進捗報告

1　今週行ったこと

Googlecolab を用いて、Cifar10 の CNN の以下の実装を行った。

1. データを 10000 個取り出す
2. データを A と B にランダムに二分割する
3. A→train,B→test で実験（A→train,B→test としたとき AtoB と表記する）
4. B→train,A→test で実験（B→train,A→test としたとき BtoA と表記する）
5. 3. と 4. でミスしたデータを調べた
6. A をテストした時、失敗したデータ　B をテストした時、成功したデータをランダムに一つ交換
7. 3. に戻る

以上の操作を 50 回繰り返し行った。

まず、比較のために交換を行わないで作業を行った結果を図 1 に示す。次に、ランダムに一つだけ交換していくものを図 2 に示す。最後に、ランダムに 100 個交換していくものを図 3 に示す。

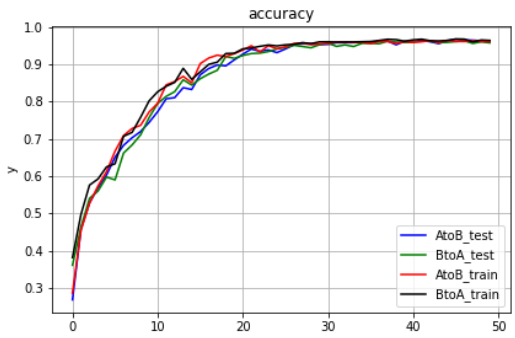

図 1: 交換なし

全て横軸が繰り返し回数、縦軸が Accuracy である。

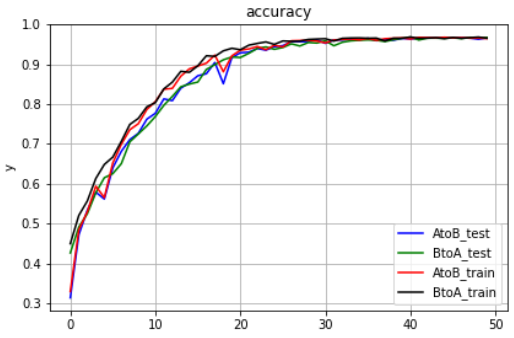

図 2: 1 個交換

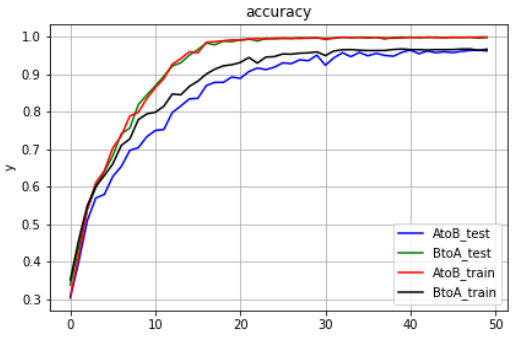

図 3: 100 個交換

繰り返し回数が 50 回しかないので、1 個交換のものと交換なしのものではあまり差異は見られず、繰り返し回数が 50 付近では、Accuracy は 96，97 %ほどであった。

100 個交換のものは以下の制約の下で行った。A をテストしたときの miss した個数を A_miss とすると、100 個交換（A_miss≥100）,10 個交換（A_miss≥10）,1 個交換（A_miss<10）を行う。下記のような結果になり、A（正解ラベルを集めたもの）が 99.8 %を超える一方、B（不正解ラベルを集めたもの）は 96, 97 %どまりという結果になった。

モデルのコンパイルは下記のように行い、
model_AtoB.compile(loss='categorical_crossentropy', optimizer=Adam(lr=0.001), metrics=['accuracy'])

モデルの訓練は下記である。
model_AtoB.fit(x_a,y_a, batch_size=32, epochs=1, verbose=1, validation_split=0.1)
また、モデルの構築は下記である。
model=Sequential()
1 層目
model.add(Conv2D(32,kernel_size=(3,3),activation='relu',padding='same',input_shape=(32,32,3)))
model.add(Conv2D(32,kernel_size=(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
2 層目
model.add(Conv2D(32,kernel_size=(3,3),activation='relu',padding='same'))
model.add(Conv2D(32,kernel_size=(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
出力層
model.add(Flatten()) model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5)) model.add(Dense(10))
model.add(Activation('softmax'))
model_AtoB=model model_BtoA=model

