Deep learning

Homework 4

Parsa Eissazadeh 97412364

سوال 1

در ابتدا ۰ Padding را اضافه می کنیم :

0	0	0	0	0	0	0
0	0	0	10	0	0	0
0	0	0	10	0	0	0
0	0	0	10	0	0	0
0	0	0	10	0	0	0
0	0	0	10	0	0	0
0	0	0	0	0	0	0

کانوالو کردن کرنل در ستون های 1 و 5 عکس (از راست به چپ شماره های 1 تا 5) خروجی صفر میدهد .
در ستون های 2 و 4 و سطر های 2و3و4 نیز مقدار یکسان دارند و برابر با 30 دارند :

$$1*10 + 1*10 + 1*10$$

در ستون های 2 و 4 و سطر های 1و5 نیز مقدار یکسان دارند و برابر با 20 دارند :

$$10*1 + 10*1$$

در ستون ۳ ، سطر های 2،3 و 4 نیز مقدار یکسان دارند :

$$1*10 - 8*10 + 1*10 = -60$$

و در اخر در ستون 3 سطر 1 و 5 ، مقدار یکسان دارند :

$$1*10 - 8*10 = -70$$

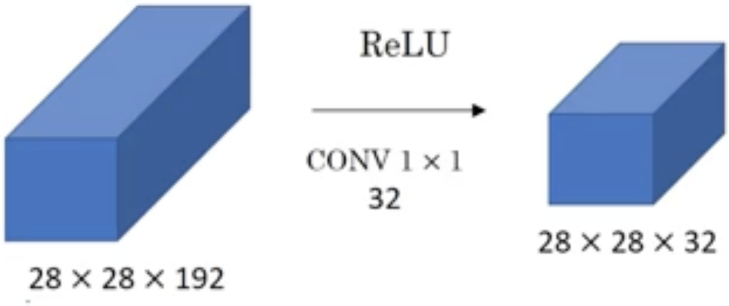
در نتیجه ماتریس خروجی به شکل زیر است :

0	0	0	0	0	0	0
0	0	20	70-	20	0	0
0	0	30	60-	30	0	0
0	0	30	60-	30	0	0
0	0	30	60-	30	0	0
0	0	20	70-	20	0	0
0	0	0	0	0	0	0

سوال 2

کانولوشن 1در 1 یا network in network ، فرآیندیست که در آن تعداد channel های یک ماتریس کاهش پیدا میکنند .

واضح است که در اثر یک کانولوشن 1*1 مقادیر height و width یک ماتریس تغییر پیدا نمی کند .



به کمک این کانولوشن میتوان عمق یک ماتریس را کاهش یافت تا یادگرفتنشان ساده تر باشد و اینکه اگر هم نخواهیم تعداد channel ها را کم کنیم ، باز هم از آنجایی که یک لایه ای اضافه میکند (مثلا relu) که nonlinearity را اضافه می کند که باعث می شود شبکه را راحت تر بیاموزیم .

سوال 3

https://medium.com/@abhisekumarpnd

الف)

وقتی که stride برابر با 2 است ، یعنی هر بار به جای اینکه فیلتر را به اندازه یک پیکسل حرکت دهیم به اندازه دو پیکسل حرکت میدهیم .
حال که فیتلر 3*3 است ، وقتی 2 پیکسل حرکت میکنیم ، جای جدید فیلتر با جای قبلی اش تنها 1 سطر یا ستون مشترک دارد .
در نتیجه به ازای هر فیلتر 2 پیکسل جدید (در سطر و ستون عکس) پوشش داده می شوند و فیلتر ابتدایی 3 پیکسل را پوشش می دهد .

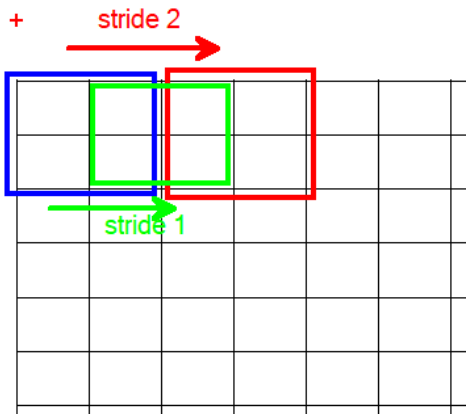
اندازه عکس 28*28 است و یک پیکسل padding داریم ، اندازه به همراه padding میشود 30*30 .

$$30 = 3 + 2 * 13 + 1$$

در نتیجه ابعاد عکس تبدیل می شود به 14*14*1 به ازای هر کرنل .

ب)

سپس همین ماتریس با ابعاد 14*14*3 وارد لایه دوم می شود . که padding از نوع valid دارد و اندازه فیلتر 2*2 است چون stride مقدار 2 دارد ، هر سری فیتلر به اندازه 2 خانه حرکت میکند .



