دانشکده‌ی مهندسی کامپیوتر

علیرضا میرکنی - ۴۰۱۱۰۶۶۱۷

# یادگیری ماشین

**پاییز ۱۴۰۳**

## تمرین چهارم

## پاسخ مسئله‌ی ۱.

**الف)**

- موارد خواسته شده در صورت سوال، همگی در جدول زیر قرار گرفته‌اند.

| ویژگی | شبکه تمام‌متصل | شبکه پیچشی |
|---|---|---|
| تعداد واحدهای خروجی | $I$ | $(W-K+1)\times(H-K+1)\times I$ |
| تعداد اتصالات | $H\times W\times I\times J$ | $K\times K\times J\times(W-K+1)\times(H-K+1)\times I$ |
| تعداد وزن‌های قابل یادگیری | $H\times W\times I\times J$ | $K\times K\times J\times I$ |

در ادامه توضیحات دقیق در ارتباط با نحوه محاسبه این مقادیر آورده شده است.

**۱. تعداد واحدهای خروجی:**

شبکه تمام‌متصل: در یک شبکه تمام‌متصل، تعداد واحدهای خروجی برابر با تعداد نورون‌های موجود در لایه خروجی است که با $I$ نمایش داده می‌شود. این مقدار به ابعاد ورودی ($H\times W$) وابسته نیست زیرا تمام ورودی‌ها به تمام خروجی‌ها متصل هستند.

شبکه پیچشی: در یک شبکه پیچشی، هر کرنل به صورت محلی عمل می‌کند و تعداد موقعیت‌های ممکن برای جایگذاری کرنل در ورودی به صورت زیر محاسبه می شود:

$$(W-K+1)\times(H-K+1)$$

است. بنابراین تعداد کل واحدهای خروجی با در نظر گرفتن تعداد فیلترها ($I$) برابر است با:

$$(W-K+1)\times(H-K+1)\times I$$

**۲. تعداد اتصالات:**

شبکه تمام‌متصل: در این شبکه، هر واحد ورودی ($H\times W$) به هر واحد خروجی ($I$) متصل است و این اتصالات برای تمام کانال‌های ورودی ($J$) برقرار است. بنابراین تعداد کل اتصالات برابر است با:

$$H\times W\times I\times J$$

شبکه پیچشی: در شبکه پیچشی، هر کرنل ($K \times K$) در هر موقعیت ممکن از ورودی اعمال می‌شود. تعداد این موقعیت‌ها برابر با تعداد واحدهای خروجی است:

$$(W - K + ۱) \times (H - K + ۱)$$

هر کرنل برای تمام کانال‌های ورودی ($J$) اعمال می‌شود و خروجی هر کرنل به تعداد فیلترها ($I$) افزوده می‌شود. بنابراین تعداد کل اتصالات برابر است با:

$$K \times K \times J \times (W - K + ۱) \times (H - K + ۱) \times I$$

**۳. تعداد وزن‌های قابل یادگیری:**

شبکه تمام‌متصل: برای هر اتصال بین واحدهای ورودی و خروجی، یک وزن مستقل وجود دارد. چون تعداد کل اتصالات برابر با $H \times W \times I \times J$ است، تعداد وزن‌های قابل یادگیری نیز دقیقاً برابر با تعداد اتصالات خواهد بود:

$$H \times W \times I \times J$$

شبکه پیچشی: در شبکه پیچشی، هر کرنل ($K \times K$) برای تمام کانال‌های ورودی ($J$) و فیلترهای خروجی ($I$) مشترک است. بنابراین تعداد کل وزن‌های قابل یادگیری برابر است با:

$$K \times K \times J \times I$$

- به طور کلی، شبکه‌های عمیق پیچشی نسبت به شبکه‌های کاملاً متصل نیازمند داده آموزشی کمتری برای آموزش هستند. دلیل این امر آن است که شبکه‌های پیچشی از اشتراک وزن‌ها و کاهش تعداد پارامترهای قابل یادگیری استفاده می‌کنند. این ویژگی‌ها باعث می‌شود که شبکه‌های پیچشی بهره‌وری بالاتری در استفاده از داده آموزشی داشته باشند و نیاز به داده کمتر برای دستیابی به همان سطح از دقت داشته باشند.

  با این حال، این نتیجه‌گیری لزوماً همیشه صحیح نیست و به عوامل متعددی بستگی دارد. به عنوان مثال، اگر داده ورودی پیچیدگی بالایی داشته باشد و یا ارتباطات غیرمحلی مهم باشند، شبکه‌های کاملاً متصل ممکن است عملکرد بهتری داشته باشند. علاوه بر این، معماری شبکه، اندازه داده آموزشی، و نحوه پردازش داده‌ها نیز می‌توانند تأثیر قابل توجهی بر نیاز به داده آموزشی داشته باشند

**ب)**

- **شبکه‌های تمام‌متصل:** در شبکه‌های تمام‌متصل، ورودی‌ها به صورت بردارهای یک‌بعدی و مسطح شده ارائه می‌شوند. در این ساختار، نرمال‌سازی دسته‌ای (Batch Normalization) به طور مستقل برای هر ویژگی اعمال می‌شود، به طوری که میانگین مقادیر ورودی به هر نورون صفر و واریانس آن‌ها یک می‌شود. این امر باعث می‌شود که تغییرات داده‌های ورودی در طول آموزش کاهش یافته و فرآیند یادگیری تسریع شود.

  همچنین، به دلیل اینکه وابستگی آماری بین ویژگی‌ها در شبکه‌های تمام‌متصل کمتر است، نرمال‌سازی به صورت مستقل انجام می‌شود. این عملیات موجب پایداری گرادیان‌ها در طول یادگیری شده و به مدل کمک می‌کند تا با نرخ یادگیری بالاتر و پارامترهای اولیه کمتر حساس، به سرعت بهینه شود.

**شبکه‌های پیچشی:** در شبکه‌های پیچشی، ورودی‌ها معمولاً داده‌های دوبعدی یا تصاویر هستند که دارای ویژگی‌های محلی مهم می‌باشند. نرمال‌سازی دسته‌ای در این شبکه‌ها به گونه‌ای طراحی شده که تمامی مقادیر یک نقشه ویژگی (Feature Map) به صورت مشابه نرمال‌سازی شوند. این کار برای حفظ همبستگی مکانی داده‌ها ضروری است. به همین دلیل، نرمال‌سازی برای هر کانال به صورت مستقل و برای تمامی مکان‌های فضایی به طور مشترک اعمال می‌شود.

نرمال‌سازی دسته‌ای در شبکه‌های پیچشی، پایداری گرادیان‌ها را افزایش داده و یادگیری را کارآمدتر می‌کند. این ویژگی به خصوص در شبکه‌های عمیق‌تر اهمیت بیشتری دارد زیرا وابستگی محلی داده‌ها حفظ می‌شود و اختلالی در این ارتباطات ایجاد نمی‌گردد.

