レポートについて

コード

- CNN_CIFAR-10_rep.ipynb
  - CNNによるCIFAR-10の分類(クラス数とデータ数可変Ver.)
  - CNNでCIFAR-10を分類するコード
- MLP_CIFAR-10_rep.ipynb
  - MLPによるCIFAR-10の分類(クラス数とデータ数可変Ver.)
  - MNISTでCIFAR-10を分類するコード

チェックリスト

レポートチェックリスト.xlsx

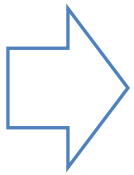
レポート用コード:
CNNによるCIFAR-10の分類(クラス数とデータ数可変Ver.)

CNN_CIFAR-10.ipynbと同様にCNNによりCIFAR-10の分類を行うコード

変更点：
CIFAR-10データセットから，任意のクラスを選ぶことができる.
さらに，各クラスの学習データ数を調整することができる.

CIFAR-10の各クラスのデータ数の内訳

	class	count_train	count_test
0	0:airplane	5000	1000
1	1:automobile	5000	1000
2	2:bird	5000	1000
3	3:cat	5000	1000
4	4:deer	5000	1000
5	5:dog	5000	1000
6	6:frog	5000	1000
7	7:horse	5000	1000
8	8:ship	5000	1000
9	9:truck	5000	1000



抜粋したデータの内訳

	class	count_train	count_test
0	0:airplane	2500	1000
1	2:bird	2500	1000
2	6:frog	5000	1000

