نام و نام خانوادگی:

رضا قربانی پاجی

شماره دانشجویی:

403206565

تمرین چهارم درس یادگیری ماشین

سوال 1)

سوال ①

for Dese layer : تعداد پارامترها = (ورودی) × (خروجی)

J = (W × H) ، I = خروجی که مسئله می‌خواهد

تعداد پارامترها = (W × H) × I

for convolution : پارامترها = (فیلتر × عمق ورودی) × تعداد فیلترها

= (K × K × 1) × تعداد فیلترها (عمق لایه بعدی)

انعاد Dense از CNN بیشتره چون در Dense هر pixel رو به عنوان یه ورودی در نظر می‌گیره اما CNN با استفاده از حرکت دادن فیلترها روی تصاویر به تعداد پارامتر خیلی کمتری نیاز دارد

- قسمت دوم :

شبکه های dense تعداد پارامتر خیلی بیشتری برای آموزش نسبت به CNN نیاز دارند و همین امر باعث می‌شود که تعداد داده های مورد نیاز برای آموزش dense نسبت به CNN به شدت بیشتر باشد . اما با توجه به وظیفه ای که داریم جواب این سوال همیشه این نیست که CNN همیشه با تعداد داده کمتری نسبت به dense عمل می‌کند لزوما صحیح نیست اما در اکثر موارد اینطور است

ب)

1- در شبکه های تماما متصل، خروجی هر نورون به صورت یک بردار بدون ساختار فضایی است و نرمال‌سازی دسته‌ای میان تمامی نورون ها در یک لایه انجام می‌شود. در مقابل در CNN، نرمال‌سازی پس از اعمال فیلترهای کانولوشنی صورت می‌گیرد که ساختار مکانی ویژگی‌ها را حفظ می‌کند. در CNN ها، BN معمولاً روی هر کانال و برای همه موقعیت فضایی به طور جداگانه ولی با پارامترهای یکسان اعمال می‌شود. این امر باعث می‌شود همگن‌سازی توزیع ویژگی‌ها در فضا تسهیل شود و به نوعی "internal covariate shift" کاهش یابد. نتیجه اینکه BN در شبکه های کانولوشنی معمولاً تاثیر پایدارتری دارد

2- در بسیاری از پیاده سازی ها هنگامی که از BN استفاده می‌کنیم دیگر نیازی به پارامتر بایاس در لایه کانولوشنی نداریم. زیرا BN خود دارای پارامترهای شیفت و اسکیل است که جایگزین نقش بایاس و به نوعی جایگزین برخی پارامترهای دیگر می‌شود

3- در مرحله استنتاج، پارامترهای میانگین و واریانس در زمان آموزش ثابت هستند. BN در این مرحله صرفاً یک تبدیل خطی ثابت بر روی ویژگی‌های ورودی اعمال می‌کند این تبدیل را حتی می‌توان با وزن های کانولوشن ادغام کرد و در نتیجه هیچ سربار محاسباتی اضافه و اختلالی در عملکرد ایجاد نمی‌شود. در عمل، استفاده از BN در مرحله استنتاج معمولاً عملکرد را پایدار نگه می‌دارد. زیرا خروجی شبکه نسبت به تغییرات ناگهانی توزیع ورودی حساسیت کمتری دارد

4- در شرایطی مانند در دسترس بودن داده های کم، اندازه mini batch های بسیار کوچک، یا برخی معماری های خاص، مثل GAN ها، BN کارایی مورد انتظار را نداشته باشد یا حتی نتیجه معکوس دهد، روش های دیگری وجود دارند.

جایگزین های نظیر Layer norm... ، weigh normalization، Group normalization، Instance normal... و Batch Renorm...

ج)

1- در هر مرحله encoder، ابعاد فضایی به نصف کاهش می یابد. ساختار U-net 4 مرحله Downsampling در encoder دارد:

input: 256×256

layer1: 128×128

layer2: 64×64

layer 3: 32×32

layer 4: 16×16 ⟶ در عمیق ترین لایه میانی که ویژگی ها در آن بازنمایی می شوند 16×16 خواهد بود

-2

تعداد پارامترها = (اندازه $k \times k \times$ کانال ورودی $+1$) $\times$ تعداد فیلتر ها

$$= (3 \times 3 \times 64 + 1) \times 128 = 73{,}856$$

-4

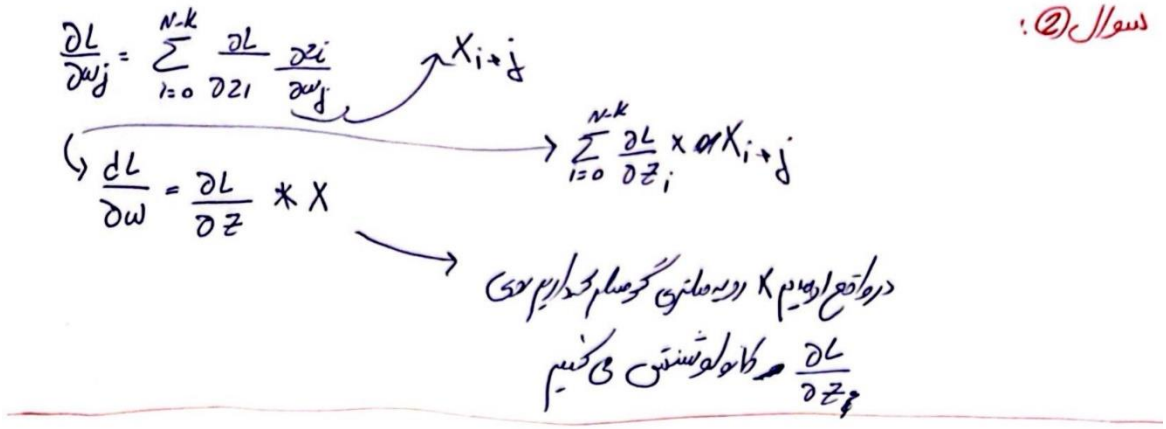
de-convolution (Deconvnet):

در این روش، از نقشه های ویژگی لایه های میانی یا خروجی شبکه شروع کرده و با استفاده از یک فرایند وارون سازی (شامل برعکس کردن عملیات کانولوشن، غیرفعالسازی و unpooling)، تصویری در فضای ورودی بازسازی می شود. این تصویر نشان می دهد هر فیلتر یا لایه دقیقاً کدام ساختارهای تصویر اصلی را برجسته می کند

