به نام خدا

یادگیری ژرف
نیم‌سال دوم ۰۳ – ۰۲
مدرس: دکتر مهدیه سلیمانی

سید امیر کسائی- ۴۰۲۲۱۲۲۱۴- همفکری با: امیر محمد عزتی

سوال اول)

(a

اثبات کنید:

$$\mathbb{E}_D[P]_{ij} = \mathbb{E}_D\left[(D\odot X)_{ij}\right] = X_{ij}\mathbb{E}_D\left[D_{ij}\right] = pX_{ij}$$

$$\mathbb{E}_D[(P^TP)]_{ij} = \begin{cases} \sum_{k=1}^{N} \mathbb{E}_D\left[D_{ki}D_{kj}X_{ki}X_{kj}\right] = \sum_{k=1}^{N} \mathbb{E}_D\left[D_{ki}\right]\mathbb{E}_D[D_{kj}]X_{ki}X_{kj}\big] = p^2(X^TX)_{ij} & if\ i \neq j \\ \sum_{k=1}^{N} \mathbb{E}_D[D_{ki}^2 X_{ki}X_{kj}] = \sum_{k=1}^{N} \mathbb{E}_D[D_{ki}^2] X_{ki}X_{kj} = p(X^TX)_{ij} & if\ i = j \end{cases}$$

حل:

$$P = D\odot X$$

$$\longrightarrow \left|\left|y - Pw\right|\right|_2^2 = y^Ty - 2w^TP^Ty + w^TP^TPw$$

$$\mathbb{E}_{D\sim Bernoulli(p)}[\left|\left|y - D\odot X\right|\right|_2^2] = \mathbb{E}_D[y^Ty - 2w^TP^Ty + w^TP^TPw]$$

مقدار امید ریاضی یک ماتریس، ماتریس مقادیر امید ریاضی عناصر آن است پس:

$$\mathbb{E}_D[P]_{ij} = \mathbb{E}_D\left[(D\odot X)_{ij}\right] = X_{ij}\mathbb{E}_D\left[D_{ij}\right] = pX_{ij}$$

در ادامه داریم:

$$\mathbb{E}_D[2w^TP^Ty] = 2pw^TX^Ty$$

$$(\mathbb{E}_D[(P^TP)])_{ij} = \sum_{k=1}^{N} \mathbb{E}_D\left[D_{ki}D_{kj}X_{ki}X_{kj}\right]$$

$$\mathbb{E}_D[(P^TP)]_{ij} = \begin{cases} \sum_{k=1}^{N} \mathbb{E}_D\left[D_{ki}D_{kj}X_{ki}X_{kj}\right] = \sum_{k=1}^{N} \mathbb{E}_D\left[D_{ki}\right]\mathbb{E}_D[D_{kj}]X_{ki}X_{kj}\big] = p^2(X^TX)_{ij} & if\ i \neq j \\ \sum_{k=1}^{N} \mathbb{E}_D[D_{ki}^2 X_{ki}X_{kj}] = \sum_{k=1}^{N} \mathbb{E}_D[D_{ki}^2] X_{ki}X_{kj} = p(X^TX)_{ij} & if\ i = j \end{cases}$$

(b

اثبات کنید:

$$\mathcal{L}(\widehat{w}) = \left|\left|y - pX\widehat{w}\right|\right|_2^2 + p(1-p)\left|\left|\hat{\Gamma}\,\widehat{w}\right|\right|_2^2$$

حل:

$$(\mathbb{E}_D[(P^TP)])_{ij} - p^2(X^TX)_{ij} = \begin{cases} 0 & if\ i \neq j \\ (p-p^2)(X^TX)_{ij} & if\ i = j \end{cases}$$

$$\mathcal{L}(w) = \mathbb{E}_D\left[\left|\left|y - D\odot Xw\right|\right|_2^2\right] = \mathbb{E}_D\left[y^Ty - 2w^TP^Ty + w^TP^TPw\right]$$

$$= y^Ty - 2w^TpX^Ty + p^2w^TX^TXw - p^2w^TX^TXw + w^T\mathbb{E}_D[(P^TP)]w$$

$$= \left|\left|y - pXw\right|\right|_2^2 + (w^T\mathbb{E}_D[(P^TP)]w - p^2w^TX^TXw)$$

$$= \left|\left|y - pXw\right|\right|_2^2 + w^T(\mathbb{E}_D[(P^TP)]w - p^2)w$$

$$= \left|\left|y - pXw\right|\right|_2^2 + (p^2 - p)w^T\left(diag(X^TX)\right)w$$

$$= \left|\left|y - pXw\right|\right|_2^2 + p(1-p)w^T\left(diag(X^{TbX})\right)w$$

$$= \left|\left|y - pXw\right|\right|_2^2 + p(1-p)\left|\left|\hat{\Gamma}\,w\right|\right|_2^2$$