- در لایه‌های پیچشی، حذف بایاس مشکلی در عملکرد شبکه ایجاد نمی‌کند. هنگام استفاده از نرمال‌سازی دسته‌ای، خروجی لایه پیچشی ابتدا نرمال می‌شود و سپس دو پارامتر قابل یادگیری $\beta$ و $\gamma$ به خروجی اضافه می‌شوند. در این فرآیند، نقش $\beta$ معادل بایاس است و بایاس قبلی نادیده گرفته می‌شود. از این رو، می‌توان بایاس را بدون تأثیر منفی حذف کرد.

- اگر وزن‌های لایه‌ها ($W$) یا ورودی‌ها در عددی ثابت مانند $\alpha$ ضرب شوند، این عمل تنها مقادیر اولیه خروجی را تغییر می‌دهد. نرمال‌سازی دسته‌ای با ثابت نگه داشتن میانگین و واریانس مقادیر ورودی، این تغییر را خنثی می‌کند. بنابراین، عملکرد شبکه در طول آزمایش تغییر نخواهد کرد، زیرا لایه نرمال‌سازی این مقیاس را به صورت خودکار تنظیم می‌کند.

- یکی از سناریوهایی که نرمال‌سازی دسته‌ای ممکن است عملکرد ضعیف‌تری داشته باشد، زمانی است که توزیع داده‌های ورودی به مرور زمان تغییر می‌کند. به عنوان مثال، اگر یک شبکه برای شناسایی اشیاء در شرایط روز و شب آموزش دیده باشد و آماره‌های نرمال‌سازی دسته‌ای بر اساس داده‌های روز محاسبه شده باشند، ممکن است در زمان آزمایش روی داده‌های شب، ویژگی‌های مهم تصویر مختوش شوند و دقت کاهش یابد. در چنین مواردی، روش‌های جایگزینی مانند نرمال‌سازی لایه‌ای (Layer Normalization) که به جای مینی‌بچ، نرمال‌سازی را برای هر لایه و نمونه به صورت مستقل انجام می‌دهد، می‌تواند مناسب‌تر باشد. همچنین، روش های دیگری از قبیل نرمال‌سازی گروهی (Group Normalization) یا نرمال‌سازی وزن‌ها (Weight Normalization) می‌توانند وابستگی به اندازه دسته را کاهش دهند و عملکرد پایدارتری در شرایط مختلف ارائه دهند.

**ج)**

- ابعاد تصویر ورودی ۲۵۶ × ۲۵۶ است. در هر لایه از انکدر، ابعاد به نصف کاهش می‌یابد. بنابراین، ابعاد در هر لایه به صورت زیر خواهد بود:
  - لایه اول: ۲۵۶ × ۲۵۶.
  - لایه دوم: ۱۲۸ × ۱۲۸.
  - لایه سوم: ۶۴ × ۶۴.
  - لایه چهارم: ۳۲ × ۳۲.
  - لایه پنجم (عمیق‌ترین لایه): ۱۶ × ۱۶.

بنابراین در پایین‌ترین لایه (عمیق‌ترین لایه)، فضای ویژگی ما ابعاد ۱۶ × ۱۶ خواهد داشت.

- تعداد فیلترهای انکدر به صورت ۶۴, ۱۲۸, ۲۵۶, ۵۱۲, ۵۱۲ است. برای لایه کانولوشن دوم با ۳ × ۳ کرنل‌ها:

$$\text{تعداد پارامترها} = (\text{ورودی} \times ۳ \times ۳) \times \text{خروجی} + \text{بایاس}$$

برای لایه کانولوشن دوم انکدر:

$$\text{تعداد پارامترها} = (۳ \times ۳ \times ۶۴) \times ۱۲۸ + ۱۲۸ = ۷۳۸۵۶$$

بنابراین تعداد پارامترهای این لایه ۷۳۸۵۶ است.

**د) شبکه‌های Up-Convolutional:** این شبکه‌ها که گاهی به نام Transposed Convolution شناخته می‌شوند، ابزارهایی برای افزایش ابعاد فضایی نقشه‌های ویژگی هستند. در مقاله‌ی "Inverting Visual Representations with Convolutional Networks"، این شبکه‌ها برای بازسازی تصاویر از نمایش‌های ویژگی (مانند HOG، SIFT، و خروجی لایه‌های مختلف AlexNet) به کار گرفته شده‌اند. در این روش، یک شبکه Up-Convolutional با یادگیری معکوس نگاشت ویژگی‌ها، تصویر اولیه را تخمین می‌زند. این بازسازی‌ها نشان می‌دهد که حتی در لایه‌های عمیق شبکه:

- رنگ‌ها و کانتورهای کلی اشیاء حفظ می‌شوند.

- اطلاعات محلی دقیق‌تر با پیشرفت به لایه‌های بالاتر شبکه کاهش می‌یابد.

- ویژگی‌های سطح بالا مانند دسته‌بندی‌های مفهومی (مانند "Apple" یا "Tree") به صورت کلی حفظ می‌شوند.

این بازسازی‌ها کمک می‌کنند تا درک شود که چگونه شبکه‌های پیچشی اطلاعات را حفظ یا حذف می‌کنند. برای مثال، بازسازی از لایه‌های بالاتر مانند FC8 در AlexNet نشان می‌دهد که حتی اطلاعاتی مانند رنگ‌ها از مقادیر پیش‌بینی کلاس‌ها نیز قابل استخراج هستند.

**شبکه‌های De-Convolutional:** در مقاله‌ی "Striving for Simplicity: The All Convolutional Net"، از شبکه‌های De-Convolutional به‌عنوان ابزاری برای تحلیل ویژگی‌های یادگرفته‌شده استفاده شده است. این شبکه‌ها با معکوس کردن جریان داده، نواحی تصویر که به‌طور خاص باعث فعال‌سازی نورون‌های خاص در لایه‌های بالاتر می‌شوند را مشخص می‌کنند. فرآیند این شبکه‌ها شامل موارد زیر است:

۱. شروع از یک فعال‌سازی خاص در لایه بالا و صفر کردن سایر مقادیر.

۲. بازگرداندن این فعال‌سازی‌ها به ورودی تصویر، به طوری که نواحی تصویر که بیشترین تأثیر را روی آن نورون دارند، بازسازی شوند.

۳. استفاده از روش‌های مختلف برای بازگشت از لایه‌های ReLU و پیچشی مانند Guided Backpropagation که نویز کمتری تولید می‌کند و مناطق مؤثرتر را برجسته می‌سازد.

نتیجه این فرآیند نشان می‌دهد که:

- در لایه‌های پایین‌تر، نواحی فعال‌سازی بسیار محلی و مرتبط با الگوهای ساده (مانند لبه‌ها) هستند.

- در لایه‌های بالاتر، نواحی فعال‌سازی به الگوهای پیچیده‌تر و انتزاعی‌تر (مانند اشیاء کامل یا دسته‌های مفهومی) گسترش می‌یابند.

**چگونگی کمک به تفسیرپذیری:** هر دو روش به صورت مکمل به تفسیرپذیری شبکه کمک می‌کنند:

- **شبکه‌های Up-Convolutional:** این شبکه‌ها بازسازی تصویر اولیه را از نمایش ویژگی ممکن می‌کنند. با این بازسازی‌ها می‌توان فهمید که هر لایه از شبکه چه اطلاعاتی را حفظ می‌کند. مثلاً حفظ رنگ و کانتورها نشان‌دهنده اهمیت این اطلاعات در فرآیند تصمیم‌گیری شبکه است.

