q2

November 14, 2019

0.1 Q2a. (40 points) In this question, you are required to implement (using Keras) a 10-output CNN with the following layers:

1. 16 channels of 2×2 convolution, with ReLU activation;
2. max-pooling layer with stride 2;
3. 16 channels of 2×2 convolution, with ReLU activation;
4. max-pooling layer with stride 2;
5. fully connected layer with 120 ReLU hidden units;
6. another fully connected layer with 64 ReLU hidden units.

In this experiment, we use the MNIST (https://hkustconnect-my.sharepoint.com/:u:/g/personal/hzhangal_connect_ust_hk/ERzaZJwExepPlf92u_1cCPABLyuC21lBggcZ9GHx0mpyPQ?e=YklceJ) dataset. The first column is the class label. The other columns are the intensity values for each individual pixel in each MNIST image. Each student will have his/her own test set (which is based on your student id). Run your code using adam as the optimizer, train your model for 10 epochs on the training set, and report the accuracy on your test set.

[1]:
```
import numpy as np
import keras
import pandas as pd
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
np.random.seed(0)
from keras import backend as K
```

```
Using TensorFlow backend.
```

[16]:
```
#loading data
train = np.genfromtxt('asgn2_data/train.csv', delimiter = ',')
test = np.genfromtxt('asgn2_data/20380937.csv', delimiter = ',')
display(train.shape)
```

```
(1581, 785)
```

[26]:
```
x_train = train[1:,1:].reshape((-1, 28, 28, 1))
x_test = test[1:,1:].reshape((-1, 28, 28, 1))
y_train = train[1:, 0]
```

```
y_test = test[1:,0]
```

[ ]:
```
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)
```

[27]:
```
# Building the model
"""
1. 16 channels of 2 × 2 convolution, with ReLU activation; 2. max-pooling layer␣
 ↪with stride 2;
3. 16 channels of 2 × 2 convolution, with ReLU activation;
1
4. max-pooling layer with stride 2;
5. fully connected layer with 120 ReLU hidden units;
6. another fully connected layer with 64 ReLU hidden units"""
model = Sequential()

model.add(Conv2D(16, kernel_size=(2, 2), activation='relu', input_shape=(28,␣
 ↪28, 1)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(16, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())

model.add(Dense(120, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

[28]:
```
# training
model.compile(loss=keras.losses.sparse_categorical_crossentropy,
 optimizer=keras.optimizers.Adam(),
 metrics=['accuracy'])
```

[29]:
```
model.fit(x_train, y_train,
          batch_size=128,
          epochs=10,
          verbose=1,
          validation_data=(x_test, y_test))
```

```
Train on 1580 samples, validate on 210 samples
Epoch 1/10
1580/1580 [==============================] - 1s 588us/step - loss: 2.0321 -
accuracy: 0.3994 - val_loss: 1.5831 - val_accuracy: 0.6714
```

```
Epoch 2/10
1580/1580 [==============================] - 0s 272us/step - loss: 1.1609 -
accuracy: 0.7291 - val_loss: 0.7597 - val_accuracy: 0.7905
Epoch 3/10
1580/1580 [==============================] - 0s 291us/step - loss: 0.6039 -
accuracy: 0.8228 - val_loss: 0.5039 - val_accuracy: 0.8619
Epoch 4/10
1580/1580 [==============================] - 0s 279us/step - loss: 0.4219 -
accuracy: 0.8772 - val_loss: 0.4423 - val_accuracy: 0.8762
Epoch 5/10
1580/1580 [==============================] - 0s 285us/step - loss: 0.3257 -
accuracy: 0.9038 - val_loss: 0.3181 - val_accuracy: 0.9143
Epoch 6/10
1580/1580 [==============================] - 1s 352us/step - loss: 0.2637 -
accuracy: 0.9259 - val_loss: 0.2813 - val_accuracy: 0.9238
Epoch 7/10
1580/1580 [==============================] - 0s 273us/step - loss: 0.2224 -
accuracy: 0.9361 - val_loss: 0.2579 - val_accuracy: 0.9238
Epoch 8/10
1580/1580 [==============================] - 0s 285us/step - loss: 0.1874 -
accuracy: 0.9494 - val_loss: 0.2659 - val_accuracy: 0.9190
Epoch 9/10
1580/1580 [==============================] - 0s 280us/step - loss: 0.1713 -
accuracy: 0.9525 - val_loss: 0.2240 - val_accuracy: 0.9429
Epoch 10/10
1580/1580 [==============================] - 0s 281us/step - loss: 0.1397 -
accuracy: 0.9652 - val_loss: 0.1883 - val_accuracy: 0.9429
```

[29]:
```
<keras.callbacks.callbacks.History at 0x1350e6748>
```

[30]:
```
#testing
loss, acc = model.evaluate(x_test, y_test, verbose=0)
print("accuracy for the given CNN architecture: ", acc)
```

```
accuracy for the given CNN architecture:  0.9428571462631226
```

#Changing parameters for part b

[34]:
```
# model
model = Sequential()
model.add(Conv2D(16, kernel_size=(4, 4), activation='relu',input_shape=(28, 28,␣
↪1))) # unknown model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(16, (4, 4), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(120, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

[35]:
```
model.compile(loss=keras.losses.sparse_categorical_crossentropy,␣
 ↪optimizer=keras.optimizers.Adam(),
metrics=['accuracy'])
```

[36]:
```
model.fit(x_train, y_train, epochs=10,
verbose=2, shuffle=False)
```

```
Epoch 1/10
 - 2s - loss: 1.0619 - accuracy: 0.6804
Epoch 2/10
 - 2s - loss: 0.3163 - accuracy: 0.9032
Epoch 3/10
 - 2s - loss: 0.1680 - accuracy: 0.9532
Epoch 4/10
 - 2s - loss: 0.1090 - accuracy: 0.9633
Epoch 5/10
 - 2s - loss: 0.0660 - accuracy: 0.9791
Epoch 6/10
 - 2s - loss: 0.0441 - accuracy: 0.9861
Epoch 7/10
 - 1s - loss: 0.0301 - accuracy: 0.9943
Epoch 8/10
 - 2s - loss: 0.0242 - accuracy: 0.9930
Epoch 9/10
 - 1s - loss: 0.0179 - accuracy: 0.9949
Epoch 10/10
 - 1s - loss: 0.0099 - accuracy: 0.9962
```

[36]:
```
<keras.callbacks.callbacks.History at 0x1360d5f60>
```

[39]:
```
loss, acr = model.evaluate(x_test, y_test)
print("accuracy for model in part b after making changes: ", acr)
```

```
210/210 [==============================] - 0s 364us/step
accuracy for model in part b after making changes:  0.9523809552192688
```