

PROJECT 2

Experiments with Convolutional Neural Network in TensorFlow

When train_x_7_10.csv and train_y_7_10.csv was runned accuracy was 0.7144

- 1) When epoch value changed from 5 to 15 then the accuracy changed from 0.7144 to 0.7692

The screenshot shows a Mac OS X desktop with a dark theme. A Visual Studio Code window is open, displaying a Python project structure. The Explorer sidebar on the left lists files: proj.py, fraction_xy.py, proj2.py, train_x_7.10.csv, and train_y_7.10.csv. The proj.py file is the active editor, showing code for a neural network model. The terminal tab at the bottom shows the output of the training process and the loading of testing data.

```
proj.py — proj-data

EXPLORER OPEN EDITORS PROJ-DATA
proj.py x fraction_xy.py
proj.py
34     tf.keras.layers.Conv2D(7, (3,3), padding='same', activation=tf.nn.relu),
35     tf.keras.layers.MaxPool2D(2, 2),
36     tf.keras.layers.Flatten(),
37     tf.keras.layers.Dense(512, activation=tf.nn.relu),
38     tf.keras.layers.Dense(10, activation=tf.nn.softmax)
39   ])
40   model.compile(optimizer='adam',
41                 loss='sparse_categorical_crossentropy',
42                 metrics=['accuracy'])
43
44   print("train")
45   model.fit(x_train, y_train, epochs=15)
46
47   print("Reading testing data")
48   x_test_2d = np.loadtxt(test_x_location, dtype="uint8")
49   x_test_3d = x_test_2d.reshape(-1,28,28,1)
50   x_test = x_test_3d
51   y_test = np.loadtxt(test_y_location, dtype="uint8")
52
53   print("Pre processing testing data")
54   x_test = x_test / 255.0

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: Python + ^ x

188/188 [=====] - 1s 4ms/step - loss: 0.6538 - accuracy: 0.7435
Epoch 12/15
188/188 [=====] - 1s 4ms/step - loss: 0.6183 - accuracy: 0.7641
Epoch 13/15
188/188 [=====] - 1s 4ms/step - loss: 0.6031 - accuracy: 0.7707
Epoch 14/15
188/188 [=====] - 1s 4ms/step - loss: 0.5984 - accuracy: 0.7714
Epoch 15/15
188/188 [=====] - 1s 4ms/step - loss: 0.6073 - accuracy: 0.7675
Reading testing data
Pre processing testing data
evaluate
313/313 [=====] - 1s 3ms/step - loss: 0.6346 - accuracy: 0.7692
(base) nimirats-MacBook:proj-data nimiratbedi$
```

- 2) Now with epoch value = 15 and network changed (now instead of 6 layers with 7 layers – additional Conv2D layer
Accuracy increased to 0.7720

The screenshot shows a Mac OS X desktop with a Visual Studio Code window open. The title bar reads "proj.py — proj-data". The left sidebar shows the "EXPLORER" view with "OPEN EDITORS" expanded, displaying files like "proj.py", "fraction_xy.py", "proj2.py", "train_x_7_10.csv", and "train_y_7_10.csv". The main editor area has "proj.py" selected. The code implements a neural network for image classification, using TensorFlow's Keras API. The terminal tab at the bottom shows the training progress and evaluation results.

```
23 x_train_2d = np.loadtxt(train_x_location, dtype="uint8")
24 x_train_3d = x_train_2d.reshape(-1,28,28,1)
25 x_train = x_train_3d
26 y_train = np.loadtxt(train_y_location, dtype="uint8")
27
28 print("Pre processing x of training data")
29 x_train = x_train / 255.0
30
31 # define the training model
32 model = tf.keras.models.Sequential([
33     tf.keras.layers.MaxPool2D(4, 4, input_shape=(28,28,1)),
34     tf.keras.layers.Conv2D(7, (3,3), padding='same', activation=tf.nn.relu),
35     tf.keras.layers.Conv2D(7, (3,3), padding='same', activation=tf.nn.relu),
36     tf.keras.layers.MaxPool2D(2, 2),
37     tf.keras.layers.Flatten(),
38     tf.keras.layers.Dense(512, activation=tf.nn.relu),
39     tf.keras.layers.Dense(10, activation=tf.nn.softmax)
40 ])
41 model.compile(optimizer='adam',
42                 loss='sparse_categorical_crossentropy',
43                 metrics=['accuracy'])

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
```

