CONTENTS

ビジュアルヘルプ – TensorFlow による MNIST 手書き数字モデルの実行

概要

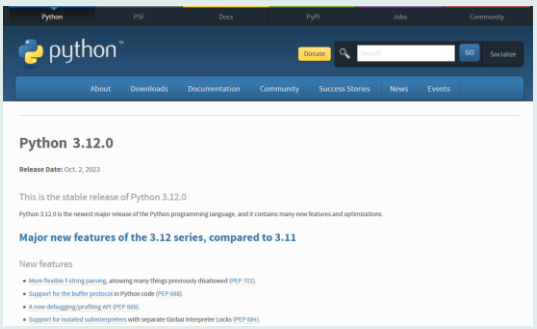
このガイドでは、TensorFlow を使って MNIST データセットを用いた手書き数字分類のためのモデルを訓練し、テストする方法について説明します。
モデルは Python でトレーニングされ、結果（予測やモデルの重みなど）を、さらなる分析と視覚化のために Igor Pro にインポートします。

前提条件と環境構築

1. Python 3.12 をインストールします。
Windows 版のインストーラーは次の場所にあります。

https://www.python.org/downloads/release/python-3120/

注意：2025 年 5 月現在、TensorFlow は Python 3.13 をサポートしていません。



2. venv を使って、TensorFlow をインストールするための新しい仮想環境を作成します（詳細はヘルプ Creating a Virtual Environment（Python.ihf）を参照）。

まず、Windows のシステムコマンドプロンプト、PowerShell などから Python のパスが通っているフォルダーに移動します。
次に、以下を実行します。

```
python -m venv machineLearning
```

Python 3.12 がデフォルトのシステム Python ではない場合、Python 3.12 の実行ファイルのフルパスを使用する必要があるかもしれません。
その場合、次のよう指定します：

```
<パス>¥Python312¥python.exe -m venv machineLearning
```

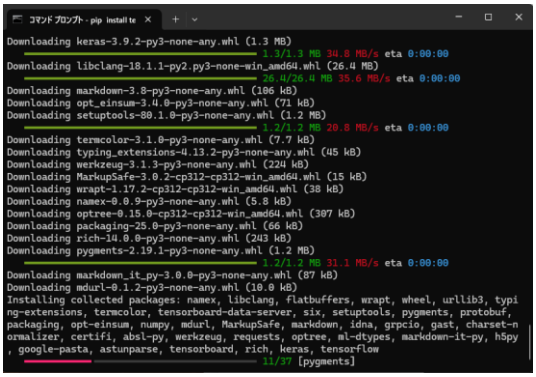
3. 新しい環境を起動します。

```
machineLearning¥Scripts¥activate
```

4. **TensorFlow の Python モジュールをインストールします。**

注意：TensorFlow は非常に大規模なパッケージであり、ディスク容量に約 2GB のスペースを消費します。

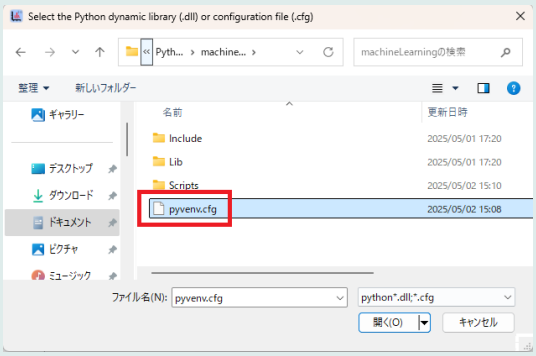
```
pip install tensorflow
```

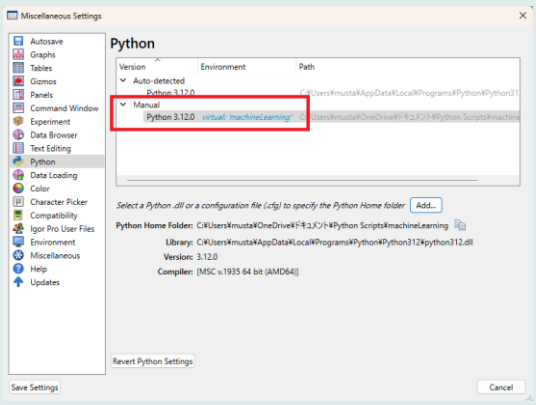


5. 新しい仮想環境を使用するように Igor Pro を設定します。
Igor Pro 10 を開いたら、Misc → Miscellaneous Settings メニューに移動し、Python セクションを見つけます。

