されている。私はこの中でも進化的アルゴリズムを取り入れて探索を行う手法（Neural Architecture Search by Hill Climbing）をベースに研究を行なった。この手法は、小さいモデルをベースとして進化的アルゴリズムを取り入れ、ネットワーク構造の探索を行うものである。本研究ではこの変形方法を改良することで手法の改善を目指すものである。

また、先行研究のプログラムのソースコードが公開されていないため、ベース研究のプログラムは私自身で実装した。言語は python でライブラリはバックエンドを TensorFlow にした Keras を用いて実装した。

3. 関連研究

この章では NAS 分野の関連研究をいくつか紹介する。今回は二つの手法について説明する

3.1 Neural Architecture Search with Reinforcement Learning

一つ目は Neural Architecture Search with Reinforcement Learning という手法で、強化学習を用いて NAS を行う手法である。

強化学習とは今ある状態から得られる報酬が最大になるような行動を学習していくという手法である。また、この論文では強化学習の手法の中でも方策勾配法という物を用いている。方策勾配法とは、現在の状態からどのような行動を取るのが最適であるかを方策の勾配を使って学習する手法である。

この論文で行なっている NAS の手法はコントローラーと子ネットワークに分けて考え、コントローラーで子ネットワークの構造を探索し、データの特性に合った最適な構造のネットワークを生成するというものである。具体的な学習の流れとしては、コントローラーが設定したパラメータで子ネットワークが学習し、検証データをもとに精度を算出する。その精度を報酬として方策勾配法をもとにコントローラーを更新する。この一連の流れを繰り返していくことでより良いネットワーク構造を探索していくものである。ここで探索することのできる子ネットワークのモデルは Convolutional Neural Network（以降 CNN）と Recurrent Neural Network（以降 RNN）の二つである。この論文は NAS 分野の最初の論文であるにも関わらず高いパフォーマンスを示す一方、膨大な計算量が大きな課題としてあげられている。論文では CIFAR-10 という画像分類データセットをもとに CNN の探索を行なっていたが、12800 アーキテクチャを学習するのに 800 台の GPU で並列に学習をしても探索完了までに 28 日かかったとされている。しかしこの論文は NAS 分野のパイオニアとしてとても高い成果をあげており、これ以降この論文を元に様々な改良が施されている。

その中でも Efficient Neural Architecture Search（以降 ENAS）という技術が提唱され、転移学習を用いることで計算量を大きく減らし、GPU1 台を用いた半日程度の学習時間で元の NAS と同等の結果を得ることができるようになった。

3.2 Neural Architecture Search by Hill climbing

3.2.1 概要

二つ目は、Neural Architecture Search by Hill climbing（以降 NASH）という手法である。この手法は ENAS とは異なり、強化学習を用いずに NAS を行う手法で、進化的アルゴリズムを用いた構造探索を行うものである。

3.2.2 NASH の手法説明

手法の具体的な説明をしていくと、まず一つのミニマムなネットワーク構造を持つモデルを用意する。そしてそのモデルに対していくつかのランダムな変形を加えたモデルを複数個用意する。それらのモデルの中で最も性能の良かったものを次世代のベースモデルとし、そこから新たに変形を加えたモデルを複数個用意する。これらを再帰的に繰り返していくことでより良い構造のモデルを探索していくというものである。

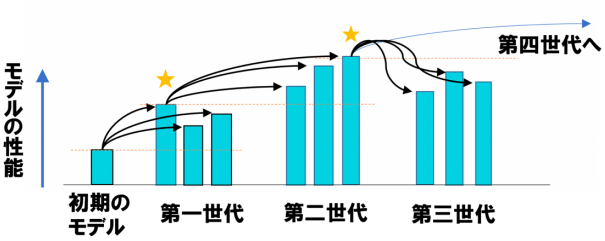

図 2 NASH の学習の進み方

図 2 を見るとわかりやすいが世代を経るごとにモデルがより良いものに収束していく。この論文では前の世代より性能の悪いものが伝わっていかないために、現在の世代の最良モデルが前世代の最良モデルに劣る場合は、前世代の最良モデルを引き継ぐようにしている。こうすることで学習が後退することを防ぐことを可能にしている。

NASH における変形は大きく分けて二つあり、層の追加とスキップコネクションの追加である。追加する層の種類は畳み込み層や全結合層、プーリング層やアクティベーション層、バッチノーマライゼーション層である。各層の役割に関してはここでは省略する。これらの種類の層をベースのモデルに対して以下の図 3 で示している様に追加する。

