HW3 400101078 Tina Halimi

01 December 2024 23:48

Q1) a) $H \times W \times M$ kernel $k \times k \times M$ تا N

same padding - stride=1 $\Rightarrow$ $N \times (k \times k \times M + 1)$ = تعداد پارامترها (بایاس)

تعداد ضرب‌های یک بار اعمال کرنل $\rightsquigarrow k \times k \times M$ تعداد جمع $\rightsquigarrow k \times k \times M - 1$

تعداد اعمال شدن کرنل $\rightsquigarrow H \times W \times N \Rightarrow$ ضرب کل $= H \times W \times N \times k^2 \times M$

جمع کل $= H \times W \times N \times (k^2 \times M - 1)$

b) input size $\rightsquigarrow 128 \times 128 \times 3$ kernel : 5×5

لایه اول: $64 \times (5 \times 5 \times 3 + 1) = 4864$ ← تعداد پارامتر (تعداد نورون، کرنل، bias)

$H_{out} = \left\lfloor \frac{H_{in} + 2p - k}{s} + 1 \right\rfloor = \left\lfloor \frac{128 + 2 \times 2 - 5}{2} + 1 \right\rfloor = 64$

$\Rightarrow$ ابعاد خروجی $= 64 \times 64 \times 64$ $\Rightarrow$ تعداد ضرب $= 64 \times 64 \times 64 \times 25 \times 3 = 19660800$

تعداد جمع $= 64 \times 64 \times 64 \times (25 \times 3 - 1) = 19398656$

لایه دوم: $128 \times (5 \times 5 \times 64 + 1) = 204928$ ← پارامتر

ابعاد: $\left\lfloor \frac{64 + 4 - 5}{2} + 1 \right\rfloor = 32$ $\Rightarrow$ output size $= 32 \times 32 \times 128$

ضرب $= 32 \times 32 \times 128 \times 5^2 \times 64 = 209715200$

جمع $= 32 \times 32 \times 128 \times (5^2 \times 64 - 1) = 209584128$

لایه سوم: $256 (5 \times 5 \times 128 + 1) = 819456$

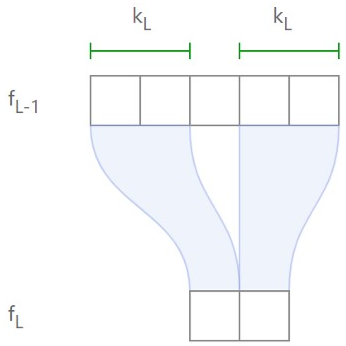
$\left\lfloor \frac{32 + 4 - 5}{2} + 1 \right\rfloor = 16$ $\Rightarrow$ output size $= 16 \times 16 \times 256$

$\Rightarrow$ ضرب $= 16 \times 16 \times 256 \times 5^2 \times 128 = 209715200$

جمع $= 16 \times 16 \times 256 \times (5^2 \times 128 - 1) = 209649664$

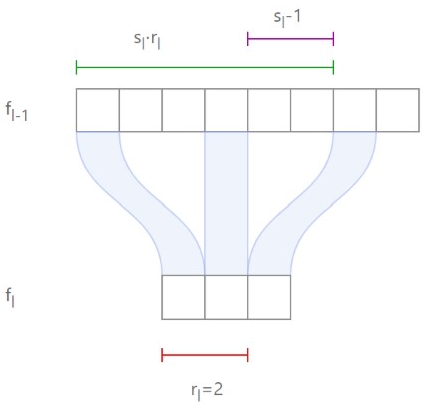
r_l ← تعداد پیکسل‌هایی از لایه l که در شکل گیری یک پیکسل لایه آخر تاثیر دارند ⇒ $r_L = 1$: receptive field

نحوه ایجاد شدن هر پیکسل:



Kernel Size (k_L): 2
Padding (p_L, q_L): 0
Stride (s_L): 3

فرض استقرا: $k_l = 1$

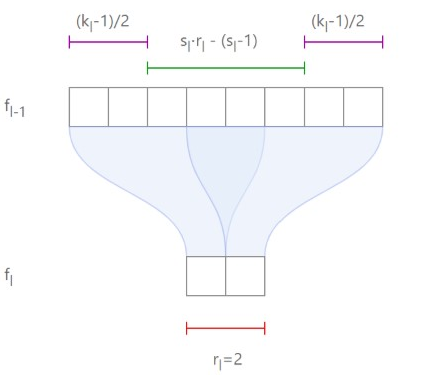


Kernel Size (k_l): 1
Padding (p_l, q_l): 0
Stride (s_l): 3

$r_{l-1} = s_l r_l - (s_l - 1)$

← هر پیکسل از شکل مستطیل است.

