دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

یادگیری ماشین

پاییز ۱۴۰۳

استاد: علی شریفی زارچی

مسئول تمرین: ایزدی

مهلت ارسال امتیازی: ۱۶ آذر | **تمرین چهارم** | مهلت ارسال نهایی: ۲۳ آذر

- مهلت ارسال پاسخ تا ساعت ۲۳:۵۹ روزهای مشخص‌شده است.
- در طول ترم، برای هر تمرین می‌توانید تا ۵ روز تأخیر مجاز داشته باشید و در مجموع حداکثر ۱۵ روز تأخیر مجاز خواهید داشت. توجه داشته باشید که تأخیر در تمرین‌های عملی و تئوری به صورت جداگانه محاسبه می‌شود و مجموع تأخیر هر دو نباید بیشتر از ۱۵ روز شود. پس از اتمام زمان مجاز، دو روز اضافی برای آپلود غیرمجاز در نظر گرفته شده است که در این بازه، به ازای هر ساعت تأخیر، ۲ درصد از نمره نهایی تمرین کسر خواهد شد.
- اگر بخش عملی یا تئوری تمرین را قبل از مهلت ارسال امتیازی آپلود کنید، ۲۰ درصد نمره اضافی به آن بخش تعلق خواهد گرفت و پس از آن، ویدیویی تحت عنوان راهنمایی برای حل تمرین منتشر خواهد شد.
- حتماً تمرین‌ها را بر اساس موارد ذکرشده درک‌شده در صورت سوالات حل کنید. در صورت وجود هرگونه ابهام، آن را در صفحه تمرین در سایت کوئرا مطرح کنید و به پاسخ‌هایی که از سوی دستیار آموزشی مربوطه ارائه می‌شود، توجه کنید.
- در صورت هم‌فکری و یا استفاده از منابع، نام هم‌فکران و آدرس منابع مورد استفاده برای حل سوال را ذکر کنید.
- فایل پاسخ‌های سوالات نظری را در قالب یک فایل pdf به فرمت HW4_T_[STD_ID].pdf آماده کنید و برای سوالات عملی، هر یک را در یک فایل zip جداگانه قرار دهید. فایل مربوط به نوتبوک i ام را به فرمت HW4_P[i]_[STD_ID].zip نامگذاری کرده و هرکدام را به صورت جداگانه آپلود کنید

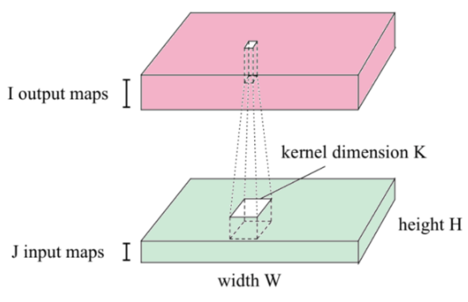
گردآورندگان تمرین: امیرحسین ایزدی، محمدحسین شالچیان، امیرعزتی، علی بختیاری

سوالات نظری (۱۰۰+۱۰ نمره)

۱. (۲۲ نمره) به سوالات زیر پاسخ دهید.

الف) دو لایه متوالی از شبکه ژرفی را در نظر بگیرید به طوری که لایه ورودی دارای J و لایه خروجی دارای I ویژگی نگاشت باشد. اتصال میان این دو لایه می‌تواند یا به صورت اتصال تمام‌متصل یا به صورت پیچشی باشد. برای هر دو حالت، ویژگی نگاشت لایه ورودی دارای ابعاد $W \times H$ است و کرنل استفاده شده در لایه پیچشی دارای ابعاد $K \times K$ است. نمای کلی این دولایه در شکل ۲ قابل مشاهده هست.

- تعداد واحدهای خروجی، اتصالات و وزن‌های قابل یادگیری را در صورت امکان محاسبه؛ گزارش و با یکدیگر مقایسه کنید.(از بایاس صرف نظر کنید)



شکل۱ : لایه ورودی با رنگ سبز و خروجی با رنگ قرمز مشخص شده است.

پاسخ:

	FC layer	CONV layer
#units output	WHI	WHI
#weights	$W^۲H^۲IJ$	$K^۲IJ$
#connections	$W^۲H^۲IJ$	$WHK^۲IJ$

(آ) از محاسبات قبلی می‌توان نتیجه گرفت که شبکه‌های عمیق کاملاً پیچشی نسبت به شبکه‌های کاملاً متصل عموماً نیازمند داده آموزشی کمتری برای آموزش هستند؟ آیا این نتیجه‌گیری لزوماً صحیح است و مرتبط است یا خیر؟

پاسخ:

- 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) شبکه CNN ای را در نظر بگیرید که از بلاک‌هایی به فرم زیر استفاده می‌کند:

$$(\text{ConvLayer}) \rightarrow (\text{BatchNorm}) \rightarrow (\text{Activation})$$

- با مطالعه این مقاله نحوه انجام نرمال سازی بچ در شبکه های تماما متصل و شبکه های پیچشی را با یکدیگر مقایسه کنید

پاسخ:

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.

- آیا حذف بایاس b از لایه کانولوشن در کارکرد این شبکه اختلالی ایجاد می‌کند؟ چرا؟

پاسخ:

- حذف بایاس در لایه‌هایی که از نرمال‌سازی دسته‌ای استفاده می‌کنند معمولاً اختلالی در شبکه ایجاد نمی‌کند. دلیل این امر این است که لایه نرمال‌سازی دسته‌ای شامل پارامترهای قابل یادگیری (مقیاس γ و شیفت β) است که می‌توانند اثر نبود بایاس را جبران کنند.
- **دلیل:** مرحله تبدیل خطی (Affine Transformation) در نرمال‌سازی دسته‌ای (پس از نرمال‌سازی) انعطاف‌پذیری لازم را برای جبران تغییرات فراهم می‌کند. بنابراین، وجود بایاس ضروری نیست.

- همچنین فرض کنید شبکه را آموزش داده‌ایم؛ آیا ضرب کردن وزن‌ها در یک عدد مانند α در زمان آزمایش (Inference)، عملکرد شبکه را تغییر می‌دهد؟ ضرب کردن این ضریب در تمام درایه‌های ورودی شبکه چطور؟

پاسخ:

تأثیر ضرب کردن وزن‌ها در حین inference:
اگر وزن‌ها را در α ضرب کنیم:

$$x_{\text{in}} \Rightarrow \text{Conv Layer} \Rightarrow \alpha x_{\text{out}} \Rightarrow \text{Batch Norm} \Rightarrow \alpha\gamma\left(\frac{\alpha x_{\text{out}} - \alpha\mu}{|\alpha|\sigma}\right) + \alpha\beta$$
$$= \alpha\left(\gamma \text{sign}\,(\alpha)\left(\frac{x_{\text{out}} - \mu}{\sigma}\right) + \beta\right) = \alpha y$$

حال اگر تابع فعال‌سازی غیرخطی باشد در این صورت مقدار $f(\alpha y)$ رابطه مشخصی با $f(y)$ ندارد، بنابراین عملکرد مدل تحت تأثیر قرار می‌گیرد و رفتار مشابه قبل نخواهد داشت.

تأثیر ضرب کردن ورودی‌ها در حین inference:
داریم:

$$\alpha x_{\text{in}} \Rightarrow \text{Conv Layer} \Rightarrow \alpha x_{\text{out}} \Rightarrow \text{Batch Norm} \Rightarrow \gamma\left(\frac{\alpha x_{\text{out}} - \alpha\mu}{|\alpha|\sigma}\right) + \beta$$
$$= \text{sign}(\alpha)\left(\gamma\left(\frac{x_{\text{out}} - \mu}{\sigma}\right) + \beta\right)$$

می‌توان دید که در صورت مثبت بودن α تغییری در عملکرد مدل ایجاد نمی‌شود.

