# 進化的な深層学習の構築に関する研究

## 1　はじめに

機械学習の成功は目覚ましく, 広い分野に適用され多くの研究がなされている. その中でディープニューラルネットワーク (deep neural network; DNN) は, 音声・画像・自然言語を対象とした問題で高い性能を示し, 強い関心を集めた. 層を深く重ねる DNN は, 層の種類やその数, 層同士の接続, 畳み込み層の場合はカーネルサイズやストライドなど, 多数のパラメータが存在する. 問題に合わせて適切なパラメータを選択しなければ良好な精度が出ない一方で, そのネットワークやアーキテクチャの設計に明確な指針はなく非常に困難である. 従来では, 専門知識を持つ人が手作業で設計し, 学習した結果を参照して繰り返し修正していた.

自動化された機械学習 (Automated Machine Learning; AutoML) と呼ばれる分野は, アーキテクチャ全体を対象とした最適化を目指す. 本研究では AutoML の開発を行う前段階として, AutoML の調査を行った. まず AutoML の分類を見た後, その例である Neural Architecture Search: NAS と Auto Augment について紹介する.

## 2　要素技術

### 2.1　AutoML

自動化された機械学習 (Automated Machine Learning; AutoML) は特定の技術ではなく, 機械学習モデルの設計を自動化する全般的な手法, または概念を指す.

問題ごとにパラメータを適切に設定する必要があるというパラメータ設定問題の解決を目的とする. 確率的なブラックボックスメタ最適化であるこの問題の解決によって,

- 時間コストの削減・性能の向上
- アルゴリズムの評価, 比較のためパラメータの最適性の影響を緩和
- パラメータが与える影響の知識の必要性を排除

の 3 つの利益が得られる.

AutoML はそのアルゴリズムによって, 以下の 3 つの主要カテゴリと 6 つのサブカテゴリに分類される [1].

**主要カテゴリ**　構造や構成による分類

1. 単純な生成・評価法
   生成段階で候補となる設定を生成, 評価段階で評価して最適な設定を見つける手法.
2. 反復的生成・評価法
   少数の設定を生成して, 最も優れたものをみつけ, 反復的に生成する新しい設定の指針とする手法.
3. 高レベルの生成・評価方法
   高レベル生成機構として既存の自動パラメータチューニング手法や探索手法を持ち, 少数精鋭の設定を生成し評価段階では慎重に評価する手法.

**サブカテゴリ**　評価方法による分類

1. 繰り返し評価法
   複数回の評価を, 平均することなどで評価する手法.
2. F-Racing
   統計的に劣る評価の設定を段階的に排除し, 有望な候補に計算を集中する手法. 繰り返し評価より効率的になる.
3. インテンシフィケーション (強化)
   問題のリストで候補設定と暫定設定の評価を次々比較し, 劣る場合は途中で排除し, そうでなければ候補が暫定設定となる手法.
4. シャープニング
   少ないテスト数で評価を始め, 将来性のある設定のテスト数を 2 倍にすることで素早く探索できる手法.
5. アダプティブキャッピング
   有望でない設定の実行を中断して計算量を削減できる手法.

### 2.2　NAS

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て重みの最適化を行うが, アーキテクチャの設計には, 高度な専門知識と手作業による構築が必要である. Neural Architecture

Search(NAS)[2] は, ニューラルネットワークのアーキテクチャ自体を最適化する.

アーキテクチャの探索は 3 つの段階で行う. まず最初にコントローラと呼ばれる再帰型ニューラルネットワーク (Recurrent Neural Network; RNN) で, アーキテクチャのハイパーパラメータを生成する. 例えば畳み込み層を利用するネットワークでは, レイヤーごとにフィルタの高さ・幅, ストライドの高さ・幅, フィルタ数が必要となる. 次にハイパーパラメータから子ネットワークを構築し, 通常のように重みを訓練して検証データセットの精度を得る. 最後に得られた精度で報酬を計算し, 方策勾配法 (Policy gradient method) によってコントローラのネットワークを更新する. これらの手順を繰り返すことで, 子ネットワークのアーキテクチャが最適化される.

NAS では子ネットワークとして, 畳み込みネットワークと RNN の探索を行い従来のネットワークより高い精度と少ないパラメータ数を達成したが, 数百台の GPU と 1ヶ月の時間がかかり計算量に問題があった. 以降の研究では計算量の削減を目的とした手法が提案された.

佐藤 怜ら [3] の研究では, NAS を改良した先行研究に比較してより高速なアルゴリズムを開発した.