$k_l > 1$:



Kernel Size (k_l): 5
Padding (p_l, q_l): 0
Stride (s_l): 3

$r_{l-1} = s_l r_l + (k_l - s_l)$

به صورت بازگشتی ⇒ $r_0 = \sum_{l=1}^{L} (k_l - 1) \prod_{i=1}^{l-1} s_i) + 1$

این سوال ← $L = 3$ برای همه لایه‌ها ← $k = 5$ و $S = 2$

$\Rightarrow r_0 = 4 + 4 \times 2 + 4 \times 2 \times 2 + 1 = 29$

c) 1) Depthwise ⟶ روی هر کانال $k \times k$ و بعد bottleneck میگیریم
$(1\times1\times n)$

input size : $H \times W \times M$

$\Rightarrow M\times(k^2+1)$ ⟶ مرحله اول $(1\times1\times M+1)$ ⟶ مرحله دوم

$\Rightarrow M\times(k^2+1)+M+1 = M(k^2+2)+1$

ضرب → $H\times W\times M\times k^2$ ⟶ پارامترهای کمتری نسبت به a دارد ⟶

2)

لایه دوم : ورودی : $64\times64\times64$ kernel : 5×5

$\Rightarrow$ پارامتر $= 64(25+2)+1 = 1729$

$\left\lfloor \frac{64+4-5}{2}+1 \right\rfloor = 32$ output ⟶ 32×32

$\Rightarrow$ ضرب $= 32\times32\times64\times25 = 1638400$

لایه سوم : ورودی ⟶ 32×32

پارامتر $= 1\times(25+2)+1 = 28$

$\left\lfloor \frac{32+4-5}{2}+1 \right\rfloor = 16$ → output 16×16

ضرب ⟶ $16\times16\times1\times25 = 6400$

پارامترها و عملیات ها به طرز قابل توجهی کمتر است.

c)
تعداد پارامترها کانولوشن عادی $= 1029248$

تعداد پارامترهای Depthwise $= 6621$

d1 کانولوشن عادی ← output size = 16×16×256

Flatten → output size = 65536 200 class output

⇒ پارامترهای FC = 65536 × 200 + 200 = 13107400 (با بایس)

→ total parameter number = 14136648 → %93 پارامترهای FC است

Depthwise → output size = 16×16

⇒ پارامترهای FC = 16×16×200+200 = 51400

کل پارامترها = 51400 + 6621 = 5802 → %88.6 سهم پارامترهای FC

برای کاهش پارامترهای FC، global average pooling، dropout

محدود bottleneck layer، Depthwise separable که در سال استفاده شده

حذف برخی نرون های لایه FC و غیره

Q2)

1) تفاوت ترکیب ویژگی ها: DenseNet خروجی هر لایه با خروجی لایه‌های قبلی concate می‌شود و در ResNet خروجی هر لایه با لایه های قبلی جمع می‌شود

استفاده از concate استفاده مجدد از ویژگی ها را تقویت می‌کند. و جمع جریان گرادیان را ساده می‌کند

2) جریان مستقیم گرادیان: از آنجایی که هر لایه با قبلی ها concate می‌شود، این تضمین وجود دارد که گرادیان به طور مستقیم منتقل می‌شود و احتمال محو شدن آن کمتر است.

