

人工知能プログラム入門

講師

西村 晋

本日の課題

一 時 間 目

セットアップ

- ・レッスン資料をダウンロード
- ・ノートブックの操作確認

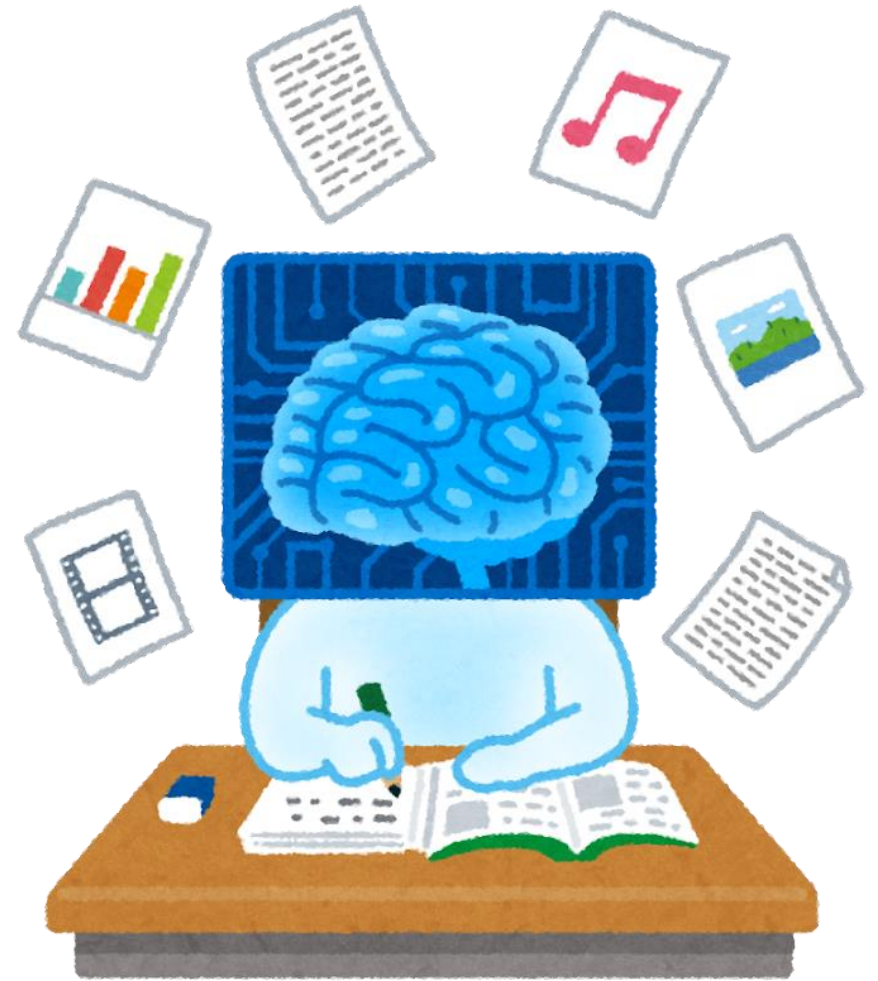
分類 A I を使ってみよう

- ・画像分類 A I について
- ・画像分類 A I を作ってみよう

二 時 間 目

A I の仕組みを学ぼう

- ・画像データ確認
- ・画像の成り立ちについて
- ・インプットとアウトプットについて
- ・手書き数字の画像分類 A I を作ってみよう
- ・ニューラルネットワークについて



セッ ト ア ッ プ

一時間目

レッスン資料をダウンロード

まずは、こちらのリンクまたはUSBメモリーから資料をダウンロードしましょう。

www.T.LY/dbFRe

ダウンロード後、ファイルの解凍を行きましょう。

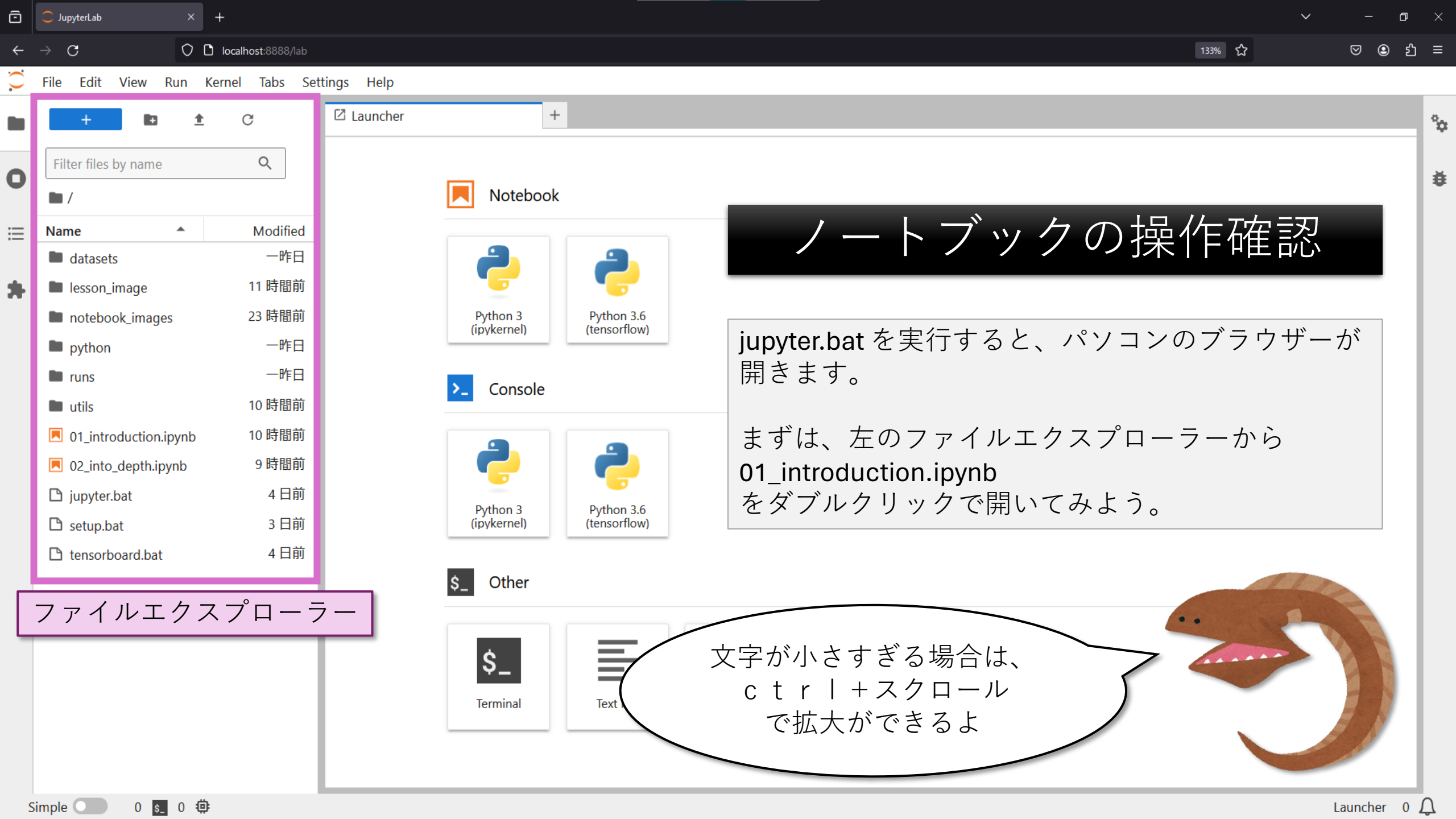
次に `setup.bat` を実行（5分ほどかかります）

そして `tensorboard.bat` を実行

最後に `jupyter.bat` を実行



右クリックして
「すべて展開」を押せば
ファイルの解凍ができるよ



ノートブックの操作確認

jupyter.bat を実行すると、パソコンのブラウザーが開きます。

まずは、左のファイルエクスプローラーから
01_introduction.ipynb
をダブルクリックで開いてみよう。

文字が小さすぎる場合は、
c t r l + スクロール
で拡大ができるよ



lab - JupyterLab

localhost:8888/lab

133%

FileEditViewRunKernelTabsSettingsHelp

+

+

↑

↺

Filter files by name

/

Name	Modified
datasets	一昨日
lesson_image	13 時間前
notebook_images	昨日
python	一昨日
runs	一昨日
utils	12 時間前
01_introduction.ipynb	59 分前
02_into_depth.ipynb	12 時間前
jupyter.bat	4 日前
setup.bat	3 日前
tensorboard.bat	4 日前

01_introduction.ipynb

Python 3 (ipykernel)

Markdown

ノートの操作確認

本日の課題 1

Lesson structure

①画像分類AIとは？
- What is an image classification AI?

