پاسخ تمرین یازدهم **مهدیه نادری: ۹۸۵۲۲۰۷۶**

سوال اول) الف : مقایسه ی لایه ی کانوولوشنی با fully connected: (۳۰ دقیقه)

منبع: https://www.aparat.com/v/Pqw4I?playlist=1001309

در لایه های کاملا متصل مقدار هر نورون وابسته به مقدار تمام نورون های لایه ی قبل بود. کانوولوشن یک بعدی فرمولش خیلی مشابه لایه ی کاملا متصل است اما سیگما روی بخشی از نرون های ورودی است نه همه ی آنها. لوکال و محلی است و همه ی نورون ها ی قبلی به این نورون متصل نیستند.

بین نورون های یک لایه ی کانولوشن میتوانیم وزن مشترک داشته باشیم یعنی وزن یک نورون لایه یn به نورون لایه ی n+1 میتواند عینا با وزن بین دو نورون از همین لایه ها برابر باشد و این یعنی یکبار وزن محاسبه و آپدیت میشود و در جاهای مشترک استفاده خواهد شد اما در لایه ی کاملا متصل وزن ها کاملا مستقل بودند. در کانولوشن مثلا میتوان سه وزن را آموزش داد و بین همه ی نورون ها استفاده کرد. بخاطر همین اشتراک گداری وزن ها تعداد پارمتر هاش کمتر شده.

همین اشتراک گذاری وزن ها کمک میکنه شبکه دنبال فرمولی بگرده که با اعمالش روی تصویر مسئله ی مورد نظر حل بشه اما در fully connected هر نورون مستقلا دنبال یک فرمول دیگر میگردد برای همین کاری که مثلا لایه ی کانولوشنی با ۳۰۰۰ نورون میکند لایه ی fully connected با ۳۰۰۰۰۰۰۰۰ نورون میتواند انجام ندهد اما احتمال overfit آن هم خیلی بالا است.

لایه ی کانوولوشنی حتما نباید ورودیش بردار باشه و میتونه ماتریس باشه و معمولا یک ماتریس سه بعدیه. برعکس فولی کانکتد که حتما باید flatten میکردیم یعنی مثلا اگر ورودی ک تصویر بود باید به یک بردار تبدیل میشد.

خروجی لایه ی کانولوشنی هم میتواند یک ماتریس باشد که به آن نقشه ی فعالیت میگوییم. یک فیلتر روی این لایه فقط یک مسخصه از تصویر را استخراج میکند. در حقیقت خروجی با توجه به تعداد فیلتر ها میتواند چند نقشه ی فعالیت باشد.

۱- یادگیری الگوی دقیق و بینش عمیق از داده های ارائه شده. (بستگی به ساختار مناسب، تمیز یا مهندسی داده ها دارد) برای دستیابی به نتایج بهتر و دقیق می‌توان شبکه را تنظیم کرد. اگر بهتر تنظیم شود و مقدار کافی از داده ها را تغذیه کند، می‌تواند نتایج بهتری نسبت به الگوریتم های یادگیری ماشین دیگر ارائه دهد.

۲- اتصال محلی. هر نورون دیگر به همه سلول‌های عصبی موجود در لایه قبلی متصل نیست، بلکه فقط به تعداد کمی نورون متصل است. این باعث کاهش بسیاری از پارامترها می‌شود.

۳- تقسیم وزن. مجموعه ای از اتصالات می‌توانند وزن یکسانی داشته باشند، به جای داشتن وزن متفاوت برای هر اتصال، که باعث کاهش بسیاری از پارامترها می‌شود.

سوال اول) ب : محاسبه ی padding لازم و تعداد پارامتر لایه ی کانوولوشنی: (۲۰دقیقه)

با توجه به فاصله ی مرکز فیلتر از مرز های اطرافش خروجی لایه میتواند از ورودی آن کمتر باشد به همین دلیل برای هم اندازه بودن ورودی و خروجی باید در اطراف تصویر padding درنظر بگیریم.