1: Python

```
188/188 [=====] - 1s 6ms/step - loss: 0.6196 - accuracy: 0.7600
Epoch 12/15
188/188 [=====] - 1s 6ms/step - loss: 0.5921 - accuracy: 0.7735
Epoch 13/15
188/188 [=====] - 1s 6ms/step - loss: 0.5674 - accuracy: 0.7870
Epoch 14/15
188/188 [=====] - 1s 6ms/step - loss: 0.5734 - accuracy: 0.7834
Epoch 15/15
188/188 [=====] - 1s 6ms/step - loss: 0.5698 - accuracy: 0.7783
Reading testing data
Pre processing testing data
evaluate
313/313 [=====] - 1s 3ms/step - loss: 0.6105 - accuracy: 0.7720
(base) nimirat-MacBook:proj-data nimirat$
```

- 3) Now with epoch value = 25 and network has 6 layers then we had accuracy after 25 epoch to be 0.7914 under 4 mins. But testing accuracy is 0.7663

```

    proj.py — proj-data
    EXPLORER      ...   proj.py   fraction_xy.py
    > OPEN EDITORS
    > PROJ-DATA
    > big
    > small
    > fraction_xy.py
    > proj.py
    > proj2.py
    train_x_710.csv
    train_y_710.csv

    34     tf.keras.layers.Conv2D(7, (3,3), padding='same', activation=tf.nn.relu),
    35     tf.keras.layers.MaxPool2D(2, 2),
    36     tf.keras.layers.Flatten(),
    37     tf.keras.layers.Dense(512, activation=tf.nn.relu),
    38     tf.keras.layers.Dense(10, activation=tf.nn.softmax)
    39   ]
    40   model.compile(optimizer='adam',
    41                   loss='sparse_categorical_crossentropy',
    42                   metrics=['accuracy'])
    43
    44   print("train")
    45   model.fit(x_train, y_train, epochs=25)
    46
    47   print("Reading testing data")
    48   x_test_2d = np.loadtxt(test_x_location, dtype="uint8")
    49   x_test_3d = x_test_2d.reshape(-1,28,28,1)
    50   x_test = x_test_3d
    51   y_test = np.loadtxt(test_y_location, dtype="uint8")
    52
    53   print("Pre processing testing data")
    54   x_test = x_test / 255.0

    PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
    1: Python  +  -  ^  x

    188/188 [=====] - 1s 4ms/step - loss: 0.5660 - accuracy: 0.7791
    Epoch 22/25
    188/188 [=====] - 1s 4ms/step - loss: 0.5596 - accuracy: 0.7892
    Epoch 23/25
    188/188 [=====] - 1s 4ms/step - loss: 0.5327 - accuracy: 0.7976
    Epoch 24/25
    188/188 [=====] - 1s 4ms/step - loss: 0.5292 - accuracy: 0.7962
    Epoch 25/25
    188/188 [=====] - 1s 4ms/step - loss: 0.5392 - accuracy: 0.7914
    Reading testing data
    Pre processing testing data
    evaluate
    313/313 [=====] - 1s 2ms/step - loss: 0.6225 - accuracy: 0.7663
    (base) nimirat-MacBook:proj-data nimiratbedi$ 
  