- یک سناریو مرتبط به حوزه پردازش تصویر را شرح دهید که در آن نرمال سازی دسته ای یا بچ ممکن است در آن کارایی کمتری داشته باشد و یا حتی نتیجه در معکوس دهد.در چنین مواردی چه تکنیک های جایگزینی می توانند مورد توجه قرار گیرند؟

پاسخ:

- **سناریو:** در حوزه‌هایی مانند تقویت تصاویر گمونر، یا تحلیل تصاویر پزشکی، که شدت پیکسل‌های ورودی توزیع‌های نامتعادلی دارند، نرمال‌سازی دستگاه ممکن است باعث نتایج ضعیف شود.
  * به عنوان مثال، اگر مقادیر پیکسل‌ها توزیع به شدت نامتعادل داشته باشند، نرمال‌سازی ممکن است جزئیات ظریفی که برای تصمیم‌گیری مهم هستند را از بین ببرد.

استراتژی‌ها برای رفع این مشکلات:

- استفاده از **نرمال‌سازی نمونه‌ای** (Instance Normalization) به جای نرمال‌سازی دستگاهی در وظایف خاص (مانند تحلیل سبک).
- استفاده از **نرمال‌سازی گروهی** (Group Normalization) **یا نرمال‌سازی لایه‌ای** (Layer Normalization) که وابسته به آمار دسته نیستند.
  * تنظیم پارامترهای اولیه γ و β برای انطباق بهتر با داده‌های خاص دامنه.
  * آموزش مجدد شبکه روی داده‌هایی که توزیع مشابهی با مجموعه آزمایش دارند، برای تطبیق مناسب لایه‌های نرمال‌سازی.

ج) معماری Unet را در نظر بگیرید:

- تصور کنید که ابعاد تصویر ورودی ما برای این شبکه ۲۵۶×۲۵۶ می‌باشد. حال فرض می‌شود که در این معماری هر لایه در انکدر ابعاد را به نصف کاهش می‌دهد و در دیکدر دو برابر می‌کند. در پایین‌ترین لایه (عمیق‌ترین لایه) این معماری، فضای ویژگی ما چند پیکسل خواهد داشت؟

پاسخ:

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.

- فرض کنید انکودر دارای لایه‌هایی با ۶۴، ۱۲۸، ۲۵۶ و ۵۱۲ فیلتر است. اگر هر لایه کانولوشن از کرنل‌های ۳×۳ استفاده کند، تعداد پارامترهای لایه کانولوشن دوم انکدر را محاسبه کنید.

پاسخ:

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

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

برای لایه کانولوشن دوم:

$$۱۲۸ \times ۶۴ \times (۳ \times ۳) + ۱۲۸ = ۷۳,۸۵۶$$

بنابراین تعداد پارامترهای این لایه ۷۳,۸۵۶ است.

د) تفسیرپذیری شبکه‌های عصبی از اهمیت بالایی برخوردار است. هنگام دسته‌بندی تصاویر، علاقه‌مندیم بدانیم کدام بخش‌های تصویر در دسته‌بندی، تأثیر بیشتری داشته‌اند. در مقاله ایده‌ی شبکه‌های de-convolutional و در مقاله ایده‌ی شبکه‌های up-convolutional مطرح شده است. با بررسی این مقالات، توضیح دهید هر کدام از دو روش به چه صورت منجر به تفسیرپذیری شبکه کانولوشنی می‌شوند.

پاسخ:

شبکه‌های Up-Convolutional: این شبکه‌ها که گاهی به نام Transposed Convolution شناخته می‌شوند، ابزاری برای افزایش ابعاد فضایی نقشه‌های ویژگی هستند. در مقاله‌ی "Inverting Visual Representations"، این شبکه‌ها برای بازسازی تصاویر از نمایش‌های ویژگی (SIFT، HOG و خروجی لایه‌های مختلف AlexNet) به کار گرفته شدند. در این روش، یک شبکه Up-Convolutional با یادگیری معکوس نگاشت ویژگی‌ها، تصویر اولیه را تخمین می‌زند. این بازسازی‌ها نشان می‌دهند 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 "Tree" یا "Apple") 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 نشان می‌دهد که حتی اطلاعاتی مانند رنگ‌ها و مقادیر پیشین کلاس‌ها نیز قابل استخراج هستند.

شبکه‌های De-Convolutional: در مقاله‌ی "Striving for Simplicity: The All Convolutional Network"، 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 مراحل زیر است:

i. شروع به فعال‌سازی خاص در لایه‌ی بالا و صفر کردن سایر مقادیر.

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

iii. استفاده از روش‌های مختلف برای بازگشت از لایه‌های ReLU و پیچشی مانند Guided Backpropagation که نویز کمی 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. بازسازی‌ها می‌توانند 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 در فعال‌سازی لایه‌های بالاتر دارند.

به‌طور 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.

۲. (۸ نمره) فرض کنید برداری به طول N دارید و قصد دارید یک لایه کانولوشن یک بعدی روی آن اعمال کنید. حاصل اعمال یک لایه کانولوشن را از طریق رابطه‌ی زیر به‌دست آورید:

$$Z = W * X \quad \longrightarrow \quad Z_i = \sum_{j=0}^{K-1} W_j X_{i+j}$$

به دست می‌آوریم که K اندازه‌ی فیلتر را نشان می‌دهد. اگر مقدار $\frac{\partial L}{\partial Z_i}$ برای تمامی مقادیر i بدانیم، رابطه‌ی مربوط به $\frac{\partial L}{\partial W_j}$ را به طور دقیق پیدا کنید. نشان دهید این رابطه، عملاً معادل اعمال یک فیلتر کانولوشن است.

پاسخ:

یک مرحله forward عملیات کانولوشن ۱ بعدی را می‌توان به صورت زیر نوشت:

$$Z_i = \sum_{j=0}^{K-1} W_j X_{i+j}, \quad \text{for} i \in [0, N-K]$$

برای محاسبه گرادیان $\frac{\partial L}{\partial W_j}$، می‌توانیم قانون مشتق زنجیری را اعمال کنیم:

$$\frac{\partial L}{\partial W_j} = \sum_{i=1}^{N} \frac{\partial L}{\partial Z_i} \cdot \frac{\partial Z_i}{\partial W_j}$$

ما می‌دانیم که $\frac{\partial L}{\partial Z_i}$ داده شده است، و می‌توانیم $\frac{\partial Z_i}{\partial W_j}$ را به صورت زیر استخراج کنیم:

$$\frac{\partial Z_i}{\partial W_j} = X_{i+j}$$

با جای‌گذاری این معادله در معادله قانون زنجیری، خواهیم داشت:

$$\frac{\partial L}{\partial W_j} = \sum_{i=1}^{N} \frac{\partial L}{\partial Z_i} \cdot X_{i+j}$$

این معادله یک عملیات کانولوشن را نشان می‌دهد، جایی که گرادیان $\frac{\partial L}{\partial Z_i}$ با ورودی X_i جابه‌جا شده با موقعیت‌های j ضرب می‌شود. بنابراین، فرمول در واقع نتیجه یک عملیات کانولوشن است. به طور خلاصه:

$$\frac{\partial L}{\partial W_j} = \left(\frac{\partial L}{\partial Z}\right) * X_{\text{by shifted} j}$$

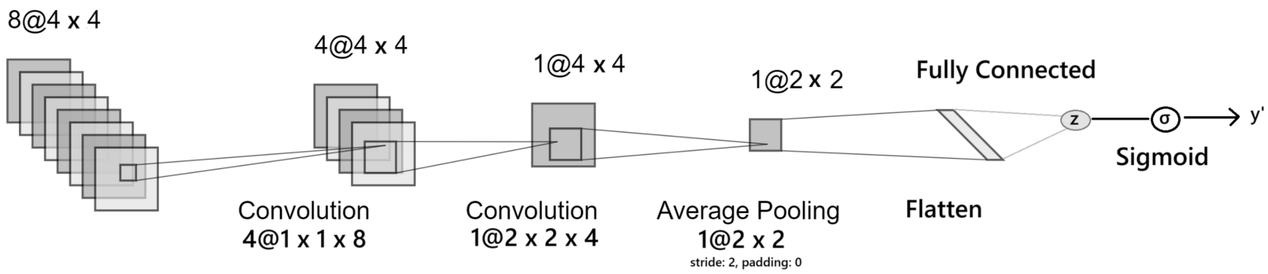
یا به صورت کلی‌تر:

$$\frac{\partial L}{\partial W} = \left(\frac{\partial L}{\partial Z}\right) * X$$

یعنی کافیست گرادیان upstream را یکبار با ورودی لایه‌ی کانولوشن، convolve کنیم تا گرادیان نسبت به کرنل کانولوشن به دست آید.