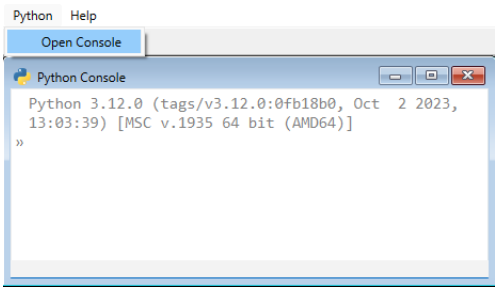
「Add...」をクリックして、仮想環境を参照します。machineLearning フォルダー内にある .cfg ファイルを選択します。

Igor のリストに machineLearning Python 環境が表示されます。
それを選択し、「Save Settings」をクリックします。





6. メニュー Python → Open Console から Python コンソールを開くと、環境設定が指示に従って行われている場合、Python 3.12 を使って Python セッションが開始されます。



モデルのトレーニング

以下の Python コードには、モデルのトレーニングとテストを行う 2 つのメソッド、trainModel() と testModel(model) が定義されています。
これらのメソッドは numberDetection.py ファイル内に含まれています。
モデルを使うには、まずモデルをトレーニングする必要があります。
Python メニュー → Open Console を選択して、Python コンソールを開きます。
Python が正しく設定されている場合、Python 3.12 を使用して起動するはずです。

起動しない場合は、Miscellaneous Settings に戻り、正しい環境を選択してから、Igor を再起動して再試行してください。

7. Python コンソールから、次のコマンドを実行してください：

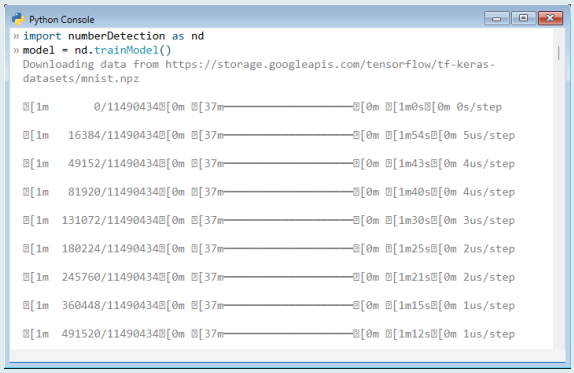
```
import numberDetection as nd
```

これはサンプルのモジュールをインポートします。
このモジュールには TensorFlow が含まれているため、完了までに数分かかる場合があります。

モジュールをインポートした後、MNIST データセットに対してモデルをトレーニングします：

```
model = nd.trainModel()
```

モデルが学習されるにつれ、コンソールに多くの出力が表示されるはずです。

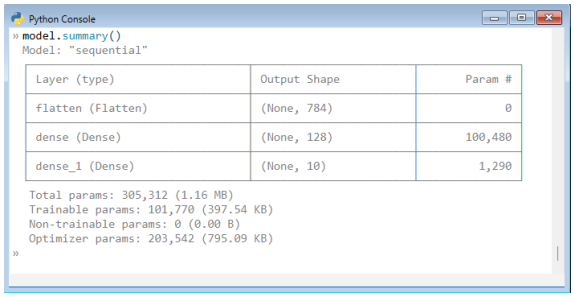


8. trainModel() が何を実行しているかをみるには numberDetection.py のコードを確認してください。
簡単に言うと、まずトレーニングデータとテストデータを Python に読み込みます。
これらのデータは、MNIST から提供された手書きの数字の 28×28 ピクセルの白黒画像の大量のデータセットです。

モデルのトレーニングが完了したら、次のコマンドを実行することで、その要約を出力できます：

```
model.summary()
```

モデルの概要は右のような形式で表示されるはずです。



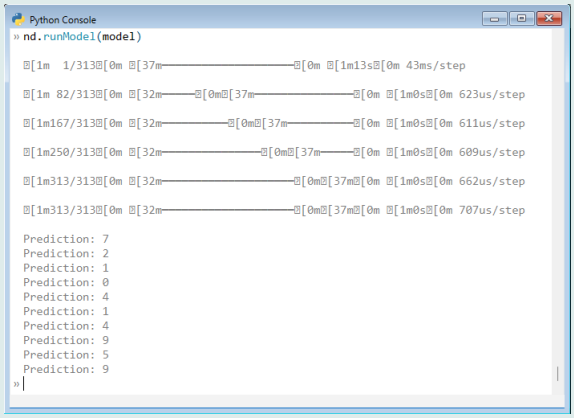
ここでトレーニングされているモデルは、28×28 の画像を入力として受け取り、それらを平坦化して 1 つの 784 要素の層に変換するシーケンシャルモデルとして定義されています。
これは 1 つの隠れた Dense レイヤーに渡し、最終的に出力層に送られます。
出力層には 10 個の要素があり、それぞれ 0 から 9 までの異なる数字の分類を表しています。

