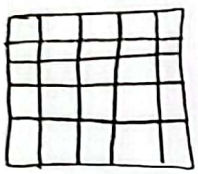
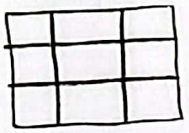
تمرین تئوری درس یادگیری عمیق

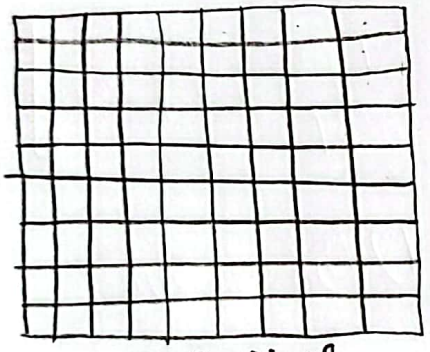
سوال ۱ الف

این سوال مربوط به مبحث receptive field می‌باشد. برای حل این سوال داریم: در واقع راه حل کلی این است که می‌گوییم ۱ پیکسل در $n+1$ معادل 3×3 پیکسل در n است و... داریم

یک نورون در لایه‌ی چهارم ⟸ 3×3 پیکسل در لایه سوم ⟸ 5×5 پیکسل در لایه دوم



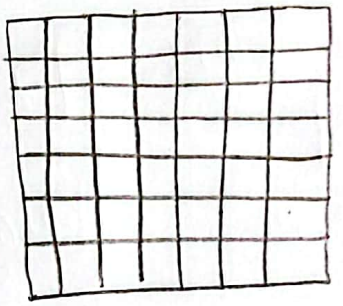




9×9 pixel
در لایه‌ی صفر

⟸

7×7 Pixel
لایه اول



⟸

بنابراین 81 Pixel ⟸ ✓

سوال ۱ ب

لایه‌های pooling ابعاد feature map ها را کاهش می‌دهند و در نتیجه پارامتر های شبکه و محاسبات را کاهش می‌دهد. همچنین لایه‌های pooling ویژگی‌های موجود در یک منطقه از feature map ایجاد شده توسط یک لایه کانولوشن را خلاصه می‌کند و در نتیجه عملیات بیشتر بر روی ویژگی‌های خلاصه شده به جای ویژگی‌های دقیقاً موقعیت یافته (تولید شده توسط لایه‌ی کانولوشن) انجام می‌شود. این باعث می‌شود که مدل نسبت به تغییرات در موقعیت ویژگی‌ها در تصویر ورودی مقاوم‌تر باشد

ج

سایز تصویر ورودی ۳۵×۳۵ و تعداد کانال‌ها 16

سایز فیلتر مورد استفاده 3×3 و طول گام 1 و تعداد فیلترها برابر 32

نوع: Padding: same

به ازای هر بار اعمال فیلتر یک تصویر 35×35 داریم. اگر 32 فیلتر اعمال کنیم سایز خروجی ما 35×35×32 می‌شود.

پارامترها ⟵ $\underbrace{3\times3\times16\times32}_{\text{وزن‌ها}} + \underbrace{32}_{\text{بایاس}} = \boxed{4640}$

د

Input [227×227×3] —conv1→ [227×227×256] → output1 (استفاده: Padding: same می‌دانیم)

پارامتر: $256\times5\times5\times3+256 = 19456$

output1 [227×227×256] —conv2→ [227×227×128] → output2

پارامتر: $128\times5\times5\times256+128 = 819328$

output2 [227×227×128] —MaxPool→ $\frac{227-5}{2}+1 = 112$ ⟹ $112\times112\times128$ → out3

پارامتر = ۰

Flatten → ورودی شبکه: $112\times112\times128 = 1605632$

fully connected → [512 → out4]

$112\times112\times128\times512+512 = 822084096$

out4 (512) → [100 → out5]