- **شبکه‌های De-Convolutional:** این شبکه‌ها با مشخص کردن نواحی حساس به هر نورون، کمک می‌کنند تا نقش هر ویژگی در پیش‌بینی‌های شبکه درک شود. به‌طور خاص، این روش نشان می‌دهد که چه بخش‌هایی از تصویر بیشترین تأثیر را در فعال‌سازی لایه‌های بالاتر دارند.

به طور کلی، شبکه‌های Up-Convolutional برای بازسازی تصویر کلی و نمایش اطلاعات موجود در ویژگی‌ها مناسب هستند، در حالی که شبکه‌های De-Convolutional روی تحلیل مناطق بحرانی تمرکز دارند که به تصمیم‌گیری شبکه کمک می‌کنند.

## پاسخ مسئله‌ی ۲.

دقت کنید که می توان نوشت:

$$\begin{aligned}\frac{\partial L}{\partial W_j} &= \sum_{i=0}^{N-K} \frac{\partial L}{\partial Z_i} \times \frac{\partial Z_i}{\partial W_j} \\ &= \sum_{i=0}^{N-K} \frac{\partial L}{\partial Z_i} \times \frac{\partial \left( \sum_{k=0}^{K-1} W_k X_{i+k} \right)}{\partial W_j} \\ &= \sum_{i=0}^{N-K} \frac{\partial L}{\partial Z_i} X_{j+i}\end{aligned}$$

بنابراین با توجه به عبارت به دست آمده، می توان نتیجه گرفت که $\frac{\partial L}{\partial W_j}$ در واقع برابر مولفه $j$ ام بردار حاصل از اعمال فیلتر کانولوشنی $F = \begin{bmatrix} \frac{\partial L}{\partial Z_1} & \dots & \frac{\partial L}{\partial Z_{N-K}} \end{bmatrix}$ بر روی بردار $X$ می باشد. به عبارت دیگر طبق به تعریف بیان شده در صورت سوال، می توان نوشت:

$$\frac{\partial L}{\partial W} = F * X \iff \frac{\partial L}{\partial W_j} = \sum_{i=0}^{N-K} F_i X_{j+i}$$

## پاسخ مسئله‌ی ۳.

در ابتدا با توجه به معماری شبکه عصبی معرفی شده، مقادیر موجود در لایه های مختلف را به دست می آوریم.

لایه اول:

$$P^{(۱)}_{i,j,t} = \sum_{k=۱}^{۸} W^{(۱)}_{۱,۱,k,t} X_{i,j,k}$$

لایه دوم:

$$P^{(۲)}_{i,j} = \sum_{k=۱}^{۴} \sum_{w=۰}^{۱} \sum_{h=۰}^{۱} W^{(۲)}_{w+۱,h+۱,k} P^{(۱)}_{i+w,j+h,k}$$

لایه سوم:

$$P^{(۳)}_{i,j} = \frac{۱}{۴} \sum_{w=۲i-۱}^{۲i} \sum_{h=۲j-۱}^{۲j} P^{(۲)}_{w,h}$$

لایه چهارم:

$$h^{(۴)} = \begin{bmatrix} P^{(۳)}_{۱,۱} & P^{(۳)}_{۱,۲} & P^{(۳)}_{۲,۱} & P^{(۳)}_{۲,۲} \end{bmatrix}$$

لایه پنجم:

$$z = \sum_{i=۱}^{۴} W^{(fc)}_i h^{(۴)}_i$$

$$\hat{y} = \sigma(z)$$

**الف)**

۱. دقت کنید که می توان نوشت:

$$\frac{\partial L}{h^{(۴)}_k} = \frac{\partial L}{\partial z} \times \frac{\partial z}{\partial h^{(۴)}_k} = \frac{\partial L}{\partial z} W^{(fc)}_k$$

از طرفی طبق تعریف داریم:

$$P^{(۳)}_{i,j} = h^{(۴)}_{۲i+j-۲} \quad (۱ \leqslant i, j \leqslant ۲)$$

پس به دست می آید:

$$\frac{\partial h^{(۴)}_k}{P^{(۳)}_{i,j}} = \mathbb{I}(k = ۲i + j - ۲)$$

و در نتیجه خواهیم داشت:

$$\begin{aligned} \frac{\partial L}{P^{(۳)}_{i,j}} &= \sum_{k=۱}^{۴} \frac{\partial L}{\partial z} \times \frac{\partial z}{\partial h^{(۴)}_k} \times \frac{\partial h^{(۴)}_k}{P^{(۳)}_{i,j}} \\ &= \frac{\partial L}{\partial z} W^{(fc)}_{۲i+j-۲} \end{aligned}$$

۲. دقت کنید که می توان نوشت:

$$\frac{\partial L}{P^{(۱)}_{i,j,k}} = \sum_{m=۱}^{۴}\sum_{n=۱}^{۴} \frac{\partial L}{P^{(۲)}_{m,n}} \times \frac{\partial P^{(۲)}_{m,n}}{P^{(۱)}_{i,j,k}}$$

$$\frac{\partial L}{P^{(۲)}_{m,n}} = \sum_{p=۱}^{۲}\sum_{q=۱}^{۲} \frac{\partial L}{P^{(۳)}_{p,q}} \times \frac{\partial P^{(۳)}_{p,q}}{P^{(۲)}_{m,n}}$$

با توجه به آنچه که در ابتدا به دست آوردیم، داریم:

$$\frac{\partial P^{(۳)}_{p,q}}{P^{(۲)}_{m,n}} = \frac{۱}{۴}\mathbb{I}\left((m = ۲p \ \vee \ m = ۲p-۱) \ \wedge \ (n = ۲q \ \vee n = ۲q-۱)\right)$$

$$\frac{\partial P^{(۲)}_{m,n}}{P^{(۱)}_{i,j,k}} = W^{(۲)}_{m-i,n-j,k}\mathbb{I}\left((m = i \ \vee \ m = i+۱) \ \wedge \ (n = j \ \vee n = j+۱)\right)$$

همچنین طبق قسمت قبل می‌دانیم:

$$\frac{\partial L}{P^{(۳)}_{p,q}} = \frac{\partial L}{\partial z} W^{(fc)}_{۲p+q-۲}$$

بنابراین جواب نهایی به شکل زیر به دست می‌آید:

$$\begin{aligned}\frac{\partial L}{P^{(۱)}_{i,j,k}} &= \sum_{m=i}^{i+۱}\sum_{n=j}^{j+۱} \frac{\partial L}{P^{(۲)}_{m,n}} W^{(۲)}_{m-i,n-j,k} \\ &= \frac{۱}{۴}\sum_{m=i}^{i+۱}\sum_{n=j}^{j+۱} \frac{\partial L}{\partial z} W^{(fc)}_{۲\lceil \frac{m}{۲}\rceil + \lceil \frac{n}{۲}\rceil - ۲} W^{(۲)}_{m-i,n-j,k}\end{aligned}$$

