به نام خدا

یادگیری عمیق، تکلیف دوم

مهدی کافی ۹۹۲۱۰۷۵۳

مسئله‌ی ۱. Network Design

(آ) *فرض کنید که شبکه ای با ساختار زیر داریم، خروجی هر لایه و تعداد پارامترهای آن را محاسبه کنید.*

$$Input = ۲۲۴ * ۲۲۴ * ۳ \rightarrow ۱$$
$$۱ \rightarrow Conv[۳ * ۳ * ۶۴, stride = ۱, pad = ۱] \rightarrow ۲$$
$$۲ \rightarrow ReLU \rightarrow ۳$$
$$۳ \rightarrow Pooling[۲ * ۲ * ۳۲, stride = ۲] \rightarrow ۴$$
$$۴ \rightarrow FC[۱۰ - class] \rightarrow ۵$$

اعداد ۱ تا ۴ برای مشخص کردن خروجی و ورودی های لایه ها استفاده شده اند.

(ب) *درصورتیکه بخواهیم فقط با یک لایه ی تمام متصل (FC) تصویری با ابعاد ورودی و خروجی مشابه داشته باشیم، تعداد پارامترهای لایه ی تمام متصل را محاسبه کنید*

(ج) *تفاوت* Deformable Convolution *با* Standard Convolution *در چیست و چه مزایایی نسبت به حالت استاندارد دارد؟*

الف) می‌دانیم که اگر تصویری با ابعاد $(h * w * c)$ داشته‌باشیم و کرنلی با ابعاد (k_1, k_2, f) و با گام s و پدینگ با p بر روی آن اعمال کنیم. ابعاد خروجی به صورت زیر خواهدبود.

$$\left(\lfloor \frac{h - k_1 + 2p}{s} \rfloor + 1, \lfloor \frac{w - k_2 + 2p}{s} \rfloor + 1, f \right)$$

و اگر لایه max pooling با ابعاد (k_1, k_2) با گام s داشته‌باشیم. ابعاد خروجی به صورت زیر خواهدبود.

$$\left(\lfloor \frac{w - k_1}{s} \rfloor + 1, \lfloor \frac{h - k_2}{s} \rfloor + 1, c \right)$$

بنابراین خروجی لایه اول به صورت زیر خواهدبود.

$$1 \rightarrow \left(\lfloor \frac{224 - 3 + 2 \times 1}{1} \rfloor + 1, \lfloor \frac{224 - 3 + 2 \times 1}{1} \rfloor + 1, 64 \right) = (224, 224, 64)$$

خروجی لایه دوم و سوم نیز به صورت زیر محاسبه می‌شود.

$$2 \rightarrow (224, 224, 64)$$

$$3 \rightarrow (112, 112, 64)$$

خروجی لایه آخر نیز یک وکتور به ابعاد روبرو خواهدبود.

$4 \rightarrow (10,\ 1)$

برای محاسبه تعداد پارامترها نیز به صورت زیر عمل می‌کنیم که یک لایه convolutional با ابعاد (k_1, k_2, f) تعداد $k_1 * k_2 * f$ پارامتر خواهدداشت و همینطور هنگامیکه می‌خواهیم تصویر را به لایه Fully Connected ببریم نیاز داریم ماتریس وزنی با ابعاد $(fc, h * w *$ $c)$ به علاوه تعداد fc تا bias داشته‌باشیم. بنابراین تعداد پارامترهای هر لایه نیز به صورت زیر محاسبه می‌شود.

