# CNN_MNIST

September 15, 2020

[ ]:

```python
# Credits: https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py


from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K

batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```

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

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
58368/60000 [============================>.] - ETA: 4s - loss: 0.2633 - acc:
0.919560000/60000 [==============================] - 163s 3ms/step - loss:
0.2595 - acc: 0.9206 - val_loss: 0.0728 - val_acc: 0.9760
Epoch 2/12
56704/60000 [===========================>..] - ETA: 8s - loss: 0.0885 - acc:
0.973160000/60000 [==============================] - 163s 3ms/step - loss:
0.0875 - acc: 0.9734 - val_loss: 0.0412 - val_acc: 0.9855
Epoch 3/12
56064/60000 [===========================>..] - ETA: 10s - loss: 0.0653 - acc:
0.980360000/60000 [==============================] - 163s 3ms/step - loss:
0.0654 - acc: 0.9804 - val_loss: 0.0335 - val_acc: 0.9884
Epoch 4/12
```

```
55808/60000 [=========================>…] - ETA: 11s - loss: 0.0541 - acc:
0.983760000/60000 [==============================] - 172s 3ms/step - loss:
0.0543 - acc: 0.9838 - val_loss: 0.0311 - val_acc: 0.9896
Epoch 5/12
55680/60000 [=========================>…] - ETA: 11s - loss: 0.0474 - acc:
0.985760000/60000 [==============================] - 161s 3ms/step - loss:
0.0470 - acc: 0.9856 - val_loss: 0.0289 - val_acc: 0.9893
Epoch 6/12
55680/60000 [=========================>…] - ETA: 11s - loss: 0.0405 - acc:
0.987360000/60000 [==============================] - 163s 3ms/step - loss:
0.0402 - acc: 0.9874 - val_loss: 0.0280 - val_acc: 0.9900
Epoch 7/12
55680/60000 [=========================>…] - ETA: 11s - loss: 0.0366 - acc:
0.988260000/60000 [==============================] - 164s 3ms/step - loss:
0.0374 - acc: 0.9881 - val_loss: 0.0279 - val_acc: 0.9909
Epoch 8/12
55680/60000 [=========================>…] - ETA: 11s - loss: 0.0347 - acc:
0.989560000/60000 [==============================] - 163s 3ms/step - loss:
0.0341 - acc: 0.9897 - val_loss: 0.0265 - val_acc: 0.9919
Epoch 9/12
55680/60000 [=========================>…] - ETA: 11s - loss: 0.0318 - acc:
0.990160000/60000 [==============================] - 162s 3ms/step - loss:
0.0321 - acc: 0.9900 - val_loss: 0.0273 - val_acc: 0.9916
Epoch 10/12
55424/60000 [=========================>…] - ETA: 11s - loss: 0.0291 - acc:
0.991260000/60000 [==============================] - 162s 3ms/step - loss:
0.0292 - acc: 0.9911 - val_loss: 0.0274 - val_acc: 0.9913
Epoch 11/12
55296/60000 [=========================>…] - ETA: 12s - loss: 0.0267 - acc:
0.991660000/60000 [==============================] - 163s 3ms/step - loss:
0.0273 - acc: 0.9915 - val_loss: 0.0241 - val_acc: 0.9922
Epoch 12/12
55296/60000 [=========================>…] - ETA: 12s - loss: 0.0261 - acc:
0.991760000/60000 [==============================] - 161s 3ms/step - loss:
0.0259 - acc: 0.9917 - val_loss: 0.0242 - val_acc: 0.9927
Test loss: 0.024196199060090292
Test accuracy: 0.9927
```

[ ]:

[ ]:

```python
# Credits: https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py



from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K

batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

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

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Epoch 1/12
469/469 [==============================] - 143s 305ms/step - loss: 2.2857 -
accuracy: 0.1107 - val_loss: 2.2550 - val_accuracy: 0.1819
Epoch 2/12
469/469 [==============================] - 143s 305ms/step - loss: 2.2397 -
accuracy: 0.1975 - val_loss: 2.1997 - val_accuracy: 0.3695
Epoch 3/12
469/469 [==============================] - 143s 305ms/step - loss: 2.1814 -
accuracy: 0.2905 - val_loss: 2.1270 - val_accuracy: 0.5413
Epoch 4/12
294/469 [==================>…] - ETA: 51s - loss: 2.1221 - accuracy:
0.3666
```

[ ]: