進捗報告

1　今週やったこと

- cnn モデルの改良
- 入れ替えデータの解析

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 N（A） と N（B）に分ける.
3. A → train, B → test で実験
4. B → train, A → test で実験
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.
7. 3. に戻る.

アルゴリズム 1 の手順 6 の操作によって, データセット A には成功データ, B には失敗データが集められる. 従って簡単なデータを学習し, 難しいデータでテストされる手順 3 の精度は低くなると予想される.

アルゴリズム 1 に示した手順の設定として, 手順 3. 4. で 2 つのモデルをそれぞれ独立させた. またデータポイントの入れ替えは, 入れ替え可能なデータを先頭から全て交換することにした.

3　実験

予備実験によって, ベースラインを測りパラメータを調整した. 最適化関数を Adam にした場合やバッチサイズを増やした場合などを試したが精度が下がる結果となった. 実験では改良した CNN と学習を 50epoch まで増やした条件で, 学習を行い, 入れ替えデータを解析した.

表 1: 実験の設定

dataset	cifar10
data N	25,000 / 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	50

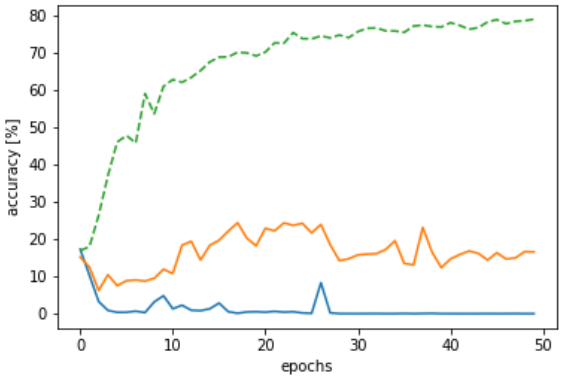

図 1: テストの精度 (横軸:epoch, 縦軸:精度 [%]) 実線の青が手順 3, 橙が手順 4, 点線がベースライン

図 1 にテストの精度を示した. ベースラインの学習は期待通り, 80 [%] の前回より高い精度になった. 一方入れ替えを行った場合, 両方の場合で学習が進まなかった.

図 2 の入れ替えたデータの数では, 初期に 4000 枚以上の入れ替えが起こっていた. 中期にも 2000 枚の入れ替えが起こっているが, いづれも図 1 の精度が落ちる時期と重なっている.

具体的なデータの入れ替えとして図 3 にデータセット A の画像と図 4 にその入れ替えの過程を示した. 分

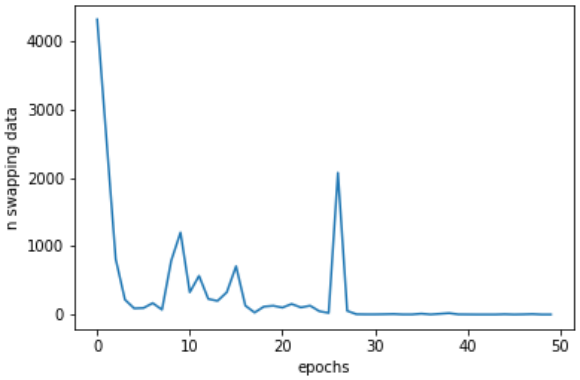

図 2: データの入れ替え数 (横軸:epoch, 縦軸:データ数)

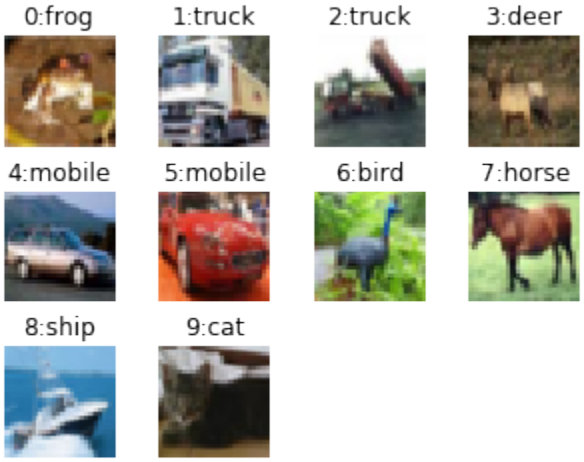

図 3: データセットAの例, 先頭 10 件 (index:label)

類を間違えてデータセットBに移動した後, 学習が進んでAに戻っているものもあった. 戻ったデータは, 自動車, トラック, 船などの機械で, この例では動物は学習できていない傾向にあった.