2 考察

正解を増やした方は正解率が高まっている一方で間違いを増やした方の正解率が特に下がっていないのは、交換することによって、実質訓練枚数が上がっているからだと思われた。交換のことを考えて、一回のエポック数を 1 としたが、訓練がある程度できるまで先に訓練をしてから交換を行った方がよさそうであり、また、一回における交換枚数も 1 枚ー 10 枚程度で実験がうまく進むと思われる。(96 %まで、Accuracy が高まると miss した枚数も 200 枚程度になるため)

3 次回行うこと

random のシードを変えて、他の乱数値でも同じような現象になるか確かめる。

- 学習を進めてから交換を行うもの
- 初めから交換を行うもの
- 交換を行わないもの

上記の 3 つに対して random のシードを変えてそれぞれ平均をとり、accuracy の推移を確かめる。

一回の交換で 100 個交換したときのソースコードを，ソースコード 1 に示す．

Listing 1: Change100

```
!pip install dlt

#dlt=Deep Learning Tools
import dlt
data=dlt.cifar.load_cifar10()
from keras.utils.np_utils import to_categorical
import numpy as np
from sklearn.model_selection import train_test_splitデータの取得確認
#,
x=data.train_images[:10000]
x=x.astype('float32')/255.0
print(x.shape)
y=data.train_labels[:10000]
y=to_categorical(y,10)
print(y.shape)
print(x[0])
print(y[0])
#とに二分割 AB
x_a,x_b=train_test_split(x,test_size=0.5,random_state=0)
print(x_a.shape)
print(x_b.shape)
y_a,y_b=train_test_split(y,test_size=0.5,random_state=0)
print(y_a.shape)
print(y_b.shape)

from keras.models import Sequential
from keras.layers import Dense,Dropout,Activation,Conv2D,MaxPooling2D,Flatten
from keras.optimizers import Adamモデルの構築
#
model=Sequential()

# 層目 1
model.add(Conv2D(32,kernel_size=(3,3),activation='relu',padding='same',
    input_shape=(32,32,3)))
model.add(Conv2D(32,kernel_size=(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

# 層目 2
model.add(Conv2D(32,kernel_size=(3,3),activation='relu',padding='same'))
model.add(Conv2D(32,kernel_size=(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

# 出力層
model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(10))
```

```
model.add(Activation('softmax'))

model_AtoB=model
model_BtoA=model

#A->train,B->test
model_AtoB.compile(
loss='categorical_crossentropy',
optimizer=Adam(lr=0.001),
metrics=['accuracy'])

#B->train,A->test
model_BtoA.compile(
loss='categorical_crossentropy',
optimizer=Adam(lr=0.001),
metrics=['accuracy'])

import random一連の流れの繰り返し

#
repetition=50
AtoB_train_loss=[]
AtoB_train_acc=[]
AtoB_test_loss=[]
AtoB_test_acc=[]
BtoA_train_loss=[]
BtoA_train_acc=[]
BtoA_test_loss=[]
BtoA_test_acc=[]
for i in range(repetition):
#A->train,B->test
model_AtoB.fit(x_a,y_a,
batch_size=32,
epochs=1,
verbose=1,
validation_split=0.1
)
AtoB_train_loss.append(model_AtoB.evaluate(x_a,y_a)[0])
AtoB_train_acc.append(model_AtoB.evaluate(x_a,y_a)[1])
AtoB_test_loss.append(model_AtoB.evaluate(x_b,y_b)[0])
AtoB_test_acc.append(model_AtoB.evaluate(x_b,y_b)[1])
y_b_pred=model.predict_classes(x_b)
y_b_true=np.argmax(y_b,axis=1)
# の正解B
B_correct=[]
for j in range(5000):
if y_b_pred[j] == y_b_true[j]:
B_correct.append(j)


#B->train,A->test
model_BtoA.fit(x_b,y_b,
batch_size=32,
```

```
103             epochs=1,
104             verbose=1,
105             validation_split=0.1
106             )
107             BtoA_train_loss.append(model_BtoA.evaluate(x_b,y_b)[0])
108             BtoA_train_acc.append(model_BtoA.evaluate(x_b,y_b)[1])
109             BtoA_test_loss.append(model_BtoA.evaluate(x_a,y_a)[0])
110             BtoA_test_acc.append(model_BtoA.evaluate(x_a,y_a)[1])
111             y_a_pred=model.predict_classes(x_a)
112             y_a_true=np.argmax(y_a,axis=1)
113             # の間違い A
114             A_miss=[]
115             x_a_miss=[]
116             y_a_miss=[]
117             for j in range(5000):