- アーキテクチャをカテゴリカルから連続的な確率分布に
- ネットワークの重みを再利用して学習を削減
- 見つけたアーキテクチャの重みを再学習する (acc 6 割 → 9 割)

#### 重みの再利用

1. 冗長にネットワーク構造を決めておく. (あるノードはそれ以前のノード全てに接続可能とする)
2. 重みを学習して、各エッジ、各演算子ごとに重みを保存
3. アーキテクチャ（接続するか？＋演算子）を探索

得られたセルでは, 直前のセルからの入力を恒等写像でそのまま出力していた.

畳み込み層は全て separable convolution を利用している. パラメータ数を減らしたかった？

#### アーキテクチャの表現法

- RNN でパラメータを生成 (カテゴリカル)
- GA で個体表現 (カテゴリカル)

表 1: 実験の設定

| | |
|---|---|
| dataset | cifar10 |
| n data | 16,000 / model |
| task | 5, 7, 10 クラス識別 |
| input | image(3x32x32) |
| output | class(5, 7, 10) |
| model | CNN(16 層) |
| optim | SDG (lr=0.001, moment=0.9) |
| loss | Cross Entropy Loss |
| batch size | 64 |
| epoch | 100 |

- 演算子の確率分布集合として表現

### 2.3 AutoAugment

[4] AutoML の一種？(編集しよう！) 画像認識の分野では、画像データを少し回転させたり左右反転させたりなどの操作をすることで画像データ数を増やす Data Augmentation(以下、DA) が広く使われています。ただ、どの操作を行うのかというのは試行錯誤で見つけるしかなく時間がかかります。そこで強化学習を使うことで自動的に DA を選択してくれるという AutoAugment が提案されました。

## 3 実験

～に関する実験の前段階として, 以下の予備実験を練習的に行った. アンサンブル学習に関連？

### 3.1 モデルの構築

cifar10[5] 前回, 5 クラス識別器を 2 つ利用したモデルの構築をしたが, 10 クラスの識別が 50%であったため, より適したパラメータを探索し, 再び実験して精度の向上を目指した.

#### 3.1.1 5 クラス x2

cifar10 に含まれる 10 クラスを 5 クラスずつに分割した。インデックスの前半 (airplane, mobile, bird, cat, deer) と後半 (dog, frog, horse, ship, truck) で分けることにした。生成した部分データセットを、それぞれモ

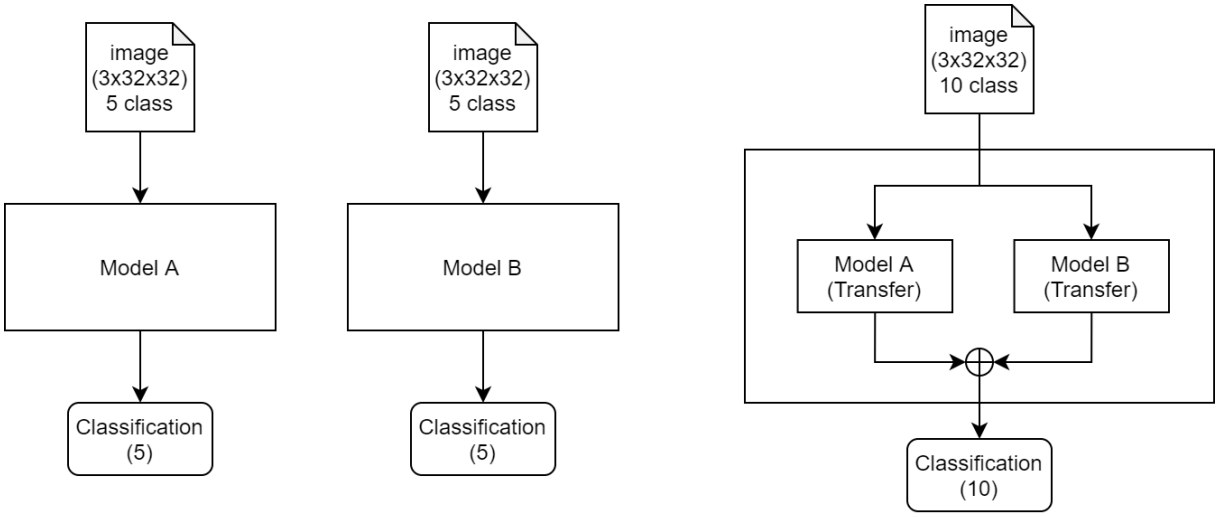


図 1: モデルの簡略図　() 内はデータの次元数

表 2: モデルごとの 7 クラスの振り分け

| model | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| A | o | o | o | o | o | o | o | | | |
| B | o | o | o | o | | | | o | o | o |
| C | o | | | | o | o | o | o | o | o |

デル A, B で学習した。5 クラス分類ができるモデル A, B を持つ結合モデル A + B を作成し、10 クラス分類の精度を計測した。このモデルの学習と相互関係を図 1 に示した。

結合モデル A + B は 10 クラスのデータセットを A, B に入力し、得られた出力をクラスインデックス順に結合して、出力とする。結合では特別な処理を行わず、そのままのデータを連結した.