(c

فرض می‌کنیم:

$$w = p\widehat{w}$$

و خواهیم داشت:

$$\mathcal{L}(w) = \left|\left|y - pX\widehat{w}\right|\right|_2^2 + p(1-p)\left|\left|\hat{\Gamma}\,\widehat{w}\right|\right|_2^2$$

$$= \left|\left|y - Xw\right|\right|_2^2 + p(1-p)\left|\left|\hat{\Gamma}\frac{w}{p}\right|\right|_2^2$$

$$= \left|\left|y - Xw\right|\right|_2^2 + \left|\left|\sqrt{\frac{1-p}{p}}\,\hat{\Gamma}\,w\right|\right|_2^2 = \left|\left|y - Xw\right|\right|_2^2 + \left|\left|\hat{\Gamma}\,\widehat{w}\right|\right|_2^2$$

ما Γ را برابر $\hat{\Gamma}\sqrt{\frac{1-p}{p}}$ در نظر میگیریم.

(d

برای حل این مساله در نظر میگیریم: $\widetilde{w} = \Gamma\, w$، پس $w = \Gamma^{-1}\widetilde{w} \leftarrow$

$$\left|\left|y - X\Gamma^{-1}\widetilde{w}\right|\right|_2^2 + \left|\left|\widetilde{w}\right|\right|_2^2 \rightarrow$$

ماتریس داده های اصلاح شده: $\tilde{X} = X\Gamma^{-1}$

(e

در نظر می‌گیریم که هر x_j و $\tilde{x}_j$ به ترتیب ستون jام بردار X و $\tilde{X}$ هستند. Γ، یک ماتریس قطری است که عناصر jام قطر آن برابر norm ستون jام X است و $\tilde{X} = cX\Gamma^{-1}$. در اینجا c یک ثابت مثبت است.

$$\tilde{X} = cX\Gamma^{-1} = [\tilde{x}_1\ \tilde{x}_2 \dots \tilde{x}_d] = c[x_1\ x_2 \dots\ x_d] = \begin{bmatrix} \frac{1}{\|x_1\|_2} & 0 & \dots & 0 \\ 0 & \frac{1}{\|x_2\|_2} & \dots & 0 \\ 0 & 0 & \ddots\ \dots & \frac{1}{\|x_d\|_2} \end{bmatrix} = [c\frac{x_1}{\|x_1\|_2}\ \ c\frac{x_2}{\|x_2\|_2}\ \dots\ c\frac{x_d}{\|x_d\|_2}]$$

است که باعث می‌شود بردارهای batch normalization. این شبیه به عملیات استانداردسازی و $\|\tilde{x}_j\|_2 = c$ و $\tilde{x}_j = c\frac{x_j}{\|x_j\|_2}$ در نتیجه ستون واریانس یکسانی داشته باشند.

(f

$$J = \frac{1}{2}\left(y_d - \sum_{k=1}^{n}(\omega_k + \delta_k)x_k\right)^2$$

$$\rightarrow \frac{\partial J}{\partial \omega_i} = -\left(y_d - \sum_{k=1}^{n}(\omega_k + \delta_k)x_k\right).\frac{\partial J}{\partial \omega_i}\left(\sum_{k=1}^{n}(\omega_k + \delta_k)x_k\right)$$

$$\frac{\partial J}{\partial \omega_i}(\omega_k + \delta_k)x_k = \begin{cases} x_i & k = i \\ 0 & O.W \end{cases} \rightarrow \frac{\partial J}{\partial \omega_i} = -\left(y_d - \sum_{k=1}^{n}(\omega_k + \delta_k)x_k\right)x_i$$

$$\delta_k \sim N(0, \alpha \cdot \omega_k^2) \longrightarrow E[\delta_k] = 0 \rightarrow E\left[\frac{\partial J}{\partial \omega_i}\right] = -\left(y_d - \sum_{k=1}^{n}\omega_k x_k\right)x_i$$

(g

در تابع هدف J، عبارتی وجود دارد که خطا بین خروجی پیشبینی شده y_d و خروجی واقعی را نشان می‌دهد که به توان ۲ رسیده تا از خطا های بزرگ تر جلوگیری کند.

$$J' = \left(\frac{1}{2}\right)\left(yd - \sum_{k=1}^{n}\omega_k x_k r_k\right)^2$$

که در اینجا $rk\ N(0, \alpha^2)$، یک متغیر رندوم است که از یک توزیع گاوسی با میانگین صفر و واریانس α^2 برگرفته شده است.

$$E[J'] = \left(\frac{1}{2}\right)E\left[\left(y_d - \sum_{k=1}^{n}\omega_k x_k r_k\right)^2\right] = \left(\frac{1}{2}\right)E\left[y_d^2 - 2y_d\sum_{k=1}^{n}\omega_k x_k r_k + \left(\sum_{k=1}^{n}\omega_k x_k r_k\right)^2\right]$$

$$= \left(\frac{1}{2}\right)(y_d^2 - 2y_d\sum_{k=1}^{n}w_k x_k E[r_k] + \sum_{k=1}^{n}w_k^2 x_k^2 E[r_k^2]) = \frac{1}{2}(y_d^2 + \sum_{k=1}^{n}w_k^2 x_k^2 \alpha^2)$$