$$1 \rightarrow 3 \times 3 \times 64 = 576$$

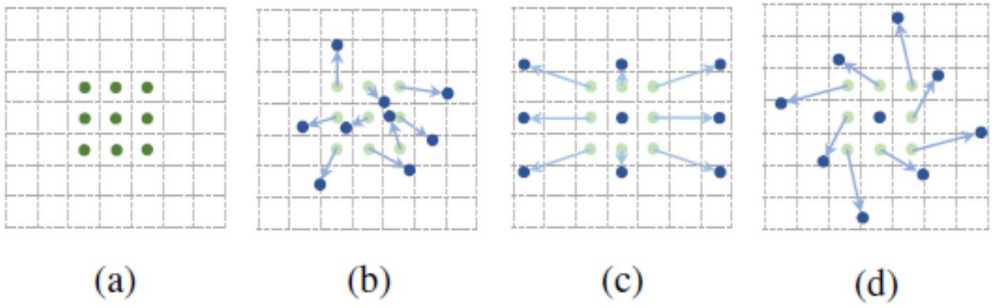
$$2 \rightarrow \text{no parameters}$$

$$3 \rightarrow \text{no parameters}$$

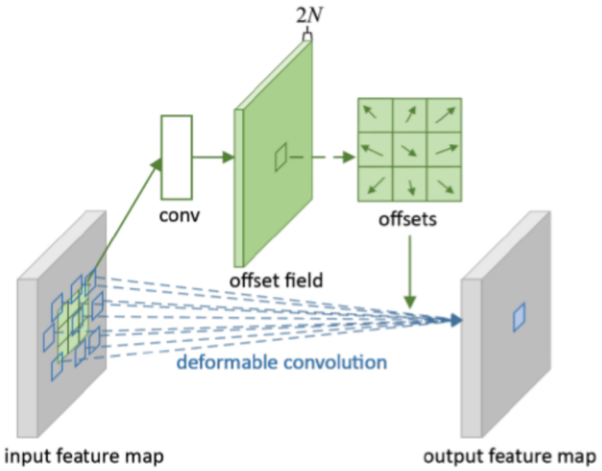
$$4 \rightarrow (112 \times 112 \times 64 \times 10) + 10 = 8028170$$

ب) در صورتیکه بخواهیم با لایه تمام متصل تصویر ورودی را به خروجی مذکور ببریم به $(224 * 224 * 3 * 10) + 10 = 1505290$ پارامتر نیاز خواهیم‌داشت.

ج) هنگام استفاده از شبکه‌های convolutional مشکلی داریم و آن این است که ممکن است تصاویر نگاشت‌های هندسی داشته‌باشند نسبت به حالت استاندارد خودشان و برای مواجهه با این مشکل می‌توانیم به طور مثال داده‌ها را Augment کنیم و این تغییرات هندسی را تا حد ممکن روی داده‌ها ایجاد کنیم. مشکل این روش این است که این تغییراتی که ایجاد می‌کنیم محدود و ثابت هستند و تعمیم پذیری کمی دارند. حال می‌توانیم از Deformable Convolutionها استفاده کنیم. این لایه‌های کانولوشنی به این صورت عمل می‌کنند که لزوما نقاط کنار هم در یک کرنل را نگاه نمی‌کنند مانند Standard Convolutionها بلکه می‌توانند نقاط را با تغییرات هندسی و یا بدون قاعده خاصی نگاه کنند و این تغییر در اعمال نقاط کرنل به صورت Adaptive صورت می‌گیرد. نشان داده شده است که این روش پارامتر و سربار محاسباتی کمی تولید می‌کند و در عوض دقت شبکه را به نسبت قابل توجهی افزایش می‌دهد.



(a) Conventional Convolution, (b) Deformable Convolution, (c) Special Case of Deformable Convolution with Scaling, (d) Special Case of Deformable Convolution with Rotation



Regular convolution

$$\mathbf{y}(\mathbf{p}_0) = \sum_{\mathbf{p}_n \in \mathcal{R}} \mathbf{w}(\mathbf{p}_n) \cdot \mathbf{x}(\mathbf{p}_0 + \mathbf{p}_n)$$

Deformable convolution

$$\mathbf{y}(\mathbf{p}_0) = \sum_{\mathbf{p}_n \in \mathcal{R}} \mathbf{w}(\mathbf{p}_n) \cdot \mathbf{x}(\mathbf{p}_0 + \mathbf{p}_n + \Delta\mathbf{p}_n)$$

where $\Delta\mathbf{p}_n$ is generated by a sibling branch of regular convolution

Deformable Convolution

به طور مثال برای Object Detection اگر object مورد نظر کوچک و یا بزرگ باشد به طور مثال یک ماشین در فاصله‌ای دور و یا یکی اتوبوس بزرگ در فاصله‌ای نزدیک، این روش به راحتی آن‌ها را شناسایی می‌کند.

Examples: three levels of 3×3 deformable filters for three activation units (green points) on the background (left), a small object (middle), and a large object (right)

مسئله‌ی ۲. Transposed Convolution

کانولوشن بین ورودی X و فیلتر W به صورت زیر است:

$$X = \begin{bmatrix} x_{(۰,۰)} & x_{(۰,۱)} & x_{(۰,۲)} \\ x_{(۱,۰)} & x_{(۱,۱)} & x_{(۱,۲)} \\ x_{(۲,۰)} & x_{(۲,۱)} & x_{(۲,۲)} \end{bmatrix} \quad W = \begin{bmatrix} w_{(۰,۰)} & w_{(۰,۱)} \\ w_{(۱,۰)} & w_{(۱,۱)} \end{bmatrix}$$

می توان عملیات کانولوشن را به صورت ضرب ماتریسی نوشت که ورودی و خروجی را به صورت یک بردار و فیلتر را به صورت یک ماتریس در نظر گرفت. ورودی X را به صورت بردار زیر نمایش می دهیم:

$$X = \begin{bmatrix} x_{(۰,۰)} & x_{(۰,۱)} & \dots & x_{(۲,۲)} \end{bmatrix}^T$$

(آ) عملیات کانولوشن بالا با $S = ۱$ را به صورت ضرب ماتریسی $Y = AX$ بنویسید.