در کل میتوانیم بدون padding با این ۱۶ فیلتر سایز خروجی را حساب کنیم و بعد به اندازه ی اختلافش با ابعاد تصویر اصلی padding درنظر بگیریم.

فرض: ۱۶ فیلتر ۵در۵در۵ داریم

Outputsize = N-F+1 = 16 – 5 + 1 = 12

پاسخ تمرین یازدهم

مهدیه نادری: ۹۸۵۲۲۰۷۶

هر فیلتر یک کانال در خروجی اضافه میکند پس در نهایت خروجی بدون padding میشود : ۱۲ در ۱۲ در ۱۶

حالا برای اینکه طول و عرض خروجی تغییر نکند چون 4 = 12-16 باید ۴ پیکسل به طول و عرض اضافه شود یعنی باید در ابتدا از هر سمت دو پیکسل padding درنظر بگیریم.

تعداد پارامتر ها = تعداد کانال خروجی × (تعداد وزن های لایه + ۱(بایاس)

= ۱۶ × (۵×۵×۵ + ۱) = ۲۰۱۶

اگر فیلتر ها را ۱۶ تا ۵ در ۵ در نظر بگیریم و عمق نداشته باشند (که درست نیست اما طبق صورت سوال فقط فرض کنیم) تعداد پارامتر ها میشود: ۱۶ × (۵×۵ + ۱) = ۴۱۶

سوال اول) پ : (۲۰دقیقه)

قسمت اول: فیلتر ۵ در ۵ از هر طرف ۲ پیکسل کم میکنه در نتیجه 28 = 4-32 ابعاد خروجی: ۲۸ در ۲۸ در ۳ (چون ۳ تا فیلتر داریم)

قسمت دوم: خروجی لایه ی اول: از هر طرف یک پیکسل کم میشود در نتیجه 30 = 2 – 32. ابعاد خروجی: ۳۰ در ۳۰ در ۹(۹ فیلتر)

خروجی لایه ی دوم: 28 = 2-30 ابعاد خروجی: ۲۸ در ۲۸ در ۹ (۹ فیلتر داریم)

سوال اول) ت: مقایسه ی max(min) pooling، average pooling و global average pooling : (۴۵دقیقه)

منبع: https://blog.faradars.org/convolutional-neural-networks/

https://blog.class.vision/1397/10/page/2/

https://www.aparat.com/v/TDK5y?playlist=1001309

کارکرد این Pooling کاهش اندازه مکانی (عرض و ارتفاع) تصویر (ورودی) بجهت کاهش تعداد پارامترها و محاسبات در داخل شبکه و بنابر این کنترل overfitting است . لایه Pooling بصورت مستقل بر روی هر برش عمقی از توده ورودی عمل کرده و آنرا با استفاده از عملیات MAXاز لحاظ مکانی تغییر اندازه (resize) میدهد.

Max(min) Pooling:

در این نوع ادغام بزرگ‌ترین مقدار در ناحیه‌ای را که فیلتر پوشانده است انتخاب می‌شود؛ بنابراین در این حالت خروجی یک نقشه‌ی ویژگی (Feature Map)است که برجسته‌ترین ویژگی‌های نقشه ویژگی (Feature Map) قبلی را دارد.

Max Poolingکار «حذف نویز» (Noise Suppressant) را نیز انجام می‌دهد. این تجمع(Pooling) ، همه فعال‌سازهای (Activations)نویزی را همزمان رها می‌کند و همچنین، کار کاهش ابعاد را همراه با حذف نویز انجام می‌دهد.

Average pooling:

در این نوع ادغام مقدار میانگین ناحیه‌ای که فیلتر روی آن قرار می‌گیرد محاسبه می‌شود؛ بنابراین میانگین ویژگی‌های نقشه‌ی ویژگی قبلی را در خروجی ارائه می‌کند.

Average Poolingکار کاهش ابعاد را به عنوان مکانیزمی برای حذف نویز اجرا می‌کند. بنابراین شاید بتوان گفت که Max Pooling خیلی بهتر از Average Pooling است.