$512\times100+100 = 51300$

$\boxed{\text{همه پارامترها} = 829974180}$

تعداد پارامتر های این شبکه بسیار بالا است. برای آن که پارامترها را کم کنیم می توانیم از روش های مختلفی استفاده نماییم می توانیم از Strid های بزرگ تر نیز استفاده کنیم. یکی از راه های ساده این است که Pooling را تغییر دهیم. از دیگر مشکلات آن می توان redundancy به دلیل Strid 2×2 در لایه Pooling اشاره کرد. درواقع پارامترهای شبکه هنگامی تبدیل Flatten زیادی می شود و باید به آن جا تا جایی که می توان با پارامتر کمی رسید. همچنین در conv2 نیز پارامتر زیادی تولید می شود بهتر است آن جا را هم سعی کنیم پارامتر آن کم شود. راه آسان عوض کردن ایجاد conv است که باعث کاهش پارامتر می شود اما راه جالب تری می تواند آن باشد که جای conv1 و conv2 عوض شود داریم:

Input [227×227×3] —conv2→ out1: [227×227×128]
پارامتر: 128×5×5×3+128 = 9728

out1 [227×227×128] —conv1→ out2: [227×227×256]
پارامتر: 256×5×5×128+256 = 819456

out2 —maxpool 7×7, strid 5×5→ $\frac{227-7}{5}+1 = 45$ ⟹ out3: 45×45×256

Flatten ← ورودی شبکه: 45×45×256 = 1163520

—FC→ out4 = 512
پارامتر: 512×1163520 + 512 = 595729752

→ out5 = 100
پارامتر = 100×512+100 = 51300

پارامتر = 597 766 756

ه) با افزایش لایه‌ها درواقع می‌توان خاصیت غیرخطی بودن را بیشتر ایجاد کرد تا اضافه کردن نورون با همان 2 لایه زیرا $f(g(x))$ از $f(x)+g(x)$ خاصیت (موازی چاق شدن) عمیق شدن غیرخطی بیشتری ایجاد می‌کند. بنابراین علی‌الرغم آن که 2 لایه خاصیت General Factorization دارد ولی بیشتر دوست داریم تا شبکه عمیق و لاغر داشته باشیم تا کم‌عمق و چاق. مورد دیگری که هست از نظر تعداد نورون می‌باشد. هنگامی که عمیق می‌شویم به تعداد نورون کمتری نسبت به حالتی که چاق می‌شویم نیاز داریم پس اگر عمیق شویم تعداد نورون کمتر و درنتیجه پارامتر کمتری نیاز خواهیم داشت. بنابراین در اکثر مواقع از بیشتر از 2 لایه استفاده می‌کنیم. (درواقع با عمیق شدن می‌توان نورون کمتری به ازای پیچیدگی یکسان داشت)

و) برای آن که vanishing را توضیح دهیم می‌دانیم با اجرای الگوریتم Error back propagation ممکن است به دلیل کوچکی گرادیان‌ها هنگامی که به لایه‌های اول می‌رسیم گرادیان کلی برای آپدیت آن قدر کم باشد که عملاً آپدیتی صورت نگیرد و درنتیجه شبکه به‌صورت خیلی خیلی کند آپدیت شود و رسیدن به شبکه بهینه سخت شود. این مشکل ناشی از بعضی توابع فعالساز مثل tanh و sigmoid است و اگر در ناحیه‌ی اشباع قرار بگیرند مشتقشان بسیار به صفر نزدیک خواهد بود و اگر چندین تا از این‌ها در هم ضرب شود (که در EP در لایه‌های اولیه اتفاق می‌افتد) گرادیان کلی برای آپدیت صفر و شبکه دیگر آپدیت نمی‌شود. در exploding گرادیان کلی برای آپدیت زیاد و زیاد می‌شود و عملاً باعث واگرایی EP می‌شود برخلاف vanishing. vanishing و exploding تفاوت دارد است که exploding برخلاف آن چه که بین vanishing براثر وزن‌ها در شبکه اتفاق می‌افتد و نه براساس توابع فعالسازی. وزن‌های لایه‌های پایین‌تر بیشتر ممکن هست برایشان exploding رخ دهد چون گرادیان‌ها ضرب در اعداد بیشتری می‌شوند. پایین می‌تواند باعث شود که لایه‌های پایین به سمت ناپایداری بروند. ما می‌توانیم از این که این تفاوت‌ها می‌تواند باعث شود که لایه‌های پایین به سمت ناپایداری بروند. ما می‌توانیم از این که مقدار loss که در training data داریم را ببینیم در هر step از یک update به update دیگر زیاد است