**ب)** دقت کنید که می توان نوشت:

$$\frac{\partial L}{\partial W^{(۱)}_{۱,۱,k}} = \frac{\partial L}{P^{(۱)}_{i,j,k}} \times \frac{\partial P^{(۱)}_{i,j,k}}{W^{(۱)}_{۱,۱,k}} = \frac{\partial L}{P^{(۱)}_{i,j,k}} X_{i,j,k}$$

حال با جایگذاری $\frac{\partial L}{P^{(۱)}_{i,j,k}}$ به دست آمده از قسمت قبل، نتیجه می شود:

$$\frac{\partial L}{\partial W^{(۱)}_{۱,۱,k}} = \frac{۱}{۴}\sum_{m=i}^{i+۱}\sum_{n=j}^{j+۱} \frac{\partial L}{\partial z} W^{(fc)}_{۲\lceil \frac{m}{۲}\rceil + \lceil \frac{n}{۲}\rceil - ۲} W^{(۲)}_{m-i,n-j,k} X_{i,j,k}$$

**ج)** دقت کنید که می توان نوشت:

$$\frac{\partial L}{\partial W^{(۲)}_{i,j,k}} = \sum_{m=۱}^{۴}\sum_{n=۱}^{۴} \frac{\partial L}{P^{(۲)}_{m,n}} \times \frac{\partial P^{(۲)}_{m,n}}{\partial W^{(۲)}_{i,j,k}}$$

با توجه به آنچه که در ابتدا به دست آوردیم، داریم:

$$P^{(۲)}_{m,n} = \sum_{k=۱}^{۴}\sum_{w=۰}^{۱}\sum_{h=۰}^{۱} W^{(۲)}_{w+۱,h+۱,k} P^{(۱)}_{m+w,n+h,k}$$

بنابراین اگر $m+i$ و $n+j$ در محدوده لایه دوم قرار بگیرند، خواهیم داشت:

$$\frac{\partial P^{(۲)}_{m,n}}{\partial W^{(۲)}_{i,j,k}} = P^{(۱)}_{m+i,n+j,k}$$

و در غیر این صورت داریم:

$$\frac{\partial P^{(۲)}_{m,n}}{\partial W^{(۲)}_{i,j,k}} = ۰$$

حال با جایگذاری $\frac{\partial P^{(۲)}_{m,n}}{\partial W^{(۲)}_{i,j,k}}$ و $\frac{\partial L}{P^{(۲)}_{m,n}}$ مشتق خواسته شده به شکل زیر محاسبه می شود:

$$\frac{\partial L}{\partial W^{(۲)}_{i,j,k}} = \frac{۱}{۴}\sum_{m=۱}^{۴}\sum_{n=۱}^{۴} \frac{\partial L}{\partial z} W^{(fc)}_{۲\lceil \frac{m}{۲} \rceil + \lceil \frac{n}{۲} \rceil - ۲} P^{(۱)}_{m+i,n+j,k}$$

## پاسخ مسئله‌ی ۴.

**الف)**

- تفاوت اصلی دو مدل ذکر شده را می توان در نحوه ارتباط لایه های مختلف شبکه، به شکل زیر بیان نمود:

- **Dense Connections در DenseNet:** در DenseNet، هر لایه به تمام لایه‌های قبلی متصل است و ویژگی‌ها از طریق **الحاق (Concatenation)** منتقل می‌شوند. این امر باعث استفاده مجدد از ویژگی‌ها و افزایش بازدهی مدل می‌شود.
- **Residual Connections در ResNet:** در ResNet، اتصال بین لایه‌ها از طریق **جمع (Summation)** ویژگی‌ها انجام می‌شود. این روش گرادیان‌ها را مستقیماً به لایه‌های قبلی بازمی‌گرداند اما در مقایسه با DenseNet ویژگی‌های کمتری را در طول شبکه حفظ می‌کند.

| ResNet | DenseNet |
|---|---|
| خروجی هر لایه با خروجی لایه قبلی جمع می‌شود. | خروجی هر لایه به تمام لایه‌های بعدی الحاق می‌شود. |
| گرادیان از طریق مسیرهای کوتاه (Shortcut Connections) به لایه‌های اولیه بازمی‌گردد. | گرادیان مستقیماً از لایه خروجی به تمام لایه‌ها منتقل می‌شود. |
| پارامترهای بیشتری به دلیل جمع ویژگی‌ها در هر لایه نیاز دارد. | پارامترهای کمتری نیاز دارد زیرا ویژگی‌های تکراری مجدداً یاد گرفته نمی‌شوند. |
| ممکن است با افزایش عمق، برخی لایه‌ها اطلاعات کمتری بیاموزند. | هر لایه از تمام ویژگی‌های قبلی بهره می‌برد که به یادگیری بهتر کمک می‌کند. |
| احتمال بیشتری برای vanishing gradient دارد (هرچند کاهش یافته است). | به طور موثرتری از مشکل vanishing gradient جلوگیری می‌کند. |

جدول ۱: مقایسه بین ResNet و DenseNet

- در DenseNet، هر لایه به طور مستقیم به تمام لایه‌های قبلی متصل است. این ارتباط مستقیم باعث می‌شود جریان گرادیان از لایه‌های ابتدایی به لایه‌های پایانی و بالعکس بدون هیچ ضعفی انجام شود. به همین دلیل مشکل vanishing gradient در این معماری کاهش می‌یابد. همچنین اتصال متراکم باعث بهبود انتشار اطلاعات و کاهش نیاز به تعداد زیاد پارامترها می‌شود.
- اگر تعداد کانال‌های ورودی اولیه ۳۲ باشد و بلوک دارای ۳ لایه باشد:

$$\text{تعداد کانال‌های خروجی} = \text{ورودی اولیه} + k \times \text{تعداد لایه‌ها}$$

$$\text{تعداد کانال‌های خروجی} = ۳۲ + ۲۴ \times ۳ = ۱۰۴ \text{ کانال}$$

این فرمول مطابق مقاله شما نشان‌دهنده افزایش تدریجی کانال‌ها در هر بلوک با توجه به نرخ رشد $k$ است.

**ب)**

- **حالت اول: کانولوشن عادی ($۵ \times ۵$)**
  - در این حالت، از ۳۲ فیلتر $۵ \times ۵$ استفاده می‌شود.
  - هر فیلتر با ورودی $۱۹۲ \times ۲۸ \times ۲۸$ اعمال شده و خروجی تولید می‌کند.
  - تعداد کل عملیات محاسباتی به صورت زیر محاسبه می‌شود:

$$\text{عملیات } ۱۲۰۴۲۲۴۰۰ = (۵ \times ۵) \times ۳۲ \times ۱۹۲ \times ۲۸ \times ۲۸$$

**حالت دوم: کانولوشن با کاهش ابعاد ($۱ \times ۱$)**

- در این حالت، ابتدا کانال‌های ورودی به ۱۶ کانال کاهش می‌یابد، سپس کانولوشن $۵ \times ۵$ اعمال می‌شود.
- تعداد عملیات محاسباتی در دو مرحله انجام می‌شود:
  - **مرحله اول: کاهش ابعاد با کانولوشن $۱ \times ۱$**
    تعداد عملیات:

$$\text{عملیات } ۲۴۰۷۴۲۴ = (۱ \times ۱) \times ۱۶ \times ۱۹۲ \times ۲۸ \times ۲۸$$

