



@katsu1110 2018年10月07日に投稿



Google Colab上でKaggleのデータをロード、モデル訓練、提出の全てを行う

Python

機械学習

DeepLearning

Kaggle

GoogleColaboratory

169



なんか最近Google Colaboratoryが流行ってるらしいですね。

Google Colabを導入

Jupyter notebookをクラウド上で走らせることができるGoogleのウェブサービスですが、注目すべきは**GPUもTPUも無料**ということでしょう。

「ま、まあオレもね、本気を出せばKaggleで上位狙えるんだけどね、ちょ、ちょっと手持ちにフツのラップトップしかなくて、スペック的にアニメ観るくらいしかできないんだよね」

という言い訳ももう通用しない時代に突入しました。

Kaggleは、というより機械学習全般は計算機的能力がものをいいます。課題のデータ量が増大している今、Kaggle内のKernelでは学習が捗らないことも多く、私も去年Kaggleの練習問題（Regression）を解いてKagglerになるという入門記事を書いてからKaggleを去りました(おい

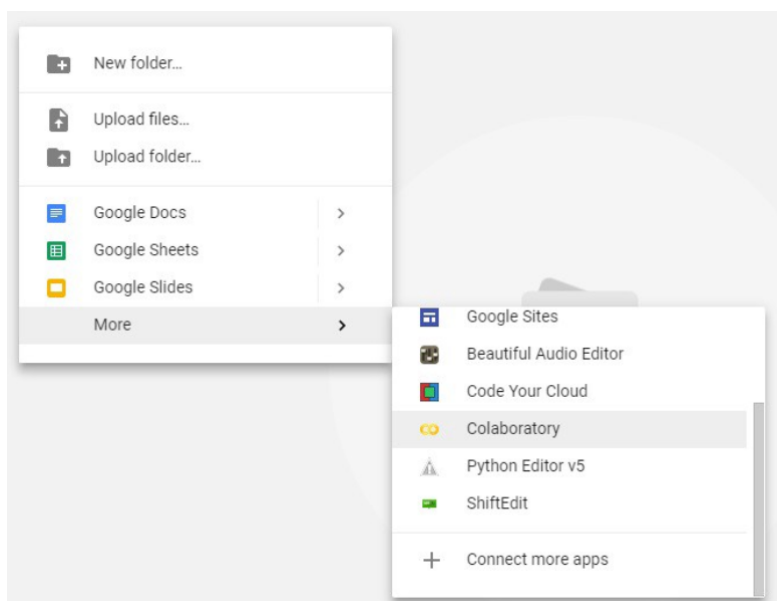
が、Google colabatoryの登場により最近「ちょっとまたKaggleやってもいいかな.....」と思い始めてきました。そこで、機械学習では定番のMNIST、KaggleではDigit Recognizerと呼ばれる練習問題を用いて、

1. Kaggle上のデータをGoogle colaboratoryにロード
2. Google colaboratory上でCNNのトレーニング
3. Google colaboratory上でKaggleに結果を提出

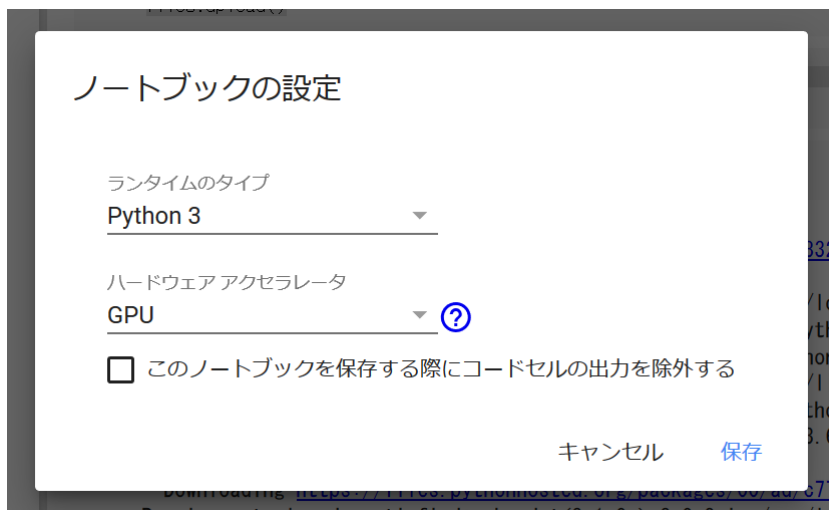
という流れをまとめたいと思います。Google、Kaggleのアカウントを持っていることは前提とします。

