سارا حسینی ۴۰۰۲۲۲۰۲۶

سوال۱-

تابع خطا برای fast RCNN:

$$L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u \geq 1]L_{\text{loc}}(t^u, v), \quad (1)$$

$$L_{\text{cls}}(p, u) = -\log p_u$$

در اینجا، عبارت اول نشاندهنده خطای کلسیفیکیشن با کراس انتروپی و عبارت دوم نشاندهنده خطای لوکالیزیشن پنجره مرزی با smooth l1 است.

p=احتمال تعلق شی به کلاس u

u=کلاس تارگت

t=مقادیر پنجره مرزی پیشبینی شده

v=پنجره مرزی حقیقی

تابع smooth l1 در x های کوچک مثل l2 و در سایر نواحی مثل l1 عمل میکند. L2 به اوتلایر بسیار حساس است و به خطاهای کوچک مقاوم است (حساس نیست) پس در x های بزرگ از l1 و در x های کوچک از l2 استفاده میکنیم تا نه به اوتلایر حساس باشد و نه خطاهای کوچک را زیاد پنالتی بدهد. ضریب لامبدا وقتی u=0 است یعنی شیی وجود ندارد و فقط بکگراند داریم، این خطا محاسبه نشود.

$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{x,y,w,h\}} \text{smooth}_{L_1}(t_i^u - v_i), \quad (2)$$

in which

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases} \quad (3)$$

تابع خطای faster RCNN:

چون فستر آر سی ان ان شامل یک شبکه region proposal و یک شبکه فست آر سی ان ان در ادامه میشود،

تابع خطا شامل دو بخش است که هر بخش آن نیز دو عبارت دارد:

بخش اول همان تابع خطای فست آر سی ان ان است که گفتیم دو بخش لوکالیزیشن (از انکر باکس به roi) و کلسیفیکیشن به تعداد کلاس ها میشود.

بخش دوم تابع خطای RPN است که دو عبارت دارد: یکی برای باینری کلسیفیکیشن (آبجکت و بکگراند) و یکی برای رگرشن (پنجره ها).

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*).$$

در این شکل، نشان داده شده که چگونه RPN، یک انکر باکس را به یک پنجره مرزی مپ میکند. در واقع RPN ، یک دلتا برای اینکه از انکر باکس به پنجره مورد نظر برسیم پیدا میکند. (آفست) RPN جایگزینی سریعتر برای سلکتیو سرچ بود که در فست آر سی ان ان استفاده میشد. چون بجای اینکه با یک الگوریتم با پیچیدگی بالا، ROI پیدا کند، فقط کافی است برای انکر باکس یک آفست بیابد و اینگونه کار سریعتر و راحتتر انجام میگیرد.

سوال۲-

الف) پیش فرض LK این است که نقاط در یک پنجره ۳ در ۳ شار نوری یکسانی دارند.

پیش فرض HS این است که هر گونه تغییر روشنایی در یک x و y یکسان، بدلیل حرکت آبجکت است.

هردو فرض میکنند که روشنایی پیکسل های یک آبجکت در دو فریم تغییر نمیکند و همچنین فرض میکند تغییرات کوچک هستند.

در الگوریتم HS بطور لوکال بررسی نمیکند و همزمان برای کل تصویر، شار بدست می‌آید. با فرض پایداری روشنایی، یک دستگاه معادلات خطی حل میشود که درآن فرض کرده‌ایم شار در کل تصویر بطور هموار تغییر

کرده بجز در مرزهای آبجکت. چون بطور لوکال نگاه نمیکند و کل عکس را بطور همزمان پردازش میکند، به نویز حساس تر است.

ب) یک وزن برای میزان اهمیت عبارت smoothness در تابع هدف است. این عبارت smoothness باعث تقویت spatial coherence در میدان میشود و گرادیان های بزرگ را جریمه میکند. در واقع یک ترید آف بین اکیورسی و smoothness بوجود می‌آورد. وقتی لامبدا زیاد میشود، اهمیت همواری بیشتر شده و میدان هموارتر میشود (پیکسلهای همسایه بردارهای جابجایی شبیه تری خواهند داشت) اما این یعنی ممکن است جزئیات از دست برود.