سؤال 2

الف

without zeropad

$$A = \begin{bmatrix} W_{11} & W_{10} & 0 & W_{01} & W_{00} & 0 & 0 & 0 & 0 \\ 0 & W_{11} & W_{10} & 0 & W_{01} & W_{00} & 0 & 0 & 0 \\ 0 & 0 & 0 & W_{11} & W_{10} & 0 & W_{01} & W_{00} & 0 \\ 0 & 0 & 0 & 0 & W_{11} & W_{10} & 0 & W_{01} & W_{00} \end{bmatrix}$$

in full size

with zero pad

$$A = \begin{bmatrix} W_{00} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ W_{01} & W_{00} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & W_{01} & W_{00} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & W_{01} & 0 & 0 & 0 & 0 & 0 & 0 \\ W_{10} & 0 & 0 & W_{00} & 0 & 0 & 0 & 0 & 0 \\ W_{11} & W_{10} & 0 & W_{01} & W_{00} & 0 & 0 & 0 & 0 \\ 0 & W_{11} & W_{10} & 0 & W_{01} & W_{00} & 0 & 0 & 0 \\ 0 & 0 & W_{11} & 0 & 0 & W_{01} & 0 & 0 & 0 \\ 0 & 0 & 0 & W_{10} & 0 & 0 & W_{00} & 0 & 0 \\ 0 & 0 & 0 & W_{11} & W_{10} & 0 & W_{01} & W_{00} & 0 \\ 0 & 0 & 0 & 0 & W_{11} & W_{10} & 0 & W_{01} & W_{00} \\ 0 & 0 & 0 & 0 & 0 & W_{11} & 0 & 0 & W_{01} \\ 0 & 0 & 0 & 0 & 0 & 0 & W_{10} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & W_{11} & W_{10} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & W_{11} & W_{10} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W_{11} \end{bmatrix}$$

سوال 2 | ب

این نوع کانولوشن در واقع در جاهایی استفاده می‌شود که تعدادی از پیکسل پیکسل‌های ما مقداری از بین رفته است مثل آثار هنری. در این کانولوشن‌ها maske می‌شود و یادگیری با پیکسل‌های دیده‌شده انجام می‌شود. رابطه‌ی آن به صورت زیر می‌باشد

$$X' = W \overset{\text{کانولوشن}}{\uparrow} (X \overset{\text{ورودی}}{\circ} M \overset{\text{ماسک}}{\frown})$$

ج |

یک spatial seprebale conv به صورت ساده یک کرنل را به دو کرنل کوچک‌تر تقسیم می‌کند. یک کیس معروف این است که یک کرنل 3x3 را به یک 1x3 و یک 3x1 تقسیم کنیم مثل زیر:

$$\begin{bmatrix} 3 & 6 & 9 \\ 4 & 8 & 12 \\ 5 & 10 & 15 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

در واقع به جای اجرای یک کانولوشن با وضرب 9 کانولوشن هر یک با 3 ضرب را اجرا می‌کنیم (مجموع 6 ضرب) تا به اثر یکسان برسیم. با تعداد کمتر ضرب پیچیدگی محاسباتی پایین و شبکه توانایی این را پیدا خواهد کرد که سریع‌تر ران شود.