CNNモデルにおける
汎化性能の向上と計算時間の短縮

夏学期ゼミ #18 プログラミング課題5 発表

2021.6.18

B4 増橋 佳菜

1.前提と事前実験

1-1. 前提

*入力画像：64ピクセル四方(グレースケール 1channel)
*CNNの設定：
[隠れ層]
・畳込みレイヤ: **(k,n)**が3層　　※ k,nは調整変数 (例:(5,16)= kernel size:5×5, フィルター数:16)
・プーリングレイヤ: strideは2ピクセルずつ**(Average Pooling or Max Pooling)**
[全結合層]
・圧縮レイヤ: 出力数 固定, **活性化関数可変(ソフトマックス関数or シグモイド関数)**

1-2. 事前実験 (エポック数 30)

向上

基準(デフォルト)	CNN0	CNN1	CNN2	CNN3	CNN4	CNN5	CNN6	CNN7	CNN8	CNN9	CNN10	CNN11
1層目の (k,n)	(10,6)	(10,6)	(9,6)	(9,6)	(7,6)	(3,6)	(11,6)	(13,6)	(11,6)	(11,6)	(10,6)	(10,6)
2層目の (k,n)	(5,16)	(5,16)	(5,16)	(5,16)	(5,16)	(5,16)	(5,16)	(5,16)	(5,16)	(5,16)	(5,16)	(5,16)
3層目の (k,n)	(5,16)	(5,16)	(3,16)	(3,16)	(3,16)	(7,16)	(3,16)	(3,16)	(3,16)	(3,16)	(5,16)	(5,16)
プーリング方法	Ave.	Ave.	Ave.	Ave.	Ave.	Ave.	Ave.	Ave.	Ave.	Ave.	Ave.	Max.
活性化関数	Sig.	Soft.	Sig.	Sig.	Sig.	Sig.	Sig.	Sig.	Soft.	Sig.	Sig.	Sig.
Dropout	-	-	-	-	-	-	0.25	0.25	0.25	-	0.25	-
総パラメータ数	64,336	64,336	81,246	81,246	81,054	52,654	81,486	81,774	81,486	81,486	64,336	64,336
モデル構築時間(s)	609.76	608.49	561.16	557.38	489.77	-	530.24	602.34	530.69	541.96	629.04	644.49
平均計算時間(ms)※	9.48	9.46	6.91	6.86	6.04	-	6.51	7.37	6.51	6.65	9.78	10.02
正答率(%)	76.65	69.38	62.11	72.38	65.89	67.83	77.91	70.64	70.74	72.00	62.69	76.65

フィルター数は16 で十分　　向上

	CNN12	CNN13	CNN14	CNN15	CNN16	CNN17	CNN18	CNN19	CNN20	CNN21	CNN22
1層目の (k,n)	(11,6)	(11,6)	(11,6)	(11,6)	(11,6)	(13,6)	(9,6)	(11,6)	(11,6)	(11,6)	(11,6)
2層目の (k,n)	(5,16)	(7,16)	(5,16)	(5,16)	(5,32)	(5,16)	(5,16)	(7,16)	(3,16)	(5,16)	(5,16)
3層目の (k,n)	(3,16)	(3,16)	(3,16)	(3,16)	(3,32)	(3,16)	(3,16)	(3,16)	(3,16)	(5,16)	(5,16)
プーリング方法	Max.	Ave.	Ave.	Ave.	Ave.	Ave.	Ave.	Ave.	Ave.	Ave.	Ave.
活性化関数	Sig.	Sig.	Sig.	Sig.	Sig.	Sig.	Sig.	Sig.	Sig.	Sig.	Sig.
Dropout	0.25	0.25	0.20	0.15	0.20	0.20	0.20	0.20	0.20	0.20	0.40
総パラメータ数	81,486	81,486	81,486	159,950	81,774	81,246	81,246	62,670	-	79,950	64,462
モデル構築時間(s)	568.30	589.34	-	534.88	662.54	614.87	559.15	573.37	505.76	547.85	555.95
平均計算時間(ms)※	6.97	7.23	-	3.34	8.10	7.57	6.88	9.15	-	6.85	8.62
正答率(%)	75.48	73.64	78.00	73.64	73.64	74.32	68.70	71.03	72.48	68.41	74.42

kernel size:

