cnn-dl

April 17, 2023

[18]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import EarlyStopping
```

[3]:
```
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [==============================] - 3s 0us/step
```

[4]:
```
x_train = x_train.astype("float32") / 255.0
x_test = x_test.astype("float32") / 255.0
```

[5]:
```
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

[6]:
```
model = keras.Sequential(
    [
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu",␣
 ↪input_shape=(32, 32, 3)),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dense(10, activation="softmax"),
    ]
)
```

0.0.1 1)Method

[ ]:
```
sgd = keras.optimizers.SGD(lr=0.01)
model.compile(loss="categorical_crossentropy", optimizer=sgd,␣
 ↪metrics=["accuracy"])
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate`
or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.SGD.
```

```
[ ]: model.fit(x_train, y_train, batch_size=64, epochs=15, validation_data=(x_test,␣
      ↪y_test))
```

```
Epoch 1/15
782/782 [==============================] - 109s 89ms/step - loss: 0.5534 -
accuracy: 0.8155 - val_loss: 1.0158 - val_accuracy: 0.7016
Epoch 2/15
782/782 [==============================] - 71s 91ms/step - loss: 0.5105 -
accuracy: 0.8251 - val_loss: 1.4288 - val_accuracy: 0.6284
Epoch 3/15
782/782 [==============================] - 72s 92ms/step - loss: 0.5066 -
accuracy: 0.8282 - val_loss: 1.1546 - val_accuracy: 0.6719
Epoch 4/15
782/782 [==============================] - 71s 91ms/step - loss: 0.4943 -
accuracy: 0.8334 - val_loss: 1.0403 - val_accuracy: 0.7066
Epoch 5/15
782/782 [==============================] - 70s 90ms/step - loss: 0.4837 -
accuracy: 0.8348 - val_loss: 1.0528 - val_accuracy: 0.7006
Epoch 6/15
782/782 [==============================] - 71s 91ms/step - loss: 0.4799 -
accuracy: 0.8369 - val_loss: 0.9942 - val_accuracy: 0.7152
Epoch 7/15
782/782 [==============================] - 71s 91ms/step - loss: 0.4718 -
accuracy: 0.8390 - val_loss: 1.0083 - val_accuracy: 0.7145
Epoch 8/15
782/782 [==============================] - 74s 94ms/step - loss: 0.4720 -
accuracy: 0.8400 - val_loss: 1.2644 - val_accuracy: 0.6668
Epoch 9/15
782/782 [==============================] - 70s 89ms/step - loss: 0.4637 -
accuracy: 0.8429 - val_loss: 1.0594 - val_accuracy: 0.6983
Epoch 10/15
782/782 [==============================] - 71s 91ms/step - loss: 0.4628 -
accuracy: 0.8411 - val_loss: 1.0478 - val_accuracy: 0.7036
Epoch 11/15
782/782 [==============================] - 71s 91ms/step - loss: 0.4562 -
accuracy: 0.8442 - val_loss: 1.0478 - val_accuracy: 0.7085
Epoch 12/15
782/782 [==============================] - 69s 88ms/step - loss: 0.4535 -
accuracy: 0.8442 - val_loss: 1.0413 - val_accuracy: 0.7073
Epoch 13/15
782/782 [==============================] - 71s 91ms/step - loss: 0.4512 -
accuracy: 0.8467 - val_loss: 1.0796 - val_accuracy: 0.6994
Epoch 14/15
782/782 [==============================] - 69s 89ms/step - loss: 0.4468 -
accuracy: 0.8460 - val_loss: 1.0542 - val_accuracy: 0.7022
Epoch 15/15
782/782 [==============================] - 71s 91ms/step - loss: 0.4457 -
```

```
accuracy: 0.8477 - val_loss: 1.2613 - val_accuracy: 0.6679
```

[ ]:
```
<keras.callbacks.History at 0x7f32507626a0>
```

[ ]:
```
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f"Test accuracy: {test_acc}")
```

```
313/313 [==============================] - 4s 13ms/step - loss: 1.2613 -
accuracy: 0.6679
Test accuracy: 0.667900025844574
```

