بسمه تعالی

دانشکده ی مهندسی برق

درس یادگیری عمیق

دکتر فاطمی زاده

تمرینات سری سوم

نویسنده : حسین رمضانی یادگار

رشته و گرایش : مهندسی برق - سیستم های دیجیتال

مقطع : کارشناسی ارشد

شماره دانشجویی : 401205767

سوال 1 :

از انجایی که تعداد پارامتر ها یکسان است باید به دیگر ویژگی های این دو معماری روی بیاوریم. معماری fully connected به تک تک پیکسل ها بعنوان یک فیچر نگاه میکند و سعی میکند وزن های لایه را با توجه به تکنیک EBP که از خروجی شبکه روی تابع loss محاسبه میشود، بهینه کند. پس نورون ها به همه ی فیچر ها یا پیکسل ها برای استخراج اطلاعات نگاه میکنند. در حالی که در لایه ی convolutional هر فیلتر مقادیر وزن و بایاس خودش را طوری بهینه میکند (در اینجا هم ebp داریم) که مسئول استخراج یک الگو مربوط به یک کلاس شود. پس لایه ی convolutional تناسب بیشتری با تسک دسته بندی تصویر دارد.

همچنین CNN از انجایی که فیلتر هایش کل تصویر را پوشش میدهد، در فیچر مپی که استخراج میشود اگر الگوی مربوط به کلاس x در تصویر جابه جا شود، تشخیص و دسته بندی دچار خطا نخواهد شد ولی همانطور که گفته شد شبکه های FC، به تک تک پیکسل ها بعنوان یک فیچر خاص توجه میکند و با تغییر در مقدار این فیچر ها (بعنوان مثال شیفت خوردن تصویر) در تشخیص دسته بندی دچار خطا خواهد شد.

سوال 2 :

الف) طبق زیر محاسبه میشود :

$$Outputsize = \left\lfloor \frac{inputsize + 2 * padding - kernelsize}{stride} \right\rfloor + 1$$

$$outputsize = \left\lfloor \frac{128 + 2 * 2 - 5}{1} \right\rfloor + 1 = 128 \rightarrow \text{size} = 128 * 128$$

با 16 فیلتر تسنور خروجی این لایه به شکل 16*128*128 در می آید.

تعداد پارامتر ها بشکل زیر محاسبه میشود :

$w * h : kernel\ size, d : number\ of\ channels\ in\ the\ perevius\ layer$

$k : the\ number\ filters\ in\ the\ current\ layer$

$number\ of\ parameters = (w * h * d + 1) * k$

$number\ of\ parameters = (5 * 5 * 3 + 1) * 16 = 1216$

ب) تعداد پارامتر ها توسط تنها لایه های کانولوشنی تعیین میشوند و ربطی به لایه ی pooling ندارد:

$number\ of\ parameters = (w * h * d + 1) * k$

$Layer1 = (5 * 5 * 3 + 1) * 16 = 1216$

$Layer2 = (5 * 5 * 16 + 1) * 16 = 6416$

$$Layer3 = (5 * 5 * 16 + 1) * 16 = 6416$$

$$total\ number\ of\ parameters\ =\ Layer1\ +\ Layer2\ + Layer3\ =\ 14{,}048$$

تعیین سایز تنسور خروجی :

$$CSize\ = \left\lfloor \frac{InputSize\ +\ 2 * padding\ -\ KernelSize}{Stride} \right\rfloor + 1$$

$$PoolSize\ = \frac{InputSize\ -\ PoolSize}{Stride}\ +\ 1$$

Layer 1 :

$$CSize\ = \left\lfloor \frac{128\ +\ 2 * 2\ -\ 5}{1} \right\rfloor + 1\ =\ 128 * 128 * 16$$

$$PoolSize\ = \frac{128\ -\ 2}{2}\ +\ 1\ =\ 64\ *\ 64 * 16$$

Layer 2 :

$$CSize\ = \left\lfloor \frac{64\ +\ 2 * 2\ -\ 5}{1} \right\rfloor + 1\ =\ 64 * 64 * 16$$

$$PoolSize\ = \frac{64\ -\ 2}{2}\ +\ 1\ =\ 32 * 32 * 16$$

Layer 3 :

$$CSize\ = \left\lfloor \frac{32\ +\ 2 * 2\ -\ 5}{1} \right\rfloor + 1\ =\ 32 * 32 * 16$$

$$PoolSize\ = \frac{32\ -\ 2}{2}\ +\ 1\ =\ 16 * 16 * 16$$

پس تنسور خروجی به شکل 16*16*16 میباشد. این تنسور flatten شده و به شبکه ی classifier پاس داده میشود