  - **مرحله دوم: کانولوشن $۵ \times ۵$ روی کانال‌های کاهش یافته**
    تعداد عملیات:

$$\text{عملیات } ۱۰۶۴۸۵۷۶ = (۵ \times ۵) \times ۱۶ \times ۳۲ \times ۲۸ \times ۲۸$$

  - **جمع کل:**

$$\text{عملیات } ۱۳۰۵۶۰۰۰ = ۱۰۶۴۸۵۷۶ + ۲۴۰۷۴۲۴$$

**درصد کاهش پیچیدگی:**

- کاهش پیچیدگی با استفاده از فرمول زیر محاسبه می‌شود:

$$\frac{۱۲۰۴۲۲۴۰۰ - ۱۳۰۵۶۰۰۰}{۱۲۰۴۲۲۴۰۰} \times ۱۰۰ \approx ۸۹/۱۶ \,\%$$

مشاهده می شود که پیچیدگی محاسباتی مدل حدودا ٪۹۰ کاهش پیدا کرده است که به شدت قابل توجه می باشد.

- محاسبه تعداد عملیات برای هر نوع فیلتر در شبکه به صورت زیر انجام می‌شود:

$$۲۸ \times ۲۸ \times ۱۹۲ \times ۶۴ \times (۱ \times ۱) = ۹۶۲۵۶۰۰$$

برای فیلتر $۱ \times ۱$ با ۶۴ کانال.

$$۲۸ \times ۲۸ \times ۱۹۲ \times ۹۶ \times (۱ \times ۱) = ۱۴۴۳۸۴۰۰$$

برای فیلتر $۱ \times ۱$ با ۹۶ کانال.

$$۲۸ \times ۲۸ \times ۱۹۲ \times ۱۶ \times (۱ \times ۱) = ۲۴۰۷۴۲۴$$

برای فیلتر $۱ \times ۱$ با ۱۶ کانال.

$$۲۸ \times ۲۸ \times ۱۹۲ \times (۳ \times ۳) = ۱۳۵۴۷۵۲$$

برای Max Pooling $۳ \times ۳$.

$$۲۸ \times ۲۸ \times ۹۶ \times ۱۲۸ \times (۳ \times ۳) = ۲۷۷۰۴۵۷۶$$

برای فیلتر $۳ \times ۳$ با ۱۲۸ کانال.

$$۲۸ \times ۲۸ \times ۱۶ \times ۳۲ \times (۵ \times ۵) = ۱۰۶۴۸۵۷۶$$

برای فیلتر $۵ \times ۵$ با ۳۲ کانال.

$$۲۸ \times ۲۸ \times ۱۹۲ \times ۳۲ \times (۱ \times ۱) = ۵۹۴۹۴۴۰$$

برای فیلتر $۱ \times ۱$ با ۳۲ کانال.

**جمع کل عملیات ها:**

$$۹۶۲۵۶۰۰ + ۱۴۴۳۸۴۰۰ + ۲۴۰۷۴۲۴ + ۱۳۵۴۷۵۲ + ۲۷۷۰۴۵۷۶ + ۱۰۶۴۸۵۷۶ + ۵۹۴۹۴۴۰ = ۷۲۱۲۸۳۶۸ \text{ عملیات}$$

**دلایل استفاده از فیلترها با اندازه های مختلف :**

- **پردازش چند مقیاسی:** فیلترهای $۳ \times ۳$ و $۵ \times ۵$ ویژگی‌های بزرگ‌تر و پیچیده‌تر را استخراج می‌کنند، در حالی که فیلتر $۱ \times ۱$ ویژگی‌های محلی را بررسی می‌کند و تعداد کانال‌ها را کاهش می‌دهد.
- **کاهش ابعاد:** کانولوشن $۱ \times ۱$ تعداد کانال‌های ورودی را کاهش داده و پیچیدگی محاسباتی فیلترهای بزرگ‌تر را بهینه می‌کند.
- **ترکیب اطلاعات:** مسیرهای موازی اطلاعات متنوع را استخراج کرده و در خروجی ترکیب می‌کنند.

- معماری GoogLeNet با معرفی طبقه‌بندهای کمکی (Auxiliary Classifiers) تلاش می‌کند تا مشکل vanishing gradient را کاهش داده و پایداری بهینه‌سازی را بهبود بخشد. این طبقه‌بندها در لایه‌های میانی شبکه، مانند Inception (4a) و Inception (4d)، قرار گرفته‌اند. وظیفه آن‌ها کمک به انتشار گرادیان در طول شبکه به سمت لایه‌های پایین‌تر است. این ویژگی باعث می‌شود که لایه‌های اولیه شبکه، که گرادیان ضعیف‌تری دریافت می‌کنند، در فرآیند آموزش تأثیر بیشتری بپذیرند.

طبقه‌بندهای کمکی در مرحله آموزش به عنوان خروجی‌های موقت عمل می‌کنند و از طریق اضافه کردن ضرر (Loss) مرتبط با آن‌ها به ضرر کل شبکه، به یادگیری بهتر شبکه کمک می‌کنند. به این ترتیب، این طبقه‌بندها به عنوان یک نوع regularization عمل می‌کنند و از بیش‌برازش (Overfitting) جلوگیری می‌کنند. اما در زمان تست، این طبقه‌بندها حذف می‌شوند تا عملکرد شبکه تنها بر اساس خروجی نهایی سنجیده شود.

این رویکرد علاوه بر کاهش vanishing gradient، به بهبود جریان گرادیان در شبکه کمک می‌کند و پایداری بهینه‌سازی را افزایش می‌دهد. با توجه به عمیق بودن معماری GoogLeNet، این ویژگی باعث می‌شود که مدل بتواند اطلاعات مفیدتری از داده‌های آموزشی استخراج کند و ویژگی‌های پیچیده‌تر را با دقت بالاتری یاد بگیرد.

با این حال، محدودیت‌های این طراحی نیز قابل توجه است. تنظیم دقیق ضرر طبقه‌بندهای کمکی و وزن‌دهی آن‌ها در یادگیری شبکه نیازمند تلاش زیادی است. همچنین، در معماری‌های بسیار عمیق‌تر یا شبکه‌های مدرن‌تر، ممکن است این روش کارایی مشابهی نداشته باشد. به همین دلیل، استفاده از این تکنیک در برخی کاربردهای جدیدتر یادگیری عمیق محدود شده است.

ج)

- شبکه‌های کانولوشن معمولی از یک ساختار شبکه نمونه‌گیری ثابت استفاده می‌کنند. به‌عنوان مثال، در یک هسته ۳ × ۳، نقاط نمونه‌گیری همیشه در موقعیت‌های از پیش تعریف‌شده قرار دارند. این ساختار ثابت باعث می‌شود که این شبکه‌ها در مواجهه با تغییرات هندسی انعطاف‌پذیری محدودی داشته باشند.