۳. (۱۵ نمره)

یکی از اساسی‌ترین اجزا در یک شبکه ژرف، انتشار به عقب یا Backpropagation است. شبکه داده شده در شکل ۲ را در نظر بگیرید. در این شکل، وزن‌های لایه اول با $W^{(۱)}$ و وزن‌های لایه دوم با $W^{(۲)}$ و وزن‌های لایه تمام‌متصل با $W^{(fc)}$ نمایش داده شده‌اند. همچنین، $P^{(۱)}$، $P^{(۲)}$ و $P^{(۳)}$ به ترتیب خروجی‌های لایه اول، دوم و سوم شبکه هستند.



شکل۲ :معماری شبکه

- ورودی شبکه: یک تنسور با ابعاد $۸ \times ۴ \times ۴$
- لایه اول: لایه کانولوشن با ۴ فیلتر $۸ \times ۱ \times ۱$ که خروجی $P^{(۱)}$ با ابعاد $۴ \times ۴ \times ۴$ تولید می‌کند.
- لایه دوم: لایه کانولوشن با ۱ فیلتر $۴ \times ۲ \times ۲$ که خروجی $P^{(۲)}$ با ابعاد $۱ \times ۴ \times ۴$ تولید می‌کند.
- لایه سوم: لایه Avg. Pooling با کرنل ۲×۲ و گام ۲، که خروجی $P^{(۳)}$ با ابعاد $۱ \times ۲ \times ۲$ ایجاد می‌کند.
- لایه چهارم: لایه Flatten که $P^{(۳)}$ را به یک بردار $\mathbf{v}$ با ابعاد ۱×۴ تبدیل می‌کند.
- لایه پنجم: لایه تمام‌متصل با تابع فعالساز Sigmoid که خروجی نهایی را تولید می‌کند.

وزن‌های فیلترهای کانولوشن در لایه‌های اول و دوم ($W^{(۱)}$ و $W^{(۲)}$) و وزن‌های لایه تمام‌متصل ($W^{(fc)}$) از جمله پارامترهای قابل‌آموزش شبکه هستند. هدف شبکه، بهینه‌سازی این وزن‌ها برای کمینه‌سازی یک تابع زیان L است.

وظایف:

(آ) با استفاده از قاعده مشتق زنجیره‌ای و بر اساس مشتق $\frac{\partial L}{\partial z}$، عبارت‌های زیر را محاسبه کنید:

i. $\frac{\partial L}{\partial P^{(۳)}_{i,j}}$: گرادیان زیان نسبت به خروجی لایه Average Pooling.

ii. $\frac{\partial L}{\partial P^{(۱)}_{i,j,k}}$: گرادیان زیان نسبت به خروجی لایه اول کانولوشن.

پاسخ:

ابتدا با توجه به معماری شبکه‌ی عصبی معرفی شده، مقادیر مختلف در لایه‌های مختلف را به دست آوریم.

لایه اول

$$P^{(1)}_{i,j,t} = \sum_{k=1}^{8} W^{(1)}_{1,1,k,t} X_{i,j,k}$$

لایه دوم

$$P^{(2)}_{i,j} = \sum_{k=1}^{4}\sum_{w=1}^{2}\sum_{h=1}^{2} W^{(2)}_{w,h,k} P^{(1)}_{i+w-1,j+h-1,k}$$

لایه سوم

$$P^{(3)}_{i,j} = \frac{1}{4} \sum_{w=2i-1}^{2i} \sum_{h=2j-1}^{2j} P^{(2)}_{w,h}$$

لایه چهارم

$$h^{(4)} = P^{(3)}_{1,1}, P^{(3)}_{1,2}, P^{(3)}_{2,1}, P^{(3)}_{2,2}$$

لایه پنجم

$$z = \sum_{i=1}^{4} W^{(\text{fc})}_i h^{(4)}_i$$
$$\hat{y} = \sigma(z)$$

i.

دقت کنید که می‌توان نوشت :

$$\frac{\partial L}{\partial h^{(4)}_k} = \frac{\partial L}{\partial \hat{y}} \times \frac{\hat{y}}{z} \times \frac{\partial z}{\partial h^{(4)}_k} = \frac{\partial L}{\partial \hat{y}} \times \hat{y}(1-\hat{y}) \times W^{(\text{fc})}_k$$

طبق تعریف داریم:

$$P^{(3)}_{i,j} = h^{(4)}_{2i+j-2} \quad (1 \le i,j \le 2)$$

پس به دست می‌آید:

$$\frac{\partial h^{(4)}_k}{\partial P^{(3)}_{i,j}} = I(k = 2i+j-2)$$

در نتیجه خواهیم داشت:

$$\frac{\partial L}{\partial P^{(3)}_{i,j}} = \sum_{k=1}^{4} \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z} \times \frac{\partial z}{\partial h^{(4)}_k} \times \frac{\partial h^{(4)}_k}{\partial P^{(3)}_{i,j}} = \frac{\partial L}{\partial \hat{y}} \times \hat{y}(1-\hat{y}) \times W^{(\text{fc})}_{2i+j-2}$$

ii.

می توان نوشت :

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

ابتدا داریم:

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

به توجه به آنچه آوردیم:

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

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

بنابراین خواهیم داشت:

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

(ب) گرادیان زیان نسبت به وزن $W^{(۱)}_{۱,۱,k}$ یکی از فیلترهای لایه اول کانولوشن را محاسبه کنید:

$$\frac{\partial L}{\partial W^{(۱)}_{۱,۱,k}}$$

پاسخ:

حال با توجه به رابطه ای که به دست آوردیم داریم :

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

با جایگذاری رابطه قبلی داریم :

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

(ج) عبارت گرادیان $\frac{\partial L}{\partial W^{(۲)}_{i,j,k}}$، یکی از وزن‌های فیلتر کانولوشن لایه دوم را به‌دست آورید.

پاسخ:

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

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

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

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 مشتق خواسته شده به شکل زیر محاسبه می‌شود:

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

۴. (۲۲ نمره) در این سوال قصد داریم برخی گونه‌های مختلف شبکه‌های پیچشی و همچنین معماری‌های مبتنی بر CNN معرفی شده در دهه اخیر را مورد بررسی و تحلیل قرار دهیم.

الف) شبکه DenseNet

- تفاوت اصلی ResNet's residual connections و DenseNet's dense connections را بیان کنید. در مورد هر کدام نیز، ابتدا توضیح مختصری بدهید.

پاسخ:

تفاوت اصلی دو مدل ذکر شده را می‌توان در نحوه ارتباط لایه‌های مختلف شبکه، به شکل زیر بیان نمود:

- **DenseNet در Dense Connections:** 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.
- **ResNet در Residual Connections:** در ResNet، اتصال بین لایه‌ها از طریق جمع ویژگی‌ها (Summation) 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 (Short-cut 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 در نظر 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 مؤثرتری از مشکل vanishing gradient جلوگیری می‌کند.

- توضیح دهید که DenseNet چگونه مشکل vanishing gradient را کاهش می‌دهد و مزیت محاسباتی آن چیست؟

پاسخ:

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 بدون کاهش ضعیف 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.