(ب) با استفاده از نمایش ماتریسی بالا می توان گرادیان پس انتشار نسبت به ورودی X، را به صورت $\frac{\partial L}{\partial X} = A^T \frac{\partial L}{\partial Y}$ نمایش داد.

عملیات transposed convolution را می توان مشابه عملیات گرادیان پس انتشار نسبت به ورودی درنظر گرفت و می توان این عملیات را به صورت کانولوشن مستقیم با ماتریس A^T به عنوان فیلتر دانست.

فرض کنید X خروجی یک عملیات کانولوشن با فیلتر W و $\text{stride} = 2$ است.

خروجی transposed convolution را با ورودی های زیر محاسبه کنید.

$$X = \begin{bmatrix} ۲ & ۱ \\ ۳ & ۶ \end{bmatrix} \quad W = \begin{bmatrix} ۱ & ۲ \\ ۴ & ۱ \end{bmatrix}$$

آ)برای انجام کانولوشن به صورت ضرب ماتریسی، به صورت زیر عمل می‌کنیم.

$$\begin{bmatrix} w_{(0,0)} & w_{(0,1)} & 0 & w_{(1,0)} & w_{(1,1)} & 0 & 0 & 0 & 0 \\ 0 & w_{(0,0)} & w_{(0,1)} & 0 & w_{(1,0)} & w_{(1,1)} & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{(0,0)} & w_{(0,1)} & 0 & w_{(1,0)} & w_{(1,1)} & 0 \\ 0 & 0 & 0 & 0 & w_{(0,0)} & w_{(0,1)} & 0 & w_{(1,0)} & w_{(1,1)} \end{bmatrix} \times \begin{bmatrix} x_{(0,0)} \\ x_{(0,1)} \\ x_{(0,2)} \\ x_{(1,0)} \\ x_{(1,1)} \\ x_{(1,2)} \\ x_{(2,0)} \\ x_{(2,1)} \\ x_{(2,2)} \end{bmatrix}$$

$$= \begin{bmatrix} w_{(0,0)}x_{(0,0)} + w_{(0,1)}x_{(0,1)} + w_{(1,0)}x_{(1,0)} + w_{(1,1)}x_{(1,1)} \\ w_{(0,0)}x_{(0,1)} + w_{(0,1)}x_{(0,2)} + w_{(1,0)}x_{(1,1)} + w_{(1,1)}x_{(1,2)} \\ w_{(0,0)}x_{(1,0)} + w_{(0,1)}x_{(1,1)} + w_{(1,0)}x_{(2,0)} + w_{(1,1)}x_{(2,1)} \\ w_{(0,0)}x_{(1,1)} + w_{(0,1)}x_{(1,2)} + w_{(1,0)}x_{(2,1)} + w_{(1,1)}x_{(2,2)} \end{bmatrix}$$

ب) ماتریس W که در ورودی اولیه ضرب شده‌است و خروجی X را تولید کرده‌است به صورت زیر بوده‌است.

$$W = \begin{bmatrix} 1 & 2 & 0 & 0 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 & 0 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 0 & 0 & 4 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 0 & 0 & 4 & 1 \end{bmatrix}$$

بنابراین برای عملیات transposed convolution که منجر به up-sampling می‌شود، نیاز داریم که این W^T را در بردار حاصل از ماتریس خروجی X به صورت زیر ضرب کنیم.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 4 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 2 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 2 \\ 1 \\ 3 \\ 6 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 1 \\ 2 \\ 8 \\ 2 \\ 4 \\ 1 \\ 3 \\ 6 \\ 6 \\ 12 \\ 12 \\ 3 \\ 24 \\ 6 \end{bmatrix} \rightarrow \text{up-sampled} = \begin{bmatrix} 2 & 4 & 1 & 2 \\ 8 & 2 & 4 & 1 \\ 3 & 6 & 6 & 12 \\ 12 & 3 & 24 & 6 \end{bmatrix}$$

مسئله‌ی ۳. Object Detection

یکی از کاربرد های شبکه های کانولوشن، آشکارسازی اشیا است. مقالات مرتبط را مطالعه کرده و به سوالات زیر پاسخ دهید.

