Assignment NO.8 Solutions

Deep learning | spring 1401 | Dr.Mohammadi

Teaching Assistant(s):

Amirmehdi Nikokaran

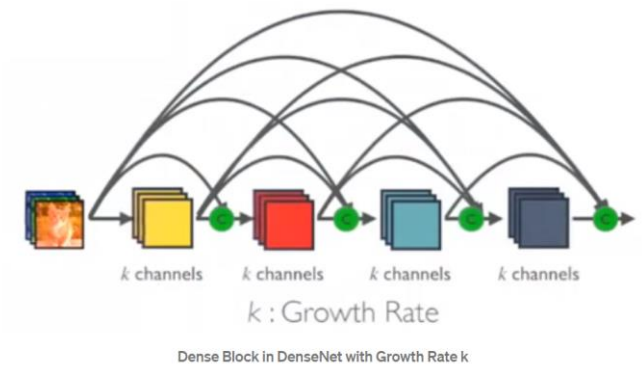
Fatemeh Anvari

Student name : **Amin Fathi**

Student id : **400722102**

Problem 1

معماری Densenet را یادتان هست ؟ که در آن با تجمیع ویژگی های استخراج شده در لایه های قبلی با هم به عنوان ورودی لایه جدید دست به پیشرفت بزرگی زده شده بود (با استفاده از کانولوشن های ۱*۱ کانال های ورودی را میزان میکردیم)



Dense Block in DenseNet with Growth Rate k

همچنین ایده کانولوشن گروهی group convolution را از AlexNet و ResNeXt به یاد می آورید ؟ که در آن بخشی از (و نه همه) فیچر ها یا ویژگی های ورودی در تعیین بخشی از (و نه همه) ویژگی های خروجی نقش داشتند . در واقع در این حالت ویژگی های ورودی به چند دسته متفاوت با هم دسته بندی می شدند و سپس آن ویژگی ها در ترکیب لایه ها کنولوشنی با هم باعث ایجاد دسته ویژگی های متفاوت خروجی می شوند.

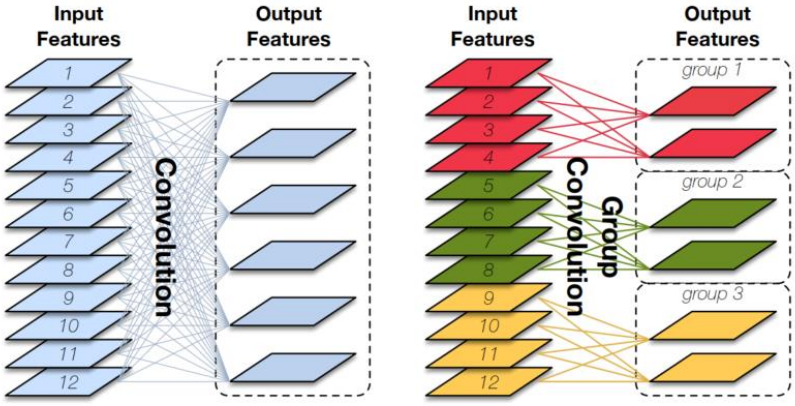
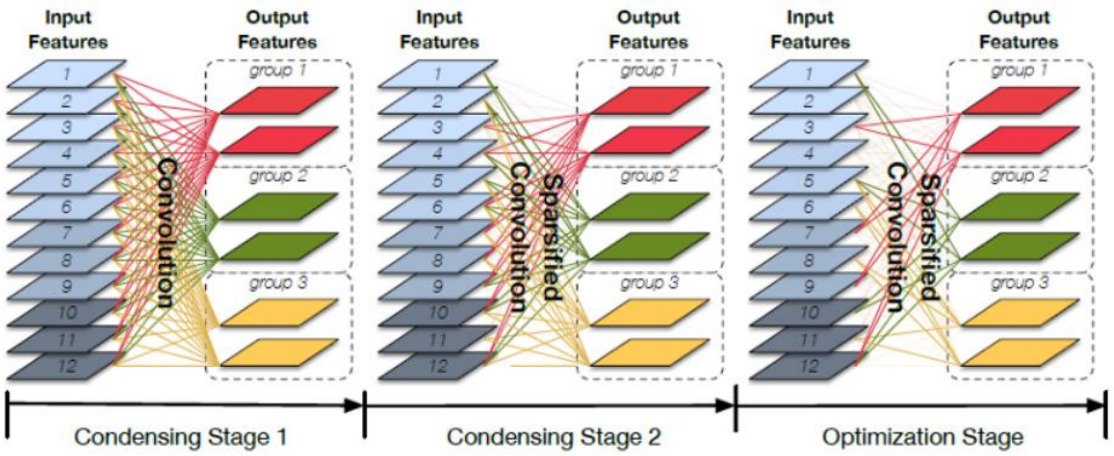