Up-convolution (transposed convolution):

در شبکه های Encoder-decoder پس از اینکه در بخش encoder ویژگی ها فشرده شدند، بخش encoder با استفاده از لایه های Up-conv ویژگی های فشرده شده مجددا به ابعاد تصویر ورودی برمی گرداند. این بازسازی تصویری به ما می گوید هر مرحله فشرده سازی چه الگویی را استخراج کرده، و چگونه آن ها را در ابعاد اصلی تصویر بازنمایی می کند

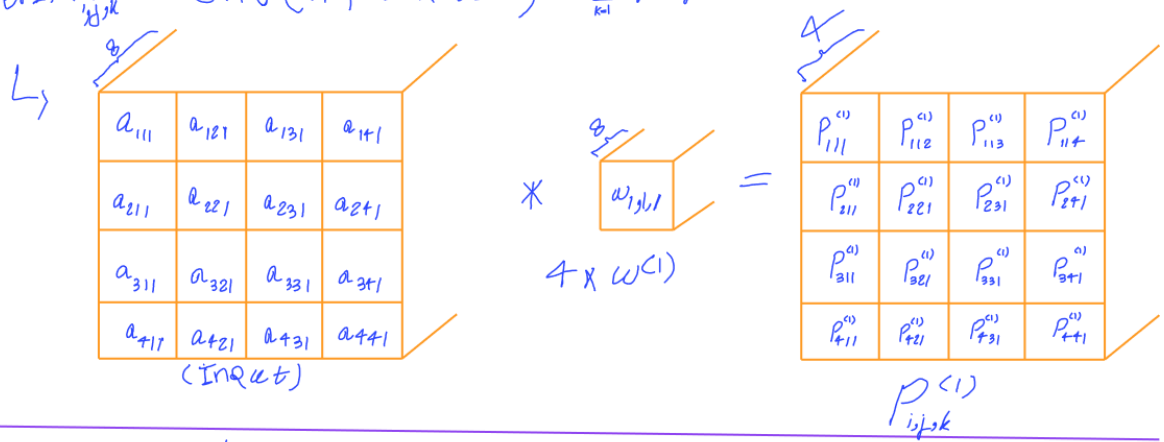
سوال2)


سوال ②:
$\frac{\partial L}{\partial w_j} = \sum_{i=0}^{N-k} \frac{\partial L}{\partial z_i} \frac{\partial z_i}{\partial w_j}$
x_{i+j}
$\sum_{i=0}^{N-k} \frac{\partial L}{\partial z_i} \times$ or x_{i+j}
$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial z} * x$
درواقع اومدیم x رو به صورتی گوسلم کردایم روی
$\frac{\partial L}{\partial z_i}$ و کانولوشنش می کنیم

سوال3)

Feed Forward

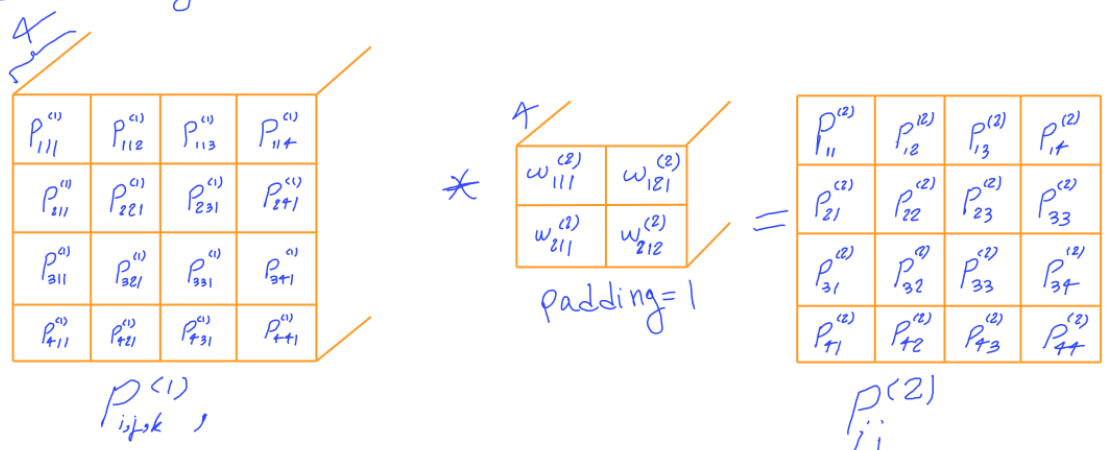
layer1: $P^{(1)}_{i,j,k} = conv(input \times w^{(1)}) = \sum_{k=1}^{8} a_{ijk} \times w_{ijk}$



$P^{(1)}_{i,j,k}$

---

layer2: $P^{(2)}_{i,j} = conv(P^{(1)}_{i,j,k}, w^{(2)})$



$P^{(1)}_{i,j,k}$, $P^{(2)}_{i,j}$

$$P^{(2)}_{i,j} = \sum_{i,j,k=1} \sum_{k=1}^{8} \left[P^{(1)}_{(i,j,k)} \times w^{(2)}_{(1,1,k)} + P^{(1)}_{(i,j+1,k)} \times w^{(2)}_{(1,2,k)} + P^{(1)}_{(i+1,j,k)} \times w^{(2)}_{(2,1,k)} + P^{(1)}_{(i+1,j+1,k)} \times w^{(2)}_{(2,2,k)} \right]$$