②画像分類AIを作ってみよう
- Let's create an image classification AI

STEP 1
AIを作るために訓練用の画像をダウンロードしましょう
- Let's download images for our AI to train on

STEP 2
ダウンロードした画像の確認
- Check out the images downloaded

STEP 3
AIモデルを作り、ダウンロードした画像を使って訓練を実行
- Let's create the AI model and train it on the images downlo

STEP 4
画像をダウンロードして正しく分類してくれるか確認しよう
- Let's download some images manually and check that our model actually works

01_introduction.ipynb を開けると、このような画面になっているはずです。

ノートブックには今回使うコードや、コードの説明、画像などが載っています。

通常のプログラミングと違い、ノートブックでは少しずつコードを実行できるのでべんりです。

では、基本操作を覚えていきましょう！

Simple

0

Python 3 (ipykernel) | Idle

Mode: Command

Ln 1, Col 1

01_introduction.ipynb

0

セルについて

コードとマークダウン

ノートブックはセルで構成されていて、セルには二種類のものがあります。

- ・コード

そのままの意味であり、Pythonのコードが含まれています。

- ・マークダウン

今回は触りませんが、コードの説明テキストや画像などが貼ってあります。

選択中のセルには左側に青い縦線が付きま

セル（マークダウン）

STEP 2

ダウンロードした画像の確認

- Check out the images downloaded

ちゃんと画像がダウンロードできたかどうか確認しよう。

Let's check if our images were downloaded correctly.

セル（コード）

```
[ ]: from utils import open_file
      from PIL import Image
      |
      Image.open(open_file(ds_path))
```

自分が思ったことと違う画像が保存されていることはよくあることだ。例えば「apple」と検索した場合、果物のリンゴが出てこず、「Apple」のブランド製品が出てきたりする。AIではどのようなデータが訓練に使われているかを確認することも大事な作業だが、今回はこのままにしておく。

選択中マーク

be what you expect. For example, if you entered the query "apple" hoping to get
ple, you may result in the brand "Apple". Checking your datasets when training AI is
important, though in this case we will not be dealing with these unexpected images.

コードの実行と基本操作

ノートブックの上部分を見ると、いくつかのボタンがあります。

- ・ 保存ボタン

万が一パソコンがシャットダウンを起こすといけないため、定期的に保存しましょう。

- ・ 実行ボタン

選択中のセルを実行します。

- ・ リセットボタン

プログラムをリセットし、変数などが消されます。

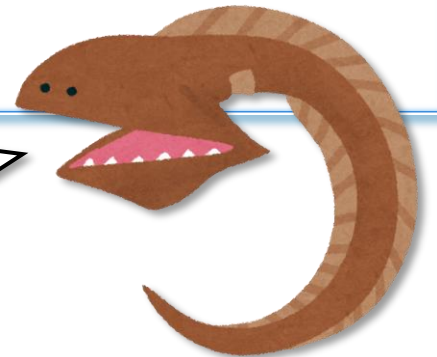
A screenshot of a Jupyter Notebook interface. The top toolbar is visible with the same icons as the diagram above. The main area shows a code cell with the following Python code:

```
[ ]: from utils import DatasetDownloader, save_file

ds_path = save_file()
dd = DatasetDownloader(ds_path)

# What images do you want to download?
queries = (
    "rose_flower",      # <<<
    "tulip_flower",     # <<<
    "sunflower_flower", # <<<
    "daisy_flower",     # <<<
    "poppy_flower",     # <<<
)
dd.dl_jobs(queries)
```

保存は `ctrl + S`
実行は `シフト + エンター`
でできますよ



分類 A I を使ってみよう

一時間目

画像分類 A I について

インプット（入力）

- ・ 画像

アウトプット（出力）

- ① 猫の確率
- ② 犬の確率
- 等々

アウトプットの数はどうの
ようにプログラムされたかに
よりますよ



画像分類
モデル

猫: 100%
犬: 0%
...

画像分類 AI を作ってみよう

① まずは画像をダウンロード

コードを実行する前に

大量ダウンロードしたい画像を
コードの **queries** の中に入力し
よう。

検索ワードは英字半角で
“クォテーションマーク”
で閉じて、右にカンマ
「,」
をつけるのを忘れずに。

STEP 1

AIを作るために訓練用の画像をダウンロードしましょう
- Let's download images for our AI to train on

まず初めに、AIとは何をすべきか全く分からない状態からスタートするため、練習（正式には訓練と言う）用の画像を渡す必要がある。今回、オンラインの検索エンジンから数百枚の画像をダウンロードして、今から作っていくAIに渡ししょう。

First and foremost, AIs do not start out knowing what to do*, and requires training. We will be downloading a few hundred different images, and we will be feeding it to the AI model for it to learn on.

② コード実行

```
[ ]: from utils import DatasetDownloader, save_file

ds_path = save_file()
dd = DatasetDownloader(ds_path)

# What images do you want to download?
queries = [
    "rose_flower",
    "tulip_flower",
    "sunflower_flower",
    "daisy_flower",
    "poppy_flower",
]

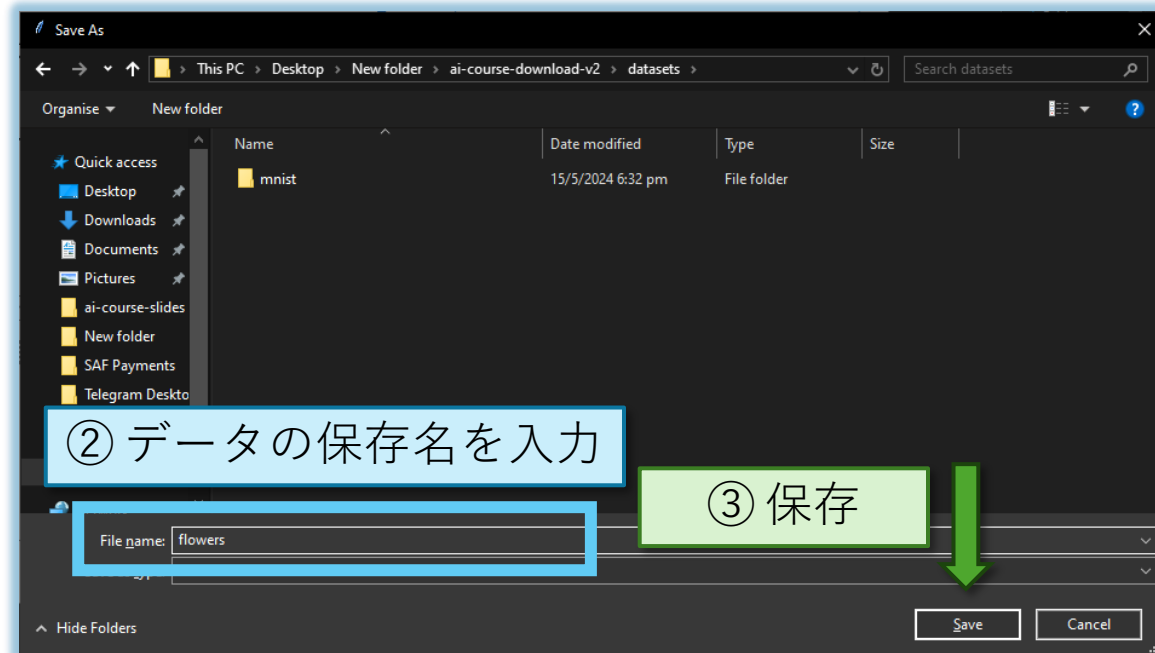
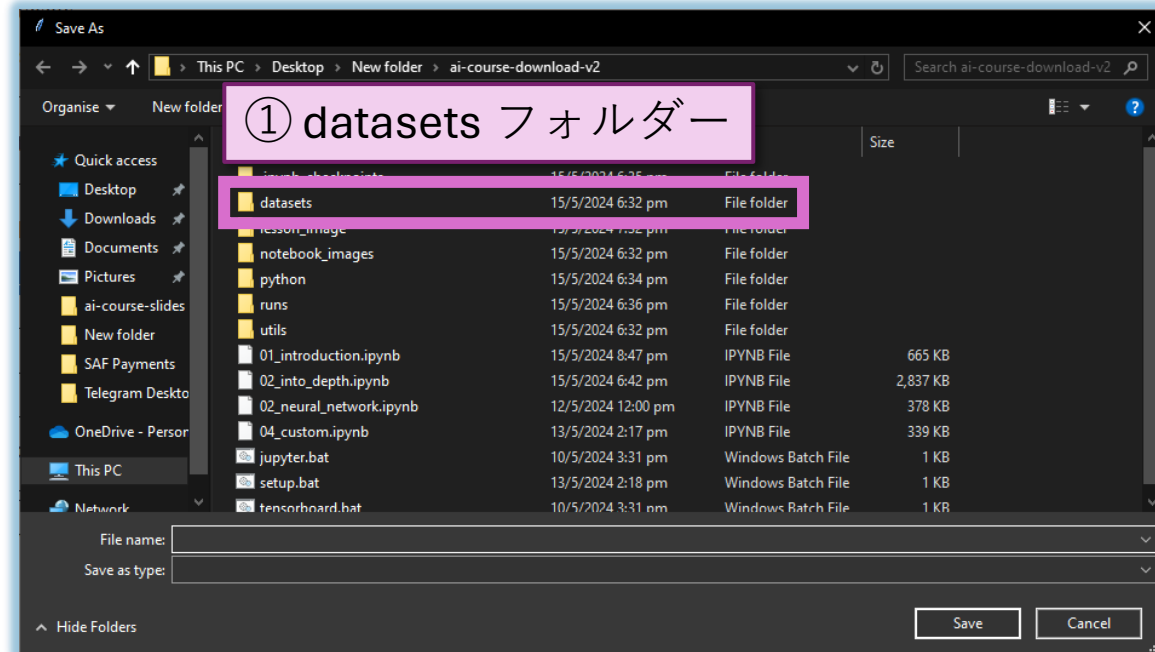
dd.dl_jobs(queries, hundreds=1)
```