از آن جایی که $E[r_k] = 0$ و $E[r_k^2] = \alpha^2$، نشان می‌دهد که این نوع dropout، یک regularization term به شکل $\sum_{k=1}^{n} w_k^2 x_k^2 \alpha^2$ به تابع خطا اضافه می‌کند که مقادیر بالای وزن را penalize می‌کند.

h) هر دو نوع dropout تکنیک هایی هستند که برای regularize کردن شبکه های عصبی استفاده می‌شوند. اما با این وجود آنها در نحوه اعمال dropout و کاربرد باهم تفاوت دارند.

dropout گاوسی جمعی نویز تصادفی را به neuron activation ها در طول آموزش اضافه می‌کند که منجر به قوی تر شدن شبکه و جلوگیری ار overfitting می‌شود. spatial dropout، بر روی feature map های لایه های کانولوشنی با حذف تصادفی feature map ها در طول آموزش عمل می‌کند. این نوع dropout به ویژه برای regularization در CNN ها در بینایی ماشین کاربرد دارد. Spatial dropout، مناطق محلی ((local regions) داده های ورودی را تغییر ناپذیر می‌کند و اطمینان به feature map های مختلف را افزایش می‌دهد که در نتیجه نمایشی robust تر خواهیم داشت.

dropout گاوسی جمعی، معمولا در شبکه های عصبی feedforward استفاده می‌شوند که نویز مستقیما به activation ها اضافه می‌شود.

Spatial dropout به ویژه در CNN هایی که اطلاعات مکانی مهم هستند مانند image classification موثر است.

سوال دوم:

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

$$\frac{\partial L}{\partial W_j} = \sum_i \frac{\partial L}{\partial Z_i} \frac{\partial Z_i}{\partial W_j}$$

$$Z_i = \sum_{j=0}^{K-1} W_j X_{i+j}$$

$$\frac{\partial Z_i}{\partial W_j} = \frac{\partial}{\partial W_j}\left(\sum_{j=0}^{K-1} W_j X_{i+j}\right) = X_{i+j}$$

$$\rightarrow \frac{\partial L}{\partial W_j} = \sum_i \frac{\partial L}{\partial Z_i} X_{i+j} = \frac{\partial L}{\partial Z} * X$$

این رابطه نشان می‌دهد که گرادیان تابع خطا نسبت به وزن فیلتر با استفاده از عمل کانولوشن با ورودی X_i و مشتق جزئی متناظر $\frac{\partial L}{\partial Z_i}$ بدست می‌آید. این مفهوم مشابه فرآیند اعمال یک لایه کانولوشن برای پیدا کردن مقادیر $\frac{\partial L}{\partial W_j}$ است. ورودی X_i و خروجی $\frac{\partial L}{\partial W_j}$ است. بنابراین در عمل، برای پیدا کردن گرادیان های تابع خطا نسبت به وزن های فیلتر، می‌توان از یک عمل کانولوشن استفاده کرد.

سوال سوم:

$$Input: values\ of\ x\ over\ a\ mini-batch: \mathcal{B} = \{x_1 \dots m\};$$
$$Parameters\ to\ be\ learned: \gamma, \beta$$
$$Output: \{y_i = BN_{\gamma,\beta}(x_i)\}$$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad //mini\text{-}batch\ mean$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad //mini-batch\ variance$$

$$\widehat{x_i} \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad //normalize$$

$$y_i \leftarrow \gamma\widehat{x_i} + \beta \equiv BN_{\gamma,\beta}(x_i) \qquad //scale\ and\ shift$$

(a

$$\frac{\partial l}{\partial x_i} = \frac{\partial l}{\partial y_i} \cdot \frac{\partial y_i}{\partial \widehat{x_i}} \cdot \frac{\partial \widehat{x_i}}{\partial x_i} = \frac{\partial l}{\partial y_i} \cdot \gamma \cdot \frac{1}{\sqrt{\sigma_B^2 + \epsilon}}$$

$$\frac{\partial l}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial l}{\partial y_i} \cdot \frac{\partial y_i}{\partial \gamma} \xrightarrow{\gamma \widehat{x_i} + \beta} \frac{\partial l}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial l}{\partial y_i} \cdot \widehat{x_i}$$

$$\frac{\partial l}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial l}{\partial y_i} \cdot \frac{\partial y_i}{\partial \beta} \xrightarrow{\gamma \widehat{x_i} + \beta} \frac{\partial l}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial l}{\partial y_i}$$

(b

همگرایی سریع تر:

با normal سازی ورودی ها در هر mini-batch، BN تغییر متغیر داخلی را کاهش می‌دهد. این پدیده به تغییر در توزیع فعال سازی های شبکه به دلیل تغییر در پارامتر های شبکه در حین آموزش اشاره دارد. با تثبیت توزیع ورودی ها در هر لایه، BN امکان آموزش پایدار تر را فراهم می‌کند که منجر به همگرایی سریع تر می‌شود. بعلاوه، BN وابستگی به مقدار دهی اولیه دقیق پارامتر های مدل را کاهش می‌دهد. از آنجایی که BN ورودی ها را عادی می‌کند، احتمال ناپدید شدن یا انفجار گرادیان ها را کاهش می‌دهد و امکان آموزش قوی تر در معماری های مختلف را فراهم می‌کند.