layer 3: $P_{ij}^{(3)} = \text{avg-pool}_{2\times 2}(P_{i,j}^{(2)})$

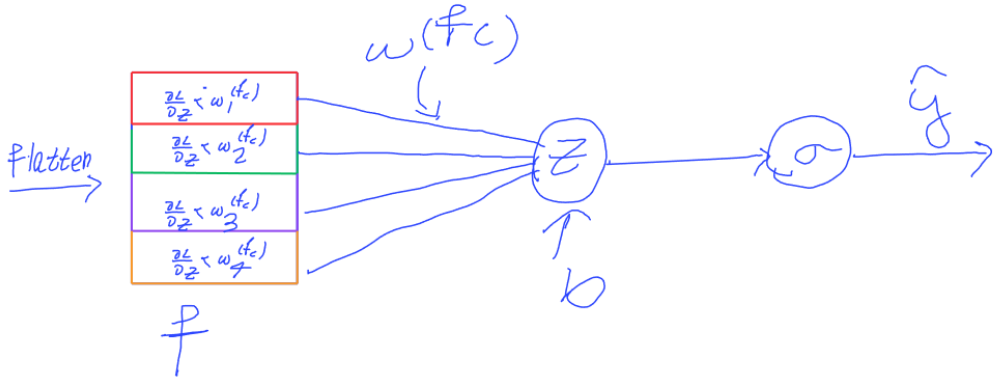
$P_{11}^{(2)}$	$P_{12}^{(2)}$	$P_{13}^{(2)}$	$P_{14}^{(2)}$
$P_{21}^{(2)}$	$P_{22}^{(2)}$	$P_{23}^{(2)}$	$P_{33}^{(2)}$
$P_{31}^{(2)}$	$P_{32}^{(2)}$	$P_{33}^{(2)}$	$P_{34}^{(2)}$
$P_{41}^{(2)}$	$P_{42}^{(2)}$	$P_{43}^{(2)}$	$P_{44}^{(2)}$

avg, stride = 2

$P_{ij}^{(2)}$

---

fc layer



$$z = f \times (w^{(fc)})^T + b$$

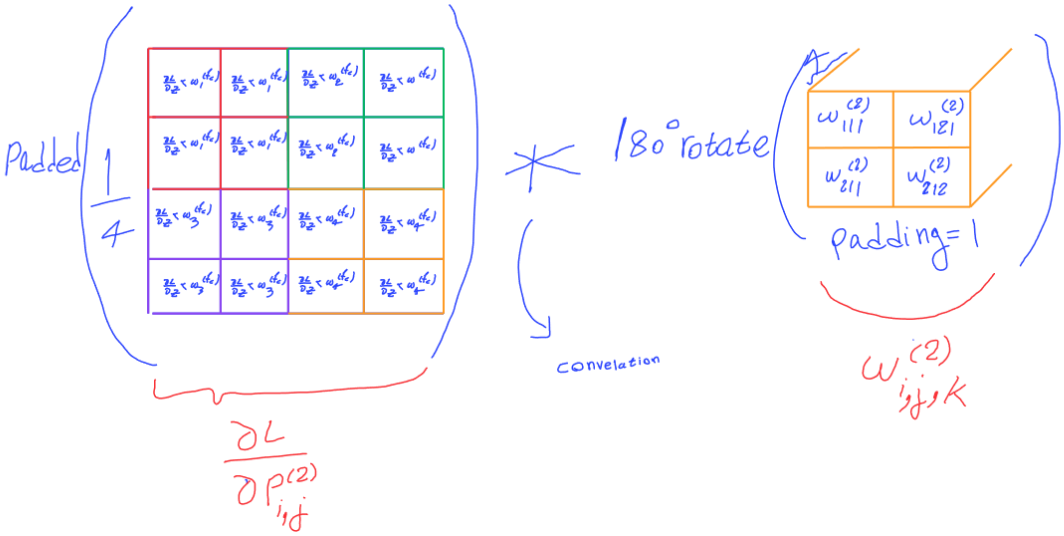
$$\frac{\partial L}{\partial f} = \frac{\partial L}{\partial Z} \times \frac{\partial Z}{\partial p^{(3)}_{i,j}} = \frac{\partial L}{\partial z} \times w^{(fc)} \rightarrow \frac{\partial L}{\partial p^{(3)}_{i,j}} = \frac{\partial L}{\partial f}.\text{reshape}\left(p^{(3)}_{i,j}\right).$$

(i)

$$Z = (w^{(fc)})^T f + b \rightarrow \frac{\partial Z}{\partial f} = w^{(fc)}$$

$$\frac{\partial L}{\partial f} = \begin{bmatrix} \frac{\partial L}{\partial z} \times w_1^{(fc)} \\ \frac{\partial L}{\partial z} \times w_2^{(fc)} \\ \frac{\partial L}{\partial z} \times w_3^{(fc)} \\ \frac{\partial L}{\partial z} \times w_4^{(fc)} \end{bmatrix} \xrightarrow[p^{(3)}_{i,j}]{\text{Reshape}} \begin{bmatrix} \frac{\partial L}{\partial z} \times w_1^{(fc)} & \frac{\partial L}{\partial z} \times w_2^{(fc)} \\ \frac{\partial L}{\partial z} \times w_3^{(fc)} & \frac{\partial L}{\partial z} \times w_4^{(fc)} \end{bmatrix}$$

(ii)



ب)

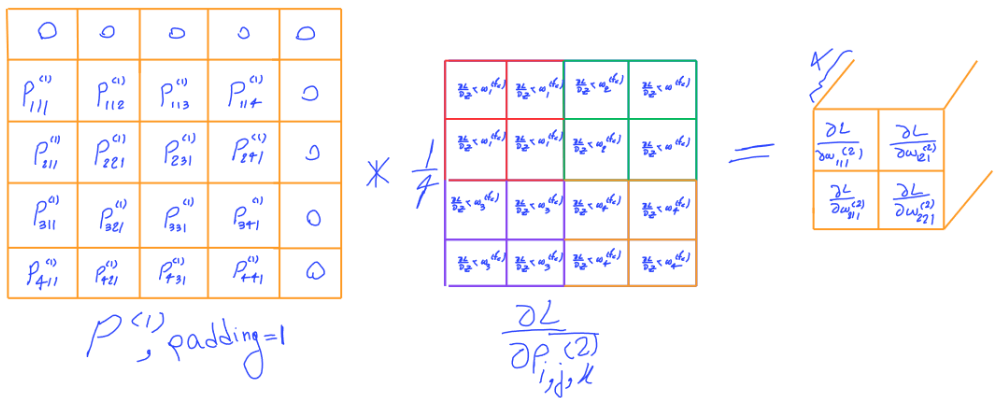
$$\frac{\partial L}{\partial w^{(1)}_{i,j,k}} = \frac{\partial L}{\partial Z} \times \frac{\partial Z}{\partial P^{(3)}_{i,j}} \times \frac{\partial P^{(3)}_{i,j}}{\partial P^{(2)}_{i,j,k}} \times \frac{\partial P^{(2)}_{i,j,k}}{P^{(1)}_{i,j,k}} \times \frac{\partial P^{(1)}_{i,j,k}}{\partial w_{i,j,k}}$$

