PS7

January 23, 2023

0.1 LeNet-5 implementation

[1]:
```
import tensorflow as tf
import tensorflow.keras as keras
from tensorflow.keras.datasets import mnist
```

0.2 MNIST data loading

[2]:
```
(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train, X_test = X_train/255.0, X_test/255.0
```

0.3 Model construction

[3]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPool2D
```

[4]:
```
model_lenet = Sequential()

#1st stack ([Conv]+[ReLU]+[Pool])
model_lenet.add(Conv2D(input_shape=(28,28,1),
                       kernel_size=(5,5),
                       strides=(1,1),
                       filters=32,
                       padding='same',
                       activation='relu'
                      ))
model_lenet.add(MaxPool2D(pool_size=(2,2),
                          strides=(2,2),
                          padding='valid'))

#2nd stack ([Conv]+[ReLU]+[Pool])
model_lenet.add(Conv2D(kernel_size=(5,5),
                       strides=(1,1),
                       filters=48,
                       padding='same',
```

```
                        activation='relu'
                       ))
model_lenet.add(MaxPool2D(pool_size=(2,2),
                          strides=(2,2),
                          padding='valid'))

# Three fully Connected Layers
model_lenet.add(Flatten())
model_lenet.add(Dense(256,activation='relu' ))
model_lenet.add(Dense(84,activation='relu'))
model_lenet.add(Dense(10,activation='softmax'))

model_lenet.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 32)        832

 max_pooling2d (MaxPooling2D  (None, 14, 14, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 14, 14, 48)        38448

 max_pooling2d_1 (MaxPooling  (None, 7, 7, 48)         0
 2D)

 flatten (Flatten)           (None, 2352)              0

 dense (Dense)               (None, 256)               602368

 dense_1 (Dense)             (None, 84)                21588

 dense_2 (Dense)             (None, 10)                850

=================================================================
Total params: 664,086
Trainable params: 664,086
Non-trainable params: 0
_________________________________________________________________
```

0.4 Compile

[5]:
```
from tensorflow.keras.optimizers import Adam

opt = Adam(learning_rate = 0.001,
           beta_1 = 0.9,
           beta_2 = 0.999)

model_lenet.compile(optimizer = opt,
                    loss='sparse_categorical_crossentropy',
                    metrics=['acc'])
```

0.5 Training & evaluation

[6]:
```
# Conversion from 3D to 4D tensor
X_train, X_test = X_train.reshape(-1,28,28,1), X_test.reshape(-1,28,28,1)
```

[7]:
```
# Training
model_lenet.fit(X_train,y_train,epochs=10)
```

```
Epoch 1/10
1875/1875 [==============================] - 32s 17ms/step - loss: 0.1206 - acc:
0.9629
Epoch 2/10
1875/1875 [==============================] - 33s 18ms/step - loss: 0.0420 - acc:
0.9870
Epoch 3/10
1875/1875 [==============================] - 33s 18ms/step - loss: 0.0288 - acc:
0.9911
Epoch 4/10
1875/1875 [==============================] - 34s 18ms/step - loss: 0.0201 - acc:
0.9938
Epoch 5/10
1875/1875 [==============================] - 34s 18ms/step - loss: 0.0175 - acc:
0.9948
Epoch 6/10
1875/1875 [==============================] - 36s 19ms/step - loss: 0.0138 - acc:
0.9956
Epoch 7/10
1875/1875 [==============================] - 38s 20ms/step - loss: 0.0117 - acc:
0.9965
Epoch 8/10
1875/1875 [==============================] - 35s 19ms/step - loss: 0.0103 - acc:
0.9968
Epoch 9/10
1875/1875 [==============================] - 35s 19ms/step - loss: 0.0094 - acc:
0.9970
```

```
Epoch 10/10
1875/1875 [==============================] - 35s 18ms/step - loss: 0.0093 - acc:
0.9975
```

[7]:
```
<keras.callbacks.History at 0x24b453b8550>
```

[8]:
```
# Evaluation
test_performance = model_lenet.evaluate(X_test, y_test)
print(test_performance)
```

```
313/313 [==============================] - 1s 4ms/step - loss: 0.0510 - acc:
0.9890
[0.05104943364858627, 0.9890000224113464]
```