① 検索ワード入力

画像の保存先を選ぼう

コードを実行すると、ウィンドウのポップアップが出てきて保存先を問われます。

フォルダー「datasets」をダブルクリックで中に入り、好きな名前前でデータを保存しましょう。



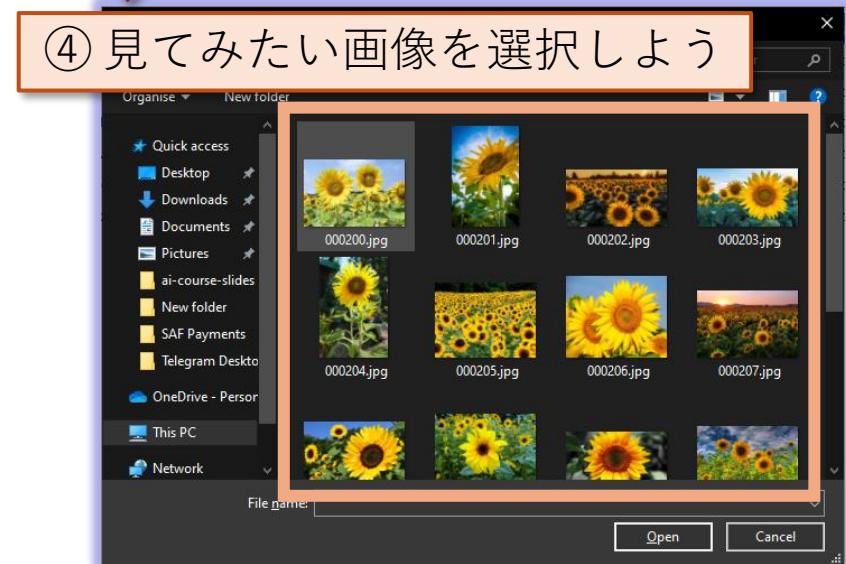
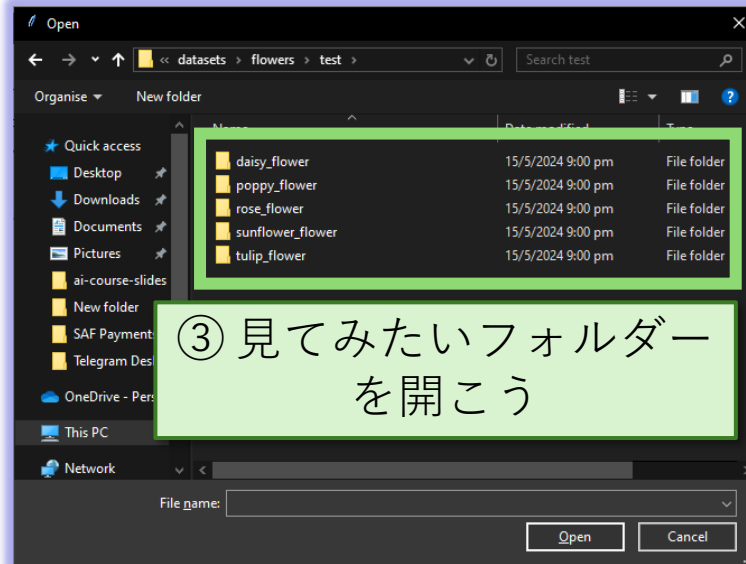
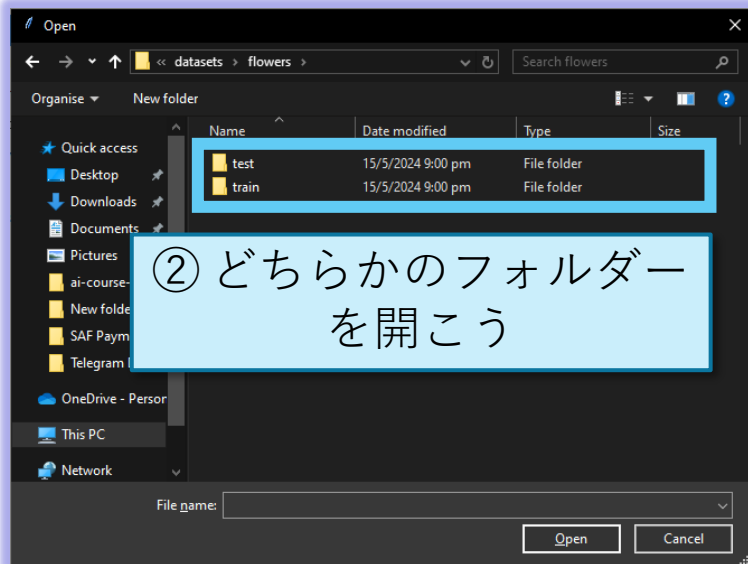
画像分類 A I を作ってみよう

②ダウンロードした画像の確認

次に、STEP 2 のコードを実行しましょう。

test あるいは train のフォルダーを開き、先ほど入力した検索ワードのフォルダーを開きましょう。

画像を選択すると、コードの下に現れます。



画像分類 AI を作ってみよう

③ 訓練を実行しよう

まずは **STEP3** のコードを実行しましょう。

次に、マークダウンのリンクあるいは以下のウェブサイトを開きましょう。

www.localhost:6006

すると、訓練の進捗を表すグラフが見えます。

STEP 3

AIモデルを作り、ダウンロードした
- Let's create the AI model and train

① コード実行

```
[ ]: from torchvision.models import shufflenet_v2_x0_5, ShuffleNet_V2_X0_5_Weights
from torchvision import datasets, transforms
from utils import train
from torch import nn
import os

os.environ["TORCH_HOME"] = os.path.join(os.getcwd(), "utils")
data_transform = ShuffleNet_V2_X0_5_Weights.IMAGENET1K_V1.transforms()

model = shufflenet_v2_x0_5(weights=ShuffleNet_V2_X0_5_Weights.IMAGENET1K_V1)
for param in model.parameters():
    param.requires_grad = False
model.fc = nn.Sequential(
    nn.Linear(1024, len(queries)),
    nn.Softmax(dim=1)
)

train_dataset = datasets.ImageFolder(
    os.path.join(ds_path, "train"),
    transform=data_transform
)

test_dataset = datasets.ImageFolder(
    os.path.join(ds_path, "test"),
    transform=data_transform
)

train(model, train_dataset, test_dataset, 10)
```