Figure 2. Standard convolution *(left)* and group convolution *(right)*. The latter enforces a sparsity pattern by partitioning the inputs (and outputs) into disjoint groups.

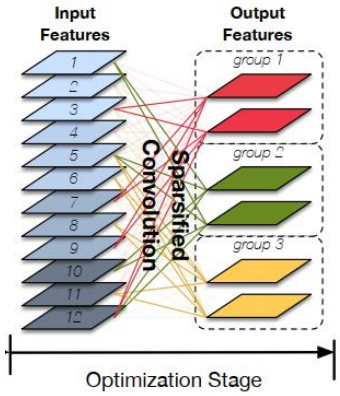
حال نگارندگان مقاله ذکر شده در صورت سوال با بهبود ایده مقاله group Convolution و DnseNet معماری قدرتمندی برای دسته بندی تصاویر ارایه داده است .



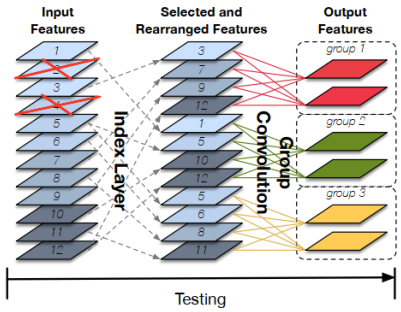
CondenseNet at Different Stages During Training

این روش از دو مرحله تراکم سازی (condensing stage) و بهینه سازی (optimization stage) تشکیل یافته است ، همانطور که در شکل بالا مشاهده می شود مرحله تراکم سازی می تواند خود چندین قسمت داشته باشد

CondenseNet می آموزد که کدام ویژگی های ورودی باید برای کانولوشن گروهی در مرحله تراکم در طول آموزش با هم گروه شوند. اگرچه همه ویژگی های قبلی در هر لایه بعدی مورد نیاز نیستند، پیش بینی اینکه کدام ویژگی ها باید در چه نقطه ای مورد استفاده قرار گیرند دشوار است. برای حل این مشکل، نگارندگان مقاله رویکردی را توسعه داده اند که گروه‌بندی ویژگی‌های ورودی را به‌طور خودکار در طول آموزش یاد می‌گیرد. این عمل به هر گروه فیلتر (output features) اجازه می دهد تا مجموعه ای از مرتبط ترین ورودی های خود را انتخاب کند. علاوه بر این، به چندین گروه اجازه داده می شود ویژگی‌های ورودی را به اشتراک بگذارند و همچنین اجازه داده می شود ویژگی‌هایی توسط همه گروه‌ها نادیده گرفته شوند . نیمه اول تکرارهای آموزشی شامل مراحل تراکم است. در این قسمت بارها و بارها شبکه را با نظم القای پراکندگی برای تعداد ثابتی از تکرارها آموزش می دهند و متعاقباً فیلترهای بی‌اهمیت با وزن‌های قدر کم را حذف می کنند. نیمه دوم آموزش شامل مرحله بهینه سازی است که پس از اینکه در مرحله قبل گروه بندی ویژگی ها را یاد گرفتیم در این مرحله فیلتر ها را یاد میگیریم که در واقع می توان گفت در آخر این مرحله هم با یک group convolution طرف هستیم .