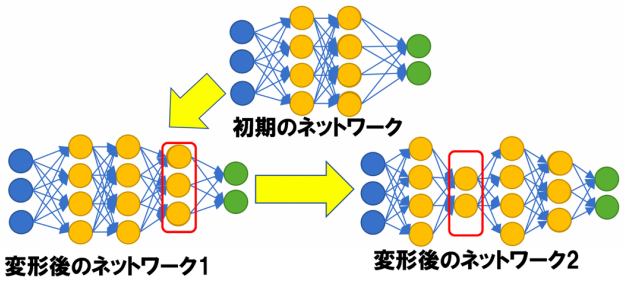

図 3 ニューラルネットワークへの層の追加

追加場所に関しては全てランダムに決定され、人為的なものが挟まらない様に探索を行う様になっている。次にスキップコネクションの追加について説明する。スキップコネクションとは、二つの層と層をつなぎ変換をせずに情報を渡すもので、間をスキップしたモデルとしなかったモデルの二つの複雑度のモデルを試すことができるという手法である。また、図 4 で示した様にスキップコネクションはランダムに選ばれた層と層を繋ぎ、変形を行うものである。

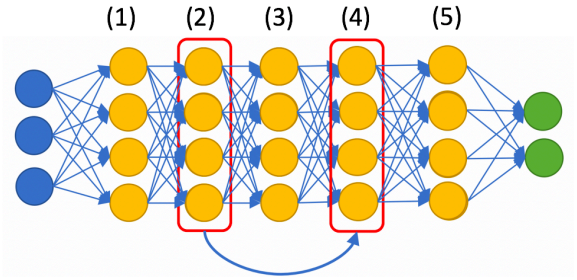

図 4 スキップコネクション

これらの二つの変形を通してベースのモデルから次世代のモデルを作成する。この際、学習を早く進めるためと考えられるが、一世代に複数回の変形を施したモデルを作成する。

3.2.3 NASH の優位性

この手法の優れている点は、ENAS と同様計算量を抑え、一つの GPU で 1 日の学習で同等の結果を残している点である。また、ENAS とは全く異なるアプローチで NAS を実現していて、NAS 分野において新たな道を示した点も優れていると言える。

3.2.4 NASH の課題点

NASH は手法として優れた点を多く持っているが、同時に課題点もいくつか抱えていると私は考えている。大きく分けて 3 つあると考えていて、それらについて説明をする。

一つ目は一世代で複数回の変形を行うため、冗長で意味のない層が追加されてしまう可能性があるという点である。これは人間の目にはわからないがほとんど同じ役割を持つ層がいくつか混ざり込んでしまったり、恒等変換に近いような変換を行う層が混ざり込んでしまったりすることが考えられる。すなわち、モデルのネットワーク構造が冗長で無駄が多いものになってしまう可能性があるということである。一世代で一つの変形にすれば解決するのではないかという意見もあるが、一世代に一つの変形のみだと収束までの試行回数が増え、学習にかなりの時間を要するため、複数回の変形を行なっている。

二つ目は、層を追加していく一方であるため、変形を行うたびに層が深くなっていってしまうことである。よって、最終的に出来上がったモデルのネットワーク構造が複雑で莫大な数の層を持ってしまうという状況になる可能性がある。複雑で莫大な数の層を持ってしまっていると、そのモデルを学習するときや利用するときに時間が多くかかってしまうという懸念点もある。

三つ目も同様に、一世代に複数回変形する都合上、良い変形の中に悪い変形が混じっていてもその世代では最良モデルに選ばれてしまったり、現時点では問題

なくとも、後々悪影響が出てしまうような変形がなされてしまったりする可能性がある点である。このような場合、うまく悪影響を回避するようにスキップコネクションの変形がなされるまで学習が停滞し続けてしまう。結果的には学習に時間がかかり、良い性能を示すモデルに収束するまでに無駄な時間が生じてしまう。また、スキップコネクションによってこれが回避できたとしても出来上がるモデルが複雑になりすぎてしまうことにつながる。

以上のような課題点が考えられる。そして、どれも層を増やしていく一方という点に起因して起きている問題であると考えられる。したがって次章ではこれらの課題点を解決することのできる提案手法について説明する。