ج) با فرض معماری 10-4096-4096 تعداد پارامتر ها بصورت زیر محاسبه میشود:

Total Trainable Parameters Between two layers in a Neural Network = [Number of Nodes in the first layer * Number of nodes in the second layer] (Weights) + [Number of nodes in the second layer] (Biases)

$$nn\_p\ =\ (4096 + 1) * 4096)\ +\ (4096 + 1) * 10\ =\ 16{,}822{,}282$$

$$total\_number\_of\_parameters\ =\ nn\_p\ +\ 14{,}048\ = 16{,}836{,}330$$

د)

سوال 3 :

1) مفهومی U-net :

a. معماری این شـبکه مبتنی بر معماری های عادی اسـت و تلاش شـده با تغییراتی که در معماری میدهند با حجم دیتاسـت کمتر به دقت بالاتر در تسـک segmentation برسـند. تفاوت های اصلی skip connection هایی است که بین مسیرencoder و decoder اتفاق میفتد بعلاوه نوع اپراتور استفاده شده در این skip con ها concatenation هست. لایه های کانولوشنی در مسیر encoder وdecoder طوری تنظیم شده است که تنسور ورودی ابتدا فیچر چنل کم و در انتهای مسـیر encoder ، زیاد دارد و دقیقا برعکس همین معماری در مسـیر decoder با اپراتور های مکمل انجام شـده اسـت یعنی در ابتدا ی مسـیر تنسـور دارای تعداد چنل زیاد و در انتهای مسیر و خروجی تعداد چنل کمتر شـده است .این تقارن شکل گرفته باعث شـده معماری کلی مدل به شکل U در آید.

b. بخاطر اینکه در هر محله ی down sampling و up sampling اطالعات تصویر از دست میرود باید بطریقی اطلاعات تصـویر ورودی و رزولوشـن آن را حفظ کرد. با اسـتفاده از skip connection اطلاعات مسـیر encoder در مرحله ی متناظر خودش در مسـیر decoder بصورت concatenation به تنسور خروجی اضافه میشود.

c. از ان جایی که در تصـاویر پزشـکی اگر تسـک segmentation تعریف میشـود دقت در حدود پیکسل مد نظر است این نوع اتصالات میتواند اطالعاتی که در هر مرحله ی کانولوشنی استخراج شـده و در مراحل down sampling نادیده گرفته شـده را بازیابی کند و در خروجی از فیچر های رسیده از مسیر encoder نیز استفاده شود.

2) محاسبه ای U-net

a. چون در معماری U-net چهار لایه ی در مسیر encoder و decoder وجود دارد ابعاد ورودی تقسیم بر 2^4 خواهد شد پس ابعاد فیچر مپ بصورت 16*16 در می آید.

b. تعداد فیلتر های بیشتر در لایه های عمیق شبکه استفاده میشود پس در لایه ی دوم از ورودی و encoder، از 128 فیتلر باید اسـتفاده شـود و این به این معنی اسـت که در ورودی 64 فیلتر مورد اسـتفاده قرار گرفته اسـت همچنین در هر لایه 2 بار conv اتفاق میفتد پس طبق روابط تعداد پارامتر های لایه های کانولوشنی داریم :

$$n1 = (w * h * d + 1) * k \xrightarrow[w*h=3*3]{} (3 * 3 * 64 + 1) * 128 = 73{,}856$$

$$n2 = (3 * 3 * 128 + 1) * 128 = 147{,}584$$

تعداد کل پارامتر های لایه ی 2 در مسیر *encoder* بصورت زیر محاسبه میشود :

$$totalNumberOfParameters = n1 + n2 = 221{,}440$$

3) مفهومی DenseNet

a. تفاوت اصـلی اتصـالات denseNet و resNet این اسـت که در شـبکه ی denseNet اولا اتصالات از نوع concatenation هست و دوما هر لایه تا یک تعداد محدودی از خروجی های لایه های قبلی بعنوان ورودی اسـتفاده میکند در حالی که در resNet هر لایه فقط از خروجی