در مقابل، شبکه‌های Deformable با اضافه کردن آفست‌های قابل یادگیری به نقاط شبکه نمونه‌گیری، این مشکل را حل می‌کنند. این آفست‌ها، که در طی فرآیند آموزش و بر اساس ویژگی‌های محلی یاد گرفته می‌شوند، به شبکه اجازه می‌دهند تا به‌صورت پویا و محلی تغییرات هندسی را مدل‌سازی کند.

شبکه‌های Deformable قادر به مدل‌سازی تغییرات هندسی پیچیده مانند تغییر مقیاس، چرخش، و تغییر نسبت ابعاد هستند. این ویژگی به دلیل وجود آفست‌های متغیر و قابل یادگیری است که امکان نمونه‌گیری از مکان‌های غیرثابت را فراهم می‌کنند.

معادله کانولوشن معمولی:

$$y(p_\cdot) = \sum_{p_n \in R} w(p_n) \cdot x(p_\cdot + p_n)$$

معادله کانولوشن Deformable با اضافه کردن آفست‌ها:

$$y(p_\cdot) = \sum_{p_n \in R} w(p_n) \cdot x(p_\cdot + p_n + \Delta p_n)$$

در اینجا $\Delta p_n$ آفست‌هایی هستند که در طی فرآیند یادگیری تعیین می‌شوند.

آفست‌ها به کمک یک لایه کانولوشن اضافی محاسبه می‌شوند که ورودی آن همان ویژگی‌های نقشه ورودی است. خروجی این لایه، یک میدان آفست است که در همان وضوح نقشه ورودی قرار دارد. این آفست‌ها برای هر مکان به‌صورت محلی و مستقل یاد گرفته می‌شوند.

برای مکان‌های کسری که به دلیل آفست ایجاد می‌شوند، از درونیابی دوجمله‌ای استفاده می‌شود:

$$x(p) = \sum_{q} G(q,p) \cdot x(q)$$

که در آن $G(q,p)$ کرنل درونیابی دوجمله‌ای است و $p$ یک مکان کسری است.

**مزایای شبکه‌های Deformable:**

- **انعطاف‌پذیری:** این شبکه‌ها می‌توانند تغییرات هندسی پیچیده را به‌صورت پویا مدل‌سازی کنند.
- **کارایی بالا:** به‌دلیل معماری سبک و ساده، اضافه کردن این ماژول‌ها سربار محاسباتی کمی دارد.
- **بهبود عملکرد:** این شبکه‌ها در وظایف پیچیده‌ای مانند تشخیص اشیاء و بخش‌بندی معنایی عملکرد بهتری دارند.

## پاسخ مسئله‌ی ۵.

**الف)** طبق فرض صورت سوال، کرنل اولیه دارای ابعاد $F \times F$ می باشد. در هر ردیف و ستون این کرنل، تعداد فضاهای خالی برابر با $F - ۱$ است و هر فضای خالی اندازه $D - ۱$ را اشغال می‌کند. بنابراین، ابعاد کرنل پس از اعمال گسترش به صورت زیر محاسبه می‌شود:

$$\text{New Kernel Dimensions} = F + (F - ۱)(D - ۱) = FD - D + ۱$$

این نشان می‌دهد که کرنل جدید دارای ابعاد $(FD - D + ۱) \times (FD - D + ۱)$ خواهد بود.

برای محاسبه ابعاد خروجی کانولوشن، از فرمول زیر استفاده می‌کنیم:

$$\text{Output Dimensions} = (N - FD + D) \times (M - FD + D)$$

که در آن:

$$N' = N - (FD - D + ۱) + ۱ = N - FD + D$$

$$M' = M - (FD - D + ۱) + ۱ = M - FD + D$$

بنابراین، اگر ماتریس ورودی دارای ابعاد $N \times M$ باشد، ابعاد خروجی پس از اعمال کرنل گسترش‌یافته به صورت $(N - FD + D) \times (M - FD + D)$ خواهد بود.

**ب)** مطابق صورت سوال، تعریف ریاضی کانولوشن گسترش‌یافته به شکل زیر است:

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

که در اینجا $D$ نشان‌دهنده گام گسترش و $I$ ورودی تصویر می‌باشد. همچنین فرض می کنیم که شماره گذاری سطر ها و ستون های ماتریس از صفر شروع می شوند.

ضرب کرونکر بین دو ماتریس $M$ و $N$ به شکل زیر تعریف می‌شود:

$$(M \otimes N)_{ij} = M_{ij} \cdot N$$

این بدان معناست که هر عنصر $M_{ij}$ ماتریس $M$ در کل ماتریس $N$ ضرب می‌شود و نتیجه حاصل یک ماتریس بزرگ‌تر خواهد بود. برای تعریف کانولوشن گسترش یافته با استفاده از ضرب کرونکر، ابتدا ماتریس $A_{D \times D}$ را به صورت زیر تعریف می‌کنیم:

$$A_{i,j} = \begin{cases} ۱ & i = j = ۰ \\ ۰ & \text{otherwise} \end{cases}$$

سپس کرنل گسترش‌یافته $K'$ را به صورت زیر محاسبه می‌کنیم:

$$K' = K \otimes A$$

بنابراین، عناصر $K'$ به صورت زیر خواهند بود:

$$K'_{i,j} = \begin{cases} K_{\frac{i}{D},\frac{j}{D}} & \frac{i}{D}, \frac{j}{D} \in \mathbb{Z} \\ ۰ & \text{otherwise} \end{cases}$$

خروجی کانولوشن با کرنل گسترش‌یافته $K'$ به صورت زیر محاسبه می‌شود:

$$\begin{aligned}(K' * I)(i,j) &= \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} K'(m,n)\cdot I(i+m, j+n) \\ &= \sum_{p=-\infty}^{\infty}\sum_{q=-\infty}^{\infty} K(p,q)\cdot I(i+pD, j+qD) \\ &= \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} K(m,n)\cdot I(i+Dm, j+Dn) = (K *_D I)(i,j)\end{aligned}$$

که این همان خواسته مسئله است.

**ج)** در ابتدا متغیر $r_l$ را برابر تعداد پیکسل های قابل مشاهده توسط عنصر $(i,j)$ خروجی در لایه $l$ ام شبکه قرار دهید. دقت کنید که به وضوح و با توجه به تعریف، $r_۳ = ۱$ می باشد. در این قسمت، هدف محاسبه $r_۰$ (لایه صفرم همان لایه ورودی شبکه است) می باشد.

حال به این نکته توجه کنید که طبق نتیجه به دست آمده در قسمت قبل، اعمال کانولوشن گسترش‌یافته با فیلتر $K_{F\times F}$ و گام $D$، معادل اعمال کانولوشن عادی با فیلتر $K'_{(FD-D+۱)\times(FD-D+۱)}$ است. به همین دلیل، از اینجا به بعد فرض می کنیم که تمامی عملیات های کانولوشن موجود در شبکه عادی هستند و با این فرض مسئله را حل می کنیم.