```

- 4) When epoch = 20 and network changed (now instead of 6 layers with 7 layers – additional Conv2D layer
Accuracy increased to 0.7803

```

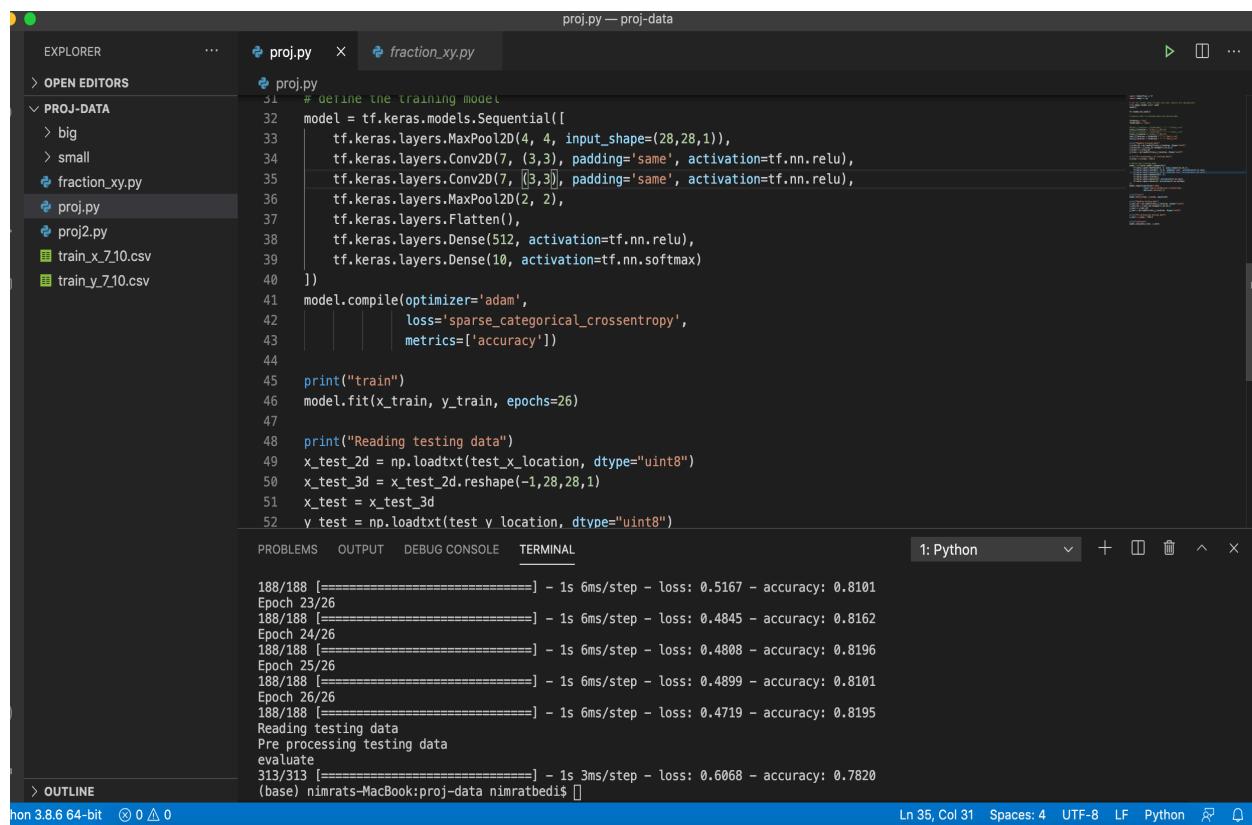
    proj.py — proj-data
    EXPLORER      ...   proj.py   fraction_xy.py
    > OPEN EDITORS
    > PROJ-DATA
    > big
    > small
    > fraction_xy.py
    > proj.py
    > proj2.py
    train_x_710.csv
    train_y_710.csv

    30     tf.keras.layers.MaxPool2D(2, 2),
    31     tf.keras.layers.Flatten(),
    32     tf.keras.layers.Dense(512, activation=tf.nn.relu),
    33     tf.keras.layers.Dense(10, activation=tf.nn.softmax)
    34   ]
    35   model.compile(optimizer='adam',
    36                   loss='sparse_categorical_crossentropy',
    37                   metrics=['accuracy'])
    38
    39   print("train")
    40   model.fit(x_train, y_train, epochs=20)
    41
    42   print("Reading testing data")
    43   x_test_2d = np.loadtxt(test_x_location, dtype="uint8")
    44   x_test_3d = x_test_2d.reshape(-1,28,28,1)
    45   x_test = x_test_3d
    46   y_test = np.loadtxt(test_y_location, dtype="uint8")
    47
    48   print("Pre processing testing data")
    49   x_test = x_test / 255.0
    50
    51   print("evaluate")
    52

    PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
    1: Python  +  -  ^  x

    188/188 [=====] - 1s 6ms/step - loss: 0.5652 - accuracy: 0.7846
    Epoch 17/20
    188/188 [=====] - 1s 6ms/step - loss: 0.5792 - accuracy: 0.7773
    Epoch 18/20
    188/188 [=====] - 1s 8ms/step - loss: 0.5442 - accuracy: 0.7929
    Epoch 19/20
    188/188 [=====] - 1s 6ms/step - loss: 0.5201 - accuracy: 0.8017
    Epoch 20/20
    188/188 [=====] - 1s 6ms/step - loss: 0.5400 - accuracy: 0.7949
    Reading testing data
    Pre processing testing data
    evaluate
    313/313 [=====] - 1s 3ms/step - loss: 0.5975 - accuracy: 0.7803
    (base) nimirat-MacBook:proj-data nimiratbedi$ 
  
