به نام خدا

نام: فاطمه زهرا بخشنده **استاد: دکتر محمدرضا محمدی**

شماره دانشجویی: 98522157 **درس مبانی بینایی کامپیوتر**

گزارش تمرین 11:

سوال اول:

الف) در لایه های fully connected، مقدار هر نورون در لایه خروجی وابسته به تمام نورون ها در لایه قبل است. کانولوشن یک بعدی مشابه با لایه fully connected است اما هر نورون خروجی تنها به بخشی از نورون های لایه ورودی متصل است.

ورودی یک لایه کانولوشنی یک ماتریس است یعنی یک تصویر را ورودی می گیرد و اتصالات محلی دارد. اما ورودی لایه fully connected حتما یک vector است پس تصویر flatten می شود. خروجی لایه کانولوشنی حاصل فیلتر کردن ماتریس ورودی با فیلتر مربوطه است که به صورت مکانی بر روی آن لغزانده میشود.

بین نورون های یک لایه ی کانولوشنی میتوانیم وزن مشترک داشته باشیم، در نتیجه یکبار وزن محاسبه و آپدیت میشود و به طور مشترک در مکان های مختلف استفاده می شود، اما در لایه ی fully connected وزن ها کاملا مستقل هستند. و به همین دلیل در لایه های کانولوشنی می توانیم تعداد پارامتر خیلی کمتری داشته باشیم.

Weight sharing در لایه کانولوشنی کمک می کند شبکه دنبال فرمولی بگردد که با اعمال آن روی تصویر، مسئله ی مورد نظر را حل کند. در واقع لایه کانولوشنی با انجام عملیات کانولوشن می تواند یک الگو را در تصویر بیاموزد. قبلا ما یک فیلتر را دستی انتخاب کرده و با تصویر convolve می کردیم تا عملیات خاصی را روی تصویر انجام داده و یک ویژگی را از تصویر استخراج کنیم. همین کار را لایه کانولوشنی به طوری انجام می دهد که مناسب ترین فیلتر ها را برای استخراج ویژگی بیابد.

اما در لایه fully connected هر نورون مستقلا دنبال یک فرمول میگردد. در نتیجه یک لایه ی کانولوشنی با تعداد نورون خیلی کمتر می تواند بهتر از یک لایه ی fully connected با تعداد نورون‌های زیاد عمل کند، که احتمال overfit آن هم بالا است.

ب) مقدار گسترش مرز باید به صورت زیر باشد تا ابعاد خروجی تغییر نکند:

$$padding = W_k - 1 = 5 - 1 = 4$$

تعداد پارامتر ها:

$$parameters = (5 \times 5 \times 5 + 1) \times 16 = 2016$$

پ) بخش اول:

$$32 - 5 + 1 = 28$$

همچنین 3 فیلتر داریم پس خروجی این لایه $28 \times 28 \times 3$ است.

بخش دوم: تصویر را به لایه اول می دهیم.

$$32 - 3 + 1 = 30$$

همچنین 9 فیلتر داریم پس خروجی این لایه $30 \times 30 \times 9$ است. سپس خروجی این لایه را به لایه دوم می دهیم.

$$30 - 3 + 1 = 28$$

همچنین 9 فیلتر داریم پس خروجی این لایه $28 \times 28 \times 9$ است.

ت) Max Pooling: این نوع pooling، ماکسیمم مقدار را در ناحیه‌ای که فیلتر در Feature Map ورودی پوشانده است انتخاب می کند و ابعاد را نیز کاهش می دهد. در نهایت Feature Map خروجی، برجسته ترین ویژگی های Feature Map ورودی را دارد. از تصویر نیز پیکسل های روشن تر را انتخاب می کند. همچنین کار حذف نویز را نیز انجام می‌دهد. معمولا از این نوع pooling پس از لایه کانولوشنی استفاده می شود.

Min Pooling: در این نوع pooling، خلاصه ویژگی های یک ناحیه با minimum مقدار در آن ناحیه نشان داده می شود. بیشتر زمانی استفاده می‌شود که تصویر پس‌زمینه روشنی داشته باشد، زیرا min pooling، پیکسل‌های تیره‌تر را انتخاب می‌کند.

Average pooling: این نوع pooling، میانگین مقادیر را در ناحیه‌ای که فیلتر در Feature Map ورودی پوشانده است انتخاب می کند و ابعاد را به این صورت کاهش می دهد. Average pooling کاهش ابعاد را به عنوان مکانیزمی برای حذف نویز اجرا می‌کند، average pooling تصویر را صاف می کند و از این رو

ممکن است هنگام استفاده از این روش pooling، ویژگی های واضح شناسایی نشود. زمانی استفاده می‌شود که edge ها مهم نباشند.

Global average pooling: این نوع pooling جایگزین لایه fully connected است. و مشکل تعداد زیاد وزن‌های لایه fully connected را حل می کند. با استفاده از این لایه در انتها Feature Map را مستقیم به softmax وصل می کنیم. از معایب این نوع pooling این است که پارامتر مکان را از دست میدهیم و نمیدانیم هر ویژگی دقیقا در کدام پیکسل ها وجود دارد.