در نتیجه height و width عکس خروجی این لایه نصف height و width عکس ورودی اش است . ← 7 در 7 در 32

پ)

ورودی لایه flatten ابعاد 7*7*32 دارد در نتیجه به هنگام flat شدن به 1*1568 میرسد .

اگر تعداد نورون های لایه Dense را x در نظر بگیریم ، ماتریس وزن های بین لایه flatten و dense (لایه یکی مانده به آخر) برابر است با :

$$(1568 + 1) * x$$

و سپس ابعاد ماتریس وزن های بین لایه dense و لایه خروجی (لایه آخر) که 5 کلاس و در نتیجه 5 نورون دارد برابر است با:

$$(x + 1) * 5$$

سوال 4

نتیجه کانوالو دو ماتریس سوال برابر است با ماتریس زیر :

X_{11} = 3*3 + 1*3 -2*1 = 10

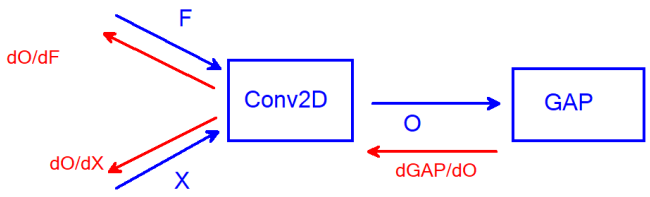
X_{12} = 4*3 + 1*1 -2*5 = 3

X_{21} = 1*3 + 1*4 -2*-1 = 9

X_{22} = 5*3 - 2*-2 -1*1 = 18

10	3
9	18

اگر روی این Global average pooling بگیریم به مقدار 10 می رسیم . پس در نهایت یک خانه با ابعاد 1*1 داریم که مقدارش 10 است . ساختار کلی شبکه به شکل زیر است :



به GAP گرادیان تابع اتلاف نسبت به خروجی نهایی وارد میشود که طبق گفته سوال مقدار 1 دارد .

مشتق GAP نسبت به هر کدام از عناصر O ، مقدار ¼ است . زیرا :

$$GAP = (X_{11} + X_{12} + X_{21} + X_{22}) / 4$$

حال باید مشتق Conv2D را نسبت به F بسنجیم . هر خانه در O برابر است با :

$$O_{ij} = x_{ij} * f_{00} + x_{(i+1)j} * f_{10} + x_{i(j+1)} * f_{01} + x_{(i+1)(j+1)} * f_{11}$$

به ازای هر O و هر f یک مشتق داریم . در مجموع 16 مشتق که هر کدام از این مشتق ها یکی از عناصر ماتریس ورودی هستند .

$$\frac{dO_{mn}}{dF_{ij}} = X_{(i+m)(j+n)}$$

مشتق خروجی کل شبکه نسبت به یک برابر است با :

$$\frac{dLoss}{dGAP} * \frac{dGAP}{dO} * \frac{dO}{dF} = 1 * 1/4 * X_{(i+m)(j+n)}$$

در نتیجه مشتق خروجی شبکه نسبت به F برابر است با ماتریس ورودی تقسیم بر 4 .

سوال 5

به کمک متد `flow_from_directory` ، داده ها را از روی عکس ها ساختیم :

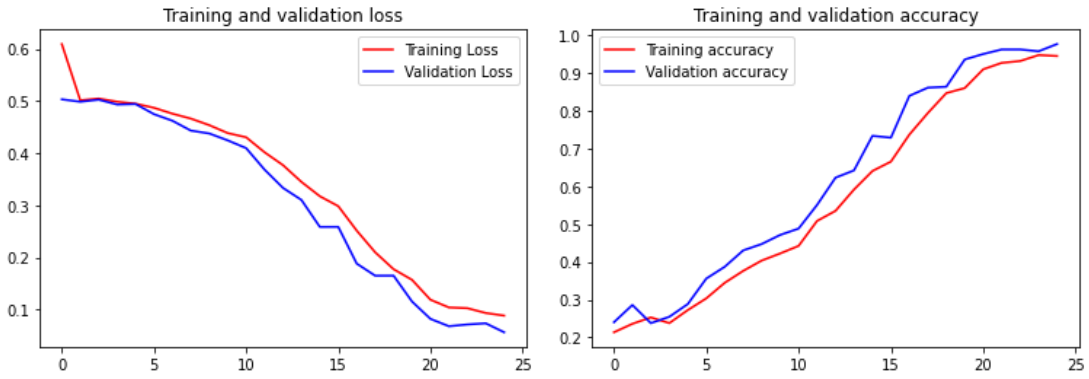
```
datagen = ImageDataGenerator(validation_split=0.2 , rescale= 1./255)

train_iterator = datagen.flow_from_directory(
    directory=images_dir,
    target_size=(224, 224),
    color_mode="rgb",
    batch_size=32,
    class_mode="categorical",
    subset='training'
)

validation_iterator = datagen.flow_from_directory(
    directory=images_dir,
    target_size=(224, 224),
    color_mode="rgb",
    batch_size=32,
    class_mode="categorical",
    subset='validation'
)
```

سپس مدل را با دولایه Conv2D و Maxpooling ساختیم و لایه های flatten و dense را اضافه کردیم .

