ffnn_trial_2

Python · Digit Recognizer

Competition Notebook	Run	Public Score	Best Score
Digit Recognizer	599.0s - GPU	0.97707	0.97707 V4

Version 4 of 4

GPU

FFNN

In [1]:

```
# Importing necessary libraries
import pandas as pd
import numpy as np
from tensorflow import keras
from keras.layers import Dense, MaxPool2D, Conv2D, Dropout,Flatten
from keras.models import Sequential
from keras.utils import np_utils
from keras.datasets import mnist
from sklearn import metrics
import numpy
from matplotlib import pyplot as plt

import tensorflow as  tf
from keras.preprocessing.image import ImageDataGenerator
```

In [2]:

```
train_dataset = pd.read_csv('../input/digit-recognizer/train.csv').values
test_dataset_X = pd.read_csv('../input/digit-recognizer/test.csv').values
test_dataset_y = pd.read_csv('../input/digit-recognizer/sample_submission.csv').values
```

In [3]:

```
train_dataset
```

Out[3]:

```
array([[1, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [1, 0, 0, ..., 0, 0, 0],
       ...,
       [7, 0, 0, ..., 0, 0, 0],
       [6, 0, 0, ..., 0, 0, 0],
       [9, 0, 0, ..., 0, 0, 0]])
```

In [4]:

```
test_dataset_X
```

Out[4]:

