Neural Network Project Demo

February 2019

1 Topic

In this project I’ll build a neural network and train it on a GPU-enabled server to recognize handwritten digits (from 0 to 9) using the MNIST dataset. At the end, I made a GUI using Tkinter to show my results.

2 Dataset

MNIST contains 70,000 images of handwritten digits: 60,000 for training and 10,000 for testing. The images are grayscale, 28x28 pixels, and centered to reduce preprocessing and get started quicker.
3  DNN Model

3.1  Architecture

This model is built on a linear stack of layers with the sequential model. There are two hidden layers in total. All of them are dense layers. A snippet of code of constructing the model/network is shown as follow:

```python
model = Sequential()
model.add(Dense(512, input_shape=(784,)))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(10))
model.add(Activation('softmax'))
```

Listing 1: Architecture

The above shows a graph of the architecture used in this project. Notice in the code section I added dropout as a way to prevent overfitting.

3.2  Input: Shape of Tensor

X_train shape is (60000, 28, 28)
X_test shape is (10000, 28, 28)
We transform it into a float32 array of shape (60000, 28 * 28).

3.3  Output: Shape of Tensor

y_train shape is (60000,)
y_test shape is (10000,)
3.4 Shape of Output Tensor for Each Layer

Output of first hidden layer: (60000, 512)
Output of second hidden layer: (60000, 512)
Output of the output layer: (60000, 10)

4 Hyperparameters

4.1 List of Hyperparameters

In this project, there are three hyperparameters I choose to tune: batch size; epochs and dropout.

4.2 Range of Value of Hyperparameters Tried

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Range of Value Tried</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch Size</td>
<td>32, 64 and 128</td>
</tr>
<tr>
<td>Epochs</td>
<td>10 – 40</td>
</tr>
<tr>
<td>Dropout</td>
<td>0.1 – 0.5</td>
</tr>
</tbody>
</table>

4.3 Optimal Hyperparameters Found

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch Size</td>
<td>128</td>
</tr>
<tr>
<td>Epochs</td>
<td>20</td>
</tr>
<tr>
<td>Dropout</td>
<td>0.2</td>
</tr>
</tbody>
</table>

5 Annotated Code

```python
import numpy as np
import os
from keras.datasets import mnist
from keras.models import Sequential, load_model
from keras.layers.core import Dense, Dropout, Activation
from keras.utils import np_utils

(X_train, y_train), (X_test, y_test) = mnist.load_data()

X_train = X_train.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')

X_train /= 255
X_test /= 255

n_classes = 10
Y_train = np_utils.to_categorical(y_train, n_classes)
Y_test = np_utils.to_categorical(y_test, n_classes)

model = Sequential()
model.add(Dense(512, input_shape=(784,)))
model.add(Activation('relu'))
```
model.add(Dropout(0.2))
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(10))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer='adam')

# training
history = model.fit(X_train, Y_train,
                      batch_size=128, epochs=20,
                      verbose=2,
                      validation_data=(X_test, Y_test))

# saving the model
save_dir = './results'
model.save(save_dir)
print('Saved trained model')

Listing 2: Training

Listing 3: Testing

mnist_model = load_model("results")
loss_and_metrics = mnist_model.evaluate(X_test, Y_test,
                                        verbose=2)

print("Test Loss", loss_and_metrics[0])
print("Test Accuracy", loss_and_metrics[1])
## 6 Training and Testing Performance

<table>
<thead>
<tr>
<th>Epoch</th>
<th>Loss</th>
<th>Accuracy</th>
<th>Validation Loss</th>
<th>Validation Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>2/20</td>
<td>0.2509</td>
<td>0.9244</td>
<td>0.1218</td>
<td>0.9593</td>
</tr>
<tr>
<td>3/20</td>
<td>0.1012</td>
<td>0.9687</td>
<td>0.0841</td>
<td>0.9735</td>
</tr>
<tr>
<td>4/20</td>
<td>0.0701</td>
<td>0.9784</td>
<td>0.0736</td>
<td>0.9770</td>
</tr>
<tr>
<td>5/20</td>
<td>0.0556</td>
<td>0.9816</td>
<td>0.0629</td>
<td>0.9810</td>
</tr>
<tr>
<td>6/20</td>
<td>0.0476</td>
<td>0.9850</td>
<td>0.0722</td>
<td>0.9783</td>
</tr>
<tr>
<td>7/20</td>
<td>0.0382</td>
<td>0.9876</td>
<td>0.0646</td>
<td>0.9812</td>
</tr>
<tr>
<td>8/20</td>
<td>0.0335</td>
<td>0.9889</td>
<td>0.0673</td>
<td>0.9822</td>
</tr>
<tr>
<td>9/20</td>
<td>0.0335</td>
<td>0.9886</td>
<td>0.0685</td>
<td>0.9817</td>
</tr>
<tr>
<td>10/20</td>
<td>0.0305</td>
<td>0.9900</td>
<td>0.0745</td>
<td>0.9793</td>
</tr>
<tr>
<td>11/20</td>
<td>0.0260</td>
<td>0.9913</td>
<td>0.0677</td>
<td>0.9824</td>
</tr>
<tr>
<td>12/20</td>
<td>0.0250</td>
<td>0.9916</td>
<td>0.0736</td>
<td>0.9807</td>
</tr>
<tr>
<td>13/20</td>
<td>0.0220</td>
<td>0.9927</td>
<td>0.0698</td>
<td>0.9836</td>
</tr>
<tr>
<td>14/20</td>
<td>0.0208</td>
<td>0.9930</td>
<td>0.0816</td>
<td>0.9808</td>
</tr>
<tr>
<td>15/20</td>
<td>0.0209</td>
<td>0.9928</td>
<td>0.0774</td>
<td>0.9825</td>
</tr>
<tr>
<td>16/20</td>
<td>0.0184</td>
<td>0.9940</td>
<td>0.0725</td>
<td>0.9837</td>
</tr>
<tr>
<td>17/20</td>
<td>0.0173</td>
<td>0.9944</td>
<td>0.0676</td>
<td>0.9839</td>
</tr>
<tr>
<td>18/20</td>
<td>0.0175</td>
<td>0.9942</td>
<td>0.0718</td>
<td>0.9839</td>
</tr>
<tr>
<td>19/20</td>
<td>0.0167</td>
<td>0.9945</td>
<td>0.0772</td>
<td>0.9824</td>
</tr>
<tr>
<td>20/20</td>
<td>0.0166</td>
<td>0.9941</td>
<td>0.0760</td>
<td>0.9836</td>
</tr>
</tbody>
</table>

('Test Loss', 0.08080737559018634)
('Test Accuracy', 0.9833)
7 Instruction on how to test the trained DNN and how to use the GUI

7.1 Install Dependencies
- Python 3
- Tkinter
- Keras
- Tensorflow
- Numpy,Scipy

7.2 Execution
Run classifier.py file and you can play around with the demo.

7.3 Code
Attached.

7.4 Video Link
Video for the GUI Demo is here.
(https://www.youtube.com/watch?v=0kXoEgENPTAfeature=youtu.be)