進捗報告

表 1: 実験の設定

model	VGG11
Optim(model)	SGD(lr=0.01, momentum=0.9)
Optim(α)	Adam(lr=0.005, β=(0.5, 0.999))
Loss	Cross Entropy Loss
dataset	cifar10
batch size	64
train data	25000 + 25000
epoch	50

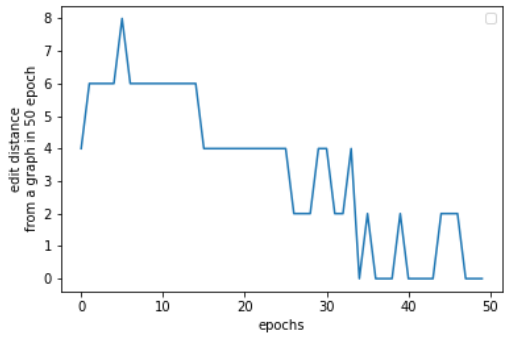

図 1: 各学習時点のグラフと終了後のグラフとの編集距離

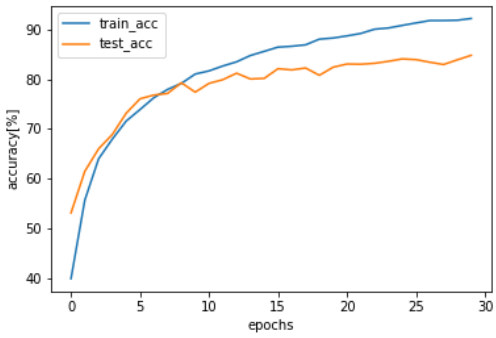

図 2: 探索時の Accuracy

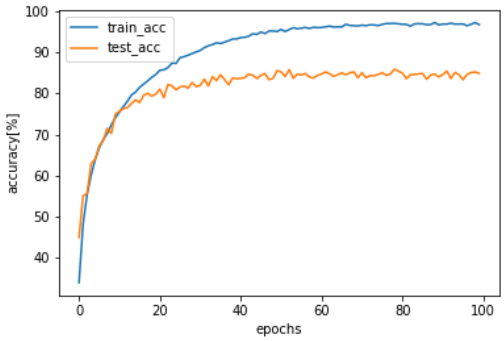

図 3: 評価時の Accuracy

1　今週やったこと

2　実験 1

グラフの学習収束性を見るために, 1 epoch ごとにグラフを取り出してその編集距離を調べた. グラフの比較対象は, 学習終了時点 (50 epoch) のグラフとした. 表 1 に実験設定を示した.

2.1　結果

図 1 に編集距離の変化を示した. 全体的な減少傾向から, グラフの収束に向かう様子が確認できた. 完全な収束判定は分からないが, 50 epoch 付近でも変化があるためさらに学習時間を延ばして確認する必要がある.

3　実験 2

探索したショートカット位置の性能を評価するため, アーキテクチャを探索する (a) 探索段階, 探索したグラフ構造で学習する (b) 評価段階, 素の VGG による (c) ベースライン, の 3 つの精度を比較した. 設定は表 1 を引き継ぐ. 実験は各 1 回, seed 41 で行った.

3.1　結果

図 2, 3, 4 に各訓練とテスト精度を, 表 2 には最大テスト精度を示した. 評価時の精度は, 探索時より改善したものの, ベースラインよりも 0.32 %低くなった.

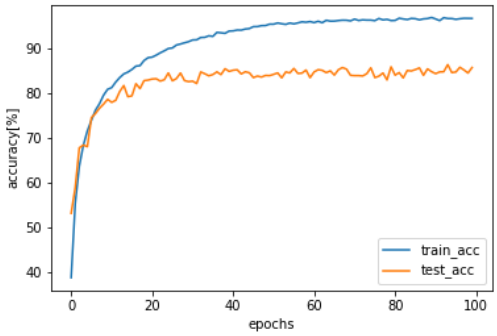

図 4: ベースライン (VGG11) の Accuracy

表 2: Accuracy の比較

	accuracy(%)	学習時間 (epoch)
探索	84.78	30
評価	85.94	100
ベースライン	86.26	100

3.2 考察

今回の設定の場合, ショートカットを入れることで精度が低くなった. 原因として, ショートカット関数の問題と層の浅さが考えられる.

今回用いたショートカット関数は, 1x1 Conv(stride¿=1) であり, 大きさの違う出力に対応するため stride で無理やり補正している. stride が大きいほど, ショートカットによって情報が欠落することが精度に悪影響を与えていると思われる. 次回は情報が失われないショートカットに替える必要がある.

また層が浅いため, pooling 層によって頻繁に大きさが変えられるため, ショートカットが行いにくく効果が薄い可能性がある. VGG19 などより深いモデルの場合よりショートカットの効果が表れると推測される.

4 今後の予定

- ショートカット関数の改善
- VGG19 への移植

5 ソースコード

github の notebook リポジトリ参照