توضیحات و تحلیل کد شبکه CNN برای مجموعه داده CIFAR-۱۰

این تمرین شامل پیاده‌سازی یک شبکه عصبی کانولوشنی (CNN) برای طبقه‌بندی تصاویر مجموعه داده CIFAR-۱۰ است. مجموعه داده CIFAR-۱۰ شامل ۱۰ کلاس از تصاویر رنگی ۳۲ در ۳۲ پیکسلی است و برای مسائل طبقه‌بندی تصویر استفاده می‌شود.

توضیحات کد

۱ .فراخوانی مجموعه داده

```
from keras.datasets import cifar10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
print('Train: X=%s, y=%s' % (x_train.shape, y_train.shape))
print('Test: X=%s, y=%s' % (x_test.shape, y_test.shape))
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```

در این قسمت، مجموعه داده CIFAR-۱۰ بارگذاری و بررسی می‌شود. این داده شامل:

- ۵۰,۰۰۰ نمونه آموزشی
- ۱۰,۰۰۰ نمونه تست

ابعاد داده‌ها به صورت x = (۳۲، ۳۲، ۳) است که نشان‌دهنده تصاویر رنگی است.

۲ .نرمال‌سازی داده‌ها

```
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0
```

تصاویر به بازه [۰, ۱] نرمال‌سازی می‌شوند تا فرآیند آموزش بهینه‌تر انجام شود.

۳ .تبدیل برچسب‌ها به فرمت one-hot

```
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```

برچسب‌های کلاس به فرمت one-hot تبدیل می‌شوند. برای مثال، کلاس ۳ به صورت [۰, ۰, ۰, ۱, ۰, ۰, ۰, ۰, ۰, ۰] نمایش داده می‌شود.

۴ .طراحی مدل CNN

```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32,
3)),
    MaxPooling2D((2, 2)),
    Dropout(0.25),

    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Dropout(0.25),

    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(10, activation='softmax')
])
```

مدل شامل:

-دو لایه کانولوشن با تعداد فیلترهای ۳۲ و ۶۴.

-لایه‌های pooling برای کاهش ابعاد ویژگی‌ها.

-لایه Dropout برای جلوگیری از بیش‌برازش.

-لایه Fully Connected با ۱۲۸ نرون.

-خروجی با تابع فعالساز softmax برای پیش‌بینی احتمال هر کلاس.

۵ .آموزش مدل

```
history = model.fit(x_train, y_train, epochs=10, batch_size=64,
validation_split=0.2, verbose=1)
```

مدل برای ۱۰ دوره با سایز دسته ۶۴ آموزش داده می‌شود. ٪۲۰ از داده‌های آموزشی برای اعتبارسنجی استفاده می‌شود.

۶ .ارزیابی مدل

```
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print("Test accuracy:", test_acc)
```

مدل روی داده‌های تست ارزیابی شده و دقت آن گزارش می‌شود.

تحلیل نمودارها و خروجی‌ها

۱ .دقت و خطای مدل

در بخش زیر، دقت و خطای مدل رسم می‌شود:

```
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation
Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Model Accuracy')

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Model Loss')

plt.show()
```

نتایج:
-با افزایش تعداد دوره‌ها، دقت مدل بهبود می‌یابد و خطا کاهش می‌یابد.
-اگر تفاوت زیادی بین خطای آموزشی و اعتبارسنجی وجود داشته باشد، نشان‌دهنده بیش‌برازش است.

۲ .تحلیل دقت تست

```
print("Test accuracy:", test_acc)
```

دقت نهایی تست نشان‌دهنده عملکرد کلی مدل است. در اینجا دقت ما ۶۷ درصد است ، مدل را بهبود می بخشیم تا دقت افزایش یابد.

بهبود مدل

۱.تعداد لایه های شبکه را افزایش میدهیم
مدل جدید شامل:

```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    MaxPooling2D((2, 2)),
    Dropout(0.25),

    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Dropout(0.25),

    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Dropout(0.25),

    Flatten(),
    Dense(256, activation='relu'),
    Dropout(0.5),
```

سه لایه کانولوشن با تعداد فیلترهای ۳۲، ۶۴ و ۱۲۸.

لایه‌های pooling برای کاهش ابعاد ویژگی‌ها.

لایه Dropout برای جلوگیری از بیش‌برازش.

لایه Fully Connected با ۲۵۶ نرون.

خروجی softmax برای پیش‌بینی احتمال هر کلاس.

۲. تعداد داده ها را افزایش می دهیم

```
datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True
)
datagen.fit(x_train)
```

تکنیک‌های افزایش داده شامل چرخش، انتقال افقی و عمودی، و برعکس کردن تصاویر برای تنوع بخشیدن به داده‌های آموزشی استفاده می‌شود.

آموزش مدل جدید

```
opt = SGD(learning_rate=0.01, momentum=0.9)

model.compile(optimizer=opt, loss='categorical_crossentropy',
metrics=['accuracy'])

history = model.fit(datagen.flow(x_train, y_train,
batch_size=64),

epochs=30, validation_data=(x_test, y_test), verbose=1)
```

مدل با استفاده از بهینه‌ساز SGD با نرخ یادگیری متغیر و ۳۰ دوره آموزش داده می‌شود.

ارزیابی مدل جدید

```
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)

print("Test accuracy:", test_acc)
```

مدل روی داده‌های تست ارزیابی شده و دقت آن گزارش می‌شود.

۱.دقت و خطای مدل جدید

در بخش زیر، دقت و خطای مدل جدید رسم می‌شود:

```
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Model Accuracy')

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
```

```
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Model Loss')

plt.show()
```

نتایج:

- با افزایش تعداد دوره‌ها، دقت مدل بهبود می‌یابد و خطا کاهش می‌یابد.
- اگر تفاوت زیادی بین خطای آموزشی و اعتبارسنجی وجود داشته باشد، نشان‌دهنده بیش‌برازش است.

۲ .تحلیل دقت تست

```
print("Test accuracy:", test_acc)
```

دقت نهایی تست نشان‌دهنده عملکرد کلی مدل است. این بار دقت مدل ۷۲ درصد است و نشان می دهد مدل جدید بهتر است.

نتیجه‌گیری

این پروژه نشان داد که یک شبکه CNN ساده می‌تواند به صورت کارآمد برای طبقه‌بندی مجموعه داده CIFAR-۱۰ استفاده شود.