Regularization:

این نرمال سازی به عنوان شکلی از منظم سازی عمل می‌کند که به جلوگیری از overfitting در شبکه های عصبی عمیق کمک می‌کند. در این روش کمی نویز به مدل اضافه می‌شود و در برخی موارد، ممکن است حتی نیازی به استفاده از روش های dropout یا سایر تکنیک های منظم سازی نباشد.

(c

وقتی پارامترهای تغییر و مقیاس را در نرمال سازی دسته ای (BatchNorm) حذف می کنیم، توانایی مدل را برای انطباق با توزیع های مختلف داده محدود می کند. بدون این پارامترها، شبکه نمی تواند میانگین مقادیر نرمال شده (پارامتر شیفت شده) را تنظیم کند یا گسترش (انحراف استاندارد) مقادیر نرمال شده (پارامتر مقیاس) را کنترل کند. در نتیجه، این مدل ممکن است برای تناسب با توزیع‌های پیچیده داده‌ها مشکل داشته باشد که منجر به عملکرد کمتر بهینه و کاهش دقت می‌شود.

(d

از تکنیک BN در شبکه های عصبی برای بهبود سرعت و stability آموزش استفاده می‌شود. اما زمانی که هر mini batch فقط شامل نمونه هایی از یک کلاس باشد، در اینجا مثلا فقط سگ ها یا فقط گربه ها، می‌تواند منجر به discrepancy بین مراحل train و test شود. به خصوص اگر مجموعه داده نامتعادل باشد. این مشکل به دلیل روشی است که BN، میانگین (μ) و انحراف معیار (σ) را در طول train در مقابل test محاسبه می‌کند. در واقع در حین آموزش، در هر mini batch، BN میانگین و انحراف معیار activation را برای آن دسته کوچک خاص محاسبه می‌کند. سپس از این مقادیر برای نرمال سازی activation ها قبل از ارسال به لایه بعدی استفاده می‌شود. این فرآیند نرمال سازی با نگه داشتن activation ها در محدوده مشابه به speeding و stabilizing مرحله train کمک می‌کند. در حین test، BN از میانگین متحرک میانگین (μ) و انحراف معیار (σ) محاسبه شده در طول train برای نرمال سازی activation ها استفاده می‌کند. این موضوع تضمین می‌کند که شبکه در طول استنتاج به طور consistent در بین mini batch ها یا نمونه های مختلف رفتار می‌کند. حال در حالتی که هر mini batch فقط شامل یک کلاس شود، میانگین و انحراف معیار محاسبه شده در طول آموزش، تنها نشان دهنده آمار آن کلاس خاص است. در نتیجه نرمال سازی انجام شده در طول train، ممکن است به سمت ویژگی های آماری کلاس موجود در mini batch منحرف شود. این موضوع می‌تواند منجر به تخمین های bias شده از میانگین و انحراف معیار برای کل مجموعه داده شود، مخصوصا اگر کلاس ها دارای توزیع های آماری بسیار متفاوت باشند. در طول test، زمانی که میانگین متحرک میانگین (μ) و

انحراف معیار (σ) استفاده می‌شود، شبکه ممکن است با نمونه هایی از کلاس دیگر مواجه شود که ویژگی های آماری متفاوتی دارند، که منجر به discrepancy در activation ها و کاهش عملکرد می‌شود.
در واقع وجود لایه BN همراه با mini batch هایی که تنها شامل یک کلاس در طول آموزش هستند، می‌تواند باعث discrepancy بین مراحل train و test شود. زیرا آمار نرمال سازی که در طول آموزش آموخته می‌شود ممکن است به خوبی به داده های دیده نشده از کلاس های دیگر در طول test تعمیم داده نشود.

سوال چهارم:

a)

حذف بایاس از لایه کانولوشن در یک شبکه CNN، با بلوک های بیان شده می‌تواند بر ظرفیت شبکه در یادگیری الگو های خاص تاثیر بگذارد. بایاس در یک لایه کانولوشن به مدل اجازه می‌دهد تا یک ثابت افزودنی را برای هر feature map یاد بگیرد. این موضوع می‌تواند برای درک الگو های خاص در داده ها بسیار مهم باشد، بخصوص زمانی که داده های ورودی جابجا می‌شوند و یا میانگین غیر صفر دارند. بدون وجود بایاس شبکه ممکن است در مدل سازی چنین الگو هایی دچار مشکل شود. بعلاوه، حذف بایاس ممکن است با اثربخشی نرمال سازی BatchNorm تداخل داشته باشد، زیرا BatchNorm وجود bias را تا حدی در نظر می‌گیرد. لایه های کانولوشن به دلیل استفاده از اشتراک وزن ذاتاً shift invariant هستند. با این حال، بایاس به شبکه این امکان را می‌دهد که ویژگی هایی را که در داده های ورودی حول محور صفر نیستند را یاد بگیرد. حذف بایاس ممکن است توانایی شبکه را برای درک چنین ویژگی هایی به طور موثر محدود کند. در برخی موارد هم حذف بایاس ممکن است انعطاف پذیری شبکه را کاهش دهد تا داده های آموزشی را به خوبی fit کند. این موضوع عملکرد شبکه را به شدت کاهش می‌دهد به خصوص اگر مجموعه داده یا task مورد نظر نیاز به یادگیری روابط یا الگو های پیچیده داشته باشد.

