یادگیری عمیق

تمرین اول

نیکی نزاکتی
98522094

۳.

الف)

- MNIST: دیتاستی شامل 60000 تصویر مربع کوچک 28×28 پیکسل در سیاه سفید از اعداد تک رقمی دست نویس بین 0 و 9 است.

CIFAR-10: دیتاستی شامل 50000 تصویر آموزشی رنگی 32x32 و 10000 تصویر آزمایشی است که در بیش از 10 کلاس برچسب گذاری شده اند.

FER-2013: دیتاستی شامل تصاویر 48x48 پیکسل در مقیاس خاکستری از چهره ها است.

- از to_categorical برای تبدیل داده های آموزشی قبل از اینکه به مدل خود منتقل کنیم، استفاده می کنیم. اگر داده های آموزشی از کلاس ها به عنوان اعداد استفاده می کند، مانند MNIST و CIFAR-10، to_categorical آن اعداد را در بردارهای مناسب برای استفاده با مدل ها تبدیل می کند تا بتوان مدل را با آن بردارها آموزش داد.

- MNIST: بعد اول در x_train و y_train نمایانگر تعداد داده های موجود که ۶۰۰۰۰ است. بعد دوم نشان دهنده تعداد فیچرها است که برای x_train پیکسل های ۲۸ در ۲۸ سیاه سفید است و y_train اعداد ۰ تا ۹.

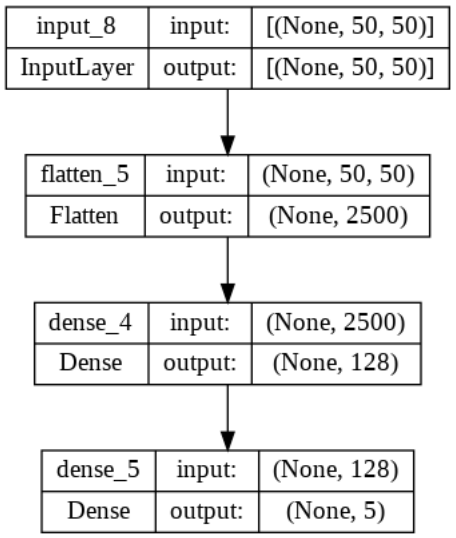
CIFAR: بعد اول در x_train و y_train نمایانگر تعداد داده های موجود که ۵۰۰۰۰ است. بعد دوم نشان دهنده تعداد فیچرها است که برای x_train پیکسل های ۳۲ در ۳۲ با ۳ کانال رنگی است و y_train شماره ی کلاس ها.

FER-2013: بعد اول در x_train و y_train نمایانگر تعداد داده های موجود که ۲۸۷۰۹ است. بعد دوم نشان دهنده تعداد فیچرها است که برای x_train پیکسل های ۴۸ در ۴۸ سیاه سفید است و y_train شماره ی کلاس ها.

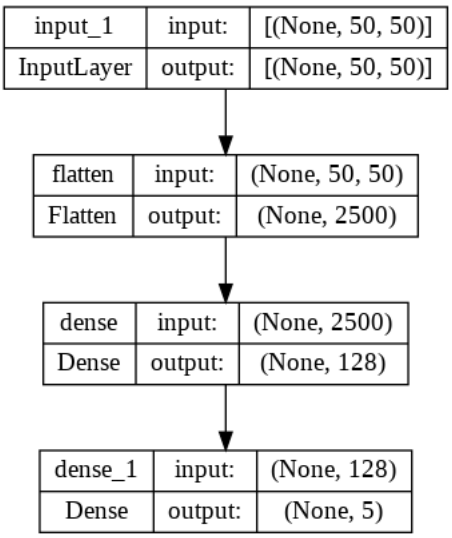
- با استفاده از imageDataGenerator می توان اول تمام عکس ها را load کرد و موقع fit کردن به جای اینکه تمام عکس ها را با هم به مدل بدهیم میتوانیم عکس ها را به صورت batch وارد کنیم و اینگونه بار کمتری به مدل وارد میشود. همچنین از imageDataGenerator برای image augmentation استفاده میشود یعنی به ازای هر عکسی که در دیتاست وجود دارد یک کپی از آن عکس ایجاد میکند و تغییراتی مانند flip, rotation, shift را روی آن ها می دهد که این باعث میشود مدل robust تر بشود چون داده های unseen را هم میتواند پوشش دهد و وقتی مدل هنگام predict کردن یک عکسی ببیند که مقداری

با عکس اصلی آموزش فرق داشته‌باشد راحت تر می‌تواند تشخیص دهد که label آن چیست و در واقع سایز دیتاست ما را بیشتر می‌کند.

