به نام خدا

حسن اردشیر ۶۱۰۳۹۸۲۰۲ پروژه دادکاوی

در ابتدا دادگان را در حالت categorical با ترتیب تصادفی دریافت کردیم. این بدان معناست که به کلاس های بیان شده وکتور های زیر را اختصاص دادیم :

AK	**[1,0,0,0,0]**
Ala_Idris	[0,1,0,0,0]
Buzgulu	[0,0,1,0,0]
Dimnit	[0,0,0,1,0]
Nazli	[0,0,0,0,1]

همانطور که میدانیم لایبری Tensorflow جهت دریافت سریع تر اطلاعات از batch استفاده می کند و batch_size را برابر ۳۲ قرار دادیم. سپس 0.7 دادگاه را در train ، 0.2 دیگر را در validate و 0.1 دیگر را در test قرار دادیم.(در مدل اولیه از augment استفاده نکردیم)

همانطور که می دانیم عکس ها به صورت پیکسلی هستند و هر پیکسل را یک تاپل سه تایی از ۰ تا ۲۵۵ تشکیل می دهد. پس دادگان را نرمالایز می کنیم (تقسیم بر ۲۵۵ می کنیم) تا اعداد بزرگ در مدل ما باعث خرابی مدل نشوند.

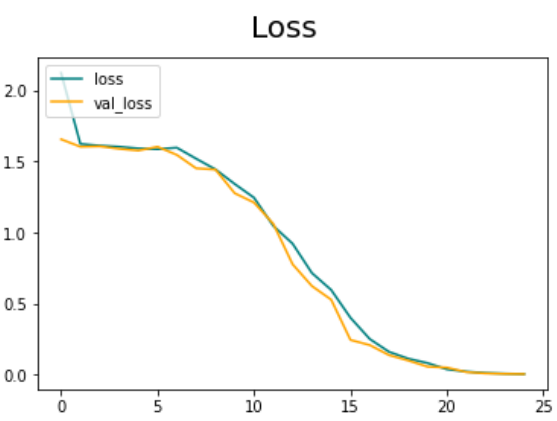
سپس مدلی از شبکه متوالی cnn ارائه دادیم :

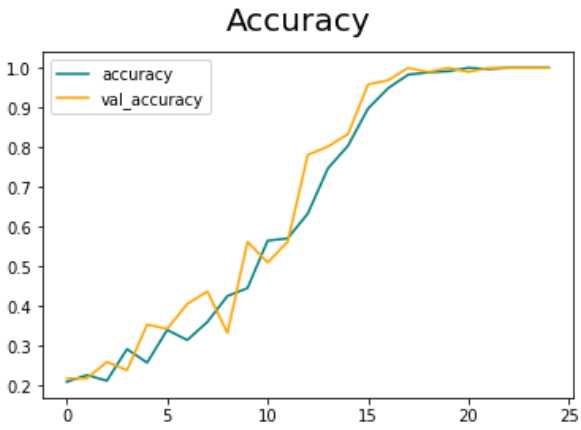
```
model = Sequential()
model.add(Conv2D(32, (3,3), activation='relu', input_shape=(256, 256, 3)))
model.add(MaxPooling2D())

model.add(Conv2D(16, (3,3), activation='relu'))
model.add(MaxPooling2D())

model.add(Flatten())

model.add(Dense(32, activation='relu'))
model.add(Dense(16, activation='relu'))

model.add(Dense(5, activation='softmax'))
```

```
model.compile('adam', loss=tf.losses.CategoricalCrossentropy(), metrics=['accuracy'])
```

در واقع در آن از دو لایه convertional2d و همچنین maxpooling2d استفاده شده است. همچنین چون دادگان به صورت categorical دریافت شده اند درنتیجه تابع خطا در حالت Categorical Crossentropy()تنظیم شده است. همانطور که مشخص است دو لایه dense به منظورdenoising نیز قرار داده ایم.