1層目のk:
7,9,11,13で11がBest **(正解率)**

3層目のk:
小さい方が**計算時間**が短縮
→最小奇数3

+α
Dropout =0.20くらいで正解率が向上する?

2.仮説とモデルの構造

2-1. 仮説

① kernel size を (10,5,5) → (11,5,3) にすることで正答率・計算時間 ともに向上

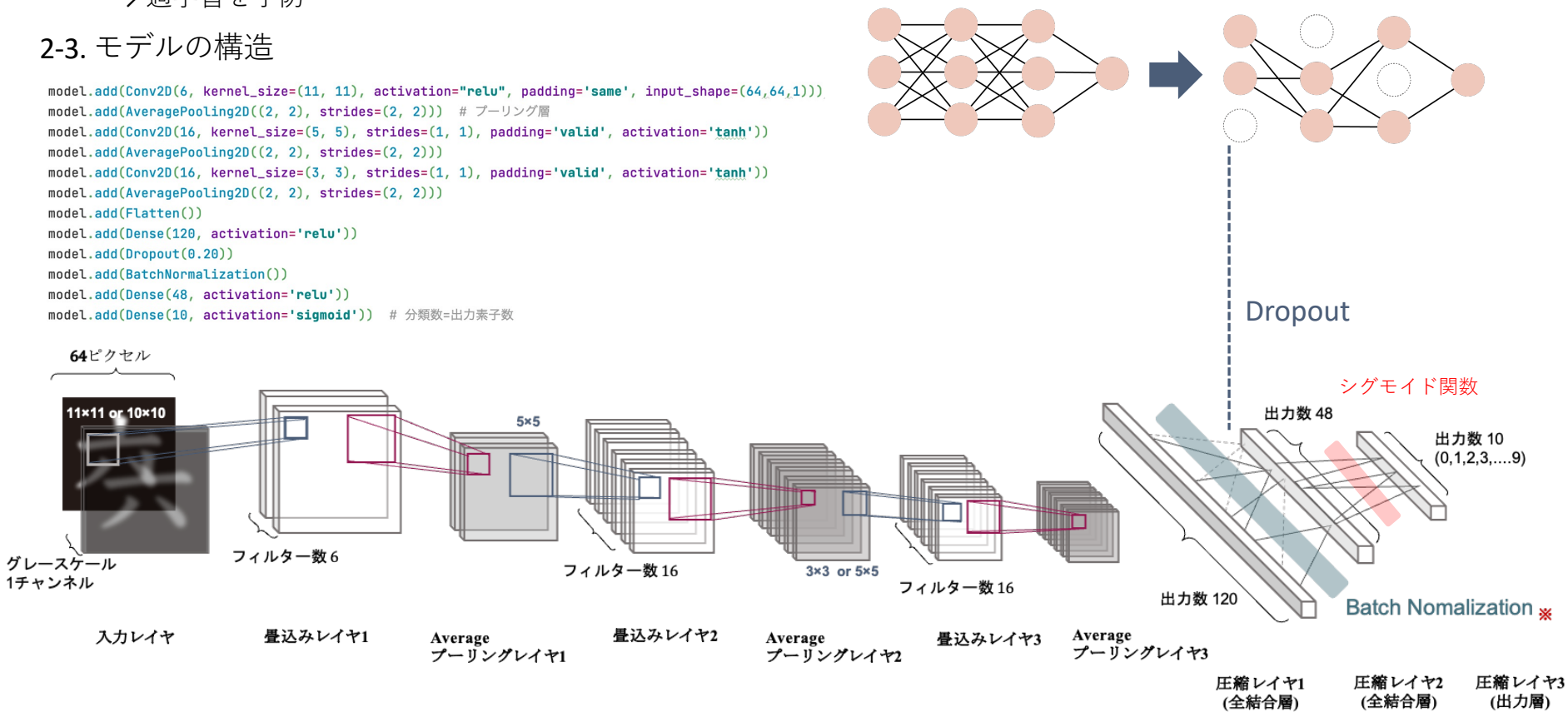
②過学習を起こす場合にはDropoutの適用で正答率が向上するのではないか

2-2. Dropoutとは

一定の確率でランダムにニューロンを無視して学習を進める=複数の独立したNNを個別に学習していることに相当する
→ 過学習を予防

2-3. モデルの構造

```
model.add(Conv2D(6, kernel_size=(11, 11), activation="relu", padding='same', input_shape=(64,64,1)))
model.add(AveragePooling2D((2, 2), strides=(2, 2)))  # プーリング層
model.add(Conv2D(16, kernel_size=(5, 5), strides=(1, 1), padding='valid', activation='tanh'))
model.add(AveragePooling2D((2, 2), strides=(2, 2)))
model.add(Conv2D(16, kernel_size=(3, 3), strides=(1, 1), padding='valid', activation='tanh'))
model.add(AveragePooling2D((2, 2), strides=(2, 2)))
model.add(Flatten())
model.add(Dense(120, activation='relu'))
model.add(Dropout(0.20))
model.add(BatchNormalization())
model.add(Dense(48, activation='relu'))
model.add(Dense(10, activation='sigmoid'))  # 分類数=出力素子数
```



3.学習結果の比較

3. 学習結果の比較

	CNN0	CNN3	CNN2	CNN1
1層目の (k,n)	(10,6)	(10,6)	(11,6)	(11,6)
2層目の (k,n)	(5,16)	(5,16)	(5,16)	(5,16)
3層目の (k,n)	(5,16)	(5,16)	(3,16)	(3,16)
Dropout	-	0.20	-	0.20
総パラメータ数	64,336	64,336	81,486	81,486
モデル構築時間(s)	1035.86	1043.88	914.91	904.87
平均計算時間(ms)※	16.10	16.23	11.23	11.10
正答率(%)	78.49	78.59	78.88	78.29