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

```
test_dataset_y
```

```
array([[    1,     0],
       [    2,     0],
       [    3,     0],
       ...,
       [27998,     0],
       [27999,     0],
       [28000,     0]])
```

```
print(train_dataset.shape)
print(test_dataset_X.shape)
print(test_dataset_y.shape)
```

```
(42000, 785)
(28000, 784)
(28000, 2)
```

```
col = test_dataset_y[:,1]
col.shape = (len(col),1)
```

```
test_dataset = np.hstack((test_dataset_X[:,:1], col, test_dataset_X[:,1:]))
```

```
X_train = train_dataset[:, 1:].reshape(train_dataset.shape[0],28,28, 1).astype( 'float32'
)
X_train = X_train / 255.0

y_train = train_dataset[:,0]

X_test = test_dataset[:, 1:].reshape(test_dataset.shape[0],28,28, 1).astype( 'float32' )
X_test = X_test / 255.0

y_test = test_dataset[:,0]
```

```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(42000, 28, 28, 1)
(28000, 28, 28, 1)
(42000,)
(28000,)
```

```
# y_train = np_utils.to_categorical(y_train)
# y_test_copy = y_test                          #for future use
# y_test = np_utils.to_categorical(y_test, num_classes = 10)
# print(y_train.shape)
# print(y_test.shape)
```

```
model=tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten(input_shape=(28,28,1)))
model.add(tf.keras.layers.Dense(512,activation='relu'))
model.add(tf.keras.layers.Dense(256,activation='relu'))
model.add(tf.keras.layers.Dense(128,activation='relu'))
model.add(tf.keras.layers.Dense(64,activation='relu'))
model.add(tf.keras.layers.Dense(32,activation='relu'))
model.add(tf.keras.layers.Dense(16,activation='relu'))
model.add(tf.keras.layers.Dense(10,activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accura
```

```
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accurac
y'])
model.fit(x=X_train,y=y_train,batch_size=32,epochs=100)
model.summary()
```

```
2022-03-12 03:01:19.799262: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:93
7] successful NUMA node read from SysFS had negative value (-1), but there must be at
least one NUMA node, so returning NUMA node zero
2022-03-12 03:01:19.809567: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:93
7] successful NUMA node read from SysFS had negative value (-1), but there must be at
least one NUMA node, so returning NUMA node zero
2022-03-12 03:01:19.810486: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:93
7] successful NUMA node read from SysFS had negative value (-1), but there must be at
least one NUMA node, so returning NUMA node zero
2022-03-12 03:01:19.811798: I tensorflow/core/platform/cpu_feature_guard.cc:142] This
TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use
the following CPU instructions in performance-critical operations:  AVX2 AVX512F FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler f
lags.
2022-03-12 03:01:19.812797: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:93
7] successful NUMA node read from SysFS had negative value (-1), but there must be at
least one NUMA node, so returning NUMA node zero
2022-03-12 03:01:19.813467: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:93
7] successful NUMA node read from SysFS had negative value (-1), but there must be at
least one NUMA node, so returning NUMA node zero
2022-03-12 03:01:19.814068: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:93
7] successful NUMA node read from SysFS had negative value (-1), but there must be at
least one NUMA node, so returning NUMA node zero
2022-03-12 03:01:21.606013: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:93
7] successful NUMA node read from SysFS had negative value (-1), but there must be at
least one NUMA node, so returning NUMA node zero
2022-03-12 03:01:21.606838: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:93
7] successful NUMA node read from SysFS had negative value (-1), but there must be at
least one NUMA node, so returning NUMA node zero
2022-03-12 03:01:21.607479: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:93
7] successful NUMA node read from SysFS had negative value (-1), but there must be at
least one NUMA node, so returning NUMA node zero
2022-03-12 03:01:21.608048: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1510] C
reated device /job:localhost/replica:0/task:0/device:GPU:0 with 15403 MB memory:  -> d