در بخش آ (ii) محاسبه شده

$$\frac{\partial L}{\partial w^{(1)}_{i,j,k}} = conv\left(\frac{\partial L}{\partial P^{(1)}_{i,j,k}}, input\right)$$

پ)

$$\frac{\partial L}{\partial w^{(2)}_{i,j,k}} = conv\left(\frac{\partial L}{\partial P^{(2)}_{i,j,k}}, P^{(1)}_{i,j,k}\right)$$



سوال 4)

پاسخ بخش (الف):

تفاوت اصلی Dense Connections و Residual Connections :

- در ResNet خروجی هر لایه به لایه بعدی اضافه می‌شود (skip connection) و تنها یک مسیر مستقیم رو به جلو ایجاد می‌کند.
- در DenseNet ، هر لایه خروجی خود را به تمام لایه‌های بعدی متصل می‌کند) به صورت الحاق یا (concatenation، بنابراین هر لایه مجموعه‌ای از تمام ویژگی‌های استخراج شده توسط لایه‌های قبلی را به عنوان ورودی دریافت می‌کند.

به بیان ساده: در ResNet یک میان‌بر (short-cut) بین لایه‌های غیرمتوالی وجود دارد، اما در DenseNet هر لایه به‌طور متراکم به تمامی لایه‌های قبل از خود مرتبط است.

چگونگی کاهش مشکل vanishing gradient در DenseNet و مزیت آن:

اتصالات متراکم DenseNet باعث می‌شوند که گرادیان‌ها هنگام پس‌انتشار، بتوانند مسیرهای متعدد و کوتاه‌تری برای برگشت به لایه‌های اولیه داشته باشند. این امر سبب می‌شود گرادیان‌ها بدون محو شدن، به راحتی به عقب منتقل شوند. در نتیجه:

- یادگیری پایدارتر و عمیق‌تر صورت می‌گیرد.
- استفاده کاراتر از پارامترها و بهبود کیفیت ویژگی‌های استخراج‌شده امکان‌پذیر می‌شود.

محاسبه تعداد کانال خروجی در Dense Block با نرخ رشد:k

در DenseNet ، اگر نرخ رشد (growth rate) برابر k باشد، هر لایه k ویژگی جدید تولید کرده و به کانال‌های موجود می‌افزاید.

- اگر ورودی بلوک 32 کانال داشته باشد و $k = 24$ باشد، پس از 3 لایه (هر لایه 24 کانال جدید) تعداد کانال‌های خروجی به صورت زیر است:

$$24 \times 3 + 32 = 32 + 72 = 104$$

پاسخ بخش (ب):

شبکهGoogleNet

پیمانه پیدایش با هدف کاهش پیچیدگی محاسباتی به‌وجود آمد. قصد داریم ابتدا با پیمانه پیدایش و سپس شبکه GoogleNet آشنا شویم.

در شکل 3، یک لایه کانولوشن به دو صورت نشان داده شده است. ورودی هر دو حالت 28×28×192 و خروجی هر دو نیز 28×28×32 است. با محاسبه تعداد کل عملیات‌های انجام شده در هر حالت، پیچیدگی محاسباتی دو حالت نشان داده شده را مقایسه کنید. مشخص کنید که افزودن فیلتر کانولوشن 1×1 (به‌عنوان پیمانه) با چند درصد کاهش یا افزایش در محاسبات همراه است؟

محاسبات حالت معمولی (بدون پیمانه):

- ابعاد ورودی: 28×28×192
- فیلتر: 5×5
- تعداد فیلتر خروجی: 32

تعداد عملیات:

$$(28\times28)\times192\times32\times(5\times5) = 784 \times 192 \times 32 \times 25 = 120{,}422{,}400$$

محاسبات با پیمانه (ابتدا کانولوشن 1×1 برای کاهش کانال):

1. کانولوشن 1×1 از 192 کانال به 16 کانال
   تعداد عملیات:

$$16\times192\times(28\times28) = 2{,}408{,}448$$

2. کانولوشن 5×5 با ورودی 16 کانال و خروجی 32 کانال

تعداد عملیات:

$$(28\times28)\times16\times32\times(5\times5) = 784 \times 16 \times 32 \times 25 = 10{,}035{,}200$$

مجموع عملیات در حالت پیمانه: $2{,}408{,}448 + 10{,}035{,}200 = 12{,}443{,}648$

مقایسه:

بدون پیمانه: حدود 120 میلیون عملیات

با پیمانه: حدود 12.4 میلیون عملیات

کاهش تقریباً 90٪ در پیچیدگی محاسباتی اتفاق می‌افتد.

توضیح ایده پیمانه‌های پیدایش (Inception) در GoogleNet:

در معماری GoogleNet ، ایده به‌کارگیری پیمانه‌های پیدایش (Inception Modules) برای کاهش پیچیدگی محاسباتی و در عین حال استخراج ویژگی‌های چند-مقیاسی (Multi-scale) از ورودی مطرح شد. در این پیمانه‌ها، به جای استفاده از یک لایه کانولوشن بزرگ با تعداد فیلتر زیاد، از چند مسیر موازی با فیلترهای مختلف (1×1، 3×3، 5×5) و حتی یک مسیر شامل Max Pooling استفاده می‌شود. هر مسیر روی همان ورودی عمل می‌کند و در نهایت خروجی‌ها در جهت عمق (Channel) با هم الحاق (Concatenate) می‌شوند.

دلایل استفاده از این ترکیب چندمسیری عبارتند از:

استخراج ویژگی‌های چند-مقیاسی :فیلترهای 1×1، 3×3 و 5×5 هر کدام محدوده فضایی متفاوتی از ویژگی‌ها را استخراج می‌کنند. فیلترهای بزرگ‌تر (مثلاً 5×5) به بافت‌های گسترده‌تر حساس‌اند، در حالی که فیلترهای کوچک‌تر (1×1 یا 3×3) جزئیات موضعی‌تر را بررسی می‌کنند.