- با در نظر گرفتن نرخ رشد k در DenseNet، اگر هر لایه k ویژگی نگاشت جدید تولید کند و ورودی یک dense block دارای ۳۲ کانال باشد، اگر ۲۴ $= k$ باشد، لایه سوم در بلوک چند کانال خروجی خواهد داشت؟

پاسخ:

اگر تعداد کانال‌های ورودی اولیه ۳۲ باشد و بلوک دارای ۳ لایه باشد:

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

$$۳۲ + ۲۴ \times ۳ = ۳۲ + ۷۲ = ۱۰۴$$

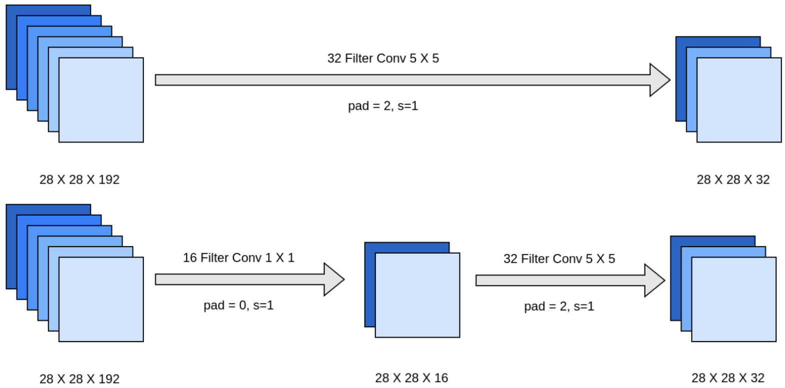
$$\text{تعداد کانال‌های خروجی} = ۱۰۴$$

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

ب) شبکه GoogleNet

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

- در شکل ۳ یک لایه کانولوشن به دو صورت نشان داده شده است. ورودی هر دو حالت $۲۸ \times ۲۸ \times ۱۹۲$ و خروجی هر دو $۲۸ \times ۲۸ \times ۳۲$ 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 ۱×۱، پیچیدگی محاسباتی را چند درصد کاهش یا افزایش داده است؟



شکل۳ :لایه کانولوشن معمولی (بالا) و لایه کانولوشن با فیلتر ۱×۱ (پایین)

پاسخ:

- **حالت اول: کانولوشن عادی** (۵×۵)
  - در این حالت، از ۳۲ فیلتر ۵×۵ استفاده می‌شود.
  - هر فیلتر با ورودی $28 \times 28 \times 192$ اعمال شده و خروجی تولید می‌کند.
  - تعداد کل عملیات محاسباتی به صورت زیر محاسبه می‌شود:

$$\text{عملیات:}\quad ۲۸ \times ۲۸ \times ۱۹۲ \times ۳۲ \times (۵ \times ۵) = ۱۲۰,۴۲۲,۴۰۰$$

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

$$۲۸ \times ۲۸ \times ۱۹۲ \times ۱۶ \times (۱ \times ۱) = ۲,۴۰۷,۴۲۴$$

    - **مرحله دوم: کانولوشن ۵×۵ روی کانال‌های کاهش یافته**
      تعداد عملیات:

$$۲۸ \times ۲۸ \times ۱۶ \times ۳۲ \times (۵ \times ۵) = ۱۰,۶۴۸,۵۷۶$$

  - **جمع کل:**

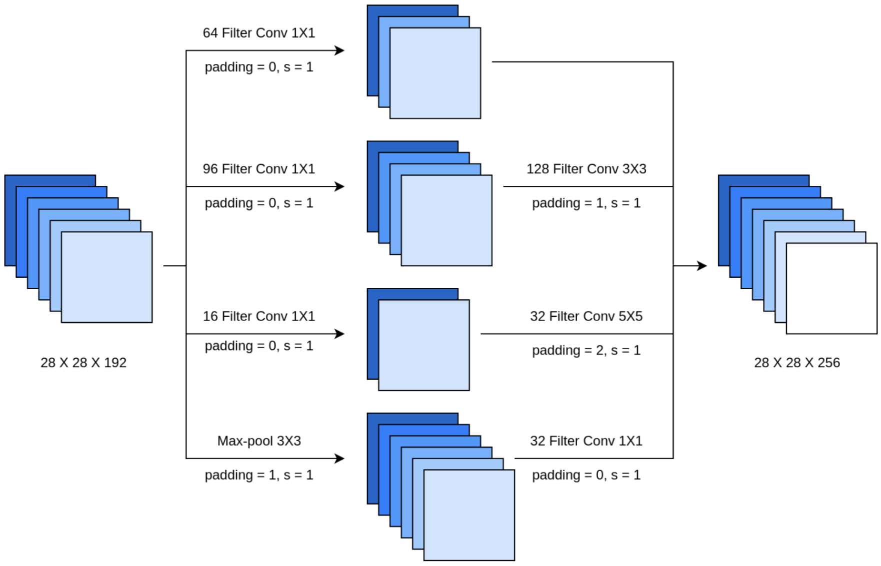
$$\text{عملیات}\; ۲,۴۰۷,۴۲۴ + ۱۰,۶۴۸,۵۷۶ = ۱۳۰,۰۵۶,۰۰۰$$

- **درصد کاهش پیچیدگی:**
  - کاهش پیچیدگی با استفاده از فرمول زیر محاسبه می‌شود:

$$\frac{۱۲۰,۴۲۲,۴۰۰ - ۱۳۰,۰۵۶,۰۰۰}{۱۲۰,۴۲۲,۴۰۰} \times ۱۰۰ \approx ۸۹/۲\%$$

مشاهده می‌شود که پیچیدگی محاسباتی مدل حدوداً ۹۰٪ کاهش پیدا کرده است که به شدت قابل توجه می‌باشد.

- در شکل ۴ یک پیمانه پیدایش به صورت کامل نشان داده شده است. ترکیب متوالی این پیمانه‌ها با هم، شبکه GoogleNet را تشکیل می‌دهند. توضیح دهید که دلیل استفاده از فیلترهایی با اندازه متفاوت (برای مثال ۱ × ۱، ۳ × ۳ و ۵ × ۵) و حتی فیلترهای با انواع متفاوت (استفاده از Max pooling در کنار کانولوشن) چیست؟ تعداد کل محاسبات در این شبکه را بدست آورید.



شکل۴ : شمای کلی پیمانه پیدایش

پاسخ:

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

- برای فیلتر ۱ × ۱ با ۶۴ کانال:
  $۲۸ \times ۲۸ \times ۱۹۲ \times ۶۴ \times (۱ \times ۱) = ۹,۶۳۳,۶۰۰$
- برای فیلتر ۱ × ۱ با ۹۶ کانال:
  $۲۸ \times ۲۸ \times ۱۹۲ \times ۹۶ \times (۱ \times ۱) = ۱۴,۴۳۸,۴۰۰$
- برای فیلتر ۱ × ۱ با ۱۶ کانال:
  $۲۸ \times ۲۸ \times ۱۹۲ \times ۱۶ \times (۱ \times ۱) = ۲,۴۰۷,۴۲۴$
- برای Max Pooling ۳ × ۳:
  $۲۸ \times ۲۸ \times ۱۹۲ \times (۳ \times ۳) = ۱,۳۵۴,۷۵۲$
- برای فیلتر ۳ × ۳ با ۱۲۸ کانال:
  $۲۸ \times ۲۸ \times ۹۶ \times ۱۲۸ \times (۳ \times ۳) = ۲۷,۷۰۴,۵۷۶$
- برای فیلتر ۵ × ۵ با ۳۲ کانال:
  $۲۸ \times ۲۸ \times ۱۶ \times ۳۲ \times (۵ \times ۵) = ۱۰,۶۴۸,۵۷۶$
- برای فیلتر ۱ × ۱ با ۳۲ کانال:
  $۲۸ \times ۲۸ \times ۱۹۲ \times ۳۲ \times (۱ \times ۱) = ۵,۹۴۹,۴۴۰$

جمع کل عملیات‌ها:

$$۹,۶۳۳,۶۰۰ + ۱۴,۴۳۸,۴۰۰ + ۲,۴۰۷,۴۲۴ + ۱,۳۵۴,۷۵۲$$

$$+۲۷,۷۰۴,۵۷۶ + ۱۰,۶۴۸,۵۷۶ + ۵,۹۴۹,۴۴۰ = ۷۲,۱۲۸,۳۶۸$$

دلایل استفاده از فیلترها با اندازه‌های مختلف:

- **پردازش چند مقیاس:** فیلترهای ۳ × ۳ و ۵ × ۵ ویژگی‌های بزرگ‌تر و پیچیده‌تر را استخراج می‌کنند، در حالی که فیلتر ۱ × ۱ ویژگی‌های محلی را بررسی می‌کند و تعداد کانال‌ها را کاهش می‌دهد.