データの抜粋の方法

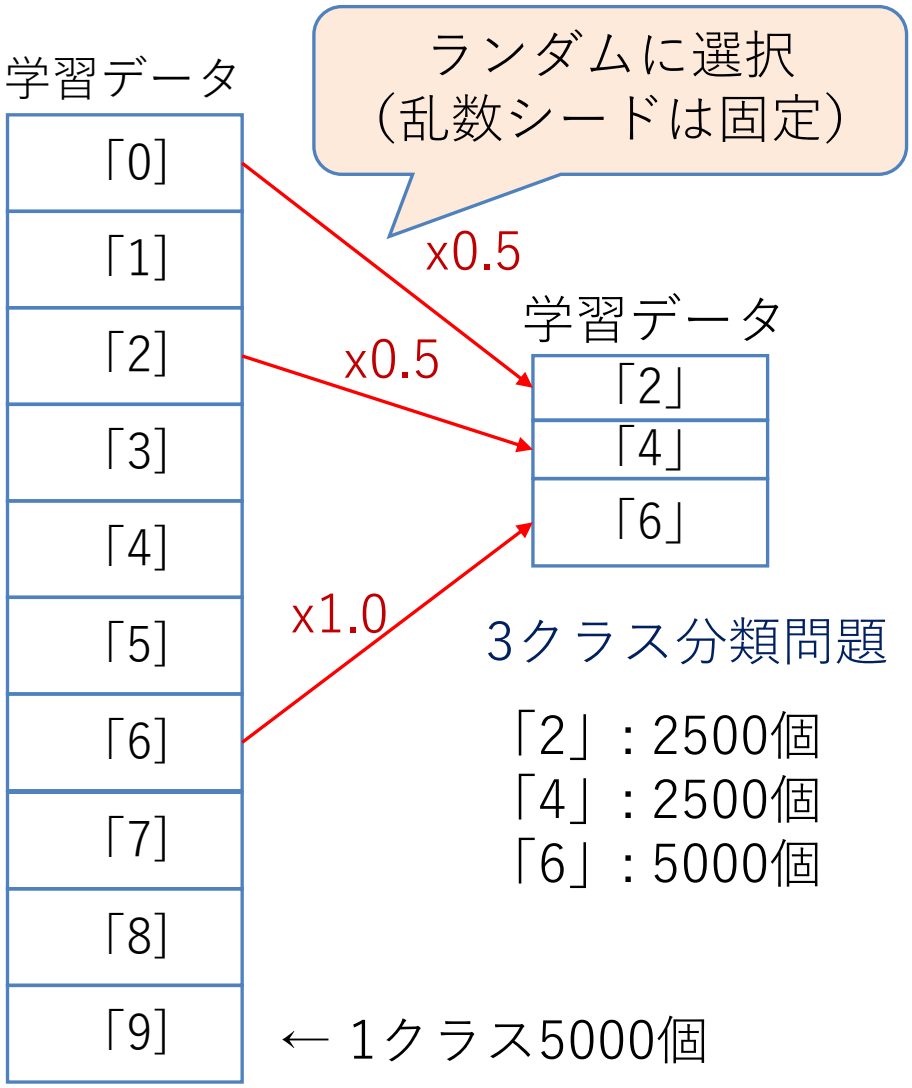
Cifar-10の全体のデータから，クラスとデータ数を指定して抜粋

問題の設定

```
1 # 使用するクラスを指定する
2 use_num = [0, 2, 6]
3
4 # 学習データの使用率
5 # (use_numと同じ要素数のリスト)
6 use_rate = [0.5, 0.5, 1.0]
7
8 print("use_num =", use_num)
9 print("use_rate =", use_rate)
```

```
use_num = [0, 2, 6]
use_rate = [0.5, 0.5, 1.0]
```

use_num=range(10)
use_rate=[1]*len(use_num)
とすると，元の10クラス分類問題

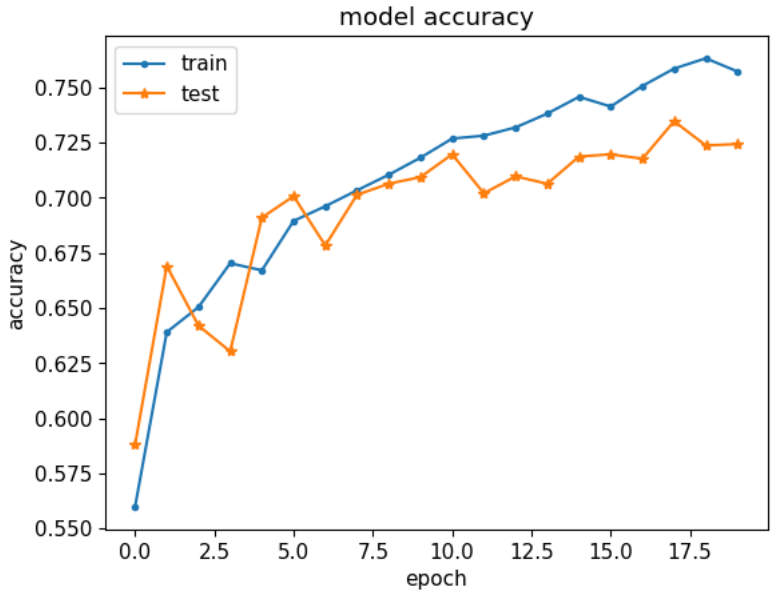


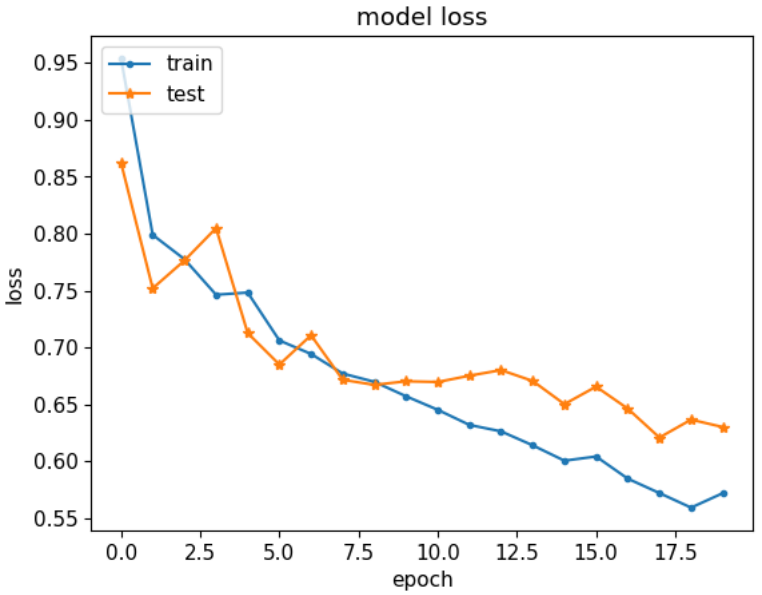
実行例

- 学習結果
- 学習結果のグラフを表示

問題設定
num_classes = 3
use_num = [0, 2, 6]
use_rate = [0.5, 0.5, 0.5]

学習終了時の結果
Train: loss = 0.526, accuracy=0.781
Test: loss = 0.63, accuracy=0.724





レポート用コード:
MLPによるCIFAR-10の分類(クラス数とデータ数可変Ver.)