به وضوح رابطه بازگشتی زیر برای $r_l$ها برقرار است:

$$r_l = r_{l+۱} + (F_l D_l - D_l + ۱) - ۱ = r_{l+۱} + D_l(F_l - ۱)$$

چرا که با اعمال کانولوشن و عبور از لایه $l$ به لایه $l+۱$، تعداد پیکسل های قابل مشاهده توسط عنصر خروجی به میزان "بعد فیلتر منهای یک" عدد افزایش می یابد (منهای یک به منظور در نظر نگرفتن خود پیکسل است). حال با توجه به رابطه بازگشتی بالا و مقادیر زیر، $r_۰$ را به دست می آوریم:

$$F_۰ = w-۱, \quad F_۱ = w, \quad F_۲ = w+۱ \quad D_۰ = d-۱, \quad D_۱ = d, \quad D_۲ = d+۱$$

خواهیم داشت:

$$\begin{aligned} r_۳ = ۱ &\Longrightarrow r_۲ = ۱ + w(d+۱) \\ &\Longrightarrow r_۱ = ۱ + w(d+۱) + (w-۱)d \\ &\Longrightarrow r_۰ = ۱ + w(d+۱) + (w-۱)d + (w-۲)(d-۱)\end{aligned}$$

و این همان خواسته مسئله است.

**د)** کانولوشن ماسک‌شده (Masked Convolution) در حوزه یادگیری عمیق، به ویژه در مدل‌هایی مانند PixelCNN که برای پیش‌بینی توالی‌ها و مدل‌های خودتوضیح‌دهنده طراحی شده‌اند، کاربرد دارد. در این روش، فیلترها با استفاده از یک ماسک (Mask) محدود می‌شوند تا تنها از اطلاعات قبلی یا نقاط خاصی بهره‌برداری کنند. به عنوان مثال:

- در پردازش تصویر، کانولوشن ماسک‌شده تضمین می‌کند که مقدار هر پیکسل تنها به مقادیر پیکسل‌های قبلی وابسته باشد.
- در مسائل توالی‌ای مانند پیش‌بینی سری‌های زمانی، این تکنیک از دسترسی به داده‌های آینده جلوگیری می‌کند.

**محدودیت‌ها:**

۱. **دامنه دید محدود:** ماسک‌گذاری باعث می‌شود که هر لایه تنها به بخشی محدود از ورودی دسترسی داشته باشد، که ممکن است برای برخی کاربردها ناکافی باشد.

۲. **عمق زیاد شبکه:** برای افزایش دامنه دید، نیاز به تعداد زیادی لایه وجود دارد که این موضوع می‌تواند محاسبات را پیچیده کرده و مشکلاتی مانند ناپدید شدن گرادیان (Vanishing Gradient) را ایجاد کند.

کانولوشن گسترش‌یافته (Dilated Convolution) با افزایش فاصله بین نقاط نمونه‌برداری، دامنه دید را بدون افزایش تعداد پارامترها یا تعداد لایه‌ها گسترش می‌دهد. این تکنیک به ویژه در ترکیب با کانولوشن ماسک‌شده مزایای قابل توجهی دارد:

**مزایا:**

۱. **گسترش دامنه دید بدون افزایش پارامترها:** با استفاده از کانولوشن گسترش‌یافته، می‌توان دامنه دید را به طور قابل توجهی افزایش داد بدون نیاز به افزودن کرنل‌های بزرگ‌تر یا لایه‌های بیشتر.

۲. **حفظ ساختار محلی و جلوگیری از دسترسی غیرمجاز:** ترکیب کانولوشن ماسک‌شده با کانولوشن گسترش‌یافته، امکان حفظ دقت محلی و در عین حال گسترش دامنه دید را فراهم می‌آورد. ماسک اعمال شده اطمینان حاصل می‌کند که تنها به اطلاعات مجاز دسترسی داشته باشیم، در حالی که کانولوشن گسترش‌یافته اجازه می‌دهد تا مدل اطلاعات گسترده‌تری را در هر لایه پردازش کند.

۳. **کاهش عمق شبکه و پیچیدگی محاسباتی:** با گسترش دامنه دید در هر لایه از طریق کانولوشن گسترش‌یافته، نیاز به افزودن لایه‌های متعدد برای افزایش دامنه دید کاهش می‌یابد. این امر منجر به کاهش پیچیدگی محاسباتی و افزایش کارایی مدل می‌شود.

۴. **افزایش انعطاف‌پذیری مدل:** کانولوشن گسترش‌یافته به مدل اجازه می‌دهد تا تعادلی بهینه بین دقت محلی و اطلاعات کلی برقرار کند، که این امر برای درک بهتر ویژگی‌های مختلف تصویر ضروری است.

## پاسخ مسئله‌ی ۶.

**الف)** در ابتدا به این نکته دقت کنید که چون مشتق تابع ReLU تنها دو حالت دارد:

$$f'(z) = \begin{cases} ۱ & z > ۰ \\ ۰ & z \leqslant ۰ \end{cases}$$

هرگاه حتی در یک مرحله از لایه‌های بعد از $i$ مشتق صفر باشد، حاصل‌ضرب کل به صفر منجر می‌شود. به منظور مشاهده‌ی بیشترین مقدار ممکن برای $|\delta_i|$ در شرایط ایده‌آل فرض می‌کنیم که در تمام مراحل مشتق ReLU برابر ۱ است. در این حالت:

$$\begin{aligned} |\delta_i| = \|\delta_i\|_۲ &= \left\| \delta_L \left( \prod_{k=i+۱}^{L} W_k \right) \right\|_۲ \\ &\leqslant \|\delta_L\|_۲ \prod_{k=i+۱}^{L} \|W_k\|_۲ \\ &= |\delta_L| \prod_{k=i+۱}^{L} \sigma_{\max}(W_k) \end{aligned}$$

در نتیجه خواهیم داشت:

$$\begin{aligned} \lim_{L\to\infty} |\delta_i| &\leqslant \lim_{L\to\infty} |\delta_L| \prod_{k=i+۱}^{L} \sigma_{\max}(W_k) \\ &= |\delta_L| \lim_{L\to\infty} \prod_{k=i+۱}^{L} \sigma_{\max}(W_k) \end{aligned}$$

از آنجا که طبق فرض مسئله $\sigma_{\max}(W_k)$ (بزرگ ترین مقدار تکین ماتریس $W_k$) برای هر $k$ کوچک تر از یک است، زمانی که $L \to \infty$ عبارت $\prod_{k=i+۱}^{L} \sigma_{\max}(W_k)$ برابر حاصلضرب بی شمار عدد کوچک تر از یک می شود و به صفر میل می کند و در نتیجه خواهیم داشت:

$$\lim_{L\to\infty} \prod_{k=i+۱}^{L} \sigma_{\max}(W_k) = ۰ \implies \lim_{L\to\infty} |\delta_i| = ۰$$

با توجه به نتیجه فوق و برای مثال تا ابتدای شبکه، داریم:

$$|\delta_۰| \leqslant |\delta_L| \prod_{k=۱}^{۱۰۰} \|W_k\| \leqslant |\delta_L| (۰/۹)^{۱۰۰}$$

با توجه به اینکه $(۰/۹)^{۱۰۰} \approx ۲/۶۵۶۱ \times ۱۰^{-۵}$ (تقریباً بسیار کوچک)، می‌توان دید که $|\delta_۰|$ بسیار کوچک شده و گرادیان در لایه‌های اولیه شبکه تقریباً به صفر نزدیک می‌شود. این امر، پدیده‌ی vanishing gradient را به خوبی نشان می‌دهد.