4. 提案手法

4.1 提案手法の概要

私が提案する手法は、NASH に改良を加えたものである。具体的には層を減らすという変形を加えることである。NASH では層の追加が主な変形であり、それに起因して上記の課題点がある。それらを解決するためには層を取り除くという処理を加えるのが適切であると考えられる。

しかし、NASH が高速で学習することができることの理由の一つに、層の追加しか行わないことがあげられる。前の世代で学習した重みを再利用することで学習時間にかかる時間的コストを削減している。ここで、層を取り除く場合、層どうしのネットワーク間での入出力が合わなくなってしまうため重みの再利用が難しい。したがってネットワーク構造を組み、モデルを作成するたびに再学習する必要はあるため、NASH に比べ速度は落ちるというデメリットはある。しかし、一つの GPU でも実用的な範囲の時間で終わるため、問題はないと考えている。

この層を取り除くという手法を取り入れることのメリットとしては、3.2.4 で述べたような課題点を解決することができる。一つ目の出来上がるネットワークに無駄が多く含まれてしまい、冗長なモデルになってしまう課題点においては、無駄な層を取り除いていくことができれば解決することができる。二つ目の最終的に出来上がるモデルが莫大な数の層を持つものになってしまうことがある点も同様に、層を取り除く処理を加えることで適度な深さを保った層にすることができる。三つ目の悪影響の出るような変形が含まれてしまう課題点に関しても、その層を取り除くことで解決することができる。また、最終的に出来上がるモデルが複雑になり過ぎないという点も挙げられる。NASH では出来上がるモデルは人手で作られたものと比較して、モデルの性能は匹敵するが複雑なものが出来上がる可能性が高く、その点において人手で作られたものに劣ると考えている。しかし私の提案する手法では最終的に出来上がるモデルの複雑さは NASH のものと比べ少なくなると考えている。

また、私の提案手法では層の減らし方を二通り考えていて、ランダムに選んだ層を減らす手法と、統計的な性質を見て選んだ層を取り除く手法である。

4.2 ランダムに層を取り除く手法

一つ目のランダムに選んだ層を減らす手法は、モデルの中の入力層と全結合層部分を除く全ての層を対象にランダムに選択し、その層を取り除く。これで取り除くものは畳み込み層、プーリング層、アクティベーション層、ドロップアウト、バッチノーマライゼーション層である。ドロップアウトは層ではないが、取り除けるようにしている。また、この手法はネットワーク構造の変形の一つとして扱っており、例えば一世代で 5 回の変形を行う場合は、3 回は層の追加、残りの 2 回は層を削除といったように扱われる。また、取り除ける層が存在しなくなってしまった場合は、層の削除の変形の代わりに層の追加の変形が行われる。

ランダムに減らすことによるメリットしては、取り除く層を選ぶ際に計算を全く挟まないため、学習の進みに影響を及ぼすことがない点である。また、人間の考えが挟まることなく取り除く層が決定されるため、人手で作られるものと比較して盲点をついたような変形が生まれるという可能性も考えられる。このようにランダムで取り除くことは学習時間にあまり影響を与えず、かつ人手ではあまり想像のつかないような構造にたどり着くことができる可能性を秘めている。

ランダムに減らすことによるデメリットとしては、あくまでもランダムに取り除くため、前世代で貢献度の良かった層が取り除かれてしまう可能性がある点である。特に序盤で層の数も少ないうちは貢献度の高い層を取り除いてしまうと一気に出来上がるモデルの精度が落ちてしまうということもある。このデメリットを考慮してもランダムで層を取り除くという手法はより良い構造を探索する上で効率的に作用するのではないかと考えている。

4.3 統計的性質を見て取り除く層を決定する手法

二つ目の統計的性質を見て取り除く層を決定する手法は、モデルの持つネットワーク構造の中で最も貢献度の低い層を探し出し、取り除くという手法である。これによって取り除かれる層はランダムで取り除くものと違い、層として役割を果たすもの、すなわち畳み

込み層、プーリング層、アクティベーション層、バッチノーマライゼーション層である。ランダムの手法と比較してドロップアウトのみ取り除くことができないようになっている。

次にどのように統計的性質から貢献度の低い層を割り出すかについて説明する。今回私が用いた方法は、情報理論を用いるものである。今回は情報理論の中でも相互情報量を取り扱う。相互情報量とは二つの確率変数の間の従属性の度合いを測る尺度である。