1 Finetune the hyperparameters to improve the performance of the CNN

1.0.1 2)Method

[7]:
```
model.compile(loss="categorical_crossentropy", optimizer="adam",␣
↪metrics=["accuracy"])
```

[8]:
```
model.fit(x_train, y_train, batch_size=64, epochs=10, validation_data=(x_test,␣
↪y_test))
```

```
Epoch 1/10
782/782 [==============================] - 71s 90ms/step - loss: 1.5390 -
accuracy: 0.4511 - val_loss: 1.2891 - val_accuracy: 0.5381
Epoch 2/10
782/782 [==============================] - 73s 94ms/step - loss: 1.2004 -
accuracy: 0.5808 - val_loss: 1.1184 - val_accuracy: 0.6150
Epoch 3/10
782/782 [==============================] - 71s 91ms/step - loss: 1.0717 -
accuracy: 0.6318 - val_loss: 1.0415 - val_accuracy: 0.6450
Epoch 4/10
782/782 [==============================] - 71s 91ms/step - loss: 0.9940 -
accuracy: 0.6590 - val_loss: 1.0068 - val_accuracy: 0.6544
Epoch 5/10
782/782 [==============================] - 70s 89ms/step - loss: 0.9354 -
accuracy: 0.6802 - val_loss: 0.9966 - val_accuracy: 0.6601
Epoch 6/10
782/782 [==============================] - 69s 88ms/step - loss: 0.8908 -
accuracy: 0.6948 - val_loss: 0.9450 - val_accuracy: 0.6745
Epoch 7/10
782/782 [==============================] - 72s 92ms/step - loss: 0.8478 -
accuracy: 0.7116 - val_loss: 0.9646 - val_accuracy: 0.6751
Epoch 8/10
782/782 [==============================] - 71s 91ms/step - loss: 0.8150 -
accuracy: 0.7216 - val_loss: 0.9199 - val_accuracy: 0.6901
Epoch 9/10
```

```
782/782 [==============================] - 69s 89ms/step - loss: 0.7828 -
accuracy: 0.7318 - val_loss: 0.9020 - val_accuracy: 0.6968
Epoch 10/10
782/782 [==============================] - 71s 91ms/step - loss: 0.7576 -
accuracy: 0.7376 - val_loss: 0.8907 - val_accuracy: 0.6960
```

[8]: <keras.callbacks.History at 0x7f32508f4700>

[9]:
```
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f"Test accuracy: {test_acc}")
```

```
313/313 [==============================] - 4s 13ms/step - loss: 0.8907 -
accuracy: 0.6960
Test accuracy: 0.6959999799728394
```

1.0.2 3)Method

[ ]:
```
early_stop = EarlyStopping(monitor='val_loss', patience=3)
```

[ ]:
```
model.fit(x_train, y_train, batch_size=64, epochs=100, validation_data=(x_test,␣
↪y_test), callbacks=[early_stop])
```

```
Epoch 1/100
782/782 [==============================] - 72s 92ms/step - loss: 0.1145 -
accuracy: 0.8063 - val_loss: 0.1603 - val_accuracy: 0.7100
Epoch 2/100
782/782 [==============================] - 71s 91ms/step - loss: 0.1134 -
accuracy: 0.8078 - val_loss: 0.1619 - val_accuracy: 0.7072
Epoch 3/100
782/782 [==============================] - 69s 89ms/step - loss: 0.1124 -
accuracy: 0.8100 - val_loss: 0.1592 - val_accuracy: 0.7181
Epoch 4/100
782/782 [==============================] - 73s 93ms/step - loss: 0.1106 -
accuracy: 0.8140 - val_loss: 0.1586 - val_accuracy: 0.7175
Epoch 5/100
782/782 [==============================] - 72s 92ms/step - loss: 0.1097 -
accuracy: 0.8167 - val_loss: 0.1599 - val_accuracy: 0.7152
Epoch 6/100
782/782 [==============================] - 72s 92ms/step - loss: 0.1087 -
accuracy: 0.8194 - val_loss: 0.1624 - val_accuracy: 0.7123
Epoch 7/100
782/782 [==============================] - 71s 91ms/step - loss: 0.1076 -
accuracy: 0.8194 - val_loss: 0.1618 - val_accuracy: 0.7153
```