Kaggle上のデータをGoogle colaboratoryにロード

Google Driveを開き、Colab Notebookというフォルダー（名前はなんでも大丈夫）を作ります。そこで**右クリック -> More -> Colaboratory**で、Google colaboratoryを始めることができます。



GPUを使いたいので、**編集 -> ノートブックの設定**より、ハードウェアアクセラレータをGPUに変更、保存します。



さて、Google colaboratory上でKaggleのデータを扱うためには、まずKaggle上でAPIを作成する必要があります。KaggleのHPへ行き、**右上の自分のアイコンをクリック -> My Account -> （下の方にある）APIの項目の"Create New API Token"をクリック**します。ダウンロードウィンドウがポップアップで出てくるので、**kaggle.json**をダウンロードします。

API

Using Kaggle's beta API, you can interact with Competitions and Datasets to download data, make submissions, and more via the command line. [Read the docs](#)

[Create New API Token](#)[Expire API Token](#)

Google colaboratoryに戻り、一行目に以下を打ち込みます。

```
from google.colab import files
files.upload()
```

このセルを実行すると、先ほどダウンロードしたkaggle.jsonを開くように求められます。kaggle.jsonを開き、以下を打ち込むことでパスを設定します。

```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
```

Kaggleをインストールします。

```
!pip install kaggle
```

すると、以下のようにばーっとインストールが進むはずです。

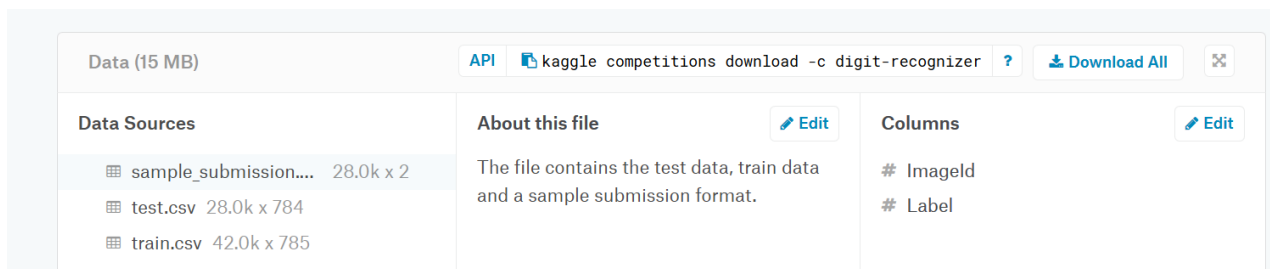
```
!pip install kaggle

Collecting kaggle
  Downloading https://files.pythonhosted.org/packages/c6/78/832b9a9ec6b3baf8ec566e1f0a695f2fd08d2c94a
    100% |#####| 61kB 2.4MB/s
Requirement already satisfied: urllib3<1.23.0,>=1.15 in /usr/local/lib/python3.6/dist-packages (from kaggle) (1.15)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.6/dist-packages (from kaggle) (1.11)
Requirement already satisfied: certifi in /usr/local/lib/python3.6/dist-packages (from kaggle) (2018.8.24)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.6/dist-packages (from kaggle) (2.6.0)
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from kaggle) (2.18.4)
Requirement already satisfied: tqdm in /usr/local/lib/python3.6/dist-packages (from kaggle) (4.26.0)
Collecting python-slugify (from kaggle)
  Downloading https://files.pythonhosted.org/packages/00/ad/c778a6df614b6217c30fe80045b365bfa08b5dd3c
    100% |#####| 235kB 7.1MB/s
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from python-slugify) (3.0.4)
Requirement already satisfied: idna<2.7,>=2.5 in /usr/local/lib/python3.6/dist-packages (from python-slugify) (2.6)
Collecting Unidecode>=0.04.16 (from python-slugify->kaggle)
  Downloading https://files.pythonhosted.org/packages/59/ef/67085e30e8bbcd76e2f0a4ad8151c13a2c5bce77
    100% |#####| 235kB 7.1MB/s
Building wheels for collected packages: kaggle, python-slugify
  Running setup.py bdist_wheel for kaggle ... done
  Stored in directory: /root/.cache/pip/wheels/44/2c/df/22a6eeb780c36c28190faef6252b739fdc47145fd87a6
  Running setup.py bdist_wheel for python-slugify ... done
  Stored in directory: /root/.cache/pip/wheels/e3/65/da/2045deea3098ed7471eca0e2460cfbd3fdfe8c1d6fa6f
Successfully built kaggle python-slugify
Installing collected packages: Unidecode, python-slugify, kaggle
Successfully installed Unidecode-1.0.22 kaggle-1.4.7.1 python-slugify-1.2.6
```