本研究ではネットワークの中の各層の出力と最終的な分類の出力を確率変数として扱い、相互情報量を求める。L 個の層を持つネットワーク構造のモデルを例に説明すると、第 1 層の出力と、最終的な分類の出力、第 2 層の出力と最終的な分類の出力、…、第 L 層の出力と最終的な分類の出力といったような組み合わせで相互情報量を求める。すると、L 個の各層における出力との依存関係、すなわち、このモデルのネットワーク構造における各層の貢献度を計算することができる。この相互情報量の数値が低ければネットワーク構造間で貢献度が低く、逆に数値が高いほど貢献度が高いと言える。しかしこのまま安直に貢献度が低い層を取り除いてしまうと、より入力に近い層ばかり取り除かれていってしまう可能性が高い。なぜかというと、あくまでもその層の出力と最終的な分類の出力との相互情報量を測っているため、ネットワークの入力に近い方の層はまだ学習初期段階であり、その層の持つ情報量が低いことは必然だからである。逆に分類の出力に近ければ近い層ほど最終的な分類の出力との依存関係が強いはずである。したがって本研究では、ネットワーク間の各層の持つ相互情報量どうしの差をとってその値を元に貢献度を決める。つまり、ネットワーク間で相互情報量の上がり幅が小さいところに着目して貢献度を決めるということである。

具体的には、第 N 層（1＜N＜L）と第 N+1 層の分類の出力との相互情報量が、0.3、0.35、第M層（1＜M＜N)と第 M+1 層の持つ分類の出力との相互情報量が 0.1、0.2 だった場合、最も相互情報量が低いのは第 M 層であるが、第 M 層から M+1 層にかけて相互情報量が 0.1 上昇している。しかし第 N 層と N+1 層の間では 0.05 しか上昇していない。このような時に層として相互情報量の低い第 M 層を取り除くのではなく、相互情報量の上がり幅が少ない第 N 層を貢献度が低いとするものである。

このようにして貢献度の低い層を割り出す手法であるが、次に相互情報量を計算する際に具体的に何を確率変数とし、どのように貢献度を計算するについて説明する。確率変数は層の出力と最終的な分類の出力の二つを使っているが、これらをどのように確率で表すかを説明する。まず分類の出力に関しては、CIFAR-10 では分類数が 10 種類であり、各画像が一つにのみ分類されるため、one-hot-vector で表すことで和が 1 の確率ベクトルとして考えることができる。これを分類の出力の確率変数とする。次に層の出力に関しては、一般に層の出力は分類の出力とは異なり確率ベクトルで表されるものではないため、層の出力を確率ベクトルで表す必要がある。具体的には層の出力を分類対象のカテゴリ数、すなわち CIFAR-10 であれば 10 個のクラスにクラスタリングをする。クラスタリングとは大量のデータから共通点を探し、自動的に分類をしていく手法である。この際クラスタリング手法はハードクラスタリングである k-means 法を用いた。入力一つ一つに対して各層の出力を集め、それらの出力を分類数と同じクラス数でクラスタリングする。この際、全ての出力に対してクラスタリングを施してしまうと計算量が膨大になってしまい、学習の進みが非常に低速になってしまう。したがって、集めた層の出力の中からランダムに 1000 個をサンプリングし、それをクラスタリングする。そしてそれらをクラスタリングすると 1000 個のサンプルに対してそれぞれの属するクラスがわかる。それを one-hot-vector で表すことで、分類の出力と同様に、確率ベクトルとして考えることができるため、層の出力を確率変数で表すことができる。この際、サンプリング数を 1000 個としている理由は、絶対数として 1000 個あればうまくいくだろうという仮定のもと決めた数であり、分類数がとても多い場合、この数はより増やした方がうまくいく可能性はある。

以上のように層の出力と最終的な分類の出力を確率ベクトルで表し、それを確率変数とすることで層の出力と分類の出力間の相互情報量を計算する。このような手順で貢献度の低い層を割り出し、層を取り除く。

5. 実験

5.1 実験環境

まず実験環境について説明する。使用したプログラミング言語は python で使用したライブラリは Keras で、バックエンドに TensorFlow を用いている。使用した GPU は一つで NVIDIA Tesla K40m である。OS は Ubuntu16.04。CPU は 4 つでどれも Intel(R) Xeon(R) CPU E5-1620 v3 @ 3.50GHz である。また、本実験では CIFAR-10 を用いた画像分類タスクで実験を行なっている。