کاهش محاسبات با Convolution: 1×1 قبل از اعمال فیلترهای بزرگ‌تر، ابتدا از کانولوشن 1×1 برای کاهش تعداد کانال‌ها استفاده می‌شود. این کار تعداد ضرب و جمع‌های مورد نیاز برای کانولوشن‌های بعدی (مثلاً 3×3 یا 5×5) را به شدت کاهش می‌دهد.

تنوع در استخراج ویژگی‌ها :استفاده از Max Pooling در کنار کانولوشن‌ها، نوعی ویژگی مجزاست که تنوع ارائه اطلاعات به لایه بعدی را افزایش می‌دهد.

تعداد محاسباتی برای یک پیمانه پیدایش:

فرض کنید ورودی یک پیمانه 28×28×192است. این پیمانه چند شاخه (Branch) دارد:

شاخه اول:

کانولوشن 1×1 با 64 فیلتر:

$$28 \times 28 \times 192 \times 64 = 9,633,792$$

شاخه دوم:

ابتدا کانولوشن 1×1 با 96 فیلتر:

$$28\times28\times192\times96 = 14,450,688$$

سپس کانولوشن 3×3 با 128 فیلتر بر خروجی قبلی (ورودی حالا 96×28×28):

$$28\times28\times96\times128\times3\times3 = 86,704,128$$

شاخه سوم:

ابتدا کانولوشن 1×1 با 16 فیلتر :

$$28\times28\times192\times16 = 2,408,448$$

سپس کانولوشن 5×5 با 32 فیلتر:

$$28\times28\times16\times32\times5\times5 = 10,035,200$$

شاخه چهارم(Max Pool و سپس 1×1)

پس از 3×3 Max Pool و سپس کانولوشن 1×1 با 32 فیلتر روی 192×28×28:

$$28\times28\times192\times32 = 4,816,896$$

هر شاخه پس از کاهش کانال‌ها، اعمال فیلتر بزرگ‌تر را بسیار ارزان‌تر می‌کند.

ترکیب نتایج شاخه‌های مختلف1×1) ، 33×، 5 5×و (Max Pool یک نمایش غنی و چندمقیاسی از ویژگی‌ها را با هزینه محاسباتی کمتر نسبت به استفاده مستقیم از فیلترهای بزرگ به دست می‌دهد.

در نهایت، محاسبات کل یک پیمانه پیدایش با جمع عملیات تمام شاخه‌ها محاسبه شده و معمولاً بسیار کمتر از حالتی است که مستقیماً از کانولوشن‌های بزرگ و بدون کاهش کانال استفاده شود. این تعادل میان تنوع فیلترها و کاهش محاسبات، از موفقیت‌های کلیدی GoogleNet در زمان معرفی آن بوده است.

سوال سوم)

معماری GoogleNet به دلیل استفاده از پیمانه‌های چندمسیره و کانولوشن‌های 1×1 در میان مسیرها، نسبت به vanishing gradientمقاوم‌تر از معماری‌های عمیق پیشین است. افزون بر این، حضورauxiliary classifier ها (خروجی‌های کمکی میانی) کمک می‌کند تا گرادیان از میانه شبکه نیز به عقب منتشر شده و لایه‌های اولیه بهتر آموزش ببینند. این امر جریان گرادیان را بهبود داده و پایداری یادگیری را افزایش می‌دهد. با این حال، در معماری‌ها و تکنیک‌های مدرن که مسئله vanishing gradientکمتر مطرح است، نیاز بهauxiliary classifier ها کمتر احساس می‌شود.

پاسخ بخش (ج):

افزودن فرمول‌ها برای تشریح تفاوت کانولوشن عادی و:Deformable

در کانولوشن عادی، خروجی در نقطه (x, y) با استفاده از جمع وزن‌دار مقادیر ورودی از یک گرید منظم محاسبه می‌شود. فرض کنید هسته کانولوشن (Kernel) به اندازه $F \times F$ باشد. فرمول کانولوشن عادی را می‌توان به شکل زیر بیان کرد:

$$G(x, y) = \sum_{(i,j) \in Kernel} w_{i,j} \cdot F(x + i, y + j)$$

در این معادله، (i, j) محدوده‌ای از اندیس‌ها است که محل نمونه‌برداری هسته کانولوشن از تصویر (یا نقشه ویژگی) Fرا مشخص می‌کند. این هسته یک گرید منظم است: برای هر موقعیت (x, y), نمونه‌برداری دقیقاً $(x + i, y + j)$ انجام می‌شود و هیچ تغییری در مکان این نقاط رخ نمی‌دهد.

اما در کانولوشن تغییرپذیر(Deformable Convolution) ، به جای این که از یک گرید منظم ثابت استفاده شود، مکان نمونه‌برداری می‌تواند با استفاده از آفست‌هایی که شبکه در حین آموزش یاد می‌گیرد، تغییر کند. فرمول Deformable Convolutionبه صورت زیر است:

$$G(x, y) = \sum_{(i,j) \in Kernel} wi, j \cdot F(x + i + \Delta x_{i,j},\ y + j + \Delta y_{i,j})$$

در این رابطه، $\Delta x_{i,j}$ و $\Delta y_{i,j}$ آفست‌هایی هستند که موقعیت نمونه‌برداری را از یک گرید منظم به یک گرید تغییرپذیر تبدیل می‌کنند. این آفست‌ها توسط شبکه (معمولاً یک زیرشبکه کوچک) محاسبه می‌شوند و در زمان آموزش یاد گرفته می‌شوند.

انعطاف‌پذیری نسبت بهGeometric Transformation :

شبکه‌های Deformable می‌توانند به طور خودکار الگوهای پیچیده در تصاویر را با اعمال تغییر در محل نمونه‌برداری فیلترها دنبال کنند. مثلاً اگر شیء مورد نظر در تصویر کمی کشیده یا کج شود، یا شکل آن اندکی تغییر کند، فیلترهای Deformable می‌توانند نقاط نمونه‌برداری خود را بازتنظیم کنند تا الگوی مورد نظر را همچنان استخراج نمایند. این خاصیت باعث می‌شود این شبکه‌ها نسبت به تصاویر با تغییرات هندسی غیرصلب (Non-rigid Transformations) انعطاف‌پذیرتر باشند.