- **کاهش ابعاد:** کانولوشن ۱ × 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.

• آیا این معماری نسبت به vanishing gradient مقاوم است؟ همچنین توضیح دهید که auxiliary classifier موجود در این معماری چگونه به بهبود جریان گرادیان و پایداری بهینه‌سازی در شبکه کمک می‌کند. در نهایت، محدودیت‌های احتمالی این طراحی در کاربردهای مدرن یادگیری عمیق را ارزیابی کنید.

پاسخ:

گوگل‌نت (GoogLeNet)، که به عنوان بخشی از خانواده معماری‌های Inception در سال ۲۰۱۴ معرفی شد، طراحی شبکه‌های عصبی کانولوشنی را با تمرکز بر کارایی، ماژولار بودن و مقیاس‌پذیری متحول کرد.

۱. مقاومت در برابر مشکل ناپدید شدن گرادیان (Vanishing Gradient)

مشکل ناپدید شدن گرادیان زمانی رخ می‌دهد که گرادیان‌ها هنگام بازپراکنش (backpropagation) در شبکه‌های عمیق به‌طور نمایی کاهش می‌یابند. این مشکل مانع از آموزش مؤثر شبکه‌های بسیار عمیق می‌شود.
گوگل‌نت به‌طور غیرمستقیم این مشکل را از طریق چندین ویژگی معماری حل می‌کند:

- **ماژولار بودن در عمق شبکه:** به‌جای انباشتن لایه‌های کانولوشنی یکپارچه، گوگل‌نت از ماژول‌های Inception استفاده می‌کند. این ماژول‌ها بلوک‌هایی ماژولار هستند که فیلترهایی با اندازه‌های مختلف را به‌صورت موازی ترکیب می‌کنند. این طراحی به شبکه اجازه می‌دهد عمیق‌تر باشد بدون اینکه مشکل ناپدید شدن گرادیان را تشدید کند، چرا که هر ماژول بر استخراج ویژگی‌ها در سطوح مختلف تمرکز دارد.
- **نرمال‌سازی دسته‌ای (Batch Normalization):** اگرچه در نسخه اولیه گوگل‌نت وجود نداشت (در Inception-v2 معرفی شد)، نرمال‌سازی دسته‌ای، فعال‌سازی‌های میانی را نرمال کرده و تغییرات ناگهانی داخلی را کاهش می‌دهد، که به پایداری گرادیان کمک می‌کند.
- **کلاس‌یفایرهای کمکی (Auxiliary Classifiers):** کلاس‌یفایرهای کمکی لایه‌های softmax میانی هستند که در بخش‌های مختلف شبکه قرار داده می‌شوند. این کلاس‌یفایرها به‌عنوان "تقویت‌کننده‌های گرادیان" عمل می‌کنند و سیگنال‌های گرادیان اضافی را به لایه‌های قبلی در طی بازپراکنش ارسال می‌کنند. این ویژگی به کاهش مشکل ناپدید شدن گرادیان کمک می‌کند.

۲. کلاس‌یفایرهای کمکی در گوگل‌نت

کلاس‌یفایرهای کمکی لایه‌های خروجی میانی هستند که برای کمک به فرآیند آموزش اضافه شده‌اند. این کلاس‌یفایرها سیگنال‌های خطای اضافی را به لایه‌های قبلی ارسال کرده و جریان گرادیان را بهبود می‌بخشند.
اثربخشی:

- **بهبود جریان گرادیان:** کلاس‌یفایرهای کمکی با تزریق سیگنال‌های گرادیان در نقاط میانی، به کاهش ناپدید شدن گرادیان کمک می‌کنند و تضمین می‌کنند که لایه‌های اولیه به‌طور مؤثری آموزش ببینند.
- **منظم‌سازی (Regularization):** با مجبور کردن لایه‌های میانی به تولید ویژگی‌های معنادار، آموزش شبکه را تثبیت کرده و تعمیم‌پذیری آن را بهبود می‌بخشند.

محدودیت‌ها:

- **راه‌حل ناقص:** این کلاس‌یفایرها مشکل ناپدید شدن گرادیان را تا حدی کاهش می‌دهند، اما به‌طور کامل حل نمی‌کنند، به‌ویژه در شبکه‌های بسیار عمیق.
- **استفاده موقتی:** تأثیر این کلاس‌یفایرها فقط در طی آموزش است و در فرآیند استنتاج (Inference) نقشی ندارند.
- **جایگزین‌های مدرن:** تکنیک‌هایی مانند اتصالات باقی‌مانده (Residual Connections) در ResNet جایگزین کلاس‌یفایرهای کمکی شده‌اند، زیرا راه‌حل‌های قوی‌تری برای جریان گرادیان ارائه می‌دهند.

با توجه به توضیحات بالا می‌توان نتیجه گرفت کلاس‌یفایرهای کمکی در گوگل‌نت تا حدی در جلوگیری از ناپدید شدن گرادیان مؤثر بودند، به‌ویژه در شبکه‌هایی با عمق مشابه. اما امروزه با ظهور روش‌های برتر در معماری‌های مدرن مانند اتصالات باقی‌مانده، این تکنیک به‌طور گسترده‌ای منسوخ شده است.

ج) شبکه‌های کانولوشنی Deformable

- تفاوت شبکه‌های کانولوشنی عادی و شبکه‌های کانولوشنی Deformable را از نظر grid sampling مقایسه کنید. همچنین شبکه‌های Deformable چگونه می‌توانند نسبت به Geometric transformation در تصاویر انعطاف‌پذیر باشند؟
- مفهوم offset در این شبکه‌ها به چه معناست و چگونه محاسبه می‌شود؟

پاسخ:

شبکه‌های کانولوشن معمولی از یک ساختار شبکه نمونه‌گیری ثابت استفاده می‌کنند. به‌عنوان مثال، در یک هسته $K \times K$، 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 آفسِت‌های قابل یادگیری به نقاط شبکه نمونه‌گیری، این مشکل را حل می‌کنند. این آفسِت‌ها، 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 آفسِت‌های متغیر و قابل یادگیری است که امکان نمونه‌گیری از مکان‌های غیرثابت را فراهم می‌کنند.
معادله کانولوشن معمولی:

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

معادله کانولوشن Deformable با اضافه کردن آفسِت‌ها:

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

در اینجا Δp_n آفسِت‌هایی هستند که در طی فرآیند یادگیری تعیین می‌شوند.

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

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

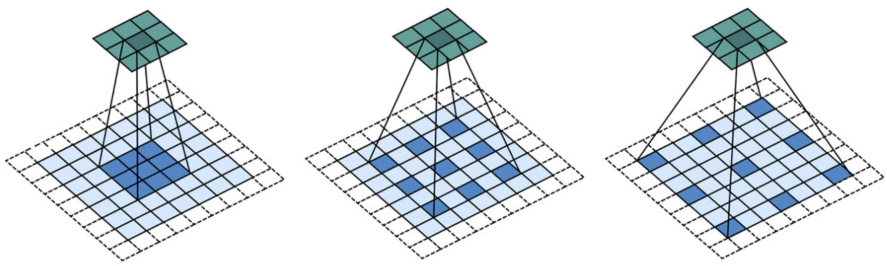
$$x(p) = \sum_{q} G(q,p) \cdot x(q)$$

که در آن $G(q,p)$ وزن 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.