正確度グラフを見てみよう

Run tensorboard and open the link below

② リンクを開きましょう

localhost のウェブサイト
は自分のパソコンでホスト
されているんですよ



画像分類 A I を作ってみよう TensorBoard の使い方

リンクを開くと、右のようなタブが開きます。

TensorBoard は A I でよく使われるライブラリであり、訓練の進捗を図に表してくれます。

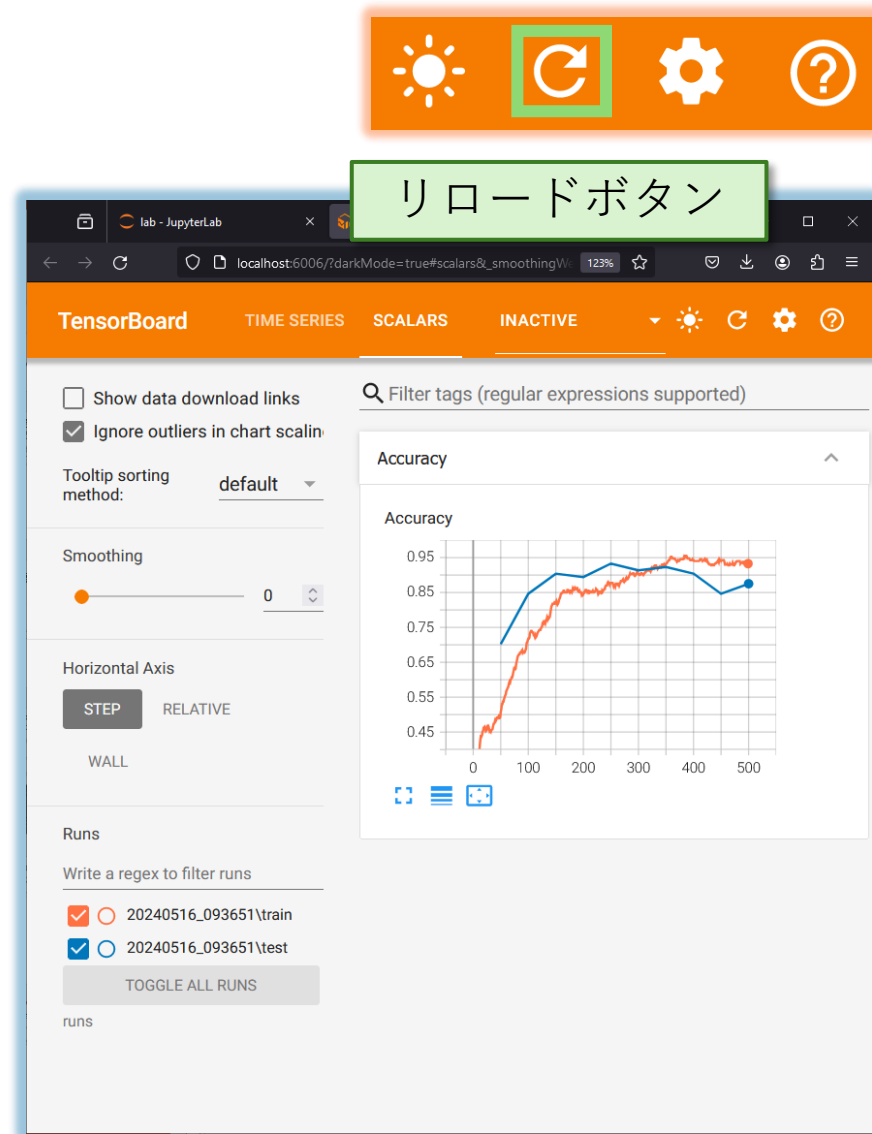
グラフは

拡大・縮小：a | t + スクロール

ドラッグ：a | t + クリック移動

エリア選択：クリック移動

で操作できます。



グラフサイズ拡大ボタン



全グラフ表示ボタン

画像分類 A I を作ってみよう

④ 作った A I を試してみよう

実際に A I に画像を送り込み、ちゃんと分類するか試してみよう。

まず、A I に分類してほしい画像をグーグルからダウンロードしましょう。

そして **STEP3** のコードを実行し、ダウンロードした画像を選択。するとコードの下にその画像と分類結果が現れます。



① 画像をダウンロード

STEP 4
画像をダウンロード
- Let's download s
model actually wo

② コード実行

```
[6]: from utils import visualize, open_file  
  
visualize(  
    open_file(), model, data_transform, test_dataset.classes  
)
```

daisy_flower	0.0
poppy_flower	0.7
rose_flower	0.0
sunflower_flower	0.0
tulip_flower	0.3

Open

← → ↑ ↓ This PC > Downloads > Search Downloads

Organize New folder

★ Quick access

- Desktop
- Shin Nishimura
- This PC
- Libraries
 - Camera Roll
 - Documents
 - Music
 - Pictures
 - Saved Pictures
 - Videos
 - Network

Today (3)

- image.jpg
- ai-course-do wnload-v2
- temp

A long time ago (1)

③ ダウンロードした画像を選択

File name: image.jpg

Open Cancel

A I の仕組みを学ぼう

二時間目

A I の仕組みを学ぼう

二時間目の目標

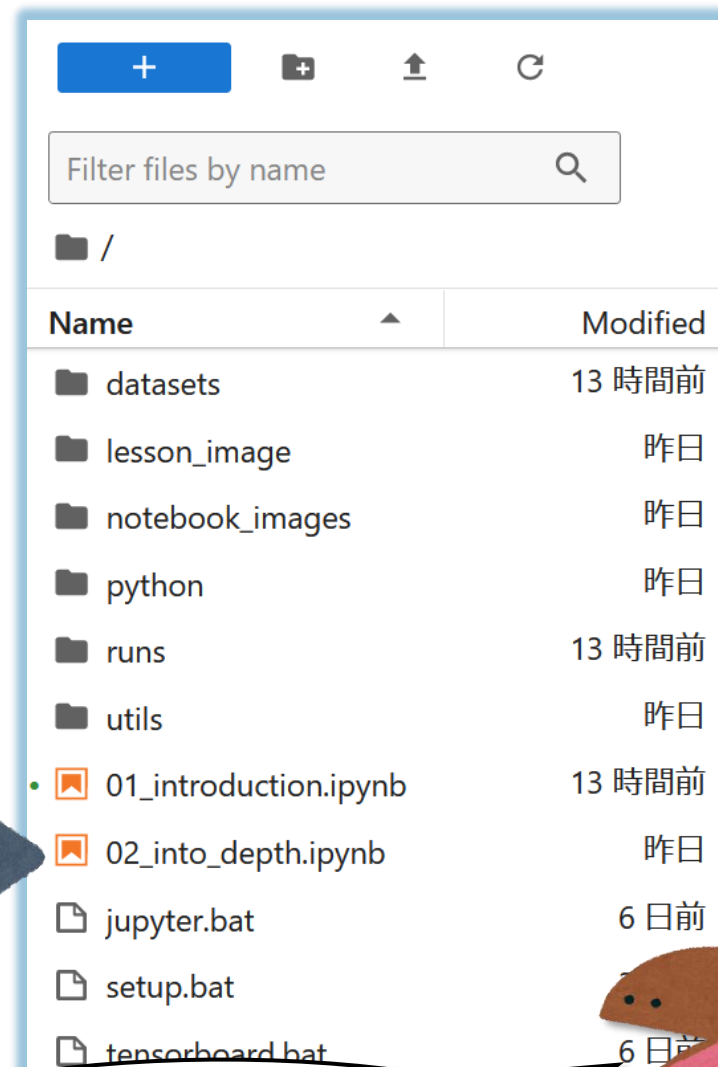
今回の学習目標

- ・ 画像の成り立ち
- ・ インプットとアウトプット
- ・ A I の訓練（難しい）
- ・ ニューラルネットワーク（難しい）

二時間目は手書き数字の分類 A I を作ります。

ノートブックから
02_into_depth.ipynb
を開きましょう。

ファイルエクスプローラー



さっきの一つ下にあるよ



画像データ確認

まずはコードを実行しましょう。

コードの下に画像データが表示されます。

見ての通り、0から9の手書きの数字があります。これが今回の訓練用のデータです。

① 今回の訓練データを確認しよう
- Let's check out the training data to

① コード実行

```
[1]: from torchvision import datasets, transforms
import torch.utils.data as data_utils
from matplotlib import pyplot as plt

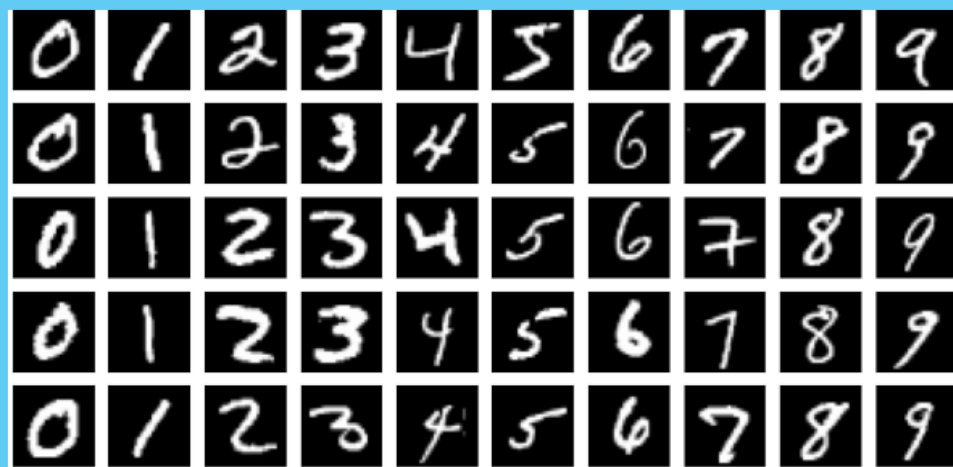
transform = transforms.Compose(
    [transforms.ToTensor(), transforms.Normalize((0.5,), (0.5,))]
)

train_dataset = datasets.MNIST(
    "datasets", download=True, train=True, transform=transform
)

test_dataset = datasets.MNIST(
    "datasets", download=True, train=False, transform=transform
)

fig, ax = plt.subplots(5, 10, figsize=(10, 5))
for x1 in range(5):
    for x2 in range(10):
        number_index = (train_dataset.targets == x2).nonzero(as_tuple=True)[0][x1]
        ax[x1][x2].imshow(train_dataset.data[number_index][0][0], cmap='gray')
plt.tight_layout()
```