استفاده مجدد از ویژگی ها: از خروجی لایه‌های قبلی استفاده می‌شود که کاهش redundancy و به حفظ جریان گرادیان کمک می‌کند. چون مجبور از فیچر ها استفاده می‌کنیم احتیاج به لایه‌های عمیق کمتر شده و gradient vanishing کم می‌شود

3) داده های کم ← با استفاده مجدد از ویژگی ها overfit را در مقایسه با سایر مدل ها کم‌تر می‌کند

وامنیت از ویژگی ها به طور موثرتری بهره می‌برد.

عملکرد بهتر با پارامترهای کمتر دارد ← پس اگر محدودیت computational داشتیم استفاده کنیم، چون دقت خوبی دارد.

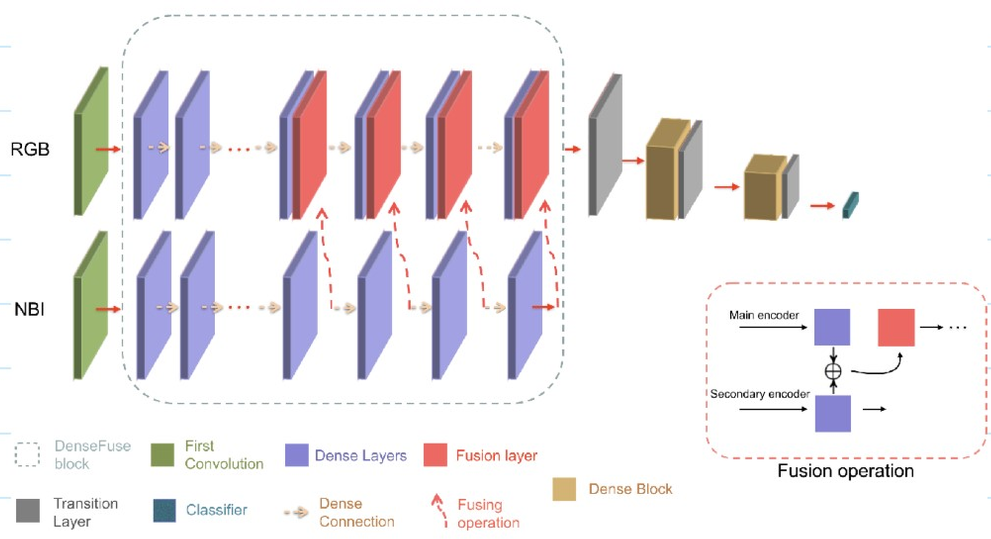
در image classification و object detection استفاده می‌شود.

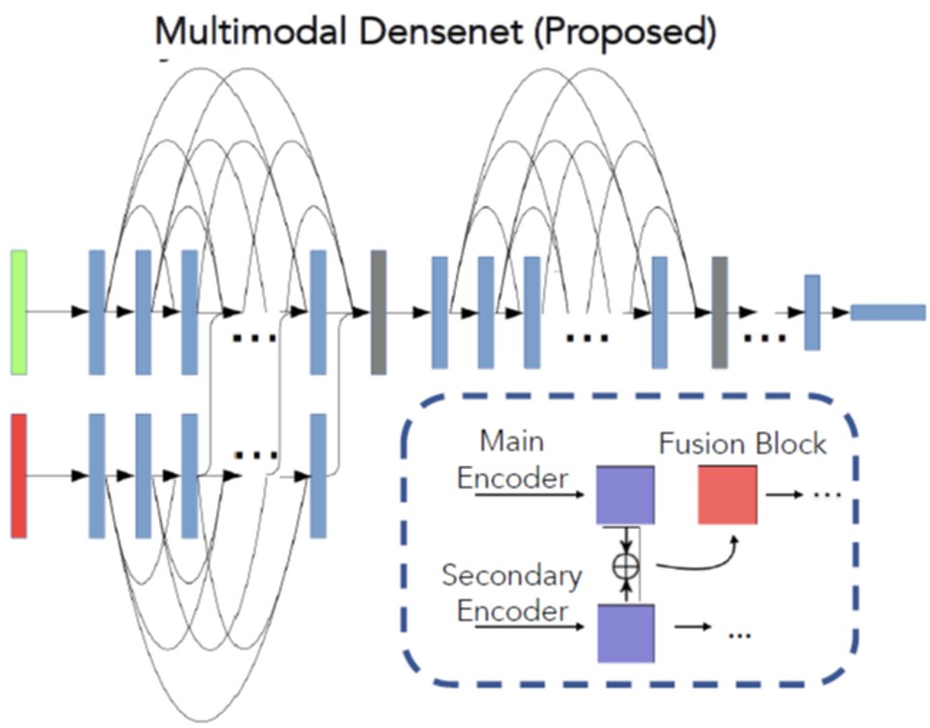
مثال کاربردی: تشخیص ناهنجاری ها از تصاویر پزشکی ← CT و MRI و X-ray