b)

ضرب کردن وزن ها در آلفا:

ضرب وزن ها در یک مقدار اسکالر مانند آلفا به طور موثری فعالیت های شبکه را scale می‌کند. در طول test، اگر آلفا در تمام لایه ها اعمال شود، این مقیاس آنچنان تاثیری بر عملکرد شبکه نمی‌گذارد زیرا شبکه در طول train با مقیاس وزن ها سازگار می‌شود و بزرگی نسبی وزن ها چیزی است که برای پیش‌بینی اهمیت دارد.

ضرب کردن آلفا در تمام درایه های ورودی شبکه:

اگر تمام درایه ورودی شبکه را در آلفا ضرب کنیم، کل فضای ورودی را بصورت یکنواخت scale می‌کند. که می‌تواند تاثیر قابل توجهی بر رفتار شبکه داشته باشد، زیرا بطور موثر توزیع داده های ورودی را تغییر می‌دهد. برای مثال اگر آلفا بزرگتر از ۱ باشد، داده های ورودی را تقویت می‌کند و منجر به activation شدید تر در سراسر شبکه می‌شود. تغییر توزیع شبکه می‌تواند بر میزان تعمیم شبکه به داده های دیده نشده تاثیر بگذارد. این موضوع حتی ممکن است منجر به تغییر در مرز های تصمیم گیری شود و بر توانایی شبکه برای تمایز بین کلاس خا تاثیر بگذارد.

c)

i) طول و عرض feature map را به کمک فرمول های زیر محاسبه می‌کنیم:

$$W = \frac{W_{input} - k}{s} + 1$$
$$H = \frac{H_{input} - k}{s} + 1$$

تعداد وزن های کرنل = k×k×c

تعداد کل محاسبات برای یک موقعیت در feature map خروجی = k×k×c

تعداد کل موقعیت ها در feature map خروجی = W×H

تعداد کل محاسبات برای کل feature map خروجی = (W×H)× (k×k×c)

ii)

$$W = \frac{W_{input} - k}{s} + 1$$
$$H = \frac{H_{input} - k}{s} + 1$$

تعداد کل عناصر در pooling window = k×k×c

تعداد کل موقعیت ها در feature map خروجی = W×H

تعداد کل محاسبات برای کل feature map خروجی = (W×H)× (k×k×c)

تعداد محاسبات برای کل یک خروجی – (W×H)× (k×k×c)×B

iii)

$$W = \frac{W_{input} - k}{s} + 1$$
$$H = \frac{H_{input} - k}{s} + 1$$

تعداد کل عناصر در pooling window = k×k×c

تعداد کل موقعیت ها در feature map خروجی = W×H

تعداد کل محاسبات برای کل feature map خروجی = (W×H)× (k×k×c)

d)

یکی از کاربردهای اصلی کانولوشن های 1×1 کاهش تعداد کانال ها (عمق) در feature map ها است.

با اعمال یک کانولوشن 1×1 با کانال های خروجی کمتر از ورودی، شبکه می تواند کاهش ابعاد را انجام دهد و هزینه محاسباتی لایه های بعدی را کاهش دهد و در عین حال ویژگی های اساسی را حفظ کند. این کاهش ابعاد می تواند به کارآمدتر شدن شبکه از نظر محاسباتی و کاهش خطر overfitting کمک کند، به خصوص در شبکه های عمیق تر با تعداد زیادی پارامتر.

با انباشتن چندین لایه کانولوشنال 1×1 با توابع فعال سازی غیرخطی (به عنوان مثال، ReLU)، شبکه می تواند تبدیل ویژگی های پیچیده را یاد بگیرد و قدرت بازنمایی آن را افزایش دهد.

کانولوشن های 1×1 از نظر محاسباتی کارآمد هستند و در مقایسه با kernel های کانولوشنال بزرگتر به پارامترهای کمتری نیاز دارند. این باعث می شود آنها برای کاهش بار محاسباتی شبکه در طول استنتاج مناسب باشند، به ویژه در سناریوهایی که منابع محاسباتی محدود هستند.

پیچیدگی های 1×1می توانند تعامل بین کانال های مختلف را در نقشه های ویژگی تسهیل کنند.

