機械学習
ハンズオン-チュートリアル

～初めてのペアモニター研究～

はじめに

- このチュートリアルは機械学習の環境を構築し、ニューラルネットワークが実行できるようになるところまで行います。

チュートリアルの流れ

1. 環境構築
2. 機械学習用プログラム実装&実行
3. プログラムの改良(精度向上のため)
4. 機械学習についてより深く理解するために

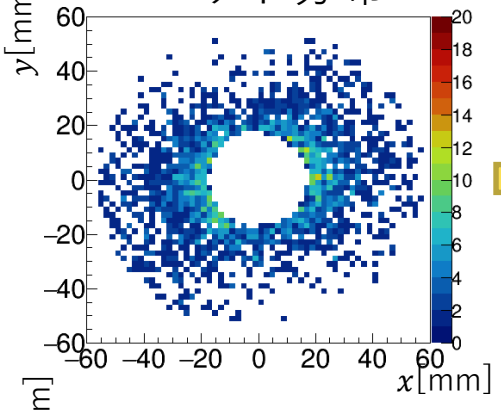
機械学習

手書き文字　　認識

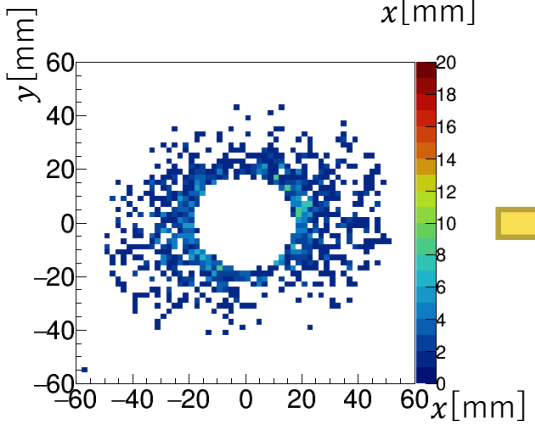
5

0

ヒット分布　　認識



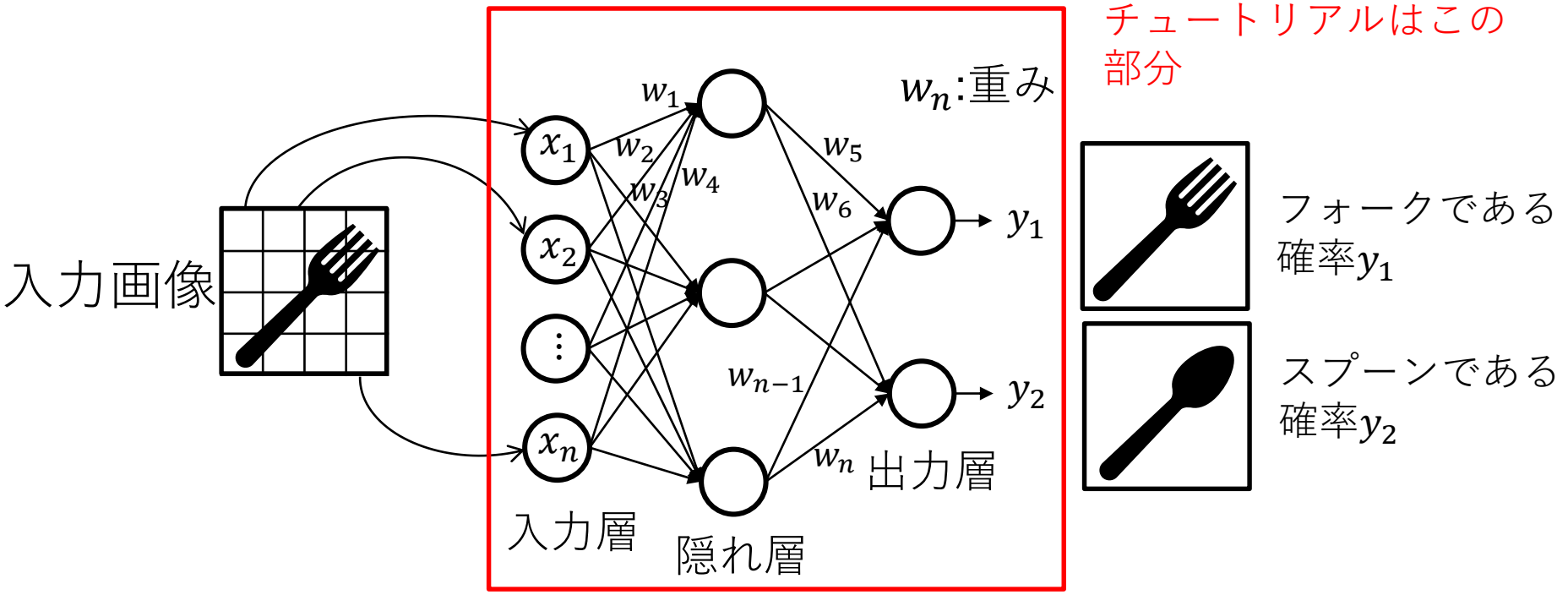
$\sigma_x = 1.0\sigma_x^{TDR}$



$\sigma_x = 1.5\sigma_x^{TDR}$

- ビームサイズ再構成は文字認識問題と同様に扱える。
- 近年大量の画像を入力として利用し文字認識や物体検知などを機械学習によって行う手法が発展している。

ニューラルネットワーク



- 神経細胞を模したパーセプトロンを層状に並べた構造を持つ
- 隠れ層が複数ある場合、
  ディープニューラルネットワークと定義

環境構築

使用するマシンについて

- 今回は実験IIIで使用しているマシンにsshで接続して作業を行います。

User	IPアドレス
cs002	172.18.33.228
cs003	172.18.33.214
cs006	172.18.33.95
cs009	172.18.33.221

Password : Jikken3

ssh接続

実験IIIマシンにssh接続を行います。

```
$ ssh –XY cs00*@172.18.33.***
```

接続後

```
$ ls
```

を打ち込み、machine_learningディレクトリがあるか確認してください。

```
Machine_learning
 ├train_data_sigmay_y0238
 ├test_data_sigmay_y0238
 ├train_data_sigmay_y0238.tar.gz
 ├test_data_sigmay_y0238.tar.gz
 ├cpf.sh
```

使用しません

仮想環境構築