۱۴) برای هر نوع داده branch متفاوتی از DenseNet استفاده می‌کنیم و فیچرها را استخراج می‌کنیم بعد در یک بلاک موسوم به Fusion block به با concate کردن فیچرها آن‌ها را ترکیب کرده و سپس با سایر لایه‌های Dense و بیشتر خطی را انجام می‌دهیم.

Multi modality input → different Densenet branches → Fusion block → Dense layer → output

نمونه:





Q3)

۳) با انتقال برخی متغیرها از encoder مستقیماً به decoder باعث می‌شود متغیرها حفظ شوند (جزئیاتشان) که بهبود decoding و طبقه‌بندی متغیر می‌شود

با ایجاد این اتصالات به اصطلاح میانبر جریان گرادیان بهبود یافته و gradient vanishing با احتمال کمتری رخ می‌دهد. همچنین باعث می‌شود که آموزش موثرتر انجام شده و همگرایی سریع‌تر اتفاق بیفتد. اطلاعات contextual و spatial را ترکیب می‌کند و به این ترتیب با دقت بیشتری segmentation انجام می‌شود.

ب) نوعی data augmentation است.

نحوه اجرا به این شکل است که یک سری تبدیلات الاستیک رخ می‌دهد:

- بردارهای جابه‌جایی کوچک بر grid ای از تصویر اعمال می‌شود. (مثلاً یک شبکه ۳×۳)
این جابه‌جایی‌ها به صورت تصادفی و با توزیع گاوسی با انحراف معیار مشخص است.
- جابجایی‌ها با استفاده از Bicubic interpolation به یک نگاشت پیوسته تبدیل می‌شوند تا هر پیکسل به صورت خاص جابجا شود.
- این تغییرات بر روی تصویر استفاده می‌شود تا تغییرات واقع‌گرایانه/شبیه تقلید یافته از تغییرات طبیعی شکل (کشیدگی، فشرده‌سازی و ... ایجاد شود.

تصادفی بودن تغییرات باعث می‌شود واقعی‌تر به نظر برسند.

تاثیر: کاهش overfit ← با train شدن روی مجموعه بزرگتر و متنوع‌تری از داده‌ها

robust شدن ← حساسیت به نویز و spatial distortion ها کم شده و به تغییرات از این قبیل حساس نخواهد بود

افزایش generalization ← عملکرد بهتر روی دیتاهای unseen

بهبود کارایی با داده‌های کم ← اگر داده‌های کمی داشته باشیم augmentation بسیار ضروری است تا بتوانیم با داده‌های محدود عملکرد خوبی داشته باشیم (مثلاً در تصاویر پزشکی)

افزایش دقت ← با دیدن داده‌های بیشتر و آموزش بهتر به طور کلی عملکرد مدل بهتر می‌شود.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ = input filter = $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ output size = input size + filter size − 1

$= 2 + 2 - 1 = 3$

initial output = $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

برای هر المان ورودی، فیلتر را اعمال می‌کنیم و سپس با مرحله قبلی جمع می‌کنیم.