アクセスパーミッションのため、以下を打ち込みます。

```
!chmod 600 /root/.kaggle/kaggle.json
```

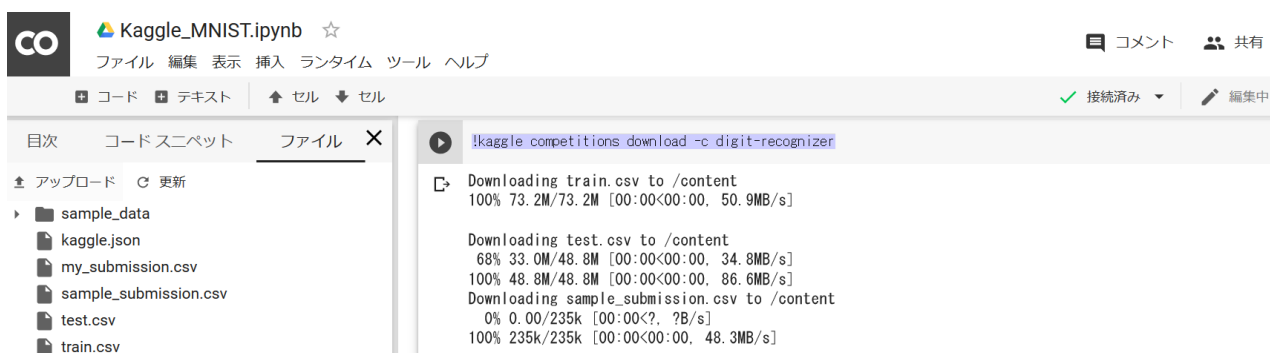
さて、これでもうKaggle上のデータをダウンロードできます。Kaggle上で各CompetitionのDataのページに行くと、APIのアドレスが見つかります。



ダウンロードのコマンドは以下のようになります。

```
!kaggle competitions download -c digit-recognizer
```

これを実行すると無事、ファイルがロードされます。



画面左側で、ロードされたファイルたちが確認できますね。

Google colaboratory上でCNNのトレーニング

データがロードできたら、後は好きにモデルを組んでトレーニングするだけです。MNISTは画像分類課題なので、何も考えずにCNN (Convolutional Neural Network) を使います。より頭を使わなくて済むよう、Kerasで実装します。

Google colaboratoryでKerasをインストールするには、以下を実行します。

```
!pip install -q keras
```

ありがたいことにNumpyやPandasは既に入っているため、今回はもう追加するものはありません。必要なライブラリをimportします。

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# keras
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers.normalization import BatchNormalization
from keras.layers import Conv2D, MaxPooling2D, ZeroPadding2D, GlobalAveragePooling2D
from keras.layers.advanced_activations import LeakyReLU
from keras.preprocessing.image import ImageDataGenerator
from keras.utils.np_utils import to_categorical
from keras.optimizers import SGD, RMSprop
from keras.callbacks import ReduceLROnPlateau
from sklearn.model_selection import train_test_split
```

ダウンロードしたデータは、/content/以下にあるので以下のようにPandasで読み込みます。

```
# load training & test datasets
train = pd.read_csv("/content/train.csv")
test = pd.read_csv("/content/test.csv")
```

今回の記事の目的はGoogle colabatory上でKaggleのデータを扱うことなので、モデルの詳細については説明しません。ここに書かれているモデルでも上位30%は入れると思いますが（保証はしません）、きっとKaggleのPublic Kernel上にもっといいがあるので、そっちをパクってやることをおすすめします。

Train, Test