۵. (۲۱ نمره) در شبکه های پیچشی به صورت متداول از لایه های کانولوشن ساده استفاده می شود که با آن آشنا هستید. نوع دیگری از لایه ها که می توان از آنان ددر شبکه های پیچشی استفاده نمود، لایه کانولوشن گسترش یافته یا متسع است. در شکل ۵ تصویر شهودی از فیلتر کانولوشن گسترش یافته ارائه شده است، این فیلتر ها میان خانه هایی که فیلتر با استفاده از اطلاعات آن‌ها لایه بعد را محاسبه می‌کنند فاصله می‌اندازند یا به بیانی دیگر در زمان اعمال فیلتر و انجام عملیات ضرب کانولوشن، بر روی ورودی با گام (dilated) بزرگتری حرکت می کنیم، توجه کنید طول گام مفهومی متفاوت نسبت به طول گام (stride) در لایه های شبکه کانولوشن دارد.



شکل۵ : شهودی از کانولوشن گسترش یافته با گام های متفاوت

همانطور که در شکل ۱ نیز مشخص است این روش، یک روش کم هزینه برای افزایش محدوده دید شبکه های پیچشی است. کانولوشن گسترش یافته بصورت فرم بسته ریاضی زیر تعریف می‌شود.

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

الف) برای ورودی $I \in \mathbb{R}^{M\times N}$ و کرنل $K \in \mathbb{R}^{F\times F}$ نشان دهید خروجی عملگر کانولوشن گسترش یافته دارای ابعاد $(M-DF+D)\times(N-DF+D)$ است.

پاسخ:

طبق فرض صورت سوال، کرنل اولیه دارای ابعاد $F \times F$ می‌باشد. 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:

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

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:

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

که در آن:

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

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

ب) نشان دهید کانولوشن گسترش یافته معادل کانولوشن با کرنل متسع شده K' است. ماتریس A را مشخص کنید. $(K' = K \otimes A)$، عملگر $\otimes$ کرونکر محصول `kronecker product` است.)

پاسخ:

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 می‌کنیم که شماره‌گذاری سطرها و ستون‌های ماتریس از صفر شروع می‌شوند.

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

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

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 گسترش‌یافته 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:

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

$$= \sum_{p=-\infty}^{\infty} \sum_{q=-\infty}^{\infty} K(p,q) \cdot I(i+pD, j+qD)$$

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

که همان خواسته مسئله است.

ج) فرض کنید یک لایه ورودی با ابعاد $M \times N$ به یک شبکه عصبی با سه لایه کانولوشن گسترش یافته داده می‌شود. در این شبکه، هر لایه کانولوشن شامل تعدادی فیلتر است که بر روی قسمت‌های مختلف ورودی حرکت می‌کند و ویژگی‌های گوناگون آن را استخراج می‌کند. هدف ما بررسی تعداد پیکسل‌های قابل مشاهده توسط یک عنصر خاص مانند (i,j) از خروجی است. به عبارت دیگر، محدوده‌ای از ورودی که این عنصر را تحت تأثیر قرار می‌دهد را در صورت وجود امکان محاسبه به صورت پارامتری بیابید.

ابعاد فیلترها به ترتیب از چپ به راست برابر هستند با:

$$(w-۱) \times (w-۱), \quad w \times w, \quad (w+۱) \times (w+۱)$$

پارامتر گسترش این فیلترها به ترتیب از چپ به راست برابر است با:

$$d-۱, \quad d, \quad d+۱$$

گستره دید در شبکه‌های عصبی کانولوشنی

۱. گستره دید در یک‌بعد

۱.۱ تنظیمات و تعاریف

ما L لایه کانولوشنی داریم که با $\ell = ۱, ۲, \dots, L$ شماره‌گذاری شده‌اند. برای هر لایه ℓ، موارد زیر تعریف می‌شوند:

- **اندازه کرنل**: $k_\ell \in \mathbb{N}$.
- **اتساع**: $d_\ell \in \mathbb{N}$.
- **گام**: $s_\ell \in \mathbb{N}$.

فرض می‌کنیم Padding اضافی وجود ندارد. (یا فقط بخش‌هایی را که واقعاً ورودی را مشاهده می‌کنند در نظر بگیریم). دو مقدار در لایه ℓ تعریف می‌شوند:

- R_ℓ: اندازه گستره دید در ورودی اصلی برای یک موقعیت خروجی در لایه ℓ.
- jump_ℓ: میزان حرکت (به واحد اندیس‌های ورودی) وقتی که یک گام در خروجی لایه ℓ حرکت کنیم.

برای لایه پایه $\ell = ۰$:

$$R_۰ = ۱, \quad \mathrm{jump}_۰ = ۱.$$

۱.۲ رابطه بازگشتی لایه‌به‌لایه

برای هر لایه $\ell \geq ۱$:

$$R_\ell = R_{\ell-۱} + (k_\ell - ۱) \cdot d_\ell \cdot \mathrm{jump}_{\ell-۱}, \quad (۱)$$

$$\mathrm{jump}_\ell = \mathrm{jump}_{\ell-۱} \cdot s_\ell. \quad (۲)$$

توضیح شهودی

- **رشد گستره دید**: اگر یک موقعیت خروجی در لایه ℓ را در نظر بگیریم، آن خروجی با کانولوشن یک فیلتر با اندازه k_ℓ و اتساع d_ℓ در خروجی لایه $\ell - ۱$ محاسبه می‌شود. گستره دید با مقدار $(k_\ell - ۱) \cdot d_\ell$ گسترش می‌یابد.
- **اندازه گام**: جابه‌جایی یک موقعیت در خروجی لایه ℓ معادل جابه‌جایی s_ℓ موقعیت در خروجی در لایه $\ell - ۱$ است.

۱.۳ روابط کلی

با استفاده از روابط بازگشتی:

$$\begin{aligned}
\mathrm{jump}_۰ &= ۱, \\
\mathrm{jump}_\ell &= \mathrm{jump}_{\ell-۱} \cdot s_\ell, \\
R_۰ &= ۱, \\
R_\ell &= R_{\ell-۱} + (k_\ell - ۱) \cdot d_\ell \cdot \mathrm{jump}_{\ell-۱}.
\end{aligned}$$

۲. بسط به دوبعدی

۲.۱ روابط در ابعاد افقی و عمودی

برای کانولوشن دوبعدی، موارد زیر تعریف می‌شوند:

- **اندازه کرنل**: (k_ℓ^H, k_ℓ^W).
- **اتساع**: (d_ℓ^H, d_ℓ^W).
- **گام**: (s_ℓ^H, s_ℓ^W).

روابط برای هر بُعد $\alpha \in \{H, W\}$:

$$\begin{aligned}
R_\ell^{(\alpha)} &= R_{\ell-۱}^{(\alpha)} + \left(k_\ell^{(\alpha)} - ۱\right) \cdot d_\ell^{(\alpha)} \cdot \mathrm{jump}_{\ell-۱}^{(\alpha)}, \\
\mathrm{jump}_\ell^{(\alpha)} &= \mathrm{jump}_{\ell-۱}^{(\alpha)} \cdot s_\ell^{(\alpha)}.
\end{aligned}$$

۲.۲ کرنل‌های مربعی و گام و اتساع یکسان

اگر $k_\ell^H = k_\ell^W = k_\ell$، $d_\ell^H = d_\ell^W = d_\ell$ و $s_\ell^H = s_\ell^W = s_\ell$، گستره دید در هر دو بُعد برابر خواهد بود:

$$R_\ell^{(H)} = R_\ell^{(W)} = R_\ell^{(۱\mathrm{D})}.$$

بنابراین گستره دید نهایی دوبعدی برابر است با:

$$R_\ell^{(۲\mathrm{D})} = \left(R_\ell^{(۱\mathrm{D})}\right)^۲.$$

پاسخ:

محاسبه گستره دید برای شبکه سه‌لایه

(آ) **لایه ۱:**

$$R_۱ = ۱ + \left(k_۱ - ۱\right) \cdot d_۱ = ۱ + (w - ۲)(d - ۱).$$

(ب) **لایه ۲:**

$$R_۲ = R_۱ + \left(k_۲ - ۱\right) \cdot d_۲ \cdot s = \left[۱ + (w - ۲)(d - ۱)\right] + (w - ۱)ds.$$

(ج) **لایه ۳:**

$$R_۳ = R_۲ + \left(k_۳ - ۱\right) \cdot d_۳ \cdot s^۲ \quad = \left[۱ + (w - ۲)(d - ۱) + (w - ۱)ds\right] + w(d + ۱)s^۲.$$