4　考察：現在の問題点

データの入れ替えを分析すると, 機械クラスが簡単で, 動物クラスが難しいという例が見られた.

しかし入れ替えを行ったときの精度が著しく落ちる原因がまだ分かっていない.

- 最適化関数が働いていない
- 表現力が高すぎて過学習している

などが考えられる.

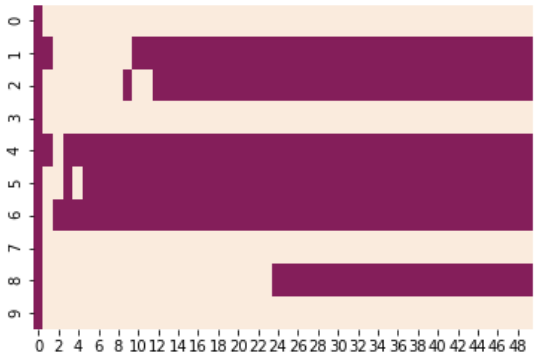

図 4: データの入れ替え例 (横軸:epoch, 縦軸:index) データセットAの先頭 10 件のデータの入れ替え履歴. 赤がA, 白がBに属していることを示す

5　今後の予定

- ソースコードに不備がないか確認
- データの分類難易度の大域的な確認

6　ソースコード

CNN の Pytorch での実装をコード 1 に示す.

Listing 1: cnn

```
class CNN(nn.Module):
  def __init__(self):
    super(CNN, self).__init__()
    self.model = nn.Sequential(
      nn.Conv2d(3, 64, 3, padding=1),
      nn.ReLU(),
      nn.Conv2d(64, 64, 3, padding=1),
      nn.ReLU(),
      nn.BatchNorm2d(64),
      nn.Conv2d(64, 64, 3, padding=1),
      nn.ReLU(),
      nn.MaxPool2d(2),
      nn.Dropout(0.25),

      nn.Conv2d(64, 128, 3, padding=1),
      nn.ReLU(),
      nn.Conv2d(128, 128, 3, padding=1),
      nn.ReLU(),
      nn.BatchNorm2d(128),
      nn.Conv2d(128, 128, 3, padding=1),
      nn.ReLU(),
      nn.MaxPool2d(2),
```

```
      nn.Dropout(0.25),

      nn.Conv2d(128, 256, 3, padding=1),
      nn.ReLU(),
      nn.Conv2d(256, 256, 3, padding=1),
      nn.ReLU(),
      nn.BatchNorm2d(256),
      nn.Conv2d(256, 256, 3, padding=1),
      nn.ReLU(),
      nn.Conv2d(256, 256, 3, padding=1),
      nn.ReLU(),
      nn.Conv2d(256, 256, 3, padding=1),
      nn.ReLU(),
      nn.BatchNorm2d(256),
      nn.Conv2d(256, 512, 3, padding=1),
      nn.ReLU(),
      nn.Conv2d(512, 512, 3, padding=1),
      nn.ReLU(),
      nn.AvgPool2d(8)
    )
    self.pc = nn.Sequential(
      nn.Linear(512, 1024),
      nn.Dropout(0.5),
      nn.Linear(1024, 1024),
      nn.Dropout(0.5),
      nn.Linear(1024, 10),
      nn.Softmax()
    )

  def forward(self, x):
    x = self.model(x)
    x = x.view(-1, 512)
    x = self.pc(x)
    return x
```