به نام خدا

توضیح بخش ج)

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.applications import Xception
from tensorflow.keras.datasets import cifar10
import matplotlib.pyplot as plt
import numpy as np
```

tensorflow به عنوان **tf**: کتابخانه **TensorFlow** را وارد می کند، که پایه و اساس ساخت و آموزش مدل های یادگیری ماشین است.

از لایه‌های **tensorflow.keras**: انواع لایه‌های خاصی را از **Keras** لود می‌کند (مانند **Dense**، **Resizing**، **GlobalAveragePooling2D**، **Dropout**).

from tensorflow.keras.applications import Xception: معماری مدل **Xception** را وارد می کند.

from tensorflow.keras.datasets import cifar10: مجموعه داده **CIFAR-10** را وارد می کند.

import matplotlib.pyplot as plt: کتابخانه **Matplotlib** را برای رسم وارد می کند (برای تجسم منحنی های از خطا و دقت استفاده می شود).

import numpy به عنوان **np**: کتابخانه **NumPy** را برای عملیات عددی وارد می کند.

```python
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

x_train, x_val = x_train[:40000], x_train[40000:]
y_train, y_val = y_train[:40000], y_train[40000:]

x_train = tf.keras.applications.xception.preprocess_input(x_train)
x_val = tf.keras.applications.xception.preprocess_input(x_val)
x_test = tf.keras.applications.xception.preprocess_input(x_test)

y_train = keras.utils.to_categorical(y_train, num_classes=10)
y_val = keras.utils.to_categorical(y_val, num_classes=10)
y_test = keras.utils.to_categorical(y_test, num_classes=10)
```
[2] Python

cifar10.load_data(): مجموعه داده CIFAR-10 را از tensorflow.keras.datasets بارگیری می کند. دو تاپل را برمی گرداند:

(x_train, y_train): داده های آموزشی (تصاویر و برچسب ها).

(x_test، y_test): داده های تست (تصاویر و برچسب ها).

❖ تقسیم کردن:

x_train، x_val = x_train[:40000] [:٤٠٠٠٠]x_train: داده‌های آموزشی اصلی را به یک مجموعه آموزشی جدید (٤٠٠٠٠ تصویر اول) و یک مجموعه اعتبارسنجی (١٠٠٠٠ تصویر باقیمانده) تقسیم می‌کند. این یک تقسیم ١٠/٨٠ است.

y_train, y_val = y_train[:40000], y_train[:٤٠٠٠٠]: همین کار را برای برچسب های مربوطه انجام می دهد.

tf.keras.applications.xception.preprocess_input(x_train): داده های تصویر را با استفاده از مراحل پیش پردازش خاص مورد نیاز مدل Xception پیش پردازش می کند. این معمولاً شامل مقیاس بندی مقادیر پیکسل به یک محدوده خاص است (به عنوان مثال،

[-۱، ۱]). مهم است که از همان پیش پردازشی استفاده کنید که در طول آموزش مدل اصلی Xception در ImageNet استفاده شد.

keras.utils.to_categorical(y_train, num_classes=10): برچسب های اعداد صحیح (۰-۹) را به بردارهای کدگذاری شده one-hot تبدیل می کند. به عنوان مثال، برچسب ۳ تبدیل به [۰، ۰، ۰، ۱، ۰، ۰، ۰، ۰، ۰، ۰] می شود. این برای استفاده از خطا طبقه‌ای ضروری است.

```
base_model = Xception(weights='imagenet', include_top=False, input_shape=(75, 75, 3))