مفهوم Offset در Deformable Convolution :

هر آفست از دو مقدار $\Delta x_{i,j}$ و $\Delta y_{i,j}$ تشکیل می‌شود که به مختصات اصلی موقعیت (i,j) هسته کانولوشن اضافه می‌شوند. به عبارت دیگر، به جای نمونه‌برداری از پیکسل $(x+i, y+j)$ در تصویر یا نقشه ویژگی، فیلتر از $(x+i+\Delta x_{i,j},\ y+j+\Delta y_{i,j})$ نمونه‌برداری می‌کند. این جابه‌جایی‌های قابل یادگیری، هسته کانولوشن را قادر می‌سازند تا خود را با الگوهای هندسی پیچیده‌تر در تصویر تطبیق دهند.

نحوه محاسبه offset :

آفست‌ها مستقیماً از داده ورودی و از طریق یک زیرشبکه (زیرمدل) کوچک محاسبه می‌شوند. مراحل معمول محاسبه آن‌ها به این صورت است:

1. **زیرشبکه تولید آفست:**

معمولاً در پیاده‌سازی Deformable Convolution ، یک لایه کانولوشن اضافی (با خروجی 2×(تعداد نقاط هسته) کانال) روی همان ویژگی‌های ورودی اعمال می‌شود.

- اگر هسته کانولوشن $F \times F$ باشد، این زیرشبکه باید $2F^2$ مقدار خروجی دهد (2 مقدار برای هر نقطه نمونه‌برداری: Δx و Δy.
- این لایه کانولوشن اغلب هسته 3×3 و پدینگ مناسب دارد تا ابعاد فضایی حفظ شده و آفست‌ها برای هر مکان محاسبه شوند.

2. **یادگیری در خلال آموزش:**

وزن‌های این لایه کانولوشن که آفست‌ها را تولید می‌کند، طی فرایند آموزش با استفاده از backpropagation و بر مبنای معیار خطای نهایی شبکه (مثلاً خطای طبقه‌بندی یا تشخیص) به‌روزرسانی می‌شوند.
در نتیجه، شبکه می‌آموزد که کدام جابه‌جایی‌ها (offsets) برای بهبود استخراج ویژگی از داده‌ها مفیدند. به طور تجربی، این آفست‌ها معمولاً یاد می‌گیرند روی مناطق مهم یا ساختارهای خاص در تصویر متمرکز شوند و فیلتر را در جهاتی تغییر دهند که حساسیت به شکل، زاویه، یا مکان اجزا در تصویر بیشتر شود.

سوال 5)

پاسخ بخش (الف):

در کانولوشن‌های گسترشی(Dilated Convolution) ، به جای نمونه‌برداری متوالی و مجاور پیکسل‌ها در یک ناحیه ثابت)مثلاً(k×k ، فاصله‌ای میان خانه‌های نمونه‌برداری وجود دارد. این فاصله با پارامتر گسترش (Dilation) کنترل می‌شود. زمانی که مقدار گسترش D>1 باشد، فیلتر با «پریدن» روی پیکسل‌ها (با فاصله‌های منظم) اعمال می‌شود و بنابراین یک «ناحیه مؤثر» بزرگ‌تر را بدون افزایش تعداد پارامترها پوشش می‌دهد.

مفهوم:

- در کانولوشن عادی (بدون گسترش)، اگر هسته یک فیلتر 3×3 داشته باشیم، این هسته مستقیماً روی یک بلوک 3×3 از پیکسل‌ها اعمال می‌شود.
- در کانولوشن گسترشی با ضریبD ، هسته همان 3×3 است، اما میان هر دو پیکسل افقی و عمودی D-1 پیکسل جهش وجود دارد. مثلاً باD=2 ، فیلتر از مختصات(i,j) ، سپس(i,j+2) ، (i+2,j)، ...نمونه‌برداری می‌کند. این باعث می‌شود ناحیه‌ای به اندازه بزرگ‌تر (D-1)*(3-1)+3) برای هسته 3×3 (تحت پوشش قرار گیرد، بدون این‌که تعداد پارامترهای فیلتر افزایش یابد.

مزایا:

- محدوده دید (Receptive Field) شبکه با هزینه کم افزایش می‌یابد. به عبارت دیگر، شبکه می‌تواند اطلاعات فضایی در مقیاس بزرگ‌تر را ببیند، بدون اینکه مجبور باشیم لایه‌های بیشتری اضافه کنیم یا اندازه کرنل را افزایش دهیم.
- برای وظایف بینایی کامپیوتری مانند درک صحنه‌های پیچیده (مثلاً در تقسیم‌بندی معنایی تصاویر)، این امر مفید است زیرا شبکه می‌تواند با همان تعداد پارامتر، بافت و ساختارهای بزرگ‌تری را تحلیل کند.

فرمول ریاضی:

فرمول کانولوشن گسترشی (Dilated Convolution) به صورت زیر است:

$$(K *_D I)(i,j) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} K(m,n) I(i + Dm, j + Dn)$$

در این رابطه:

- I ورودی با ابعاد M×N است.
- K هسته کانولوشن با اندازه F×F است.
- D پارامتر گسترش است.

- وقتی D=1 باشد، همان کانولوشن عادی را داریم.
- وقتی D>1 باشد، فواصل میان پیکسل‌های ورودی که فیلتر روی آن‌ها اعمال می‌شود، افزایش می‌یابد و محدوده مؤثر فیلتر برابر خواهد بود با .F+(F−1)(D−1)
- $Height\,(rows) = M - (kernel\_size - 1) = M - (DF - D + 1 - 1) = M - DF + D$
- $Width\,(columns) = N - (kernel\_size - 1) = N - (DF - D + 1 - 1) = N - DF + D$

نتیجه خروجی کانولوشن گسترشی نیز با ابعاد $(M - DF + D)(N - DF + D)$خواهد بود که نشان می‌دهد با افزایشD ، بدون اضافه کردن پارامترهای بیشتر، می‌توان محدوده فضایی تحت پوشش فیلتر را افزایش داد.

پاسخ بخش (ب):

برای نشان دادن اینکه کانولوشن گسترشی معادل کانولوشن با کرنل متسع شده (K ') است، نیاز داریم مفهوم محصول کرونکر (Kronecker Product)را معرفی کنیم. محصول کرونکر یک روش ریاضی برای توسعه ماتریس‌ها است که در اینجا برای گسترش کرنل Kاستفاده می‌شود.