**ب)** برای مقداردهی اولیه (Initialization) از روش Kaiming Initialization یا He Initialization استفاده می‌کنیم. در این روش، برای یک توزیع نرمال، واریانس وزن‌ها برابر است با:

$$\mathrm{Var}(w) = \frac{۲}{n_{\text{in}}}$$

که $n_{\text{in}}$ تعداد ورودی‌های موثر بر هر Kernel است. اگر فرض کنیم لایه کانولوشن ما دارای کانال ورودی ۳ و کرنل با ابعاد ۸ × ۸ باشد، آنگاه:

$$n_{\text{in}} = ۳ \times ۸ \times ۸ = ۱۹۲.$$

در نتیجه:

$$\mathrm{Var}(w) = \frac{۲}{۱۹۲} = \frac{۱}{۹۶} \approx ۰/۰۱۰۴$$

انحراف معیار $\sigma$ به صورت:

$$\sigma = \sqrt{\mathrm{Var}(w)} = \sqrt{۰/۰۱۰۴} \approx ۰/۱۰۲$$

پس کرنل‌های این لایه از توزیع نرمال:

$$w \sim \mathcal{N}(۰,\ ۰/۱۰۲^{۲})$$

نمونه‌برداری می‌شوند.

**ج)** مقایسه‌ی ReLU و Sigmoid از منظر گرادیان:

- تابع Sigmoid:

$$\mathrm{Sigmoid}(x) = \frac{۱}{۱ + e^{-x}}$$

برای ورودی‌های بزرگ مثبت، $\mathrm{Sigmoid}(x) \approx ۱$ و مشتق آن بسیار کوچک می‌شود. همچنین برای ورودی‌های بزرگ منفی $\mathrm{Sigmoid}(x) \approx ۰$ و مشتق آن باز هم ناچیز است. بنابراین گرادیان در لایه‌های ابتدایی شبکه که ورودی‌شان به نواحی اشباع منتقل می‌شود، بسیار کوچک خواهد شد:

$$\frac{d}{dx}\mathrm{Sigmoid}(x) = \mathrm{Sigmoid}(x)(۱ - \mathrm{Sigmoid}(x))$$

که در حالت اشباع به مقدار نزدیک صفر میل می‌کند.

- تابع ReLU:

$$\mathrm{ReLU}(x) = \max(۰, x)$$

در نواحی مثبت، مشتق ReLU ثابت و برابر ۱ است:

$$\frac{d}{dx}\mathrm{ReLU}(x) = \begin{cases} ۱ & x > ۰ \\ ۰ & x \leqslant ۰ \end{cases}$$

این ویژگی مانع از محوشدن گرادیان می‌شود و به یادگیری سریع‌تر و مؤثرتر شبکه‌های عمیق کمک می‌کند.

**د)** مشکل Dead Neurons (نورون‌های مرده) در ReLU وقتی رخ می‌دهد که ورودی نورون همواره منفی باشد، به طوری که خروجی ReLU آن همیشه ۰ باقی مانده و گرادیانی برای به‌روزرسانی وزن‌ها دریافت نکند. در نتیجه نورون برای همیشه غیرفعال می‌شود.

توابع Leaky ReLU، PReLU ( Parametric ReLU ) و ELU جهت رفع این مشکل معرفی شده‌اند:

- Leaky ReLU:

$$f(x) = \begin{cases} x & x > ۰ \\ \alpha x & x \leqslant ۰ \end{cases}$$

$\alpha$ یک مقدار کوچک ثابت (مثلا ۰/۰۱) است. این تابع حتی در ناحیه منفی، خروجی منفی غیرصفر تولید می‌کند و مشتق آن:

$$f'(x) = \begin{cases} ۱ & x > ۰ \\ \alpha & x \leqslant ۰ \end{cases}$$

- PReLU (Parametric ReLU):

$$f(x) = \begin{cases} x & x > ۰ \\ ax & x \leqslant ۰ \end{cases}$$

که $a$ پارامتری قابل یادگیری است و مشتق:

$$f'(x) = \begin{cases} ۱ & x > ۰ \\ a & x \leqslant ۰ \end{cases}$$

این تابع انعطاف‌پذیری بیشتری نسبت به Leaky ReLU دارد.

- ELU (Exponential Linear Unit):

$$f(x) = \begin{cases} x & x > ۰ \\ \alpha(e^x - ۱) & x \leqslant ۰ \end{cases}$$

مشتق آن:

$$f'(x) = \begin{cases} ۱ & x > ۰ \\ \alpha e^x & x \leqslant ۰ \end{cases}$$

این توابع با ایجاد خروجی‌های منفی و در نتیجه گرادیان غیرصفر در ناحیه منفی، مانع از مردن نورون‌ها می‌شوند.

البته دقت کنید که این توابع هر یک مزایا و معایبی دارند:

- Leaky ReLU: پیاده‌سازی ساده ولی $\alpha$ ثابت ممکن است بهینه نباشد.
- PReLU: انعطاف‌پذیری بالاتر با قابلیت یادگیری $\alpha$، ولی افزایش پارامترها ممکن است خطر overfitting را بالا ببرد.
- ELU: همگرایی سریع‌تر و میانگین خروجی نزدیک به صفر، اما محاسبات نسبت به ReLU پیچیده‌تر و هزینه‌برتر است.

**ه)** در شبکه‌های دارای بلاک‌های باقی‌مانده (Residual Blocks)، خروجی بلاک به صورت:

$$y_l = F(x_l) + x_l$$

تعریف می‌شود. در هنگام محاسبه گرادیان تابع هزینه $J$ نسبت به ورودی بلاک:

$$\frac{\partial J}{\partial x_l} = \frac{\partial J}{\partial y_l}(F'(x_l) + ۱)$$

وجود عبارت $(+۱)$ که از مسیر میانبر (Skip Connection) ناشی می‌شود، یک جریان گرادیان مستقیم را فراهم می‌کند. حتی اگر $F'(x_l)$ کوچک باشد، عبارت $+۱$ مانع از نابودی گرادیان می‌شود و به حفظ مقدار گرادیان در لایه‌های اولیه کمک می‌کند.

در شبکه‌های بدون Residual Blocks:

$$y_l = F(x_l) \implies \frac{\partial J}{\partial x_l} = \frac{\partial J}{\partial y_l} F'(x_l)$$

اگر $F'(x_l)$ کوچک باشد (مثلا در شبکه‌هایی که از Sigmoid استفاده می‌کنند و ورودی در ناحیه اشباع است)، گرادیان به سرعت در عمق شبکه از بین می‌رود (پدیده vanishing gradient). اما در صورت وجود Residual Blocks، با توجه به جمله $+۱$، نحوه‌ی انتقال گرادیان به گونه‌ای تغییر می‌یابد که از محو شدن زودهنگام آن جلوگیری شود و شبکه‌های بسیار عمیق بتوانند مؤثرتر آموزش ببینند.