evice: 0, name: Tesla P100-PCIE-16GB, pci bus id: 0000:00:04.0, compute capability: 6.
0
2022-03-12 03:01:22.404081: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.c
c:185] None of the MLIR Optimization Passes are enabled (registered 2)

Epoch 1/100
2100/2100 [==============================] - 6s 3ms/step - loss: 0.2782 - accuracy: 0.
9183
Epoch 2/100
2100/2100 [==============================] - 5s 3ms/step - loss: 0.1227 - accuracy: 0.
9641
Epoch 3/100
2100/2100 [==============================] - 4s 3ms/step - loss: 0.0905 - accuracy: 0.
9743
Epoch 4/100
2100/2100 [==============================] - 4s 3ms/step - loss: 0.0720 - accuracy: 0.
9793
Epoch 5/100
2100/2100 [==============================] - 5s 3ms/step - loss: 0.0603 - accuracy: 0.
9829
Epoch 6/100
2100/2100 [==============================] - 5s 3ms/step - loss: 0.0481 - accuracy: 0.
9867
Epoch 7/100
2100/2100 [==============================] - 5s 3ms/step - loss: 0.0472 - accuracy: 0.
9881
Epoch 8/100
2100/2100 [==============================] - 4s 3ms/step - loss: 0.0399 - accuracy: 0.
9896
Epoch 9/100
2100/2100 [==============================] - 4s 3ms/step - loss: 0.0317 - accuracy: 0.
9918
Epoch 10/100
```

```
2100/2100 [==============================] - 4s 2ms/step - loss: 0.0267 - accuracy: 0.
9920
Epoch 11/100
2100/2100 [==============================] - 4s 2ms/step - loss: 0.0229 - accuracy: 0.
9936
Epoch 12/100
2100/2100 [==============================] - 5s 2ms/step - loss: 0.0282 - accuracy: 0.
9931
Epoch 13/100
2100/2100 [==============================] - 5s 2ms/step - loss: 0.0218 - accuracy: 0.
9941
Epoch 14/100
2100/2100 [==============================] - 5s 2ms/step - loss: 0.0258 - accuracy: 0.
9923
Epoch 15/100
2100/2100 [==============================] - 4s 2ms/step - loss: 0.0191 - accuracy: 0.
9951
Epoch 16/100
2100/2100 [==============================] - 4s 2ms/step - loss: 0.0199 - accuracy: 0.
9948
Epoch 17/100
2100/2100 [==============================] - 5s 2ms/step - loss: 0.0189 - accuracy: 0.
9954
Epoch 18/100
2100/2100 [==============================] - 4s 2ms/step - loss: 0.0180 - accuracy: 0.
9956
Epoch 19/100
2100/2100 [==============================] - 5s 2ms/step - loss: 0.0182 - accuracy: 0.
9957
Epoch 20/100
2100/2100 [==============================] - 5s 2ms/step - loss: 0.0156 - accuracy: 0.
9963
Epoch 21/100
2100/2100 [==============================] - 5s 2ms/step - loss: 0.0148 - accuracy: 0.
9960
Epoch 22/100
2100/2100 [==============================] - 5s 2ms/step - loss: 0.0153 - accuracy: 0.
9966
Epoch 23/100
2100/2100 [==============================] - 4s 2ms/step - loss: 0.0131 - accuracy: 0.
9968
Epoch 24/100
2100/2100 [==============================] - 5s 2ms/step - loss: 0.0154 - accuracy: 0.
9964
Epoch 25/100
2100/2100 [==============================] - 4s 2ms/step - loss: 0.0146 - accuracy: 0.
9967
Epoch 26/100
2100/2100 [==============================] - 5s 2ms/step - loss: 0.0132 - accuracy: 0.
9971
Epoch 27/100
2100/2100 [==============================] - 5s 2ms/step - loss: 0.0139 - accuracy: 0.
9969
Epoch 28/100
2100/2100 [==============================] - 5s 2ms/step - loss: 0.0129 - accuracy: 0.
9974
Epoch 29/100
2100/2100 [==============================] - 5s 2ms/step - loss: 0.0143 - accuracy: 0.
9971
Epoch 30/100
2100/2100 [==============================] - 4s 2ms/step - loss: 0.0118 - accuracy: 0.
9975
Epoch 31/100
2100/2100 [==============================] - 5s 2ms/step - loss: 0.0121 - accuracy: 0.
9976
Epoch 32/100
2100/2100 [==============================] - 4s 2ms/step - loss: 0.0095 - accuracy: 0.
9982
Epoch 33/100
2100/2100 [==============================] - 5s 2ms/step - loss: 0.0159 - accuracy: 0.
9972
Epoch 34/100
```

```
2100/2100 [==============================] - 5s 3ms/step - loss: 0.0121 - accuracy: 0.
9977
Epoch 35/100
2100/2100 [==============================] - 5s 3ms/step - loss: 0.0098 - accuracy: 0.
9982
Epoch 36/100
2100/2100 [==============================] - 5s 3ms/step - loss: 0.0130 - accuracy: 0.
9974
Epoch 37/100
2100/2100 [==============================] - 4s 3ms/step - loss: 0.0093 - accuracy: 0.
9983
Epoch 38/100
2100/2100 [==============================] - 5s 3ms/step - loss: 0.0103 - accuracy: 0.
9981
