# 知識転移グラフによる複数ネットワークの共同学習に関する研究

TP18009 南蒼馬　　指導教授：山下隆義

## 1. はじめに

複数のネットワークを用いて互いの知識を転移しながら学習する知識転移手法が提案されている [1, 3]．これらの手法は，優秀なネットワークの出力を疑似ラベルとして利用したり，同一構造のネットワーク同士で出力を模倣し合うことで精度が向上する．しかし，従来の知識転移方法は人が設計しており，そのバリエーションは非常に限定的である．

本研究では，従来の知識転移手法を内包しつつ，新しい学習方法も含むようなグラフ表現を提案する．このグラフ表現と 4 種類のゲート関数を用いて損失値を制御することで，多様な知識転移が可能な共同学習を実現する．また，ハイパーパラメータ探索により知識転移グラフ構造を最適化する．

## 2. 知識転移手法

知識転移には一方向の学習手法である knowledge distillation (KD) [1] と，双方向の学習手法である deep mutual learning (DML) [3] がある．KD はネットワークを学習させる際に，通常の教師ラベルに加えて事前学習済みネットワークの出力を疑似的な教師ラベルとして利用する手法である．一方，DML は未学習のネットワーク同士のみで相互に学習を行う手法である．共同で学習するネットワークの数を増やしていくことで，KD よりも精度を向上させることができる．しかし，DML はネットワークの種類や使用する損失関数が同一であるため，ネットワーク間で伝達される情報に多様性が生まれにくい．

## 3. 提案手法

従来手法を拡張し，より多様性のある学習方法を実現するために，ネットワーク間の知識転移を表すグラフ表現を提案する．各ネットワークをノードで表現し，知識の伝達方向をエッジで表現する．各エッジ上に異なる損失関数を定義することで，様々な学習方法を表現することができる．本研究では，損失関数に 4 種類のゲート関数を組み込むことで，多様な学習方法を実現する．

### 3.1 知識転移グラフ表現

ノード数 $M$ を 3 としたときの知識転移グラフ表現を図 1 に示す．$i$ 番目のモデルをノード $m_i$ とし，各ノード間に双方向のエッジを定義する．このエッジは知識が伝わる方向を表す．知識転移元のノードを source ノード，転移先のノードを destination ノードと呼ぶ．これら 2 つのノードの出力から損失を計算し，destination ノードへのみ誤差を逆伝播する．

### 3.2 損失関数

$n$ サンプル目の入力画像 $\boldsymbol{x}_n$ とその教師ラベル $\hat{y}_n$ で構成されるミニバッチを $\mathcal{B} = \{\boldsymbol{x}_n, \hat{y}_n\}_{n=1}^{N}$ で表し，ミニバッチ $\mathcal{B}$ のバッチサイズを $|\mathcal{B}|$ で表す．教師ラベル $\hat{y}_n$ は正解クラスの ID である．学習に使用するモデルの数を $M$，source ノードを $m_s$，destination ノードを $m_t$ とする．

ノード間の出力確率の差異を求める際には，Kullback-Leibler (KL) divergence $KL(\boldsymbol{p}_s(\boldsymbol{x}_n)||\boldsymbol{p}_t(\boldsymbol{x}_n))$ を使用する．ここで，$\boldsymbol{p}_s, \boldsymbol{p}_t$ はそれぞれ，source ノードと destination ノードの応答値であり，softmax 関数によって正規化された確率分布である．また，教師ラベルを 0 番目のノード $m_0$ とし，$\hat{y}_n$ の one-hot ベクトル表現を $\boldsymbol{p}_0(\boldsymbol{x}_n)$ として表す．

source ノード $m_s$ から destination ノード $m_t$ へ知識を伝える際に使用する損失関数 $L_{s,t}$ を式 (1) に示す．

$$L_{s,t} = \sum_{n}^{|\mathcal{B}|} G_{s,t}(KL(\boldsymbol{p}_s(\boldsymbol{x}_n)||\boldsymbol{p}_t(\boldsymbol{x}_n))) \tag{1}$$

ここで，$G_{s,t}(\cdot)$ はゲート関数である．ゲート関数は，destination ノードへ誤差を逆伝播する際に，勾配情報を制御するための機構である．最終的に，destination ノード $m_t$ の損失関数は以下のように全ノードの損失を総和する式となる．