5.2 実験概要

この節では実際の実験に関して説明をする。今回は二通りの実験を行なっていて、一つ目は NASH の変形にランダムに層の削除を取り入れたものであり、二つ

目はランダムな層を削除する変形を取り入れた上で、出来上がったモデルに対して貢献度を測り、層を取り除く処理を行うものである。大まかに実験手順を説明していくと以下のようになる。

① 第一世代のベースとなるネットワーク構造を持つモデルを定義する

② 繰り返す世代数、子モデルの数、epoch 数など、必要なパラメータの定義する

③ ベースとなるモデルに変形を加えたモデルを複数個用意する

④ 新たに作られたモデルとベースとなったモデルを学習する

⑤ その中で最も性能の良かったものを次世代のベースとする

⑥ ③～⑤を繰り返す

⑦ 最終的に出来上がったモデルを学習する

以上のような実装がなされている。以降で、本実験で用いたパラメータや各ステップでの詳しい説明を①から順にしていく。

まず、①では第一世代のベースとなるミニマムなネットワーク構造を持つモデルを作成する。今回の実験では CIFAR-10 の画像分類タスクであるため、CNN を用いてモデルを作成している。具体的には、最低限の構造をしていれば良いため、入力層、畳み込み層、アクティベーション層、全結合層、出力層というネットワーク構造を最低限の構造とし、ベースのモデルとして定義している。

次に②の必要なパラメータについて説明する。今回定義しているパラメータは世代数、変形する回数、一世代あたりの作られるモデル数、新しく作られるモデルの学習回数、最終的に出来上がるモデルの学習回数の 5 つである。本実験では世代数を 5、変形回数も 5、新しく作られるモデルの学習回数は 20、最終的に出来上がるモデルの学習回数は 50 である。なおこれらのパラメータは使用するデータセットによって変わるものであり、CIFAR-10 を用いた本実験ではこの程度であるが、例えばもっと規模の大きい分類を学習させたい場合は学習回数を増やす必要はある。

③に関しては②で定義したパラメータに沿って変形を加え、モデルをいくつか作っていく。ここでの変形について具体的に説明すると、ここで行われる変形の種類大きく分けて 2 種類あり、層の追加と削除である。層の追加の変形は畳み込み層とアクティベーション層のセット、プーリング層、ドロップアウト、バッチノーマライゼーション層の追加がある。ただし、ドロップアウトは層ではないがネットワークに組み込むに当たって変形 1 回分としている。層の削除に関してはここで行われるのはいずれの実験でもランダムに選んだ一つの層を削除するものである。ただし、ここにも層ではないがドロップアウトを取り除くことも一つの変形としてカウントしている。さらにベースのネットワーク構造より小さくはならない。また、層の追加時のハイパーパラメータはランダムに決定している。畳み込み層におけるフィルタサイズは 3×3～5×5 の範囲でランダムに決定している。アクティベーション層の活性化関数は ReLU 関数を採用している。また、ドロップアウトの割合に関しても 0~0.5 の範囲でランダムに決定している。ネットワークの変形をする際、層の追加または層の削除の二択をランダムに決定し、層の追加が選択された場合は 4 種類の変形からランダムに選ばれランダムな位置に追加され、層の削除が選択された場合はランダムに選ばれた層が削除される。

次に④に関しては、これも②で定義されたパラメータに準拠して、新しく作られたモデルとベースになったモデルを学習する。この際オプティマイザーは Adam、損失関数はクロスエントロピーを使用している。一つ目のランダムに層を取り除くだけの手法はこれだけである。二つ目の層の貢献度を元に無駄な層を取り除く手法に関しては、ここの学習で貢献度の計算を行う。具体的には学習の最後の epoch で各層の出力を集め、貢献度を計算する。そして貢献度を元に層を取り除いたモデルをもう一度学習し、取り除く前のモデルと取り除いたあとのモデルを比較し、性能の良いものを子モデルとして採用する。

