

通信・計算機システムと 知的情報処理

(0/4)

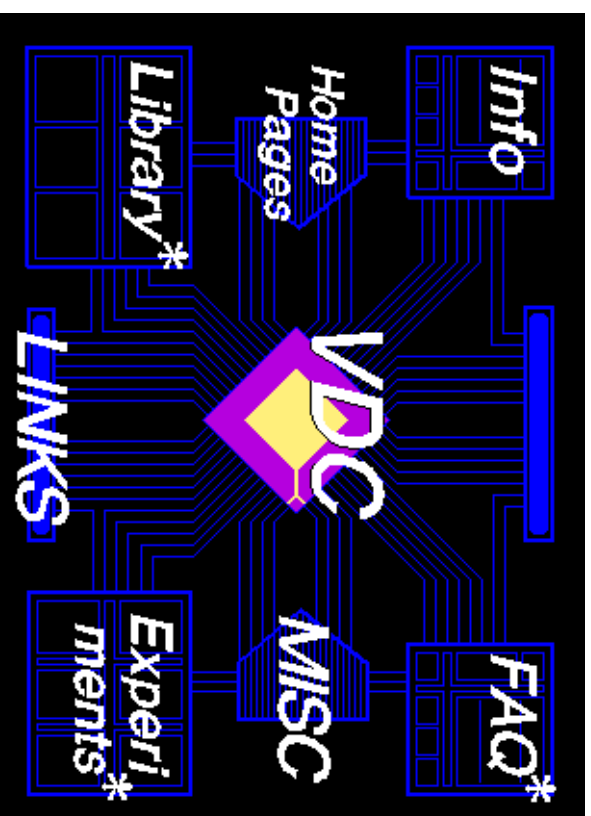
情報通信系

工学リテラシー

資料作成
篠崎、中原、上垣外
およびTAメンバー

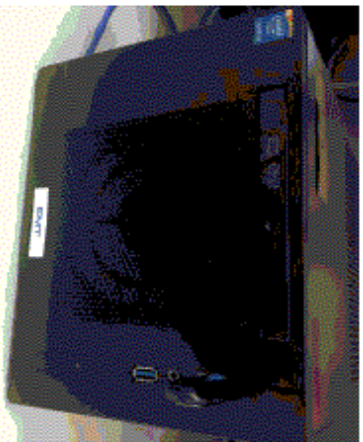
VLSI設計室の利用について

- 土足厳禁、下足入れ番号とスリッパ番号は一致
- 本講義の時間外も、他の講義の時間外は利用可
- 利用については、VLSI設計室のルールに従うこと
 - <http://www.vdc.ict.e.titech.ac.jp/index.html>



VLSI設計室の計算機の起動方法

- ・「工学リテラジ」で利用する計算機は机の下に本体
 - ・ 利用にはアカウントが必要
 - ・ 学生証を見せて、アカウント名とパスワードを受け取る
 - ・ Linux(ネットワークブート)を用いる
 - ・ いくつかの機種が混在



ASUSのPC



DELLのPC

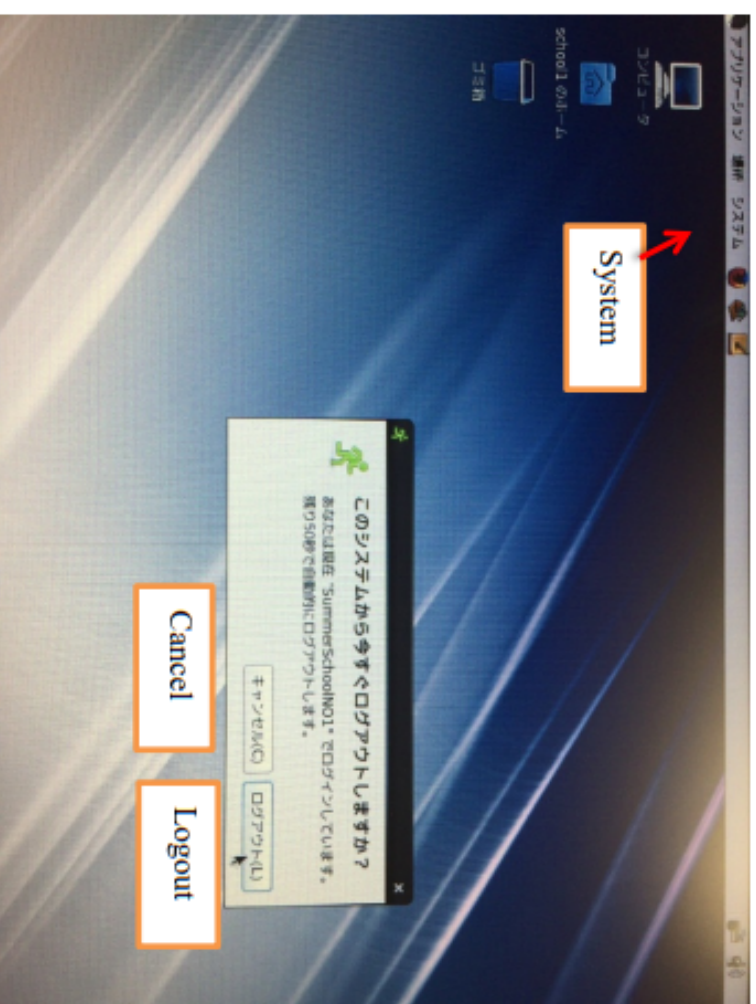


DELLのPC

- ・ 自分のノートパソコン等を使用してもよい
 - ・ キャンパス無線LANに接続し、東工大ポータルへのアクセスが必要
 - ・ win/mac/linuxいずれでもよい
 - ・ サポートはできないので自己責任で使用するこ

VLSI設計室の計算機の終了方法

1. 画面左上の「システム」を選択(左クリック)
2. 「ログアウト」を選択(左クリック)してログアウト
3. 電源ボタンを押して計算機をシャットダウン

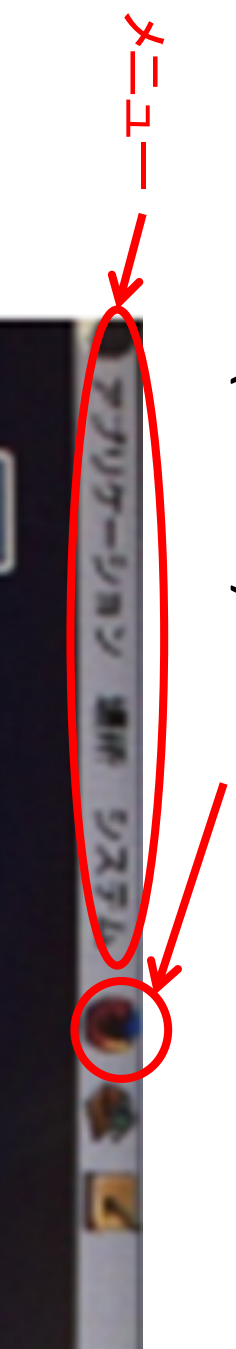


Logout: Click "System" at the top left of the screen, and then click "Logout"