تعریف Kronecker Product:

اگر k یک ماتریس $n \times m$ باشد و A یک ماتریس $q \times p$، Kronecker Product آن‌ها $(A \otimes K)$ یک ماتریس با ابعاد $(q.n) \times (p.m)$ است که به صورت زیر تعریف می‌شود:

$$K' = A \otimes K$$

If **A** is an $m \times n$ matrix and **B** is a $p \times q$ matrix, then the Kronecker product **A** $\otimes$ **B** is the $pm \times qn$ block matrix:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix},$$

more explicitly:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & \cdots & a_{11}b_{1q} & \cdots & \cdots & a_{1n}b_{11} & a_{1n}b_{12} & \cdots & a_{1n}b_{1q} \\ a_{11}b_{21} & a_{11}b_{22} & \cdots & a_{11}b_{2q} & \cdots & \cdots & a_{1n}b_{21} & a_{1n}b_{22} & \cdots & a_{1n}b_{2q} \\ \vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots \\ a_{11}b_{p1} & a_{11}b_{p2} & \cdots & a_{11}b_{pq} & \cdots & \cdots & a_{1n}b_{p1} & a_{1n}b_{p2} & \cdots & a_{1n}b_{pq} \\ \vdots & \vdots & & \vdots & \ddots & & \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots & & \ddots & \vdots & \vdots & & \vdots \\ a_{m1}b_{11} & a_{m1}b_{12} & \cdots & a_{m1}b_{1q} & \cdots & \cdots & a_{mn}b_{11} & a_{mn}b_{12} & \cdots & a_{mn}b_{1q} \\ a_{m1}b_{21} & a_{m1}b_{22} & \cdots & a_{m1}b_{2q} & \cdots & \cdots & a_{mn}b_{21} & a_{mn}b_{22} & \cdots & a_{mn}b_{2q} \\ \vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{p1} & a_{m1}b_{p2} & \cdots & a_{m1}b_{pq} & \cdots & \cdots & a_{mn}b_{p1} & a_{mn}b_{p2} & \cdots & a_{mn}b_{pq} \end{bmatrix}$$

Photo from Wikipedia

در اینجا A یک ماتریس است که برای گسترش مکان‌های نمونه‌برداری استفاده می‌شود. اگر A به عنوان ماتریس گسترش با تک 1 در مکان مشخص و بقیه عناصر صفر باشد، محصول کرونکر بین K و A باعث ایجاد یک کرنل جدید K' می‌شود که گسترش‌یافته است و مکان‌های نمونه‌برداری D-1 فاصله دارند.

کرنل گسترش‌یافته:(' K)

برای گسترش کرنل Kبا استفاده از پارامتر گسترش :D

1. ماتریس A را به صورت یک ماتریس هویت $D \times D$ در نظر بگیرید که فاصله‌های صفر اضافی بین المان‌های اصلی دارد.
2. Kronecker Product، K'=A⊗K اعمال می‌شود، که کرنل گسترشی را ایجاد می‌کند. این کرنل ' K حالا فاصله D−1 بین نقاط اصلی گرید نمونه‌برداری دارد.

به عنوان مثال A را ماتریسی 2 در 2 و D را نیز 2 در نظر میگیریم حاصل Kronecker Product به صورت زیر می شود:

x	y
z	r

⊗

1	0
0	0

=

x	0	y	0
0	0	0	0
z	0	r	0
0	0	0	0

پاسخ بخش (ج):

محاسبه محدوده تأثیر (Receptive Field):

در کانولوشن، هر فیلتر تأثیر خود را بر محدوده‌ای از پیکسل‌های ورودی اعمال می‌کند. این محدوده، که به آن **Receptive Field** گفته می‌شود، وابسته به ابعاد فیلتر (w×w) و میزان گسترش (Dilation Factor) در هر لایه است.

محاسبه R_1:

$$R_1 = (\mathrm{w}-1) + (\mathrm{w}-1-1)(\mathrm{d}-1-1)$$

$$R_1 = (\mathrm{w}-1) + (\mathrm{w}-2)(\mathrm{d}-2)$$

$$R_1 = wd - w - 2d + 3$$

محاسبه R_2:

$$R_2 = R_1 + [w + (w-1)(d-1) - 1]$$

محاسبه R_3:

$$R_3 = R_2 + [(w+1) + (w+1-1)(d+1-1) - 1]$$

$$R_3 = 3(\mathrm{wd} - \mathrm{d} + 1)$$

د)

Masked Convolution چیست؟

Masked Convolution روشی است که در آن وزن‌های فیلتر کانولوشن به‌صورت انتخابی ماسک (Mask) می‌شوند تا دسترسی به برخی از پیکسل‌های ورودی محدود شود. این روش معمولاً در مسائل مربوط به پیش‌بینی‌های ترتیبی یا داده‌های وابسته به زمان (مانند مدل‌های autoregressive) به کار می‌رود. ماسک‌ها می‌توانند به شکلی طراحی شوند که تضمین کنند فقط اطلاعات از گذشته یا از نقاط خاص ورودی استفاده شود.

کاربردها:

1. **مدل‌های Autoregressive:**
   - در شبکه‌هایی مانند PixelCNN، از Masked Convolution برای تولید تصویر پیکسل به پیکسل استفاده می‌شود. ماسک تضمین می‌کند که پیکسل فعلی فقط به پیکسل‌های قبلی خود وابسته است.
2. **مدل‌های زمان‌سری:**
   - در تحلیل داده‌های زمانی، Masked Convolution می‌تواند تضمین کند که مقدار پیش‌بینی‌شده در زمان t تنها به داده‌های قبل از t وابسته است.
3. **پردازش زبان طبیعی (NLP):**
   - در معماری‌های مبتنی بر کانولوشن برای NLP، Masked Convolution محدودیت ترتیب زمانی را رعایت می‌کند تا پیش‌بینی فقط به کلمات قبلی متکی باشد.

