پروژه دوم – کاربرد شبکه عصبی کانولوشنی

در پروژه قبل، پیاده سازی شبکه عصبی کانولوشنی را به صورت لایه به لایه با جزییات بررسی کردیم. در این پروژه با استفاده از کتابخانه Keras دسته بندی داده ها (لینک دانلود) را انجام می دهیم. داده ها تصاویری از حروف علامت با دست هستند.

مرحله صفر: اضافه کردن کتابخانه ها

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import keras
from keras.models import Sequential
from keras.callbacks import ReduceLROnPlateau
from keras.layers import Dense, Conv2D, Flatten, MaxPool2D, Dropout, BatchNormalization
from sklearn.metrics import classification_report
```

مرحله یک: بارگذاری مجموعه داده و پیش پردازش داده ها

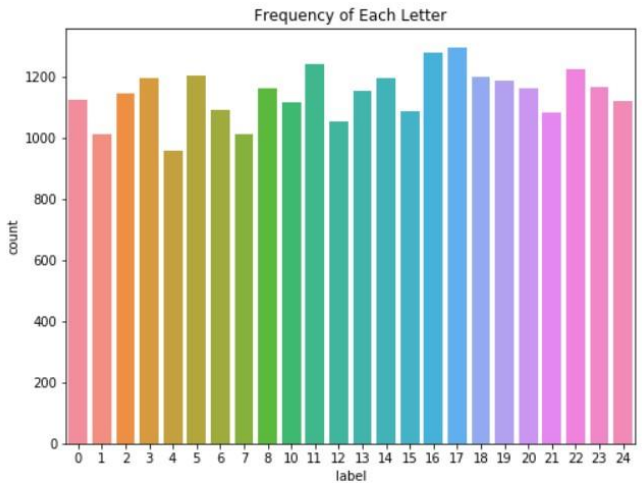
پس از دانلود مجموعه داده ها آن ها را در در کد بارگذاری کنید.

ویژگی ها و برچسب ها را تعیین و داده ها را در بازه صفر تا یک نرمال سازی کنید. همانطور که میبینید داده ها به صورت مقادیر پیکسل ها در فایلی با فرمت CSV هستند. با توجه به این که داده ها تصاویری علامت های نشان داده شده با دست هستند، آن ها را به صورت تصاویر ۲۸*۲۸ تبدیل میکنیم.

```
# Reshaping
x_train = x_train.values.reshape(-1, 28, 28, 1)
x_test = x_test.values.reshape(-1, 28, 28, 1)
```

با استفاده از رسم نمودار فرکانس داده ها از وجود تعادل در آنها اطمینان حاصل کنید.

```
plt.figure(figsize=(8,6))
sns.countplot(x=y_train, saturation=0.9)
plt.title("Frequency of Each Letter")
```



مرحله دو: ایجاد شبکه عصبی کانولوشنی با Keras

شبکه کانولوشنی را با استفاده از keras با معماری زیر ایجاد کنید:

- لایه کانولوشنی conv2D با ۴۹ فیلتر ۴*۴ و تابع فعالساز ’relu‘
- لایه BatchNormalization
- لایه MaxPooling با اندازه ۴*۴ و stride برابر با ۲
- لایه کانولوشنی conv2D با ۹۸ فیلتر ۴*۴ و تابع فعالساز ’relu‘
- لایه MaxPooling با اندازه ۴*۴ و stride برابر با ۱
- لایه کانولوشنی conv2D با ۱۹۶ فیلتر ۴*۴ و تابع فعالساز ’relu‘
- لایه Dropout با نرخ ۳۰٪
- لایه Flatten
- لایه Fully Connected با ۴۹ نورون و تابع فعالساز ’relu‘
- لایه Dropout با نرخ ۲۵٪
- لایه خروجی Fully Connected با تابع فعالساز ’softmax‘

مرحله سه: آموزش شبکه عصبی کانولوشنی

شبکه را با نرخ آموزش ۰/۰۰۰۱ ، بهینه ساز adam و تابع خطای sparse_cross_entropy آموزش دهید. اندازه batch را برابر با ۱۲۸ قرار دهید. مقادیر متفاوت نرخ آموزش، انواع بهینه ساز و اندازه batch را آزمایش کنید.

مقدار epoch ها را برابر با ۲۵ قرار دهید.

مرحله چهار: ارزیابی مدل و مشاهده نمودارهای دقت و خطا

گزارش confusion_matrix را مشاهده و بررسی کنید.

```
predictions = model.predict_classes(x_test, batch_size=128)
labels = ["Label " + str(i) for i in range(25) if i != 9]
print(classification_report(y_test, predictions, target_names = labels))
```

به ازای آزمایش های مراحل قبل نمودارهای دقت و خطا را مقایسه و بررسی کنید.

```
fig, axes = plt.subplots(2,1)
fig.set_size_inches(14,10)
axes[0].plot(history.history['loss'], color='b', label="Training loss")
axes[0].plot(history.history['val_loss'], color='r', label="Validation loss",axes =axes[0])
axes[0].set_xlabel("Epochs")
axes[0].set_ylabel("Loss")
legend = axes[0].legend()

axes[1].plot(history.history['accuracy'], color='b', label="Training accuracy")
axes[1].plot(history.history['val_accuracy'], color='r',label="Validation accuracy")
axes[1].set_xlabel("Epochs")
axes[1].set_ylabel("Accuracy")
legend = axes[1].legend()
```