Python実行用の仮想環境を作成します。これによりPython本体と異なる機械学習用の環境を構築できます。

メリット

- ソフトウェアアップデートをしたらPythonのバージョンが上がりライブラリが動かなくなるなどのトラブルを防ぐ
- 共有サーバーで権限がないため、ライブラリをインストールできない場合でも仮想環境ならインストール可能になる

構築方法

Pythonのvenvモジュールを使用します。

構築方法

Homeディレクトリで

```
$ python3 –m venv [仮想環境名]
```

を実行します。

-m: モジュールを使うためのオプション
venv:モジュール名
[仮想環境名]:短い方がおすすめ。ここではtf

仮想環境名のディレクトリが作成されたら成功。

仮想環境有効化・終了方法

- 有効化

```
$ source ./tf/bin/activate
```

端末の表示が

```
cs00*@cs00* ~:$
```

から

```
(tf) cs00*@cs00* ~:$
```

に変わります。

- 終了

```
$ deactivate
```

有効後と終了後に`$ python –V`と入力して両者を比べるとバージョンが変わっていることが確認できます。

機械学習用ライブラリ

機械学習を行うためのライブラリとして
TensorFlow,Kerasを使用します。

- TensorFlow 

  Googleが開発した機械学習のためのオープンソースライブラリ。
- Keras 

  Pythonで書かれた，TensorFlowまたはCNTK，Theano上で実行可能な高水準のニューラルネットワークライブラリです．(公式より)

  ざっくり言うと、TensorFlowスペシャリストでなくてもTensorFlowを動かすことが可能になるラッパーです。

ライブラリインストール

仮想環境を有効化し

```
$ pip install tensorflow keras
```

を実行します。

他の必要なライブラリもインストールしてください

```
$ pip install matplotlib opencv-python pillow tqdm
```

グラフ描画,画像処理,
など…

```
$ pip install pydot graphviz
```

```
$ sudo apt-get install python3-tk graphviz
```