base_model.trainable = False

inputs = keras.Input(shape=(32, 32, 3))
x = layers.Resizing(75, 75)(inputs)
x = base_model(x, training=False)
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dropout(0.2)(x)
outputs = layers.Dense(10, activation='softmax')(x)
model = keras.Model(inputs, outputs)
```

base_model: مدل Xception از پیش آموزش دیده را بارگذاری می کند.

weights='imagenet': وزن‌هایی را که در طول آموزش آموخته‌اند روی مجموعه داده ImageNet بارگیری می‌کند.

include_top=False: لایه های طبقه بندی (کاملا متصل) مدل اصلی Xception را حذف می کند. این به این دلیل است که ما می خواهیم آنها را با لایه های خودمان برای طبقه بندی CIFAR-10 جایگزین کنیم.

input_shape=(۷۵, ۷۵, ۳): شکل ورودی را که مدل پایه باید انتظار داشته باشد را مشخص می کند. از آنجایی که Xception دارای حداقل اندازه ورودی ۷۱x۷۱ است، در اینجا از ۷۵x۷۵ استفاده می کنیم.

base_model.trainable = False: وزن تمام لایه ها را در مدل پایه Xception منجمد می کند. این برای یادگیری انتقال بسیار مهم است. ما نمی خواهیم وزنه های از پیش تمرین شده را در طول تمرین اولیه به روز کنیم.

❖ساخت یک مدل جدید:

inputs = keras.Input(shape=(32, 32, 3)): لایه ورودی را برای مدل جدید ما تعریف می کند که تصاویر CIFAR-10 (32x32x3) را می پذیرد.

x = layers.Resizing: یک لایه تغییر اندازه اضافه می کند تا تصاویر ورودی را از ۳۲x۳۲ به ۷۵x۷۵ ارتقاء دهد که نیاز به اندازه ورودی مدل پایه Xception را برآورده می کند.

x = base_model(x, training=False): تصاویر را از مدل پایه فریز شده Xception عبور می دهد. training=False در اینجا برای اطمینان از اینکه مدل پایه در حالت استنتاج قرار دارد مهم است (به عنوان مثال، لایه های نرمال سازی دسته ای در طول آموزش و استنتاج رفتار متفاوتی دارند).

x = layers.GlobalAveragePooling2D()(x): یک لایه ادغام میانگین سراسری اضافه می کند. این باعث کاهش ابعاد فضایی نقشه‌های ویژگی می‌شود (به عنوان مثال، اگر خروجی مدل پایه ٤x4x2048 بود، آن را به ۱x1x2048 کاهش می‌دهد).

x = layers.Dropout(0.2)(x): یک لایه حذفی با نرخ ۰.۲ اضافه می کند. این امر با حذف تصادفی ۲۰ درصد از واحدها در حین تمرین، به جلوگیری از اضافه‌کاری کمک می‌کند.

در خط بعدی یک لایه خروجی ن را اضافه می کند که یک لایه متراکم (کاملا متصل) با ۱۰ واحد (یکی برای هر کلاس **CIFAR-10**) و یک تابع فعال سازی **softmax** است. . **Softmax** یک توزیع احتمال در ۱۰ کلاس تولید می کند.

model = keras.Model(inputs, outputs): مدل **Keras** جدید را با اتصال ورودی (ورودی ها) به خروجی (خروجی ها) ایجاد می کند.

کامپایل کردن مدل:

در این قسمت فرآیند یادگیری را تنظیم میکنیم.

بهینه ساز **Adam** را اضافه میکنیم.

خطا: تابع خطا را با استفاده از روش برچسب های **one-hot** تنظیم میکنیم. کلاس بندی ما چند کلاسه است. که این تابع خطا برای آن مناسب است.

Model.summary:

معماری مدل را پرینت میکند. ویژگی های هر لایه را مانند تعداد پارامتر و ... پرینت میکند.

Layer (type)	Output Shape	Param #
input_layer_9 (InputLayer)	(None, 32, 32, 3)	0
resizing_1 (Resizing)	(None, 75, 75, 3)	0
xception (Functional)	(None, 3, 3, 2048)	20,861,480
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 2048)	0
dropout_1 (Dropout)	(None, 2048)	0
dense_1 (Dense)	(None, 10)	20,490

Total params: 20,881,970 (79.66 MB)

Trainable params: 20,490 (80.04 KB)

Non-trainable params: 20,861,480 (79.58 MB)

❖ بخش یادگیری:

