به نام خدا

دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

شبکه‌های عصبی

تکلیف چهارم

استاد درس: دکتر صفابخش

امیرحسین کاشانی

۴۰۰۱۳۱۰۷۱

amkkashani@gmail.com

نیم سال اول ۱۴۰۱-۱۴۰۲

Contents

i)

بر اساس مشاهدات انجام شده residual network ها رفتاری مشابه روش‌های Ensemble بر روی شبکه‌های کم عمق (shallow) دارند. حال سوال ما این است که یک شبکه residual با n بلاک نسبت به یک شبکه با c n بلاک چگونه رفتار می‌کند.[1]

با توجه به اینکه وزن‌ها در فرآیند Backpropagation در یک دیگر ضرب می‌شوند انتظار می‌رود که تاثیر این تغییر به شکل نمایی در عملکرد مشاهد گردد انتخاب تعداد بلاک‌ها یا به بیان دیگر عمق بلاک‌ها یک هایپر پارامتر است که براساس پیچیدگی وظیفه، داده‌ها و تعداد فیچرها باید مشخص گردد. هر چه شرایط مساله پیچیده تر باشد بهتر است، طول دسته‌های بلند تر باشد این کار معادل این است که در یک روش Ensemble بدلیل سختی مساله از مدل‌های پیچیده تر استفاده کنیم و زمانی که مساله پیچیدگی کمتری برخوردار است بهتر است طول بلاک‌ها کمتر باشد.

[1] M. Abdi and S. Nahavandi, ‘Multi-residual networks: Improving the speed and accuracy of residual networks’, *arXiv preprint arXiv:1609. 05672*, 2016.

ii)

بر اساس پاراگرافی که در مقاله معرفیِ Densenet آمده است (به خصوص دو خط آخر) استفاده از average pooling به این دلیل توصیه شده که max-pooling در تشخیص لبه با توجه به اینکه ماکسیمم را بر می‌گرداند می‌تواند دچار خطا شود[2] زیرا فقط بزرگ ترین خانه را بر می‌گرداند و این باعث از بین رفتن تاثیر خانه‌هایی که مقدار کمتری دارند شود. اما از طرف دیگر average pooling تاثیر همه خانه‌ها را نگه می‌دارد و در صورتی که خانه‌های ضعیف تر بیان کننده خط یا لبه تصویر باشند هنوز شانس نشان دادن خود را خواهند داشت.

Using this approach, however, in turn raises another issue. Given the kernel/window sizes of the convolutional and max-pooling commonly found in CNNs, each descriptor from a deep convolutional layer can have a large (perhaps ~200px) receptive field size (or supporting image patch size) in the input image. Thus, stitching could lead to edge/corner artifacts and receptive field pollution between neighboring pyramid scales that are adjacent in the large stitched images. To mitigate this, we add a 16px border to each image, for a total of at least 32px of of padding between any pair of images on a plane. We fill the background with the mean pixel value used for data centering (as discussed below). Finally, we linearly interpolate all image padding between the image's edge pixel and the centering mean pixel value. Experimentally, we find that this scheme seems successful in avoiding obvious edge/corner artifacts and receptive field pollution.

[2] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer, ‘Densenet: Implementing efficient convnet descriptor pyramids’, arXiv preprint arXiv:1404. 1869, 2014.

iii)

همیشه با استفاده از فیلترهای کوچکتر شروع می‌کنیم به این معنی که تا حد امکان اطلاعات محلی را جمع آوری کرده و به تدریج عرض فیلتر را افزایش می‌دهیم تا فضای ویژگی ایجاد شده کاهش یابد و ویژگی‌های کم ولی سطح بالا و کلی از شبکه نتیجه گیری شود.

طبق این اصل، تعداد کانال‌ها باید در ابتدا کم باشد به طوری که ویژگی‌های سطح پایین را تشخیص دهد که با هم ترکیب می‌شوند تا اشکال پیچیده زیادی را تشکیل دهند (با افزایش تعداد کانال‌ها) که به تمایز بین کلاس‌ها کمک می‌کند.

تعداد فیلترها برای افزایش عمق فضای ویژگی افزایش می‌یابد و در نتیجه به یادگیری سطوح بیشتری از ساختارهای انتزاعی جهانی کمک می‌کند. یکی دیگر از کاربردهای عمیق‌تر و باریک‌تر کردن فضای ویژگی، کوچک کردن فضای ویژگی برای ورودی به شبکه‌های متراکم است.

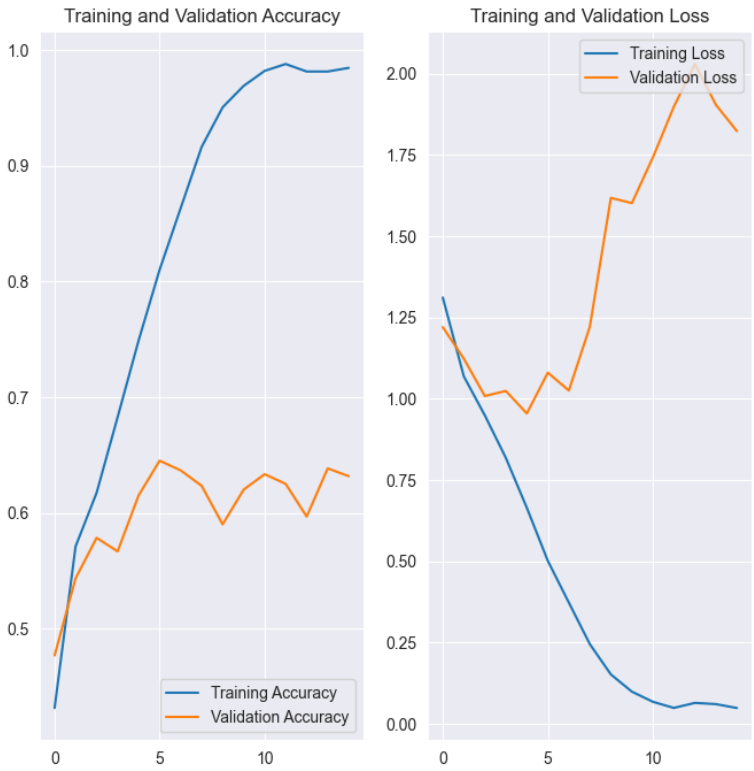
Q1)

از دو شبکه عصبی استفاده کرده ایم.

مدل اول عمق کمتری داشت

```
num_classes = len(class_names)

model = Sequential([
    layers.Rescaling(1. / 255, input_shape=(128, 128, 3)),
    layers.Conv2D(16, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(32, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(64, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(num_classes)
])
```




```
accuracy_score(test_y, pred)
```