ニューラルネット
モデル可視化

ソースコードダウンロード

ソースコードをgithubからダウンロードします。

Homeディレクトリで

```
$ git clone https://github.com/ykoba84/pair_monitor_study.git
```

ディレクトリ構成

```
Home
 ├pair_monitor_study
 ├machine_learning
```

pair_monitor_study中身

pair_monitor_study
- ├ **VGGnet.py** (CNNプログラム。 σ_y用)
- ├ **VGGnet_2para.py** (CNNプログラム。 σ_x ,σ_y用)
- ├ **count.py** (ヒット分布が何枚あるかカウント)
- ├ **dataset.py** (データセット読み込み時に使用)
- ├ **graph_plot.py** (グラフプロット)
- ├ **nn.py** (ニューラルネットワークプログラム。動作確認用)
- ├ **outputs.py** (学習結果の出力に使用)
- ├ **utilities/** (ビームサイズ再構成の計算に使用)

プログラム実装&実行

プログラム実行

pair_monitor_studyディレクトリ内のnn.pyを実行します。

その前にnn.pyを書き換えます。

nn.pyの31~34行目

```
# make train_data
x_train, y_train = ds.read_data_sigmay(" ../ML_Hans-on_Tutorial/train_data_sigmay_y0238 ")
# make test_data
x_test, y_test = ds.read_data_sigmay("../ML_Hans-on_Tutorial/test_data_sigmay_y0238")
```

```
# make train_data
x_train, y_train = ds.read_data_sigmay(”../machine_learning/train_data_sigmay_y0238")
# make test_data
x_test, y_test = ds.read_data_sigmay(”../ machine_learning/test_data_sigmay_y0238 ")
```