لامبدا وقتی کوچک باشد، میدان آزادی بیشتری دارد و اکیوریت تر میشود و تغییرات کوچک و لوکال را میفهمد اما همجنین باعث نویز و عدم یکدستی میدان میشود.

ج) اگر از دریچه کوچکی به تصویر نگاه کنیم، مثلا

اگر خطوط در هر جهتی حرکت کنند ما باز هم حرکت در جهت جنوب شرقی را فقط درک میکنیم. وقتی لبه ای در عکس حرکت میکند، فقط حرکت در راستای عمود بر لبه بطور خوبی توسط شار درک میشود. به همین علت، صحت الگوریتمهای تخمین حرکت توسط این مسئله تحت تاثیر قرار خواهند گرفت و حرکتهای لبه ها را فقط در راستای عمود بر آن لبه میتوان مشخص کرد.

سوال۳-

الف) فلونت ساده عکس ها را پشت هم استک میکند و به مدل می دهد اما فلونت کوریلیشن، عکس ها را جداگانه از چند لایه عبور داده و سپس لایه کوریلیشن ، کوریلیشن آنان را بدست اورده به ادامه شبکه میدهد. همچنین از چندین ایکیپ کانکشن نیز برای از دست نرفتن اطلاعات استفاده کرده است. فلونت ساده تر و سبک تر است و سریعتر و با دیتای کمتری آموزش میبیند . فلونت کوریلیشن سنگینتر اما اکیورسی بهتری معمولا میگیرد.

ب) اجزای اصلی شبکه FlowNetC2:

1. ساختار انکدر

- وظیفه انکدر استخراج ویژگی‌های مهم از تصاویر ورودی است.

- در FlowNetC2، دو انکدر مجزا برای دو تصویر ورودی وجود دارد که ویژگی‌های هر تصویر را به طور جداگانه استخراج می‌کنند.

- هر انکدر شامل چندین لایه کانولوشن است که به تدریج اندازه ویژگی‌ها را کاهش و تعداد کانال‌ها را افزایش می‌دهد.

2. لایه‌های کوریلیشنی

- این لایه‌ها وظیفه محاسبه شباهت بین ویژگی‌های استخراج شده از دو تصویر را دارند.

- کوریلیشن بین دو تصویر به مدل کمک می‌کند تا جابجایی پیکسل‌ها را بین دو تصویر تشخیص دهد.

- این لایه‌ها با محاسبه دات پردوداکت بین ویژگی‌های دو تصویر در نقاط مختلف، نقشه‌ای از شباهت‌ها را ایجاد می‌کنند.

3. ساختار دیکدر :

- دیکدر وظیفه بازسازی اطلاعات را از ویژگی‌های فشرده شده در انکدر بر عهده دارد.

- دیکدر شامل چندین لایه کانولوشن معکوس (Transposed Convolution) یا لایه‌های Up-sampling است که اندازه ویژگی‌ها را افزایش می‌دهد.

- در نهایت، دیکدر یک نقشه شار نوری با همان اندازه تصویر ورودی تولید می‌کند.

4. Warping Operations:

- این عملیات‌ها وظیفه تطبیق ویژگی‌های تصویر دوم با تصویر اول بر اساس تخمین شار نوری اولیه را دارند.

- با استفاده از نقشه شار نوری اولیه، ویژگی‌های تصویر دوم تغییر مکان داده می‌شوند تا بهتر با تصویر اول هماهنگ شوند.

- این عملیات به مدل کمک می‌کند تا تخمین دقیق‌تری از شار نوری به دست آورد.

5. Fusion و Refinement:

- بعد از انجام عملیات کوریلیشن و واریپینگ، ویژگی‌های تطبیق یافته ترکیب می‌شوند.