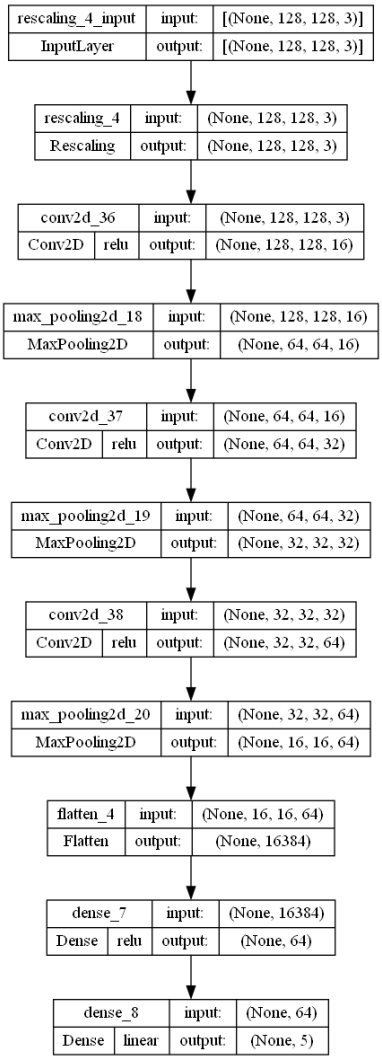
```
0.621
```

```
confusion_matrix(test_y, pred)
```

```
array([[238,  45,  28,  65,  24],
       [ 20, 212,  70,  30,  68],
       [  4,  93, 265,   7,  31],
       [ 37,  30,   8, 302,  23],
       [ 28,  80,  43,  24, 225]], dtype=int64)
```

```
1  print("Total time: ", end, "seconds")
```

```
Total time:  1086.2040882110596 seconds
```

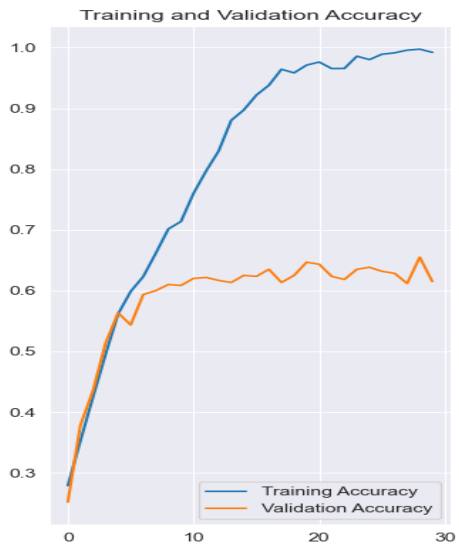


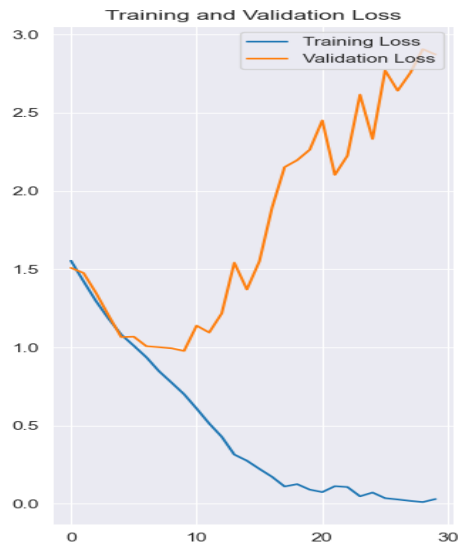
مدل دوم یک لایه عمیق تر می‌باشد

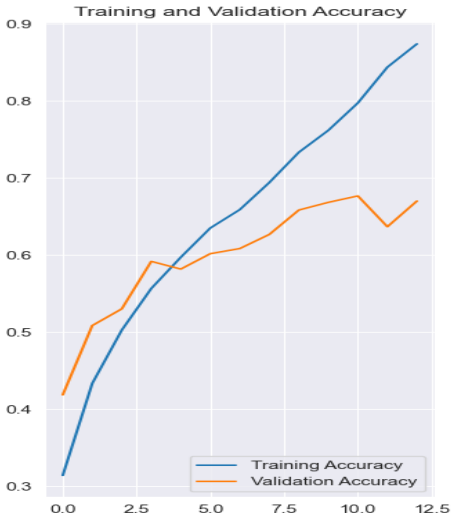
```python
num_classes = len(class_names)

model_q1_2 = Sequential([
    layers.Rescaling(1. / 255, input_shape=(128, 128, 3)),
    layers.Conv2D(16, 3, padding='same', activation='relu'),
    layers.AvgPool2D(),
    layers.Conv2D(32, 5, padding='same', activation='relu'),
    layers.AvgPool2D(),
    layers.Conv2D(64, 7, padding='same', activation='relu'),
    layers.AvgPool2D(),
    layers.Conv2D(128, 7, padding='same', activation='relu'),
    layers.AvgPool2D(),
    layers.Flatten(),
    layers.Dense(256, activation='relu'),
    layers.Dense(num_classes)
])
```

یک با train را با ۳۰ epoch انجام می‌دهیم همگرایی در حدود epoch = 15 رخ داده است از این رو یک بار با ۳۰ و یکبار با ۱۵ اجرا کردم.



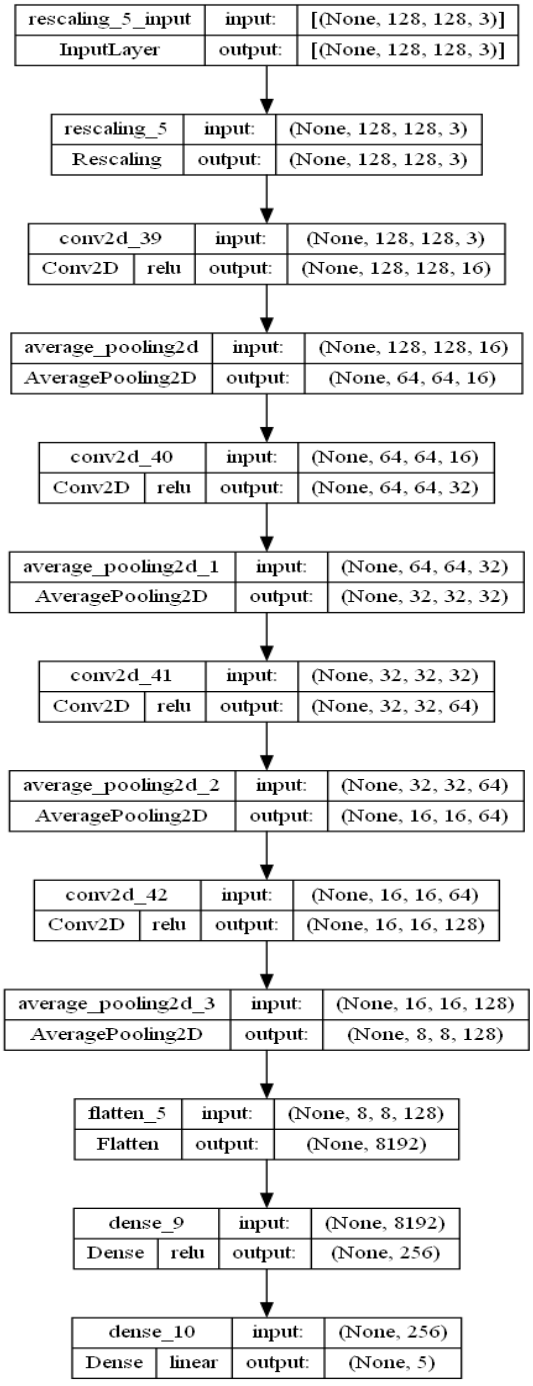




مدت زمان بدست آمده برای ۱۳ epoch

```
Total time:  674.8484659194946 seconds
```

میانگین هر epoch حدود ۵۰ ثانیه


rescaling_5_input | input: | [(None, 128, 128, 3)]
InputLayer | output: | [(None, 128, 128, 3)]
rescaling_5 | input: | (None, 128, 128, 3)
Rescaling | output: | (None, 128, 128, 3)
conv2d_39 | input: | (None, 128, 128, 3)
Conv2D | relu | output: | (None, 128, 128, 16)
average_pooling2d | input: | (None, 128, 128, 16)
AveragePooling2D | output: | (None, 64, 64, 16)
conv2d_40 | input: | (None, 64, 64, 16)
Conv2D | relu | output: | (None, 64, 64, 32)
average_pooling2d_1 | input: | (None, 64, 64, 32)
AveragePooling2D | output: | (None, 32, 32, 32)
conv2d_41 | input: | (None, 32, 32, 32)
Conv2D | relu | output: | (None, 32, 32, 64)
average_pooling2d_2 | input: | (None, 32, 32, 64)
AveragePooling2D | output: | (None, 16, 16, 64)
conv2d_42 | input: | (None, 16, 16, 64)
Conv2D | relu | output: | (None, 16, 16, 128)
average_pooling2d_3 | input: | (None, 16, 16, 128)
AveragePooling2D | output: | (None, 8, 8, 128)
flatten_5 | input: | (None, 8, 8, 128)
Flatten | output: | (None, 8192)
dense_9 | input: | (None, 8192)
Dense | relu | output: | (None, 256)
dense_10 | input: | (None, 256)
Dense | linear | output: | (None, 5)

```
accuracy_score(test_y, pred)
```