ブラウザ，端末の起動方法

- ブラウザの起動方法

- 画面左上のメニューから選ぶ(左クリック)、または
- ブラウザ(Firefox)アイコンを左クリック



- Linux「端末」の起動方法

- 画面左上のメニューの中から選ぶ、または
- デスクトップ上で右クリック
 - 表示されるメニューから「端末」を選ぶ

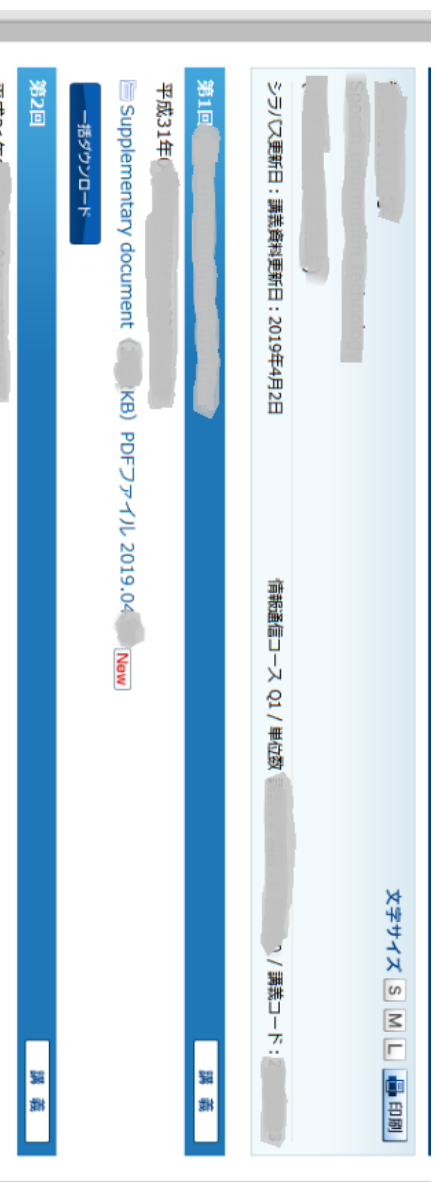
- PDFファイルの閲覧方法

- Evince (Document Viewer)を起動

VLSI設計室での講義資料の見方

- OCWi または OCM で本講義のページを開く
- OCM/OCWi から参照したい PDF をダウンロード

OCWi



OCM

開講元		工学院
担当教員名	各教員	青木 洋貴 高橋 篤司 篠崎 隆宏 中田 和秀 吉村 奈津江 中原 啓貴 上垣外 英剛
授業形態	講義 / 演習	
曜日・時限(講義室)	金5-6(<前半>, 南2号館3階VLSI設計室, <後半>, 西9号館3階311号室)	
クラス	e	
科目コード	XEG.8101	単位数 1
開講年度	H31年度	開講クォーター 1Q
シラバス更新日	H31年5月19日	講義資料更新日 H31年4月17日
使用言語	日本語	アクセスランキング ★★★★★

シラバス	講義ノート	ユーザーアンケート
------	-------	-----------

第1回 ニューラルネットの原理

★★★★★ 添付資料(2715KB)

OCMの提出方法の説明を加えました。

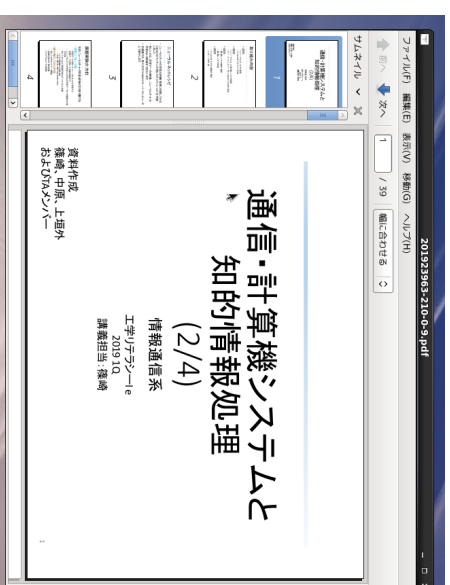
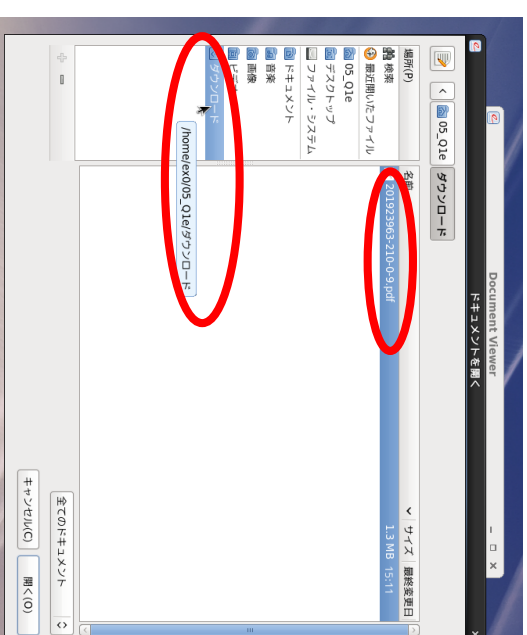
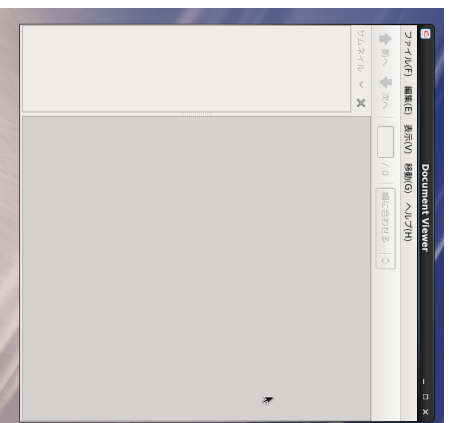
ニューラルネットの学習と演習課題にある微分の関係の説明を加えました。

平成31年04月12日(金) 5-6時限開講

講義

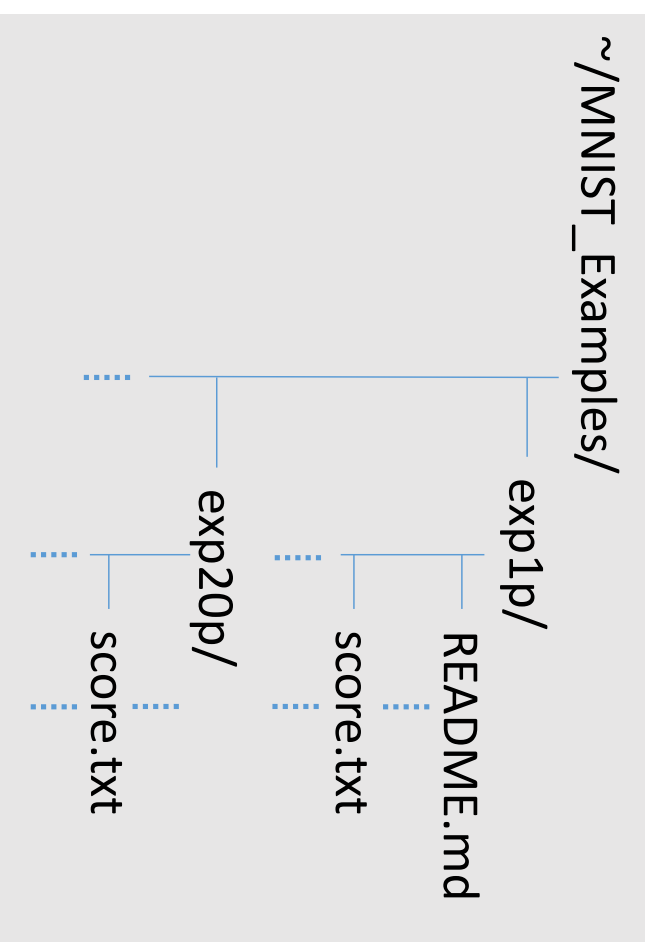
Evince (Document Viewer)の起動

- ① 「端末」で`evince`と入力、`Enter`キーを押す
- ② Evince (Document Viewer)が起動
- ③ ファイル(PDF)を選択
- ④ ファイル(PDF)が開く



Linux (Unix系OSカーネル)