Epoch 39/100
2100/2100 [==============================] - 4s 3ms/step - loss: 0.0107 - accuracy: 0.
9977
Epoch 40/100
2100/2100 [==============================] - 5s 3ms/step - loss: 0.0109 - accuracy: 0.
9976
Epoch 41/100
2100/2100 [==============================] - 5s 3ms/step - loss: 0.0093 - accuracy: 0.
9983
Epoch 42/100
2100/2100 [==============================] - 5s 3ms/step - loss: 0.0117 - accuracy: 0.
9983
Epoch 43/100
2100/2100 [==============================] - 5s 3ms/step - loss: 0.0177 - accuracy: 0.
9973
Epoch 44/100
2100/2100 [==============================] - 4s 3ms/step - loss: 0.0143 - accuracy: 0.
9974
Epoch 45/100
2100/2100 [==============================] - 4s 3ms/step - loss: 0.0113 - accuracy: 0.
9976
Epoch 46/100
2100/2100 [==============================] - 4s 3ms/step - loss: 0.0064 - accuracy: 0.
9985
Epoch 47/100
2100/2100 [==============================] - 5s 3ms/step - loss: 0.0073 - accuracy: 0.
9988
Epoch 48/100
2100/2100 [==============================] - 5s 3ms/step - loss: 0.0197 - accuracy: 0.
9971
Epoch 49/100
2100/2100 [==============================] - 5s 3ms/step - loss: 0.0069 - accuracy: 0.
9988
Epoch 50/100
2100/2100 [==============================] - 5s 3ms/step - loss: 0.0075 - accuracy: 0.
9984
Epoch 51/100
2100/2100 [==============================] - 4s 3ms/step - loss: 0.0103 - accuracy: 0.
9981
Epoch 52/100
2100/2100 [==============================] - 4s 3ms/step - loss: 0.0068 - accuracy: 0.
9988
Epoch 53/100
2100/2100 [==============================] - 5s 3ms/step - loss: 0.0117 - accuracy: 0.
9977
Epoch 54/100
2100/2100 [==============================] - 4s 3ms/step - loss: 0.0098 - accuracy: 0.
9983
Epoch 55/100
2100/2100 [==============================] - 6s 3ms/step - loss: 0.0120 - accuracy: 0.
9983
Epoch 56/100
2100/2100 [==============================] - 5s 3ms/step - loss: 0.0098 - accuracy: 0.
9984
Epoch 57/100
2100/2100 [==============================] - 5s 3ms/step - loss: 0.0090 - accuracy: 0.
9986
Epoch 58/100
2100/2100 [==============================] - 4s 3ms/step - loss: 0.0121 - accuracy: 0.
```

```
2100/2100 [==============================] - 4s 2ms/step - loss: 0.0133 - accuracy: 0.
9970
Epoch 59/100
2100/2100 [==============================] - 4s 2ms/step - loss: 0.0080 - accuracy: 0.
9989
Epoch 60/100
2100/2100 [==============================] - 5s 2ms/step - loss: 0.0093 - accuracy: 0.
9985
Epoch 61/100
2100/2100 [==============================] - 4s 2ms/step - loss: 0.0129 - accuracy: 0.
9983
Epoch 62/100
2100/2100 [==============================] - 6s 3ms/step - loss: 0.0048 - accuracy: 0.
9992
Epoch 63/100
2100/2100 [==============================] - 5s 2ms/step - loss: 0.0179 - accuracy: 0.
9975
Epoch 64/100
2100/2100 [==============================] - 5s 2ms/step - loss: 0.0123 - accuracy: 0.
9980
Epoch 65/100
2100/2100 [==============================] - 4s 2ms/step - loss: 0.0036 - accuracy: 0.
9993
Epoch 66/100
2100/2100 [==============================] - 4s 2ms/step - loss: 0.0086 - accuracy: 0.
9990
Epoch 67/100
2100/2100 [==============================] - 5s 2ms/step - loss: 0.0109 - accuracy: 0.
9982
Epoch 68/100
2100/2100 [==============================] - 4s 2ms/step - loss: 0.0068 - accuracy: 0.
9986
Epoch 69/100
2100/2100 [==============================] - 6s 3ms/step - loss: 0.0045 - accuracy: 0.
9991
Epoch 70/100
2100/2100 [==============================] - 4s 2ms/step - loss: 0.0150 - accuracy: 0.
9974
Epoch 71/100
2100/2100 [==============================] - 5s 2ms/step - loss: 0.0083 - accuracy: 0.
9984
Epoch 72/100
2100/2100 [==============================] - 4s 2ms/step - loss: 0.0084 - accuracy: 0.
9985
Epoch 73/100
2100/2100 [==============================] - 4s 2ms/step - loss: 0.0093 - accuracy: 0.
9984
Epoch 74/100
2100/2100 [==============================] - 5s 2ms/step - loss: 0.0090 - accuracy: 0.
9988
Epoch 75/100
2100/2100 [==============================] - 4s 2ms/step - loss: 0.0017 - accuracy: 0.
9995
Epoch 76/100
2100/2100 [==============================] - 6s 3ms/step - loss: 0.0130 - accuracy: 0.
9983
Epoch 77/100
2100/2100 [==============================] - 5s 2ms/step - loss: 0.0040 - accuracy: 0.
9994
Epoch 78/100
2100/2100 [==============================] - 5s 2ms/step - loss: 0.0223 - accuracy: 0.