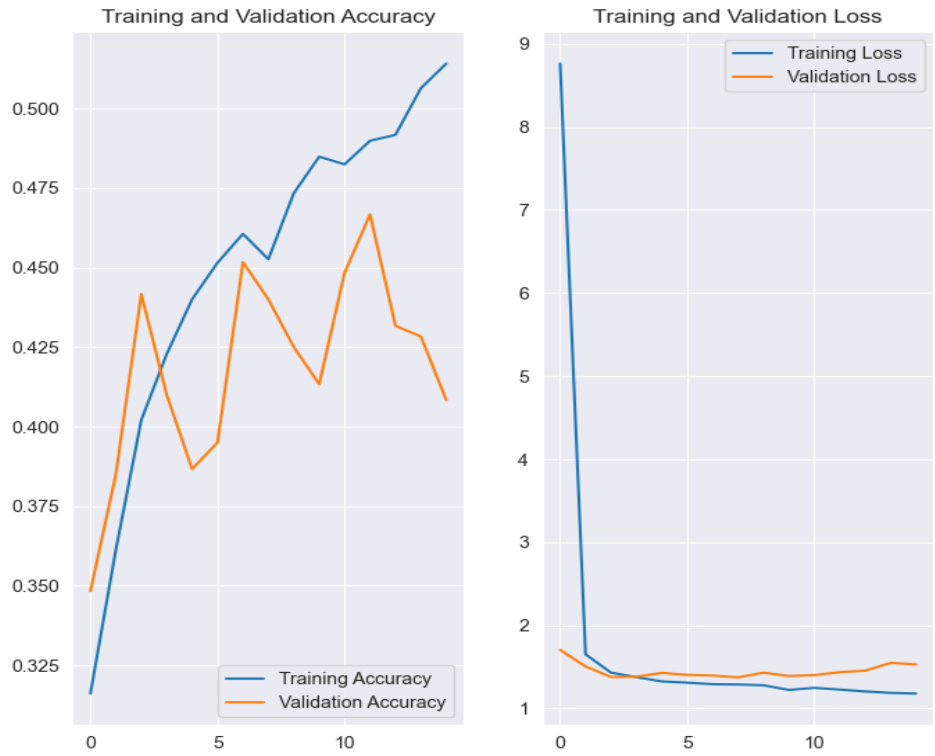
```
0.679
```

```
confusion_matrix(test_y, pred)
```

	0	1	2	3	4
0	285	30	19	41	25
1	25	220	66	15	74
2	2	52	325	0	21
3	54	21	13	282	30
4	25	73	32	24	246

Q2)

در این سوال با افزودن کانکشن یک مدل residual ایجاد می‌کنیم. از آنجا که حجم داده‌های ما کم است و طول بلاک‌های ما در اینجا کوچک است عملکرد ضعیف تری را شاهد خواهیم بود اما بسیار سریع تر بود به حدی که هر epoch در مدل مشابه در بخش Q1 (اولین مدل بیان شده) در حدود ۷۰ ثانیه به طول می‌انجامید اما در این مدل هر epoch در حدود ۴۰ ثانیه اما درصد صحت بدست آمده و همگرایی بسیار سریع تر اتفاق می‌افتد.



```
print(f'duration is :{duration}')
```

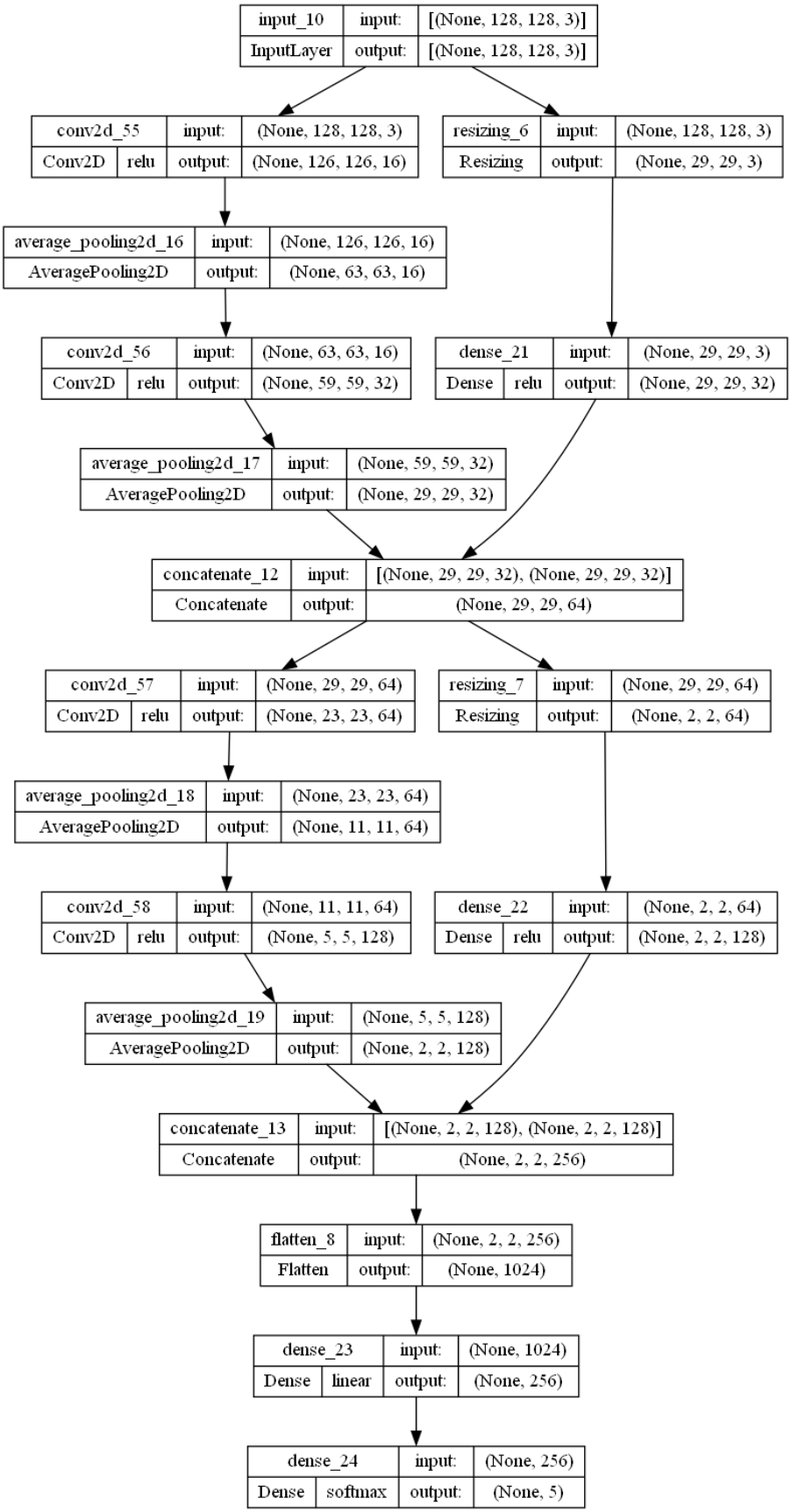
```
duration is :494.6948444843292
```

```
accuracy_score(test_y, pred)
```

```
0.4255
```