- این ترکیب‌ها به دیکدر داده می‌شوند تا تخمین نهایی شار نوری را تولید کنند.

- لایه‌های refinement یا بهبود می‌توانند به اصلاح شار نوری اولیه کمک کنند.

-یکی از چالش‌های اصلی در تخمین شار نوری، تشخیص جابجایی‌های بزرگ بین دو تصویر است. این مسئله زمانی که اشیاء در تصویر به طور سریع حرکت می‌کنند یا دوربین به طور سریع تغییر مکان می‌دهد، به وجود می‌آید. مدل‌ها معمولاً در تشخیص این جابجایی‌ها دچار خطا می‌شوند، زیرا ویژگی‌های مشابه در دو تصویر ممکن است به شدت متفاوت باشند.

-مناطق محدود به مناطقی گفته می‌شود که در یک تصویر دیده می‌شوند ولی در تصویر دیگر به دلیل جابجایی یا وجود اشیاء دیگر دیده نمی‌شوند. این مناطق چالش بزرگی برای مدل‌ها ایجاد می‌کنند، زیرا اطلاعات کافی برای تخمین شار نوری در این مناطق وجود ندارد.مدل‌ها باید قادر باشند تا این مناطق را تشخیص داده و به طور مناسبی مدیریت کنند.

- تغییرات در روشنایی یا رنگ بین دو تصویر می‌تواند باعث خطا در تخمین شار نوری شود.مدل‌ها باید نسبت به این تغییرات مقاوم باشند و ویژگی‌های معنایی بیشتری را استخراج کنند که تحت تأثیر این تغییرات قرار نمی‌گیرند.

- جزئیات کوچک در تصویر و نویز می‌توانند باعث ایجاد خطا در تخمین شار نوری شوند. مدل‌ها باید قادر باشند تا جزئیات مهم را از نویز جدا کرده و تخمین دقیقی از شار نوری ارائه دهند.

سوال۴-

خودتوجهی (Self-Attention) یکی از مفاهیم کلیدی در شبکه‌های ترنسفرمر است که برای مدل‌سازی روابط داخلی و وابستگی‌های بین المان‌های ورودی استفاده می‌شود. در زمینه شبکه‌های ترنسفرمر بینایی (Vision Transformers یا ViTs)، خودتوجهی به این صورت عمل می‌کند که هر پچ (Patch) از تصویر می‌تواند به هر پچ دیگری توجه کند و اهمیت آن را برای درک تصویر کلی بسنجد.

در روش خودتوجهی، سه ماتریس اصلی به کار می‌روند:

1. ماتریس پرسش‌ها (Query)

2. ماتریس کلیدها (Key)

3. ماتریس مقادیر (Value)

این سه ماتریس از ورودی اولیه به دست می‌آیند و سپس برای محاسبه وزن توجه استفاده می‌شوند. این محاسبات به صورت زیر انجام می‌شود:

امتیاز توجه برای هر جفت از المان‌های ورودی با محاسبه حاصل‌ضرب نقطه‌ای (Dot Product) بین پرسش و کلید مربوط به هر المان محاسبه می‌شود. امتیازهای توجه با استفاده از تابع Softmax نرمال‌سازی می‌شوند تا احتمال‌های توجه به دست آیند. وزن‌های توجه حاصل، با ماتریس مقادیر ضرب می‌شوند تا خروجی نهایی توجه محاسبه شود.

در Vision Transformers، تصویر ورودی ابتدا به پچ‌های کوچکتر تقسیم می‌شود و هر پچ به یک (embedding) تبدیل می‌شود. سپس، بردارهای حاصل وارد لایه‌های خودتوجهی می‌شوند تا وابستگی‌ها و روابط بین پچ‌های مختلف تصویر شناسایی و مدل‌سازی شود. این فرایند به مدل کمک می‌کند تا اطلاعات محلی و گلوبال تصویر را همزمان یاد بگیرد.