② コード出力



画像の成り立ちについて

コードを実行すると、一つ一つの画像のピクセル数値が見えるようになります。

この数字の画像は28行28列の計784ピクセルで作られている。

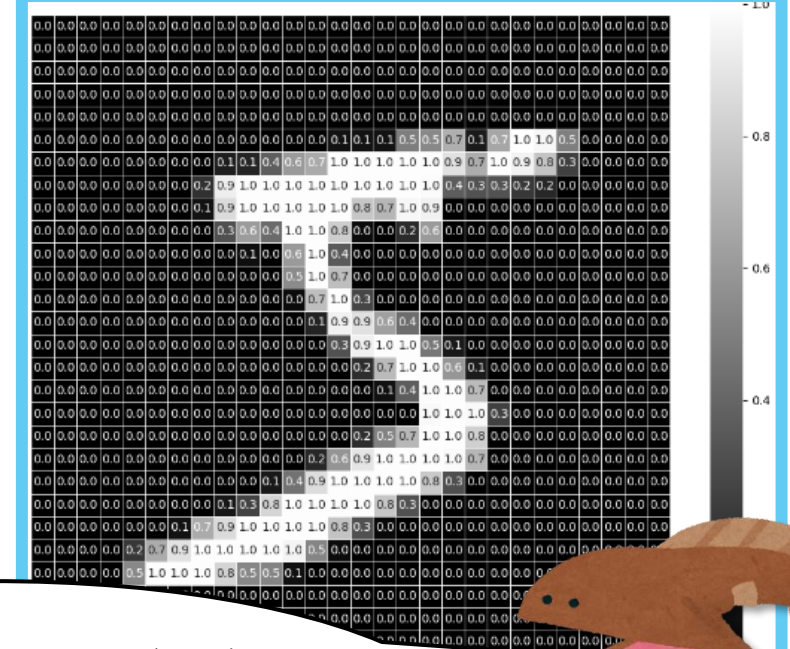
ピクセルは（AIで使う場合）0から1の数値を持つことが多いです。

```
[2]: import seaborn as sns
import numpy as np

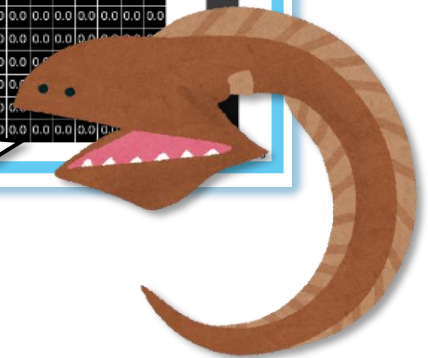
plt.figure(figsize=(12, 10))

sns.heatmap(
    (train_dataset[0][0][0].detach().numpy() + 1) / 2,
    linewidth=.5, cmap="gray",
    xticklabels=False, yticklabels=False,
    square=True, annot=True, fmt=".1f")
```

② コード出力



この画像は白黒だけど、色がある場合は $28 \times 28 \times 3$ になるんだよ



インプットとアウトプット 【クイズ】

選択問題

モデルのインプット（入力）
【1】 _____

インプット数値の数
【2】 _____

モデルのアウトプット（出力）
【3】 _____

アウトプット数値の数
【4】 _____

選択肢

【1 & 3】

A 数字の画像

B A | モデル

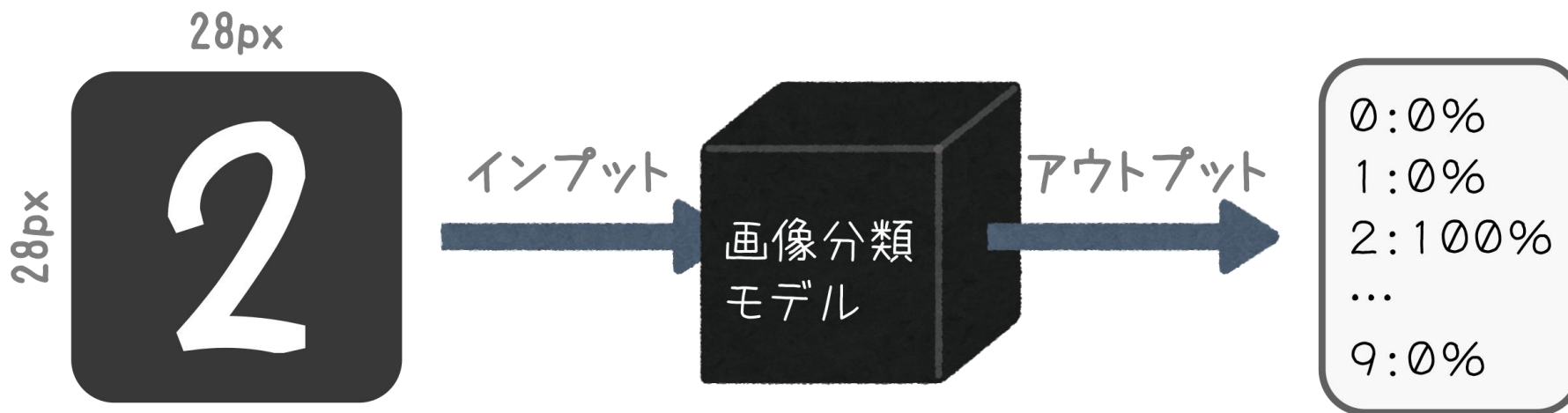
C 各数字の確率

【2 & 4】

A 1 0

B 7 8 4

C 2 8



手書き数字の分類 A I を作ってみよう

コードを実行する前に

STEP1 のコードの変数

input_size と output_size

に先ほど考えた数値の数を入力しましょう。

STEP2 のコードを実行しましょう。
一時間目の例と同じように、実行したらTensorBoard のグラフを見ていこう。

STEP 1

モデルのインプットとアウトプットの数を入力しよう

- Let's insert the number of inputs and outputs used by the model

コードの部分「# <<<」の左側に当てはまる数値を入力しよう

Type the numbers accordingly into the code

② STEP1 コード実行

```
[ ]: from torch import nn

input_size = # <<<
output_size = # <<<

model = nn.Sequential(
    nn.Flatten(),
    nn.Linear(input_size, 20),
    nn.ReLU(),
    nn.Linear(20, output_size),
    nn.Softmax(dim=1)
)
```

① 変数を入力

STEP 2

正確度が訓練を通して上がって

- Let's check if the model's accuracy is improving

③ STEP2 コード実行

```
[ ]: import torch.utils.data as data_utils
from utils import train

train(
    model,
    data_utils.Subset(train_dataset, range(10000)),
    data_utils.Subset(test_dataset, range(1000)),
    20
)
```

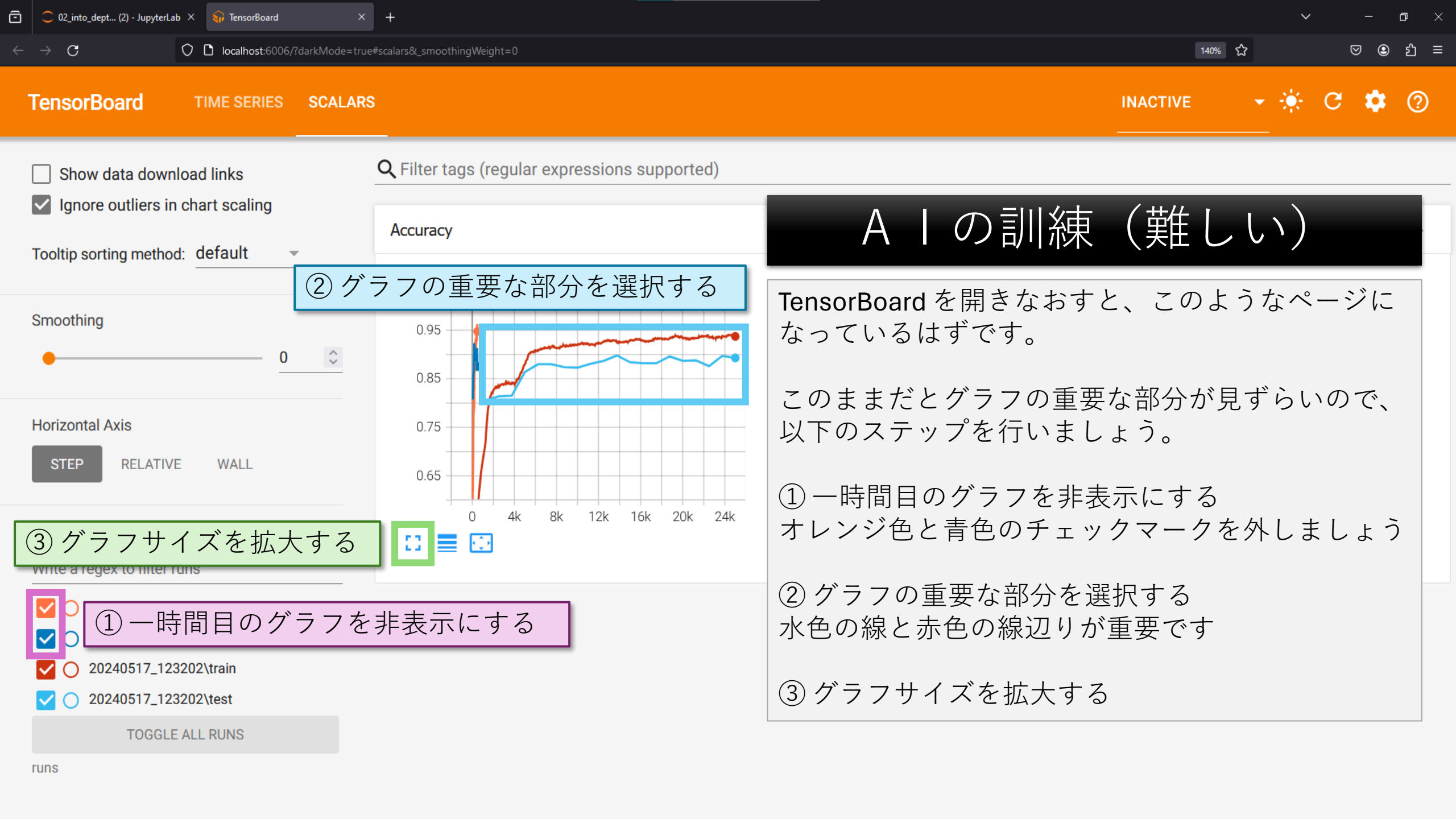
tensorboard.datを実行し、下のリンクを開いてください

正確度グラフを見てみよう

④ リンクを開きましょう

Run tensorboard.dat and open the link below.