```

- 5) When epoch = 26 and network changed (now instead of 6 layers with 7 layers – additional Conv2D layer
Accuracy increased to 0.7820



```

proj.py — proj-data

EXPLORER      ...  proj.py  fraction_xy.py
PROJ-DATA
> big
> small
fraction_xy.py
proj.py
proj2.py
train_x_7.10.csv
train_y_7.10.csv

31 # define the training model
32 model = tf.keras.models.Sequential([
33     tf.keras.layers.MaxPool2D(4, 4, input_shape=(28,28,1)),
34     tf.keras.layers.Conv2D(7, (3,3), padding='same', activation=tf.nn.relu),
35     tf.keras.layers.Conv2D(7, (3,3), padding='same', activation=tf.nn.relu),
36     tf.keras.layers.MaxPool2D(2, 2),
37     tf.keras.layers.Flatten(),
38     tf.keras.layers.Dense(512, activation=tf.nn.relu),
39     tf.keras.layers.Dense(10, activation=tf.nn.softmax)
40 ])
41 model.compile(optimizer='adam',
42                 loss='sparse_categorical_crossentropy',
43                 metrics=['accuracy'])
44
45 print("train")
46 model.fit(x_train, y_train, epochs=26)
47
48 print("Reading testing data")
49 x_test_2d = np.loadtxt(test_x_location, dtype="uint8")
50 x_test_3d = x_test_2d.reshape(-1,28,28,1)
51 x_test = x_test_3d
52 y_test = np.loadtxt(test_y_location, dtype="uint8")

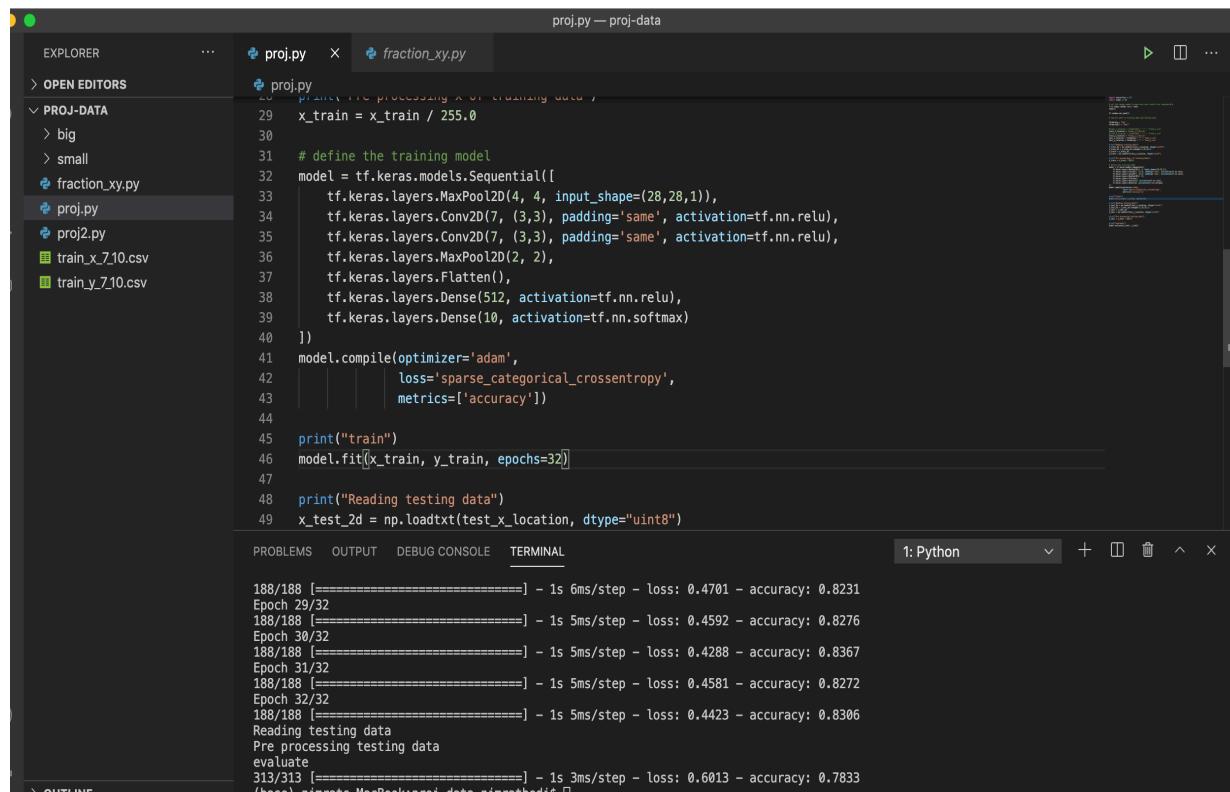
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
1: Python  +  □  ^  ×

188/188 [=====] - 1s 6ms/step - loss: 0.5167 - accuracy: 0.8101
Epoch 23/26
188/188 [=====] - 1s 6ms/step - loss: 0.4845 - accuracy: 0.8162
Epoch 24/26
188/188 [=====] - 1s 6ms/step - loss: 0.4808 - accuracy: 0.8196
Epoch 25/26
188/188 [=====] - 1s 6ms/step - loss: 0.4899 - accuracy: 0.8101
Epoch 26/26
188/188 [=====] - 1s 6ms/step - loss: 0.4719 - accuracy: 0.8195
Reading testing data
Pre processing testing data
evaluate
313/313 [=====] - 1s 3ms/step - loss: 0.6068 - accuracy: 0.7820
(base) nimirats-MacBook:proj-data nimiratbedi$ []