ResNet50 به صورت غیر مستقیم از average pooling استفاده کرده. به این صورت که لایه pooling آخرش اندازه خروجی (2048 * 1* 1) شده است ولی از fully connected ها هم برای کلاس بندی انتهایی استفاده کرده است.

Global average pooling:

برای حل مشکل تعداد وزن های لایه flat وfully connected از این لایه ایتفاده میشود. به جای fully connected ها در انتها global average poolingمیزنیم تا future map ها را مستقیم به softmax وصل کنیم و از fully connected استفاده نکنیم. در شبکه ی عمیقی مثل googlenet که شبکه ی عمیقی با ۲۲ لایه است و در آن تعداد زیادی ماژول inception وجود دارد، در شبکه ای مثل vgg-net بیشترین پارامتر های شبکه جایی بود که بعد از چند لایه ی کانولوشنی لایه ی dence قرار میدادیم و بعد flatten میگذاشتیم و ماتریس را به بردار تبدیل میکردیم، در واقع global average pooling میگه نیازی به flatten نیست. برای کاهش تعداد بعد میتوان کار ساده تری کرد. میانگین هر کدام از ویژگی ها را حساب میکنیم. یعنی مثلا اگر خروجی قبل از لایه ی flatten دارای ابعاد ۷ در ۷ در ۵۱۲ باشد به معنی ۵۱۲ ویژگی در ۴۹ مکان مختلف، به ازای هر ویژگی میانگین آن در کل تصویر را حساب میکنیم. در واقع خروجی لایه ی global average pooling میشود ۱در ۱ در Depth . این لایه جایگزین flatten است اما تفاوتش این است که پارامتر مکان را از دست میدهیم و نمیدانیم هر ویژگی دقیقا در کدام پیکسل هاست اما میدانیم آن ویژگی در تصویر موجود است و این از دست دادن جزئیات از معایب آن است اما مزایای آن این است که پارامتر ها را به اندازه ی width در height برابر کاهش میدهد و خیلی خوب است که این ایده در googlenet استفاده شده است.

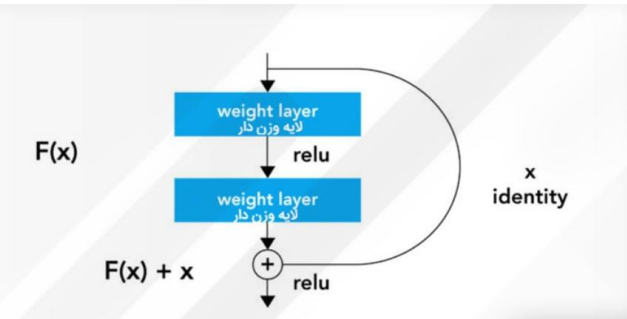
سوال اول) ث: مقایسه بین Resnet و VGG-net :(۲۰دقیقه)

منبع: https://www.aparat.com/v/2UYMp?playlist=1001309

ResNet: ایده ی اصلی این شبکه افزایش عمق شبکه بود. تا۱۵۲ لایه میتواند داشته باشد. مشکلی که وجود داشت این بود که وقتی تعداد لایه های کانوولوشنی ساده را بیشتر میکردند عملکرد شبکه بدتر میشد هم روی داده های آموزشی و هم تست. مشکل overfitting نبود. مشکل از بهینه سازی بود دیگر کاهش گرادیان جواب نمیداد. چون گرادیان به لایه های اول نمیرسید. ایدش اینه که حتی اگر شبکه نتونست خوب ترین بشه و فرمول مورد نظر را یادبگیره بتونه فرمول همانی را یاد بگیره. با این کار مشکل انفجار یا ناپدید شدن گرادیان را تا حد خوبی حل میکند. برای همین یک اتصال اضافه از ورودی به خروجی داره و ورودی را با خروجی جمع میکند. خیلی شبیه vgg هست و از آن الهام گرفته است فقط بین هر چند لایه این اتصال residual اضافه شده تا خروجی مناسب تر باشد. در ابتدای آن هم یک لایه ی کانوولوشنی داریم.