$ python nn.pyと入力し実行

lossとは

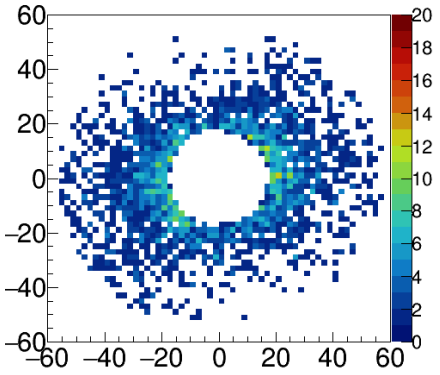
ニューラルネットワークの出力がどれだけラベル(目標値)に近いかを測る尺度

d_{nk}：目標値
y_{nk}：ニューラルネット出力
N :画像数
K :ラベル数

$$L = -\sum_{n}^{N}\sum_{k}^{K} d_{nk} \log y_{nk}$$

例

$\sigma_x = 1.0\sigma_x^{TDR}$



画像の目標値はラベルと対応する要素が1、他は0で表現

$$d_n = [0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad \ldots 0]$$

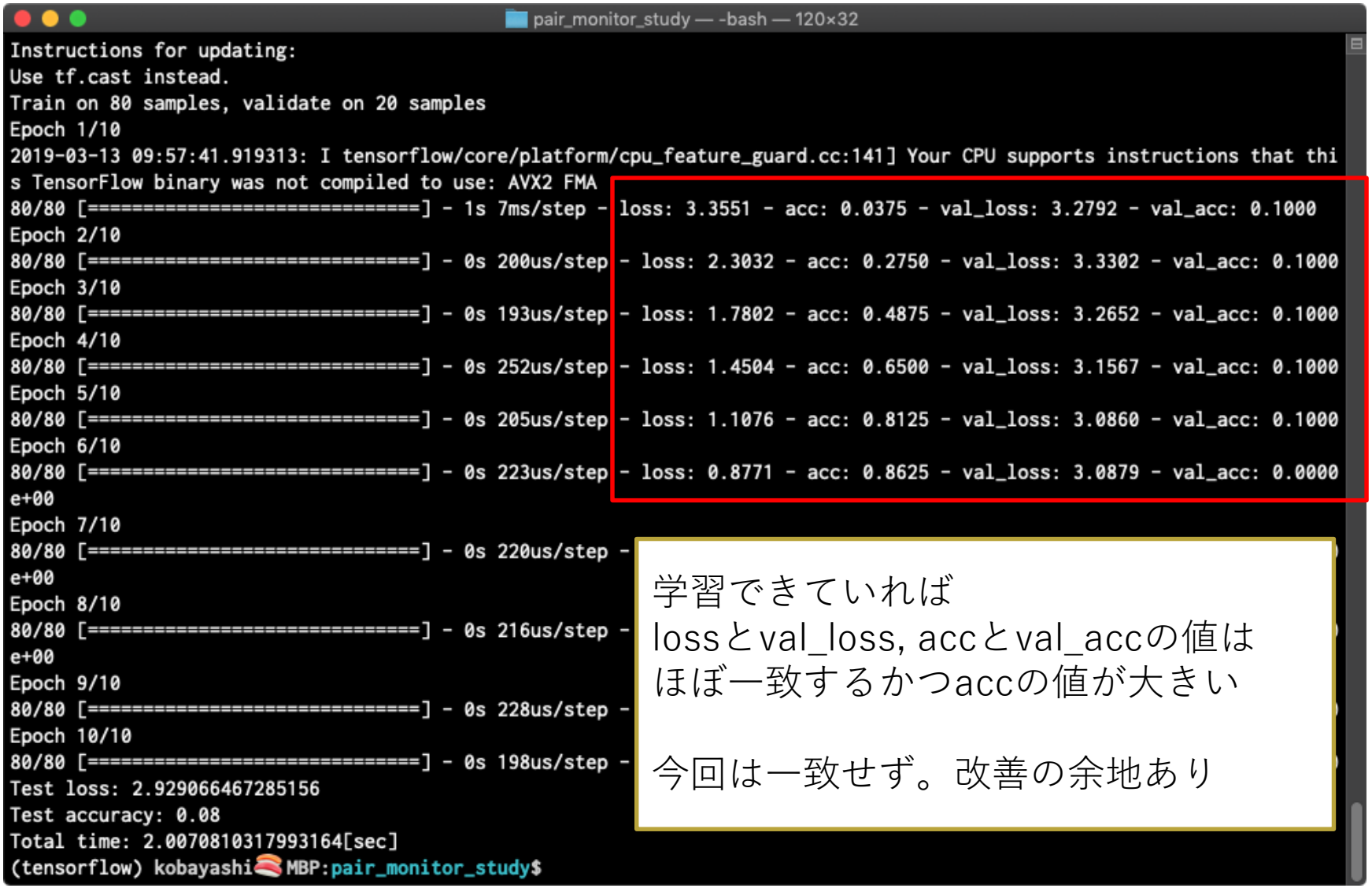
σ_x/σ_x^{TDR}	0.2	0.4	0.6	0.8	1.0	1.2	…	3.8

出力=目標値となるようにwを調整する。 y_n =~であれば…

$$y_n = [0 \quad 0 \quad 0 \quad 0.01 \quad 0.98 \quad 0.01 \quad \ldots 0]$$

$$L = -\log 0.98 = 0.020$$

出力=目標値のとき$L = 0$　　**$L = 0$となるように学習**



学習できていれば
lossとval_loss, accとval_accの値は
ほぼ一致するかつaccの値が大きい

今回は一致せず。改善の余地あり

生成されたファイルの確認

pair_monitor_study/result/nn/190202_151902

- ├score.txt (損失関数値と正解率の最終結果)
- ├model.png (ニューラルネットの構造)
- ├total_time.txt (計算時間)
- ├prediction.csv (ビームサイズの確率分布)
- ├hyperparameter.txt (抜粋したハイパーパラメータの値)
- └image.png (トレーニング中の損失関数と正解率の推移)