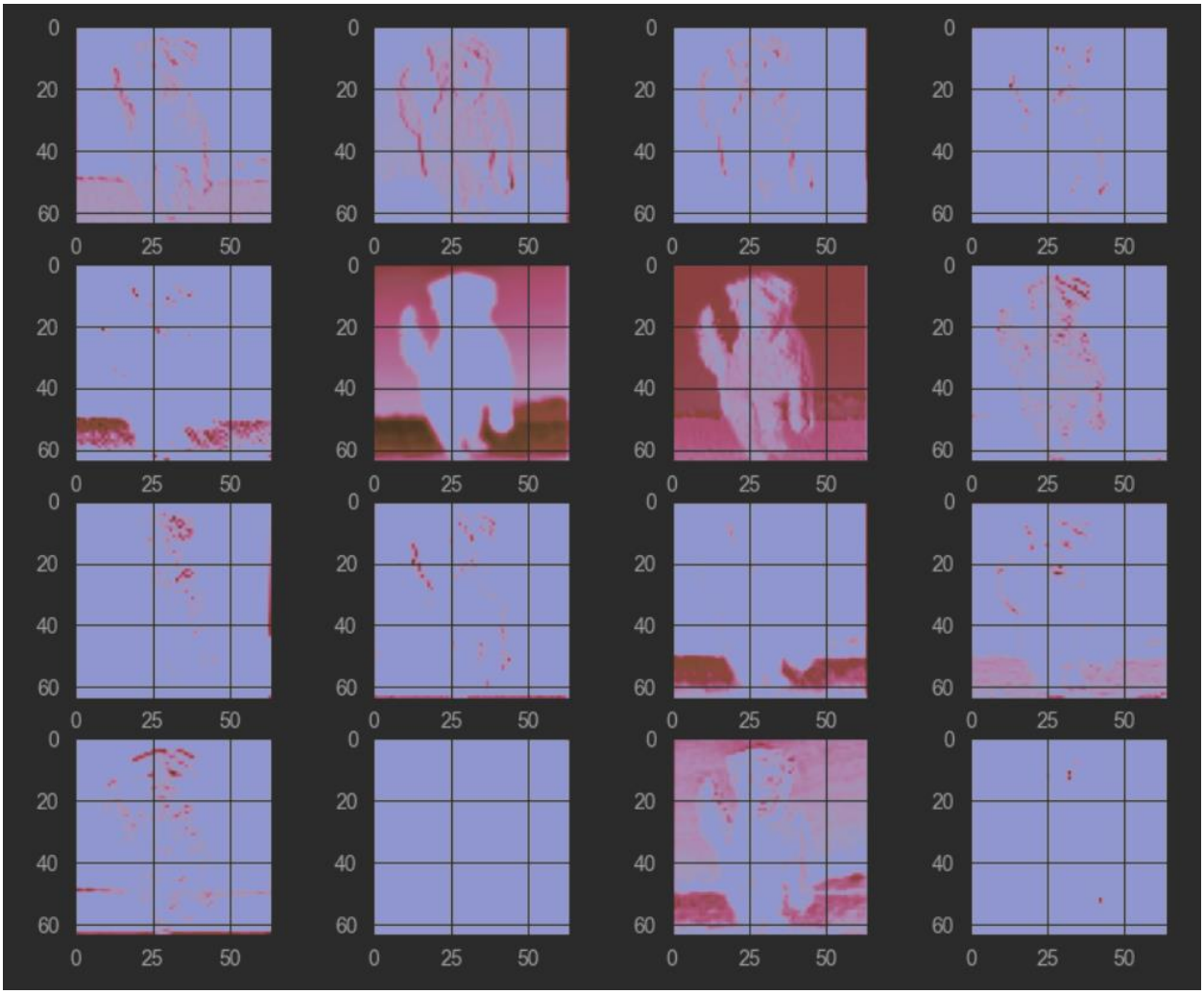
```
confusion_matrix(test_y, pred)
```

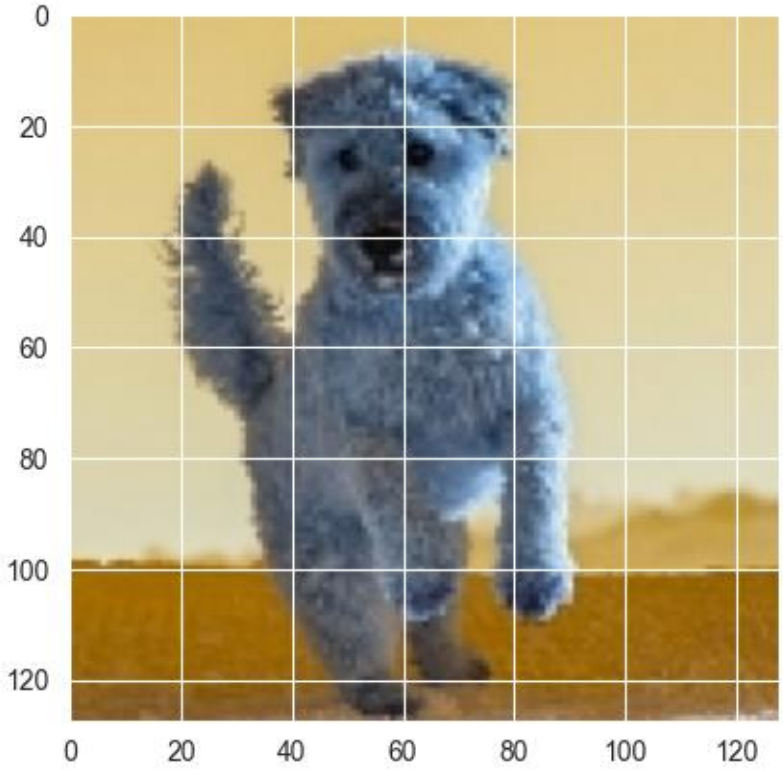
```
array([[168,   9,  43, 161,  19],
       [ 51,  37, 170, 109,  33],
       [ 21,  11, 269,  81,  18],
       [ 53,   8,  44, 282,  13],
       [ 45,  17, 158,  85,  95]], dtype=int64)
```