```

Ln 35, Col 31 Spaces: 4 UTF-8 LF Python ⚙ ⌂

- 6) When epoch = 32 and network changed (now instead of 6 layers with 7 layers – additional Conv2D layer
Accuracy increased to 0.7833



```

proj.py — proj-data

EXPLORER      ...  proj.py  fraction_xy.py
PROJ-DATA
> big
> small
fraction_xy.py
proj.py
proj2.py
train_x_7.10.csv
train_y_7.10.csv

20 print("Pre processing X of training data")
21 x_train = x_train / 255.0
22
23 # define the training model
24 model = tf.keras.models.Sequential([
25     tf.keras.layers.MaxPool2D(4, 4, input_shape=(28,28,1)),
26     tf.keras.layers.Conv2D(7, (3,3), padding='same', activation=tf.nn.relu),
27     tf.keras.layers.Conv2D(7, (3,3), padding='same', activation=tf.nn.relu),
28     tf.keras.layers.MaxPool2D(2, 2),
29     tf.keras.layers.Flatten(),
30     tf.keras.layers.Dense(512, activation=tf.nn.relu),
31     tf.keras.layers.Dense(10, activation=tf.nn.softmax)
32 ])
33 model.compile(optimizer='adam',
34                 loss='sparse_categorical_crossentropy',
35                 metrics=['accuracy'])
36
37 print("train")
38 model.fit(x_train, y_train, epochs=32)
39
40 print("Reading testing data")
41 x_test_2d = np.loadtxt(test_x_location, dtype="uint8")

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
1: Python  +  □  ^  ×

188/188 [=====] - 1s 6ms/step - loss: 0.4701 - accuracy: 0.8231
Epoch 29/32
188/188 [=====] - 1s 5ms/step - loss: 0.4592 - accuracy: 0.8276
Epoch 30/32
188/188 [=====] - 1s 5ms/step - loss: 0.4288 - accuracy: 0.8367
Epoch 31/32
188/188 [=====] - 1s 5ms/step - loss: 0.4581 - accuracy: 0.8272
Epoch 32/32
188/188 [=====] - 1s 5ms/step - loss: 0.4423 - accuracy: 0.8306
Reading testing data
Pre processing testing data
evaluate
313/313 [=====] - 1s 3ms/step - loss: 0.6013 - accuracy: 0.7833
(base) nimirats-MacBook:proj-data nimiratbedi$ []