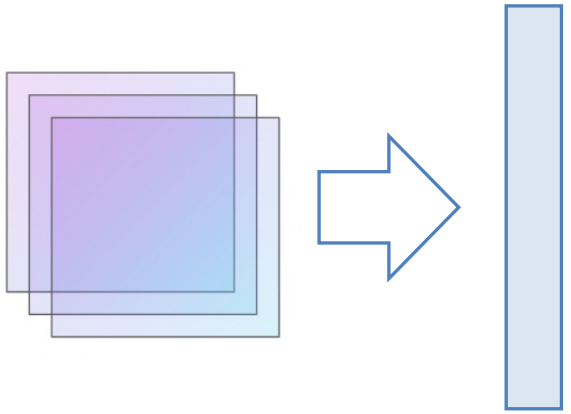
CNN版とほぼ同様，ただしカラー画像を平坦化してモデルに入力

モデルの定義

CIFAR10(RGB画像)の image_shape = (32, 32, 3) を平坦化してMLPに入力する.

- 最初の層は， model.add(Flatten(input_shape=image_shape))
- 最後の層は， Dense(num_classes, activation='softmax')

R, G, Bを
まとめて平坦化



```
image_shape = (32, 32, 3)  # RGB画像
```

```
 8 # モデルを構築
 9 model = Sequential()
10 model.add(Flatten(input_shape=image_shape)) # RGB画像を平坦化
```

平坦化する関数

レポートのテーマ

大きなテーマ：CIFAR-10に対するMLPとCNNの性能比較

各自で詳細なテーマを設定してレポートを作成する

テーマの例
- データ数に関する検証
- クラスの違いに関する検証
- MLP，CNNのモデルの構成に関する検証

- 問題の設定(用いるデータセットについて)：
  - クラス数 n は３以上
  - 各クラスのデータ数は任意

  として，n クラス分類問題を設定する.

クラス数やデータ数が多いと学習に時間がかかる(GPUリソースを多く消費する)ので，実験はある程度少なめのクラスで(３～５クラス程度）で行ってください.

=> あとで余裕があれば，もう少し大規模なデータセットで試す

レポートのテーマ

大きなテーマ：CIFAR-10に対するMMPとCNNの性能比較

各自で詳細なテーマを設定してレポートを作成する

- 問題の設定(用いるデータセットについて)：
  - クラス数 n は３以上
  - 各クラスのデータ数は任意

  として，n クラス分類問題を設定する.

クラス数やデータ数が多いと学習に時間がかかる(GPUリソースを多く消費する)ので，実験はある程度少なめのクラスで(３～５クラス程度）で行ってください.

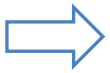
=> あとで余裕があれば，もう少し大規模なデータセットで試す

データ数に関する実験

- データ数を大きくしていくと，学習結果は？

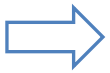
use_rate = [0.1, 0.1, 0.1]

	class	count_train	count_test
0	0:airplane	500	1000
1	2:bird	500	1000
2	6:frog	500	1000



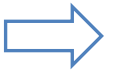
use_rate = [0.3, 0.3, 0.3]

	class	count_train	count_test
0	0:airplane	1500	1000
1	2:bird	1500	1000
2	6:frog	1500	1000



use_rate = [0.5, 0.5, 0.5]

	class	count_train	count_test
0	0:airplane	2500	1000
1	2:bird	2500	1000
2	6:frog	2500	1000

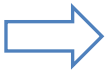


…

- クラス間のデータ数の非均一度合いが大きくなると，学習結果は？

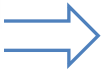
use_rate = [0.1, 0.1, 0.1]

	class	count_train	count_test
0	0:airplane	500	1000
1	2:bird	500	1000
2	6:frog	500	1000