118             if y_a_pred[j] != y_a_true[j]:
119             A_miss.append(j)
120
121             random.shuffle(A_miss)
122             random.shuffle(B_correct)
123
124             if len(A_miss) >= 100:
125             for j in range(100):
126             x_a[A_miss[j]],x_b[B_correct[j]]=x_b[B_correct[j]],x_a[A_miss[j]]
127             y_a[A_miss[j]],y_b[B_correct[j]]=y_b[B_correct[j]],y_a[A_miss[j]]
128             elif len(A_miss) >= 10:
129             for j in range(10):
130             x_a[A_miss[j]],x_b[B_correct[j]]=x_b[B_correct[j]],x_a[A_miss[j]]
131             y_a[A_miss[j]],y_b[B_correct[j]]=y_b[B_correct[j]],y_a[A_miss[j]]
132             else:
133             x_a[A_miss[0]],x_b[B_correct[0]]=x_b[B_correct[0]],x_a[A_miss[0]]
134             y_a[A_miss[0]],y_b[B_correct[0]]=y_b[B_correct[0]],y_a[A_miss[0]]
135
136
137
138             import numpy as np
139             import matplotlib.pyplot as plt
140
141             fig, ax = plt.subplots()
142
143             t = np.linspace(0, 10, 1000)
144             y1 = AtoB_test_acc
145             y2 = BtoA_test_acc
146             y3 = AtoB_train_acc
147             y4 = BtoA_train_acc
148
149             c1,c2,c3,c4 = "blue","green","red","black" # 各プロットの色
150             l1,l2,l3,l4 = "AtoB_test","BtoA_test","AtoB_train","BtoA_train" # 各ラベル
151
152             # 軸ラベル x
153             ax.set_ylabel('y') # 軸ラベル y
154             ax.set_title('accuracy') # グラフタイトル
155             # ax.set_aspect('equal') # スケールを揃える
```

```
156                 ax.grid() # 罫線
157                 #ax.set_xlim([-10, 10]) # 方向の描画範囲を指定 x
158                 #ax.set_ylim([0, 1]) # 方向の描画範囲を指定 y
159                 ax.plot(y1, color=c1, label=l1)
160                 ax.plot(y2, color=c2, label=l2)
161                 ax.plot(y3, color=c3, label=l3)
162                 ax.plot(y4, color=c4, label=l4)
163                 ax.legend(loc=0) # 凡例
164                 fig.tight_layout() # レイアウトの設定
165                 plt.savefig("/content/drive/My Drive/googlecolab/cifar10_acc_100.png") # 画像の
                        保存
166                 plt.show()
167
168                 fig, ax = plt.subplots()
169
170                 t = np.linspace(0, 10, 1000)
171                 y1 = AtoB_test_loss
172                 y2 = BtoA_test_loss
173                 y3 = AtoB_train_loss
174                 y4 = BtoA_train_loss
175
176                 c1,c2,c3,c4 = "blue","green","red","black" # 各プロットの色
177                 l1,l2,l3,l4 = "AtoB_test","BtoA_test","AtoB_train","BtoA_train" # 各ラベル
178
179                 # 軸ラベル x
180                 ax.set_ylabel('y') # 軸ラベル y
181                 ax.set_title('loss') # グラフタイトル
182                 # ax.set_aspect('equal') # スケールを揃える
183                 ax.grid() # 罫線
184                 #ax.set_xlim([-10, 10]) # 方向の描画範囲を指定 x
185                 #ax.set_ylim([0, 1]) # 方向の描画範囲を指定 y
186                 ax.plot(y1, color=c1, label=l1)
187                 ax.plot(y2, color=c2, label=l2)
188                 ax.plot(y3, color=c3, label=l3)
189                 ax.plot(y4, color=c4, label=l4)
190                 ax.legend(loc=0) # 凡例
191                 fig.tight_layout() # レイアウトの設定
192                 plt.savefig("/content/drive/My Drive/googlecolab/cifar10_loss_100.png") # 画像の
                        保存
193                 plt.show()
```