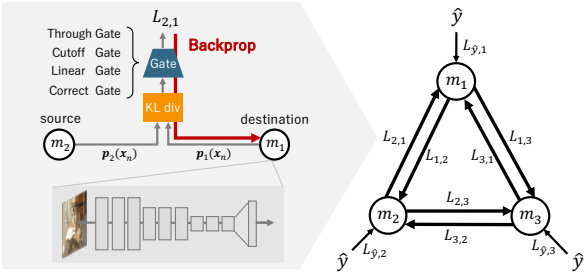



**図 1: 知識転移グラフ**（ノード数 $M = 3$ の場合）

$$L_t = \sum_{s=0, s\neq t}^{M} L_{s,t} \tag{2}$$

### 3.3 ゲート関数

パラメータ数の大きい優秀なネットワークを学習させるとき，それよりも小さなネットワークからの知識転移は学習の妨げになる場合がある．このような不要な知識転移を抑制するためゲート関数を導入する．ゲート関数として Through Gate，Cutoff Gate，Linear Gate，Correct Gate の 4 種類の関数を定義する．Through Gate は入力されたサンプルごとの損失をそのまま通すゲートである．

$$G_{s,t}^{Through}(a) = a \tag{3}$$

Cutoff Gate は損失計算を行わないゲートである．Cutoff Gate を用いることで，知識転移グラフのうち，任意のエッジを切断することができる．この関数は，一方向の知識転移を行う場合（KD など）に必要となる．

$$G_{s,t}^{Cutoff}(a) = 0 \tag{4}$$

Linear Gate は学習時間によって損失の重みを線形に変化させるゲートである．学習初期には重みを小さくし，学習が進むにつれて重みを大きくする．

$$G_{s,t}^{Linear}(a) = \frac{c}{c_{end}} \cdot a \tag{5}$$

ここで，$c$ は累積更新回数であり，$c_{end}$ は学習終了時における更新回数である．

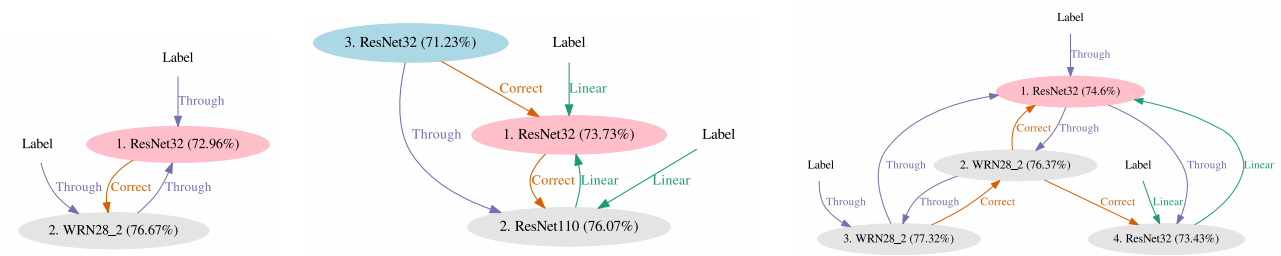
Correct Gate は，source ノードが正解したサンプルの損失のみを通す．source ノード $m_s$ が出力する確率分布 $p_s$ の Top-1 クラス番号を $y_s$ とすると以下の式で表される．$\delta$ はデルタ関数である．

$$G_{s,t}^{Correct}(a; \hat{y}, y_s) = \delta_{\hat{y},y_s} \cdot a \tag{6}$$

source ノードが事前学習済みモデルでない場合，学習初期に誤った情報伝播を抑制できる．Linear Gate が損失全体に重みをかけるのに対し，Correct Gate は損失を計算するサンプルを選別する．

### 3.4 更新方法

はじめに，全てのモデルの重みを乱数で初期化する．次に，全てのモデルに同じサンプルを入力し，損失を求める．得られた損失から勾配を求め，全てのモデルを同時に更新する．式 (2) から求めた損失 $L_t$ の勾配は，ノード $m_t$ にのみ逆伝播され，ノード $m_s$ には影響を与えない．




**(a) ノード数 2 (72.67%)** **(b) ノード数 3 (73.73%)** **(c) ノード数 4 (74.60%)**

**図 2: 最適化後のグラフ**: 赤いノードは最適化対象ノード，青いノードは事前学習済みノード，“Label” は教師ラベルを表す．紫のエッジは Through Gate，緑は Linear Gate，オレンジは Correct Gate を表す．Cutoff Gate は省略されている．カッコ内の数値は，5 回試行したうちのある 1 回の精度である．

**表 1: CIFAR100 における最適化結果**：“Fixed” はグラフ中の全てのゲートを Through Gate とした場合である．

| ノード数 | ゲート | 精度 (ノード 1) [%] |
|---|---|---|
| 1 | - | 70.71 ± 0.39 |
| 2 | Fixed (Through Gate) | 72.47 ± 0.78 |
| | Optimized | **72.88** ± 0.41 |
| 3 | Fixed (Through Gate) | 71.88 ± 0.43 |
| | Optimized | **73.46** ± 0.42 |
| 4 | Fixed (Through Gate) | 73.40 ± 0.39 |
| | Optimized | **74.34** ± 0.32 |
| 5 | Fixed (Through Gate) | 73.40 ± 0.28 |
| | Optimized | **74.54** ± 0.59 |

