# تمرین ۱۱

سینا اسکندری ۹۷۵۲۱۰۵۴

## ۱-

### الف)

چون لایه Dense الگوهای سراسری در فضای ویژگی را آموزش می بیند اما لایه های کانولوشنی الگو های محلی را یاد می گیرند.

### ب)

مقدار خروجی کانولوشن سایز ورودی منهای فیلتر به علاوه ۱ است پس برای فیلتر ۵ در ۵ اگر نخواهیم تغییر کند باید مقدار گسترش برابر ۴ باشد. تعداد پارامتر:

(5 * 5 * 5 + 1) * 16 = 2016

### پ)

حالت اول: سایز تصویر به اندازه یکی کمتر از سایز فیلتر کم می شود و بعد سوم آن برابر تعداد فیلتر است(1 + 5 - 32).
پس (28, 28, 3)

حالت دوم: تقریبا مثل حالت اول فقط با این تفاوت که ۲ لایه است و سایز فیلتر فرق می کند (1 + 3 – 1 + 3 – 32).
پس (28, 28, 9)

### ت)

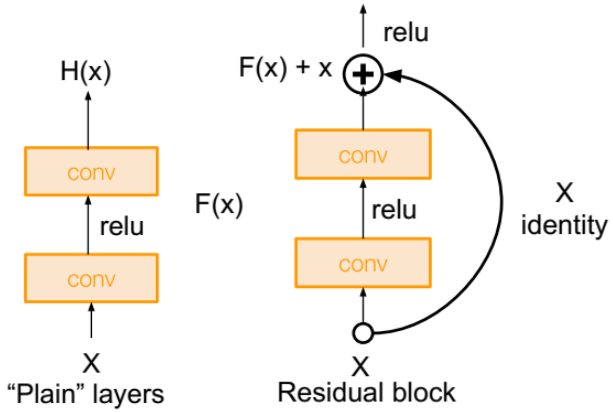
Max pooling چون مقدار بیشینه را انتخاب می کند یعنی پیکسل های روشن تر را نگه می دارد و داده هایی که بک گراند سیاه دارند مفید تر است مثل MNIST.

Average pooling اطلاعات تصویر را smooth می کند و تغییرات شدید خیلی نمایان نمی شود بنابراین مثلا اگر قرار است لبه‌یابی انجام گیرد بهتر است از آن استفاده نشود.

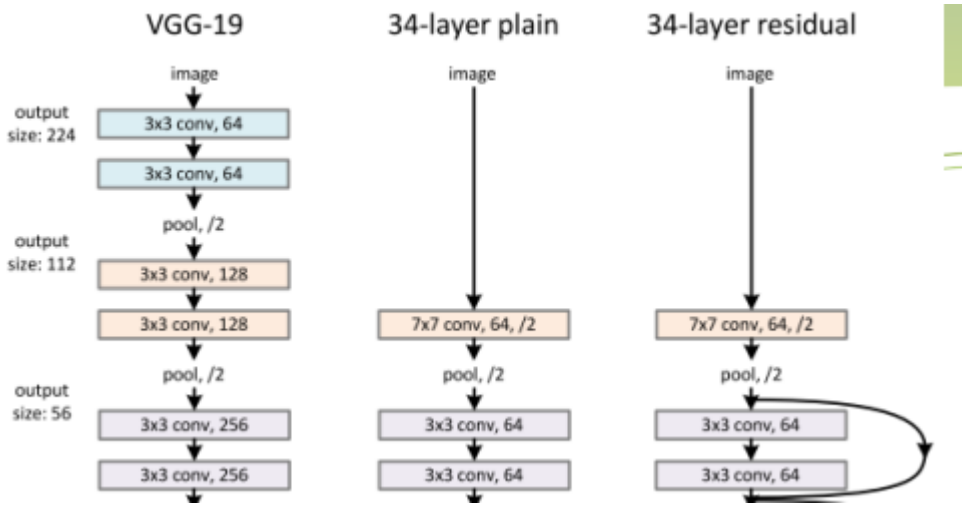
GAP هم از میانگین گیری برای کاهش ابعاد استفاده می کند که برخلاف لایه flatten تعداد پارامتر ها را کاهش می دهد و به نوعی اطلاعات مکانی را خلاصه می کند.

### ث)

VGG نسبت به resnet تعداد پارامتر بیشتری دارد ولی دقت resnet و مدت زمان آن بهتر است. در شبکه resnet از residual block استفاده می شود.



که لایه ها باید $F(x) = H(x) - x$ را بیاموزند. همچنین در resnet در لایه اول نیز سایز تصویر نصف می شود بر خلاف VGG که این اتفاق بعد از ۲ لایه کانولوشنی رخ می دهد و همین امر باعث کاهش تعداد محاسبات و افزایش سرعت می شود.



## ۲-

ساختار مدل ها و تعداد پارامتر ها در نوتبوک مشخص است.

### الف)

در مدل FC میزان دقت و خطا ۰/۴۴ و ۱/۵۵ است و در مدل conv برابر با ۰/۶۹ و ۰/۹۴ است.