همانطور که در شکل مشخص است فرق این شبکه با شبکه‌های معمولی این است که یک اتصال میان‌بر دارد که از یک یا چند لایه عبور می‌کند و آن‌ها را در نظر نمی‌گیرد؛ درواقع به‌نوعی میان‌بر می‌زند و یک لایه را به لایه‌ی دورتر متصل می‌کند.

یک مشکل وجود دارد؛ از آنجا که در طول شبکه‌ی خروجی لایه‌های مختلف کانولوشن ابعاد مختلفی را دارد، بنابراین ممکن است مقدار x که اتصال میان‌بر از لایه‌های قبلی به f(x) اضافه می‌کند ابعاد متفاوتی را رقم بزند. برای حل این مشکل میتوانیم ابعاد اتصال میان‌بر را با استفاده از فرایند لایه‌گذاری با صفر (Zero padding) افزایش دهیم و از لایه‌های کانولوشن 1*1 برای کاهش ابعاد ورودی استفاده می‌کنیم



پاسخ تمرین یازدهم **مهدیه نادری: ۹۸۵۲۲۰۷۶**

VGG-Net:حداکثر ۱۹ لایه و ۱۴۰ میلیون پارامتر دارد. VGGNet، اغلب کانولوشنی و سپس لایه‌های کاملاً متصل Fully) (Connectedبرای طبقه‌بندی دارد، بدون هیچ‌گونه اتصال میان‌بر. ما در اینجا آن‌ها را شبکه‌های ساده (Plain Networks) می‌نامیم. وقتی شبکه‌ی ساده (Plain Networks) عمیق‌تر هستند (یعنی لایه‌ها افزایش می‌یابند)، مشکل محوشدگی گرادیان Vanishing) (Gradientیا انفجار گرادیان (Exploding Gradient) رخ می‌دهد؛ بنابراین عمیق‌ترکردن شبکه کار راحتی محسوب نمی‌شد که تنها با اضافه‌کردن لایه به شبکه آن را عمیق‌تر کنیم. در این مدل 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. ایده ی اصلیش هم استفاده از فیلتر های کوچکتر و شبکه ی عمیق تر است. به عنوان مثال تمام فیلتر ها ۳ در۳ درنظر گرفته میشود. به عنوان مثال در VGG دو فیلتر پشت سرهم سه در سه روی یک تصویر معادل اعمال یک فیلتر ۵ در ۵ است. اما مزیت دو لایه شدن این عملیات در VGG این است که میتوان بعد از اعمال هر فیلتر یک تابع غیر خطی استفاده کرد. درنتیجه حاصل آن هم معادل فیلتر ۵ در ۵ غیرخطی است. تعداد پارامتر ها هم کمتر شده است یعنی در همین مثال به جای ۵*۵ = ۲۵، تاثیر ۲*۳*۳ = ۱۸ در تعداد کل پارامترها دارد. بعد از هر pooling سعی میکنه تعداد ویژگی ها رو افزایش بده.

هر قدر شبکه عمیق‌تر می‌شود، پیچیدگی زمانی نیز افزایش می‌یابد که برای حل آن یک گلوگاه (Bottleneck) طراحی شده است. راه‌حل به این شکل بود که به اول و آخر هر لایه کانولوشن یک لایه‌ی کانولوشن ۱×۱ اضافه شد. تکنیک کانولوشن ۱×۱ در شبکه‌ی گوگل‌نت (GoogleNet)استفاده شده است و نشان می‌دهد کانولوشن‌های ۱×۱ می‌توانند تعداد پارامترهای شبکه را کاهش دهند و درعین‌حال کارایی آن را کاهش ندهند. با طراحی این گلوگاه رزنت ۳۴ لایه به ۵۰، ۱۱۰ و ۱۵۲ نیز افزایش یافت.