پروژه چهارم – شبکه عصبی کانولوشنی و یادگیری انتقالی

در این پروژه ابتدا دسته بندی را با شبکه عصبی کانولوشنی انجام میدهیم. سپس با استفاده از یادگیری انتقالی و شبکه از پیش آموزش داده شده VGG16 مجددا دسته بندی را انجام میدهیم.

مرحله صفر: اضافه کردن کتابخانه ها

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_files
from keras.utils import np_utils
from sklearn.model_selection import train_test_split
from keras.preprocessing.image import array_to_img, img_to_array, load_img
from keras.models import Sequential
from keras.layers import Conv2D,MaxPooling2D
from keras.layers import Activation, Dense, Flatten, Dropout
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ModelCheckpoint
from keras import backend as K
from keras.optimizers import SGD, Adam, RMSprop
from keras.applications.vgg16 import VGG16
from keras.optimizers import SGD, Adam, RMSprop
```

مرحله یک: بارگذاری مجموعه داده ها و پیش پردازش

مجموعه داده لازم را از این لینک دانلود کنید. پس از بارگذاری داده های آموزش و آزمون، داده های آموزش را به دو بخش آموزش و اعتبار سنجی (Validation) تقسیم کنید. درصدی که برای داده های اعتبارسنجی انتخاب میکنید با توجه به تعداد کل داده ها انتخاب کنید.

برای پیش پردازش داده ها ابتدا تصاویر را به آرایه های کتابخانه numpy تبدیل کنید و مقادیر آن ها را در بازه صفر تا یک نرمال سازی کنید.

مرحله دوم: رویکرد اول- دسته بندی با استفاده از شبکه عصبی کانولوشنی

در رویکرد اول تمام مراحل مانند پروژه ۲ خواهد بود.

معماری شبکه عصبی کانولوشنی در این رویکرد به صورت زیر است:

- لایه کانولوشنی conv2D با ۱۶ فیلتر با اندازه ۲ و تابع فعالساز ’relu‘ و مقدار ’same‘ = padding
- لایه MaxPooling با اندازه ۲
- لایه کانولوشنی conv2D با ۳۲ فیلتر با اندازه ۲ و تابع فعالساز ’relu‘ و مقدار ’same‘ = padding
- لایه MaxPooling با اندازه ۲
- لایه کانولوشنی conv2D با ۶۴ فیلتر با اندازه ۲ و تابع فعالساز ’relu‘ و مقدار ’same‘ = padding
- لایه MaxPooling با اندازه ۲
- لایه کانولوشنی conv2D با ۱۲۸ فیلتر با اندازه ۲ و تابع فعالساز ’relu‘ و مقدار ’same‘ = padding
- لایه MaxPooling با اندازه ۲

- لایه Dropout با نرخ ۳۰٪
- لایه Flatten
- لایه Fully Connected با ۱۵۰ نورون و تابع فعالساز ’relu‘
- لایه Dropout با نرخ ۴۰٪
- لایه خروجی Fully Connected با تابع فعالساز ’softmax‘

مرحله سوم: ارزیابی شبکه و مشاهده نمودارهای دقت و خطا

در نهایت شبکه را با تابع خطای categorical_cross_entropy آموزش دهید و نمودارهای مربوط به دقت و خطای آموزش را مشاهده کرده و نتایج را ارزیابی کنید.

```
plt.figure(1, figsize = (10, 10))
plt.subplot(211)
plt.plot(CNN_model.history['acc'])
plt.plot(CNN_model.history['val_acc'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train', 'validation'], loc='upper left')

# plotting model loss
plt.subplot(212)
plt.plot(CNN_model.history['loss'])
plt.plot(CNN_model.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

مرحله چهارم: رویکرد دوم- دسته بندی با استفاده از یادگیری انتقالی

در این روش ، ما برای آماده سازی لایه های پایه ای از یادگیری انتقالی استفاده خواهیم کرد. VGG16 یک معماری شبکه عصبی است که برای طبقه بندی ۱۰۰۰ تصویر مختلف بر روی مجموعه داده imagenet آموزش داده شده است و ما برای رویکرد ۲ از وزن هایی که قبلاً روی VGG16 آموزش داده شده استفاده خواهیم کرد.

معماری شبکه و یادگیری انتقالی

برای شروع یادگیری انتقالی در ابتدا باید شبکه از پیش آموزش داده شده را در کد معرفی کنیم و لایه ورودی آن را حذف کنیم. در واقع پارامتر include_top را برابر با False قرار میدهیم و شبکه را از حالت قابل آموزش بودن خارج می کنیم. در کد مقادیر None را با مقادیر مناسب جایگزین کنید.

```
vgg_model = VGG16(input_shape = None, weights= None , include_top=None)

for layer in vgg_model.layers:
    layer.trainable = None
```

در معماری شبکه پس از اضافه کردن شبکه VGG لایه های زیر را اضافه کنید:

- لایه کانولوشنی conv2D با ۱۰۲۴ فیلتر با اندازه ۳ و تابع فعالساز ’relu‘ و مقدار padding = ’same‘
- لایه MaxPooling با اندازه ۲

- لایه Dropout با نرخ ۳۰٪
- لایه Fully Connected با ۱۵۰ نورون و تابع فعالساز ’relu‘
- لایه Dropout با نرخ ۴۰٪
- لایه خروجی Fully Connected با تابع فعالساز ’softmax‘

مرحله پنجم: ارزیابی شبکه و مشاهده نمودارهای دقت و خطا

در نهایت شبکه را با تابع خطای categorical_cross_entropy آموزش دهید و نمودارهای مربوط به دقت و خطای آموزش را مشاهده کرده و نتایج را ارزیابی کنید.

```
plt.figure(1, figsize = (10, 10))
plt.subplot(211)
plt.plot(transfer_learning_cnn.history['acc'])
plt.plot(transfer_learning_cnn.history['val_acc'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train', 'validation'], loc='upper left')

# plotting model loss
plt.subplot(212)
plt.plot(transfer_learning_cnn.history['loss'])
plt.plot(transfer_learning_cnn.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```