در لایه امبدینگ=

512*64 + 64

و چون ۳ هد داریم 3*(512*64 + 64)-> 98496

در مولتی هد اتنشن=

(512*512 + 512) تا پارامتر در لایه خطی آخر مولتی هد اتنشن داریم و در کل

98496*8 + (512*512 + 512) = 361152

در هر انکدر =

361152 +2*(512*512 + 512) + 4 = 886468

و ۱۲ انکدر داریم پس

12*886468 = 10637616

مدل به ازای هر پچ یک توکن خروجی میدهد و یک توکن cls نیز خواهیم داشت یعنی ۱۹۷ پچ داریم.

- ViTs: به لطف مکانیزم خودتوجهی، ViTها قادر به مدل‌سازی وابستگی‌های طولانی‌برد بین پچ‌های مختلف تصویر هستند. این به مدل اجازه می‌دهد تا روابط بین نقاط دور از هم در تصویر را بهتر شناسایی کند.

- شبکه‌های CNN: محدود به فیلترهای محلی هستند و اغلب برای مدل‌سازی وابستگی‌های طولانی‌برد نیاز به استفاده از چندین لایه دارند، که ممکن است باعث از دست رفتن اطلاعات شود.

- ViTs: از مکانیزم خودتوجهی استفاده می‌کنند که به آن‌ها اجازه می‌دهد به طور انتخابی بر روی بخش‌های مختلف تصویر تمرکز کنند و اطلاعات مهم را استخراج کنند.

- CNN: وابسته به فیلترهای ثابت و غیر قابل تغییر هستند که نمی‌توانند به طور انتخابی بر روی بخش‌های مختلف تصویر تمرکز کنند.

- ViTs: مکانیزم خودتوجهی نیاز به محاسبات بیشتری دارد و پیچیدگی زمانی آن به طور مربع با افزایش اندازه تصویر افزایش می‌یابد. این ممکن است منجر به نیاز به منابع محاسباتی بیشتر شود.

- CNN: دارای پیچیدگی محاسباتی کمتری هستند و برای پردازش تصاویر بزرگ به منابع کمتری نیاز دارند.

- ViTs: به دلیل تعداد بالای پارامترها، معمولاً نیاز به داده‌های بیشتری برای آموزش دارند تا به عملکرد مناسبی برسند.

- CNN: به داده‌های کمتری برای آموزش نیاز دارند و با داده‌های کمتر می‌توانند عملکرد خوبی داشته باشند.

- ViTs: آموزش آن‌ها به دلیل معماری پیچیده‌تر و نیاز به داده‌های بیشتر، ممکن است سخت‌تر باشد و نیاز به تنظیمات دقیق‌تر و استفاده از استراتژی‌های خاصی مانند یادگیری تدریجی و تقویت داده‌ها داشته باشد.

- CNN: معماری ساده‌تر و ثابت‌تری دارند که آموزش آن‌ها را نسبتاً آسان‌تر می‌کند.

- ViTs: معمولاً نیاز به پیش‌آموزش با استفاده از داده‌های بزرگ دارند تا بتوانند عملکرد خوبی داشته باشند.

- CNN: بسیاری از مدل‌های CNN از قبل آموزش دیده‌اند و به راحتی می‌توان آن‌ها را با داده‌های جدید تطبیق داد.

ترنسفرمرهای بینایی (ViTs) با توجه به قابلیت‌های منحصر به فرد خود در مدل‌سازی وابستگی‌های طولانی‌برد و انعطاف‌پذیری بالا، پتانسیل زیادی برای بهبود عملکرد در پردازش تصاویر دارند. با این حال، محدودیت‌های

محاسباتی و نیاز به داده‌های بیشتر، چالش‌هایی را در استفاده از این مدل‌ها ایجاد می‌کند. شبکه‌های CNN با وجود محدودیت‌های خود، همچنان به دلیل سادگی و inductive bias و کارایی بالا در بسیاری از کاربردهای پردازش تصویر محبوب هستند. انتخاب بین این دو مدل بستگی به نیازها و منابع موجود در پروژه دارد.