```
# pandas to numpy
y_train = train["label"]
X_train = train.drop(labels=["label"], axis=1)

del train

# normalize
X_train = X_train/255.0
test = test/255.0

# reshape the data so that the data
# represents (label, img_rows, img_cols, grayscale)
X_train = X_train.values.reshape(-1, 28, 28, 1)
test = test.values.reshape(-1, 28, 28, 1)

# one-hot vector as a label
y_train = to_categorical(y_train, num_classes=10)
```

CNN

```
# convolution -> batch normalization -> ReLU activation -> pooling
model = Sequential()

model.add(Conv2D(32, (3, 3), input_shape=(28,28,1)))
model.add(BatchNormalization(axis=-1))
model.add(Activation('relu'))
model.add(Conv2D(32, (3, 3)))
model.add(BatchNormalization(axis=-1))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(64,(3, 3)))
model.add(BatchNormalization(axis=-1))
model.add(Activation('relu'))
model.add(Conv2D(64, (3, 3)))
model.add(BatchNormalization(axis=-1))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())
```

```
# Fully connected layer
model.add(Dense(512))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(10))

model.add(Activation('softmax'))
```

Compile

```
# compile model
optimizer = RMSprop(lr=0.001, rho=0.9, epsilon=1e-08, decay=0.0)
model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
```

for cross validation

```
# cross validation
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.10, random
```



data argumentation

```
# data argumentation
gen = ImageDataGenerator(rotation_range=8, width_shift_range=0.08, shear_range=0.3,
                        height_shift_range=0.08, zoom_range=0.08)
train_generator = gen.flow(X_train, y_train, batch_size=64)
```


decreasing learning rate

```
# learning rate
learning_rate_reduction = ReduceLROnPlateau(monitor='val_acc', patience=3, verbose=1, fact
```

training model

```
# model training
model.fit_generator(train_generator, epochs=30, validation_data = (X_val, y_val), verbose=
                    callbacks=[learning_rate_reduction])
```

あとはGPUがモデルを訓練し終わるのを待つだけです。

Google colaboratory上でKaggleに結果を提出

モデルの訓練が終わったら、Google colaboratory上から結果をKaggleに提出しましょう。

まず、モデルのテストデータに対する予測は、

```
# model prediction on test data
predictions = model.predict_classes(test, verbose=0)
```

次に、提出用csvを作ります。

```
# make a submission file
submissions = pd.DataFrame({"ImageId": list(range(1,len(predictions)+1)),
                            "Label": predictions})
```

```
submissions.to_csv("my_submission.csv", index=False, header=True)
```

これを、以下のコマンドでKaggleに提出することができます。

```
# submit the file to kaggle
!kaggle competitions submit digit-recognizer -f my_submission.csv -m "Yeah! I submit my fi
```

```
# submit the file to kaggle
!kaggle competitions submit digit-recognizer -f my_submission.csv -m "Yeah! I submit my file through the C
```

☞ Successfully submitted to Digit Recognizer

上のように表示されたら、結果はKaggleに提出されています。

終わりに

Google colaboratoryに登場によって、計算能力が比較的劣るコンピューターしかない個人でも、PythonをGPUやTPUを使って無料で走らせられるようになりました。こうした、個人の可能性が広がるサービスって素晴らしいですね!

参考

- [Kaggle/kaggle-api](#)
- [Using kaggle datasets into Google Colab](#)
- [Applying Convolutional Neural Network on the MNIST dataset](#)

🔗 編集リクエスト

📁 ストック

👍 いいね 169



**@katsu1110**

ドイツから帰国しました。ごはん美味しいです。

<https://katsu1110.github.io/>

フォロー

ユーザー登録して、Qiitaをもっと便利に使ってみませんか。

登録する

ログインする

富士通のAI「Zinrai-ジンライ」

富士通

【30日間無料トライアル】画像認識／音声合成／感情認識など、人気のAPIを無料で

開く

© 2011-2019 Increments Inc. [利用規約](#) [ガイドライン](#) [プライバシー](#) [ヘルプ](#)

[Qiita とは](#) [ユーザー](#) [タグ](#) [記事](#) [ブログ](#) [API](#) [Qiita:Team](#) [Qiita:Zine](#) [広告掲載](#) [ご意見](#)