進捗報告

1　今週やったこと

- 入れ替えコードの実装と実験少し

2　問題設定

データを正解と不正解に分割して学習する以下のアルゴリズム 1 を考える.

Algorithm 1 Swap two datasets

1. データ 2N 抜き出す
2. そのデータを ランダムに N（A） と N（B）に分ける.
3. X : A → train, B → test で学習
4. Y : B → train, A → test で学習
5. 3. と 4. で test でミスしたデータを調べる
6. A をテストとしたとき失敗したデータと B をテストとしたとき成功したデータを入れ替える. ただし, 同じクラスのデータ同士.
7. モデルを初期化して, 3. に戻る.

前回とは異なり, データのクラスが同じもの同士を入れ替えることで, データセットのクラス比が変化しないようにした. 学習中に入れ替えるのではなく, 学習が終わった後入れ替え, 再び学習する際にモデルのパラメータは初期化した.

3　実験

実験設定は表 1 とした. 30 epoch 学習した後, データを入れ替える手順を 10 回行った. はじめは 50 epoch に設定したが, 長時間の実験をすると Google Colab がいつの間にか中断したため, 30 epoch と短めにした.

表 1: 実験の設定

dataset	cifar10
data N	5,000 / model
task	10 クラス識別
input	image(3x32x32)
output	class(10)
model	CNN(16 層)
optim	SDG (lr=0.001, moment=0.9)
loss	Cross Entropy Loss
batch size	16
epoch	30
swap	10

3.1　結果

テストの精度は図 1 に示した. 横軸はデータを入れ替えた回数で, 今回は 10 回である. 1 回目の精度が 40 〜 45 [%] でこれは入れ替えなかった場合の指標である. 学習 epoch 数が少なかったために精度が高まらない結果となった. その後入れ替えを行うたびに精度は減少し, 共に 20 %程度となった. 条件が異なるので単純に比較できないが, 前回まで手順 3 が手順 4 より大幅に低い精度であったため, 予想を裏切る結果であった. 入れ替えをランダムに行った場合は精度がほとんど変わっていないため, 入れ替え操作が精度に影響していると言える.

図 2 に学習がまだ進んでいない初期の 0 epoch 目のテストの結果を示した. ベースラインは 10 %であるため, 交換が重ねられるごとにデータセットの難しさが高まっていると思われる. また手順 3 と手順 4 を比較すると, 不正解データが集められた手順 4 がより難しいことが分かる.

入れ替えられたデータの数は, 図 3 のようにラベルのクラスを区別して示した. データサンプル数は 5000 なので, 1 回目の入れ替えでは 2000 個の半数近くが交換されていた. その後も 1000 個程度が常に入れ替えられている. クラスの比率はあまり偏りが見られず, ほぼ同数であった.