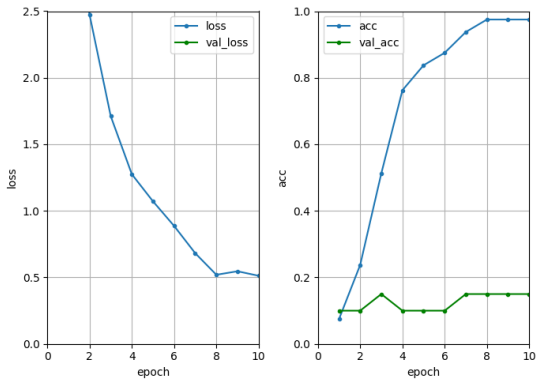


image.png

精度向上のための手法

お疲れ様でした。ここまで出来れば機械学習の導入は終了です。

ここからはプログラムを改良して精度が高いニューラルネットの構築を目指します。

画像を増やす

実は画像はトレーニングとテストデータでそれぞれ100枚しか読み込んでいません。トレーニングデータの枚数は多くあった方が精度向上しやすいです。

dataset.py 88~90行目

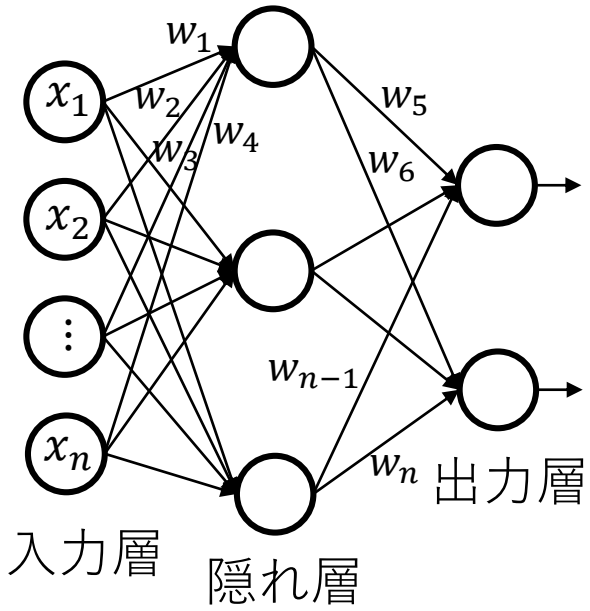
```
nfile=nfile+1
if nfile>100:
        break
```

```
“””
nfile=nfile+1
if nfile>100:
        break
“””
```

コメントアウト
(クォーテーションを3回)

ユニットを増やす

- 隠れ層のユニットの数を増やすとパラメータが増え特徴を捕らえやすくなります。ただし増やしすぎると過学習しやすくなる危険性もあります。
- 入力層、出力層のユニット数は固定



nn.py 51行目

```
h1 = Dense(100)(inputs)
```

```
h1 = Dense(1000)(inputs)
```

層を増やす

- 隠れ層の数を増やします。ただ個人的に4,5層増やしても精度は向上しませんでした。増やすとしても1,2層が限度。

nn.py 55~67行目

コメントアウトを外す

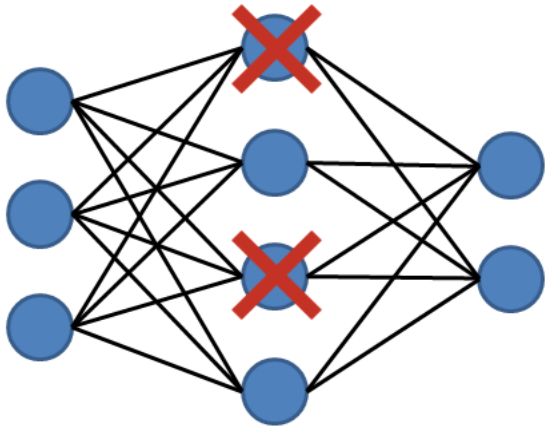
```
"""
# Hiddin2 layer as follows:
h2 = Dense(500)(h1)
#h2 = BatchNormalization()(h2)
h2 = Activation('relu')(h2)
#h2 = Dropout(0.5)(h2)


# Hiddin3 layer as follows:
h3 = Dense(500)(h2)
#h3 = BatchNormalization()(h3)
h3 = Activation('relu')(h3)
#h3 = Dropout(0.5)(h3)
"""
```

その他の改良

- Dropout

  設定した割合に基づいてユニットを無効化させ過学習を抑制する。



- BatchNormalization

  値を平均0分散1に正規化する。収束を早める
- 畳み込みニューラルネットワーク

より勉強するために

- 参考になった図書

岡谷貴之,『機械学習プロフェッショナルシリーズ 深層学習』, 講談社

ニューラルネットワーク、畳み込みニューラルネットワークの理論について説明している。東大の人工知能論講義の教科書にも選ばれている

ubuntu コマンドで画像表示

$ eog [画像]