لایه ی قبل استفاده میکند و اتصال آن هم از نوع جمع شونده است یعنی تنسور خروجی لایه ی قبل با تنسور خروجی لایه ی فعلی که از توابع لایه ی کانولوشنی فعلی با تنسور خروجی لایه ی قبلی بدست امده جمع میشود. توصیف ریاضی دو نوع شبکه بصورت زیر است :

$ResNet: X_l = H(X_{l-1})$

$DenseNet: X_l = H([X_0, X_1, \dots, X_{l-1}]$

H functionality : Batch Normalization , ReLU , 3*3 Conv

b. در این شبکه ها هدف اصلی این بوده که با مشکل gradient vanishing مقابله شود. تا حد خیلی خوبی هم نتیجه گرفته شده و لایه های عمیق به حدود 250 تا نیز رسیده است. دلیل حل این مشکل نیز وجود skip connection هایی هست که هر لایه با لایه های قبل دارد یعنی در فرایند EBP اینک، انتشار گرادیان از مسیر های مختلفی صورت میپذیرد و از صفر شدن زود هنگام آن در نتیجه ی ضرب شدن وزن های لایه های مختلف جلو گیری به عمل می آید. مزیت محاسباتی آن این است که از استخراج ویژگی های مشابه به واسطه skip conn ها بی نیاز خواهیم شد.

4) محاسبه ای DenseNet

a. فیچر مپ ورودی لایه سوم از *concatenation* فیچر مپ های لایه های قبل محاسبه میشود :

$$L_3 FeatureMaps = k_0 + k * (l - 1)$$

فرمول بالا در صورتی است که هر لایه k فیچر مپ تولید کند پس انگار فیچر مپ های هر لایه را باهم جمع میکنیم : با فرض تعداد کانال ورودی k0=3

$$L_3 FeatureMaps = k_0 + 64 + 128 = 195$$

b. مثل بالا محاسبه میشود با این تفاوت که هر لایه به تعداد k فیچر مپ تولید میکند :

$$L_3 OutputFeatureMaps = k_0 + k * l = 32 + 24 * 3 = 104$$

سوالات تئوری و گزارشات مربوط به بخش عملی :

سوال عملی 2 :

1) از لحاظ grid sampling در شبکه های سنتی conv ابعاد و شکل کرنل و stride یکسان و فیکس و منظم است در نتیجه ی آن نمیتواند فیچر های پیچیده و الگو های خاص که transform های deformable خورده اند و یا اشکالی که با حرکت کرنل aligned نیستند، را تشخیص دهند. اما در شبکه های deformable conv ابعاد و شکل کرنل مکان سمپل برداری بصورت flexible و adaptive متناسب با ورودی تنظیم میشود در نتیجه فیچر و الگوی پیچیده تری را میتوانند استخراج کنند. درنتیجه شبکه های deformable نتایج بهتری روی تسک های دسته بندی و segmentation گرفته اند.

2) در شبکه های def از مقادیر offset برای موقعیت و اشکال هر کرنل استفاده میشود که قابل یادگیری هستند. به کمک این افستها و کرنل مورد استفاده میتوانند از فیچر های محلی هر عکس سمپل برداری کنند مثلا اگر تصویر اسکیل شده است مقدار افست سایز کرنل را طوری تنظیم میکند که شیئ اسکیل شده اسکن شود.
3) شبکه های کانولوشنی سنتی از موقعیت های منظمی برای سمپل برداری استفاده میکنند و همچنین شکل کرنل نیز فیکس در نظر گرفته میشود. وقتی یک شکل بزرگ تر میشود نمیتوان با کرنل ها و حرکت آن ها این تشخیص را انجام داد چون اصلا در قالب یک کرنل ممکن است شکل شیئ مورد نظر گنجانیده نشود.
4) طی فرایند یادگیری این پارامتر ها طبق یک لاس فانکشن و مفهوم EBP توسط شبکه یاد گرفته میشوند.