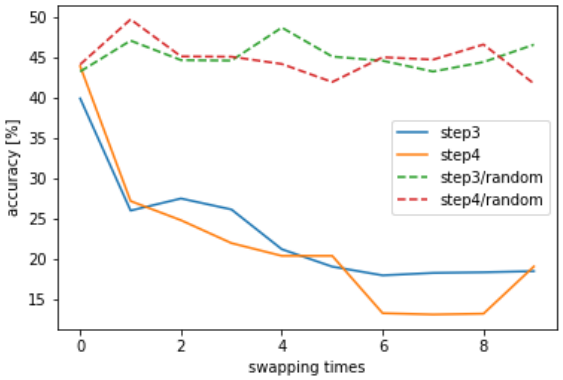

図 1: 学習後モデルのテストの精度　実線がテスト結果で入れ替えた場合, 点線がランダムに入れ替えた場合

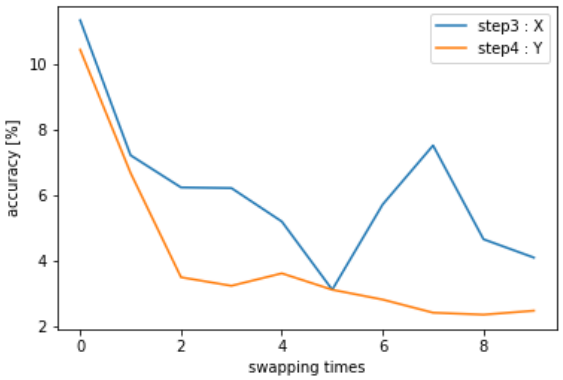

図 2: 0epoch 目のテストの精度　(ベースライン:10 %)
　実線の青が手順 3, 橙が手順 4

4　考察

　図 1 の精度は両方とも減少した．訓練セットとテストセットの偏りが大きくなるにつれて, 学習してないテスト画像を間違えたと思われる．パラメータは初期化しているため, モデルが学習した画像にテストセットは含まれていない．従って精度が増加する要因はないと考えられる．

　データの入れ替え数ではラベル比に偏りが見られなかったため, 交換されるデータの中で交換頻度の高い画像などを見たい．

　パラメータのチューニングに時間がかかったので, optuna が適用できるかを調べたい．

5　今後の予定

- 入れ替え頻度の高いデータの分析

6　ソースコード

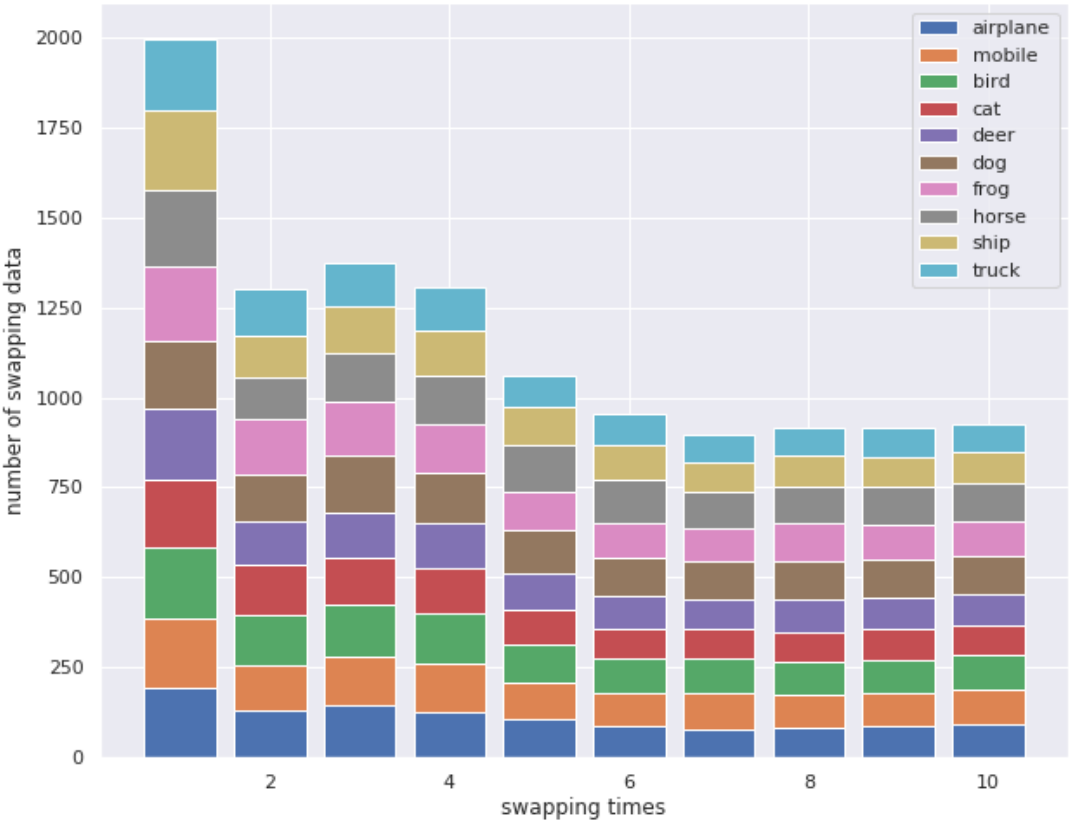

図 3: 入れ替えたデータのクラスラベル