محدودیت‌های Masked Convolution:

1. **کاهش کارایی محاسباتی:**
   ماسک کردن وزن‌های فیلترها ممکن است باعث افزایش پیچیدگی محاسبات و کاهش بهره‌وری GPU شود.
2. **سختی در طراحی ماسک‌ها:**
   طراحی مناسب ماسک‌ها به صورتی که محدودیت‌های مسئله را رعایت کند، می‌تواند چالش‌برانگیز باشد.
3. **کاهش اطلاعات در دسترس:**
   ماسک کردن می‌تواند اطلاعات مفید موجود در نواحی خاصی از ورودی را محدود کند، که ممکن است منجر به کاهش دقت مدل شود.

بهبود محدودیت‌های Masked Convolution با کانولوشن گسترشی:

کانولوشن گسترشی (Dilated Convolution) می‌تواند این محدودیت‌ها را کاهش دهد:

1. **افزایش محدوده دید (Receptive Field):**
   کانولوشن گسترشی بدون نیاز به ماسک کردن نقاط اضافی، می‌تواند نواحی بزرگ‌تری از ورودی را تحت پوشش قرار دهد، که به مدل اجازه می‌دهد اطلاعات بیشتری را از گذشته استخراج کند.

2. **حفظ ترتیب زمانی:**

با تنظیم صحیح پارامتر گسترش(Dilation Factor) ، می‌توان ترتیب زمانی یا ترتیبی داده‌ها را بدون نیاز به طراحی ماسک پیچیده رعایت کرد.

3. **افزایش کارایی محاسباتی:**

برخلاف Masked Convolution که مستلزم حذف وزن‌ها و نمونه‌برداری‌های غیرمؤثر است، کانولوشن گسترشی کارایی بیشتری داشته و از عملیات معمول کانولوشن بهره می‌برد.

سوال 6)

الف)

$$\delta^{(l)} = \frac{\partial E}{\partial z^{(l)}} = f'(z^{(l)})\, w^{(l+1)}\, \delta^{(l+1)}$$

(مشتق af) (گرادیان خروجی)

$$\Rightarrow \delta^{(l)} = \prod_{k=l+1}^{L} \left(w^{(k)} f'(z^{(k)}) \right) \delta^{(L)}$$

توضیح:
- $\sigma_{max}(w^{(k)})$ → بزرگترین مقدار ویژه‌ها
- af = Relu → مشتق Relu برابر ۱ یا ۰ است

$$\|\delta^{(l)}\| \le \prod_{k=l+1}^{L} \underbrace{\sigma_{max}(w^{(k)})}_{*} \|\delta^{(L)}\|$$

* اگر این مقدار کمتر از ۱ باشد با زیاد شدن تعداد لایه‌ها به صفر میل میکند که این یعنی gradient vanishing

مشکل خاص

$\sigma_{max}(w^{(k)}) = 0.9 \qquad L = 100$

$$\|\delta^{0}\| = (0.9)^{100} \|\delta^{(L)}\|$$

$$\hookrightarrow \approx 2.87 \times 10^{-5} \longrightarrow$$

اگر بزرگترین وزن به اندازه کافی باشد گرادیان حفظ میشه

ب)

$$\sigma = \sqrt{\frac{2}{\text{پارامترهای ورودی}}}$$

تعداد پارامترها = (کانال × سایز فیلتر) = $(3 \times 8 \times 8) = 192$

$$\sigma = \sqrt{\frac{2}{192}} = 0.102 \longrightarrow \mathcal{N}\left(0, \sqrt{\frac{2}{192}}\right)$$

ج) تابع Sigmoid به دلیل خروجی محدود (بین ۰ و ۱) و مشتق کوچک در نواحی اشباع مشکل vanishing gra... را ایجاد می‌کند و یادگیری شبکه‌های عمیق را مختل می‌کند. تابع ReLU با داشتن گرادیان ثابت برای مقادیر مثبت و نبود ناحیه اشباع، این مشکل را برطرف کرده و یادگیری کارآمدی را در شبکه‌های عمیق ممکن می‌سازد

۱) مشکل Dead neurons زمانی رخ میدهد که ورودی یک نورون منفی به حدی منفی شود و وزن‌های آن تقریباً
در نتیجه نورون در کل آموزش غیرفعال باقی می‌ماند و هیچ‌گاه بروز رسانی نمی‌شود

توابع جایگزین

- Leaky ReLU → $f(x) = \begin{cases} x & x \geq 0 \\ \alpha x & x < 0 \end{cases}$ مشتق $f'(x) = \begin{cases} 1 & x > 0 \\ \alpha & x \leq 0 \end{cases}$

اجازه می‌دهد گره‌ها یان در نواحی منفی نیز منفی، صفر نشوند. بنابراین از مشکل dead neurons جلوگیری می‌کند
اسا ب α چالش دارد ممکن است بر عملکرد مدل تاثیر بگذارد

- Parametric ReLU → شبیه قبلی است
صفر α در اینجا قابل یادگیری است

انعطاف پذیری بیشتری نسبت به Leaky ReLU و توانایی تطبیق مقدار α با داده‌ها

- ELU (Exponential Linear Unit)

→ $f(x) = \begin{cases} x & x > 0 \\ \alpha(e^x - 1) & x \leq 0 \end{cases}$ مشتق $f'(x) = \begin{cases} 1 & x > 0 \\ \alpha e^x & x \leq 0 \end{cases}$

کاهش میانگین خروجی لایه‌ها و بهبود سرعت همگرایی، گره‌ها یان در نواحی منفی صفر نمی‌شوند
اما پیچیدگی محاسباتی بیشتر نسبت به ReLU و Leaky ReLU دارد

$x^{(i+1)} = F(x^{(i)})$ ⟶ خروجی بدون RB $x^{(i+1)} = F(x^{(i)}) + x^{(i)}$ ⟶ خروجی با RB (5

مزایا شبکه های باقیمانده:

1- جلوگیری از vanishing gradient : زیرا احتمال رسیدن به خروجی باعث می‌شود این مشکل را نداشته باشیم

2- به دلیل نداشتن مشکل vanishing gra... احتمال این وجود دارد که شبکه های عمیق تر داشته باشیم

3- شبکه می تواند سریعتر همگرا شود

* در این شبکه ها skip connection اجازه می دهد گرادیان بدون تغییر یا تضعیف شدید به لایه های قبلی منتقل شود. این امر یادگیری شبکه های بسیار عمیق را امکان پذیر می کند