9987
Epoch 79/100
2100/2100 [==============================] - 4s 2ms/step - loss: 0.0113 - accuracy: 0.
9990
Epoch 80/100
2100/2100 [==============================] - 4s 2ms/step - loss: 0.0083 - accuracy: 0.
9986
Epoch 81/100
2100/2100 [==============================] - 5s 2ms/step - loss: 0.0157 - accuracy: 0.
9987
Epoch 82/100
2100/2100 [==============================] - 4s 2ms/step - loss: 0.0058 - accuracy: 0.
```

```
9991
Epoch 83/100
2100/2100 [==============================] - 5s 3ms/step - loss: 0.0115 - accuracy: 0.
9963
Epoch 84/100
2100/2100 [==============================] - 6s 3ms/step - loss: 0.0129 - accuracy: 0.
9963
Epoch 85/100
2100/2100 [==============================] - 5s 3ms/step - loss: 0.0102 - accuracy: 0.
9964
Epoch 86/100
2100/2100 [==============================] - 4s 3ms/step - loss: 0.0106 - accuracy: 0.
9962
Epoch 87/100
2100/2100 [==============================] - 4s 3ms/step - loss: 0.0133 - accuracy: 0.
9964
Epoch 88/100
2100/2100 [==============================] - 5s 3ms/step - loss: 0.0130 - accuracy: 0.
9963
Epoch 89/100
2100/2100 [==============================] - 4s 3ms/step - loss: 0.0067 - accuracy: 0.
9985
Epoch 90/100
2100/2100 [==============================] - 5s 3ms/step - loss: 0.0081 - accuracy: 0.
9989
Epoch 91/100
2100/2100 [==============================] - 5s 3ms/step - loss: 0.0155 - accuracy: 0.
9963
Epoch 92/100
2100/2100 [==============================] - 5s 3ms/step - loss: 0.0120 - accuracy: 0.
9963
Epoch 93/100
2100/2100 [==============================] - 4s 3ms/step - loss: 0.0069 - accuracy: 0.
9980
Epoch 94/100
2100/2100 [==============================] - 4s 3ms/step - loss: 0.0222 - accuracy: 0.
9968
Epoch 95/100
2100/2100 [==============================] - 5s 3ms/step - loss: 0.0090 - accuracy: 0.
9967
Epoch 96/100
2100/2100 [==============================] - 4s 3ms/step - loss: 0.0095 - accuracy: 0.
9963
Epoch 97/100
2100/2100 [==============================] - 5s 3ms/step - loss: 0.0147 - accuracy: 0.
9965
Epoch 98/100
2100/2100 [==============================] - 5s 3ms/step - loss: 0.0144 - accuracy: 0.
9960
Epoch 99/100
2100/2100 [==============================] - 6s 3ms/step - loss: 0.0113 - accuracy: 0.
9963
Epoch 100/100
2100/2100 [==============================] - 4s 3ms/step - loss: 0.0175 - accuracy: 0.
9975
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 784)               0
_________________________________________________________________
dense (Dense)                (None, 512)               401920
_________________________________________________________________
dense_1 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_2 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_3 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_4 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_5 (Dense)              (None, 16)                528
```

```
_________________________________________________________________
dense_4 (Dense)              (None, 10)                170
=================================================================
Total params: 577,178
Trainable params: 577,178
Non-trainable params: 0
_________________________________________________________________
```

In [14]:

```
y_pred = model.predict(X_test)
y_pred = y_pred.argmax(axis=-1)
y_pred
```

Out[14]:

```
array([2, 0, 9, ..., 3, 9, 2])
```

In [14]:

```
#SUBMITTING PREDICTIONS
sub=pd.read_csv('/kaggle/input/digit-recognizer/sample_submission.csv', header='infer')
sub['Label']=y_pred
sub.to_csv('submission.csv', index=False)
```

In [14]:

```
# prediction = model.predict(X_test)
# prediction = prediction.argmax(axis=-1)
# prediction
```

In [16]:

```
# print('accuracy score for the MNIST classification of the dataset is :',metrics.accuracy_
score(y_test, prediction, normalize = True))
# print(metrics.classification_report(y_test, prediction))
# print(metrics.confusion_matrix(y_test, prediction))
```

In [ ]:

```

```

In [17]:

```
# output = pd.DataFrame({"ImageId": [i for i in range(1,len(prediction)+1)], "Label": predi
ction})
# print(output)
# output.to_csv('[illegible]', index=False)
```


Continue exploring

Data
1 input and 1 output →

Logs
598.0 second run - successful →

Comments
0 comments →