main

November 7, 2022

[1]:
```
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

[2]:
```
# load train and test dataset
def load_dataset():
    # load dataset
    (trainX, trainY), (testX, testY) = mnist.load_data()
    # data pre-process
    trainX = trainX.reshape(60000, 784)
    testX = testX.reshape(10000, 784)
    trainX = trainX.astype('float32')/255.0
    testX = testX.astype('float32')/255.0
    trainY = to_categorical(trainY)
    testY = to_categorical(testY)
    return trainX, trainY, testX, testY

trainX, trainY, testX, testY = load_dataset()
print(trainX.shape, trainY.shape, testX.shape, testY.shape)
```

```
(60000, 784) (60000, 10) (10000, 784) (10000, 10)
```

[3]:
```
model = Sequential()
model.add(Dense(units=10, input_dim=784, activation="relu",␣
 ↪kernel_initializer='he_uniform'))
model.add(Dense(units=10, activation="relu", kernel_initializer='he_uniform'))
model.add(Dense(units=10, activation="relu", kernel_initializer='he_uniform'))
model.add(Dense(units=10, activation="softmax",␣
 ↪kernel_initializer='he_uniform'))
model.compile(optimizer="adam", loss="categorical_crossentropy",␣
 ↪metrics=["accuracy"])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
```

```
dense (Dense)                (None, 10)                7850
_________________________________________________________________
dense_1 (Dense)              (None, 10)                110
_________________________________________________________________
dense_2 (Dense)              (None, 10)                110
_________________________________________________________________
dense_3 (Dense)              (None, 10)                110
=================================================================
Total params: 8,180
Trainable params: 8,180
Non-trainable params: 0
_________________________________________________________________
2022-11-07 11:53:23.112997: I tensorflow/core/platform/cpu_feature_guard.cc:145]
This TensorFlow binary is optimized with Intel(R) MKL-DNN to use the following
CPU instructions in performance critical operations:  SSE4.1 SSE4.2
To enable them in non-MKL-DNN operations, rebuild TensorFlow with the
appropriate compiler flags.
2022-11-07 11:53:23.113316: I
tensorflow/core/common_runtime/process_util.cc:115] Creating new thread pool
with default inter op setting: 8. Tune using inter_op_parallelism_threads for
best performance.
```

[4]:
```
train_history =model.fit(x=trainX, y=trainY, validation_split=0.25, epochs=20,␣
↪batch_size=200, verbose=2)
```

```
Train on 45000 samples, validate on 15000 samples
Epoch 1/20
45000/45000 - 1s - loss: 1.4960 - accuracy: 0.4906 - val_loss: 0.8788 -
val_accuracy: 0.7063
Epoch 2/20
45000/45000 - 1s - loss: 0.5890 - accuracy: 0.8289 - val_loss: 0.4298 -
val_accuracy: 0.8783
Epoch 3/20
45000/45000 - 1s - loss: 0.3994 - accuracy: 0.8895 - val_loss: 0.3606 -
val_accuracy: 0.8968
Epoch 4/20
45000/45000 - 1s - loss: 0.3438 - accuracy: 0.9042 - val_loss: 0.3318 -
val_accuracy: 0.9051
Epoch 5/20
45000/45000 - 1s - loss: 0.3154 - accuracy: 0.9114 - val_loss: 0.3088 -
val_accuracy: 0.9117
Epoch 6/20
45000/45000 - 1s - loss: 0.2951 - accuracy: 0.9164 - val_loss: 0.2988 -
val_accuracy: 0.9148
Epoch 7/20
45000/45000 - 1s - loss: 0.2832 - accuracy: 0.9188 - val_loss: 0.2875 -
val_accuracy: 0.9207
```

```
Epoch 8/20
45000/45000 - 1s - loss: 0.2716 - accuracy: 0.9235 - val_loss: 0.2843 -
val_accuracy: 0.9203
Epoch 9/20
45000/45000 - 1s - loss: 0.2631 - accuracy: 0.9246 - val_loss: 0.2765 -
val_accuracy: 0.9235
Epoch 10/20
45000/45000 - 1s - loss: 0.2543 - accuracy: 0.9273 - val_loss: 0.2751 -
val_accuracy: 0.9238
Epoch 11/20
45000/45000 - 1s - loss: 0.2469 - accuracy: 0.9302 - val_loss: 0.2690 -
val_accuracy: 0.9261
Epoch 12/20
45000/45000 - 1s - loss: 0.2395 - accuracy: 0.9328 - val_loss: 0.2644 -
val_accuracy: 0.9263
Epoch 13/20
45000/45000 - 1s - loss: 0.2318 - accuracy: 0.9342 - val_loss: 0.2585 -
val_accuracy: 0.9286
Epoch 14/20
45000/45000 - 1s - loss: 0.2255 - accuracy: 0.9363 - val_loss: 0.2583 -
val_accuracy: 0.9295
Epoch 15/20
45000/45000 - 1s - loss: 0.2210 - accuracy: 0.9383 - val_loss: 0.2565 -
val_accuracy: 0.9285
Epoch 16/20
45000/45000 - 1s - loss: 0.2168 - accuracy: 0.9380 - val_loss: 0.2575 -
val_accuracy: 0.9300
Epoch 17/20
45000/45000 - 1s - loss: 0.2144 - accuracy: 0.9394 - val_loss: 0.2566 -
val_accuracy: 0.9283
Epoch 18/20
45000/45000 - 1s - loss: 0.2086 - accuracy: 0.9413 - val_loss: 0.2555 -
val_accuracy: 0.9304
Epoch 19/20
45000/45000 - 1s - loss: 0.2074 - accuracy: 0.9420 - val_loss: 0.2562 -
val_accuracy: 0.9293
Epoch 20/20
45000/45000 - 1s - loss: 0.2041 - accuracy: 0.9429 - val_loss: 0.2507 -
val_accuracy: 0.9317
```

[5]:
```
scores = model.evaluate(testX, testY, verbose=0)
print("Test acc=",scores[1])
```

```
Test acc= 0.9305
```