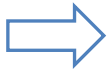
use_rate = [0.1, 0.3, 0.1]

	class	count_train	count_test
0	0:airplane	500	1000
1	2:bird	1500	1000
2	6:frog	500	1000



use_rate = [0.1, 0.5, 0.1]

	class	count_train	count_test
0	0:airplane	500	1000
1	2:bird	2500	1000
2	6:frog	500	1000



…

- データ数に応じてバッチサイズも変えた方が良い？
- 最適なバッチサイズは存在する？

バッチサイズ小　⇔　バッチサイズ大

抜粋するクラスの違いに関する実験

CIFAR10のクラス

0: airplane	機械
1: automobile	機械
2: bird	生物
3: cat	生物
4: deer	生物
5: dog	生物
6: frog	生物
7: horse	生物
8: ship	機械
9: truck	機械

- 動物同士は誤分類しやすい？
  - 例）猫を犬と分類，犬を猫と分類
- 誤分類を減らすためには？
- 同じ３クラス分類問題でも，結果はどう異なるか？

設定１

3: cat	生物
4: deer	生物
5: dog	生物

設定２

0: airplane	機械
1: automobile	機械
8: ship	機械

設定３

3: cat	生物
4: deer	生物
9: truck	機械

モデルの構成に関する実験

モデルの規模・構成と性能の関係は？

パラメータ数：小

	Layer (type)	Output Shape	Param #
0	Conv2D	(30, 30, 32)	896
1	MaxPooling2D	(15, 15, 32)	0
2	Conv2D	(13, 13, 16)	4624
3	MaxPooling2D	(6, 6, 16)	0
4	Flatten	(576,)	0
5	Dense	(16,)	9232
6	Dense	(3,)	51

モデルのパラメータ総数: 14803

パラメータ数：中

	Layer (type)	Output Shape	Param #
0	Conv2D	(30, 30, 64)	1792
1	MaxPooling2D	(15, 15, 64)	0
2	Conv2D	(13, 13, 32)	18464
3	MaxPooling2D	(6, 6, 32)	0
4	Conv2D	(4, 4, 16)	4624
5	MaxPooling2D	(2, 2, 16)	0
6	Flatten	(64,)	0
7	Dense	(16,)	1040
8	Dense	(3,)	51

モデルのパラメータ総数: 25971

パラメータ数：大

	Layer (type)	Output Shape	Param #
0	Conv2D	(30, 30, 128)	3584
1	MaxPooling2D	(15, 15, 128)	0
2	Conv2D	(13, 13, 64)	73792
3	MaxPooling2D	(6, 6, 64)	0
4	Conv2D	(4, 4, 32)	18464
5	MaxPooling2D	(2, 2, 32)	0
6	Flatten	(128,)	0
7	Dense	(64,)	8256
8	Dense	(32,)	2080
9	Dense	(3,)	99

モデルのパラメータ総数: 106275

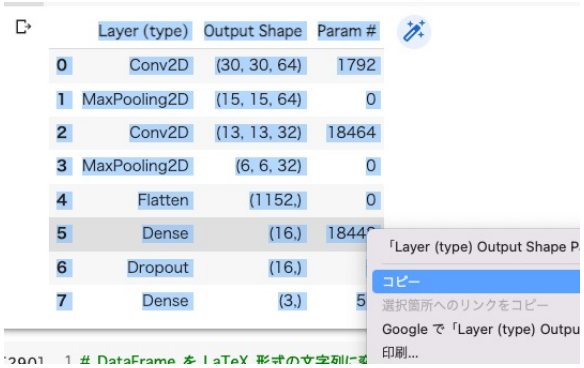
- 大規模なモデル（パラメータ数が多い or 層が多い）ほど性能は良い？
- 学習に必要なエポック数は？　(学習時間は？)
- 効果的な層の組み合わせ(層の数，種類)は？
- Dropoutの効果は？
- なるべく小さな規模のモデルで性能を向上させるには？

データフレームの変換(1)