- コマンド(命令)を入力
- 階層型ディレクトリ構造(ディレクトリ=フォルダ)
 - ルートディレクトリ **/**
 - ホームディレクトリ **~/**
 - 最初の作業ディレクトリ
 - カレントディレクトリ **./**
 - 現在の作業ディレクトリ **../**
- ディレクトリの指定
 - ルートからの絶対パス
 - カレントディレクトリからの相対パス
- プロンプト(コマンドの入力位置を示す)
 - カレントディレクトリの表示
 - 設定によって表示は異なる



```
19M1 [ ] 1@login1:~> cd MNIST_Examples/exp1p
19M1 [ ] 1@login1:~/MNIST_Examples/exp1p> ls
README.md      mnist.model      runtsubame.sh
eval.log       model            runtsubame.sh.e4716010
eval.py        prep.py          runtsubame.sh.o4716010
eval.sh        prep.sh          score.txt
mkgraph.py     requirement.txt  train.log
mkgraph.sh     run.sh           train.py
19M1 [ ] 1@login1:~/MNIST_Examples/exp1p>
```

カレントディレクト



Linuxの基本的なコマンド1

- `ls` (ディレクトリ名、ファイル名の表示)(**list**)
 - `$ ls` : カレントディレクトリの表示
 - `$ ls ~/` : ホームディレクトリの表示
 - `$ ls ..` : 親ディレクトリの表示
 - `$ ls path` : `path`で指定のディレクトリの表示
 - ✓ `path`はルートからの絶対パス、または、カレントディレクトリからの相対パス
- `echo` (表示)
 - `$ echo text` : `text`を表示
- `pwd` (カレントディレクトリのパス表示) (**print working directory**)
 - `$ pwd`
- `cd` (カレントディレクトリの変更)(**change directory**)
 - `$ cd ~/` : ホームディレクトリへ
 - `$ cd ..` : 親ディレクトリへ
 - `$ cd path` : `path`で指定のディレクトリへ
- `mkdir` (ディレクトリの作成)(**make directory**)
 - `$ mkdir dirname` : `dirname`の名前のディレクトリ作成
 - ✓ `dirname`で作成するディレクトリ位置も指定可能

スペース(空白)は「`_`」で表示
・ 引数、オプションの区切り

Linuxの基本的なコマンド2

- cat (ファイルの連結、中身の表示) (concatenate)
\$ **cat** **filename** : **filename**のファイルの中身の表示
✓ **filename**でファイルのディレクトリ指定も可能
- less (ファイルの中身の表示)
\$ **less** **filename** : **filename**のファイルの中身の表示
✓ **"q"**を押す終了
- cp (ファイルのコピー) (**copy**)
- mv (ファイルの移動) (**move**)
\$ **cp** **source** **destination** : **source**ファイルを**destination**にコピー
\$ **mv** **source** **destination** : **source**ファイルを**destination**に移動
✓ カレントディレクトリ内のファイルの場合、ディレクトリ(へのパス)は省略可
✓ ファイル名を変更しない場合、**destination**でファイル名は省略可
- リモート計算機の利用
 - ssh (リモート計算機へログイン) (**secure shell**)
\$ **ssh** **ID@login.t3.gsic.titech.ac.jp** : TSUBAMEへのログイン
 - scp (リモート計算機からファイルをコピー) (**secure copy**)
\$ **scp** **ID@machine:source** **destination** : リモートマシンの**source**ファイルを**destination**にコピー
 - ssh-keygen (秘密鍵公開鍵の生成)
\$ **ssh-keygen** **-t** **rsa**

Shell(コマンドインタプリタ)の機能

- 「Tab」キー
 - コマンドやファイル名の補完
 - 一意に定まるところまで名前を補完してくれる
- 「矢印」キー
 - コマンド履歴の表示、編集
 - ↑: 前の入カコマンド
 - ↓: 次の入カコマンド
 - ←、→: カーソル(編集位置)移動
- 文字の削除
 - 「Del」 : カーソル位置の文字
 - 「Backspace」: カーソル位置の前の文字



TSUBAMEで用いるコマンド、スクリプト

- git (分散型バージョン管理システム)
`$ git clone URL`
- qsub (TSUBAMEへのジョブの提出(queue submit))
`$ qsub script-file-name.sh`
`$ qsub -g tga-egliteracy -l h_rt=time runtsubame.sh`
- qstat (ジョブの状態(queue status))
`$ qstat`
- qdel (TSUBAMEからのジョブの削除(queue delete))
`$ qdel job-ID`

✓ 講義で用意したスクリプト

- mkgraph.sh (折れ線グラフの作成)

```
$ ./mkgraph.sh 'Xlabel' 'X列' 'Ylabel' 'Y列' '出力ファイル名'
```

日本語キーボード

shift-7 : シンブルクオート「」
shift-@ : バッククオート「`」



TSUBAMEでのエディタ (nano)の起動

- 課金グループ(tga-egliteracy)参加後に利用可能
- パスの事前指定
 - 方法1 (設定ファイルで指定. 一度行えば, 次回以降は不要)
`$ cp ~/.bashrc ~/.dot.bashrc.bak`
`$ cp /gs/hs0/tga-egliteracy/etc/dot.bashrc ~/.bashrc`
 - 方法2 (ログアウトまで有効. ログイン後毎回行う)
`$ alias nano=/gs/hs0/tga-egliteracy/local/bin/nano`
- 実行方法
`$ nano`
- パスの事前指定なしで実行 (パスを直接指定)
`$ /gs/hs0/tga-egliteracy/local/bin/nano`

注意: VLSI設計室のアカウメントについて

- 本クォーターの終了後、本講義で使用了たVLSI設計室のアカウメントおよびユーザーデイルクトリは使用できなくなります
- 本クォーター終了後も必要なデータがある場合は、各自でバックアップしてください



Google Colaboratory

Google Colab(Colaboratory)とは

- 機械学習の普及を目的としたGoogleのサービス
- クラウドで提供されるPythonの実行環境
- 環境構築が不要
- 誰でも無料で利用可能
- プログラムを簡単に共有可能
- Googleアカウントが必要

Google Colabの使い方(目次)



新規作成する場合

(Googleにログインした状態で)