input_10 | input: | [(None, 128, 128, 3)]
InputLayer | output: | [(None, 128, 128, 3)]
conv2d_55 | input: | (None, 128, 128, 3)
Conv2D | relu | output: | (None, 126, 126, 16)
resizing_6 | input: | (None, 128, 128, 3)
Resizing | output: | (None, 29, 29, 3)
average_pooling2d_16 | input: | (None, 126, 126, 16)
AveragePooling2D | output: | (None, 63, 63, 16)
conv2d_56 | input: | (None, 63, 63, 16)
Conv2D | relu | output: | (None, 59, 59, 32)
dense_21 | input: | (None, 29, 29, 3)
Dense | relu | output: | (None, 29, 29, 32)
average_pooling2d_17 | input: | (None, 59, 59, 32)
AveragePooling2D | output: | (None, 29, 29, 32)
concatenate_12 | input: | [(None, 29, 29, 32), (None, 29, 29, 32)]
Concatenate | output: | (None, 29, 29, 64)
conv2d_57 | input: | (None, 29, 29, 64)
Conv2D | relu | output: | (None, 23, 23, 64)
resizing_7 | input: | (None, 29, 29, 64)
Resizing | output: | (None, 2, 2, 64)
average_pooling2d_18 | input: | (None, 23, 23, 64)
AveragePooling2D | output: | (None, 11, 11, 64)
conv2d_58 | input: | (None, 11, 11, 64)
Conv2D | relu | output: | (None, 5, 5, 128)
dense_22 | input: | (None, 2, 2, 64)
Dense | relu | output: | (None, 2, 2, 128)
average_pooling2d_19 | input: | (None, 5, 5, 128)
AveragePooling2D | output: | (None, 2, 2, 128)
concatenate_13 | input: | [(None, 2, 2, 128), (None, 2, 2, 128)]
Concatenate | output: | (None, 2, 2, 256)
flatten_8 | input: | (None, 2, 2, 256)
Flatten | output: | (None, 1024)
dense_23 | input: | (None, 1024)
Dense | linear | output: | (None, 256)
dense_24 | input: | (None, 256)
Dense | softmax | output: | (None, 5)

Q3)

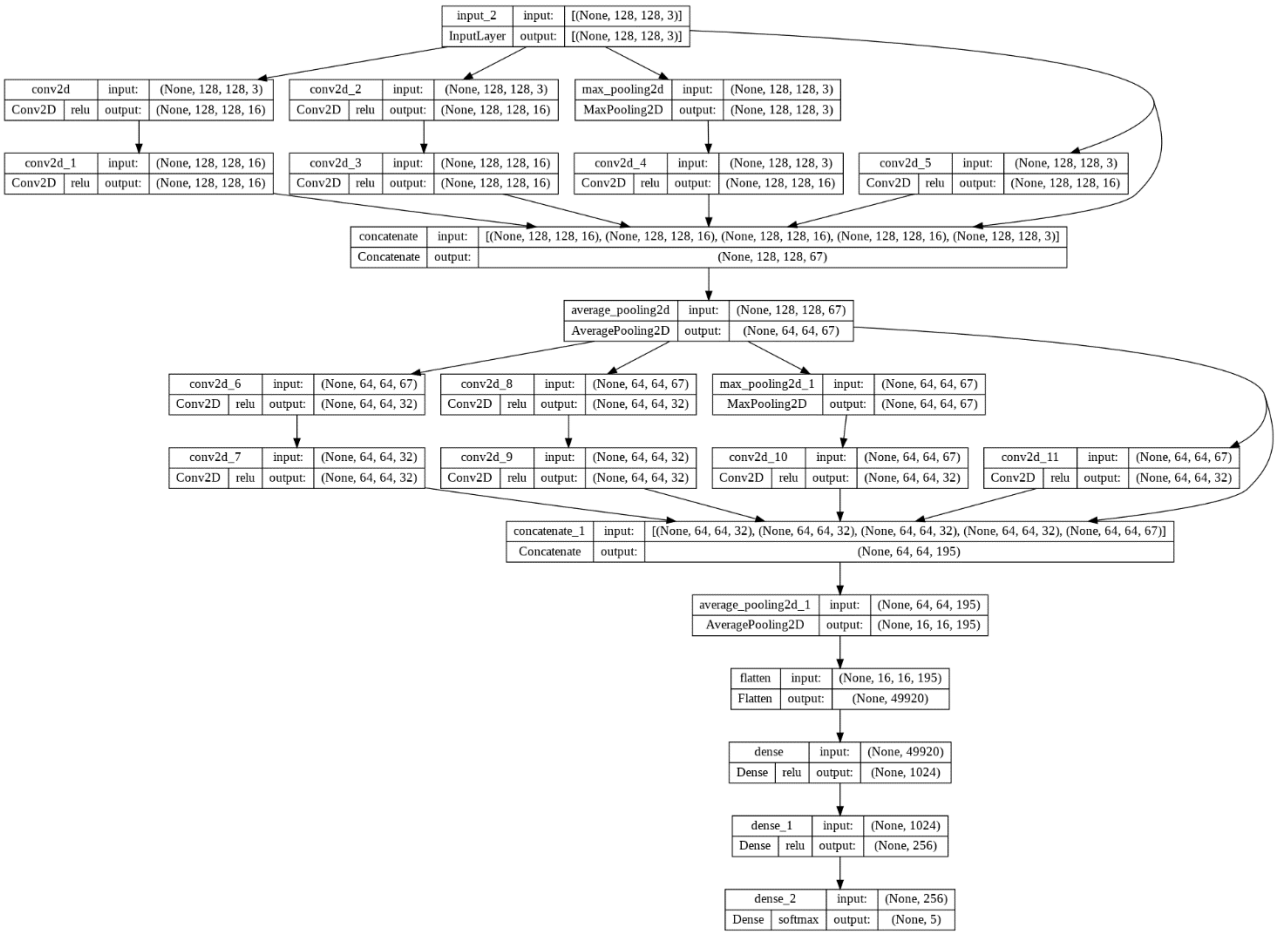
عکس بدست آمده نشان می‌دهد که مدل در برخی از چنل‌ها به خوبی توانسته سوژه را از محیط جدا کند و حتی زمین را نیز از بک گراند در عکس مربوطه تشخیص داده است همچنین در برخی از چنل‌ها خطوط لبه تصویر را می‌توانیم مشاهده کنیم.

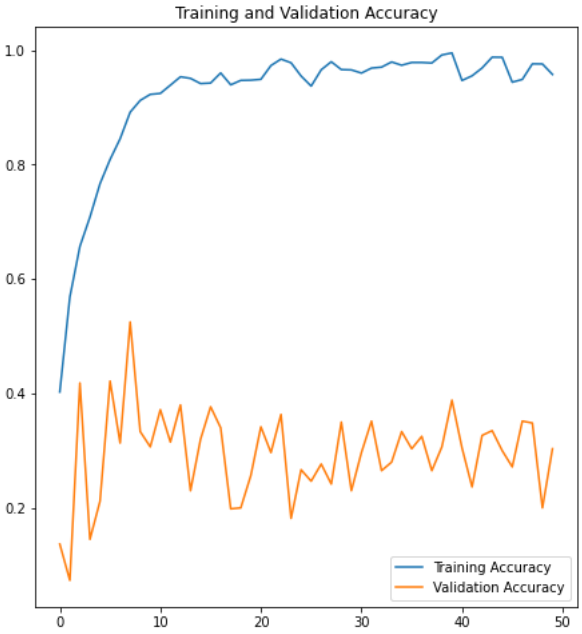


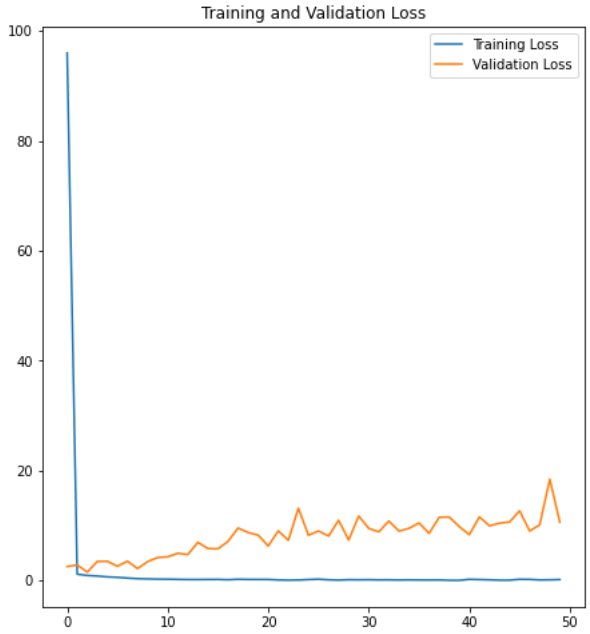

0
20
40
60
80
100
120
0
20
40
60
80
100
120

Q4)