```

Ln 35, Col 31 Spaces: 4 UTF-8 LF Python ⚙ ⌂

7) When epoch changed to 40 the accuracy came out to be 0.7832

The screenshot shows a Mac OS X desktop environment with a terminal window open. The terminal window has a dark theme and displays Python code for a neural network. The code imports TensorFlow and defines a Sequential model with multiple layers: MaxPool2D, Conv2D, Flatten, and Dense layers. It then compiles the model with Adam optimizer, sparse categorical crossentropy loss, and accuracy metric. The code prints "train" and fits the model to training data for 40 epochs. Below the code, the terminal output shows the training progress with 188/188 steps, followed by 37 to 40 epochs, each taking about 6ms per step. The accuracy starts at 0.4132 and rises to 0.7832. Finally, it reads testing data, processes it, evaluates the model, and prints the accuracy of 0.7832.

```
Code File Edit Selection View Go Run Terminal Window Help
EXPLORER ... proj.py fraction_xy.py
OPEN EDITORS
PROJ-DATA
> big
> small
fraction_xy.py
proj.py
proj2.py
train_x_7.10.csv
train_y_7.10.csv
proj.py — proj-data
26     y_train = np.loadtxt(train_y_location, dtype="uint8")
27
28     print("Pre processing x of training data")
29     x_train = x_train / 255.0
30
31     # define the training model
32     model = tf.keras.models.Sequential([
33         tf.keras.layers.MaxPool2D(4, 4, input_shape=(28,28,1)),
34         tf.keras.layers.Conv2D(7, (3,3), padding='same', activation=tf.nn.relu),
35         tf.keras.layers.Conv2D(7, (3,3), padding='same', activation=tf.nn.relu),
36         tf.keras.layers.MaxPool2D(2, 2),
37         tf.keras.layers.Flatten(),
38         tf.keras.layers.Dense(512, activation=tf.nn.relu),
39         tf.keras.layers.Dense(10, activation=tf.nn.softmax)
40     ])
41     model.compile(optimizer='adam',
42                     loss='sparse_categorical_crossentropy',
43                     metrics=['accuracy'])
44
45     print("train")
46     model.fit(x_train, y_train, epochs=40)
47
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: Python
188/188 [=====] - 1s 6ms/step - loss: 0.4132 - accuracy: 0.8361
Epoch 37/40
188/188 [=====] - 1s 5ms/step - loss: 0.4044 - accuracy: 0.8437
Epoch 38/40
188/188 [=====] - 1s 6ms/step - loss: 0.3738 - accuracy: 0.8535
Epoch 39/40
188/188 [=====] - 1s 6ms/step - loss: 0.3822 - accuracy: 0.8536
Epoch 40/40
188/188 [=====] - 1s 6ms/step - loss: 0.3935 - accuracy: 0.8454
Reading testing data
Pre processing testing data
evaluate
313/313 [=====] - 1s 3ms/step - loss: 0.6371 - accuracy: 0.7832
(base) nimrats-MacBook:proj-data nimratbedi$
```

So after many trial and error the best accuracy turn out to be 78.33% before any error it was 71%