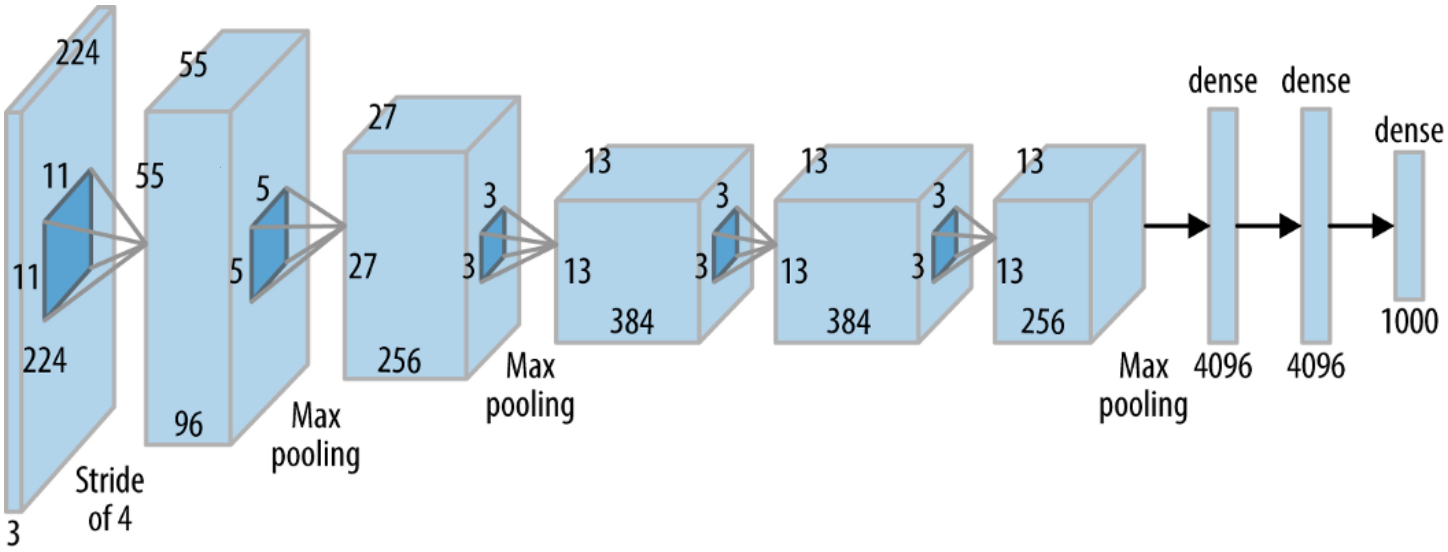
در مورد مرحله تست هم میتوان گفت ، وزن های هرس شده حذف شده و مدل اسپاس شده بر اساس لایه شاخصی که در هنگام train به شبکه ای با الگوی اتصال منظم تبدیل می شود (group conv) و نتایج بر اساس این گروه ها به دست می آید



لازم به ذکر است که در مرحله train چنانچه فرض کنیم شکل ابتدای صفحه پیش را داشته باشیم که در آن c=3 مرحله داریم که c-1 = 2 مرحله آن condensing است ، در هر مرحله 1/c condensing فیلتر ها هرس می شوند .بنابر این در پایان مرحله training ، در هر گروه تنها 1/c وزن های هر گروه باقی است و هرس نشده است .