1. Colab(<https://colab.research.google.com/>)にアクセス
 2. 新しいノートブックを押す
 3. Pythonプログラムを書いて、実行()を押す
- ファイルをGithubから読み込む場合

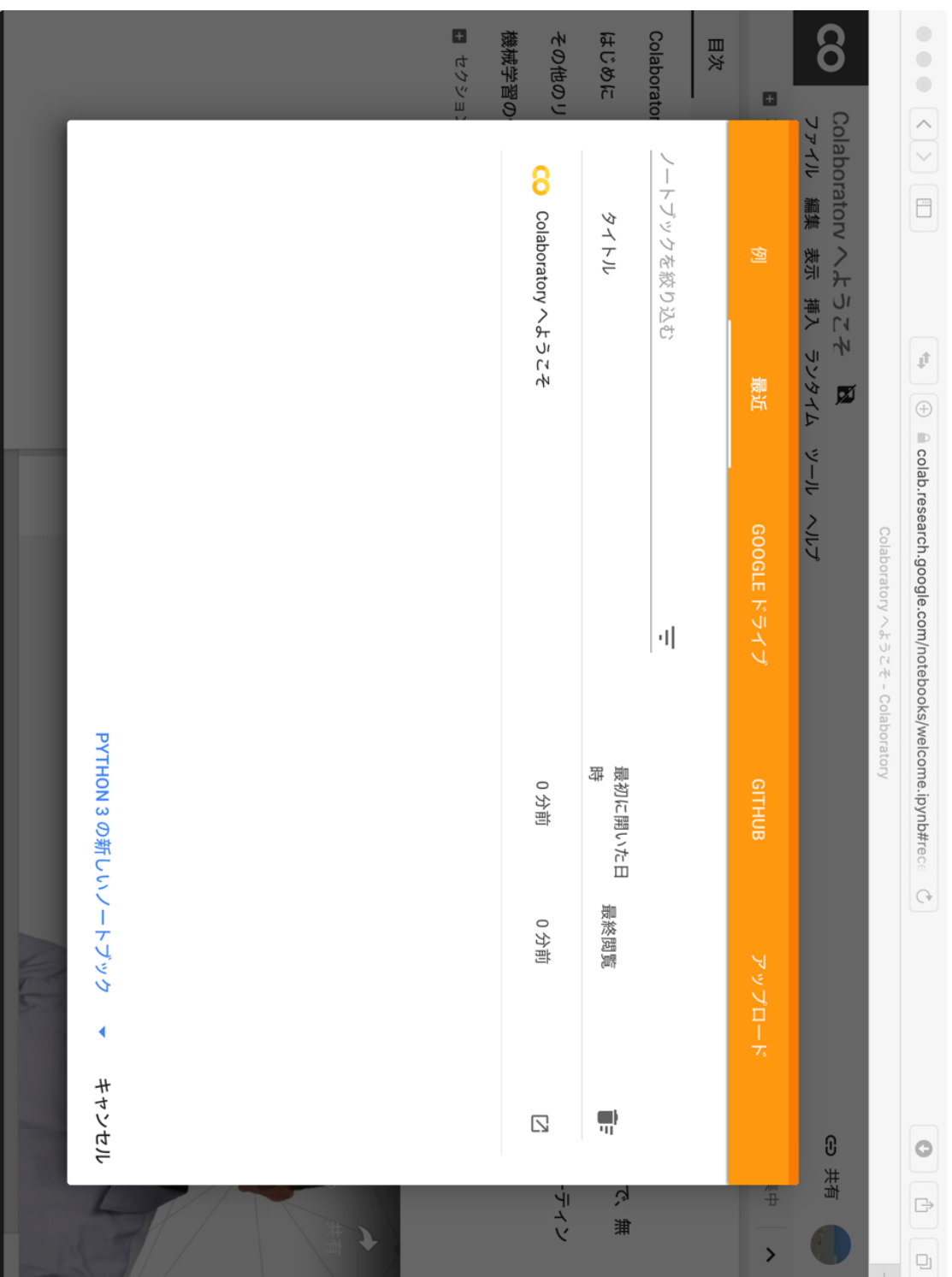
(Googleにログインした状態で)