- データフレームをエクセルに貼り付け

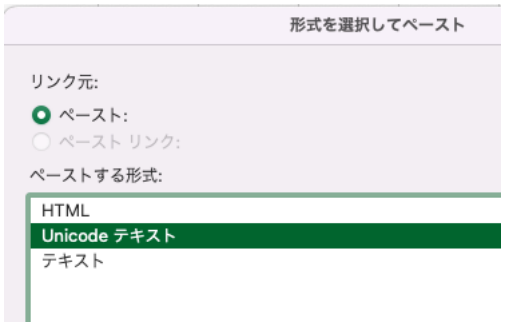
```
# モデルの情報をDataFrameで表示する
# model_summary_to_dataframe関数は上のセルで定義
model_df = model_summary_to_dataframe(model)
model_df
```

	Layer (type)	Output Shape	Param #
0	Conv2D	(30, 30, 64)	1792
1	MaxPooling2D	(15, 15, 64)	0
2	Conv2D	(13, 13, 32)	18464
3	MaxPooling2D	(6, 6, 32)	0
4	Flatten	(1152,)	0
5	Dense	(16,)	18448
6	Dropout	(16,)	0
7	Dense	(3,)	51



データフレームをマウスで
範囲指定して右クリック
(or 2本指タップ)
でメニューからコピー

エクセルを開く



形式を選択してペーストから
Unicodeテキストを選択

Layer (type)	Output Shape	Param #	
0	Conv2D	(30, 30, 64)	1792
1	MaxPooling2	(15, 15, 64)	0
2	Conv2D	(13, 13, 32)	18464
3	MaxPooling2	(6, 6, 32)	0
4	Flatten	(1152,)	0
5	Dense	(16,)	18448
6	Dropout	(16,)	0
7	Dense	(3,)	51

LaTeX Table Generator
(LaTexへ変換)

データフレームの変換(2)

データフレームの内容をLatexの表として貼り付け

```python
# DataFrame を LaTeX 形式の文字列に変換する
latex = model_df.to_latex(index=False)

# LaTeX コードに横線を追加する
styled_latex = latex.replace('\\begin{tabular}', '\\be
                             .replace('\\hline', '\\midrule')\
                             .replace('\\end{tabular}', '\\bott

# LaTeX コードを出力する
print(styled_latex)
```

```
\begin{tabular}{llr}
\toprule
Layer (type) & Output Shape &  Param \# \\
\midrule
    Conv2D & (30, 30, 64) &    1792 \\
MaxPooling2D & (15, 15, 64) &       0 \\
    Conv2D & (13, 13, 32) &   18464 \\
MaxPooling2D &  (6, 6, 32) &       0 \\
   Flatten &     (1152,) &       0 \\
     Dense &      (16,) &   18448 \\
   Dropout &      (16,) &       0 \\
     Dense &       (3,) &      51 \\
\bottomrule
\bottomrule
\end{tabular}
```

```
<ipython-input-315-31c3882ac0cd>:2: FutureWarning: In
 latex = model_df.to_latex(index=False)
```

コピー

```
\begin{tabular}{llr}
⋮
\end{tabular}
```

をLaTexの表のサンプルに貼り付ける

```latex
% 表の作成
\begin{table}[htbp]
\begin{center}
\caption{表のサンプル}
\label{tbl:sample_table}

\begin{tabular}{|l|c|c|}
\hline
      & 平均 & 標準偏差   \\ \hline
手法1 & 10.5 & 4.5    \\ \hline
手法2 & 11.4 & 2.1    \\ \hline
手法3 & 21.5 & 3.8    \\ \hline
\end{tabular}

\end{center}
\end{table}
```

表 1: 表のサンプル

	平均	標準偏差
手法 1	10.5	4.5
手法 2	11.4	2.1
手法 3	21.5	3.8

```latex
% 表の作成
\begin{table}[htbp]
\begin{center}
\caption{表のサンプル}
\label{tbl:sample_table}

\begin{tabular}{llr}
\toprule
Layer (type) & Output Shape &  Param \# \\
\midrule
      Conv2D & (30, 30, 64) &     1792 \\
MaxPooling2D & (15, 15, 64) &        0 \\
      Conv2D & (13, 13, 32) &    18464 \\
MaxPooling2D &  (6, 6, 32) &        0 \\
     Flatten &     (1152,) &        0 \\
       Dense &      (16,) &    18448 \\
     Dropout &      (16,) &        0 \\
       Dense &       (3,) &       51 \\
\bottomrule
\bottomrule
\end{tabular}

\end{center}
\end{table}
```