فرمول نهایی گستره دید در یک‌بعد

$$R_۳ = ۱ + (w - ۲)(d - ۱) + (w - ۱)ds + w(d + ۱)s^۲.$$

گستره دید نهایی در دوبعد

برای کرنل‌های مربعی:

$$R_۳^{(2\mathrm{D})} = \left(R_۳\right)^۲.$$

د) در مورد Masked Convolution، کاربرد و محدودیت های آن تحقیق کنید و ضمن توضیح مختصری دراین باره پاسخ دهید چگونه می توان با استفاده از کانولوشن گسترش یافته محدودیت Masked Convolution را بهبود بخشید؟

پاسخ:

کانولوشن ماسک‌شده (Masked Convolution) در حوزه یادگیری عمیق، به ویژه در مدل‌هایی مانند Pix-elCNN 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 یک یا نقاط خاصی بهره‌برداری کنند. به عنوان مثال:

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

محدودیت‌ها:

- **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.
- **عمق زیاد شبکه:** با افزایش دامنه دید، نیاز به تعداد زیادی لایه وجود دارد که این موضوع می‌تواند محاسبات را پیچیده کرده و ممکن است باعث 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:

- **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.
- **حفظ ساختار محلی و جلوگیری از دسترسی غیرمجاز:** ترکیب کانولوشن ماسک‌شده با کانولوشن گسترش‌یافته، امکان حفظ نواحی 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.
- **کاهش عمق شبکه و پیچیدگی محاسباتی:** با گسترش دامنه دید در هر لایه از طریق کانولوشن گسترش‌یافته، نیاز به وجود لایه‌های متعدد برای افزایش عمق دید کاهش می‌یابد. این امر منجر به کاهش پیچیدگی محاسباتی و افزایش کارایی می‌شود.
- **افزایش انعطاف‌پذیری مدل:** کانولوشن گسترش‌یافته به مدل امکان می‌دهد تا تعادل بهینه بین دقت محلی و اطلاعات کلی برقرار کند که این امر برای درک بهتر ویژگی‌های مختلف تصویر ضروری است.

۶. (۲۲ نمره) در این سوال به بررسی پدیده vanishing gradient یا "گرادیان ناپدید شونده" می‌پردازیم.

الف) (امتیازی) پیش از ابداع روش‌های مبتنی بر شبکه‌های عصبی کانولوشنی، feedforward neural network ها قادر به استخراج ویژگی‌ها از تصاویر به‌طور مستقل نبودند و معمولاً استخراج ویژگی‌ها با روش‌هایی غیر از یادگیری عمیق انجام می‌شد. از طرفی، می‌دانیم که عمیق‌تر کردن شبکه عصبی می‌تواند باعث شود مدل ویژگی‌های پیچیده‌تری را بیاموزد، اما عمیق‌تر کردن شبکه بدون استفاده از تکنیک‌های خاص مشکلاتی نیز به وجود می‌آورد، یکی از مشکلات اصلی که در این فرآیند بروز می‌کند، پدیده vanishing gradient است.

فرض کنید یک شبکه عصبی با تعداد زیادی لایه داریم. می‌دانیم که گرادیان وزن‌های لایه i از طریق عبارت زیر محاسبه می‌شود (که در آن، δ_L مشتق تابع ضرر نسبت به خروجی لایه آخر، W_k وزن لایه k، f تابع فعال‌سازی و z ورودی تابع فعال‌سازی است.)

$$\frac{\partial J}{\partial W^i} = (\delta^i)^T \times \frac{\partial z^i}{\partial W^i} = (\delta^i)^T \times a^{i-۱}$$

$$\delta_i = \delta_L \left(\prod_{k=i+۱}^{L} W_k \cdot f'_{k-۱}(z_{k-۱}) \right)$$

فرض کنید بزرگ‌ترین singular value (مقدار تکین) برای همه ماتریس‌های وزن W_k کوچک‌تر از یک باشد، با فرض این‌که $\delta_L = M$ و تابع فعال‌سازی ما از نوع ReLU باشد، در این صورت نشان دهید که اگر $L \to \infty$ در این صورت، $|\delta_i| \to ۰$.

همچنین فرض کنید بزرگترین singular value همه ماتریس های وزن برابر ۰/۹ باشد، برای $L = ۱۰۰$ با شرایطی که بیان شد یک حد بالا برای $|\delta_i|$ در صورتی که $i = ۰$ باشد؛ پیدا کنید.

پاسخ:

اثبات $|\delta_i| \to ۰$ وقتی $L \to \infty$

گرادیان δ_i به‌صورت زیر تعریف شده است:

$$\delta_i = \delta_L \prod_{k=i+۱}^{L} W_k \cdot f'(z_{k-۱}).$$

با فرض اینکه:

- بزرگ‌ترین مقدار تکین هر ماتریس وزن $\sigma_{\max}(W_k) < ۱$.
- مشتق تابع فعال‌سازی ReLU $\|f'(z)\| \leq ۱$.

طبق قاعده مثلثی:

$$\|\delta_i\| \leq \|\delta_L\| \cdot \prod_{k=i+۱}^{L} \|W_k\| \cdot \|f'(z_{k-۱})\|.$$

از آنجا که $\|W_k\| = \sigma_{\max}(W_k)$ و $\sigma_{\max} < ۱$، نتیجه می‌گیریم:

$$\|\delta_i\| \leq \|\delta_L\| \cdot \prod_{k=i+۱}^{L} \sigma_{\max}(W_k).$$

با فرض اینکه $\sigma_{\max}(W_k) = \sigma < ۱$ برای همه k، داریم:

$$\|\delta_i\| \leq \|\delta_L\| \cdot \sigma^{L-i}.$$

وقتی $L \to \infty$، $\sigma^{L-i} \to ۰$ زیرا $\sigma < ۱$. بنابراین:

$$\|\delta_i\| \to ۰.$$

حد بالا برای $|\delta_i|$ وقتی $L = ۱۰۰$ و $\sigma_{\max} = ۰/۹$ و $i = ۰$

برای $L = ۱۰۰$ و $\sigma = ۰/۹$، داریم:

$$\|\delta_i\| \leq \|\delta_L\| \cdot \sigma^{L-i}.$$

برای مثال:

(آ) اگر $i = ۰$:

$$\|\delta_۰\| \leq \|\delta_L\| \cdot (۰/۹)^{۱۰۰-۰} = \|\delta_L\| \cdot (۰/۹)^{۱۰۰}.$$

با محاسبه $(۰/۹)^{۵۰}$:

$$(۰/۹)^{۵۰} \approx ۰/۰۰۰۰۲۷$$

بنابراین:

$$\|\delta_{۵۰}\| \leq \|\delta_L\| \cdot ۰/۰۰۰۰۲۷$$

ب) یکی از راهکارها برای جلوگیری از vanishing gradient استفاده از یک initialization مناسب است. فرض کنید یک لایه کانولوشن با طول و عرض ۸×۸ داریم که یک لایه نهفته با تعداد کانال ۳ را به یک لایه نهفته با تعداد کانال ۵ متصل می‌کند. اگر از Kaiming initialization با توزیع نرمال استفاده کنیم، پارامترهای توزیع نرمال را برای مقداردهی اولیه کرنل‌های این لایه به دست آورید.

راهنمایی: تعداد واحدهای ورودی به هر کرنل، حاصل‌ضرب تعداد کانال‌های ورودی در مساحت کرنل است.

پاسخ:

برای Kaiming Initialization با فرض تابع فعال‌سازی ReLU، وزن‌های هر کرنل از توزیع نرمال زیر نمونه‌برداری می‌شوند:

$$W \sim \mathcal{N}(۰, \mathrm{Var}(W)).$$

واریانس این توزیع به‌صورت زیر محاسبه می‌شود:

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

که در آن n_{in} تعداد واحدهای ورودی به هر کرنل است.