سپس این مدل را با دادگان داده شده ، آموزش دادیم و نتایج زیر را به دست آوردیم :






```
Precision: 1.0
Recall: 1.0
Acc: 1.0
```

در نتیجه همان طور که مشخص است تنها با کمتر از ۲۰ مرحله به accuracy نزدیک به یک رسیدیم و در مرحبه ۲۳ دقت ۱ را می بینیم !!! پس این بدان معنی نمی باشد که این مدل لزوماً مدل خوبی می باشد.

که این موضوع را می توان از ماتریس Confusion matrix آن نیز بیان کرد.

	True Positive	True Negative
Predicted Positive	1	0
Predicted Negative	0	1

در واقع در اینجا ما به حالت overfitting رسیده ایم یا دادگان تست ما نسبت به دادگان آموزش بسیار کم تر بوده اند در نتیجه در دادگان تست جدید شاید نتیجه مطلوب را ندهند. (اطلاعات داده شده بهترین نتیجه این مدل بعد از ۱۰ بار آزمایش بود) (میانگین این مدل دقت ۰.۸ را داشت)

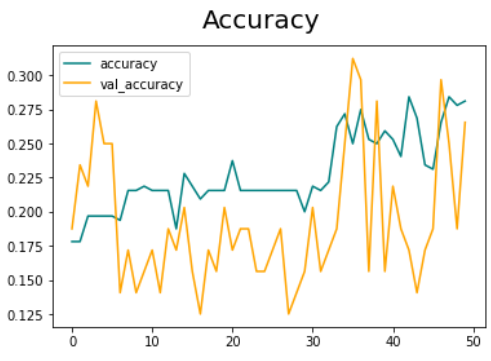
بنابراین برای رفع این ایراد از augmentation استفاده می کنیم تا دیگر overfitting را نبینیم.
بدین منظور ابتدا ۲۰ درصد دادگان را جدا کرده و در test قرار می دهیم و ۸۰ درصد دیگر دادگان را به وسیله تابع ImageDataGenerator و مقادیر زیر گسترش دادیم:

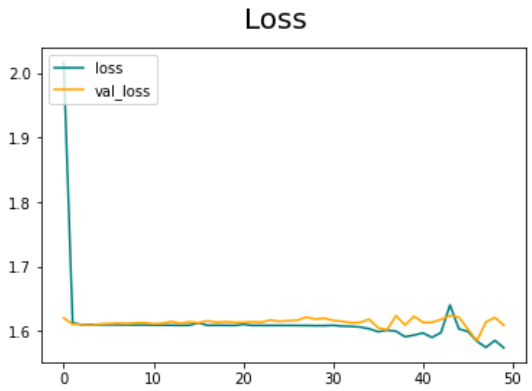
```
trainval_gen = ImageDataGenerator(
    rotation_range = 40,
    zoom_range = 0.15,
    rescale=1 / 255.0,
    brightness_range=[0.8,1.6],
    channel_shift_range=0.15,
    height_shift_range=0.15,
    shear_range=0.1,
    horizontal_flip=True,
    vertical_flip=True,
    validation_split=0.2,
    fill_mode='nearest')
```

```
train_generator = trainval_gen.flow_from_dataframe(
    dataframe=pd_trainval,
    class_mode="categorical",
    subset='training',
    target_size=(256,256),
    shuffle=True
)
```

(تعداد عکس های گسترش داده شده در مدل ارائه شده بر اساس تعداد مراحل مورد نیاز متغیر می باشد و این تابع به طور خودکار دادگان جدید را تولید می کند)

سپس همان مدل قبلی cnn متوالی به این دادگان جدید داده شده اما نتیجه به طور فاحشی ضعیف بوده و تنها دقت ۰.۲ را حتی با بیش از ۵۰ مرحله ، دریافت کرد . (به طور میانگین)