```
epochs = 10
batch_size = 32

history = model.fit(x_train, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    validation_data=(x_val, y_val))
```

epochs = 10: تعداد دوره های آموزشی (چند بار تکرار در کل مجموعه داده آموزشی) را تنظیم می کند.

batch_size = 32: اندازه دسته را تنظیم می کند که تعداد نمونه های پردازش شده در هر به روز رسانی گرادیان است.

history = model.fit (...): مدل را آموزش می دهد.

x_train ،y_train: داده‌های آموزشی (تصاویر و برچسب‌های کدگذاری شده یک‌بار).

batch_size=batch_size: اندازه دسته.

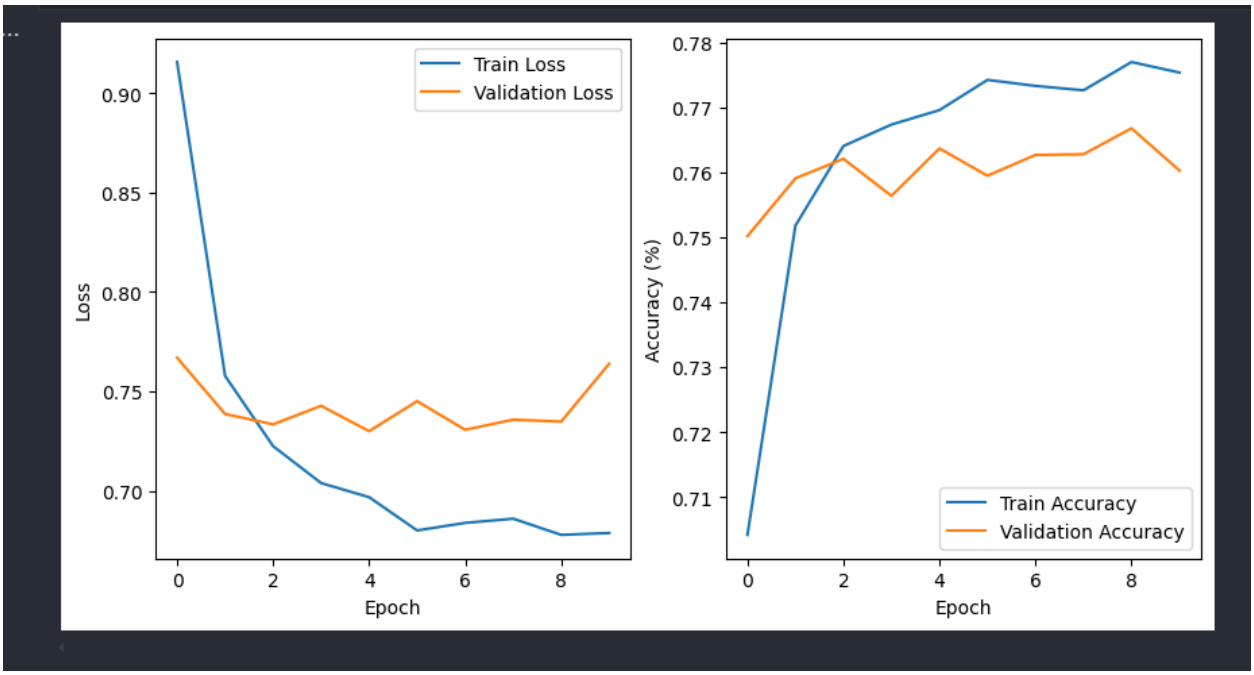
epochs=epochs: تعداد دورها.

validation_data=(x_val, y_val): داده‌های اعتبارسنجی (برای نظارت بر عملکرد داده‌های دیده نشده در طول آموزش استفاده می‌شود).

متد model.fit یک شی را برمی‌گرداند که حاوی اطلاعاتی درباره فرآیند آموزش، از جمله از خطا و دقت در هر دوره برای مجموعه‌های آموزشی و اعتبارسنجی است.

❖ بخش plot کردن:

در این بخش صرفا کتابخانه را وارد کردیم و استفاده کردیم.



❖ ارزیابی مدل:

مدل را بر روی داده های تست ارزیابی میکند. و مقدار دقت و خطا را برمیگردانیم.

در نهایت تعداد پارامتر را گزارش میکنیم.

یک لیست از وزن ها که همان تنسور ها هستند گزارش میکنیم در خط model.trainable_weights.

در نهایت همه پارامتر ها را جمع میکند.

مقادیر پارامتر های غیر آموزشی هم گزارش میکنیم.