1. Colab(<https://colab.research.google.com/>)にアクセス
2. GithubのレポジトリのURLを入力
3. 読み込むipynbファイルを選択
4. 順番に実行()を押す



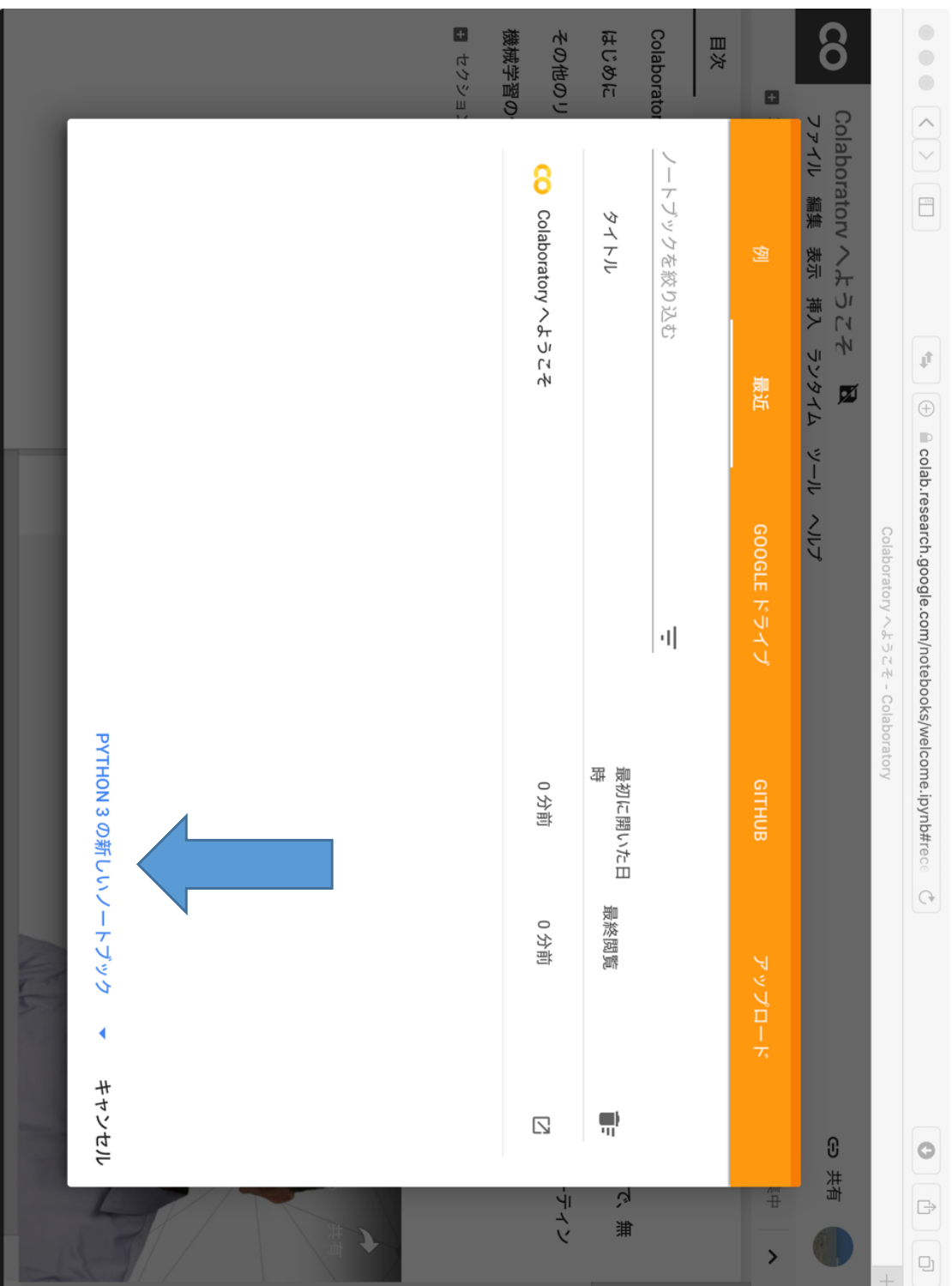
[新規作成] Colabにアクセス

<https://colab.research.google.com/> にアクセス



[新規作成] 新しいノートブック

新しいノートブックを選択



[新規作成] プログラムの実行



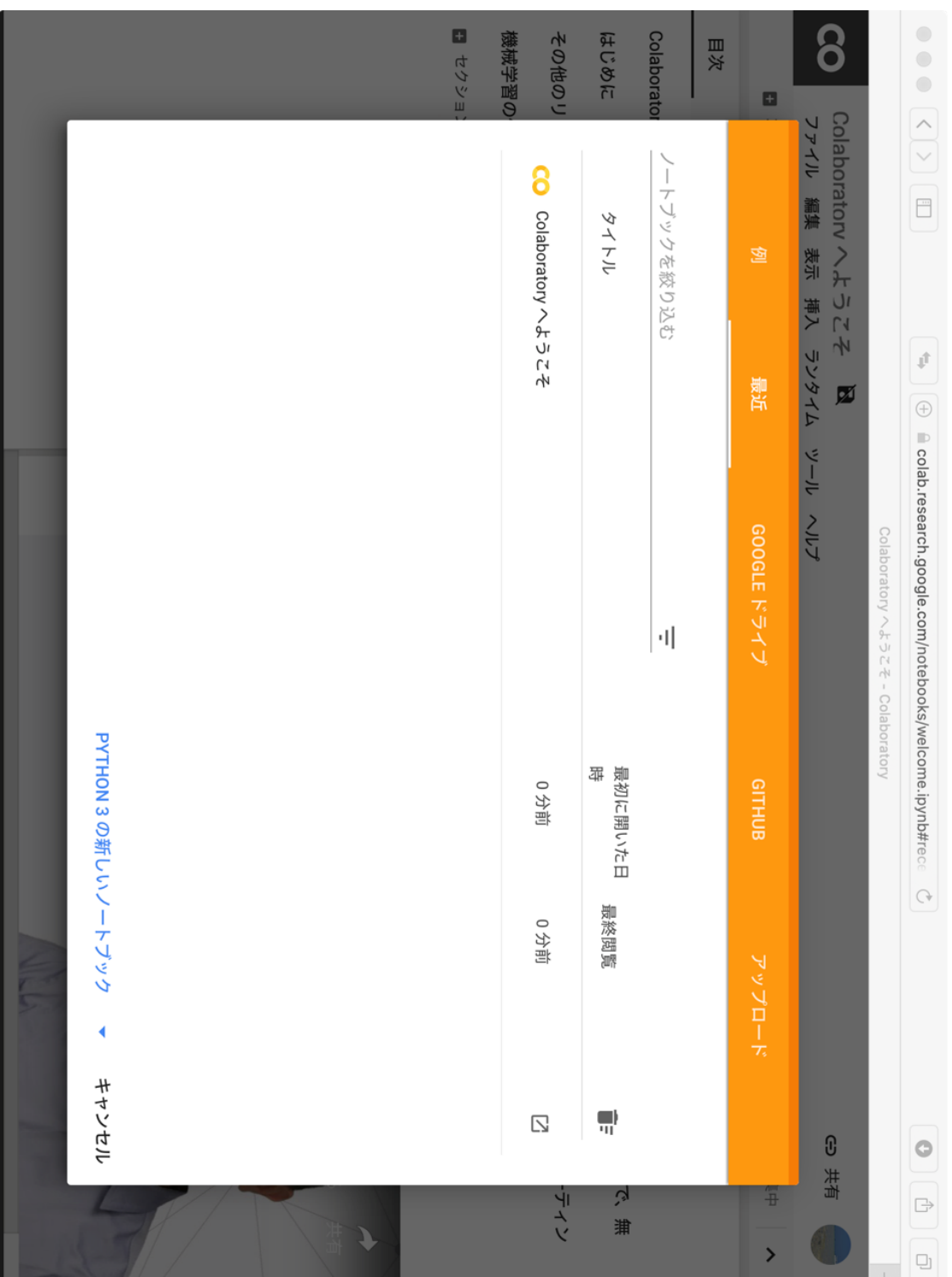
プログラムを書いて、実行()を押す



[レシピ実行] Githubから開く



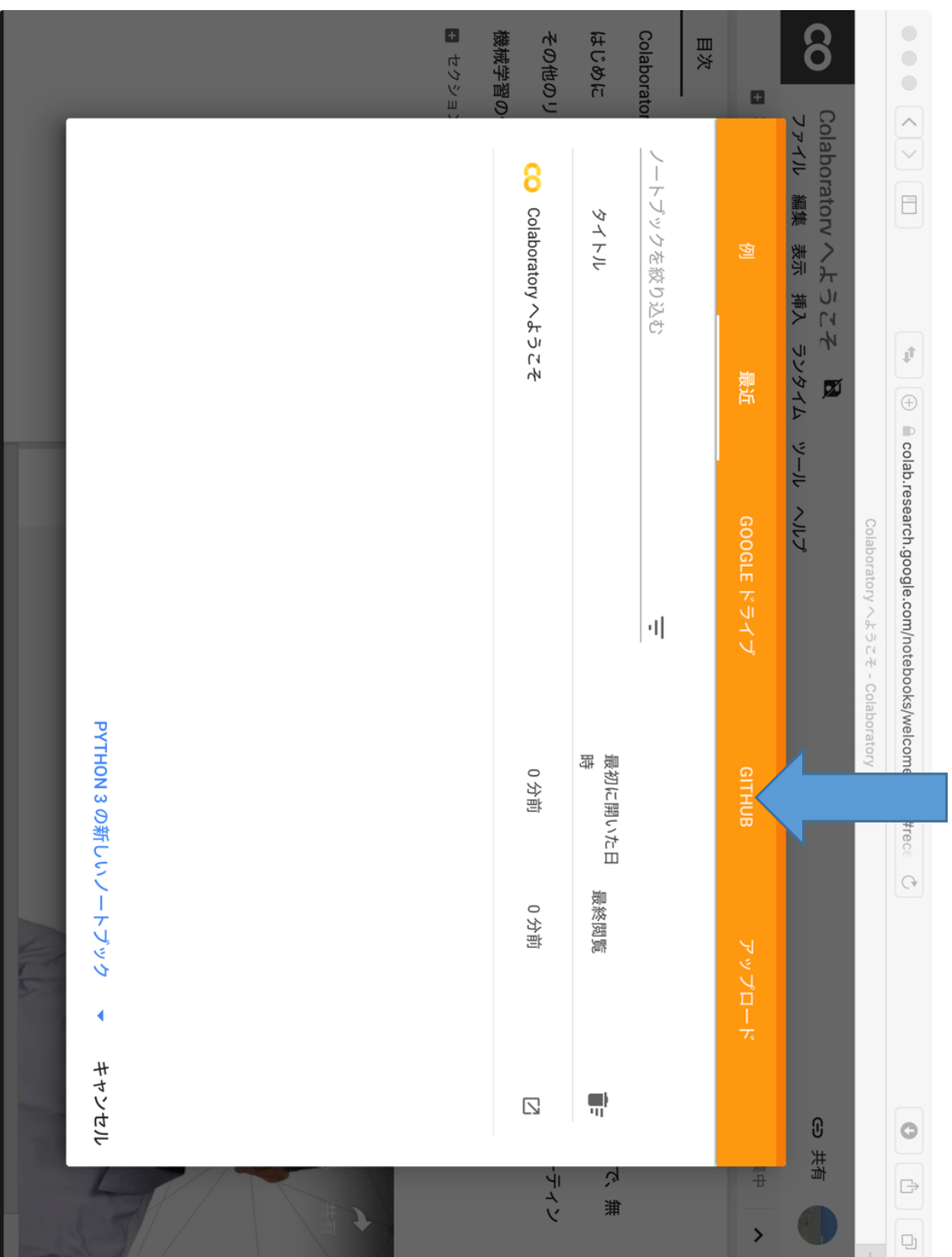
<https://colab.research.google.com/> にアクセス