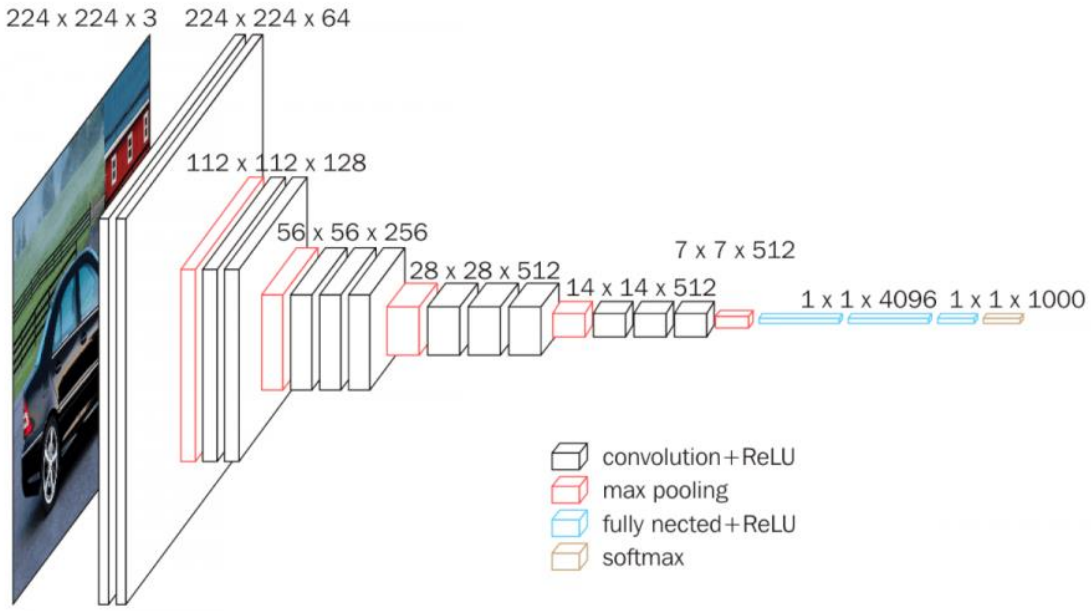
Problem 2

معماری و تعداد پارامتر های alexnet را مشاهده می کنید بر رودی دادگان imagenet که برابر است با ۶۲ میلیون پارامتر :



AlexNet Network - Structural Details													
Input			Output			Layer	Stride	Pad	Kernel size		in	out	# of Param
227	227	3	55	55	96	conv1	4	0	11	11	3	96	34944
55	55	96	27	27	96	maxpool1	2	0	3	3	96	96	0
27	27	96	27	27	256	conv2	1	2	5	5	96	256	614656
27	27	256	13	13	256	maxpool2	2	0	3	3	256	256	0
13	13	256	13	13	384	conv3	1	1	3	3	256	384	885120
13	13	384	13	13	384	conv4	1	1	3	3	384	384	1327488
13	13	384	13	13	256	conv5	1	1	3	3	384	256	884992
13	13	256	6	6	256	maxpool5	2	0	3	3	256	256	0
						fc6			1	1	9216	4096	37752832
						fc7			1	1	4096	4096	16781312
						fc8			1	1	4096	1000	4097000
Total													**62,378,344**

معماری و تعداد پارامتر های vgg16 را مشاهده می کنید بر رودی دادگان imagenet که برابر است با 138 میلیون پارامتر



VGG16 - Structural Details													
#	Input Image			output			Layer	Stride	Kernel		in	out	Param
1	224	224	3	224	224	64	conv3-64	1	3	3	3	64	1792
2	224	224	64	224	224	64	conv3064	1	3	3	64	64	36928
	224	224	64	112	112	64	maxpool	2	2	2	64	64	0
3	112	112	64	112	112	128	conv3-128	1	3	3	64	128	73856
4	112	112	128	112	112	128	conv3-128	1	3	3	128	128	147584
	112	112	128	56	56	128	maxpool	2	2	2	128	128	65664
5	56	56	128	56	56	256	conv3-256	1	3	3	128	256	295168
6	56	56	256	56	56	256	conv3-256	1	3	3	256	256	590080
7	56	56	256	56	56	256	conv3-256	1	3	3	256	256	590080
	56	56	256	28	28	256	maxpool	2	2	2	256	256	0
8	28	28	256	28	28	512	conv3-512	1	3	3	256	512	1180160
9	28	28	512	28	28	512	conv3-512	1	3	3	512	512	2359808
10	28	28	512	28	28	512	conv3-512	1	3	3	512	512	2359808
	28	28	512	14	14	512	maxpool	2	2	2	512	512	0
11	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
12	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
13	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
	14	14	512	7	7	512	maxpool	2	2	2	512	512	0
14	1	1	25088	1	1	4096	fc		1	1	25088	4096	102764544
15	1	1	4096	1	1	4096	fc		1	1	4096	4096	16781312
16	1	1	4096	1	1	1000	fc		1	1	4096	1000	4097000
Total													138,423,208