⑤では④で学習されたモデルを実際にテストデータでテストし、その結果の精度が最も高かったものを次世代のベースとする。

⑥では②で定義されたパラメータを元に世代の数だけ繰り返しを行う。

最後の⑦では最後の世代で最も良かったモデルを学習し、機械学習モデルを完成させるものである。
実験は以上のような実装で行われた。

またこれに加え、NASH を追実装したものも実験を行った。

6. 実験結果

今回は 5 で示したように三つの手法で実験を行なった。その結果を以下に示す。

	追実装	ランダム削除	貢献度をもとに削除
精度	0.82	0.76	0.75
層数	29	16	14

今回行なった二つの手法ではどちらも先行研究によって出来上がるものに勝ることはなかった。実験で用いるパラメータは三つ全て世代数が 5、各世代で生成さ

れるネットワーク数は 5、変形数は 5 で行なった。ランダム削除のみの手法では精度が 0.76 で出来上がったモデルの層数は 16 層､貢献度を元に削除を取り入れた手法では精度が 0.75、出来上がったモデルの層数は 14 層であった。また、貢献度をもとに削除されたものが採用された回数は 25 回中 14 回であった。

7. 考察

実際に二つの手法と既存手法である NASH を比較して考察して行く。どちらの手法も既存手法である NASH のパフォーマンスを上回ることはなかった。これに関しては、原因がいくつか考えられる。

一つ目は元の論文を完全再現できていない可能性があることである。元の論文は実装の詳細が省かれている部分が所々あり、そこが完全に再現できていないため、パフォーマンスが追いついていないのではないかと考えられる。具体的には変形が行われる時、どのくらいの確率でどのような変形が行われるかが明記されていないため、本研究では全て等確率で実装している。よってここの最適化された確率や、他の実装の細部を探って行くことでパフォーマンスが向上することが考えられる。

二つ目は探索効率が悪い可能性があることである。現在は全ての変形が等確率で行われるが、人手で作られたものを教師データとして層の出現割合など決めることで、層の追加の割合を適度なものにすることができ、探索効率が上がるのではないかと考えられる。また、探索効率を上げる手段として、世代ごとの変形数を減衰させていくことが挙げられる。具体的には一世代で変形する回数が多いため、無駄な層を取り除くことはできても良い層まで取り除かれてしまったり、一つだけ層が追加されば良くなるが余計なものが混じってしまったりすることがあると考えられる。つまり変形数が多いことで学習序盤では効率よく進むが、学習終盤では変形数が多いことがネットワーク構造の収束を妨げているのではないかと考えられる。この変形数の減衰を取り入れることで学習序盤の進みの良さを維持しつつ学習終盤の収束の速さを実現することができると考えられる。

三つ目は実験に用いるパラメータが最適でないということがあげられる。今回の実験はどちらも世代数が 5、各世代で生成されるネットワーク数が 5、変形数が 5 で実験を行なっているが、ネットワーク構造を探索するものとしては数が少なすぎた可能性が考えられる。二つ目で挙げたように学習序盤は変形数を高く持ち、後半は変形数を減衰させるのに加え、こなす世代数や生成されるネットワーク数をより多くする必要があると考えられる。当然用いるデータセットによると考えられるが、いずれのデータセットでも探索数はある程度必要である。また、世代数をパラメータとして決めず、一定世代数最良モデルが変わらなかったり一定基準の精度に達するまで学習を続けたりするような終了条件を決め、これを満たすまで学習を続ける手法にすることも検討している。

四つ目は、NASH はアンサンブル学習を採用している点である。NASH は 4 つのモデルを作成し、アンサンブル学習で 0.95 の精度を達成している。しかし本研究ではアンサンブル学習は採用していない。したがってアンサンブル学習を取り入れることで NASH に近いパフォーマンスを発揮することができると考えられる。

以上のようなことが原因で NASH のパフォーマンスに追いついていないのではないかと考えられる。それらを改善することで NASH のパフォーマンスを満たしつつより良い構造を探索することができると考えられる。

また、どちらの手法も精度はほとんど変わらず、かつ貢献度を見て層を取り除く手法は層の数がランダム削除のみの手法と比較して少なかったため、この手法は有効であることがわかる。さらに NASH と比較してどちらも精度としては劣るが少ない層数で一定の精度に達していることからこの点において優れていると考えられる。

8. 終わりに

8.1 今後の展望

今回の実験では以上で示すような結果が得られたが、その中で以下に示すような様々な改良点や手法が適用できる可能性が見えてきた。

一つ目は 7 章の考察でも述べたように、世代ごとの変形数に減衰を取り入れることである。これを取り入れることで学習序盤の進みの速さを保証しつつ、学習終盤の収束しやすさを上げることができると考えられる。