با اعمال فیلترهایی با اندازه 1×1، شبکه می‌تواند ترکیب خطی ویژگی‌ها را در کانال‌ها محاسبه کند، که امکان نمایش ویژگی‌های غنی‌تر و تمایز بهبود یافته بین کلاس‌ها را فراهم می‌کند.

معرفی کانولوشن های 1×1انعطاف بیشتری در طراحی شبکه فراهم می کند.
آنها را می توان در مراحل مختلف معماری شبکه برای دستکاری ابعاد نقشه های ویژگی و تنظیم ظرفیت شبکه با توجه به نیازهای کار درج کرد.

e)

عملیات pooling ابعاد spatial مربوط به feature map ها را کاهش می دهد و منجر به کاهش تعداد پارامترها و پیچیدگی محاسباتی در لایه های بعدی می شود.
با downsample کردن feature map ها، pooling به استخراج برجسته‌ترین ویژگی‌ها و در عین حال دور انداختن اطلاعات اضافی یا کمتر مهم کمک می‌کند. این می تواند منجر به پردازش کارآمدتر و آموزش سریعتر شود.
pooling شبکه را تشویق می کند تا ویژگی های انتزاعی و تعمیم یافته بیشتری را با جمع آوری اطلاعات از مناطق محلی ورودی بیاموزد.
با خلاصه کردن اطلاعات در هر منطقه pooling، شبکه نسبت به تغییرات کوچک در داده های ورودی کمتر حساس می شود و منجر به بهبود عملکرد تعمیم می شود.
عملیات pooling از نظر محاسباتی کارآمد هستند و تعداد نسبتا کمی از پارامترها را در مقایسه با لایه‌های کانولوشن معرفی می‌کنند.
آنها به کاهش ابعاد spatial نقشه های ویژگی کمک می کنند، که به نوبه خود هزینه محاسباتی لایه های بعدی را کاهش می دهد، به ویژه در معماری شبکه های عصبی عمیق.
همچنین می‌تواند با کاهش وضوح spatial مربوط به feature map ها و اعمال سلسله مراتب spatial در نمایش‌های آموخته شده، به جلوگیری از overfitting کمک کند.
با کنار گذاشتن جزئیات دقیق و تمرکز بر برجسته‌ترین ویژگی‌ها، ادغام می‌تواند توانایی شبکه را برای تعمیم داده‌های دیده نشده بهبود بخشد و خطر به خاطر سپردن نویز در داده‌های آموزشی را کاهش دهد.
این عملیات با کاهش اندازه feature map ها، نیازهای حافظه و ذخیره سازی شبکه را کاهش می دهد.
این امر به ویژه در شرایطی با منابع حافظه محدود، مهم است.

f)

$$RF_0 = 1, RF_n = RF_{n-1} + (K-1)\prod_{i}^{n-1} S_i$$

3×3 convolution with stride 2: $RF_1 = 1 + (3-1) \times 1 = 3$

2×2 pooling: $RF_2 = RF_1 + (2-1) \times 2 = 5$

3×3 convolution with stride 2: $RF_3 = RF_2 + (3-1) \times 2 = 9$

2×2 pooling: $RF_4 = RF_3 + (2-1) \times 4 = 13$

بنابراین receptive field لایه آخر برابر 13*13 خواهد بود.

سوال پنجم:

a)

برای هر لایه کانولوشنال داریم:

$$output_{size} = \frac{input_{size} - kernel_{size} + 2 \times padding}{stride} + 1$$

Network A:

i) $output_{size} = \frac{512-4+2\times1}{2} + 1 = 256$, $output\ dimensions = 256 \times 256 \times 3$
ii) $output\ dimensions = 256 \times 256 \times 64$
iii) $output_{size} = \frac{256-4+2\times1}{2} + 1 = 128$, $output\ dimensions = 128 \times 128 \times 64$
iv) $output\ dimensions = 128 \times 128 \times 128$
v) $output_{size} = \frac{128-4+2\times1}{2} + 1 = 64$, $output\ dimensions = 64 \times 64 \times 128$
vi) $output\ dimensions = 64 \times 64 \times 256$

Network B:

i) $output_{size} = \frac{512-4+2\times1}{2} + 1 = 256$, $output\ dimensions = 256 \times 256 \times 64$
ii) $output_{size} = \frac{256-4+2\times1}{2} + 1 = 128$, $output\ dimensions = 128 \times 128 \times 128$
iii) $output_{size} = \frac{128-4+2\times1}{2} + 1 = 64$, $output\ dimensions = 64 \times 64 \times 256$

(b

Standard convolutional layer:

$$n_{parameters} = k \times k \times C_{in} \times C_{out}$$

Pointwise convolutional layer:

$$n_{parameters} = 1 \times 1 \times C_{in} \times C_{out}$$

Depthwise Separable convolutional layer: including both depthwise and pointwise

$$n_{parameters} = (k \times k \times C_{in}) + (C_{in} \times C_{out})$$

Depthwise convolutional layer:

$$n_{parameters} = k \times k \times C_{in}$$

Because of the fact that Depthwise Separable convolutional includes both depthwise and pointwise, and in the problem statement we do not have output channel for Depthwise Separable convolutional, we consider Depthwise Separable convolutional and the following pointwise conv as a single Depthwise Separable convolutional and consider the first one as a single depthwise conv

