ISSN 2186-5647

2-33

NASにおける半教師付き性能予測器の改良

日大生産工　○野口　尚馬　　日大生産工　山内　ゆかり

1. まえがき

近年、画像分類や文字認識など様々な分野で機械学習が用いられているが、専門的な知識のない人が手動で機械学習モデルを設計するのは難しい。そうした背景から、Zophらによりニューラルネットワークモデルの構造最適化を自動的に行う手法であるNeural Architecture Search:NAS[1]が提案された。

しかし、NASは学習に数時間から数日を要することが多い。性能予測器を組み込むことで学習時間を削減することは出来るが、完全教師付きの予測器では学習のために大量の性能ラベル付きニューラルアーキテクチャが必要となり、ラベル付けのために膨大な時間がかかるという問題がある。

Tangらは性能予測器を半教師付きにする[2]ことで、必要なラベル付きアーキテクチャの数を削減することに成功している。本研究では、性能予測器の予測精度を維持しつつ、ラベル付きアーキテクチャの更なる削減を試みる。

2. 従来手法

NASはZophらにより提案されたニューラルネットワークの構造最適化アルゴリズムであり、強化学習や遺伝的アルゴリズム(GA)などを用いてニューラルネットワークの構造決定、パラメータ最適化、学習、評価を繰り返し、最適なネットワーク構造を探索する。

図1にNASの流れを示す。

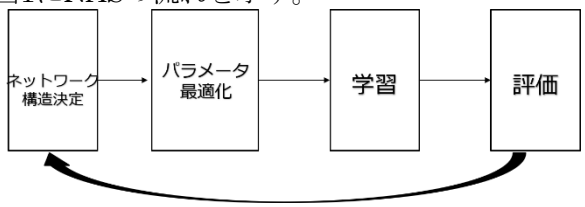

図1.NASの流れ

性能予測器はNASの計算時間を削減するために学習の代わりに用いられる手法であり、ニューラルアーキテクチャを入力として受け取り、そのアーキテクチャの推定性能を出力する。

完全教師付きの性能予測器はすべての学習データがラベル付けされている必要があるが、Tangらの手法は半教師付きであるため、ラベル付きアーキテクチャが一部しかない場合でも学習することが出来る。

Tangらの手法では、まずアーキテクチャを自己符号化器に送って特徴表現を得る。次に、得られた表現を元にアーキテクチャ間の類似度を測定し、関係グラフを構築する。最後に、特徴表現と関係グラフをグラフ畳み込みニューラルネットワーク(Graph Convolutional Neural Network:GCN)[3]に送り、アーキテクチャの推定性能を出力する。

自己符号化器は出力値を入力値に近づけることを目的としたニューラルネットワークモデルであり、入力層、中間層、出力層の3層からなる。入力層と出力層のノード数は同じであり、中間層のノード数はそれより少ない。そのため、中間層の値を入力値の特徴表現として使用することができる。

GCNは畳み込みニューラルネットワークの一種であり、グラフ内の各ノードが持っている特徴量に、隣接関係にあるノードの特徴量を畳み込む。

図2にTangらの手法の流れを示す。

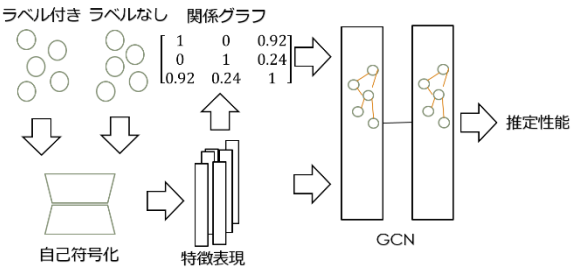

図2.Tangらの手法の流れ

以下に全体の流れを示す。

Step1）探索空間Xから少数のアーキテクチャをランダムに選択し、学習させてラベル付けする。

Step2）探索空間$X = X^l \cup X^u$と教師ラベルy^lを入力する。ここで、X^lはラベル付きアーキテクチャの空間、X^uはラベルなしアーキテクチャの空間である。

Step3）X^lとX^uからランダムにデータを選択し、ミニバッチ$\mathcal{B}$を形成する。

Step4）アーキテクチャ$\boldsymbol{x} \in \mathcal{B}$を自己符号化器$\varepsilon$に送り、特徴表現$\varepsilon(\boldsymbol{x})$を得る。

Improvements to a Semi-Supervised Performance Predictor in NAS

Shoma NOGUCHI and Yukari YAMAUCHI

Step5) 式(1)でアーキテクチャ間の類似度を測定し、関係グラフGを構築する。

$$s(x_i, x_j) = \exp\left(-\frac{d(\varepsilon(\boldsymbol{x_i}), \varepsilon(\boldsymbol{x_j}))}{2\sigma^2}\right) \quad (1)$$

dは距離関数(ユークリッド距離など)、σはスケールファクターである。

Step6) 特徴表現$\varepsilon(\boldsymbol{x})$と関係グラフGをGCN評価器$\mathcal{P}$に送り、推定性能$\hat{y}$を出力する。