[レシピ実行] Githubから開く



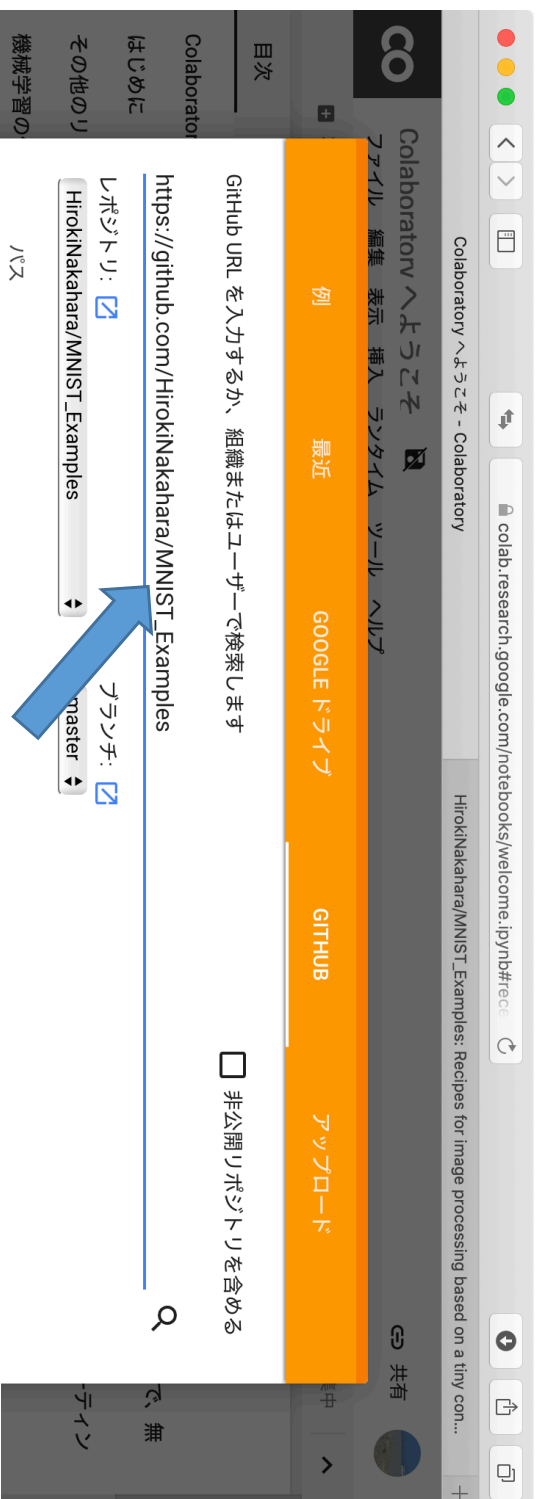
GITHUBを選択



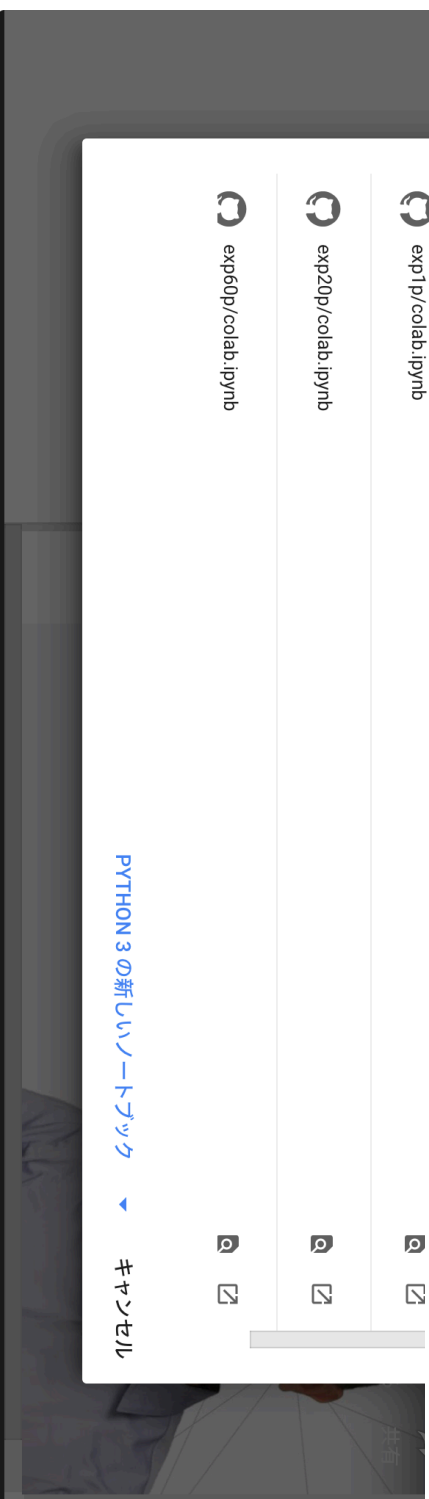
[レシピ実行] Githubから開く



GithubのURLを入力してファイルを選択

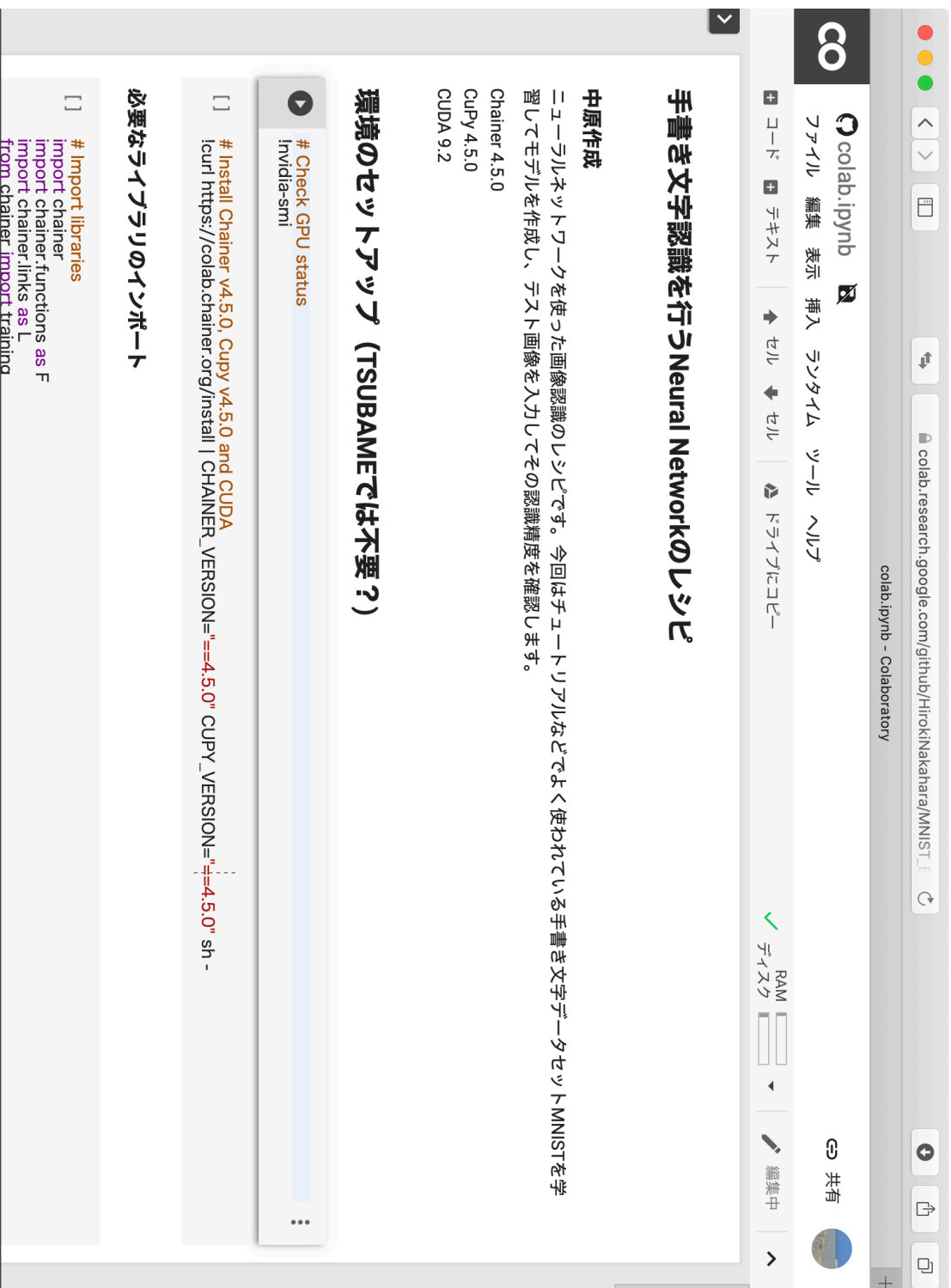


この例では画像認識の課題ファイルを開く
他の課題についても同様



[レシピ実行] プログラムの実行

順番に実行()を押す



The screenshot shows the Google Colaboratory web interface. The top bar includes the Colab logo, a file explorer, and a toolbar with buttons for code, text, cells, and a share button. The main area displays a Jupyter Notebook with a single code cell. The code cell contains the following text:

```
[ ] # Import libraries
import chainer
import chainer.functions as F
import chainer.links as L
from chainer import training
```

Below the code cell, the output of the execution is shown, indicating that the environment is set up correctly. The output includes the following text:

```
[ ] # Install Chainer v4.5.0, Cupy v4.5.0 and CUDA
!curl https://colab.chainer.org/install | CHAINER_VERSION=="4.5.0" CUPY_VERSION=="4.5.0" sh -
```

The bottom status bar shows the RAM usage (100%) and the device (CPU).