مدل ایجاد شده به شکل زیر می‌باشد عملکرد بدست آمده از این مدل بر روی داده‌ها، نسبت به مدل‌های قبلی ضعیف تر و طولانی تر بود که از علل آن می‌توان نا مناسب بودن حجم داده‌های ما برای آموزش این مدل و همچنین افزایش تعداد پارامترها اشاره نمود.







همان طور که از شکل پیداست در حدود epoch ۸ به بهترین دقت رسیده است و درحدود ۵۰ درصد عمل کرده است و به مرور عملکرد آن افت نموده است.

```
accuracy_score(test_y, pred)
```

```
0.431
```

```
confusion_matrix(test_y, pred)
```

```
array([[190,  34,  30, 115,  31],
       [ 51, 146,  99,  62,  42],
       [ 43,  99, 165,  45,  48],
       [ 82,  18,  23, 249,  28],
       [ 65,  71,  65,  87, 112]])
```

Q5)

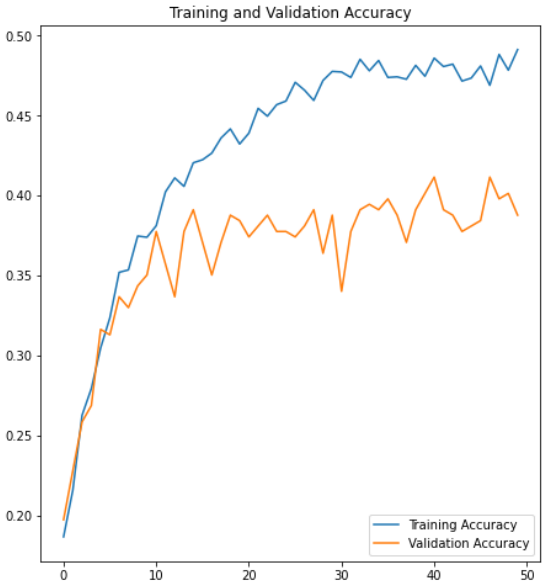
همانطور که در بخش‌هایی قبلی اشاره شده در مدل‌های ایجاد شده به هر چه در مدل به سمت جلو حرکت می‌کنیم امکان استنتاج مسائل پیچیده تر بیشتر می‌شود برای مثال در لایه‌های اول صرفا خطوط یا بک گراند و target از بخش‌های دیگر جدا می‌شود اما در بخش‌های انتهایی تصمیم گیری نهایی و سطح بالا صورت می‌گیرد. در transfer learning نیز از آن جهت که لایه‌های اولیه مفاهیم مشابهی در هر دو حوزه دارند می‌توانیم در ابتدا بصورت مشترک در هر دو مساله از آن‌ها بهره ببریم و تمرکز خودمان را بر روی آموزش لایه یا لایه‌های پایانی بگذاریم و سپس برای اینکه عملکرد کلی مدل را بالا ببریم شروع به آموزش همه بخش‌ها بکنیم.

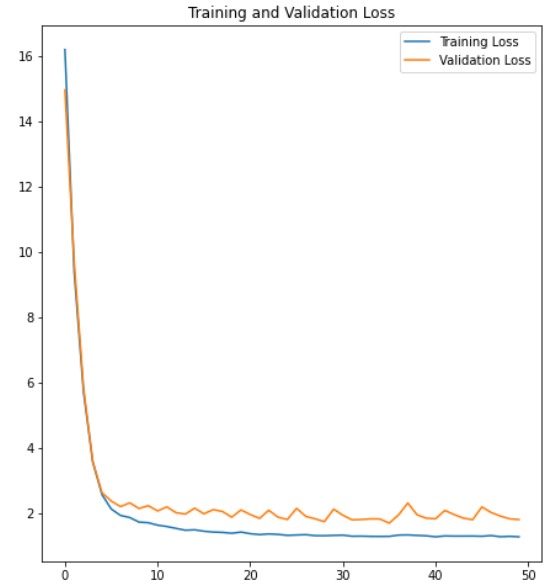
Q6)

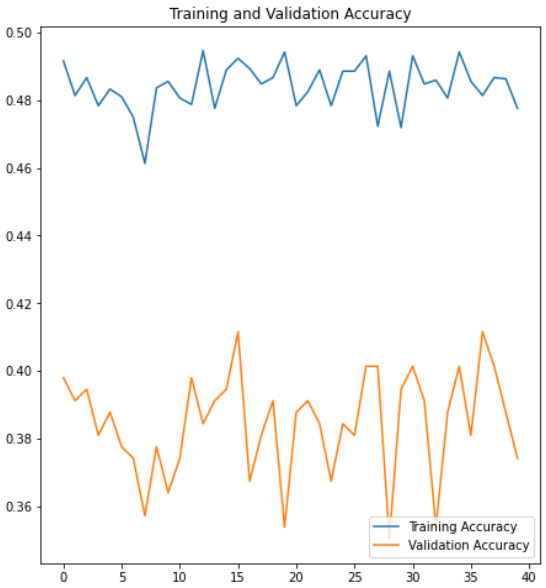
سوال ۶ به دو شکل حل شده است ابتدا همانطور که سوال گفته بود از مدل سوال ۴ بهره برده شد و مدل منتقل گردید اما به دلیل عملکرد ضعیف، مدل سوال یک نیز که بر روی داده‌های دیتاست اول آموزش دیده شده بود استفاده شد و توانستیم با استفاده از انتقال مدل به مساله جدید درصد عملکرد بالاتری دریافت کنیم.

