

CNN ORG Mode

Jesus Sierralaya

April 11, 2024

Contents

1	Load libraries	1
2	Set parameters	2
3	Pre-process	2
4	Create the model	2
5	Set the parameters, compile and train the model	4
6	Evaluate the performance of the trained model	4
7	Learning graph	4

1 Load libraries

```
1 import os; os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import keras
5 from keras import layers
6 import tensorflow as tf
7 import argparse
8 from sklearn.preprocessing import LabelBinarizer
9 from keras.datasets import mnist
```

2 Set parameters

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

3 Pre-process

```
1 (x_train, y_train), (x_test, y_test) =
   ↪ keras.datasets.mnist.load_data()
2
3 x_train = x_train.astype("float32") / 255
4 x_test = x_test.astype("float32") / 255
5
6 x_train = np.expand_dims(x_train, -1)
7 x_test = np.expand_dims(x_test, -1)
8 print("x_train shape:", x_train.shape)
9 print(x_train.shape[0], "train samples")
10 print(x_test.shape[0], "test samples")
11
12 label_binarizer = LabelBinarizer()
13 y_train = label_binarizer.fit_transform(y_train)
14 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

```
1 model = keras.Sequential(
2     [
3         keras.Input(shape=input_shape),
4         layers.Conv2D(32, kernel_size=(3, 3),
5             ↪ activation="relu"),
6         layers.MaxPooling2D(pool_size=(2, 2)),
```

```

6         layers.Conv2D(64, kernel_size=(3, 3),
7           ↪ activation="relu"),
8         layers.MaxPooling2D(pool_size=(2, 2)),
9         layers.Flatten(),
10        layers.Dropout(0.5),
11        layers.Dense(num_classes, activation="softmax"),
12    ]
13 )
14 model.summary()

```

Model: "sequential_15"

Layer (type)	Output Shape	Param #
conv2d_30 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_30 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_31 (Conv2D)	(None, 11, 11, 64)	18496
max_pooling2d_31 (MaxPooling2D)	(None, 5, 5, 64)	0
flatten_15 (Flatten)	(None, 1600)	0
dropout_15 (Dropout)	(None, 1600)	0
dense_15 (Dense)	(None, 10)	16010
Total params: 34826 (136.04 KB)		
Trainable params: 34826 (136.04 KB)		
Non-trainable params: 0 (0.00 Byte)		

5 Set the parameters, compile and train the model

```
1 batch_size = 128
2 epochs = 10
3
4 model.compile(loss="categorical_crossentropy",
5               ↪ optimizer="adam", metrics=["accuracy"])
6
7 history = model.fit(x_train, y_train, batch_size=batch_size,
8                   ↪ epochs=epochs, validation_split=0.1, verbose = False)
```

6 Evaluate the performance of the trained model

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

The test accuracy is 99.01%.

7 Learning graph

```
1 plt.figure()
2 plt.plot(history.history['accuracy'], label='Training
3     ↪ Accuracy')
4 plt.plot(history.history['val_accuracy'], label='Validation
5     ↪ Accuracy')
6 plt.title('Training and Validation Accuracy')
7 plt.xlabel('Epoch')
8 plt.ylabel('Accuracy')
9 plt.legend()
```