二つ目も 7 章の考察で述べたように、一定以上の精度を達成した場合やネットワーク構造が数世代に渡って変わらない場合などの学習終了条件を設け、学習を収束するまで続けるようにすることである。これを取り入れることで時間はかかるようになるが、より良い構造を見つけるまで学習が続き、今よりパフォーマンスの高いモデルが出来上がると考えられる。

三つ目は層の削除だけでなく、モデル圧縮という技術を取り入れることである。モデル圧縮とは複数の層を一つの層で近似するというものである。これを取り入れることで、そのときのネットワークの性能を維持したまま層を減らし、より無駄のないネットワーク構造を作ることができる。

四つ目は、貢献度を見て層を取り除く処理を行う際に、現在はハードクラスタリングである k-means 法を用いているが、ここでソフトクラスタリングの手法を用いることである。こうすることで、より情報量のロスが少なくなるため、より貢献度を正確に知ることができるのではないかと考えられる。

五つ目は NASH との差異である、「ネットワーク構造の変形に削除があること」を生かして新たにネットワーク構造を生成するのではなく、ある程度作られたネットワーク構造を最適化して行くことに使うことである。本研究の特性上、ネットワーク構造の探索を行う際、学習序盤はどうしても変形の削除を活かしにくいという性質がある。したがってあらかじめ人間の手で作られたネットワーク構造に対して本研究を用いることでより良い構造に達することができるのではないかと考えている。

このように今回の実験を通して様々な改善手法を見いだすことができた。これらを改善して行くことでより良い NAS の手法になると考えている。

8.2 結論

本研究では既存研究である NASH の改良に取り組んだ。結果としてはまだまだであるが、より無駄のないネットワーク構造の生成に成功した。そして NAS 分野においてネットワーク構造を最適化して行く中で新たな道を示すことはできた。また、この研究自体まだまだ改良できる部分が多いため、それらを改善していくことでより良いネットワーク構造を探索して行くことができるのではないかと考えている。さらにネットワーク構造を変形する中で、層の削除を行えるという利点を生かし、既に出来上がっているモデルを最適化することを目的とした研究に応用することができるとも考えている。このようにこの研究手法、実験結果から NAS 分野において新たな道筋を見いだすことができた。これらの結果をもとにさらに NAS の発展をしてきたいと考えている。

謝辞

最後に、良い指導していただいた指導教員の手塚先生、共同研究室の若林先生に深く感謝の意を表する。また、共同研究室の同期の先輩方、同期の方々にも様々な意見をいただけたことに感謝の意を表する。また、本研究は JSPS 科研費 JP16K00228, JP16H02904 の助成を受けたものである。

参 考 文 献

[1] Thomas Elsken, Jan-Hendrik Metzen, Frank Hutter. “Simple And Efficient Architecture Search for Convolutional Neural Networks”. 2017.

[2] Bowen Baker, Otkrist Gupta, Nikhil Naik, Ramesh Raskar. “Designing Neural Network Architectures using Reinforcement Learning”. 2016.

[3] ICLR2018 conference reviewer . “ Simple and efficient architecture search for Convolutional Neural Networks”. OpenReview.net, https://openreview.net/forum?id=SySaJ0xCZ, (2018/06/08).

[4] Hanxiao Liu, Karen Simonyan, Oriol Vinyals. “Hierarchical Representations for Efficient Architecture Search”. 2018

[5] Masanori Suganuma, Shinichi Shirakawa, Tomoharu Nagao. “A Genetic Programming Approach to Designing Convolutional Neural Network Architectures”. 2017

[6] Hanxiao Liu, Karen Simonyan, Oriol Vinyals. “Neural Architecture Search: A Survey”. 2018

[7] Karen Simonyan, Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. 2015.

[8] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet. “Going Deeper with Convolutions”. 2014.

[9] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi. “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning”. 2016.

[10] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe. “Rethinking the Inception Architecture for Computer Vision”. 2015.

[11] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. “ImageNet classification with deep convolutional neural networks”. 2012.

[12] Barret Zoph, Quoc V. Le. “Neural Architecture Search with Reinforcement Learning”. 2016.

[1] [13] Yu Cheng, Duo Wang, Pan Zhou, Tao Zhang. “A Survey of Model Compression and Acceleration for Deep Neural Networks”. 2017.