المان (1,1) ~ filter×1 → output = $\begin{bmatrix} 1 & 2 & 0 \\ 3 & 4 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

المان (1,2) ~ filter×2 → output = $\begin{bmatrix} 1 & 4 & 4 \\ 3 & 10 & 8 \\ 0 & 0 & 0 \end{bmatrix}$

المان (2,1) ~ filter×3 → output = $\begin{bmatrix} 1 & 4 & 4 \\ 6 & 16 & 8 \\ 9 & 12 & 0 \end{bmatrix}$

المان (2,2) ~ filter×4 → output = $\begin{bmatrix} 1 & 4 & 4 \\ 6 & 20 & 16 \\ 9 & 24 & 16 \end{bmatrix}$

سوال ۴، ۲)

YOLO V1 ← تعداد باکس هر cell = 2

تعداد کانال خروجی = $5B + C = 5 \times 2 + 80 = 90$

YOLO V3 ← تعداد باکس هر cell = $3 \times 3 = 9$

۳ اسکیل متفاوت ۳ باکس در هر اسکیل

تعداد کانال خروجی = $3(5 + C) = 3(5 + 80) = 255$

YOLO V3 ← از اسکیل‌های متفاوتی استفاده می‌کند تا object های جفت شده و اشیا با اندازه‌های مختلف بهتر detection شوند. همچنین تعداد بیشتری bounding box دارد که احتمال یافتن یا هم دقیق‌تر را افزایش می‌دهد.

ب) در YOLO V3 از independent logistic classifier استفاده شده که در مراحل آموزش از BCE (binary cross-entropy) به عنوان loss اضافه‌شده که برای داده‌های که دقیقا به یک کلاس تعلق ندارند موثر واقع می‌شود. در YOLO V1 از softmax استفاده می‌شد که در YOLO V3 آن را غیر ضروری باشد)

ج) از الگوریتم non-maximum suppression (NMS) استفاده شده است. در این الگوریتم با توجه به میزان confidence و میزان Intersection over union میان bounding box ها، تکرارها حذف می‌شوند و تنها جعبه با بالاترین confidence (IOU) برای یک شیء آزاد می‌شود.

د) در YOLOv1 ، اندازه ورودی مشخص بود و مدل scale varient بود. در YOLOv2 و YOLOv3

از تکنیک Multi scale training استفاده شده که در آن در فرآیند آموزش هر چند iteration

اندازه تصاویر ورودی به صورت تصادفی بین 320×320 و 608×608 تغییر می‌کند و باعث می‌شود

که مدل دیگر به تغییر ابعاد تصویر ورودی حساس نباشد و روی تصاویر با اندازه‌های مختلف به خوبی عمل کند

و scale invarient باشد.

علت اینکه استفاده از این تکنیک امکان‌پذیر است این است که در YOLOv3 از bounding box

با اسکیل های مختلف برای object detection استفاده می‌کند.

ه) ۱) ابعاد جعبه ها دستی تعیین می‌شود. اگرچه مدل ابعاد جعبه را یاد می‌گیرد اما اگر ما

prior بهتری برای شروع به شبکه بدهیم یادگیری آسان‌تر می‌شود و سریع‌تر همگرا می‌شود

که برای حل این مشکل ابتدا الگوریتم k-means روی داده‌های آموزشی اعمال می‌شود تا

ابعاد بهینه و خوب به دست آورد و همچنین به جای نرم ۲ از IOU استفاده شده است.

۲) ناپایداری مدل مخصوصا در iteration های اول (بیشتر به خاطر یافتن (x,y) جعبه‌ها بود)

راه حل: از تکنیک مکان معلوم آغاز می‌کند و به تدریج می‌آموزد که موقعیت را به روز کند. مختصات

bounding box ها نسبت به مکان Grid cell مشخص می‌شود و از تابع فعالیت

logistic برای قرار دادن پیشبینی ها بین 0 و 1 استفاده می‌شود.

ب - استفاده از darknet-53 که 53 لایه کانولوشنی و ارتباطات residual

است پس gradient بهتری دارد و ویژگی‌ها را بهتر استخراج می‌کند.

flow

- با استفاده از independent logistic regression، می‌توان طبقه‌بندی را برای داده‌هایی

که دقیقا به یک کلاس تعلق ندارند نیز انجام داد. (multi label classification)

- از bounding box با scale های مختلف استفاده می‌کند که در تشخیص

اشیا با ابعاد مختلف بهتر عمل می‌کند.

- در هر scale از تعداد بیشتری bounding box استفاده می‌کند و امتیاز مکان یابی دقیق‌تر

استخراج می‌شود.