محاسبه n_{in}:

تعداد واحدهای ورودی به هر کرنل برابر است با:

$$n_{\text{in}} = (\text{تعداد کانال‌های ورودی}) \times (\text{مساحت کرنل}) = ۳ \times (۸ \times ۸) = ۱۹۲.$$

توزیع نرمال:

با جایگذاری $n_{\text{in}} = ۱۹۲$ در فرمول واریانس:

$$\mathrm{Var}(W) = \frac{۲}{۱۹۲} = ۰/۰۱۰۴.$$

انحراف معیار توزیع برابر است با:

$$\mathrm{Std}(W) = \sqrt{\mathrm{Var}(W)} = \sqrt{۰/۰۱۰۴} \approx ۰/۱۰۲.$$

نتیجه نهایی:

برای هر کرنل، وزن‌ها از توزیع نرمال با مشخصات زیر نمونه‌برداری می‌شوند:

$$W \sim \mathcal{N}(۰, ۰٫۰۱۰۴).$$

توضیح تکمیلی: در صورتی که در پارامترهای توزیع نرمال، به جای واریانس ($\text{Var} = ۰٫۰۱۰۴$)، انحراف معیار ($\text{Std} = ۰٫۱۰۲$) ذکر شود نیز پاسخ معتبر است.

ج) در اوایل توسعه‌ی شبکه‌های عصبی، تابع فعال‌سازی Sigmoid برای بسیاری از شبکه‌ها انتخاب می‌شد. این تابع، که مقادیر ورودی را بین صفر و یک نگاشت می‌کند، مناسب به نظر می‌رسید. اما به مرور زمان و با افزایش عمق شبکه‌ها، معلوم شد که استفاده از تابع Sigmoid نیز می‌تواند باعث بروز پدیده vanishing gradient شود، یکی از دلایلی که معماری‌های Modern Convolutional Neural Network در ابتدا نمی‌توانستند عملکرد کافی داشته باشند عدم امکان عمیق‌سازی آن‌ها به اندازه کافی بود.

در سال ۲۰۱۲ معماری AlexNet معرفی شد که نسبت به معماری مشابه قبلی خود LeNet پیشرفت بسیار قابل توجهی داشت، یکی از تفاوت‌های این معماری عمق بیشتر و استفاده از تابع فعال‌سازی ReLU به جای Tanh / Sigmoid بود.

با توجه به این موضوع، پاسخ دهید: چرا استفاده از تابع ReLU به جای Sigmoid به جلوگیری از مشکل vanishing gradient کمک می‌کند؟

پاسخ:

استفاده از تابع ReLU به جای Sigmoid به دلایل زیر به کاهش پدیده vanishing gradient کمک می‌کند:

- **مشکل گرادیان کوچک در Sigmoid:** در تابع Sigmoid، اگر مقدار ورودی x کمی از ۰ دور شود (چه مثبت چه منفی)، مقدار گرادیان بسیار کوچک می‌شود:

$$\sigma'(x) = \sigma(x)(۱ - \sigma(x)).$$

این مسئله باعث می‌شود که گرادیان‌ها هنگام انتشار به لایه‌های ابتدایی تقریباً صفر شوند. در مقابل، ReLU در محدوده $x > ۰$ گرادیانی برابر ۱ دارد و از این کاهش جلوگیری می‌کند.

- **عدم اشباع در ReLU:** برخلاف Sigmoid که خروجی آن بین ۰ و ۱ محصور است و به اشباع منجر می‌شود، ReLU هیچ محدودیتی برای مقادیر مثبت ندارد. این ویژگی کمک می‌کند گرادیان در لایه‌های عمیق کاهش نیابد و یادگیری بهتر انجام شود.

د) اگرچه ReLU مزایای زیادی ارائه می‌دهد، یکی از معایب آن مسئله‌ی Dead Neurons است، توضیح دهید این مشکل چیست. Leaky ReLU ، Parametric ReLU و ELU توابعی هستند که برای رفع مشکلات تابع ReLU طراحی شده اند، با استفاده از معادله و معادله مشتق این توابع بیان کنید هرکدام چه معایب و مزایایی نسبت به تابع ReLU دارند.

پاسخ:

اگر ورودی یک نورون مقدار منفی داشته باشد، خروجی ReLU صفر می‌شود و گرادیان نیز صفر خواهد بود. این موضوع باعث می‌شود نورون دیگر به‌روزرسانی نشود و به اصطلاح "مرده" بماند.

توابع جایگزین ReLU:

۱. **Leaky ReLU:**

$$f(x) = \begin{cases} x & x \geq ۰ \\ \alpha x & x < ۰ \end{cases}, \quad f'(x) = \begin{cases} ۱ & x \geq ۰ \\ \alpha & x < ۰ \end{cases}$$

مزایا: جلوگیری از Dead Neurons با دادن یک شیب کوچک ($\alpha > ۰$) به مقادیر منفی.
معایب: انتخاب مقدار α ثابت ممکن است برای همه مسائل بهینه نباشد.

۲. **Parametric ReLU (PReLU):**

$$f(x) = \begin{cases} x & x \geq ۰ \\ \alpha x & x < ۰ \end{cases}, \quad f'(x) = \begin{cases} ۱ & x \geq ۰ \\ \alpha & x < ۰ \end{cases}$$

α پارامتری قابل یادگیری است.
مزایا: انعطاف‌پذیری بیشتر با یادگیری مقدار بهینه‌ی α.
معایب: پیچیدگی بیشتر در آموزش و خطر بیش برازش.

۳. **Exponential Linear Unit (ELU):**

$$f(x) = \begin{cases} x & x \geq ۰ \\ \alpha(e^x - ۱) & x < ۰ \end{cases}, \quad f'(x) = \begin{cases} ۱ & x \geq ۰ \\ \alpha e^x & x < ۰ \end{cases}$$

مزایا: کاهش Dead Neurons و میانگین خروجی نزدیک به صفر برای سرعت‌دهی به همگرایی.
معایب: محاسبات بیشتر به دلیل استفاده از تابع نمایی.

ه) یک شبکه با ۲ بلاک residual را در نظر بگیرید. نشان دهید چگونه این بلاک‌ها در شبکه residual کمک می‌کند تا از مشکل vanishing gradient جلوگیری شود؟ این شبکه را با شبکه‌های نرمال بدون residual blocks مقایسه کنید. برای سادگی در نوشتار، تابعی که در هر بلاک روی ورودی اعمال می‌شود را با F نشان دهید که شامل activation function هم می‌شود.

پاسخ:

بلاک‌های residual با افزودن ورودی (x) به خروجی یک لایه ($\mathcal{F}(x)$) به حل مشکل vanishing gradient کمک می‌کنند. ساختار ریاضی شبکه دو بلاکی residual به صورت زیر است:

$$y_۱ = \mathcal{F}_۱(x) + x,$$
$$y_۲ = \mathcal{F}_۲(y_۱) + y_۱.$$

برای محاسبه گرادیان‌ها:

$$\frac{\partial y_۲}{\partial w_۲} = \mathcal{F}_۲'(y_۱).$$

$$\frac{\partial y_۲}{\partial w_۱} = (\mathcal{F}_۲'(y_۱) + ۱) \cdot \mathcal{F}_۱'(x),$$

در شبکه نرمال:

$$\frac{\partial y_۲}{\partial w_۲} = \mathcal{F}_۲'(y_۱).$$

$$\frac{\partial y_۲}{\partial w_۱} = \mathcal{F}_۲'(y_۱) \cdot \mathcal{F}_۱'(x),$$

در شبکه residual، وجود جملهٔ اضافی $+۱$ در گرادیان باعث می‌شود که جریان گرادیان بدون کاهش شدید از یک بلاک به بلاک دیگر منتقل شود. اما در شبکه‌های معمولی، گرادیان تنها به توابع تبدیل $\mathcal{F}_i$ وابسته است و در اثر ضرب‌های متوالی، با افزایش عمق شبکه به شدت کاهش می‌یابد، که این مسئله مشکل vanishing gradient را ایجاد می‌کند، اما در لایه‌های residual تا حد خوبی از این مشکل جلوگیری می‌شود.