Step7) 式(2)で回帰損失L_{rg}を計算する。

$$L_{rg} = \frac{1}{N_l}\sum_{i=1}^{N_l} \left\| \hat{y}_i^l - y_i^l \right\|_2^2 \quad (2)$$

N_lはラベル付きアーキテクチャの数であり、$\| \quad \|_2^2$は二乗誤差である。

Step8) 特徴表現$\varepsilon(\boldsymbol{x})$をデコーダー$\mathcal{D}$に送り、式(3)で再構成損失$L_{rc}$を計算する。

$$L_{rc} = \frac{1}{N_l}\sum_{i=1}^{N_l} \left\| \mathcal{D}\left(\varepsilon(\boldsymbol{x_i^l})\right) - \boldsymbol{x_i^l} \right\|_2^2 + \frac{1}{N_u}\sum_{j=1}^{N_u} \left\| \mathcal{D}\left(\varepsilon(\boldsymbol{x_j^u})\right) - \boldsymbol{x_j^u} \right\|_2^2 \quad (3)$$

N_uはラベル付きアーキテクチャの数である。

Step9) 式(4)で最終損失Lを計算する。

$$L = (1-\lambda)L_{rg} + \lambda L_{rc} \quad (4)$$

Step10) 自己符号化器ε、GCN評価器$\mathcal{P}$、デコーダー$\mathcal{D}$のパラメータを逆伝播で更新する。

Step11) Step2~10を繰り返す。

しかし、Tangらの手法はラベル付けするアーキテクチャをランダムに選択するため、ネットワーク構造が類似しているアーキテクチャが複数選択される可能性がある。

3. 提案手法

本研究では、従来手法の問題に対し、ネットワーク構造が類似しているアーキテクチャを除外し、残ったアーキテクチャのみにラベル付けをすることで、予測精度を維持しつつラベル付きアーキテクチャを更に削減することを試みる。

図3に提案手法の流れを示す。

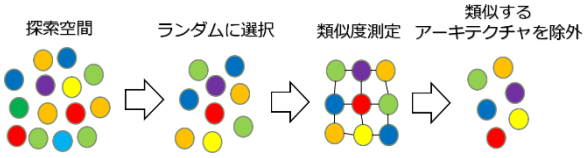

図3.提案手法の流れ

次に全体の流れを示す。

Step1) 探索空間Xから少数のアーキテクチャをランダムに選択する。

Step2) アーキテクチャ間の類似度を測定し、類似度が閾値を超えた場合はどちらかを除外する。

Step3) 残ったアーキテクチャを学習させてラベル付けする。

Step4) 従来手法と同様に性能予測器を学習させる。

4. 実験

性能予測器をGAベースのNASアルゴリズム[4]に組み込み、NAS-Bench-101[5]上で最良のアーキテクチャを探索する。NAS-Bench-101はCifar-10[6]上で学習された42万3000個のCNNアーキテクチャからなるデータセットであり、全てのアーキテクチャがラベル付けされている。しかし、実験ではラベル付きアーキテクチャとして用いるものを除いて、全てのアーキテクチャをラベルなしとして使用する。

従来手法の実験と同様に、性能予測器が出力した推定性能のランキングと実際の性能のランキングの相関を平均二乗誤差(Mean Square Error:MSE)と相関係数で測定し、従来手法と提案手法を比較する。

5. まとめ

本研究では、従来手法が抱えていたネットワーク構造が類似しているアーキテクチャが複数選択される可能性があるという問題に対し、ネットワーク構造が類似しているアーキテクチャを除外し、残ったアーキテクチャのみにラベル付けをすることで、予測精度を維持しつつ、ラベル付きアーキテクチャの更なる削減を試みている。

参考文献

[1]Barret Zoph, Quoc V Le,"Neural Architecture Search with Reinforcement Learning"
[2] Yehui Tang, Yunhe Wang, Yixing Xu, Hanting Chen, Chunjing Xu,Boxin Shi, Chao Xu, Qi Tian, Chang Xu," A Semi-Supervised Assessor of Neural Architectures"(2020)
[3] Ziwei Zhang, Peng Cui, Wenwu Zhu," Deep Learning on Graphs: A Survey"
[4] Esteban Real, Alok Aggarwal, Yanping Huang, Quoc V Le," Regularized Evolution for Image Classifier Architecture Search"
[5]https://github.com/google-research/nasbench
[6]https://www.cs.toronto.edu/~kriz/cifar.html