モデルの実行

9. Python コンソールから、次のコマンドを実行してください：

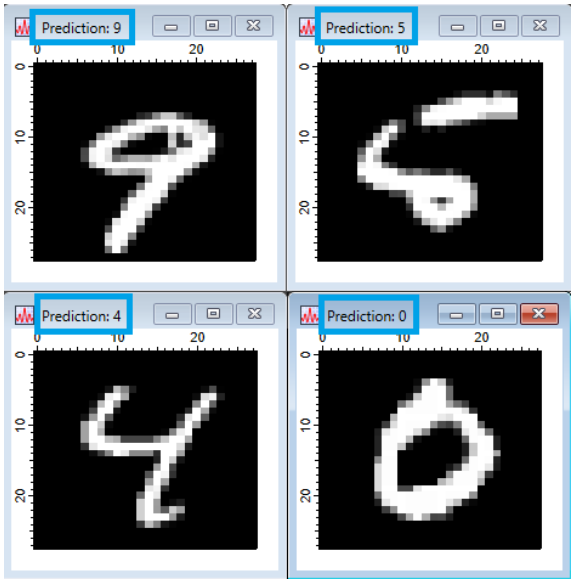
```
nd.runModel(model)
```

このコードは、モデルがこれまで見たことのないテスト画像に対してモデルを実行します。
モデルが学習されるにつれ、コンソールに多くの出力が表示されるはずです。
ここでは、データセットの最初の 10 枚の画像のみをテストしていますが、コードを修正することで、テストデータのうち任意の量に対してモデルを実行するように変更可能です。



10. 各画像がモデルで実行されると、Igor はテストデータの表示を行います。
画像のタイトルバーには、モデルによって各画像が予測された数字が表示され、その値はコンソールにも出力されます。

例えば、右に４つの出力画像を示します。
各画像のウィンドウタイトルには、手書きの数字の予測結果が表示されています：



numberDetection.py の内容

```
import numpy as np
import setuptools
import tensorflow as tf
import igorpro

# Train a model to categorize handwritten digits
def trainModel():
    # Load the mnist handwritten numbers dataset
    (x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
    # Assert that the data set loaded correctly into the expected shapes
    assert x_train.shape == (60000, 28, 28)
    assert x_test.shape == (10000, 28, 28)
    assert y_train.shape == (60000,)
    assert y_test.shape == (10000,)

    # Normalize the data
    x_train = x_train / 255.0
    x_test = x_test / 255.0

    # Define the model
    model = tf.keras.Sequential([
        tf.keras.Input(shape=(28,28)),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(128, activation=tf.keras.activations.relu),
        tf.keras.layers.Dense(10, activation=tf.keras.activations.softmax)
    ])

    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

    model.fit(x_train, y_train, epochs=5, validation_data=(x_test, y_test))
```

```
    eval = model.evaluate(x_test, y_test)
    print(f"Loss: {eval[0]}")
    print(f"Accuracy: {eval[1]}")

    # Get the weights into Igor waves
    weights = model.get_weights() # list of numpy arrays
    for i, weight_array in enumerate(weights):
        igorpro.wave.createfrom("ModelWeights_" + str(i), weight_array, overwrite = True)

    return model

# Run the handwritten digit model
def runModel(model):
     # Load the mnist handwritten numbers dataset
    (x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()

    # Pass the test data into the model, and make the predictions
    tensor = tf.convert_to_tensor(x_test)
    probs = model.predict(tensor)
    predictions = np.argmax(probs, axis=1)
    print("")
    for i in range(10):
        print(f"Prediction: {predictions[i]}")
        w = igorpro.wave.createfrom("predTest_" + str(i), x_test[i], overwrite = True)
        igorpro.execute(f"NewImage/N=numberTest_{i} " + w.path() + w.name())
        igorpro.execute(f"DoWindow/T numberTest_{i} ¥"Prediction: {predictions[i]}¥"")

if __name__ == '__main__':
    model = trainModel()
    runModel(model)
```