[レシピ実行]プログラムの実行



初回は警告が出るので「このまま実行」を押す

The screenshot shows the Google Colab interface. A warning dialog box is displayed in the center, with the text: "警告: このノートブックは Google が作成したものではありません。このノートブックは [GitHub](\"#\") から読み込まれています。Google に保存されているデータへのアクセスが求められたり、他のセッションからデータや認証情報が読み取られたりする場合があります。このノートブックを実行する前にソースコードをご確認ください。このノートブックが他のセッションから状態を読み取ることができないように、ランタイムをすべてリセットできます。"

Below the warning text, there are two buttons: "実行前にランタイムをすべてリセットする" (Reset runtime before execution) and "このまま実行" (Run as is). A blue arrow points from the "このまま実行" button to the "キャンセル" (Cancel) button, which is highlighted with a red border.

The background shows the Colab interface with the following elements:

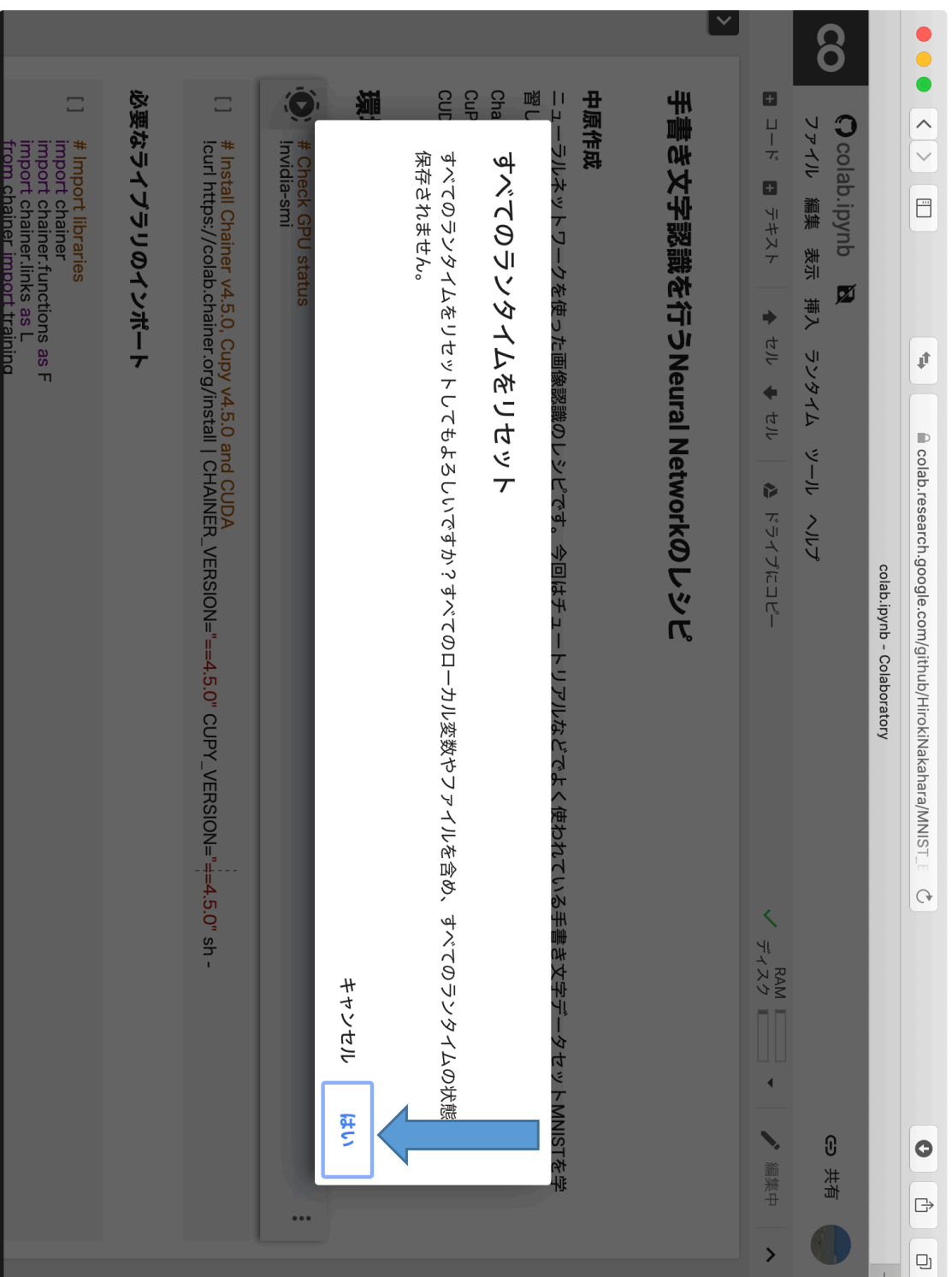
- Top bar: colab.ipynb, ファイル (File), 編集 (Edit), 表示 (View), 挿入 (Insert), ランタイム (Runtime), ツール (Tools), ヘルプ (Help).
- Left sidebar: コード (Code), テキスト (Text), セル (Cell), セル (Cell), フライアウト (Flyout).
- Bottom bar: 接続 (Connect), 編集 (Edit), 共有 (Share).
- Code editor: A code cell containing the following text:

```
[ ] # Import libraries
import chainer
import chainer.functions as F
import chainer.links as L
from chainer import training
```
- Environment section: 中元作成 (Original creation), ニューラルネットワーク (Neural network), 習してモデルを作成 (Learn to create model), Chainer 4.5.0, CuPy 4.5.0, CUDA 9.2.
- Environment settings section: 環境のセット (Set environment), # Check GPU (Check GPU), # Check GPU (Check GPU), # Check GPU (Check GPU).
- Warning dialog box: 警告: このノートブックは Google が作成したものではありません。このノートブックは [GitHub](\"#\") から読み込まれています。Google に保存されているデータへのアクセスが求められたり、他のセッションからデータや認証情報が読み取られたりする場合があります。このノートブックを実行する前にソースコードをご確認ください。このノートブックが他のセッションから状態を読み取ることができないように、ランタイムをすべてリセットできます。
- Buttons: 実行前にランタイムをすべてリセットする (Reset runtime before execution), このまま実行 (Run as is), キャンセル (Cancel).