表 1: 表のサンプル

Layer (type)	Output Shape	Param #
Conv2D	(30, 30, 64)	1792
MaxPooling2D	(15, 15, 64)	0
Conv2D	(13, 13, 32)	18464
MaxPooling2D	(6, 6, 32)	0
Flatten	(1152,)	0
Dense	(16,)	18448
Dropout	(16,)	0
Dense	(3,)	51

レポートの構成の例

1. **目的**
   ✓レポートの目的を書く.
2. **CIFAR-10データセット**
   ✓データセットの説明を書く.
3. **問題の設定**
   ✓各自の問題の設定を説明する.
4. **モデルの設定**
   ✓実験で用いるモデルの種類や構成を説明する.
5. **実験**
   ✓実験の手順や結果を説明する.
6. **考察（or まとめ）**

参考文献

実験結果のまとめや，モデルの性能差が付いた理由など，考えらえることを書く.

参考文献について

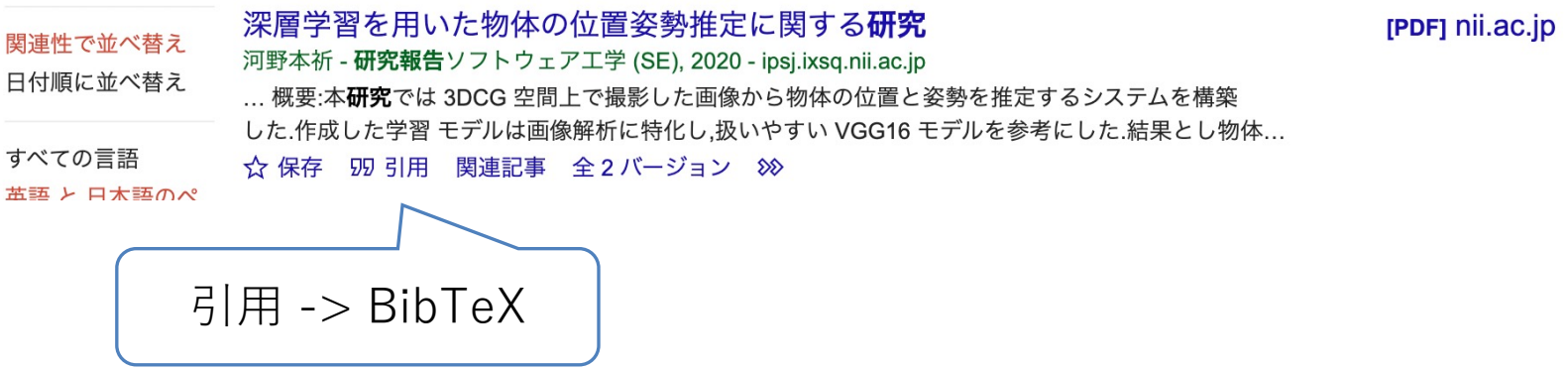
Google schlorで「Cifar10 研究報告」や「畳み込み 研究報告」など

「キーワード + 研究報告」とすると，**研究会などの原稿**がヒットしやすい

情報処理学会研究報告

- ページ数は少なめ
- (基本的に)査読なし

文献情報について



文献情報

```
@article{河野本祈2020深層学習を用いた物体の位置姿勢推定に関する研究,
  title={深層学習を用いた物体の位置姿勢推定に関する研究},
  author={河野本祈 and others},
  journal={研究報告ソフトウェア工学 (SE)},
  volume={2020},
  number={5},
  pages={1--8},
  year={2020}
}
```

参考文献

[1] Xiaodong Li, "Efficient differential evolution using speciation for multimodal function optimization", Proc.of GECCO2005, pp.873-880, 2005.

[2] 木村孝作, 原章, 市村匠, 高濱徹行, "近傍グラフに基づく種分化を導入した Differential Evolution による複数解の探索", 第 26 回ファジィシステムシンポジウム (FSS2010), pp.566-571, 2010.

著者，タイトル，雑誌名，ページ数，年

補足

- レポートチェックリスト

第2週レポートチェックリストの項目を，すべて確認(チェック)した上で提出すること

- オンラインtex環境