[ ]: <keras.callbacks.History at 0x7f32506070a0>

[ ]:
```
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f"Test accuracy: {test_acc}")
```

```
313/313 [==============================] - 5s 15ms/step - loss: 0.1618 -
accuracy: 0.7153
Test accuracy: 0.7153000235557556
```

1.0.3 4)Method

[ ]:
```
model.compile(loss="binary_crossentropy", optimizer="adam",␣
↪metrics=["accuracy"])
```

[ ]:
```
model.fit(x_train, y_train, batch_size=128, epochs=15, validation_data=(x_test,␣
↪y_test))
```

```
Epoch 1/15
391/391 [==============================] - 68s 174ms/step - loss: 0.1393 -
accuracy: 0.7592 - val_loss: 0.1613 - val_accuracy: 0.7025
Epoch 2/15
391/391 [==============================] - 69s 178ms/step - loss: 0.1359 -
accuracy: 0.7648 - val_loss: 0.1574 - val_accuracy: 0.7089
Epoch 3/15
391/391 [==============================] - 67s 171ms/step - loss: 0.1336 -
accuracy: 0.7678 - val_loss: 0.1571 - val_accuracy: 0.7064
Epoch 4/15
391/391 [==============================] - 68s 175ms/step - loss: 0.1314 -
accuracy: 0.7716 - val_loss: 0.1557 - val_accuracy: 0.7092
Epoch 5/15
391/391 [==============================] - 67s 172ms/step - loss: 0.1291 -
accuracy: 0.7767 - val_loss: 0.1567 - val_accuracy: 0.7080
Epoch 6/15
391/391 [==============================] - 69s 177ms/step - loss: 0.1280 -
accuracy: 0.7798 - val_loss: 0.1539 - val_accuracy: 0.7128
Epoch 7/15
391/391 [==============================] - 67s 171ms/step - loss: 0.1259 -
accuracy: 0.7837 - val_loss: 0.1543 - val_accuracy: 0.7148
Epoch 8/15
391/391 [==============================] - 74s 190ms/step - loss: 0.1242 -
accuracy: 0.7867 - val_loss: 0.1547 - val_accuracy: 0.7168
Epoch 9/15
391/391 [==============================] - 68s 175ms/step - loss: 0.1227 -
accuracy: 0.7891 - val_loss: 0.1531 - val_accuracy: 0.7183
Epoch 10/15
391/391 [==============================] - 68s 174ms/step - loss: 0.1210 -
accuracy: 0.7919 - val_loss: 0.1553 - val_accuracy: 0.7126
Epoch 11/15
391/391 [==============================] - 69s 175ms/step - loss: 0.1196 -
accuracy: 0.7949 - val_loss: 0.1536 - val_accuracy: 0.7195
Epoch 12/15
391/391 [==============================] - 71s 182ms/step - loss: 0.1184 -
accuracy: 0.7968 - val_loss: 0.1543 - val_accuracy: 0.7148
```

```
Epoch 13/15
391/391 [==============================] - 71s 182ms/step - loss: 0.1174 -
accuracy: 0.8014 - val_loss: 0.1542 - val_accuracy: 0.7183
Epoch 14/15
391/391 [==============================] - 67s 170ms/step - loss: 0.1159 -
accuracy: 0.8040 - val_loss: 0.1558 - val_accuracy: 0.7158
Epoch 15/15
391/391 [==============================] - 67s 171ms/step - loss: 0.1150 -
accuracy: 0.8058 - val_loss: 0.1553 - val_accuracy: 0.7194
```

[ ]:
```
<keras.callbacks.History at 0x7f32506daee0>
```

[ ]:
```
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f"Test accuracy: {test_acc}")
```

```
313/313 [==============================] - 5s 17ms/step - loss: 0.1553 -
accuracy: 0.7194
Test accuracy: 0.7193999886512756
```

After applying multiple method the last method has highest accuracy so we consider this model compiler and fit for training