#### 3.1.2　7 クラス x3

さらに今回は, 細川君に指摘してもらったアイデアでも実験を行った. 7 クラス分類器を 3 つ組み合わせて,10 クラスの分類を行った. 表 2 のように, 2 つ以上の分類器で各クラスを推定するように, クラスを振り分けた.

### 3.2　結果

5 クラス分類の精度を図 2 に, 7 クラス分類の精度を図 3 に示した.

5 クラス分類の場合, 80% ～ 90%の正答率で前回よりも 10%程向上した. 前回のデータ数 2000 よりもデータ数を増やしたことで精度がよくなった. 結合した結果も, 70%(前回+20%) となった.

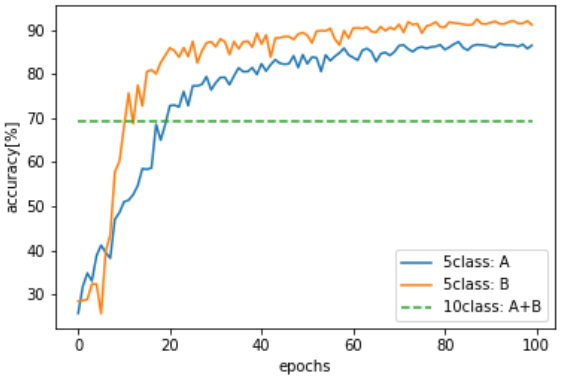


図 2: 5 クラス分類の正答率

7 クラス分類では, 正答率が平均 80%程度で, 結合した結果 10 クラス分類では 86.7%となった.

### 3.3　考察

データ数とバッチサイズを増やして, 精度の向上と学習の安定化ができた. 特に 5 クラス分類ではあるが, 正答率 9 割を超えることができた. データオーギュメントはしていないので, さらに精度を上げることはできると思われる.

図 2, 3 ともにインデックスが前半のクラスを持つモデルでは, 正答率が低い傾向が見られた. 誤差の影響ではなく, 困難なクラスの分類によって精度が下がってい

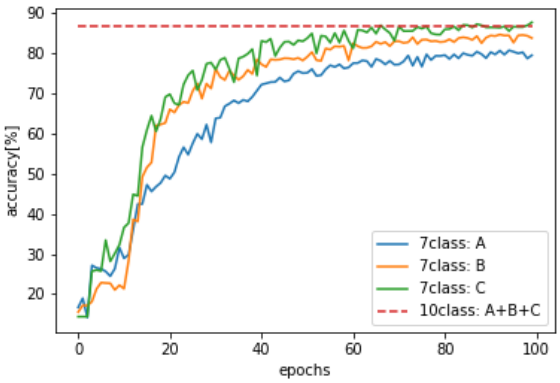


図 3: 7 クラス分類の正答率

ると考えられる．これはクラスを単純に分割したことによる偏りに原因がある．

分割する組み合わせを自由に替える実装ができたので, 様々な組み合わせパターンで実験して, その差異を見たい. 特にモデル間の正答率の差が目立ったため, これを埋めるような組み合わせを探したい.

## 4　考察

## 5　まとめと今後の課題

論文の調査をした AutoML とその関連技術についてまとめた

NAS について確率的なアーキテクチャ表現したシステムを構築し簡単な問題を解く小さなネットワークを探索する実験をしたい

## 参考文献

[1] Changwu Huang, Yuanxiang Li, and Xin Yao. A survey of automatic parameter tuning methods for metaheuristics. IEEE Transactions on Evolutionary Computation, 24:201–216, 06 2019.

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

[3] 佐藤 怜, 秋本 洋平, and 佐久間 淳. 貢献度分配を導入した方策勾配による neural architecture search の高速化. 人工知能学会全国大会論文集, JSAI2019:2P3J202–2P3J202, 2019.

[4] Ekin Dogus Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data. CoRR, abs/1805.09501, 2018.

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