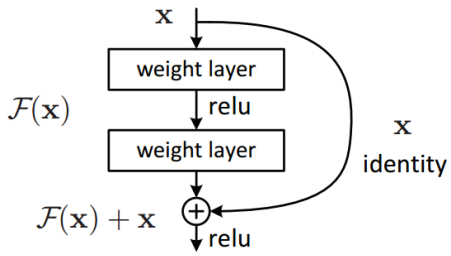
ث) Vggnet: شبکه vggnet با توجه به نیاز به کاهش پارامتر ها در لایه های کانولوشنی و بهبود زمان اجرا بوجود آمد. در vggnet برای بهبود لایه های کانولوشنی سعی شده هر چه به لایه های جلوتر میرویم خروجی از نظر مکانی کوچتر و از نظر تعداد ویژگی ها بزرگتر شود. این مدل از کرنل های کانولوشنی در اندازه 3×3 و max pooling با گام 2 استفاده می کند. ایده پشت kernel های با اندازه ثابت این است که تمام kernel های کانولوشنی با اندازه متغیر مورد استفاده در Alexnet ($11 \times 11, 5 \times 5, 3 \times 3$) را می‌توان با استفاده از چندین کرنل 3×3 به عنوان بلوک‌های ساختمانی تکرار کرد. این کار هم تعداد پارامتر ها را کاهش می دهد. هم میتوان بعد از اعمال هر کرنل، یک تابع غیر خطی استفاده کرد.

Vggnet حداکثر 19 لایه دارد. اگر این شبکه عمیق تر شود مشکل محو شدگی گرادیان بوجود می آید.

Resnet: برای حل مشکلی که برای vggnet ذکر شد، Resnet بوجود آمد. فرض کنید یک شبکه f(x) داریم که در یک مجموعه داده به دقت n% رسیده است. اکنون افزودن لایه‌های بیشتر به این شبکه g(f(x)) باید حداقل دقت n% داشته باشد، یعنی در بدترین حالت g باید یک نگاشت یکسان باشد که دقتی برابر با f(x) داشته باشد، در غیر این صورت بیشتر. اما متاسفانه اینطور نیست. آزمایش‌ها نشان داده‌اند که با افزودن لایه‌های بیشتر به شبکه، دقت کاهش می‌یابد. چون مشکل محو شدگی گرادیان بوجود می آید.

معماری ResNet از اتصالات میانبر برای حل مشکل محو شدگی گرادیان استفاده می کند. بلوک اصلی ResNet یک بلوک Residual است که در سراسر شبکه تکرار می شود. به جای یادگیری نقشه برداری از $x \rightarrow F(x)$، شبکه، نقشه برداری را از $x \rightarrow F(x)+G(x)$ یاد می گیرد. وقتی ابعاد ورودی x و خروجی F(x) یکسان باشد، تابع $G(x) = x$ یک تابع همانی است و اتصال میانبر را Identity Connection می نامند. نقشه برداری یکسان با صفر کردن وزن ها در لایه میانی در طول training آموخته می شود.

برای مواردی که ابعاد F(x) با x متفاوت است (به دلیل طول گام> 1 در لایه های کانولوشنی میانی)، اتصال Projection به جای اتصال Identity اجرا می شود. تابع G(x) ابعاد ورودی x را به خروجی F(x) تغییر می دهد.



دو نوع mapping در Resnet استفاده می شود.

mapping غیر قابل آموزش (Padding): از zero padding استفاده می شود تا ورودی x با F(x) مطابقت داشته باشد.

نگاشت قابل آموزش (لایه کانولوشنی): لایه کانولوشنی 1×1 برای نگاشت x به G(x) استفاده می‌شود. توسط این لایه، ابعاد مکانی ثابت می مانند یا اگر stride=2 باشد نصف می شوند و عمق را می توان با توجه به تعداد کرنل ها کنترل کرد. ایده کانولوشن 1×1 به منظور کاهش بیشتر تعداد عملیات و در عین حال استفاده از تعداد لایه ها و فیلترهای بیشتر در لایه های کانولوشنی مورد استفاده قرار گرفت. این نوع کانولوشن اجازه می دهد تا عمق کانال را قبل از اعمال کانولوشن 3×3 کاهش داده و پس از آن را ارتقاء ببخشیم.

دلایل سرعت بیشتر Resnet نسبت به vgg: علاوه بر تعداد کمتر پارامتر ها که اشاره شد، می دانیم سرعت کانولوشن به اندازه ورودی بستگی دارد. در دو لایه اول Resnet موفق می شود ارتفاع و عرض تصویر را تا 4 برابر کاهش دهد. Vgg در دو لایه اول convolution را در تصویر کامل با ابعاد بالا اعمال می‌کند که هزینه بر است و تعداد flops بالایی دارد. برای جلوگیری از این مشکل محاسباتی در Resnet، آنها در لایه اول به این موضوع می پردازند. به این صورت که تعداد سطرها و ستون ها را به میزان 2 کاهش می‌دهد و از flops کمتری استفاده می کند و عملیات max pooling نیز کاهش ابعاد دیگری را با ضریب 2 اعمال می کند.

فیلتر های Resnet در مقایسه با VGG از kernel های کمتری استفاده می کنند، اما تعداد بیشتری از آنها به طور متناوب بین عملیات کانولوشن و توابع فعال سازی غیرخطی روی هم چیده شده‌اند. آنها از ایده استفاده از شبکه های نازک تر اما عمیق تر استفاده کردند.

ایده کانولوشن 1×1 نیز به منظور کاهش بیشتر تعداد عملیات و در عین حال استفاده از شبکه هایی با تعداد فیلترهای بیشتر در لایه های کانولوشن مورد استفاده قرار گرفت. این نوع کانولوشن اجازه می دهد تا عمق کانال را قبل از اعمال کانولوشن 3×3 کاهش داده و پس از آن آن را ارتقاء بخشد.

منابع: لینک و لینک و لینک و لینک و لینک و لینک