(آ) *روش های* YOLO *و* RCNN *و* RFCN *را بررسی کرده و با یکدیگر مقایسه کنید.*

(ب) *در آشکارسازی های* real-time *استفاده از کدام یک را پیشنهاد می کنید.*

(ج) *لایه* Spatial Pyramid Pooling *را مختصرا توضیح داده و مزایای استفاده از آن را، در آشکارسازی تصاویر بیان کنید.*

آ) روش RCNN در ابتدا ۲۰۰۰ مکان از تصویر را انتخاب می‌کند و سپس مراحل بعدی را فقط با این ۲۰۰۰ مکان ادامه می‌دهد و نه تعداد زیادی مکان در تصویر. برای انتخاب ۲۰۰۰ مکان در ابتدا مکان‌هایی را انتخاب می‌کند و سپس این مکان‌ها را با یکدیگر ترکیب می‌کند تا در نهایت به ۲۰۰۰ مکان مورد نظر برسد و سپس این مکان‌ها را برای کلاس بندی به شبکه Convolutional وارد می‌کند. این روش نسبت به روش‌های غیر کانولوشنی مزیت‌های بسیاری دارد اما برای کاربردهای real-time بسیار کند است و همینطور گذاشتن عدد ثابت ۲۰۰۰ برای تمام کاربردها مناسب نخواهدبود.

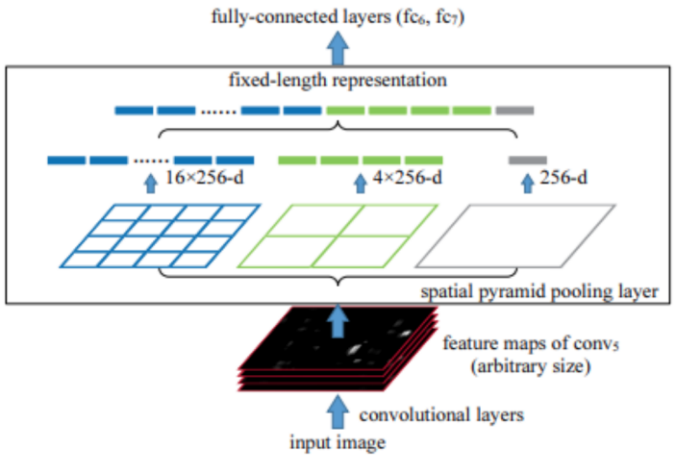
برخلاف روش قبلی به هر مکان یک بخش از شبکه را صدها بار اعمال می‌کند، روش RFCN، روشی دقیق و بهینه است که محاسبات را بر روی تمام تصویر پخش می‌کند. این روش یک جدول امتیاز برای مکان‌ها ایجاد می‌کند و برای محاسبه این امتیاز یک trade-off بین مستقل از نگاشت در کلاس بندی تصویر و وابسته به نگاشت در پیدا کردن اشیا استفاده می‌کند.

روش You Only Look Once یا به اختصار YOLO به این صورت عمل می‌کند که در هنگام تست به تمام تصویر به یکباره نگاه می‌کند و تخمین آن بر اساس مفهوم کلی تصویر است و همینطور برخلاف مدل‌هایی نظیر RCNN این مدل تخمین را با یک شبکه انجام می‌دهد. این

روش یک grid بر روی تصویر اعمال می‌کند و اگر مرکز یک شی در یکی از خانه‌های grid بیافتد. آن خانه مسئول تشخیص آن شی می‌شود. هر خانه grid دارای ۵ bounding box است و برای این bounding boxها امتیاز اطمینان محاسبه می‌کند این امتیاز میزان اطمینان از وجود یک شی در bounding box را مشخص می‌کند. برای هر bounding box یک کلاس نیز تخمین می‌زند. ترکیب امتیاز اطمینان و کلاس، احتمال وجود یک شی از کلاس خاص را در bounding box مشخص می‌کند.

ب) برای کابردهای real-time معمولا روش YOLO ترجیح داده‌می‌شود.

ج) هنگام استفاده از شبکه‌های Covolutional معمولا به این صورت عمل می‌کنیم که پس از چند لایه Convolutional ویژگی‌های استخراج شده را در قالب یک بردار در می‌آوریم و به لایه Fully Connected می‌دهیم که این وکتور می‌تواند سایزهای متفاوتی داشته‌باشد. روش Spatial Pyramid Pooling به این صورت عمل می‌کند که در سطوح متفاوتی عمل Pooling را انجام می‌دهد. به طور مثال در تصویر زیر سه سطح Pooling داریم که در سطح اول که خاکستری است فقط یک bin داریم که از کل تصویر در لایه‌های متفاوت آن max-pool می‌گیرد بنابراین با داشتن ۲۵۶ لایه ویژگی، ۲۵۶ عدد حاصل از max-pooling خواهیم داشت. در سطح بعدی که سبز رنگ است ۴ bin داریم و بنابراین در هر یک از binها عمل max-pooling انجام می‌شود و ۴*۲۵۶ عدد خواهیم داشت. سپس این اعداد را در کنار یکدیگر قرار می‌دهیم و بردار حاصل را به لایه تمام متصل می‌دهیم. این روش اطلاعات مکانی را در جایگاه خودشان نگه می‌دارد و همینطور ما را از یکسان کردن سایز تصاویر بی‌نیاز می‌کند.



Spatial Pyramid Pooling (Credits: Paper)