Network A:

i) $n_{parameters} = (4 \times 4 \times 3) = 48$
ii) $n_{parameters} = 1 \times 1 \times 3 \times 64 = 192$
iii) $n_{parameters} = (4 \times 4 \times 64) = 1024$
iv) $n_{parameters} = 1 \times 1 \times 64 \times 128 = 8192$
v) $n_{parameters} = (4 \times 4 \times 128) = 2048$
vi) $n_{parameters} = 1 \times 1 \times 128 \times 256 = 32768$

Network B:

i) $n_{parameters} = 4 \times 4 \times 3 \times 64 = 3072$
ii) $n_{parameters} = 4 \times 4 \times 64 \times 128 = 131072$
iii) $n_{parameters} = 4 \times 4 \times 128 \times 256 = 524288$

(c

Standard convolutional layer:

$$n_{Multiplication} = output_{size}^2 \times k^2 \times C_{in} \times C_{out}$$

Pointwise convolutional layer:

$$n_{Multiplication} = output_{size}^2 \times C_{in} \times C_{out}$$

Depthwise convolutional layer:

$$n_{Multiplication} = output_{size}^2 \times k^2 \times C_{in}$$

Network A:

i) $n_{Multiplication} = 256^2 \times 4^2 \times 3 = 3145728$

ii) $n_{Multiplication} = 256^2 \times 3 \times 64 = 12582912$

iii) $n_{Multiplication} = 128^2 \times 4^2 \times 64 = 16777216$

iv) $n_{Multiplication} = 128^2 \times 64 \times 128 = 134217728$

v) $n_{Multiplication} = 64^2 \times 4^2 \times 128 = 8388608$

vi) $n_{Multiplication} = 64^2 \times 128 \times 256 = 134217728$

Total number of multiplication operations = 309,329,920

Network B:

i) $n_{Multiplication} = 256^2 \times 4^2 \times 3 \times 64 = 201326592$

ii) $n_{Multiplication} = 128^2 \times 4^2 \times 64 \times 128 = 2147483648$

iii) $n_{Multiplication} = 64^2 \times 4^2 \times 128 \times 256 = 2147483648$

Total number of multiplication operations = 4,496,293,888

(d

ابعاد feature map برای شبکه A و شبکه B برای هر لایه کانولوشن محاسبه شد. این ابعاد نشان دهنده اندازه spatial (عرض و ارتفاع) و تعداد کانال ها در هر feature map است. شبکه A از کانولوشن های قابل تفکیک عمیق و به دنبال آpointwise convolutions استفاده می‌کند در حالی که شبکه B از کانولوشن های استاندارد استفاده می‌کند. به طور کلی feature map های شبکه A ابعاد فضایی بزرگ تری دارند اما کانال های کمتری نسبت به شبکه B دارند. شبکه A از کانولوشن های depthwise separable استفاده می‌کند که در مقایسه با کانولوشن های استاندارد مورد استفاده در شبکه B، پارامتر های کمتری دارند. با وجود داشتن پارامتر های کمتر در هر عملیات کانولوشن، شبکه A در کل، تعداد پارامتر های بیشتری به دلیل تعداد لایه های بیشتر و استفاده از کانولوشن های pointwise اضافی دارد. شبکه B در مقایسه با شبکه A به عملیات ضرب کمتری نیاز دارد. اگرچه شبکه A پارامتر های کمتری در هر عملیات کانولوشن دارد، استفاده از کانولوشن های depthwise separable و به دنبال آن کانولوشن های pointwise، پیچیدگی محاسباتی را افزایش می‌دهد و در نتیجه تعداد بیشتری عملیات ضرب را به طور کلی انجام می‌دهد.

(e

مدل Mobile Net بر depthwise separable convolutions متکی است که عملیات standard convolution را به دو لایه مجزای depthwise convolutions و pointwise convolutions تجزیه می‌کند. Depthwise convolution یک فیلتر را در هر کانال ورودی اعمال می‌کند در حالی که pointwise convolution یک کانولوشن ۱× 1 را در تمام کانال های ورودی اعمال می‌کند. این جداسازی تعداد پارامتر ها و محاسبات را در مقایسه با کانولوشن های قبلی کاهش می‌دهد و مدل های Mobile Net را کارآمد تر می‌کند. این کاهش محاسبات و پارامتر ها به ما این امکان را می‌دهد که شبکه های عمیق تری داشته باشیم و مدل ویژگی های پیچیده تری را یاد بگیرد.

$$computations = \frac{depthwise\ conv}{standard\ conv} = \frac{depthwise\ conv + pointwise\ conv}{standard\ conv} = \frac{D_k^2 \cdot M \cdot D_F^2 + N \cdot M \cdot D_F^2}{N \cdot D_k^2 \cdot M \cdot D_F^2}$$

$$computations = \frac{1}{N} + \frac{1}{D_k^2}$$

(f

در این مقاله، دو hyper parameter معرفی شده. Width Multiplier و Resolution Multiplier.