در مورد تعداد پارامتر های googlenet می توان گفت :

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Table 1: GoogLeNet incarnation of the Inception architecture

که در واقع دارای 6.8 میلیون پارامتر است .

همچنین تعداد پارامتر های NiN برابر است با :

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 96, 54, 54]          11,712
              ReLU-2           [-1, 96, 54, 54]               0
            Conv2d-3           [-1, 96, 54, 54]           9,312
              ReLU-4           [-1, 96, 54, 54]               0
         MaxPool2d-5           [-1, 96, 26, 26]               0
            Conv2d-6          [-1, 256, 26, 26]         614,656
              ReLU-7          [-1, 256, 26, 26]               0
            Conv2d-8          [-1, 256, 26, 26]          65,792
              ReLU-9          [-1, 256, 26, 26]               0
        MaxPool2d-10          [-1, 256, 12, 12]               0
           Conv2d-11          [-1, 384, 12, 12]         885,120
             ReLU-12          [-1, 384, 12, 12]               0
           Conv2d-13          [-1, 384, 12, 12]         147,840
             ReLU-14          [-1, 384, 12, 12]               0
        MaxPool2d-15            [-1, 384, 5, 5]               0
          Dropout-16            [-1, 384, 5, 5]               0
           Conv2d-17             [-1, 10, 5, 5]          34,570
             ReLU-18             [-1, 10, 5, 5]               0
           Conv2d-19             [-1, 10, 5, 5]             110
             ReLU-20             [-1, 10, 5, 5]               0
AdaptiveAvgPool2d-21             [-1, 10, 1, 1]               0
          Flatten-22                   [-1, 10]               0
================================================================
Total params: 1,769,112
Trainable params: 1,769,112
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.19
Forward/backward pass size (MB): 16.44
Params size (MB): 6.75
Estimated Total Size (MB): 23.38
----------------------------------------------------------------
```

منابع :

Difference between AlexNet, VGGNet, ResNet, and Inception | by Aqeel Anwar | Towards Data Science

machine learning - I can't understand part of the GoogleNet incarnation table - Cross Validated (stackexchange.com)

بخش 7.3 کتاب مرجع

problem 2.b

در googlenet به جای pooling از کانولوشن ۱*۱ استفاده می شود که این کار باعث کاهش ابعاد عمق فیچر ها به ابعاد مد نظر ما می شود و در NiN هم با استفاده از GAP(global average pooling) به جای لایه های کاملا متصل تعداد پارامتر ها به نسبت کم باقی می ماند .

منبع : اسلاید های استاد و بخش 7.3 کتاب

Problem 3.b

مورد اول غلط است چرا که برعکس می توان انتظار داشت نرمال سازی دسته ای زمان هر iteration در train را افزایش دهد چرا که به پردازش اضافی مورد هم در روند forward و هم در backward نیاز دارد ، ما انتظار داریم که شبکه سریعتر از آن بدون نرمال سازی دسته ای همگرا شود، بنابراین انتظار می رود آموزش شبکه به طور کلی سریعتر تکمیل شود.

مورد دوم درست است ، مشخصا نرمال سازی دسته ای از ارتباط بین لایه ها کم می کند .

منبع :

cs230exam_win20_soln.pdf (stanford.edu)