CNN ORG Mode

Jesus Sierralaya

April 12, 2024

Contents

1 Load libraries

```python
import os; os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
import matplotlib.pyplot as plt
import numpy as np
import keras
from keras import layers
import tensorflow as tf
import argparse
```

```python
from sklearn.preprocessing import LabelBinarizer
from keras.datasets import mnist
```

2 Set parameters

```python
num_classes = 10
input_shape = (28, 28, 1)
```

3 Pre-process

```python
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()

x_train = x_train.astype("float32") / 255
x_test = x_test.astype("float32") / 255

x_train = np.expand_dims(x_train, -1)
x_test = np.expand_dims(x_test, -1)
print("x_train shape:", x_train.shape)
print(x_train.shape[0], "train samples")
print(x_test.shape[0], "test samples")

label_binarizer = LabelBinarizer()
y_train = label_binarizer.fit_transform(y_train)
y_test = label_binarizer.fit_transform(y_test)
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
```

4 Create the model

```
model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3),
        ↪  activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3),
        ↪  activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)

model.summary()
```

5 Set the parameters, compile and train the model

```
batch_size = 2# 128
epochs = 2

model.compile(loss="categorical_crossentropy",
↪  optimizer="adam", metrics=["accuracy"])

# history = model.fit(x_train, y_train,
↪  batch_size=batch_size, epochs=epochs,
↪  validation_split=0.1, verbose = False)
```

6 Evaluate the performance of the trained model

```
# test_loss, test_accuracy = model.evaluate(x_test, y_test, verbose=2)
# print(f"Test accuracy: {test_accuracy * 100:.2f}%")
# f"{test_accuracy * 100:.2f}%"
```

The test accuracy is ==.

7 Learning graph

```
# plt.figure()
# plt.plot(history.history['accuracy'], label='Training Accuracy')
# plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
# plt.title('Training and Validation Accuracy')
# plt.xlabel('Epoch')
# plt.ylabel('Accuracy')
# plt.legend()
```