### 3.5 最適化方法

知識転移グラフの最適化を行うため，ハイパーパラメータ探索を行う．$m_1$ を最適化対象ノード，それ以外を補助ノードと呼ぶ．最適化対象ノードの精度が最大化するように，補助ノードのネットワークの種類やエッジ上に定義されているゲートの種類を選択する．最適化手法として Asynchronous Successive Halving Algorithm (ASHA) [2] を用いる．ASHA は，並列分散環境でハイパーパラメータのランダムサーチを行う手法である．まず，ASHA によって提案された $D$ 個の知識転移グラフを，$D$ 個の GPU サーバ上で非同期に分散学習する．合計で $T$ 個の知識転移グラフを探索するまで，この学習を繰り返す．本研究では，$D = 30, T = 1500$ の条件で最適化を行った．ただし，探索時間を短縮するため，見込のない知識転移グラフについては学習中に早期終了を行う．各知識転移グラフは，$1, 2, 4, \cdots, 2^k$ エポック目で，検証データを用いて最適化対象ノードの精度を評価する．この精度が，過去に評価された全ての精度のうち下位 50%に入っていれば，そのグラフの学習を早期終了し，新たなグラフを生成し学習する．

## 4. 実験

ASHA により最適化した知識転移グラフによる効果を検証する．

### 4.1 実験概要

データセットは CIFAR100 を用いる．データセットのうち 50000 枚を訓練データ，10000 枚を検証データとする．バッチサイズは 64，エポック数は 200 である．モデルには，ResNet32，ResNet110，Wide ResNet 28-2 の 3 種類を用いる．評価対象モデルは ResNet32 である．報告する検証データの精度は，最適化されたグラフが得られた後，グラフ構造を固定し 5 回試行したときの平均値とする．

### 4.2 実験結果

共同学習するノード数を 2 から 5 とした場合の最適化結果を表 1 に示す．提案手法はノード数が 2 のとき 72.88%，3 のとき 73.46%，4 のとき 74.34%，5 のとき 74.54%を達成した．一方，ゲートが全て Through Gate の場合，精度が低下している．これにより，ゲートによって伝達される情報を制御することが，ネットワークの学習に効果的であることが分かる．

**表 2: 従来手法との比較**：*は事前学習済みモデルを表す．$T$ は温度パラメータである．

| 手法 | 精度 (ノード 1) [%] | 補助ノード |
|---|---|---|
| KD ($T = 2$) | 71.43 ± 0.43 | ResNet110* |
| DML | 71.49 ± 0.24 | ResNet110 |
| DML | 72.09 ± 0.43 | ResNet32, ResNet32 |
| DML | 72.20 ± 0.47 | ResNet32, ResNet110 |
| Ours | **73.46** ± 0.42 | ResNet32*, ResNet110 |

図 2 に，最適化して得られた知識転移グラフを示す．ノード数 2 のグラフは，DML に似たグラフであり，小さなネットワークから大きなネットワークへのエッジのみが Correct Gate になっている．ノード数 3 のグラフは，KD と DML が融合したグラフであることがわかる．Linear Gate により，初めは KD に類似した学習が行われ，学習が進むにつれて KD と DML が組み合わさった学習が行われる．ノード数 4 のグラフは，ResNet32 と Wide ResNet28-2 が 2 つずつの組合せで学習を行っている．ノード 2 の Wide ResNet28-2 には教師ラベルからの誤差情報が伝わっておらずノード 1 と 3 の出力のみから学習を行っている．ノード 2 はノード 3 から伝達される情報の中継役として存在している．

従来手法との比較を表 2 に示す．最適化された知識転移グラフによって学習したモデルは，KD や DML を上回る精度を達成した．提案する共同学習手法は，知識転移グラフの最適化により，KD や DML を内包する新たな学習法を獲得していると考えられる．

## 5. おわりに

本研究では，ネットワーク間の知識転移を表現したグラフによって学習する共同学習手法を提案した．また，ハイパーパラメータ探索によって最適な知識転移グラフを探索できる手段を実現した．最適化されたグラフは，従来手法を超える精度を達成した．また，各エッジにゲートを導入し，ネットワーク間で伝達される情報を制御することで，ネットワークが効果的に学習することを確認した．今後は，アンサンブルモデルの導入や中間層からの知識転移について研究を行う．

## 参考文献

[1] G. Hinton, *et al.*, “Distilling the knowledge in a neural network”, NeurIPS workshop, 2015.

[2] L. Li, *et al.*, “Massively Parallel Hyperparameter Tuning”, arXiv preprint arXiv:1810.05934, 2018.

[3] Y. Zhang, *et al.*, “Deep mutual learning”, CVPR, 2018.

## 研究業績

[1] S.Minami, *et al.*, “Gradual Sampling Gate for Bidirectional Knowledge Distillation”, MVA, 2019.

[2] 南蒼馬 等, “複数ネットワークの共同学習における知識転移グラフの自動最適化”, 第 22 回 画像の認識・理解シンポジウム, 2019.