در ابتدا که تعداد epoch ها 150 تا بود شاهد این بودم که در epoch 35 ، شبکه به accuracy خیلی بالایی روی داده های train رسید که نشان از overfit شدن شبکه بود . برای همین ، دوباره شبکه را با epoch 25 به اجرا درآوردم .

```
Epoch 23/150
52/52 [==============================] - 11s 208ms/step - loss: 0.0377 - accuracy: 0.9903 - val_loss: 0.0349 - val_accuracy: 0.9904
Epoch 24/150
52/52 [==============================] - 11s 212ms/step - loss: 0.0347 - accuracy: 0.9855 - val_loss: 0.0178 - val_accuracy: 1.0000
Epoch 25/150
52/52 [==============================] - 11s 209ms/step - loss: 0.0272 - accuracy: 0.9939 - val_loss: 0.0205 - val_accuracy: 0.9928
Epoch 26/150
52/52 [==============================] - 11s 211ms/step - loss: 0.0194 - accuracy: 0.9945 - val_loss: 0.0116 - val_accuracy: 1.0000
Epoch 27/150
52/52 [==============================] - 11s 213ms/step - loss: 0.0201 - accuracy: 0.9958 - val_loss: 0.0108 - val_accuracy: 1.0000
Epoch 28/150
52/52 [==============================] - 11s 212ms/step - loss: 0.0118 - accuracy: 0.9976 - val_loss: 0.0071 - val_accuracy: 1.0000
Epoch 29/150
52/52 [==============================] - 11s 213ms/step - loss: 0.0098 - accuracy: 0.9982 - val_loss: 0.0096 - val_accuracy: 1.0000
Epoch 30/150
52/52 [==============================] - 11s 213ms/step - loss: 0.0069 - accuracy: 0.9988 - val_loss: 0.0069 - val_accuracy: 0.9976
Epoch 31/150
52/52 [==============================] - 11s 213ms/step - loss: 0.0204 - accuracy: 0.9933 - val_loss: 0.0216 - val_accuracy: 0.9904
Epoch 32/150
52/52 [==============================] - 11s 218ms/step - loss: 0.0348 - accuracy: 0.9909 - val_loss: 0.0350 - val_accuracy: 0.9904
Epoch 33/150
52/52 [==============================] - 11s 211ms/step - loss: 0.0615 - accuracy: 0.9812 - val_loss: 0.0400 - val_accuracy: 0.9904
Epoch 34/150
52/52 [==============================] - 11s 210ms/step - loss: 0.0437 - accuracy: 0.9867 - val_loss: 0.0357 - val_accuracy: 0.9856
Epoch 35/150
52/52 [==============================] - 11s 213ms/step - loss: 0.0311 - accuracy: 0.9921 - val_loss: 0.0091 - val_accuracy: 0.9928
```

اما وقتی که تعداد تکرار ها را هم کم کردیم باز هم به overfit رسیدیم . ولی طبق نمودار ها همچنان loss و accuracy روی train و validation به هم نزدیک هستند .



مدل را ذخیره کردم و دوباره load کردم و داده های test ای که ساخته بودم (و از اول جدا کرده بودم را پاس دادم به متد) . فایل csv ذخیره شده . قسمتی از فایل :

peugeot_206/26.jpg	3
peugeot_206/27.jpg	3
peugeot_206/28.jpg	3
peugeot_206/29.jpg	3
peugeot_206/3.jpg	3
peugeot_206/30.jpg	3
peugeot_206/31.jpg	3
peugeot_206/32.jpg	3
peugeot_206/33.jpg	3
peugeot_206/34.jpg	3
peugeot_206/35.jpg	3
peugeot_206/36.jpg	3
peugeot_206/37.jpg	3
peugeot_206/38.jpg	3
peugeot_206/39.jpg	3
peugeot_206/4.jpg	3
peugeot_206/40.jpg	3
peugeot_206/41.jpg	3
peugeot_206/42.jpg	3
peugeot_206/43.jpg	3
peugeot_206/44.jpg	3
peugeot_206/45.jpg	3

همانطور که مشاهده می شود تا حد خوبی برچسب عکس ها درست پیش بینی شده است .