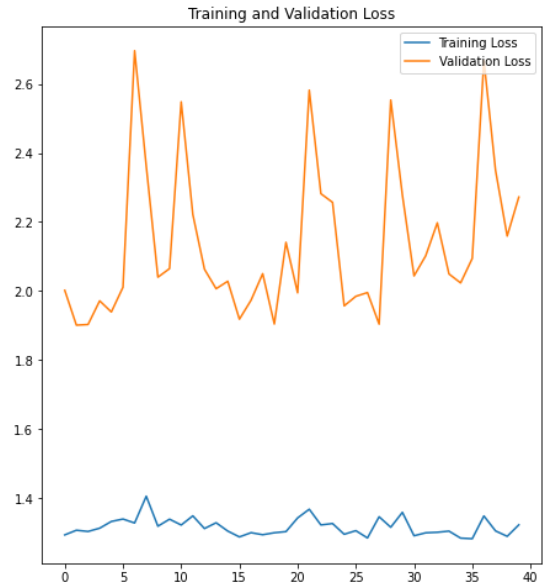
Model 1

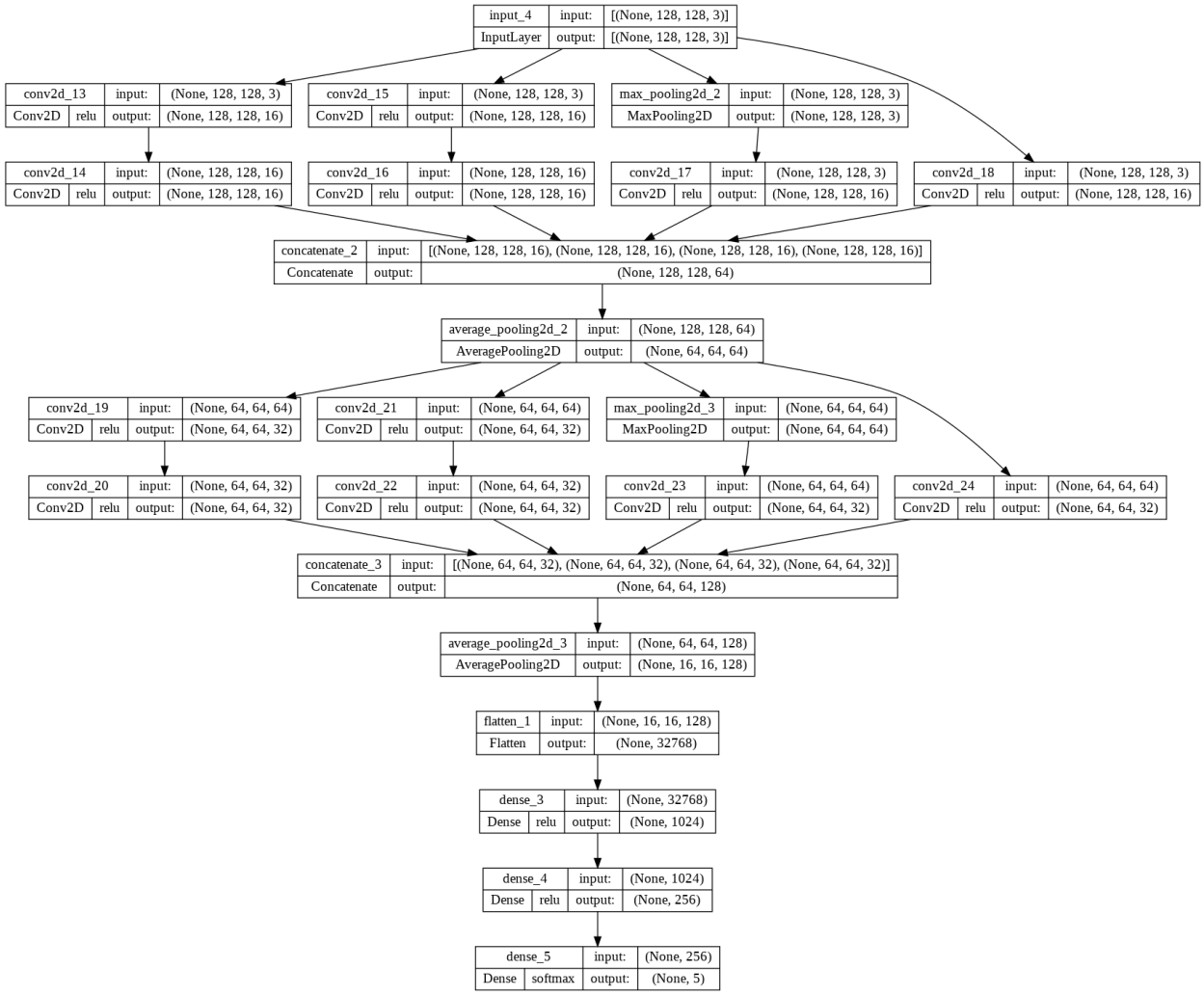
نمودار اولیه برای زمانی است که فقط لایه آخر اجازه آموزش داشته است و نمودار دوم در ادامه آموزش امکان تغییر وزن در همه وزن‌ها را می‌دهد. با توجه به عملکرد مدل در می‌یابیم که به احتمال زیاد مدل ما زمانی که از مدل سوال ۴ استفاده می‌کنیم در یک نقطه محلی به دام می‌افتد و نمی‌تواند بهبود چندانی در خود ایجاد کند.












```
accuracy_score(flower_test_y, pred)
```

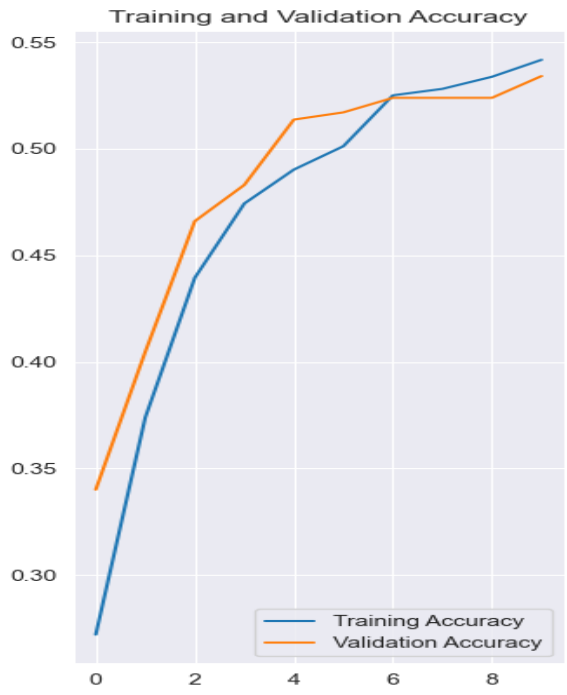
```
0.3555858310626703
```

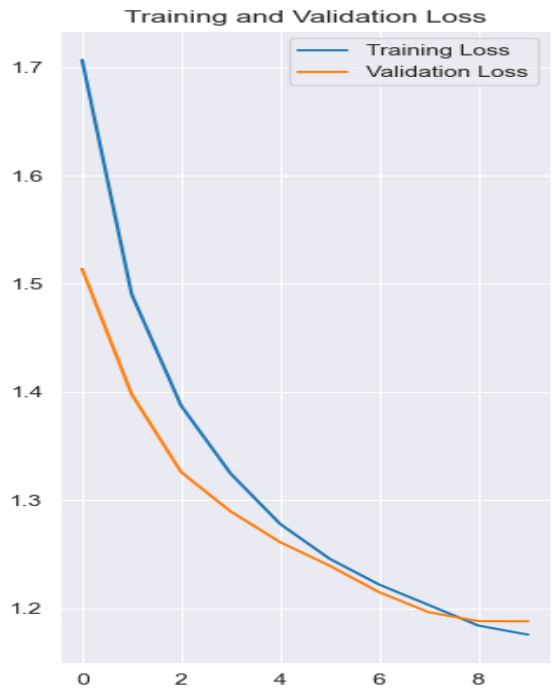
```
confusion_matrix(flower_test_y, pred)
```