[レシピ実行] プログラムの実行



同様に「はい」を押す



[レシビ実行]プログラムの実行



プログラムの出力が表示される

colab.ipynb

ファイル 編集 表示 挿入 ランタイム ツール ヘルプ

コード テキスト セル セル ドライバにコピー

RAM デイスク 編集

共有

手書き文字認識を行うNeural Networkのレシビ

中原作成

ニューラルネットワークを使った画像認識のレシビです。今回はチュートリアルなどでよく使われている手書き文字データセットMNISTを学習してモデルを作成し、テスト画像を入力してその認識精度を確認します。

Chainer 4.5.0
CuPy 4.5.0
CUDA 9.2

環境のセットアップ (TSUBAME)

```
# Check GPU status
!nvidia-smi
```

Mon Jul 1 06:18:51 2019

NVIDIA-SMI 410.38.06 Driver Version: 410.79 CUDA Version: 10.0	
GPU Name	Perseus
Fan Temp	Pwr:Usage/
=====	
0 Tesla T4	Of
N/A 65C P8 17W / 7	

プログラム入力欄

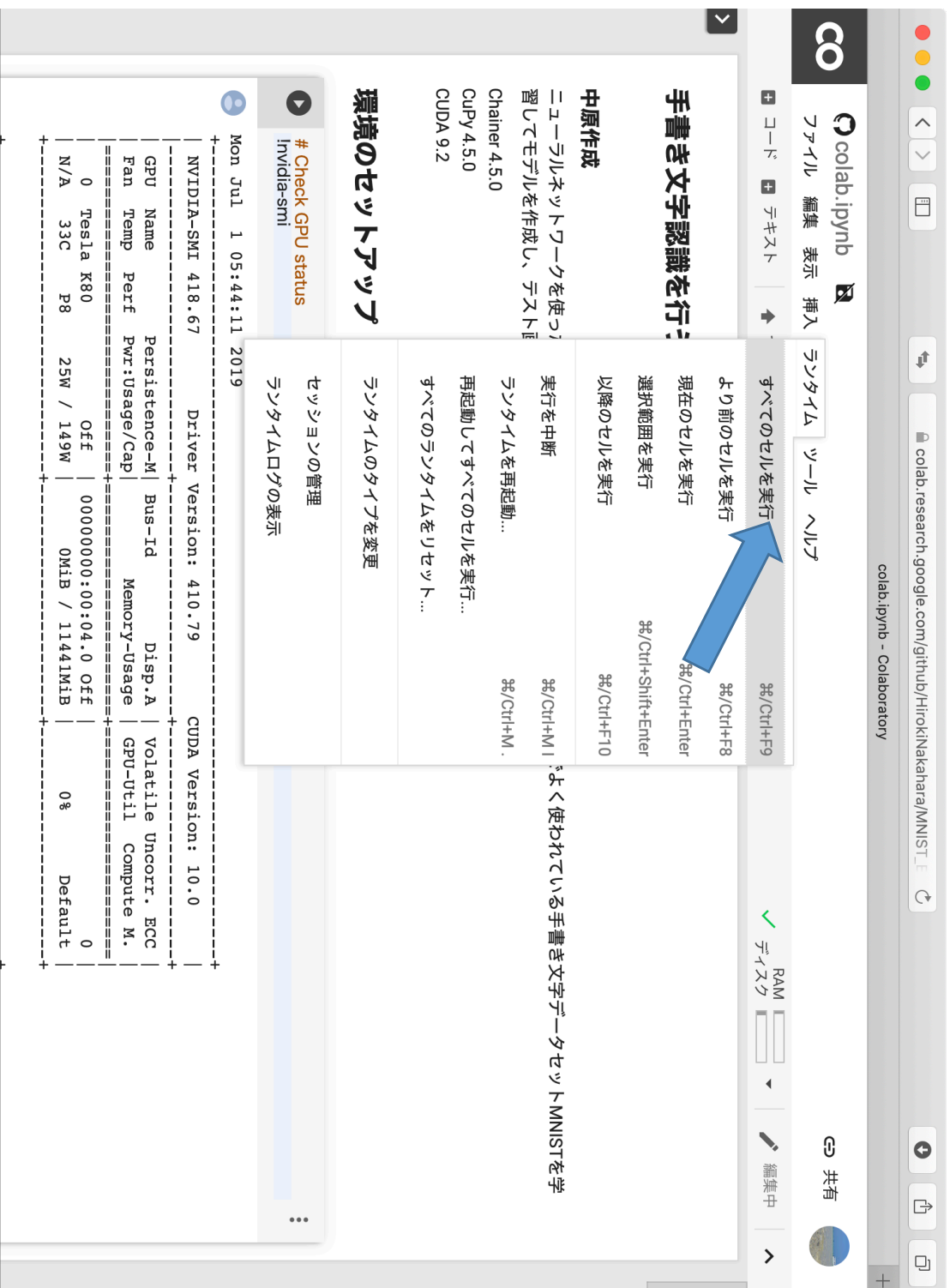
このプログラムはGPUの状態を確認するコマンド

出力結果

画面にはプログラムの最終行の出力が表示される

[レシビ実行] プログラムの実行

全てのプログラムを実行することもできる



The screenshot shows the Google Colaboratory interface. At the top, the browser address bar displays the URL: `colab.research.google.com/github/HirokiNakahara/MNIST_F`. The Colab logo is in the top right corner. Below the browser bar, the Colab toolbar is visible, including buttons for 'Code', 'Text', and 'Insert'. The 'Run' button (a play icon) is highlighted with a blue arrow. Below the toolbar, the 'Runtime' tab is active, showing a list of runtime actions with their corresponding keyboard shortcuts:

- すべてのセルを実行 (Run All): `⌘/Ctrl+F9`
- より前のセルを実行 (Run Previous): `⌘/Ctrl+F8`
- 現在のセルを実行 (Run Current): `⌘/Ctrl+Enter`
- 選択範囲を実行 (Run Selection): `⌘/Ctrl+Shift+Enter`
- 以降のセルを実行 (Run Next): `⌘/Ctrl+F10`
- 実行を中断 (Interrupt): `⌘/Ctrl+M`
- ランタイムを再起動... (Restart): `⌘/Ctrl+M`
- 再起動してすべてのセルを実行... (Restart & Run All): `⌘/Ctrl+M`
- すべてのランタイムをリセット... (Reset All): `⌘/Ctrl+M`

Below the runtime actions, there are buttons for 'ランタイムのタイプを変更' (Change Runtime Type), 'セッションの管理' (Manage Session), and 'ランタイムログの表示' (Show Runtime Log). At the bottom, a status bar shows 'RAM' usage and a 'デイスク' (Disk) icon. The bottom right corner shows a '共有' (Share) button and a user profile icon.

Environment information (環境のセットアップ) is displayed at the bottom of the interface:

```
Mon Jul 1 05:44:11 2019
+-----+
| NVIDIA-SMI 418.67      | Driver Version: 410.79      | CUDA Version: 10.0 |
+-----+-----+
| GPU   Name           Persistence-M| Bus-Id  Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
+-----+-----+-----+
| 0    Tesla K80        Off          |    00000000:00:04:0  Off |                    0 |
| N/A   33C    P8      25W   / 149W |    0MiB / 11441MiB     |      0%      Default |
+-----+-----+-----+
```

[レシビ実行] 学習結果の確認



授業プログラムは最後のセルに学習結果が出力

```
colab.ipynb
ファイル 編集 表示 挿入 ランタイム ツール ヘルプ
コード テキスト セル セル ドライバにコピー
RAM デバイス 編集
共有

print('val_loss: {:.04f} val_accuracy: {:.04f}'.format(
    np.mean(test_losses), np.mean(test accuracies)))

end_time = time.time()
print("-----")
print("Training complete")
trainer.serializers.save_npz('mnist_model', model)
print('accuracy: {:.04f}'.format(np.mean(test accuracies)))
print('time cost: {:.04f}'.format(end_time - start_time), 's')

epoch:01 train_loss:2.2551 train_accuracy:0.1600 val_loss:2.2796 val_accuracy:0.1749
epoch:02 train_loss:2.2331 train_accuracy:0.2900 val_loss:2.2537 val_accuracy:0.2423
epoch:03 train_loss:2.2037 train_accuracy:0.4300 val_loss:2.2286 val_accuracy:0.3202
-----
Training complete
accuracy: 0.3202
time cost: 7.4747 s
```

結果は実際に実行して確認してください