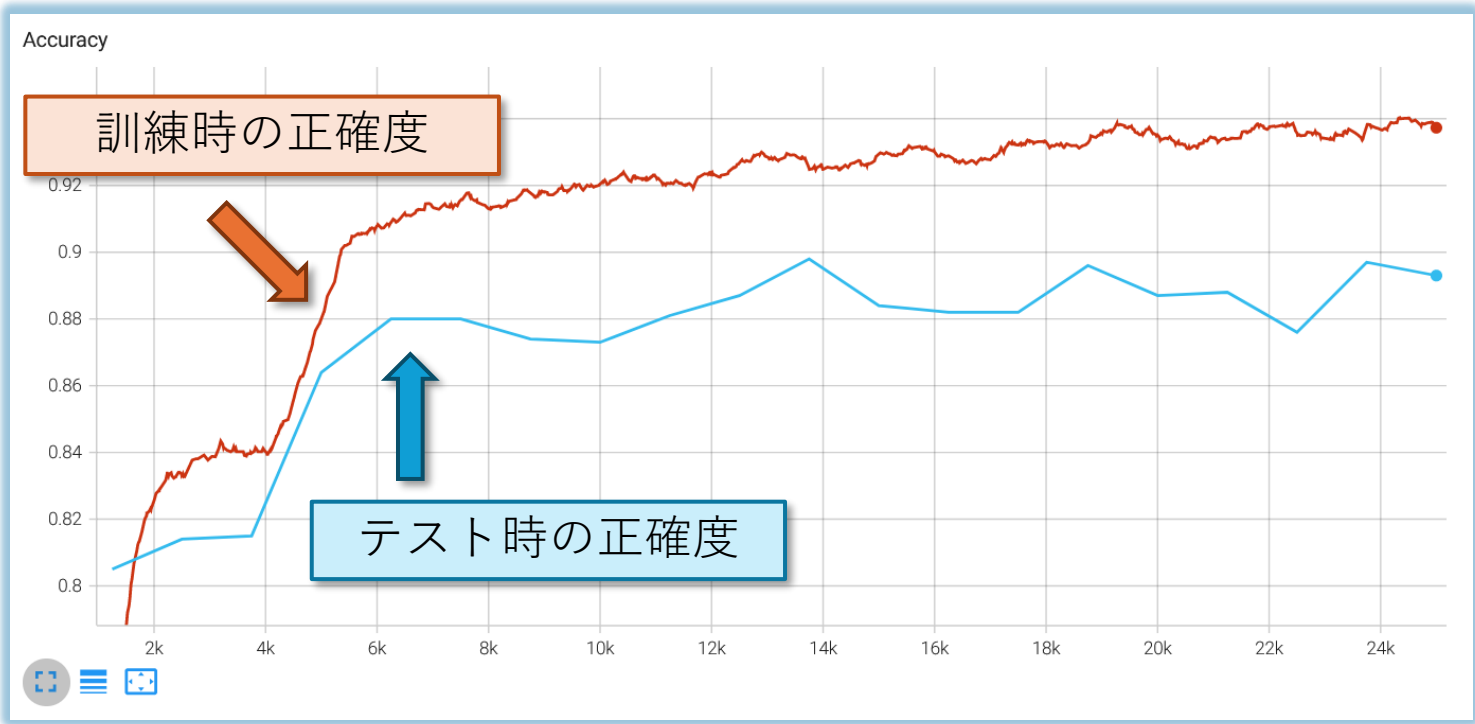
[Check out the training accuracy graph](#)



A I の訓練（難しい）

訓練データ・テストデータとは

A I の訓練には 2 つの段階があり、これを「訓練」と「テスト」と言う。この 2 つの段階では異なる画像データが使われている。



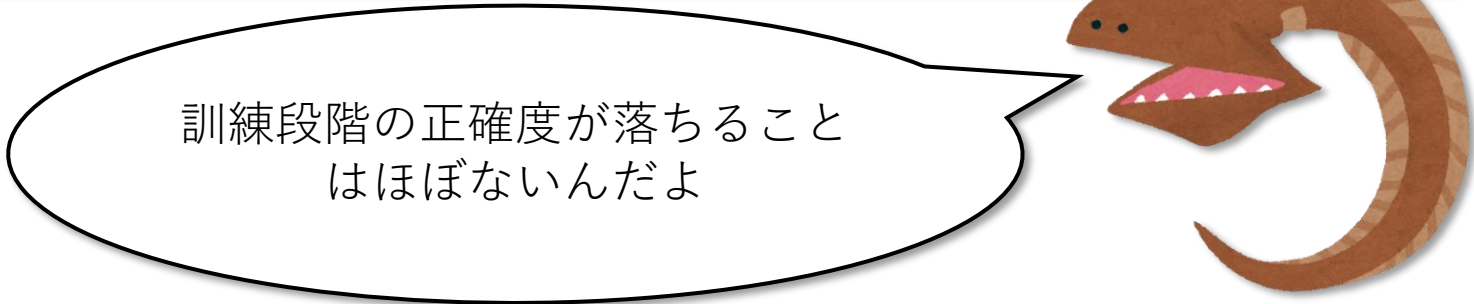
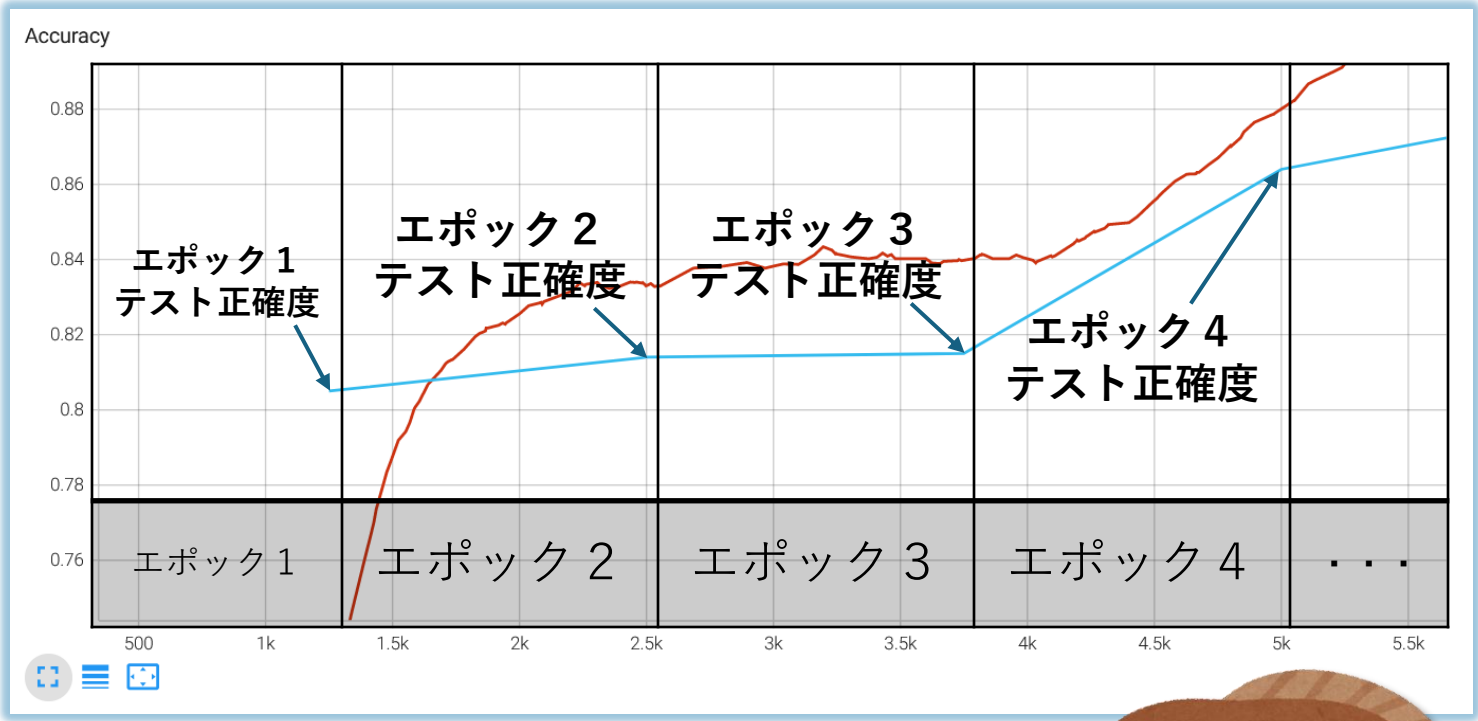
段階	データセット	行動	目的
訓練段階	訓練データはたくさん必要であり、今回は 1 0 0 0 0 枚の訓練データが用意されている。	モデルは訓練データに予測値をつけ、自分で答え合わせをする。回答と予測が異なった場合、間違いから学習する。	モデルの正確度を上げること
テスト段階	訓練データと比べて、テストデータは少なく、今回は 1 0 0 0 枚のテストデータが用意されている。	モデルはテストデータに予測値をつけるが、答えはモデルに明かさず、こちら側で正確度を記録する。	モデルの正確度を測ること

