پروژه دوم – شناسایی تابلوهای راهنمایی و رانندگی در اتومبیل های خودران

با توجه به آنچه که در مباحث هفته دوم آمده است، کمی به کاربردهای دنیای واقعی نزدیک تر میشویم. پروژه هفته دوم در دو مرحله تعریف میشود.

مجموعه داده: از مجموعه داده این لینک استفاده میشود. اما به علت حجم بالای داده و محاسبات فقط از فایل (‘Train.p’) برای هر دو دسته آموزش و آزمایش استفاده میکنیم.

مرحله اول: استفاده مستقیم از داده ها برای آموزش شبکه عصبی

مرحله اول در فایل (‘Traffic_Sign_Detection_1’) تعریف شده است. در این مرحله هدف ما استفاده مستقیم از مجموعه داده تابلوهای راهنمایی و رانندگی برای آموزش شبکه عصبی است.

پس از اضافه کردن کتابخانه های مورد نیاز، مجموعه داده ها را در برنامه وارد می کنیم.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
import pickle
import tensorflow as tf
from PIL import Image
import os
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from keras.utils import to_categorical
from keras.models import Sequential, load_model
from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout
```

```
input_file = 'train.p'

with open(input_file, mode='rb') as f:
    data = pickle.load(f)

X_data, y_data = data['features'], data['labels']
```

با توجه به تعداد داده ها و کاربرد، داده ها را به دسته های X_train,y_train,X_test,y_test تقسیم بندی کنید. (راهنمایی: از تابع train_test_split استفاده کنید.)

مدل شبکه عصبی را با معماری زیر در keras ایجاد کنید.

- دو لایه از نوع Conv2D ، ۶۴ فیلتر با کرنلی به اندازه ۵*۵ و تابع فعالساز ‘relu’
- لایه MaxPooling با اندازه ۲*۲
- لایه Dropout با نرخ ۰/۲۵
- دو لایه از نوع Conv2D ، ۳۲ فیلتر با کرنلی به اندازه ۳*۳ و تابع فعالساز ‘relu’
- لایه MaxPooling با اندازه ۲*۲
- لایه Dropout با نرخ ۰/۲۵
- لایه Flatten برای یک بعدی کردن لایه ها

- لایه Fully Connected با ۲۵۶ نورون
- لایه Dropout با نرخ ۰/۵
- لایه Fully Connected با ۴۳ نورون (به تعداد دسته های مجموعه داده) و تابع فعالساز softmax

با توجه به کاربرد معیار metric مناسب را انتخاب کرده و با categorical_cross_entropy و بهینه ساز adam مدل را آموزش دهید.

با استفاده از کد زیر نمودارهای دقت و خطا را مشاهده کنید.

```
# Accuracy
plt.figure(0)
plt.plot(history.history['accuracy'], label='training accuracy')
plt.plot(history.history['val_accuracy'], label='val accuracy')
plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()
plt.show()
```

```
# Loss
plt.plot(history.history['loss'], label='training loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```

در نهایت مدل را ذخیره کنید.

```
model.save("traffic_classifier_2a.h5")
```

مرحله دوم: استفاده از داده های تقویت شده و نرمال سازی شده برای آموزش شبکه عصبی

مرحله اول در فایل (‘Traffic_Sign_Detection_2’) تعریف شده است. در این مرحله هدف ما استفاده از داده های تقویت شده و نرمال سازی شده از مجموعه داده تابلوهای راهنمایی و رانندگی برای آموزش شبکه عصبی است.

در این مرحله پس از اضافه کردن کتابخانه ها و مجموعه داده ها پیش پردازشی به منظور تقویت داده ها (Data_Augmentation) و نرمال سازی داده ها (Data_Normalization) انجام می شود.

تابع تقویت داده ها را با استفاده از rotate کردن تصاویر ایجاد کنید. در مورد سایر روش های تقویت داده مطالعه کنید.

تابع نرمال سازی داده ها را با Min-Max Scaling پیاده سازی کنید و داده ها را در بازه (۰/۵ و ۰/۵-) قرار دهید. در مورد سایر روش های نرمال سازی داده ها مطالعه کنید.

شبکه عصبی را مانند مرحله قبل ایجاد کرده و آموزش دهید و در نهایت نمودارهای دقت و خطا را مشاهده کنید.

نمودار و نتایج دو مرحله را مقایسه کنید. برای این کاربرد کدام مدل بهتر عمل میکند؟