Width Multiplier، برای کنترل تعداد کانال های فیلتر های کانولوشن در شبکه استفاده می‌شود. در واقع به عنوان یک scaling factor برای تنظیم تعداد کانال ها در هر لایه از شبکه کاربرد دارد. این هایپرپارامتر به ما این امکان را می‌دهد که در صورت نیاز شبکه‌ای کوچک و سبک داشته باشیم. مثلا به دلیل محدودیت منابع یا هزینه ها.

برای تعداد لایه مشخص، تعداد کانال های تصویر ورودی αM ، تعداد کانال های خروجی αN و $Width\ Multiplier = \alpha$، هزینه محاسباتی شبکه برابر:

$$D_k^2 \cdot \alpha M \cdot D_F^2 + \alpha M \cdot \alpha N \cdot D_F^2$$

و α می‌تواند بین ۰ و ۱ باشد.

هایپرپارامتر Resolution Multiplier مانند Width Multiplier امکان کاهش پارماتر و هزینه محاسباتی را برایمان فراهم می‌کند. با کاهش وضوح ورودی با Resolution Multiplier، پیچیدگی محاسباتی مدل کاهش پیدا می‌کند. این کاهش در spatial dimensions منجر به feature map های کوچک تر و عملیات کمتر در لایه های بعدی می‌شود که در نتیجه شبکه کارآمد تر می‌شود. هزینه محاسباتی شبکه برابر:

$$D_k^2 \cdot \alpha M \cdot \rho^2 D_F^2 + \alpha M \cdot \alpha N \cdot \rho^2 D_F^2$$

سوال ششم:

$$L_c = \frac{1}{2}\sum_{i=1}^{m}\left\|x_i - C_{yi}\right\|^2$$

(a

تابع center loss ویژگی های عمیق کلاس های مشابه را به مرکز متناظر با آن نزدیک می‌کند. این کار سبب میشود که ویژگی ها از لحاظ فضای embedding در یک منطقه قرار بگیرند. اما این موضوع سبب ناپایداری در یادگیری میشود زیرا خود مراکز کلاس ها نیز در هر iteration در حال به روز رسانی هستند و این موضوع در کل داده ها اتفاق نمیفتد بلکه در هر mini batch صورت میگیرد. برای بهبود عملکرد شبکه، این تابع را همراه با softmax برای دسته بندی تصاویر استفاده می‌کنند.

$$L_{combined} = L_{softmax} + \lambda L_{centerloss}$$

(b

$$x_i = Wz_i + b, L_c = \frac{1}{2}\sum_{i=1}^{m}\left\|x_i - C_{yi}\right\|_2^2$$

$$\frac{\partial L_c}{\partial C_j} = \ \Delta C_j = \frac{\sum_{i=1}^{m}\delta(y_i = j)\cdot(C_j - x_i)}{\varepsilon + \sum_{i=1}^{m}\delta(y_i = j)}$$

$$C_j = C_{j-1} - \Delta C_j$$

Updating:

$$\frac{\partial L_c}{\partial x_i} = (x_i - C_{yi})$$

$$\frac{\partial x_i}{\partial Z_i} = W, \frac{\partial x_i}{\partial b} = 1, \frac{\partial x_i}{\partial W} = Z_i^T$$

$$\begin{cases} Z_i = Z_i - \Delta \mathrm{Z_i} = Z_i - (x_i - C_{yi})W \\ W = W - \Delta \mathrm{W} = \mathrm{W} - (x_i - C_{yi})Z_i^T \\ b = b - \Delta \mathrm{b} = \mathrm{b} - (x_i - C_{yi}) \end{cases}$$

(c

در مقاله ارجاع شده، روش جدیدی به نام PEDCC_Loss برای شبکه های CNN معرفی شده است. این رویکرد از تکنیک Center Loss بهتر عمل می کند. PEDCC_Loss از مراکز کلاس توزیع شده یکنواخت از پیش تعریف شده برای بهینه سازی توزیع ویژگی در فضای پنهان استفاده می کند. برخلاف روش‌های مرسوم که پارامترهای لایه طبقه‌بندی را به صورت پویا تنظیم می‌کنند، PEDCC_Loss

از وزن‌های ثابت PEDCC در طول آموزش مدل برای به حداکثر رساندن فاصله بین کلاس‌ها استفاده می‌کند. با ترکیب تابع cross entropy و میانگین مربعات خطا (MSE) بین ویژگی‌های نمونه و مراکز کلاس، این روش سعی می‌کند به توزیعی از ویژگی ها برسد که بیشترین compactness بین نمونه های یک کلاس را داشته باشد و در عین حال تمایز بین نمونه‌های کلاس‌های مختلف را به حداکثر برساند. این استراتژی نه تنها دقت طبقه بندی را بهبود می بخشد، بلکه همگرایی پایدار شبکه را تضمین می کند.