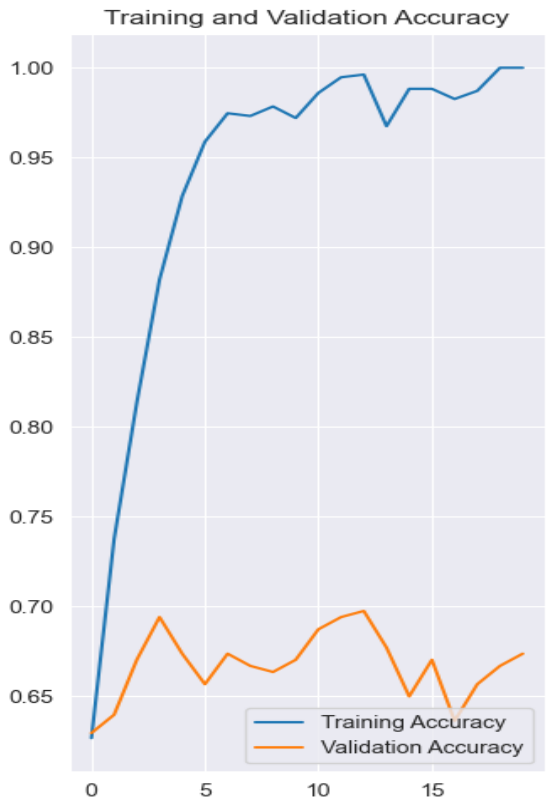
```
array([[39, 29, 39, 20,  2],
       [17, 80, 40, 38,  5],
       [ 8, 17, 63, 28, 12],
       [ 7, 22, 47, 59,  4],
       [ 5, 20, 84, 29, 20]])
```

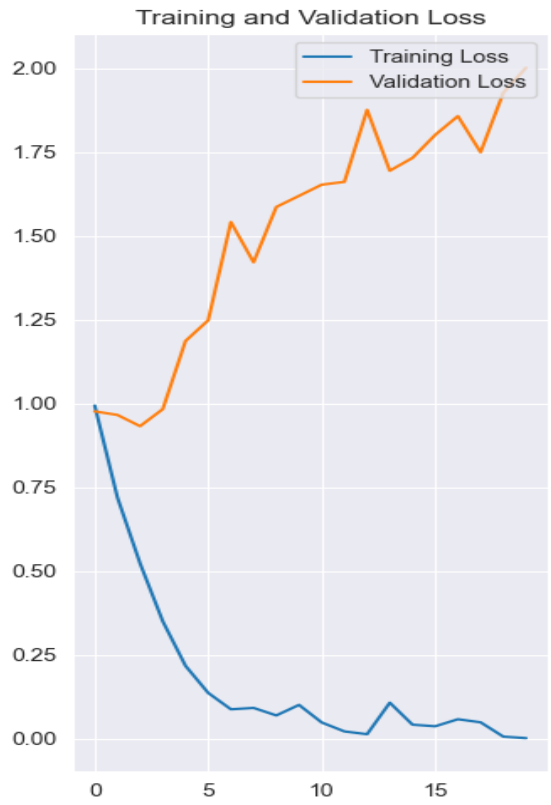
Model 2

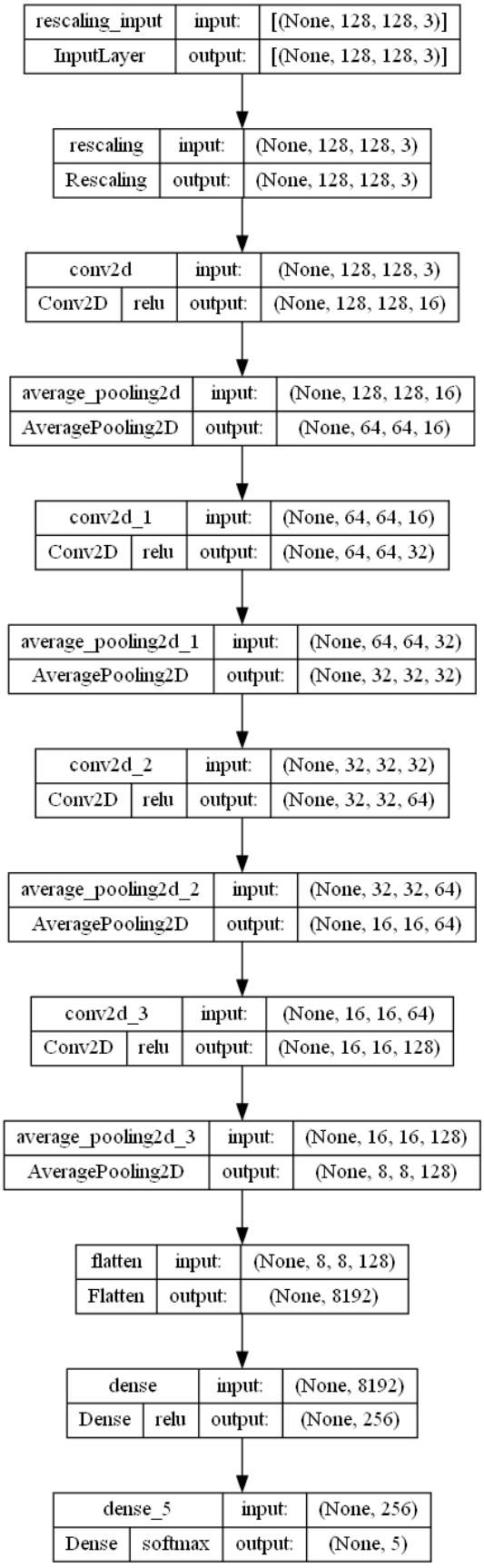
در این حالت عملکرد مدل بسیار بهتر از قبل بوده ، عکس اول مربوط به بخش اول انتقال است که فقط وزن‌های لایه آخر آموزش داده می‌شوند و ۲ نمودار بعدی مربوط به زمانی است که همه ی وزن‌های امکان یادگیری را دارند لازم به ذکر است که مدل در ۵ epoch بیش برازش نموده است و عملکردی در حدود ۶۸ از خود نشان داده است.






Training and Validation Accuracy
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0
5
10
15
Training Accuracy
Validation Accuracy


Training and Validation Loss
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
5
10
15
Training Loss
Validation Loss


rescaling_input | input: | [(None, 128, 128, 3)]
InputLayer | output: | [(None, 128, 128, 3)]
rescaling | input: | (None, 128, 128, 3)
Rescaling | output: | (None, 128, 128, 3)
conv2d | input: | (None, 128, 128, 3)
Conv2D | relu | output: | (None, 128, 128, 16)
average_pooling2d | input: | (None, 128, 128, 16)
AveragePooling2D | output: | (None, 64, 64, 16)
conv2d_1 | input: | (None, 64, 64, 16)
Conv2D | relu | output: | (None, 64, 64, 32)
average_pooling2d_1 | input: | (None, 64, 64, 32)
AveragePooling2D | output: | (None, 32, 32, 32)
conv2d_2 | input: | (None, 32, 32, 32)
Conv2D | relu | output: | (None, 32, 32, 64)
average_pooling2d_2 | input: | (None, 32, 32, 64)
AveragePooling2D | output: | (None, 16, 16, 64)
conv2d_3 | input: | (None, 16, 16, 64)
Conv2D | relu | output: | (None, 16, 16, 128)
average_pooling2d_3 | input: | (None, 16, 16, 128)
AveragePooling2D | output: | (None, 8, 8, 128)
flatten | input: | (None, 8, 8, 128)
Flatten | output: | (None, 8192)
dense | input: | (None, 8192)
Dense | relu | output: | (None, 256)
dense_5 | input: | (None, 256)
Dense | softmax | output: | (None, 5)

```
accuracy_score(flower_test_y, pred)
```

```
0.6485013623978202
```

```
confusion_matrix(flower_test_y, pred)
```

```
array([[ 86,  18,   7,   4,   8],
       [ 13, 131,  17,  11,   8],
       [ 11,  13,  68,   5,  38],
       [  4,  15,   4,  99,   9],
       [  9,  15,  37,  12,  92]], dtype=int64)
```