ب)



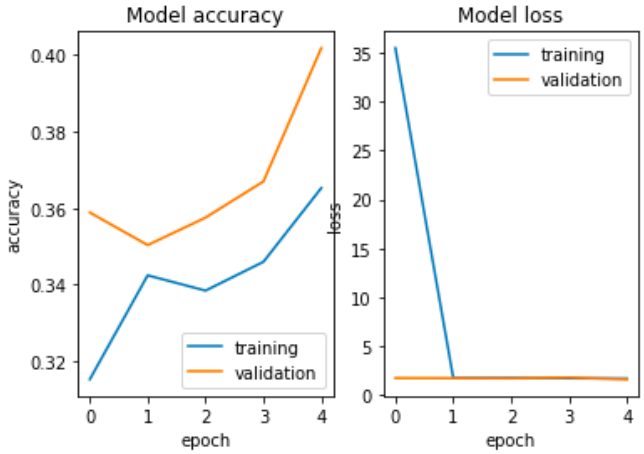
Functional API



Sequential

در summary ساختار لایه های مدل و اطلاعات مربوط به هر لایه شامل نوع لایه، ابعاد ورودی و خروجی، و تعداد نود های آن لایه است.

آموزش و ارزیابی مدل اول: همانطور که مشاهده می شود میزان دقت برای داده train و val در ابتدا نوساناتی داشته ولی در نهایت روند افزایشی داشته اند.



Evaluating Test set

```
# Write your code here
model_mnist.evaluate(
    x=x_test_1,
    y=y_test_1,
    batch_size=64,
)
```

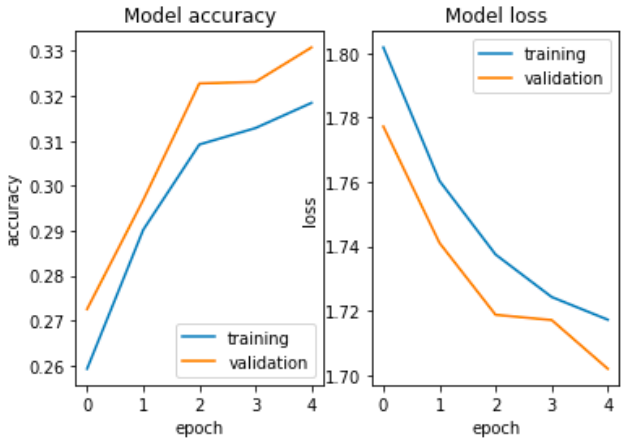
```
157/157 [==============================] - 0s 2ms/step - loss: 1.6051 - accuracy: 0.3966
[1.6051076650619507, 0.39660000801086426]
```

Predicting Some samples from Test set

```
[56]
# Write your code here
sample_mnist = [0, 10, 20]
for idx in sample_mnist:
    y_predicted = model_mnist.predict(np.array([x_test_1[idx]]))
    print(f"y_predicted : {np.argmax(y_predicted)} , y_true : {np.argmax(y_test_1[idx])}")
```

```
1/1 [==============================] - 0s 85ms/step
y_predicted : 7 , y_true : 7
1/1 [==============================] - 0s 45ms/step
y_predicted : 9 , y_true : 0
1/1 [==============================] - 0s 59ms/step
y_predicted : 9 , y_true : 9
```

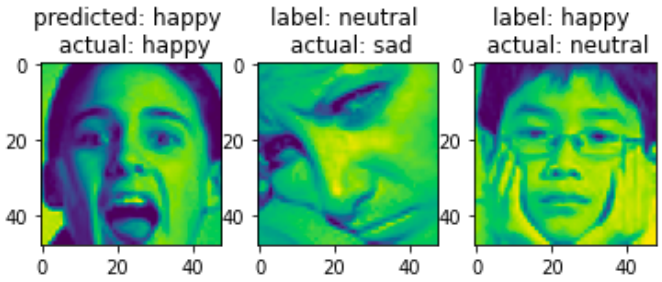
آموزش و ارزیابی مدل دوم: میزان دقت برای train همیشه در حال افزایش و برای val به غیر از یک مرحله نیز همیشه در حال افزایش بوده است.



Evaluating Test set

```
[64]
model_fer.evaluate(
    x=test_set,
)
```

```
113/113 [==============================] - 3s 27ms/step - loss: 1.6786 - accuracy: 0.3504
[1.6786015033721924, 0.350376158952713]
```



خیر مشابهت ندارند چون مدلی که استفاده کردیم مدل مناسبی نیست: تعداد لایه ها و نورون های کم است و همچنین سپس برای تصاویر بهتر است که از لایه های convolutional neural net استفاده شود.