بین دقت و خطا هیچ رابطه ریاضی وجود ندارد و خطا به عنوان فاصله بین مقدار واقعی و پیش بینی شده است. به طور مثال اگر در یک دیتاست فقط برای یک مورد اشتباه پیش بینی شود ولی فاصله بین مقدار پیش بینی و واقعی خیلی فاحش باشد که در این حالت مقدار خطا زیاد و مقدار دقت نیز بالا است اما در حالتی که مثلا تعداد اشتباه بیشتر است ولی فاصله بین مقدار پیش بینی و واقعی زیاد نیست مقدار خطا ممکن است از حالت قبل کمتر باشد ولی دقت نیز کمتر است.

### ب)

در مدل FC مدت زمان هر ایپاک به طور میانگین ۵/۶ ثانیه است و در مدل conv برابر ۸/۸ ثانیه است.

### پ)

خیر. در مدل FC با اینکه ۲ برابر پارامتر داریم زمان کمتری صرف می شود و مدت زمان هر ایپاک بیشتر به تعداد عملیات انجام شده در مدل وابسته است.

## ۳-

### الف)

ابتدا با استفاده از کد زیر اگر تصویر مربعی باشد صرفا resize می کنیم ولی اگر مربعی نباشد ابتدا به اندازه کافی padding اضافه کرده تا مربعی شود و سپس به سایز مورد نظر resize می کنیم.

```
def resize_img(img, desired_size = 224):
    # write your code here
    h, w = img.shape[:2]
    if h == w:
        return cv2.resize(img, (desired_size, desired_size))
    s = max(h, w)
    new_img = np.zeros((s, s, 3)).astype(np.uint8)
    ax, ay = (s - img.shape[1])//2, (s - img.shape[0])//2
    new_img[ay:img.shape[0]+ay,ax:ax+img.shape[1]] = img
    new_img = cv2.resize(new_img, (desired_size, desired_size))
    return new_img
```

### ب)

در این قسمت از خود شبکه resnet با تعداد کلاس ۲۴ حالت استفاده می کنیم.

```
# Write your code here
resnet = tf.keras.applications.ResNet50(include_top=True, weights=None, input_shape=(224, 224, 3), classes=24)
resnet.summary()
```

پ)

در این حالت include_top مدل resnet را False می کنیم و همچنین یک لایه Dense هم قبل خروجی قرار می دهیم.

```
fine_tune_resnet = tf.keras.models.Sequential()
# write your code here
conv_base = tf.keras.applications.ResNet50(include_top=False, weights='imagenet', input_shape=(224, 224, 3))
fine_tune_resnet.add(conv_base)
fine_tune_resnet.add(tf.keras.layers.Flatten())
fine_tune_resnet.add(tf.keras.layers.Dense(256, activation='relu'))
fine_tune_resnet.add(tf.keras.layers.Dense(24, activation='softmax'))

conv_base.trainable = False

fine_tune_resnet.summary()
```

برای فریز کردن لایه های resnet مقدار trainable را False می کنیم.

ت)

در حالت اول با وزن های رندوم بعد از ۲۰ ایپاک دقت train برابر ۹۹ درصد شده ولی دقت test برابر ۲۰ درصد که مشخصا overfit شده است.

```
Epoch 19/20
65/65 [==============================] - 66s 1s/step - loss: 0.0129 - acc: 0.9961
Epoch 20/20
65/65 [==============================] - 66s 1s/step - loss: 0.0325 - acc: 0.9894
<keras.callbacks.History at 0x7f0d3c5ced30>
```

```
resnet.evaluate(test_generator)
```

```
33/33 [==============================] - 19s 549ms/step - loss: 14.3637 - acc: 0.1968
[14.363670349121094, 0.19678457081317902]
```

در حالت pretrained که از وزن های imagenet استفاده شده است دقت داده train برابر ۹۹٫۸ درصد و دقت داده تست برابر ۱۰۰ درصد شده است.

```
Epoch 19/20
65/65 [==============================] - 33s 501ms/step - loss: 0.0257 - acc: 0.9978
Epoch 20/20
65/65 [==============================] - 33s 499ms/step - loss: 0.0244 - acc: 0.9981
<keras.callbacks.History at 0x7f0d05331490>
```

```
fine_tune_resnet.evaluate(test_generator)
```

```
33/33 [==============================] - 17s 483ms/step - loss: 0.0183 - acc: 1.0000
[0.018269604071974754, 1.0]
```

منابع:


https://stats.stackexchange.com/questions/280179/why-is-resnet-faster-than-vgg

https://paperswithcode.com/method/global-average-pooling#:~:text=Global%20Average%20Pooling%20is%20a,in%20the%20last%20mlpconv%20layer.

https://medium.com/@bdhuma/which-pooling-method-is-better-maxpooling-vs-minpooling-vs-average-pooling-95fb03f45a9

https://stackoverflow.com/questions/44231209/resize-rectangular-image-to-square-keeping-ratio-and-fill-background-with-black