A I の訓練 (難しい)

エポックとは

A I の訓練ではこの 2 段階を何度も繰り返して正確率を上げていき、訓練段階とテスト段階の一巡を「エポック」という。

一定のエポックを満たすと訓練が終了する。



訓練段階の正確度が落ちることはほぼないんだよ

エポック 1		→	エポック 2		→	エポック 3		...
訓練段階	テスト段階		訓練段階	テスト段階		訓練段階	テスト段階	
正確度を上げる	正確度を測定		正確度を上げる	正確度を測定		正確度を上げる	正確度を測定	

A I の訓練（難しい）

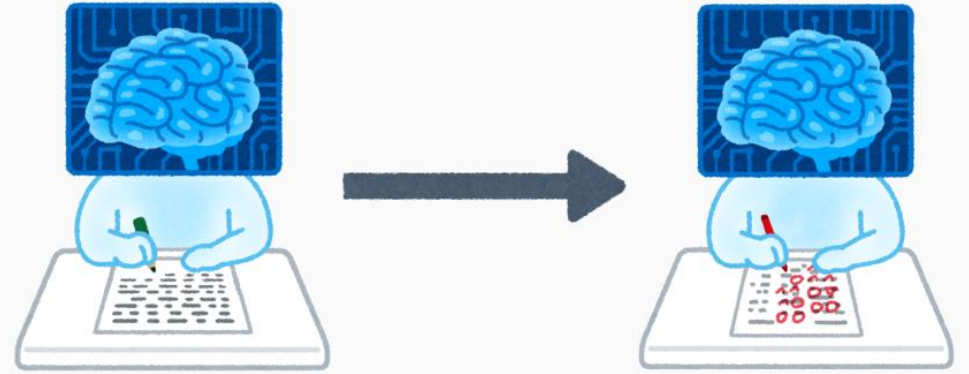
ではなぜ一段階で済ませられないのか？

もし正確度を確認める問題が訓練段階で出てきていたら、A I が問題を暗記していることに気が付かないからです。

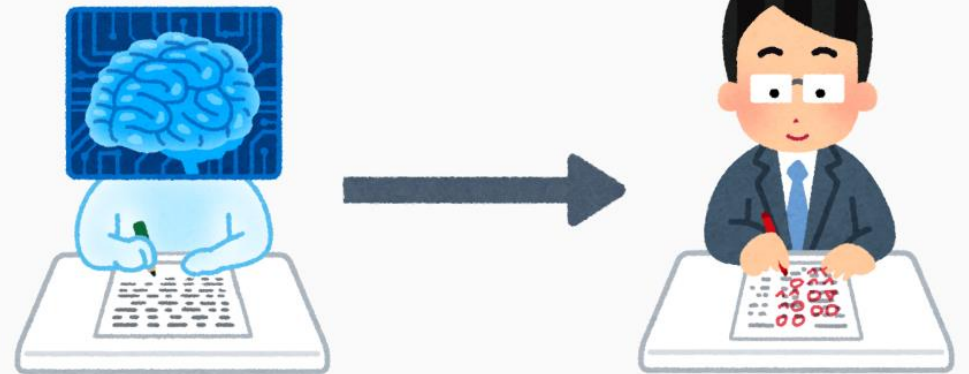
問題を暗記してしまった A I は見たことない問題に苦勞します。

A I が暗記することを「オーバーフィッティング」と言い、テスト段階でこの現象を監視することが重要です。

訓練段階



テスト段階



A I の訓練【クイズ】

選択問題

今回の A I の訓練は二段階に分かれており、「訓練」と「テスト」と名づけられている。

訓練用の画像データは多くの場合、テスト用の画像と比べて枚数が【1】____、訓練段階の目的は正確度の【2】____とされる。

その一方で、テストの段階では正確度の【3】____が目的だ。

そして、A I の訓練を二段階に分ける主な理由は、正確度を【3】するためには A I 【4】____データが必要だからだ。

選択肢

【1】

- A 少なく
- B 同じで
- C 多く

【2 & 3】

- A 安定
- B 向上
- C 測定

【4】

- A と合性がよい
- B が見たことない
- C が見たことある

画像分類 A I を作ってみよう

④自分の数字を当てさせよう

STEP4 のコードを実行しましょう。

すると、右下のようなウィンドウが出てくるはずです。黒いエリアに数字を書くとモデルが右側に予測結果を出してくれます。

うまく予測できたでしょうか？

▼ **STEP 4**
自分で数字を書いてモデルの機能を確認
- Try out the model with your own digits

① STEP4 コード実行

```
[*]: from utils import MnistApp  
  
app = MnistApp(model, transform)  
app.root.call('wm', 'attributes', '.', '-topmost', '1')  
app.root.mainloop()
```

② 数字を書いてみよう



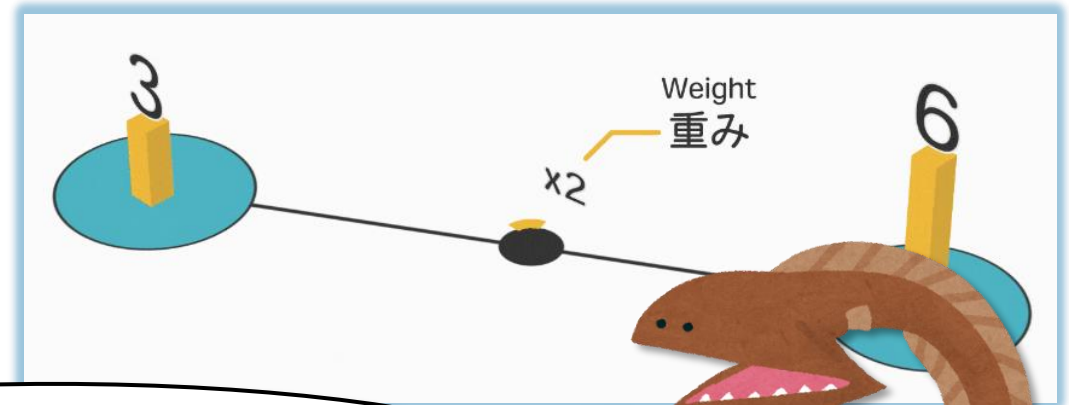
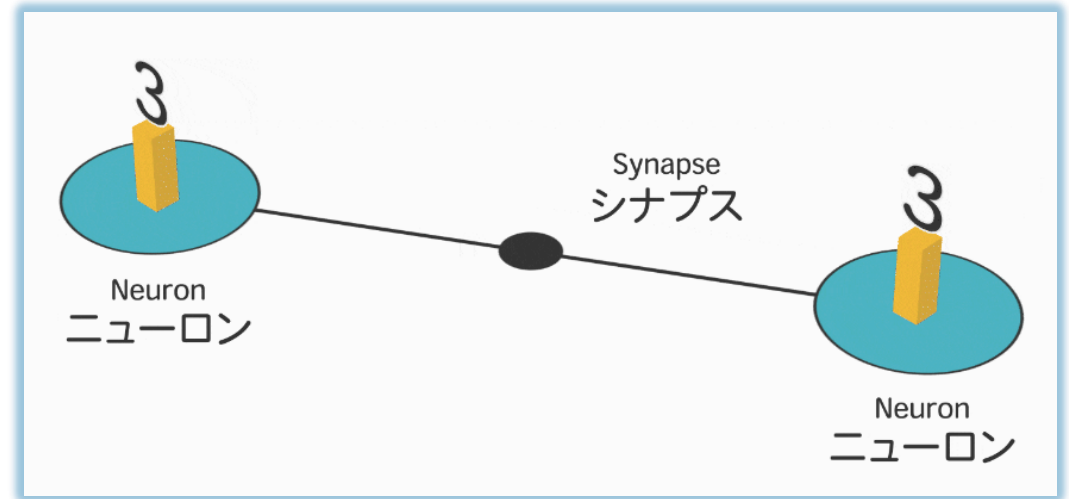
0:	0.0%
1:	0.0%
2:	99.6%
3:	0.0%
4:	0.0%
5:	0.0%
6:	0.0%
7:	0.0%
8:	0.4%
9:	0.0%