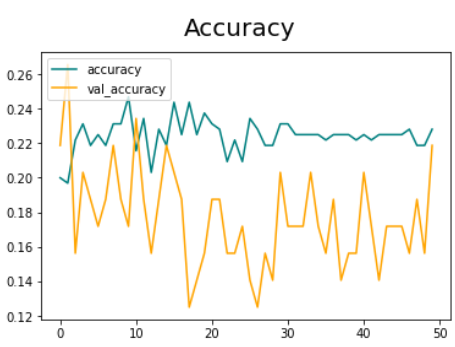


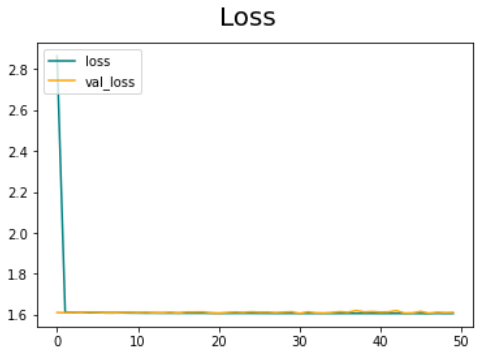


همانطور که مشخص است مشکل این مدل این است که بسیار کم آموزش داده شده است و درنتیجه مقادیر خوبی را دریافت نمی کند. برای ترمیم این مدل ، در ابتدا مقادیر مربوط به augmentation را کم تر کرده ولی در این صورت باز به حالت overfitting رسیدیم.

درنتیجه مشکل از مدل طراحی شده برای آن بود.

چرا که حتی با تغییرات تعداد گره های Con2d ها به نتیجه مطلوبی نرسیدیم :



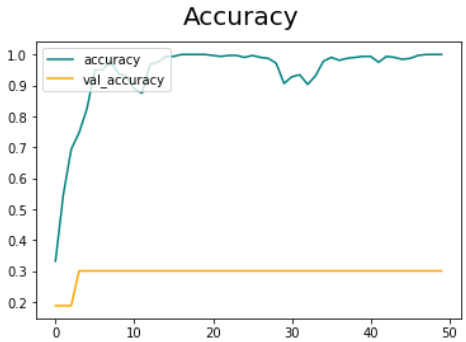


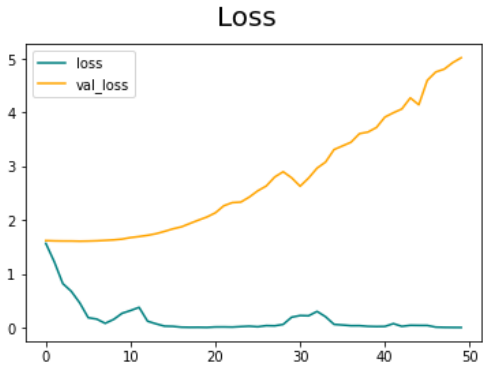
سپس از مدل mobilenetv2 در دادگان augmentation شده استفاده کردیم.

```
model = MobileNetV2(input_shape=(256, 256, 3),
            include_top=True,
            weights = None,
            classes=5)
model.compile('adam', loss=tf.losses.CategoricalCrossentropy(), metrics=['accuracy'])
```

که در ازمایشات اولیه طی ۵۰ مرحله توانست به دقت 0.68 برسد اما هنوز هم دقت validate آن پایین بود و اصلا در دادگان تست نتیجه مطلوبی نداشت.

حال در ImageDataGenerator تنها نرمالایز کردن را نگه داشتیم در نتیجه دادگان validate و train از نظر ویژگی هایشان تغییر پیدا نمی کنند و سپس مدل mobilenetv2 را در آن پیاده کردیم. در این حالت به دقت ۱ در دادگان آموزش بعد از ۵۰ مرحله رسیدیم اما هنوز دقت دادگان validate و test خوب نبوده و داریم:





پس در اینجا نسبت به مدل اولیه بیان شده بیشتر overfitting بر روی دادگان آموزش بود . پس باید augmentation را بهتر کنیم. بدین منظور به جای استفاده از ImageDataGenerator از لایه های مربوط به augmentation در مدل متوالی خود (همان مدل اول که ارائه شد) استفاده می کنیم .