※ ともにMax Pooling
※ 活性化関数はともにシグモイド関数

① : kernel sizeの変更→計算時間 減少

kernel size(10,5,5): 計算時間増加, **正解率向上**

kernel size(11,5,3): **計算時間減少**, 正解率減少

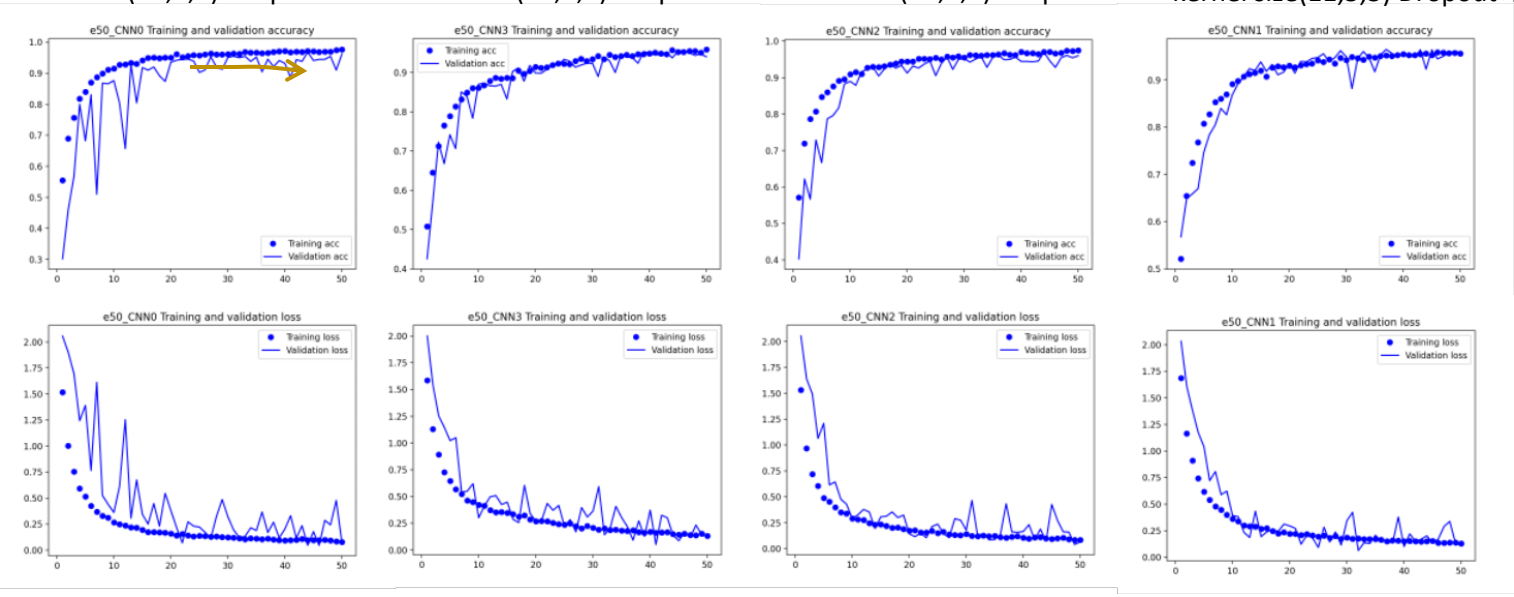
②: 1層目の畳込み層のフィルタサイズが小さい方が局所的な特徴抽出

→ 過学習のリスクが高い

→ Dropoutによる過学習予防が有効だった???

cf. CNN2→CNN1は単純にNNが減少して頭が悪くなっただけ?

kernel size(10,5,5) Dropoutなし kernel size(10,5,5) Dropoutあり kernel size(11,5,3) Dropoutなし kernel size(11,5,3) Dropoutあり



参考資料：

Dropoutとは?の参考：

「【ニューラルネットワーク】Dropout(ドロップアウト)についてまとめる - Qiita」https://qiita.com/shu_marubo/items/70b20c3a6c172aaeb8de(2021年6月17日参照)

「【Deep Learning】過学習とDropoutについて - sonickun.log」https://sonickun.hatenablog.com/entry/2016/07/18/191656(2021年6月17日参照)

「Dropoutによる過学習の抑制. Kerasを使用して機械学習を行い、実アプリケーションに適用する場合、いかに過学… | by Kazuki Kyakuno | axinc | Medium」

https://medium.com/axinc/dropoutによる過学習の抑制-be5b9bba7e89(2021年6月17日参照)

「Dropout：ディープラーニングの火付け役、単純な方法で過学習を防ぐ - DeepAge」

https://deepage.net/deep_learning/2016/10/17/deeplearning_dropout.html(2021年6月17日参照)

Dropout割合の参考：

「kerasのConv2D（2次元畳み込み層）について調べてみた - Qiita」https://qiita.com/kenichiro-yamato/items/60affeb7ca9f67c87a17 (2021年6月17日参照)

Dropoutのコード/効果の参考：

「【Kerasの使い方解説】Dropout：Conv2D（CNN）の意味・用法 | 子供プログラマー」

https://child-programmer.com/ai/keras/dropout/ (2021年6月17日参照)

※ BatchNomalizationはDropoutと同じ層で用いてはだめらしい！：

「[1801.05134] Understanding the Disharmony between Dropout and Batch Normalization by Variance Shift」Xiang Li, Shuo Chen, Xiaolin Hu, Jian Yang

https://arxiv.org/abs/1801.05134