ニューラルネットワーク（難しい）

ニューラルネットワークは「ニューロン」と「シナプス」でできています。

ニューロン1が数値を受け取ったとき、シナプスを通してニューロン2へ数値を送り込む仕組みになっています。

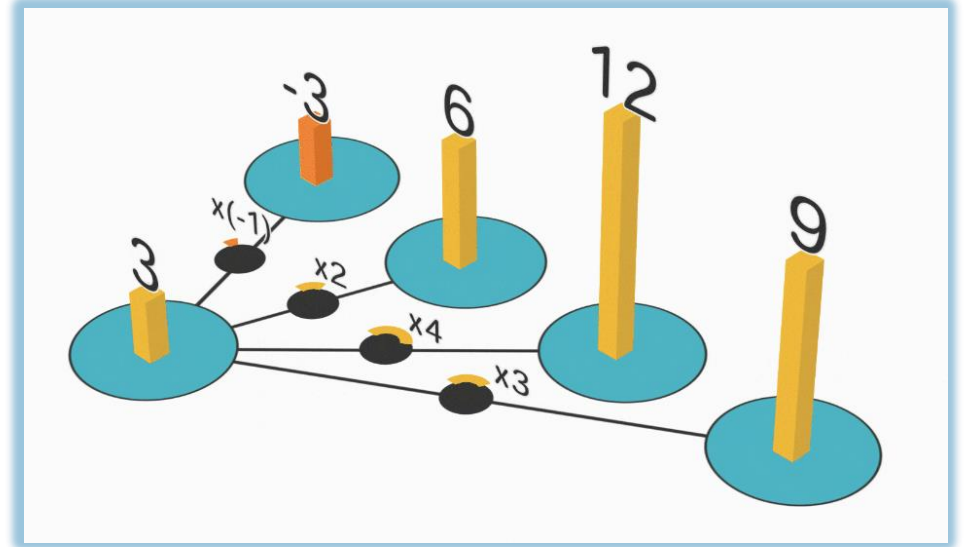
一つ一つのシナプスには「重み」があり、数値がシナプスを通るときに掛け算を行い、次のニューロンへ渡していく。



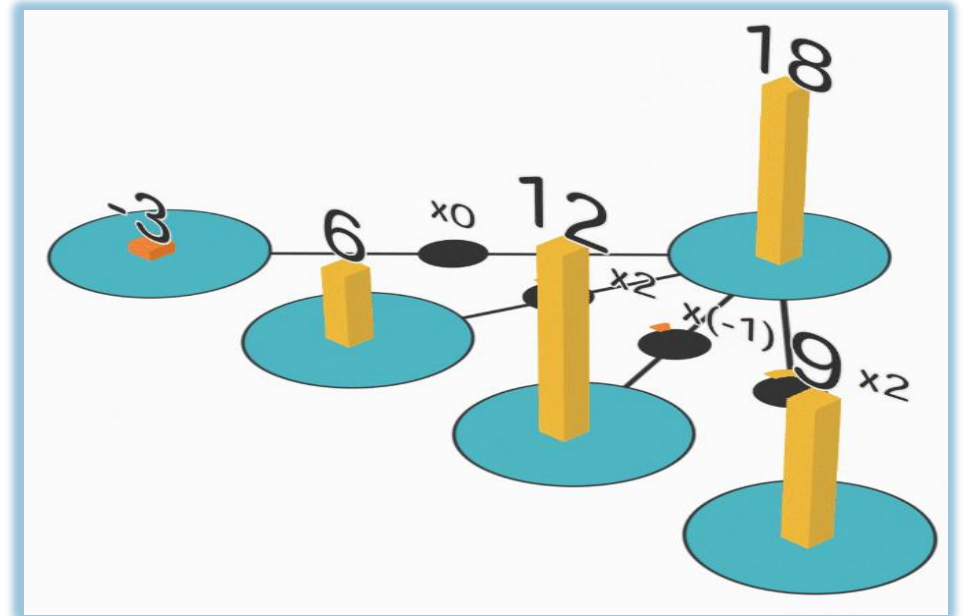
「シナプス」をそのまま「重み」と呼ぶことも多いよ

ニューラルネットワーク（難しい）

一つのニューロンが複数のニューロンに繋がる場合は、最初のニューロンの数値が複数の掛け算に使われる。

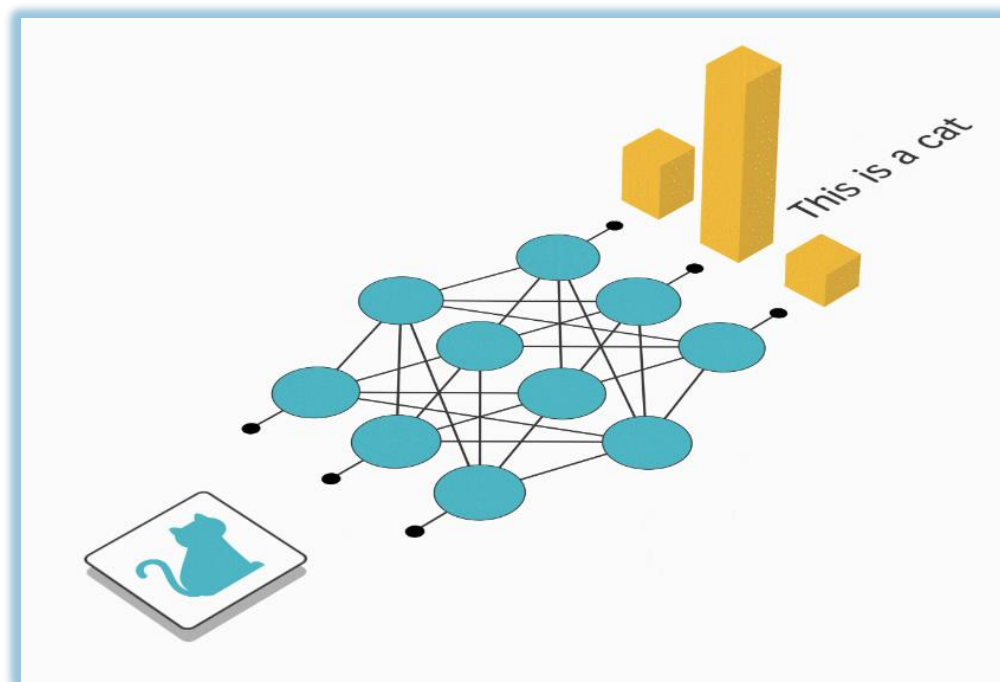


逆に複数のニューロンが一つのニューロンに繋がる場合、全ての計算の和をとる。

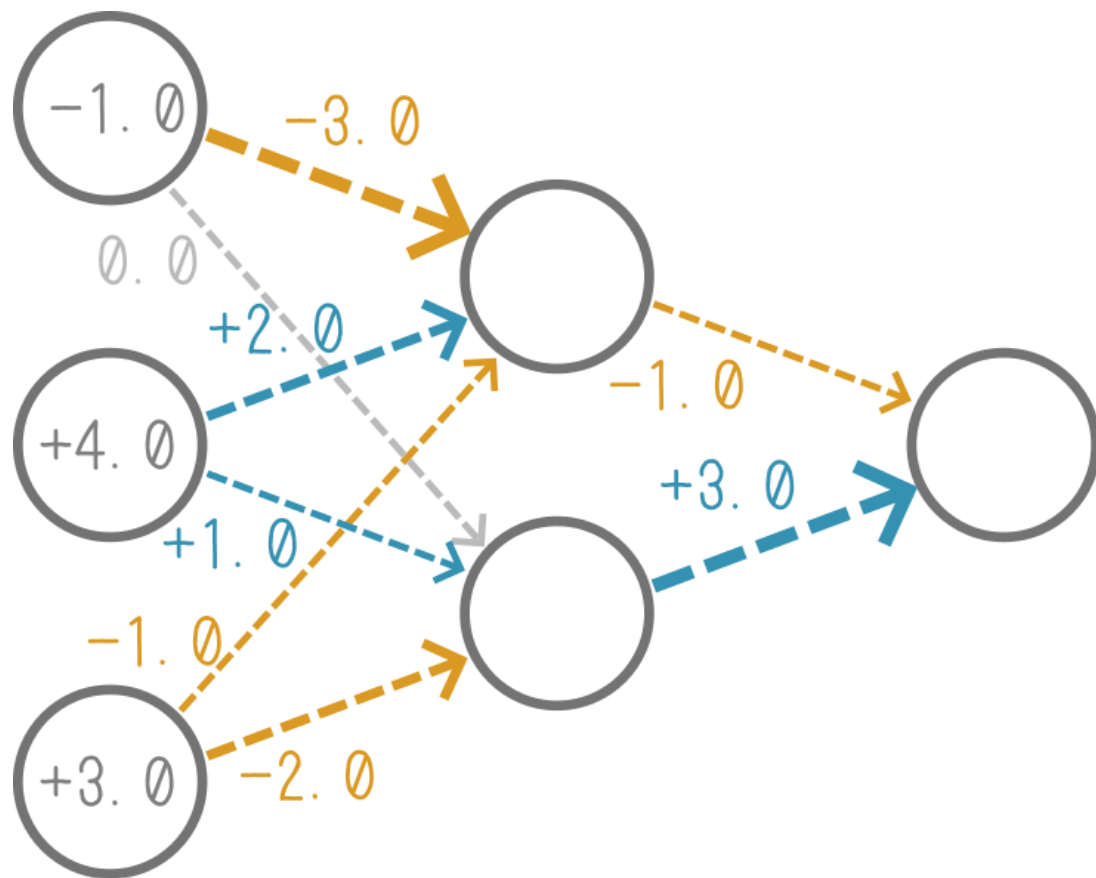


ニューラルネットワーク（難しい）

ニューロンが増えることによって右のようなつながり方ができ、複雑な計算ができるようになるのです。



ニューラルネットワーク【クイズ】



右には簡単なニューラルネットワークがあります。

三つの丸の中に入る数値を考えてみよう。

学校で習ったかな？
マイナス×マイナス
はプラスだよ



よく頑張りました！

終了

